软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007049]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

SIMON 算法的白盒实现与分析*

罗一诺[1], 陈 杰[1,2], 王 超[1]

[1](西安电子科技大学 通信工程学院, 陕西 西安 710071)

[2](河南省网络密码技术重点实验室, 河南 郑州 450001)

通信作者: 陈杰, E-mail: jchen@mail.xidian.edu.cn

摘 要: 在白盒攻击环境下, 攻击者可以访问密码算法的实现过程, 观测算法运行的动态执行和内部细节, 并任意修改. 2002 年 Chow 等人首次提出了白盒密码的概念, 利用查找表技术提出了 AES 算法和 DES 算法的白盒实现, 所采用的方法称为 CEJO 框架. 白盒实现将已有的密码算法进行编码混淆, 在白盒攻击环境下以软件的形式达到保护密钥的目的, 同时保证算法结果的正确性. SIMON 算法是一种轻量级分组密码算法, 因其良好的软硬件实现性能被广泛应用于物联网设备中, 研究该算法的白盒实现具有重要现实意义. 给出 SIMON 算法的两种白盒实现. 第 1 种方案 (SIMON-CEJO) 采用经典的 CEJO 框架, 利用网络化编码对查找表进行保护, 从而混淆密钥. 该方案占用内存为 369.016 KB, 安全性分析表明 SIMON-CEJO 方案可以抵抗 BGE 攻击和仿射等价算法攻击, 但不能抵抗差分计算分析. 第 2 种方案 (SIMON-Masking) 采用 Battistello 等人提出的编码方式, 对明文信息和密钥信息进行编码, 利用编码的同态性, 将异或运算和与运算转化为模乘运算和表查找操作; 最后进行解码, 得到对应的密文结果. 在算法运行过程中, 对与运算添加布尔掩码, 编码的随机性保护了真实密钥信息, 提高了方案抵抗差分计算分析和其他攻击的能力. SIMON-Masking 占用内存空间为 655.81 KB, 基于勒让德符号的二阶差分计算分析的时间复杂度为 $O(n^2k\log_2 p)$. 这两种方案的对比结果表明, 经典的 CEJO 框架无法有效防御差分计算分析, 运用新型编码并添加掩码是一种有效的白盒实现方法.

关键词: 白盒攻击环境; 白盒实现; SIMON 算法; 掩码; Benaloh 编码

中图法分类号: TP309

中文引用格式: 罗一诺, 陈杰, 王超. SIMON 算法的白盒实现与分析. 软件学报. http://www.jos.org.cn/1000-9825/7049.htm

英文引用格式: Luo YN, Chen J, Wang C. White-box Implementation and Analysis of SIMON. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7049.htm

White-box Implementation and Analysis of SIMON

LUO Yi-Nuo[1], CHEN Jie[1,2], WANG Chao[1]

[1](School of Telecommunications Engineering, Xidian University, Xi'an 710071, China)

[2](Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 100190, China)

Abstract: In the white-box attack context, an attacker can access the implementation process of the cryptographic algorithm, observe the dynamic execution and internal details of the algorithm, and modify it arbitrarily. In 2002, Chow et al. proposed the concept of white-box cipher and pointed out the white-box implementation of the AES algorithm and DES algorithm by using lookup table technology, which is called the CEJO framework. The white-box implementation obfuscates the existing cryptographic algorithms, protects the key in the form of software under white-box attack, and ensures the correctness of the algorithm results. SIMON is a lightweight block cipher algorithm, which is widely used in Internet of Things devices because of its great software and hardware performance. It is of great practical significance to study the white-box implementation of this algorithm. This study presents two white-box implementations of the SIMON

* 基金项目: 陕西省自然科学基础研究计划 (2021JM-126); 河南省网络密码技术重点实验室研究课题 (LNCT2022-A08); 国家自然科学基金 (62302285); 陕西省重点研发计划 (2023-YBGY-015)
收稿时间: 2022-07-22; 修改时间: 2023-04-28, 2023-07-06; 采用时间: 2023-09-03; jos 在线出版时间: 2024-01-17

algorithm. The first scheme (SIMON-CEJO) uses the classical CEJO framework to protect the lookup tables by using network codings, so as to confuse the key. In this scheme, the occupied memory space is 369.016 KB. The security analysis shows that the SIMON-CEJO scheme can resist BGE attack and affine equivalent algorithm attack, but it fails to resist differential computing analysis. The second scheme (SIMON-Masking) uses the encoding method proposed by Battistello et al. to encode the plaintext information and key information, and it uses the homomorphism of encoding to convert the XOR operation and AND operation into modular multiplication and table lookup operation. Finally, the corresponding ciphertext result is obtained by decoding. During the operation of the algorithm, the Boolean mask is added to the AND operation. The randomness of the codings protects the real key information and improves the ability of the scheme to resist differential computing analysis and other attacks. SIMON-Masking occupies 655.81 KB of memory space, and the time complexity of the second-order differential computing based on the Legendre symbol is $O(n^2k\log_2 p)$. The comparison results of the two schemes show that the classical CEJO framework cannot effectively defend against differential computing analysis, but using new coding and adding masks are effective white-box implementation methods.

Key words: white-box attack context; white-box implementation; SIMON algorithm; masking; Benaloh encoding

随着网络与通信技术的迅速发展, 物联网成为新一代信息技术的重要组成部分, 但是各种数据在网络中存储、传输、共享时面临着潜在的安全威胁. 在资源受限的物联网环境下, 轻量级密码算法因其软硬件执行效率高、内存小等优势, 受到越来越多的关注, 用来保护 RFID (radio frequency identifcation) 标签、无线传感器、智能卡等微电子产品. 然而传统的密码算法只考虑了在黑盒模型下的安全性, 即假设算法的运行环境是安全的, 攻击者只能访问算法的输入输出. 随着敌手攻击能力的增强, 终端设备暴露在不可信的环境中, 密码算法的执行过程对于攻击者是完全可见的. 2002 年, Chow 等人[1]将这种攻击环境定义为白盒攻击环境. 它赋予了攻击者更大的能力, 在当今的物联网环境下具有极强的现实意义.

白盒实现是应对白盒攻击环境的密码技术, 它采用软件的方式来保护密钥. 2002 年, Chow 等人采用查找表技术设计了 AES 算法和 DES 算法的白盒实现[1,2], 其主要思想是将密钥嵌入查找表中, 并加入非线性编码和输入输出置乱编码来进行混淆, 算法的执行过程被转化为一系列的查找表和异或操作, 称为 CEJO 框架. 此后, 出现了针对这两种白盒设计方案的多种攻击方法, 如 BGE 攻击[3]、注入错误攻击[4]、MGH 攻击[5]等, 证明了 Chow 等人的白盒 AES 和白盒 DES 是不安全的. 近 20 年来, 基于 CEJO 框架, 学术界针对各种分组密码算法相继提出了相应的白盒实现方案[6−13], 但几乎都被证明是不安全的[14−21].

其中在轻量级分组密码的白盒实现研究方面: 2014 年, Su 等人[22]通过添加干扰项, 提出了 CLEFIA 算法的白盒实现. 随后宫雅婷[23]利用 MGH 攻击, 以 $O(2.5\times2^{29})$ 的时间复杂度攻破了白盒 CLEFIA. 2018 年 Zhou 等人[24]提出了对轻量级密码 KLEIN、PRESENT 以及 LBlock 的白盒实现方案, 但 2021 年姚思[25]利用仿射等价算法, 恢复出了这 3 种方案的密钥. 2021 年, 陈杰等人[26]利用查找表技术, 对 GIFT 算法进行了白盒实现设计. 2022 年, Ranea 等人基于隐式函数和自等价编码的思想, 提出了针对 ARX 结构的分组密码的白盒实现[27], 并设计了 SPECK 算法的一种白盒实现[28]. 设计一种安全高效的轻量级分组密码的白盒实现, 来保证物联网设备之间安全的数据传输, 对于物联网的更广泛应用具有重要的意义.

