密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2021, 8(5): 844–855

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

Camellia 算法 S 盒的紧凑硬件实现*

魏子豪[1,4], 张英杰[1,4], 胡 磊[1,4], 孙思维[2,3], 史丹萍[1,4], 罗宜元[5]

1. 中国科学院 信息工程研究所 信息安全国家重点实验室, 北京 100093
2. 中国科学院大学 密码学院, 北京 100049
3. 密码科学技术国家重点实验室, 北京 100878
4. 中国科学院大学 网络空间安全学院, 北京 100049
5. 惠州学院 计算机科学与工程学院, 惠州 516007

通信作者: 孙思维, E-mail: sunsiwei@ucas.ac.cn

摘 要: Camellia 算法是一种在国际上应用广泛的密码算法, 当需要在资源受限的硬件设备上实现该算法时, 通常会使用塔域实现技术来优化 S 盒的面积. 本文研究 S 盒的紧凑实现, 通过穷搜基于正规基构造的塔域空间下的每一种方案, 并对每种方案使用最新的优化技术, 得到了一个比目前最佳情况面积更小的方案. 对方案的仿真实验证明, 综合结果与理论分析结果一致, 打破了目前紧凑实现的记录.

关键词: Camellia 算法; S 盒; 塔域实现; 乘法逆运算

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000481

中文引用格式: 魏子豪, 张英杰, 胡磊, 孙思维, 史丹萍, 罗宜元. Camellia 算法 S 盒的紧凑硬件实现[J]. 密码学报, 2021, 8(5): 844–855. [DOI: 10.13868/j.cnki.jcr.000481]

英文引用格式: WEI Z H, ZHANG Y J, HU L, SUN S W, SHI D P, LUO Y Y. A compact hardware implementation of S-box for Camellia[J]. Journal of Cryptologic Research, 2021, 8(5): 844–855. [DOI: 10.13868/j.cnki.jcr.000481]

A Compact Hardware Implementation of S-Box for Camellia

WEI Zi-Hao[1,4], ZHANG Ying-Jie[1,4], HU Lei[1,4], SUN Si-Wei[2,3], SHI Dan-Ping[1,4], LUO Yi-Yuan[5]

1. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
2. School of Cryptology, University of Chinese Academy of Sciences, Beijing 100049, China
3. State Key Laboratory of Cryptology, Beijing 100878, China
4. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
5. School of Computer Science and Engineering, Huizhou University, Huizhou 516007, China

Corresponding author: SUN Si-Wei, E-mail: sunsiwei@ucas.ac.cn

* 基金项目: 国家重点研发计划 (2018YFA0704704, 2018YFB0803801); 国家自然科学基金 (61732021, 61772519, 61802400, 62072207); 上海市自然科学基金 (19ZR1420000)

Foundation: National Key Research and Development Program of China (2018YFA0704704, 2018YFB0803801); National Natural Science Foundation of China (61732021, 61772519, 61802400, 62072207); Natural Science Foundation of Shanghai Municipality (19ZR1420000)

收稿日期: 2020-11-13 定稿日期: 2021-01-06

Abstract: Camellia is an internationally and regionally standardized block cipher which has been used worldwide. The tower field implementation is usually the key technique to optimize the area of S-box when the cipher needs to be implemented in a resource-constrained hardware device. In this paper, we conduct an exhaustive search of the tower field representation based on normal basis with several latest optimization ways. As a result, a new implementation scheme is proposed which is better than the existing ones. The experiments show that the experimental data matches the theoretical result well, which is so far the best implementation of Camellia S-box.

Key words: Camellia; S-box; tower field implementation; inverter

1 引言

Camellia 分组密码算法由三菱公司和日本电信电话公司在 2000 年共同设计 [1]. 由于其较高的安全性以及在软、硬件平台上高效的实现性能, 该算法在 2003 年欧洲 NESSIE 项目中被评选为获胜算法, 同年在日本 CRYPTREC 计划中被评选为推荐算法, 2004 年成为 IETF 标准算法, 2005 年成为 ISO/IEC 标准算法, 2006 年成为 PKCS#11 认证密码.

在硬件平台上实现时, 存在这样一种应用场景: 由于环境约束, 电路芯片的面积必须比较小, 因此能使用的电路资源也相对有限, 除了满足正常运行功能之外, 设计者还需要提供安全加解密能力, 例如银行卡芯片、无线传感器节点等. 在这种情况下, 密码算法实现者对资源面积这一参数更敏感. 在 Camellia 算法中, S 盒作为唯一的非线性部件, 需要消耗的资源最多, 可以优化的空间也比线性部件大, 因此通常我们会研究 S 盒的实现方法.

尽管 Camellia 算法与 AES 算法采用了具有同样代数结构的 S 盒, 即有限域 $\mathbb{F}_{2^8}$ 上的求逆映射, 但二者却使用了不同的多项式基. 由线性代数可知, 向量空间中的向量在两个不同基下的系数向量相差一个可逆矩阵, 因此, 若使用 AES 的 S 盒的硬件实现方案, 则 Camellia 的 S 盒应该先将输入向量乘以一个可逆矩阵 $M \in \mathbb{F}_2^{8\times 8}$, 之后用 AES 的 S 盒实现方式完成求逆运算, 再将输出向量乘以 M^{-1}. 这样我们需要考虑 AES 的 S 盒实现之外的两个线性变换的成本. 在本文中, 我们借鉴了很多 AES 的 S 盒的紧凑实现方案 [2–10] 的优化方法, 以塔域实现 [11–13] 为基础, 给出了一个直接实现 Camellia 的 S 盒的实现方案, 其成本小于上述利用 AES 实现的复合方式, 也小于现有的 Camellia 的实现方式 [10,14,15].

本文结构如下: 第 2 节简单介绍塔域实现的基础知识和通用方法; 第 3 节将塔域实现和 Camellia 的具体结构相结合, 根据该密码算法的特点采用了针对性的优化方式; 第 4 节汇总了实现方案的数据, 并通过仿真实验将它与以前的方案对比; 第 5 节是对全文的总结, 以及对未来工作的展望.