但是, 上述几种轻量级密码的白盒实现方案在设计时均未考虑差分计算分析 (简称为 DCA 攻击)[21], 这些方案存在被该分析攻破的风险. 对于差分计算分析的解释性研究成果[29−33]表明了正确选择编码对白盒实现的安全性至关重要, 同时文献 [29,31,34,35] 提出了在非线性编码的基础上添加线性掩码的应对方法. 但是这种应对方法部署的代价昂贵, 设计者需要在安全性和效率中做出权衡. 为此, 2021 年 Battistello 等人[36]提出了一种新的编码方式, 利用半同态加密方案, 在 Benaloh 密码系统[37]的基础上, 构造了一种新的编码, 并证明了基于该种编码的白盒实现在抵抗代数攻击和 DCA 攻击方面有一定的安全性, 同时兼顾内存和性能.

SIMON 算法[38]是 2013 年美国国家安全局 (NSA) 发布的一族轻量级分组密码, 在硬件和软件环境中都表现出优异的性能, 适用于物联网环境. SIMON 族算法自从发布以来, 引起了很多密码分析者的关注, 并对它进行了各种密码分析, 比如差分分析[39,40]、不可能差分分析[39,41]、线性分析[42,43]等.

由于网络环境的复杂化, 传统的 SIMON 算法不具备抵抗白盒攻击的能力, 无法保证物联网设备的安全运行. 但目前为止没有其他 SIMON 白盒实现方法的提出. 研究实用性的白盒 SIMON 对于物联网环境下的信息安全有重要的意义. 为了确保 SIMON 算法在白盒攻击环境下的安全性, 同时保证它的轻量性, 本文给出了两种 SIMON

算法的白盒实现方案.

(1) 第 1 种方案采用基于查找表的经典 CEJO 框架的思路, 将密钥嵌入到查找表中, 算法的执行过程被转化为一系列的循环移位、表查找操作、仿射变换和异或操作, 简称为 SIMON-CEJO. 该白盒实现的安全性分析表明: 方案可以抵抗 BGE 攻击、仿射等价算法攻击, 但不能抵抗 DCA 攻击.

(2) 第 2 种方案采用 Battistello 等人[36]提出的编码方式, 对与运算和异或运算进行编码, 利用编码的同态性, 整个加密过程由模乘运算、查找表操作和循环移位来完成; 同时对与运算添加布尔掩码, 来提高方案抵抗 DCA 攻击的安全性. 该方案简称为 SIMON-Masking.

本文第 1 节给出了符号定义和 SIMON 算法描述. 第 2 节介绍本文所需的基础知识, 包括 Benaloh 同态加密方案及它的改进方案. 第 3 节介绍本文设计的 SIMON-CEJO 白盒实现方案. 第 4 节介绍 SIMON-CEJO 的效率及安全性分析. 第 5 节介绍 SIMON-Masking 白盒实现方案. 第 6 节介绍 SIMON-Masking 的效率及安全性分析以及这两种白盒实现方案在效率和安全性方面的对比. 最后总结全文.

1 初步认知

为了便于理解文中的符号, 我们首先给出符号含义的解释; 其次, 给出 SIMON 加密算法的描述.

1.1 符号定义

表 1 给出了文中所用到的符号的解释.

表 1 符号描述

符号	含义
$<<<j$	循环左移 j 位
&	与运算
$\oplus$	异或运算
$gcd(a, b)$	a 与 b 的最大公约数
$a\|b$	a 整除 b
$\equiv$	同余
$\left(\frac{n}{p}\right)$	勒让德符号

1.2 SIMON 的加密过程

SIMON 算法[38]是一种 Feistel 结构的密码算法, 它的轮函数只包括循环移位、与运算和异或运算, 因此具有良好的软硬件实现性能. SIMON 算法根据不同的分组长度和密钥长度给出了 10 个版本, 记为 SIMON2n/mn. 其中分组长度为 $2n$, 密钥长度为 mn, 10 个版本和对应的加密轮数 N_r 见表 2. SIMON 算法的轮函数为 $R_k(x_{i+1}, x_i) = (x_i \oplus f(x_{i+1}) \oplus k_i, x_{i+1})$, 其中 $f(x_{i+1}) = ((x_{i+1} <<< 1) \& (x_{i+1} <<< 8)) \oplus (x_{i+1} <<< 2)$. 轮函数的示意图如图 1 所示.

表 2 SIMON 版本

分组长度 ($2n$)	密钥长度 (mn)	加密轮数
32	64	32
48	72	36
	96	36
64	96	42
	128	44
96	96	52
	144	54
128	128	68
	192	69
	256	72

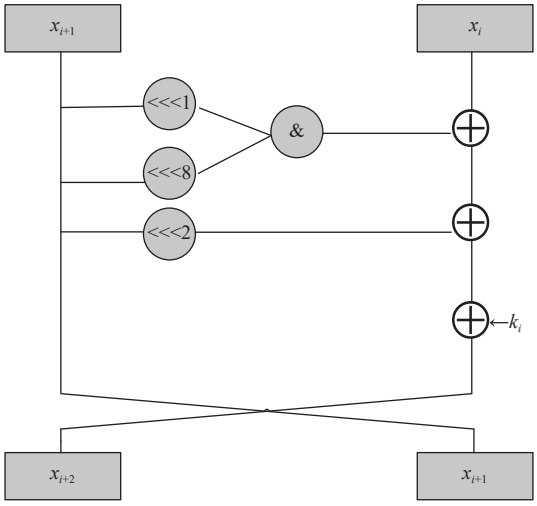

图 1 SIMON 算法轮函数

密钥编排: 每轮所使用的密钥称为轮密钥, 是由主密钥$\{K_0, K_1,\ldots, K_{Nr-1}\}$经过密钥编排而来的, 由密钥编排算法生成 N_r 个子密钥$\{k_0, k_1,\ldots, k_{Nr-1}\}$. 根据 m 的值的不同, 具体过程如下:

当 i=0, 1,…, m–1 时, $k_i=K_i$;

当 $i=m,\ldots, N_r-1$ 时, $k_i=c\oplus(z_j)_{i-m}\oplus k_{i-m}\oplus Y_{i-m}\oplus(Y_{i-m}<<<1)$, 其中 Y_{i-m} 如公式 (1) 定义.

$$Y_{i-m}=\begin{cases} k_{i-m+1}<<<3, & m=2 \\ k_{i-m+1}<<<3, & m=3 \\ k_{i-m+1}<<<3\oplus k_{i-m+1}, & m=4 \end{cases} \tag{1}$$

这里的 $c=2^n-4$ 和 z_j 是和版本相关的常数, 具体描述可参考文献 [38].

2 基础知识

本文所提方法主要基于 Benaloh 同态加密方案[37]及它的改进方案[36], 下面就相关概念和基本知识予以介绍.

2.1 Benaloh 同态加密方案

定义 1. 二次剩余. 设 m>1, 若 $x^2\equiv n(\mathrm{mod}\, m), gcd(n,m)=1$ 有解, 则 n 叫作模数 m 的二次剩余; 若无解, 则 n 叫作模数 m 的二次非剩余.

定义 2. 勒让德符号. 设 p 为奇素数, $gcd(p, n)$=1, 令:

$$\left(\frac{n}{p}\right)=\begin{cases} 1, & \text{若}n\text{是模数}p\text{的二次剩余} \\ -1, & \text{若}n\text{是模数}p\text{的二次非剩余} \end{cases}.$$

函数$\left(\frac{n}{p}\right)$叫作勒让德符号. 由欧拉判别准则可知, $\left(\frac{n}{p}\right)\equiv n^{\frac{p-1}{2}}(\mathrm{mod}\, p)$.

定义 3. 高次剩余. 设 k>1, m>1, 若 $x^k\equiv n(\mathrm{mod}\, m), gcd(n,m)=1$ 有解, 则 n 叫作模数 m 的 k 次剩余; 若无解, 则 n 叫作模数 m 的 k 次非剩余.

定义 4. 高次剩余问题 HRP. 给定整数 n, m, 判断 n 是模数 m 的 k 次剩余还是模数 m 的 k 次非剩余, 称为高次剩余问题.

Benaloh 同态加密方案是 Goldwasser-Micali 同态加密 (GM) 方法的一个扩展. 不同于 GM 方案, 它的安全性依赖于高次剩余问题, 允许一次加密一个比特块.

密钥生成: 选择比特块长度为 r, 两个大素数 p, q, 满足:

(1) $r|(p-1)$;

(2) $gcd(r, (p-1)/r)=1$;

(3) $gcd(r, q-1)=1$.

令 $n=p\times q$, 计算 $\varphi(n)=(p-1)\times(q-1)$. 选取 $y\in Z_n^*$, 使得对于 r 的任意素因子 r_i, 都有 $y^{\varphi(n)/r_i}\not\equiv 1 \mod n$. 公钥为 (n, r, y), 私钥为 (p, q).

加密: 给定公钥 (n, r, y), 对于明文 $m\in Z_r$, 加密函数 E_r 定义为 $E_r(m)=y^m u^r \mod n$, 其中 $u\in Z_n^*$, 是一个随机数.

解密: 给定私钥为 (p, q) 和密文 c, 解密函数 D_r 定义为 $D_r(c)=\log_x(c^{\varphi(n)/r}) \mod n$, 这里 $x=y^{\varphi(n)/r} \mod n$.

Benaloh 同态加密方案具有部分同态性质, 如公式 (2) 所示.

$$\begin{aligned} E_r(a)\times E_r(b) &\equiv y^a u_0^r \times y^b u_1^r \bmod n \\ &\equiv y^{a+b}(u_0u_1)^r \bmod n \\ &\equiv E_r(a+b) \bmod n \end{aligned} \tag{2}$$

2.2 对 Benaloh 同态加密方案的改进

为了便于后续的白盒实现, Battistello 等人[36]对 Benaloh 同态加密方案进行了改进. 改进方案将原方案的两个大素数 p, q 用一个素数 p 代替; 同时为了保证后续白盒实现的正确性, 对块长度 r 作了限制; 引入的随机数 t 保证了加密阶段的随机性. 具体改进如下.

密钥生成:

(1) 选择素数 p;

(2) 选定块长度 $r=2^k$, $k\geqslant 2$. 对于任意的 $s\geqslant 2$, $2^s|(p-1)$, 都有 $k\geqslant s$;

(3) 随机选取 Z_p^* 的生成元 y;

(4) 随机选取 $t\in Z_p^*$.

加密: 给定参数 (t, y) 和 p, 对于明文 $m\in Z_p$, 加密 E_t 定义为 $E_t(m)=ty^m \bmod p$. 可以看到, y 的指数上包含明文信息 m, 随机数 t 用来隐藏明文比特.

解密: 给定参数 (t, y), (p, r) 和密文 c, 解密函数 D_t 定义为 $D_t(c)=\log_x\left((t^{-1}c)^{(p-1)/r}\right) \bmod p$. 这里 $x=y^{(p-1)/r} \bmod p$.

同样地, 改进的方案也满足部分同态, 如公式 (3) 所示. 因此使用这种密码系统作为编码, 允许同态地对编码值执行一些操作, 是白盒实现的一种新思路.

$$\begin{aligned} E_t(m_0)\times E_t(m_1) &\equiv ty^{m_0} \times ty^{m_1} \bmod p \\ &\equiv t^2 y^{m_0+m_1} \bmod p \\ &\equiv E_{t^2}(m_0+m_1) \bmod p \end{aligned} \tag{3}$$

3 SIIMON 算法的 CEJO 白盒实现

在本节中, 采用经典的 CEJO 框架对 SIMON64/128 进行白盒实现. 主要思路是将密钥嵌入到查找表中, 通过输入输出置乱编码混淆密钥. 算法的执行过程由一系列的循环移位、表查找操作、仿射变换和异或操作来完成.

在 SIMON64/128 算法中, $x_i, x_{i+1}\in GF_2^{32}$. 对于第 i 轮, 设计 8 入 32 出的查找表, 如图 2 所示. 其中 $k_i=(k_{i,0}, k_{i,1},\ldots, k_{i,7})$, $f_{i,j}$ 为随机的可逆 4 阶输入置乱编码, $f_i=diag(f_{i,0}, f_{i,1},\ldots, f_{i,7})$, f_{-1} 为恒等映射; $g_{i,j}$ 为 32×4 的输出置乱编码, $g_i=(g_{i,0}, g_{i,1},\ldots, g_{i,7})$ 为 32×32 的可逆仿射变换. 每轮需要 8 张查找表.

假设第 i 轮的输入为 x_i, x_{i+1}, 算法的白盒实现流程如下.

(1) 计算 $x_{i+1,0}=x_{i+1}\lll 1$, $x_{i+1,1}=x_{i+1}\lll 8$, $x_{i+1,2}=x_{i+1}\lll 2$;

(2) 令 $x_{i+1,k}=(x_{i+1,k}^0, x_{i+1,k}^1, \ldots, x_{i+1,k}^8)$, $x_{i+1,k}^j\in GF_2^4$, $j=0, 1,\ldots, 7$, $k=0, 1$. 对每个 4 比特数据分别查找相应的查找表, 将 8 个查找表 I 的输出结果进行异或, 得到 $(x_{i+1}\lll 1)\&(x_{i+1}\lll 8)\oplus k_i$ 的编码值 $Y_{i,1}$;

(3) 利用仿射变换操作, 对 x_i 和 $x_{i+1,2}$ 进行输入输出置乱编码混淆, 分别得到 $Y_{i,0}$ 和 $Y_{i,2}$, 如图 3 所示;

(4) 计算 $Y_i=Y_{i,0}\oplus Y_{i,1}\oplus Y_{i,2}$;

(5) 为了抵消当前轮的输出置乱编码和下一轮的输入置乱编码, 对 Y_i 作用上 $f_{i+1}^{-1}\circ g_i^{-1}$, 得到 x_{i+2}.

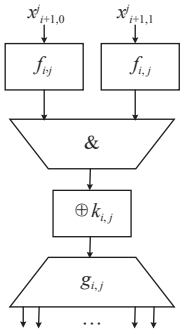

图 2 SIMON-CEJO 的查找表

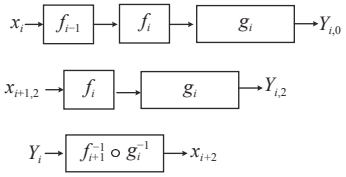

图 3 对 x_i 和 $x_{i+1,2}$ 进行编码; 对 Y_i 进行解码

SIMON-CEJO 前两轮的流程图如图 4 所示. 该白盒实现方案需要添加外部编码, 在首轮和末轮中被抵消, 以保证算法运行结果的正确性. 外部编码部署在安全平台上.