2 预备知识

2.1 塔域表现形式

设 $\mathbb{F}_2 = \{0,1\}$, 通过一个代数次数为 8 的不可约多项式 $q(x) \in \mathbb{F}_2[x]$, 可以将 $\mathbb{F}_2$ 扩张为 $\mathbb{F}_{2^8}$, 即 $\mathbb{F}_{2^8} \cong \mathbb{F}_2[x]/(q(x))$. 若 X 为 $q(x)$ 在 $\mathbb{F}_{2^8}$ 上的一个根, 则 $[X^7, X^6, \cdots, X, 1]$ 为 $\mathbb{F}_{2^8}$ 在 $\mathbb{F}_2$ 上的一组多项式基. $\mathbb{F}_{2^8}$ 上的任意元素 b 均可表示为

$$b = \sum_{i=0}^{7} b_i X^i,\ b_i \in \mathbb{F}_2,$$

我们可以用其系数组成的向量 $(b_7, b_6, \cdots, b_0)^{\mathrm{T}}$ 来表示 b.

类似地, $\mathbb{F}_{2^8}$ 可以通过另一种方式从 $\mathbb{F}_2$ 扩张而来. 如图 1 所示, 设 $r(y) \in \mathbb{F}_{2^4}[y]$, $s(z) \in \mathbb{F}_{2^2}[z]$ 和 $t(w) \in \mathbb{F}_2[w]$ 分别是对应域上的不可约多项式, $Y \in \mathbb{F}_{2^8}$, $Z \in \mathbb{F}_{2^4}$ 和 $W \in \mathbb{F}_{2^2}$ 分别是 $r(y)$, $s(z)$ 和 $t(w)$ 的根, 则 $[Y^{16}, Y]$ 为 $\mathbb{F}_{2^8}$ 在 $\mathbb{F}_{2^4}$ 上的一组正规基, $[Z^4, Z]$ 为 $\mathbb{F}_{2^4}$ 在 $\mathbb{F}_{2^2}$ 上的一组正规基, $[W^2, W]$ 为 $\mathbb{F}_{2^2}$

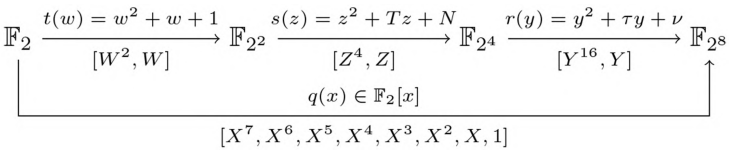


图 1 两种不同的域扩张方式

Figure 1 Two different ways of field extensions

在 $\mathbb{F}_2$ 上的一组正规基. 对任意元素 $b\in\mathbb{F}_{2^8}$ 有

$$\begin{aligned}
&b=\gamma_1Y^{16}+\gamma_0Y,\\
&\gamma_1=\Gamma_3Z^4+\Gamma_2Z,\ \ \gamma_0=\Gamma_1Z^4+\Gamma_0Z,\\
&\Gamma_3=g_7W^2+g_6W,\ \Gamma_2=g_5W^2+g_4W,\ \Gamma_1=g_3W^2+g_2W,\ \Gamma_0=g_1W^2+g_0W,
\end{aligned}$$

其中, $\gamma_1,\gamma_0\in\mathbb{F}_{2^4}$, $\Gamma_3,\Gamma_2,\Gamma_1,\Gamma_0\in\mathbb{F}_{2^2}$, $g_i\in\mathbb{F}_2$, $0\le i\le 7$, 即

$$\begin{aligned}
b=&g_7W^2Z^4Y^{16}+g_6WZ^4Y^{16}+g_5W^2ZY^{16}+g_4WZY^{16}\\
&+g_3W^2Z^4Y+g_2WZ^4Y+g_1W^2ZY+g_0WZY.
\end{aligned}$$

我们将这组由 W, Z 和 Y 组成的基叫做塔域基, 记为 $\mathcal{TB}$,

$$\mathcal{TB}=[W^2Z^4Y^{16},WZ^4Y^{16},W^2ZY^{16},WZY^{16},W^2Z^4Y,WZ^4Y,W^2ZY,WZY],$$

在 $\mathcal{TB}$ 下, b 可由另一组系数向量 $(g_7,g_6,\cdots,g_0)^{\mathrm{T}}$ 来表示.

由于塔域基中的每个基底分量都是 $\mathbb{F}_{2^8}$ 中的元素, 都可以用多项式基来表示,

$$\mathcal{TB}[i]=W^hZ^jY^k=(X^7,X^6,\cdots,X^0)V_i,\ i=7,6,\cdots,0,$$

其中 $\mathcal{TB}[i]$ 表示塔域基的第 i 个分量, V_i 表示它在多项式基下的系数列向量, 因此塔域基整体也可以用多项式基表示为

$$\mathcal{TB}=(X^7,X^6,\cdots,X^0)M_t,M_t=\left[V_7,V_6,\cdots,V_0\right],$$

b 的两组系数向量之间存在如下转换关系

$$\begin{aligned}
&(b_7,b_6,\cdots,b_0)^{\mathrm{T}}=M_t\cdot(g_7,g_6,\cdots,g_0)^{\mathrm{T}},\\
&(g_7,g_6,\cdots,g_0)^{\mathrm{T}}=M_t^{-1}\cdot(b_7,b_6,\cdots,b_0)^{\mathrm{T}}.
\end{aligned}$$

2.2 $\mathbb{F}_{2^8}$ 逆运算的塔域实现

设不可约多项式 $r(y)=y^2+\tau y+\nu\in\mathbb{F}_{2^4}[y]$ 构成了 $\mathbb{F}_{2^4}$ 上的扩张 $\mathbb{F}_{2^8}\cong\mathbb{F}_{2^4}[y]/(r(y))$, 它在 $\mathbb{F}_{2^8}$ 上的一个根 Y 形成了正规基 $[Y^{16},Y]$, 则对任意元素 $g=\gamma_1Y^{16}+\gamma_0Y\in\mathbb{F}_{2^8},\gamma_1,\gamma_0\in\mathbb{F}_{2^4}$, 有

$$\begin{aligned}
g^{-1}&=(gg^{16})^{-1}g^{16}\\
&=[(\gamma_1Y^{16}+\gamma_0Y)(\gamma_0Y^{16}+\gamma_1Y)]^{-1}(\gamma_0Y^{16}+\gamma_1Y)\\
&=[\gamma_1\gamma_0\tau^2+(\gamma_1+\gamma_0)^2\nu]^{-1}\gamma_0Y^{16}+[\gamma_1\gamma_0\tau^2+(\gamma_1+\gamma_0)^2\nu]^{-1}\gamma_1Y.
\end{aligned}\tag{1}$$

因此可以将 $\mathbb{F}_{2^8}$ 上的逆运算转化为 $\mathbb{F}_{2^4}$ 上的乘法运算和逆运算.