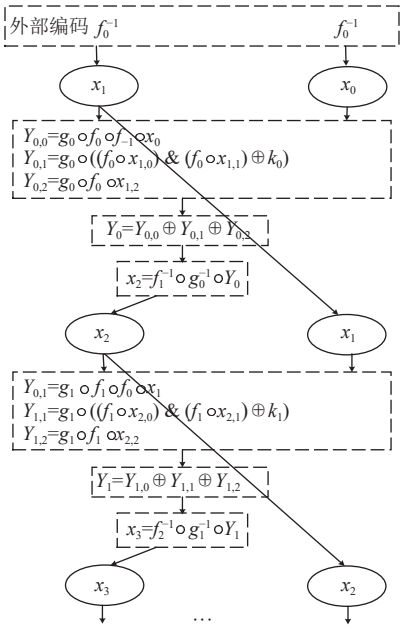

图 4 SIMON-CEJO 前两轮的流程图

4 SIIMON-CEJO 的效率及安全性分析

本节介绍 SIMON-CEJO 白盒实现方案的效率和安全性, 其中方案的安全性主要从白盒多样性/白盒含混度、BGE 攻击、仿射等价算法攻击以及 DCA 攻击的角度来分析.

4.1 效率分析

在 SIMON-CEJO 方案中, 每轮需要 8 张 8 入 32 出的查找表, 2 次 32 比特串异或以及 3 次 (32×32) 仿射变换. 因此内存占用如下.

查找表所需空间: $44\times8\times(2^8\times32)$=352 KB;

仿射变换所需空间为: $44\times3\times(32\times32+32)$=17.016 KB;

综上, SIMON-CEJO 白盒实现的内存占用为 44×(8×(2^8×32)+3×(32×32+32))=369.016 KB.

在该白盒实现方法运行过程中, 共需要 352 次表查找操作, 88 次异或操作和 132 次仿射变换操作.

4.2 白盒多样性和白盒含混度

2002 年 Chow 等人[1]提出了白盒多样性和白盒含混度, 用来评估白盒实现方案的安全性. 白盒多样性是指查找表所有可能的构造方法, 由输入输出编码的种类以及密钥大小来决定. 白盒含混度是指有多少种不同的构造方法能产生相同的查找表, 白盒含混度可以由白盒多样性与实际查找表的数量的比值决定. 白盒含混度在一定程度上体现了攻击者在查找表中获取信息的难易程度, 白盒含混度越高, 查找表越能抵抗穷举攻击.

可逆的 4 阶仿射变换的数量为 20160×2^4≈2^{18}; 嵌在查找表中的密钥长度为 4 比特; 32×4 的仿射变换的数量为 $2^{32\times4}$×2^{32}. 因此在 SIMON-CEJO 白盒实现方案中, 查找表的白盒多样性为 20160×2^4×2^4×$2^{32\times4}$×2^{32}≈2^{182}; 白盒含混度为 2^4×$2^{32\times4}$×2^{32}=2^{164}.

从 SIMON-CEJO 的白盒多样性和白盒含混度来看, 查找表可以抵抗穷举攻击, 攻击者仅从查找表很难得到编码信息和密钥信息.

4.3 BGE 攻击

2004 年, Billet 等人对 Chow 等人的白盒 AES 方案提出了一种代数攻击[3], 简称为 BGE 攻击, 他们以 2^{30} 的时间复杂度攻破了 Chow 等人[1]的白盒 AES 方案. 若方案不能建立起 8 比特到 8 比特的映射关系, 或者只能建立更大的映射, 则可以有效抵抗 BGE 攻击.

该攻击主要包含以下 5 个步骤.

步骤 1. 结合相邻的查找表, 去除非线性编码;

步骤 2. 建立 8 比特到 8 比特的映射关系;

步骤 3. 寻找仿射映射之间的关系;

步骤 4. 恢复仿射变换编码;

步骤 5. 恢复轮密钥.

在 SIMON-CEJO 白盒实现方案中, 对图 2 的查找表建立 8 比特到 32 比特的映射, 输入为 $x_{i+1,0}^j \| x_{i+1,1}^j$, 输出为 $y_{i,j}$, 如公式 (4) 所示, 其中 T 盒的表达式为 $Tbox\left(x_{i+1,0}^j, x_{i+1,1}^j\right) = \left(x_{i+1,0}^j \& x_{i+1,1}^j\right) \oplus k_{i,j}$.

$$y_{i,j} = g_{i,j} \circ Tbox \circ \begin{pmatrix} f_{i,j} & \\ & f_{i,j} \end{pmatrix} \circ \begin{pmatrix} x_{i+1,0}^j \\ x_{i+1,1}^j \end{pmatrix} \tag{4}$$

其中, g_i=($g_{i,0}$, $g_{i,1}$,…, $g_{i,7}$) 为 32×32 的可逆仿射变换, 但它的分块 $g_{i,j}$ 不一定是可逆的, 它不能拆分成可逆的 4×4 或者 8×8 的仿射变换. 因此无法建立 8 比特到 8 比特或 4 比特到 4 比特的映射, 无法寻找映射之间的线性关系, 该方案可以抵抗 BGE 攻击.

4.4 仿射等价算法攻击

定义 5. 仿射等价. 对于 n 比特输入, m 比特输出的 S 盒 S_1 和 S_2, 称 S_1 和 S_2 是仿射等价的, 当且仅当存在可逆的 n 阶仿射变换 A 和 m 阶仿射变换 B 使得 $B \circ S_1 \circ A = S_2$.

寻找 A_1 和 A_2 的问题称为仿射等价问题. 2003 年, Biryukov 等人提出了线性/仿射等价算法[44], 用来求解 S 盒的线性/仿射等价问题. 以线性等价算法为例, 该算法具体描述如下.

(1) 选取两个线性无关的值 x_1 和 x_2, 分别猜测 $A(x_1)$ 和 $A(x_2)$. 查找 S_1 盒下的对应值 $S_1(A(x_1))$ 和 $S_1(A(x_2))$, 分别记为 y_1 和 y_2.

(2) 查找 x_1 和 x_2 在 S_2 盒下的对应值 $B(y_1)$ 和 $B(y_2)$. 令 y_3= $y_1 \oplus y_2$, $B(y_3)$= $B(y_1) \oplus B(y_2)$.

(3) 查找 y_3 和 $B(y_3)$ 在逆 S_1 盒和逆 S_2 盒下的值, 分别记为 x_3 和 $A(x_1)$. 若 x_3 与 x_1, x_2 是线性相关的, 则重新选取 x_1 和 x_2; 若 x_3 与 x_1, x_2 是线性无关的, 则可以得到新的 y_4 和 $B(y_4)$, 检查该点与已知点的相关性. 直到获得 n 对线性无关的 (x, $A(x)$) 和 m 对线性无关的 (y, $B(y)$).

(4) 根据得到的 $(x, A(x))$ 和 $(y, B(y))$, 分别建立关于未知量 A 和 B 的线性方程组, 利用高斯消元法进行求解. 并根据式 $B \circ S_1 \circ A = S_2$ 验证 A 和 B 的正确性.

文献 [44] 证明求解该问题的时间复杂度为 $O(n^3 \cdot 2^n \cdot (2^{n-m}!)^{\frac{n}{2^{n-m}}})$. 对于白盒实现方案, 该方法较 BGE 攻击而言更通用, 攻击能力更强.

在 SIMON-CEJO 方案中, 将 *Tbox* 视作一个 64 比特输入, 32 比特输出的 S_1. 将第 i 轮的 8 个查找表结合起来, 得到公式 (5). 其中 F_i 为 8 个随机的可逆 4 阶输入置乱编码的级联, 如公式 (6) 所示.

$$y_i = g_i \circ Tbox \circ F_i \circ \begin{pmatrix} x_{i+1,0} \\ x_{i+1,1} \end{pmatrix} \tag{5}$$

$$F_i = \begin{pmatrix} f_{i,0} & & & & & & & \\ & f_{i,0} & & & & & & \\ & & f_{i,1} & & & & & \\ & & & f_{i,1} & & & & \\ & & & & \ddots & & & \\ & & & & & f_{i,7} & & \\ & & & & & & f_{i,7} & \end{pmatrix} \tag{6}$$

因此, 利用仿射等价算法求解公式 (5) 的时间复杂度为 $O\left(2^{82} \cdot (2^{32}!)^{2^{-26}}\right)$.

4.5 DCA 攻击

2016 年, Bos 等人[21]提出了差分计算分析 (differential computational analysis, DCA), 它是一种应用于白盒实现安全性分析的侧信道分析方法. 该分析方法可以采集到算法运行过程的内部细节, 得到的软件迹反映了密码算法中间数据的比特. 通过确定选择函数、分类、求均值迹、求差分迹、判定正确密钥等步骤, 得到白盒实现算法的真实密钥. DCA 攻击不考虑外部编码, 并且将分析重点放在查找表的首轮和末轮.

本节利用 DCA 攻击分析 SIMON-CEJO 方案. 已知图 2 查找表的输入输出, 对 4 比特密钥 $k_{i,j}$ 的每一比特位进行猜测, 同时遍历 8 比特输入值, 对输出编码的每种可能情况进行计分, 将最高分定为该猜测密钥比特的得分. 改变密钥比特的猜测值, 得到不同的分数值, 最高分对应的情况即为密钥在该比特的值. 若猜测得分等于 1, 则猜测正确; 若猜测得分小于 1, 则猜测错误. 如果所有密钥的猜测值均为 1, 则说明该方法无法区分出正确密钥.

采用 C 语言实现差分计算分析, 对 SIMON-CEJO 方案首轮的轮密钥进行提取. 在 CPU 为 Intel(R) Core(TM) i3-8100 CPU@3.6 GHz, 运行内存大小为 4 GB 的计算机上执行该分析, 假设轮密钥为 0xa5726b81. 对轮密钥的前 4 比特进行猜测的结果如图 5 所示, 图 5 中横坐标表示猜测密钥, 纵坐标表示猜测得分. 猜测得分最高的为正确密钥, 但是由图 5 可以看到, 猜测得分均为 1, 即 DCA 攻击无法区分出正确密钥和错误密钥.

考虑结合 SIMON-CEJO 前两轮的操作, 利用前两轮的输出结果来确定首轮的轮密钥. 在 CPU 为 Intel(R) Core(TM) i3-8100 CPU@3.6 GHz, 运行内存大小为 4 GB 的计算机上进行 DCA 分析, 假设首轮的轮密钥为 0x57d23749. 由于第 2 轮的与运算之前需要循环左移, 因此将正确密钥的第 2–5 位和第 9–12 位关联起来作为攻击对象, 此时的分析结果如图 6 所示. 由于只有正确密钥的猜测得分才为 1, 因此图 6 表明 DCA 攻击可以将第 2–5 位和第 9–12 位同时区分出来. 同理可恢复出首轮轮密钥的其他比特位; 随后, 结合第 2、3 轮可得到第 2 轮的轮密钥. 以此类推, DCA 攻击可确定出 SIMON-CEJO 白盒实现的各轮轮密钥. 因此, 该方案不能抵抗 DCA 攻击.

Bos 等人[21]的差分计算分析之所以能攻击成功, 是在于 CEJO 框架本身的问题. CEJO 框架采用网络化编码的方式, 需要保证所加入的输入输出编码在轮与轮之间可以相互抵消, 从而实现算法功能的一致性. 而差分计算分析恰恰利用了这一点, 重点分析白盒实现算法的首末轮, 并根据 CEJO 框架的特性, 假设查找表的输入值与不加编码的情况相同, 以此来简化分析.

因此为了抵抗差分计算分析, 白盒实现方案需要在编码的选取做出改进. 对于 SIMON 算法, 可以考虑将与运算和轮密钥设计成查找表, 利用与运算的自等价对查找表编码. 这样由于自等价的特性, 与运算的左自等价不可以被抵消, 可以使得差分计算分析的假设不成立.

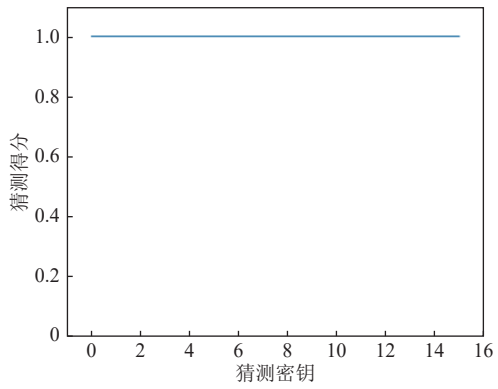

图 5 DCA 攻击一轮的结果

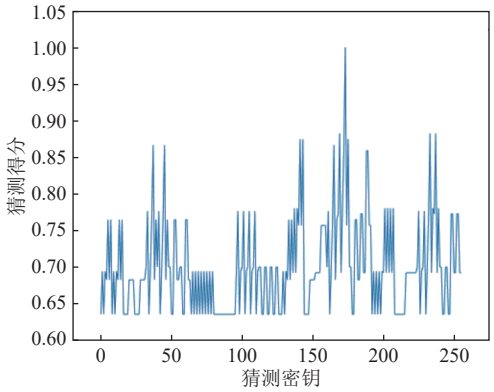

图 6 DCA 攻击两轮的结果

5 SIIMON 算法的 Benaloh 编码白盒实现

在本节中, 利用 Benaloh 编码对 SIMON64/128 进行白盒实现. 不同于经典 CEJO 框架的白盒实现方法, 该方案采用对单比特明文信息进行编码的思路[36], 利用编码函数的同态性, 直接对编码值执行运算, 最后再进行解码, 回到对应的密文结果. SIMON 算法仅包含异或运算、与运算和循环移位运算, 首先考虑对异或运算和与运算进行编码; 其次, 为了提高方案在 DCA 攻击下的安全性, 对与运算添加掩码, 加入随机数来进行混淆; 最后, 介绍整体算法的白盒实现流程.

5.1 对异或运算和与运算进行编码[36]

首先介绍编解码函数. 考虑对一个比特 a 进行编码, 利用改进的 Benaloh 同态加密方案进行密钥生成, 随机选取随机数 s, $0 \leqslant s < (p-1)/2$, 对 $2s+a$ 进行加密. 编码函数 Enc 如公式 (7) 所示, 对应的解码函数 Dec 如公式 (8) 所示.

$$Enc(a) = E_t(2s+a) = ty^{2s+a} \bmod p \tag{7}$$

$$Dec(Enc(a)) = D_t(Enc(a)) \bmod 2 \tag{8}$$

其中, 解码函数的正确性由公式 (9) 可得:

$$\begin{aligned} Dec(Enc(a)) &= D_t(Enc(a)) \bmod 2 \\ &= D_t\left(ty^{2s+a}\right) \bmod 2 \\ &= \log_x\left(\left(t^{-1}ty^{2s+a}\right)^{(p-1)/r}\right) \bmod p \bmod 2 \\ &= 2s+a \bmod 2 \\ &= a \end{aligned} \tag{9}$$

由于 SIMON 算法仅包含异或运算、与运算和循环移位运算, 因此考虑对异或运算和与运算编码.