类似地, 我们用不可约多项式 $s(z)=z^2+Tz+N\in\mathbb{F}_{2^2}[z]$ 扩张 $\mathbb{F}_{2^2}$, 它的一个根是 $Z\in\mathbb{F}_{2^4}$, 则对

$\mathbb{F}_{2^4}$ 上的任意元素 $\gamma=\Gamma_1 Z^4+\Gamma_0 Z$ 和 $\lambda=\Lambda_1 Z^4+\Lambda_0 Z$, Γ_i, $\Lambda_j\in\mathbb{F}_{2^2}$, 乘法运算表达式为

$$\begin{aligned}\gamma\lambda&=(\Gamma_1 Z^4+\Gamma_0 Z)(\Lambda_1 Z^4+\Lambda_0 Z)\\&=[\Gamma_1\Lambda_1 T+(\Gamma_1+\Gamma_0)(\Lambda_1+\Lambda_0)NT^2]Z^4+[\Gamma_0\Lambda_0 T+(\Gamma_1+\Gamma_0)(\Lambda_1+\Lambda_0)NT^2]Z.\end{aligned}$$

我们用一种新的方法求解逆运算, 具体细节见 3.2 节. 因此 $\mathbb{F}_{2^4}$ 上的运算均可用 $\mathbb{F}_{2^2}$ 上的乘法运算来表示.

最后我们用不可约多项式 $t(w)=w^2+w+1\in\mathbb{F}_2[w]$ 来定义域扩张 $\mathbb{F}_2\subseteq\mathbb{F}_{2^2}$, 它在 $\mathbb{F}_{2^2}$ 上的一个根是 W, 基于正规基 $[W^2,W]$ 可将 $\mathbb{F}_{2^2}$ 上的任意元素 Γ 和 Δ 表示为 $\Gamma=u_1W^2+u_0W$, $\Delta=v_1W^2+v_0W$, 其中 $u_i,v_j\in\mathbb{F}_2$, 乘法运算表达式为

$$\begin{aligned}\Gamma\Delta&=(u_1W^2+u_0W)(v_1W^2+v_0W)\\&=[u_1\cdot v_1\oplus(u_1\oplus u_0)\cdot(v_1\oplus v_0)]W^2+[u_0\cdot v_0\oplus(u_1\oplus u_0)\cdot(v_1\oplus v_0)]W.\end{aligned}$$

2.3 常用电路元件

由于制程工艺的差异, 同一种电路元件的面积在不同工艺库下差距较大, 为了便于比较, 一般我们使用等效门 (GE) 这一单位来衡量面积, 1 GE 表示同一工艺库下一个输入信号数量为 2、驱动能力为 1 的 NAND 门的面积. 除了基础的门电路 (NOT, AND, NAND, OR, NOR, XOR, XNOR) 外, 本文中还会用到两种复合元件 MUX 和 NANDN, 它们的逻辑表达式为 $\mathrm{MUX}(s,a,b)=s\cdot a\oplus s\cdot b\oplus b$, $\mathrm{NANDN}(a,b)=\overline{a\cdot b\oplus b}$. 表 1 展示了在三种工艺库下各种元件的电路面积, N/A 表示该库不支持此种元件.

表 **1** 三种工艺库下常用电路元件的面积

Table 1 Area of frequently-used gates in three CMOS technology libraries

电路元件	面积 (GE)		
	SMIC 130 nm	**SMIC 65 nm**	**Nangate 45 nm**
NOT	0.67	0.75	0.67
AND	1.33	1.5	1.33
NAND	1	1	1
OR	1.33	1.5	1.33
NOR	1	1	1
XOR	2.33	2.25	2
XNOR	2.33	2.25	2
MUX	2.67	2.25	2.33
NANDN	1.33	1.5	N/A

3 实现方案

Camellia 算法一共使用了 4 个不同的 S 盒 s_1,s_2,s_3,s_4, 其中

$$\begin{aligned}s_2(x)&=s_1(x)\lll 1,\\s_3(x)&=s_1(x)\ggg 1,\\s_4(x)&=s_1(x\lll 1),\end{aligned}$$

因此只要实现 s_1, 其余 3 个 S 盒就能通过对 s_1 的输入/输出信号简单移位后直接实现. 在后文中我们用 S 来指代 s_1. 由文献 [1] 可知, S 仿射等价于 $\mathbb{F}_{2^8}$ 上的逆运算, 但其所使用的基并不是像 AES 一样的多

项式基, 如果将其转化为多项式基, 则其表达式为

$$S(b) = M_2 \cdot I^q_{\mathcal{PB}}(M_1 \cdot b \oplus C_1) \oplus C_2,\ b \in \mathbb{F}_{2^8}$$

其中 $I^q_{\mathcal{PB}}$ 为基于不可约多项式 $q(x) = x^8 + x^6 + x^5 + x^3 + 1 \in \mathbb{F}_2[x]$ 和多项式基的逆运算,

$$M_1 = \begin{pmatrix} 0&0&1&0&1&0&0&1\\ 1&0&1&0&1&0&0&0\\ 0&0&0&0&0&0&1&1\\ 0&1&1&0&0&0&0&1\\ 1&1&1&0&1&1&0&0\\ 0&1&0&1&0&1&0&1\\ 0&0&1&1&1&1&1&1\\ 0&0&0&0&0&1&1&0 \end{pmatrix}, C_1 = \begin{pmatrix} 1\\1\\1\\0\\1\\1\\0\\1 \end{pmatrix}, M_2 = \begin{pmatrix} 0&0&0&1&1&1&0&0\\ 0&1&1&1&0&0&0&0\\ 0&1&1&1&0&1&1&0\\ 1&0&0&0&1&1&0&1\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&1&0&1\\ 0&0&1&0&0&1&0&0\\ 0&0&0&1&0&0&0&0 \end{pmatrix}, C_2 = \begin{pmatrix} 0\\1\\1\\0\\1\\1\\1\\0 \end{pmatrix}.$$

由于直接用电路实现 $I^q_{\mathcal{PB}}$ 较为困难, 因此我们考虑用塔域实现逆运算. 设 $I_{\mathcal{TB}}$ 为基于塔域基的逆运算, 由 2.1 节可知, 多项式基和塔域基下的系数向量可以通过线性变换相互转化, 因此

$$I^q_{\mathcal{PB}}(A) = M_t \cdot I_{\mathcal{TB}}(M_t^{-1} \cdot A), \tag{2}$$

图 2 直观地表示了两种基下逆运算之间的联系.