考虑计算 $a \oplus b$. 首先计算 a 与 b 的编码值乘积. 由于 $Enc(a) \equiv t_0 y^{2s+a} \bmod p$, $Enc(b) \equiv t_1 y^{2s'+b} \bmod p$, 且 $a+b=2(a\&b)+a \oplus b$, 因此可以得到公式 (10):

$$\begin{aligned} Enc(a)Enc(b) &\equiv t_0 t_1 y^{2(s+s')+a+b} \bmod p \\ &\equiv t_0 t_1 y^{2(s+s'+(a\&b))+(a\oplus b)} \bmod p \end{aligned} \tag{10}$$

根据改进的 Benaloh 方案的同态性, 可以得到 $Enc(a)Enc(b)=Enc(a \oplus b)$, 即 $Dec(Enc(a)Enc(b))=a \oplus b$. 因此两个单比特的编码值结果相乘, 即可得到异或结果的编码值. 异或运算转化成模乘运算来实现, 同时隐藏了真实的 $a \oplus b$ 的值, 防止了秘密信息的泄露.

下面考虑计算 $a\&b$, 通过构造查找表来实现 $Enc(a\&b)$. 由公式 (10) 可以看到, y 的指数部分为 $2(s+s'+(a\&b))+a \oplus b$, 也就是 y 的右移量为 $s+s'+(a\&b)$. 同样地, 在 $Enc(a)$ 中 y 的右移量为 s, 在 $Enc(b)$ 中 y 的右移量为 s'. 若将这

3 个右移量相加, 则可以得到 2(s+s')+(a&b). 因此考虑下面 3 个查找表, 如图 7 所示. 输入分别是 $Enc(a)$、 $Enc(b)$ 和 $Enc(a)$ $Enc(b)$, 输出分别是对右移量的编码结果, 即 $Enc(s)$ 、$Enc(s')$ 和 $Enc(s+s'+(a\&b))$.

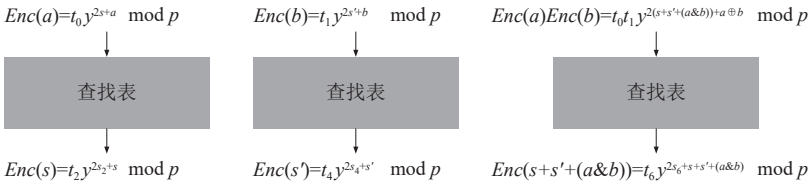

图 7 SIMON-Masking 的查找表

在查找表内部首先要对输入值进行解密, 再得到右移量, 最后对右移量利用编码函数进行编码, 得到输出值. 将这 3 个查找表的输出结果相乘, 可以得到结果 $ty^{2(S+s+s')+(a\&b)} \bmod p$, 这里 $t=t_2t_4t_6$, $S=s_2+s_4+s_6$. 将这个结果作为 $Enc(a\&b)$, 即 $Enc(a\&b)= ty^{2(S+s+s')+(a\&b)} \bmod p$. 可以验证, $Dec(Enc(a\&b))=a\&b$. 因此, 与运算转化成了对 3 个查找表的输出结果进行模乘运算操作.

5.2 对与运算添加布尔掩码

掩码防护技术[45]是一个用来抵抗侧信道分析的方法, 属于算法级的防护, 易于实现. 它通过随机化密码设备所处理的中间值, 让攻击者无法准确定位泄露点, 增加攻击难度. 布尔掩码是最早提出的、最常见、最易实现的掩码技术, 是轻量级分组密码白盒实现能抵抗 DCA 分析的一个行之有效而且必要的对策. 布尔掩码将每个秘密信息分成两个或多个分量, 保证这些分量的异或值等于真实秘密信息, 并对其进行操作, 以避免秘密信息的泄漏. 分组密码的布尔掩码通过将每个未受保护的操作, 替换为其掩码的对应操作. 下面考虑对与运算添加布尔掩码.

在 Battistello 等人所选用的布尔掩码方案[46]中, 对与运算添加掩码时引入了非运算. 考虑到 SIMON 算法只涉及循环移位运算、异或运算和与运算, 为了便于算法的白盒实现, 我们不希望再引入额外的运算操作, 因此选用 Trichina 等人[47]提出的对与运算的掩码方案. 文献 [42] 提出的掩码方案用到了异或运算和与运算, 以下是该方案的介绍.

对于两个比特 a_i, b_j, 欲计算 $a_i\&b_j$, 选取随机比特 u_i, v_j, 令 $\tilde{a}=a_i\oplus u_i, \tilde{b}=b_j\oplus v_j$, 将两个掩码值作与运算, 如公式 (11) 所示.

$$\tilde{a}\&\tilde{b}=(a_i\oplus u_i)\&(b_j\oplus v_j)=(a_i\&b_j)\oplus(a_i\&v_j)\oplus(u_i\&b_j)\oplus(u_i\&v_j) \tag{11}$$

由于 $u_i\&b_j=\left(u_i\&\tilde{b}\right)\oplus(u_i\&v_j), a_i\&v_j=(v_j\&\tilde{a})\oplus(u_i\&v_j)$, 所以公式 (12) 和公式 (13) 成立.

$$\tilde{a}\&\tilde{b}=(a_i\&b_j)\oplus\left(u_i\&\tilde{b}\right)\oplus(v_j\&\tilde{a})\oplus(u_i\&v_j) \tag{12}$$

$$a_i\&b_j=\left(\tilde{a}\&\tilde{b}\right)\oplus\left(u_i\&\tilde{b}\right)\oplus(v_j\&\tilde{a})\oplus(u_i\&v_j) \tag{13}$$

为进一步增强掩码的安全性, 选取随机比特 z_i, 计算公式 (14). 这样计算 a_i & b_j 转化成了对应掩码值与随机比特之间的运算, 由 4 次与运算和 4 次异或运算组成. 随机比特 z_i 在算法执行过程中被抵消.

$$(a_i\&b_j)\oplus z_i=\left(\tilde{a}\&\tilde{b}\right)\oplus\left(u_i\&\tilde{b}\right)\oplus(v_j\&\tilde{a})\oplus(u_i\&v_j)\oplus z_i \tag{14}$$

5.3 SIMON-Masking 白盒实现

使用改进的 Benaloh 编码[36]与布尔掩码技术[39]相结合的方法, 对 SIMON64/128 算法进行白盒实现.

步骤 1. 利用改进的 Benaloh 同态加密方案进行密钥生成, 生成需要的参数. 为了保证编解码的正确性, 参数值是固定的, 编码值随着明文比特信息的改变而改变.

步骤 2. 对数据比特进行编码, 分为对主密钥编码和对明文编码. 密钥 k 长度为 l 比特, 将编码的主密钥比特表嵌入到白盒中, 输入为 l 个密钥比特, 输出为每个比特对应的编码值, 即 $t_jy^{2s_j+k_j} \bmod p$, 其中 $1\leqslant j\leqslant l$, k_j 为密钥 k 的第 j 个比特. 对明文比特进行编码, 并生成 1 比特输入的编码表, 表的形式如公式 (15) 所示.

$$T_i = \{Enc(0), Enc(1)\} = \left\{t_i y^{2s_i} \bmod p, t_i y^{2s_i'+1} \bmod p\right\} \tag{15}$$

步骤 3. 执行密钥编排算法和加密轮函数. 由于密钥编排算法和轮函数都只包含循环移位运算、与运算和异或运算, 因此可以按照第 5.1 节所述的方式进行运算, 同时对与运算添加掩码.