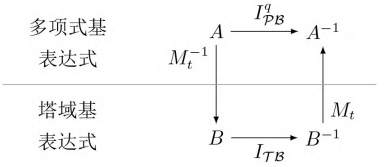


图 2 多项式基下求逆和塔域基下求逆之间的关系

Figure 2 Relationship between inversion under polynomial basis and inversion under tower basis

进而 $S(b)$ 可表示为

$$S(b) = M_2 M_t \cdot I_{\mathcal{TB}}(M_t^{-1} M_1 \cdot b \oplus M_t^{-1} C_1) \oplus C_2.$$

第 2.2 节的运算表达式显示, $I_{\mathcal{TB}}$ 的计算过程受 $r(y), s(z), t(w)$ 三个不可约多项式影响, 参考文献 [10] 可知, 三个多项式共会产生 720 种有效情况. 我们运用文献 [6–10] 中的知识, 对所有有效情况进行了搜索, 得到了一种和文献 [10] 不同, 并且效果更好的实现方案, 其中多项式的系数和它们的根如表 2 所示, 它们的值均是基于多项式基的表示.

表 **2** 三个不可约多项式的系数及其根

Table 2 Values of coefficients and roots of three irreducible polynomials

多项式	系数		根
$t(w)$			$W = 0\text{x}7\text{E}$
$s(z)$	$T = 1 = 0\text{x}01$	$N = W = 0\text{x}7\text{E}$	$Z = 0\text{x}6\text{C}$
$r(y)$	$\tau = N \cdot Z^4 + N \cdot Z = 0\text{x}7\text{E}$	$\nu = N \cdot Z^4 + 0 \cdot Z = 0\text{x}6\text{B}$	$Y = 0\text{x}\text{FC}$

我们使用 5 个模块来完成 S 盒的功能, 如图 3 所示, 每个部分的实现细节见下列各小节.

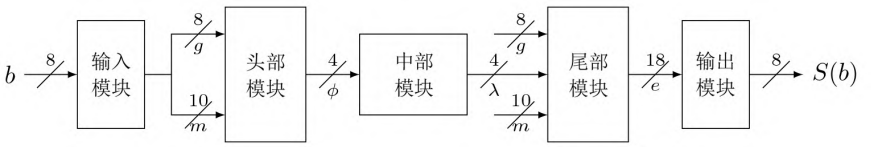


图 3 S 盒整体结构
Figure 3 S-box structure

3.1 头部模块实现方案

头部模块实现了等式(1)中的 $\gamma_1\gamma_0\tau^2+(\gamma_1+\gamma_0)^2\nu$ 部分, 记结果为 ϕ. 用 2.1 节的塔域表示符号将 $\gamma_1\gamma_0\tau^2$ 和 $(\gamma_1+\gamma_0)^2\nu$ 展开到比特级, 可得表达式

$$\begin{aligned}\gamma_1\gamma_0\tau^2 &= (\Gamma_3Z^4+\Gamma_2Z)(\Gamma_1Z^4+\Gamma_0Z)\cdot(N\cdot Z^4+N\cdot Z)^2\\ &= [g_6g_2\oplus(g_7\oplus g_6)(g_3\oplus g_2)\oplus(g_7\oplus g_5)(g_3\oplus g_1)\oplus(g_7\oplus g_6\oplus g_5\oplus g_4)(g_3\oplus g_2\oplus g_1\oplus g_0)]W^2Z^4\\ &\quad+[g_7g_3\oplus g_6g_2\oplus(g_6\oplus g_4)(g_2\oplus g_0)\oplus(g_7\oplus g_6\oplus g_5\oplus g_4)(g_3\oplus g_2\oplus g_1\oplus g_0)]WZ^4\\ &\quad+[g_4g_0\oplus(g_5\oplus g_4)(g_1\oplus g_0)\oplus(g_7\oplus g_5)(g_3\oplus g_1)\oplus(g_7\oplus g_6\oplus g_5\oplus g_4)(g_3\oplus g_2\oplus g_1\oplus g_0)]W^2Z\\ &\quad+[g_5g_1\oplus g_4g_0\oplus(g_6\oplus g_4)(g_2\oplus g_0)\oplus(g_7\oplus g_6\oplus g_5\oplus g_4)(g_3\oplus g_2\oplus g_1\oplus g_0)]WZ\\ (\gamma_1+\gamma_0)^2\nu &= [(\Gamma_3+\Gamma_1)Z^4+(\Gamma_2+\Gamma_0)Z]^2\cdot(N\cdot Z^4+0\cdot Z)\\ &= (g_7\oplus g_6\oplus g_3\oplus g_2)W^2Z^4+(g_6\oplus g_2)WZ^4+(g_7\oplus g_5\oplus g_3\oplus g_1)W^2Z\\ &\quad+(g_7\oplus g_6\oplus g_5\oplus g_4\oplus g_3\oplus g_2\oplus g_1\oplus g_0)WZ.\end{aligned}$$

设

$$\begin{aligned}&m_9=g_7\oplus g_6 \quad m_8=g_3\oplus g_2 \quad m_7=g_7\oplus g_5 \quad m_6=g_3\oplus g_1\\ &m_5=g_6\oplus g_4 \quad m_4=g_2\oplus g_0 \quad m_3=g_7\oplus g_6\oplus g_5\oplus g_4 \quad m_2=g_3\oplus g_2\oplus g_1\oplus g_0\\ &m_1=g_5\oplus g_4 \quad m_0=g_1\oplus g_0,\end{aligned}$$

则

$$\begin{aligned}\phi &= (g_6g_2\oplus m_9m_8\oplus m_7m_6\oplus m_3m_2\oplus m_9\oplus m_8)W^2Z^4\\ &\quad+(g_7g_3\oplus g_6g_2\oplus m_5m_4\oplus m_3m_2\oplus g_6\oplus g_2)WZ^4\\ &\quad+(g_4g_0\oplus m_1m_0\oplus m_7m_6\oplus m_3m_2\oplus m_7\oplus m_6)W^2Z\\ &\quad+(g_5g_1\oplus g_4g_0\oplus m_5m_4\oplus m_3m_2\oplus m_3\oplus m_2)WZ.\end{aligned}$$

运用文献 [6–10] 中的优化方法, 最终表达式为

$$\begin{aligned}\phi &= (\overline{g_6\cdot g_2}\oplus\overline{m_9\mid m_8}\oplus\overline{m_7\cdot m_6}\oplus\overline{m_3\cdot m_2})W^2Z^4\\ &\quad+(\overline{g_7\cdot g_3}\oplus\overline{g_6\mid g_2}\oplus\overline{m_5\cdot m_4}\oplus\overline{m_3\cdot m_2})WZ^4\\ &\quad+(\overline{g_4\cdot g_0}\oplus\overline{m_1\cdot m_0}\oplus\overline{m_7\mid m_6}\oplus\overline{m_3\cdot m_2})W^2Z\\ &\quad+(\overline{g_5\cdot g_1}\oplus\overline{g_4\cdot g_0}\oplus\overline{m_5\cdot m_4}\oplus\overline{m_3\mid m_2})WZ.\end{aligned}$$

在不考虑 $m_9,m_8,\cdots,m_0$ 实现方式 (即假设比特序列 g 和 m 均已知) 的情况下, 实现该表达式需要 8 个 NAND 门、4 个 NOR 门和 12 个 XOR 门.

3.2 中部模块实现方案

中部模块完成了 $\mathbb{F}_{2^4}$ 逆运算的功能, 记 $\lambda=\phi^{-1}$, 表达式为

$$\begin{aligned}\phi &= p_3W^2Z^4+p_2WZ^4+p_1W^2Z+p_0WZ\\ \lambda &= (p_2p_1p_0\oplus p_3p_1\oplus p_2p_1\oplus p_1\oplus p_0)W^2Z^4\\ &\quad+(p_3p_1p_0\oplus p_3p_1\oplus p_2p_1\oplus p_2p_0\oplus p_0)WZ^4\\ &\quad+(p_3p_2p_0\oplus p_3p_1\oplus p_3p_0\oplus p_3\oplus p_2)W^2Z\\ &\quad+(p_3p_2p_1\oplus p_3p_1\oplus p_3p_0\oplus p_2p_0\oplus p_2)WZ.\end{aligned}$$

传统方法利用了逆运算在塔域上的结构特点来实现, 当我们舍弃这一特征, 转而应用文献 [9] 中的方法时, 可以得到一个面积更小的实现方案:

$$\begin{aligned}&t_1=\mathrm{NAND}(p_2,p_0) && t_2=\mathrm{NOR}(p_3,p_1) && t_3=\mathrm{XNOR}(t_1,t_2) && t_4=\mathrm{MUX}(p_3,p_0,1)\\ &t_5=\mathrm{MUX}(p_1,p_2,1) && t_6=\mathrm{MUX}(p_0,t_3,p_1) && t_7=\mathrm{MUX}(t_3,p_1,t_4) && t_8=\mathrm{MUX}(p_2,t_3,p_3)\\ &t_9=\mathrm{MUX}(t_3,p_3,t_5),\end{aligned}$$

$$\lambda=(t_7,t_6,t_9,t_8),$$

共使用 1 个 NAND 门, 1 个 NOR 门, 1 个 XNOR 门和 6 个 MUX 元件.

通过观察可知,

$$\mathrm{MUX}(s,a,1)=s\cdot a\oplus s\oplus 1=\mathrm{NANDN}(a,s),$$

一般情况下, NANDN 元件的面积小于 MUX 元件, 当工艺库支持 NANDN 元件时, 我们可以进一步优化 t_4 和 t_5:

$$t_4=\mathrm{NANDN}(p_0,p_3),\quad t_5=\mathrm{NANDN}(p_2,p_1).$$

3.3 尾部模块实现方案

尾部模块实现了等式(1)中 $\lambda\gamma_0$ 和 $\lambda\gamma_1$ 的部分表达式. 设

$$\lambda=l_3W^2Z^4+l_2WZ^4+l_1W^2Z+l_0WZ,$$

$$k_4=l_3\oplus l_2,\quad k_3=l_3\oplus l_1,\quad k_2=l_2\oplus l_0,\quad k_1=l_3\oplus l_2\oplus l_1\oplus l_0,\quad k_0=l_1\oplus l_0,$$

$$\begin{aligned}&e_{17}=\mathrm{NAND}(g_3,l_3) && e_{16}=\mathrm{NAND}(m_8,k_4) && e_{15}=\mathrm{NAND}(g_2,l_2) && e_{14}=\mathrm{NAND}(m_4,k_2)\\ &e_{13}=\mathrm{NAND}(m_6,k_3) && e_{12}=\mathrm{NAND}(m_2,k_1) && e_{11}=\mathrm{NAND}(g_1,l_1) && e_{10}=\mathrm{NAND}(m_0,k_0)\\ &e_9=\mathrm{NAND}(g_0,l_0) && e_8=\mathrm{NAND}(g_7,l_3) && e_7=\mathrm{NAND}(m_9,k_4) && e_6=\mathrm{NAND}(g_6,l_2)\\ &e_5=\mathrm{NAND}(m_5,k_2) && e_4=\mathrm{NAND}(m_7,k_3) && e_3=\mathrm{NAND}(m_3,k_1) && e_2=\mathrm{NAND}(g_5,l_1)\\ &e_1=\mathrm{NAND}(m_1,k_0) && e_0=\mathrm{NAND}(g_4,l_0),\end{aligned}$$

则

$$\begin{aligned}\lambda\gamma_0 &= (e_{17}\oplus e_{16}\oplus e_{14}\oplus e_{13})W^2Z^4+(e_{15}\oplus e_{16}\oplus e_{12}\oplus e_{13})WZ^4\\ &\quad+(e_{11}\oplus e_{10}\oplus e_{14}\oplus e_{13})W^2Z+(e_9\oplus e_{10}\oplus e_{12}\oplus e_{13})WZ,\\ \lambda\gamma_1 &= (e_8\oplus e_7\oplus e_5\oplus e_4)W^2Z^4+(e_6\oplus e_7\oplus e_3\oplus e_4)WZ^4\\ &\quad+(e_2\oplus e_1\oplus e_5\oplus e_4)W^2Z+(e_0\oplus e_1\oplus e_3\oplus e_4)WZ.\end{aligned}$$

从表达式中可以看到, 最后的结果是从 18 个 NAND 门的输出中分别抽取 4 个异或在一起, 形成的 8 个组合. 参考文献 [7] 中的优化方式, 在尾部模块中, 我们只实现比特序列 k 和 e, 将更进一步的异或操作留到之后的模块中去完成. 这样尾部模块一共使用了 5 个 XOR 门和 18 个 NAND 门.