步骤 4. 对数据比特进行解码. 使用编码比特对应的解码表, 由于编码时每个比特所使用的私钥不一定相同, 因此每个密文比特都需要解码表, 以此来获得算法的真正密文.

在执行算法的过程中, 比特信息都是以编码形式进行运算, 编码位之间的所有操作都是通过循环移位、模乘和查找表操作来完成的.

(1) 对于循环移位操作, 将已编码比特值组成的比特串进行循环移位, 等同于先循环移位再进行编码, 因此不需要对此操作进行额外的白盒实现保护;

(2) 对于异或运算, 将两个编码的比特相乘, 即可得到编码的两比特异或的结果 $Enc(a \oplus b)$;

(3) 对于与运算, 将 3 个查找表的输出结果相乘, 即可得到编码的两比特作与运算的结果 $Enc(a\&b)$.

考虑对与运算添加掩码. 在算法的轮函数中, 涉及与运算的操作为 $(x_{i+1}<<<1)\&(x_{i+1}<<<8) \oplus x_i$, 记比特串 $a_{i,0}a_{i,1}\ldots a_{i,31}=x_{i+1}<<<1$, $b_{i,0}b_{i,1}\ldots b_{i,31}=x_{i+1}<<<8$. 利用第 3.2 节的掩码方案, 随机选取随机比特 $u_{i,j}$, $v_{i,j}$ 和 $z_{i,j}$, j=0, 1,…,31. 将计算 $(a_{i,j}\&b_{i,j}) \oplus z_{i,j}$ 转化成计算 $\left(\tilde{a}_{i,j}\&\tilde{b}_{i,j}\right) \oplus \left(u_{i,j}\&\tilde{b}_{i,j}\right) \oplus (v_{i,j}\&\tilde{a}_{i,j}) \oplus (u_{i,j}\&v_{i,j}) \oplus z_{i,j}$, 其中 $\tilde{a}_{i,j} = a_{i,j} \oplus u_{i,j}, \tilde{b}_{i,j} = b_{i,j} \oplus v_{i,j}$, 如公式 (16) 所示.

$$(a_{i,j}\&b_{i,j}) \oplus z_{i,j} = \left(\tilde{a}_{i,j}\&\tilde{b}_{i,j}\right) \oplus \left(u_{i,j}\&\tilde{b}_{i,j}\right) \oplus (v_{i,j}\&\tilde{a}_{i,j}) \oplus (u_{i,j}\&v_{i,j}) \oplus z_{i,j} \tag{16}$$

由 $Dec(Enc(a)Enc(b))= a \oplus b$ 可推知 $Dec\left(\prod_{i=1}^{n} Enc(a_i)\right) = a_1 \oplus \ldots \oplus a_n$, 因此公式 (17) 成立.

$$\begin{aligned} Enc((a_{i,j}\&b_{i,j}) \oplus z_{i,j}) &= Enc\left(\left(\tilde{a}_{i,j}\&\tilde{b}_{i,j}\right) \oplus \left(u_{i,j}\&\tilde{b}_{i,j}\right) \oplus (v_{i,j}\&\tilde{a}_{i,j}) \oplus (u_{i,j}\&v_{i,j}) \oplus z_{i,j}\right) \\ &= Enc\left(\tilde{a}_{i,j}\&\tilde{b}_{i,j}\right) Enc\left(u_{i,j}\&\tilde{b}_{i,j}\right) Enc(v_{i,j}\&\tilde{a}_{i,j}) Enc(u_{i,j}\&v_{i,j}) Enc(z_{i,j}) \end{aligned} \tag{17}$$

从公式 (17) 可以看到: 一个添加布尔掩码的比特与运算在白盒实现时需要 12 个查找表, 12 次模乘运算. 令 x_i 异或 z_i, 即 $x_i \oplus z_i = (x_{i,0}x_{i,1}\ldots x_{i,31}) \oplus (z_{i,0}z_{i,1}\ldots z_{i,31})$, 使得 $(x_{i+1}<<<1)\&(x_{i+1}<<<8)$ 与 x_i 作异或运算后, 随机比特 z_i, i=0, 1,…, 31 被抵消. 需要计算 $Enc(x_{i,j} \oplus z_{i,j}) = Enc(x_{i,j}) Enc(z_{i,j})$, 因此在添加掩码后, 每轮多了 32 次模乘运算.

6 SIMON-Masking 的效率及安全性分析

本节分析 SIMON-Masking 白盒实现的效率和安全性.

由于该方案的设计思路和经典的 CEJO 框架的思路不同, 因此对 CEJO 框架的代数攻击对 SIMON-Masking 均不适用. Battistello 等人[36]在评估该新型编码的白盒实现安全性时, 主要考虑了输入输出平方攻击、频率攻击以及 DCA 攻击. SIMON-Masking 通过对单比特信息进行编码混淆, 实现了算法的白盒运算. 这 3 种攻击通过分析方案的编码方式以及查找表的输入输出, 来研究秘密比特的勒让德符号, 进一步获取比特信息.

6.1 效率分析

在内存占用方面, 对 128 比特密钥进行编码, 需要 128 张编码表; 对 64 比特明文进行编码, 需要 64 张编码表; 对 64 比特密文进行解码, 需要 64 张解码表. 另外执行与运算时, 在添加布尔掩码后, 每次比特与运算需要 12 张查找表, 每轮需要执行 32 次比特与运算, 一共有 44 轮, 需要查找表 32×12×44=16896 张.

表的总数为 128+64+64+16896=17152. 每张表占用内存为 $p \times \lceil \log_2 p \rceil$ 比特. 表 3 给出了 SIMON-Masking 方案的内存占用上下界, 上下界的大小取决于素数 p 的值. 这里选择 6 比特的素数 p, 既不会使白盒实现的安全性过于低, 又保证了内存空间不会很大. 当 p=53 时, 方案的内存占用为 665.81 KB.

算法需要查表的次数为 32×12×44=16896, 模乘运算的次数为 5×40+44×(12×32+4×32)=22728, 其中“5×40”表示执行密钥编排时所需的模乘次数, “12×32”表示两个 32 比特串执行与运算所需的模乘次数, “4×32”表示每轮需要执行 4 次 32 比特两两异或, 每次比特异或需要一次模乘运算.

现场可编程阵列 (FPGA) 由于灵活性强、适用性强和并行计算等特点, 成为有吸引力的硬件性能测试评估平台. 对于 SIMON-Masking 白盒实现方案, 在真实硬件 FPGA 设计时, 可分为初始化模块、密钥流生成模块、编解码模块和运算模块.

表 3 根据 p 的比特长度所得的内存占用上下界

$\lceil \log_2 p \rceil$	内存占用 (MB)
4	[0.07, 0.12]
6	[0.39, 0.77]
8	[2.09, 4.17]
10	[10.47, 20.92]
12	[50.25, 100.48]
14	[234.5, 468.97]
16	[1072, 2143.97]

2022 年程碧倩等人[48]提出的一种多项式展开的交叉蒙哥马利模乘算法在 FPGA 实现上具有良好的性能, 因此在进行模乘操作的 FPGA 设计时可以考虑采用该方法. 同时, 使用循环移位寄存器和异或门来分别实现循环左移操作和异或操作. 根据效率分析, SIMON-Masking 需要 22 728 次模乘, 16 896 次查表. 另外, 对主密钥和明文编码和对数据比特解码时, 由于各数据间不涉及运算, 可以考虑并行化处理, 以保证 FPGA 的高效运行. 研究该白盒实现方案在硬件 FPGA 上的实现具有很强的实践意义, 在后续的工作中我们会继续对方案的 FPGA 实验进行研究, 使方案设计更实际.