3.4 输入、输出模块实现方案

由上述各节可知, 比特序列 b 通过仿射变换转换为比特序列 g, 同时比特序列 g 通过线性变换生成比特序列 m, 因此序列 g 和 m 都存在 b 的线性表达式, 输入模块可以看作是由一个矩阵 $M_{\text{in}} \in \mathbb{F}_2^{18\times 8}$ 和一个列向量 $C_{\text{in}} \in \mathbb{F}_2^{18}$ 组成; 同理, 尾部模块的输出序列 e 通过线性变换组合成塔域基的逆运算结果, 再通过仿射变换映射成 S 盒的输出序列 $S(b)$, 因此 $S(b)$ 是 e 的线性组合, 输出模块由一个矩阵 $M_{\text{out}} \in \mathbb{F}_2^{8\times 18}$ 和一个列向量 $C_{\text{out}} \in \mathbb{F}_2^8$ 组成.

在此基础上, 文献 [9] 提出了一种对矩阵的优化方法. 设常数参数 (α, β), $\alpha \in \mathbb{F}_{2^8} - \{0\}$, $\beta \in \{0, 1, \cdots, 7\}$, 在多项式基底下,

$$A^{-1} = \sqrt[2^\beta]{\alpha \cdot (\alpha \cdot A^{2^\beta})^{-1}}, \quad A \in \mathbb{F}_{2^8},$$

在有限域中, 元素与常数的乘法运算以及元素的 2^β 次幂运算均可转化为对该元素的线性变换, 即

$$\begin{aligned} \alpha \cdot A &\iff M_\alpha \cdot A, \\ A^{2^\beta} &\iff M_\beta \cdot A, \\ \sqrt[2^\beta]{A} &\iff M_\beta^{-1} \cdot A, \end{aligned}$$

其中 M_α, M_β 分别表示常数 α 和 2^β 次幂运算对应的矩阵, 因此等式(2)可以重写为

$$\begin{aligned} I^q_{\mathcal{PB}}(A) &= M_\beta^{-1} \cdot M_\alpha \cdot I^q_{\mathcal{PB}}(M_\alpha \cdot M_\beta \cdot A) \\ &= M_\beta^{-1} M_\alpha \cdot M_t \cdot I_{\mathcal{TB}}(M_t^{-1} \cdot M_\alpha M_\beta \cdot A), \end{aligned}$$

优化方法在求逆运算过程产生的额外效果可以从图 4 中看出来.

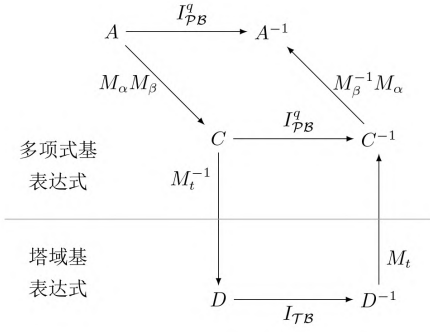


图 4 优化方法对逆运算的影响

Figure 4 Influence of optimization

进而 $S(b)$ 可以对应重写为

$$S(b) = M_2 M_\beta^{-1} M_\alpha M_t \cdot I_{\mathcal{TB}}(M_t^{-1} M_\alpha M_\beta M_1 \cdot b \oplus M_t^{-1} M_\alpha M_\beta C_1) \oplus C_2.$$

这个优化只影响输入、输出模块, 对其他部分没有影响.

实现输入、输出模块的过程本质上是求解 SLP (shortest linear straight-line programs) 问题, 而 SLP

问题已经被证明是 NP-hard 问题 [7]. 目前已经存在多种方法来对其优化, 如文献 [7–9, 16–18], 其中文献 [17] 是将 SLP 问题转换为 SAT 问题, 利用 SAT 求解器来运算, 对维数较小的矩阵能获得可证明的最优解, 而其他文献则都是启发式算法, 只能得到较好的结果. 针对当前规模较大的输入、输出模块, 我们使用了文献 [7] 中的算法.

α 有 255 种取值, β 有 8 种取值, 因此 (α, β) 共有 2040 种组合情况. 在尝试了所有组合对应的矩阵后, 我们发现其中 6 种组合能使输入、输出模块的实现代价之和最小, 其取值和实现代价总和的情况见表3, 其中 α 的值是多项式基下的表达式. 而其他组合则会使用更多的 XOR/XNOR 门电路, 通常会增加 1–16 个.

表 **3** $\boldsymbol{\alpha}$、$\boldsymbol{\beta}$ 的最佳取值和矩阵实现情况

Table 3 Best values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and sum of implementations of their corresponding matrices

$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	电路元件	
		XOR/XNOR	**NOT**
25	7	45	1
31	3		
163	7		
186	7		
206	3		
209	3		

当我们取第一组最佳组合时, 输入模块具体为:

$$M_{\text{in}} = \begin{pmatrix} 0&0&0&1&0&0&1&0 \\ 1&1&0&0&1&0&0&1 \\ 0&1&0&0&1&1&1&0 \\ 0&0&0&1&0&1&0&0 \\ 1&1&0&1&1&0&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&1&1&1&1&0&1&0 \\ 0&1&0&0&0&0&1&0 \\ 1&1&0&1&1&0&1&1 \\ 1&1&0&1&1&0&1&0 \\ 0&1&0&1&1&1&0&0 \\ 1&0&1&0&0&0&1&0 \\ 1&1&0&1&1&1&0&1 \\ 0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&1 \\ 1&1&1&0&0&0&1&0 \\ 0&1&0&1&1&0&1&0 \\ 0&0&1&1&1&0&0&0 \end{pmatrix}, \qquad C_{\text{in}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

一共使用了 17 个 XOR/XNOR 门和 1 个 NOT 门, 实现方案如下:

$$\begin{array}{llll} t_1 = \text{XOR}(b_4, b_1) & t_2 = \text{XNOR}(b_4, b_2) & t_3 = \text{XNOR}(b_6, b_1) & t_4 = \text{XOR}(b_7, b_0) \\ t_5 = \text{XOR}(b_6, b_3 & t_6 = \text{XNOR}(t_4, t_5) & t_7 = \text{XOR}(t_1, t_6) & t_8 = \text{XNOR}(b_0, t_7) \\ t_9 = \text{XOR}(b_1, t_8) & t_{10} = \text{XNOR}(t_2, t_5) & t_{11} = \text{XOR}(t_1, t_{10}) & t_{12} = \text{XOR}(t_4, t_{10}) \\ t_{13} = \text{XOR}(t_4, t_7) & t_{14} = \text{XOR}(b_5, t_{13}) & t_{15} = \text{XOR}(t_9, t_{14}) & t_{16} = \text{XNOR}(b_6, t_{15}) \end{array}$$

$$t_{17} = \mathrm{XOR}(t_8, t_{16}),$$

$$g = (t_1, t_6, t_{11}, t_2, t_9, b_1, t_{14}, t_3)$$
$$m = (t_7, t_8, t_{10}, t_{15}, t_{12}, \mathrm{NOT}(b_6), t_4, t_{16}, t_{13}, t_{17}).$$

输出模块具体为:

$$M_{\mathrm{out}} = \begin{pmatrix} 0&0&0&1&1&0&1&1&0&1&0&1&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&0&1&1 \\ 1&1&0&0&0&0&1&1&0&1&1&0&1&1&0&0&0&0 \\ 1&1&0&0&1&1&1&0&1&0&1&1&1&0&1&1&1&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1 \\ 0&1&1&0&0&0&0&1&1&1&1&0&1&0&1&0&1&1 \\ 0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&0&0&0 \\ 0&0&0&0&1&1&0&1&1&0&1&1&0&1&1&0&0&0 \end{pmatrix}, \quad C_{\mathrm{out}} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix},$$

一共使用了 28 个 XOR/XNOR 门, 实现方案如下:

$$\begin{array}{llll} t_1 = \mathrm{XOR}(e_8, e_5) & t_2 = \mathrm{XOR}(e_6, e_0) & t_3 = \mathrm{XOR}(e_6, e_3) & t_4 = \mathrm{XNOR}(t_1, t_3) \\ t_5 = \mathrm{XOR}(e_1, t_2) & t_6 = \mathrm{XNOR}(e_7, t_5) & t_7 = \mathrm{XOR}(e_4, t_1) & t_8 = \mathrm{XNOR}(t_5, t_7) \\ t_9 = \mathrm{XNOR}(e_{10}, t_6) & t_{10} = \mathrm{XNOR}(e_9, t_4) & t_{11} = \mathrm{XOR}(e_{16}, t_9) & t_{12} = \mathrm{XOR}(t_{10}, t_{11}) \\ t_{13} = \mathrm{XNOR}(e_{15}, t_{12}) & t_{14} = \mathrm{XOR}(e_{17}, e_{11}) & t_{15} = \mathrm{XOR}(e_{13}, e_{12}) & t_{16} = \mathrm{XOR}(t_{12}, t_{15}) \\ t_{17} = \mathrm{XOR}(e_{10}, e_8) & t_{18} = \mathrm{XOR}(e_2, t_2) & t_{19} = \mathrm{XOR}(t_{17}, t_{18}) & t_{20} = \mathrm{XNOR}(e_{16}, t_8) \\ t_{21} = \mathrm{XOR}(t_{16}, t_{20}) & t_{22} = \mathrm{XNOR}(t_8, t_{11}) & t_{23} = \mathrm{XNOR}(t_{14}, t_{22}) & t_{24} = \mathrm{XOR}(t_{14}, t_{16}) \\ t_{25} = \mathrm{XOR}(t_{19}, t_{24}) & t_{26} = \mathrm{XOR}(e_{14}, e_{13}) & t_{27} = \mathrm{XOR}(e_{11}, t_{19}) & t_{28} = \mathrm{XOR}(t_{26}, t_{27}), \end{array}$$

$$S(b) = (t_{28}, t_8, t_{23}, t_{25}, t_6, t_{13}, t_4, t_{21}).$$

4 实验结果

我们将第 3 节各小节描述的数据汇总到表 4 中. 利用 Synopsys Design Compiler 软件, 结合三种工艺库的库文件, 可以对 Camellia 的各种实现方案做综合仿真, 结果见表 5.

表 4 各模块所用电路元件数量

Table 4 Gate number for each module

模块	电路元件					
	NOT	**NAND**	**NOR**	**XOR/XNOR**	**MUX**	**NANDN**
输入模块	1			17		
头部模块		8	4	12		
中部模块		1	1	1	6	
		1	1	1	4	2
尾部模块		18		5		
输出模块				28		

表 **5** 不同论文下的实现方案综合结果

Table 5 Synthesis results of different implementations

文献	电路元件							综合结果		
	NOT	**AND**	**NAND**	**NOR**	**XOR/XNOR**	**MUX**	**NANDN**	**SMIC 130 nm**	**SMIC 65 nm**	**Nangate 45 nm**
[15]	9	25			113			316.33	313.5	278.67
[10]	1		33	8	68			200.33	194.75	177.67
本文	1		27	5	63	6		195.67	188	**172.67**
	1		27	5	63	4	2	**193**	**186.5**	N/A

5 总结

通过划分 AES 密码算法 S 盒轻量化实现时的优化方法, 我们成功地将这些技术运用到 Camellia 中, 在此基础上当实际使用的工艺库允许一些复合元件时, 我们可以做更进一步的优化. 实验结果显示, 运用了新技术的方案比现有的最佳方案减少了从 4.66 GE 到 8.25 GE 不等的面积.

下一步的研究方向是针对深度做优化实现, 以及在同时考虑面积和深度两个维度的情况下, 设计一种比较平衡的方案.

参考文献

[1] AOKI K, ICHIKAWA T, KANDA M, et al. Camellia: A 128-bit block cipher suitable for multiple platforms—Design and analysis[C]. In: Selected Areas in Cryptography—SAC 2000. Springer Berlin Heidelberg, 2012: 39–56. [DOI: 10.1007/3-540-44983-3_4]

[2] SATOH A, MORIOKA S, TAKANO K, et al. A compact Rijndael hardware architecture with S-box optimization[C]. In: Advances in Cryptology—ASIACRYPT 2001. Springer Berlin Heidelberg, 2001: 239–254. [DOI: 10.1007/3-540-45682-1_15]

[3] RUDRA A, DUBEY P K, JUTLA C S, et al. Efficient Rijndael encryption implementation with composite field arithmetic[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2001. Springer Berlin Heidelberg, 2001: 171–184. [DOI: 10.1007/3-540-44709-1_16]

[4] WOLKERSTORFER J, OSWALD E, LAMBERGER M. An ASIC implementation of the AES Sboxes[C]. In: Topics in Cryptology—CT-RSA 2002. Springer Berlin Heidelberg, 2002: 67–78. [DOI: 10.1007/3-540-45760-7_6]

[5] MENTENS N, BATINA L, PRENEEL B, et al. A systematic evaluation of compact hardware implementations for the Rijndael S-box[C]. In: Topics in Cryptology—CT-RSA 2005. Springer Berlin Heidelberg, 2005: 323–333. [DOI: 10.1007/978-3-540-30574-3_22]

[6] Canright D. A very compact S-box for AES[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2005. Springer Berlin Heidelberg, 2005: 441–455. [DOI: 10.1007/11545262_32]

[7] BOYAR J, MATTHEWS P, PERALTA R. Logic minimization techniques with applications to cryptology[J]. Journal of Cryptology, 2013, 26(2): 280–312. [DOI: 10.1007/s00145-012-9124-7]

[8] REYHANI-MASOLEH A, TAHA M M I, ASHMAWY D. Smashing the implementation records of AES S-box[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018, 2018(2): 298–336. [DOI: 10.13154/tches.v2018.i2.298-336]

[9] MAXIMOV A, EKDAHL P. New circuit minimization techniques for smaller and faster AES Sboxes[C]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019, 2019(4): 91–125. [DOI: 10.13154/tches.v2019.i4.91-125]

[10] WEI Z H, SUN S W, HU L, et al. Searching the space of tower field implementations of the $\mathbb{F}_{2^8}$ inverter—with applications to AES, Camellia, and SM4[J]. International Journal of Information and Computer Security.

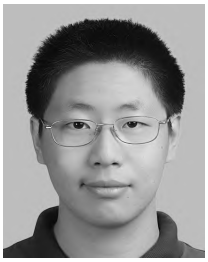
[11] ITOH T, TSUJII S. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases[J]. Information and Computation, 1988, 78(3): 171–177. [DOI: 10.1016/0890-5401(88)90024-7]

[12] GUAJARDO J, PAAR C. Efficient algorithms for elliptic curve cryptosystems[C]. In: Advances in Cryptology—CRYPTO '97. Springer Berlin Heidelberg, 1997: 342–356. [DOI: 10.1007/BFb0052247]

[13] PAAR C, SORIA-RODRIGUEZ P. Fast arithmetic architectures for public-key algorithms over Galois fields GF($(2^n)^m$)[C]. In: Advances in Cryptology—EUROCRYPT '97. Springer Berlin Heidelberg, 1997: 363–378.

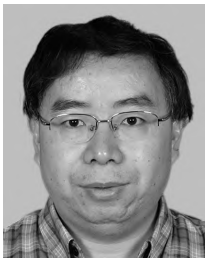
[DOI: 10.1007/3-540-69053-0_25]
[14] SATOH A, MORIOKA S. Unified hardware architecture for 128-bit block ciphers AES and Camellia[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2003. Springer Berlin Heidelberg, 2003: 304–318. [DOI: 10.1007/978-3-540-45238-6_25]
[15] MARTÍNEZ-HERRERA A F, MEX-PERERA J C, NOLAZCO-FLORES J A. Some representations of the S-box of Camellia in GF($((2^2)^2)^2$)[C]. In: Cryptology and Network Security—CANS 2012. Springer Berlin Heidelberg, 2012: 296–309. [10.1007/978-3-642-35404-5_22]
[16] PAAR C, ROSNER M. Comparison of arithmetic architectures for Reed-Solomon decoders in reconfigurable hardware[C]. In: Proceedings of 5th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 1997). IEEE Computer Society, 1997: 219–225. [DOI: 10.1109/FPGA.1997.624622]
[17] FUHS C, SCHNEIDER-KAMP P. Synthesizing shortest linear straight-line programs over GF(2) using SAT[C]. In: Theory and Applications of Satisfiability Testing—SAT 2010. Springer Berlin Heidelberg, 2010: 71–84. [DOI: 10.1007/978-3-642-14186-7_8]
[18] Tan Q Q, Peyrin T. Improved heuristics for short linear programs[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(1): 203–230. [DOI: 10.13154/tches.v2020.i1.203-230]

作者信息



魏子豪 (1993–), 江苏常州人, 博士生在读. 主要研究领域为轻量级密码算法的安全性分析和高效实现.
weizihao@iie.ac.cn

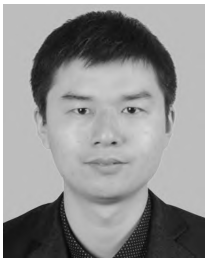
张英杰 (1992–), 河北邯郸人, 博士生在读. 主要研究领域为对称密码算法的安全性分析.
zhangyingjie@iie.ac.cn



胡磊 (1967–), 湖北麻城人, 博士, 研究员, 博士生导师. 主要研究领域为密码学与信息安全.
hulei@iie.ac.cn

孙思维 (1985–), 北京人, 博士, 教授. 主要研究领域为密码分析和密码算法的高效与安全实现.
sunsiwei@ucas.ac.cn

史丹萍 (1989–), 江苏扬州人, 博士, 助理研究员. 主要研究领域为对称密码的设计与分析.
shidanping@iie.ac.cn



罗宜元 (1986–), 河南信阳人, 博士, 副教授. 主要研究领域为对称密码算法的设计与安全性分析.
luoyy@hzu.edu.cn