6.2 输入输出平方攻击

在白盒攻击环境下, 攻击者可以得到查找表的输入输出值. 输入输出平方攻击指的是攻击者通过对查找表的输入和输出值进行平方运算, 利用勒让德符号尝试去获取秘密比特信息.

对于秘密比特 b, 令 $n_{i,b} = t_i y_b^{2s+b} \bmod p$ 为图 7 查找表的输入值, $n_{o,b} = t_o y_b^{2s+s} \bmod p$ 为查找表的输出值. 简约起见, 令 $s=0$. 将输入值与输出值的平方相乘, 可得到 $n_{i,b} n_{o,b}^2 \equiv t_i t_o^2 y^{4s_b+b} \bmod p$. 当 $b=0$ 和 $b=1$ 时, 攻击者可以得到 $n_{i,0} n_{o,0}^2$ 和 $n_{i,1} n_{o,1}^2$ 这两个值, 将这两个值相乘可以得到公式 (18).

$$z = n_{i,0}, n_{i,1}, n_{o,0}^2, n_{o,1}^2 \bmod p = t_i^2 t_o^4 y^{4(s_0+s_s)+1} \bmod p \tag{18}$$

由于 y 是 Z_p^* 的生成元, 且 $t_i, t_o \in Z_p^*$, 因此存在 $\tau_i, \tau_o \in \{0, \ldots, p-1\}$ 使得 $t_i = y^{\tau_i} \bmod p$, $t_o = y^{\tau_o} \bmod p$, 进而可以得到 $z = y^{4(\tau_0+s_0+s_1)+2\tau_i+1} \bmod p$. 由于 $\left(\frac{t_i}{p}\right) = \tau_i \bmod 2$, 如果攻击者可以从 z 中获得 τ_i 的奇偶性, 那么就可以得到 $\left(\frac{t_i}{p}\right)$ 的值, 进而利用 $n_{i,b} = t_i y_b^{2s+b} \bmod p$ 获取秘密信息 b. 但实际上, 取 Z_p^* 的另一个生成元 y', 则存在整数 a, 使得 $y = y'^{2a+1}$, 得到公式 (19) 成立.

$$\begin{aligned} \log_{y'}(z) &= \log_{y'}(y) \log_y(z) \\ &\equiv (2a+1)(2\tau_i+1) \bmod 4 \\ &\equiv 2(a+\tau_i)+1 \bmod 4 \end{aligned} \tag{19}$$

可以看到, $\log_y z$ 的值实际上与所选取的生成元 y 是相互独立的. 由于 a 的奇偶性不定, 因此 τ_i 的奇偶性无法确定, 即从 z 的值是无法获得 τ_i 的奇偶性的, 攻击者无法获取秘密信息 b.

6.3 频率攻击

频率攻击指的是根据编码时所选取的随机数的概率, 得出查找表输出值的勒让德符号, 从而获取到秘密比特间的关系.

与运算不添加布尔掩码时, 考虑两个比特 a_0, a_1 进行与运算, 给出 4 个整数 $s_0^{(0)}, s_0^{(1)}, s_1^{(0)}, s_1^{(1)}$ 作为对 a_0 和 a_1 编码时的随机数. 当 a_i 为 0 时, 选择 $s_i^{(0)}$ 作为随机数; 当 a_i 为 1 时, 选择 $s_i^{(1)}$ 作为随机数. 令 a_0 和 a_1 的编码值分别为 $Enc(a_0) = t_0 y^{2s_0^{(a_0)}+a_0} \bmod p$ 和 $Enc(a_1) = t_1 y^{2s_1^{(a_1)}+a_1} \bmod p$, 当查找表的输入为 $Enc(a_0)Enc(a_1)$ 时, 设输出值为 $\alpha = t y^{2\delta+\beta} \bmod p$,

其中 $\beta=(a_0\&a_1)\oplus\left(\left(s_0^{(a_0)}+s_1^{(a_1)}\right) \bmod 2\right)$. 根据勒让德符号的性质, 有$\left(\frac{\alpha}{p}\right)=\left(\frac{t}{p}\right)(1-2\beta)$. 当$s_0^{(0)}\equiv s_0^{(1)} \bmod 2$ 和 $s_1^{(0)}\equiv s_1^{(1)} \bmod 2$ 时, $\left(\frac{\alpha}{p}\right)=\left(\frac{t}{p}\right)$ 的概率为 3/4. 此时 β=0, 会泄露 $(a_0\&a_1)$ 的值.

添加布尔掩码后, 进行与运算的都是掩码值, 真实值不直接参与查找表的操作. 即使攻击者可以获取到比特间的关系, 他们得到的也是掩码值之间的关系, 而不是真实比特信息. 布尔掩码的混淆作用保护了秘密信息不被泄露, 保证了方案在频率攻击下的安全性.

6.4 基于勒让德符号的二阶 DCA 攻击

密钥以编码的形式嵌入在白盒中, 并以编码值来参与运算. 通过这种方式编码, 仅查看密钥的编码形式无法提取任何信息. 每个密钥位 k_j 都有各自的随机私钥 t_j, $Enc(k_j)$ 的值保护了真实的 k_j 值.

将密码视为一个逻辑电路, 任何门输出都使用一个攻击者未知的随机数 t 进行 Benaloh 编码, 但白盒实现的所有执行过程都是固定的, 门输出的勒让德符号的任何变化都只是由于编码位的变化. 攻击者可以将注意力集中在与门上, 与门的输出取决于密钥比特位. 通过对密钥比特位进行假设, 当输入明文发生变化时, 攻击者可以计算目标与门的预期输出序列. 通过将该序列与白盒实际输出的勒让德符号进行比较, 攻击者可以接受或拒绝对子密钥的假设. 根据其他密钥位使用不同的与门重复此过程, 可以减少可能的密钥子集.

针对添加布尔掩码的方案, Biryukov 等人在文献 [29] 中提出了组合 DCA 分析方法. 通过猜测秘密值 v 被分成的 t 个分量, 来构造新的向量 m; 并分别计算向量 m 和预测值的相关性和二者各分量间的相关性. 对于 v 的每个预测值, 若相关性的值高于所预定的值, 则说明攻击者可以从预测值中获取到部分秘密信息. 他们指出利用该攻击方法, 分析基于掩码的白盒实现方案的时间复杂度为 $O(n^tk2^{2t})$. 其中 n 是攻击所选取的电路大小; k 是进行密钥假设的数量; 2^{2t} 表示在 t 比特向量 m 上迭代并计算相关性的复杂度.

因此, 对于 SIMON-Masking 方案, 基于勒让德符号的二阶 DCA 攻击复杂度为 $O(n^2k\log_2 p)$. $\log_2 p$ 是执行欧拉判别的复杂度, 需要通过勒让德符号判别是否为模 p 的二次剩余. 相比于不添加布尔掩码的白盒实现方案, SIMON-Masking 在抵抗差分计算分析上更有优势. 对于高阶差分计算分析, 该方案的安全性还需要进一步研究.

6.5 SIMON-CEJO 与 SIMON-Masking 的对比

本节将 SIMON 算法的两种白盒实现的效率和安全性分析作对比.

SIMON-CEJO 和 SIMON-Masking 在效率上的对比如表 4 所示. 可以看到, SIMON-CEJO 在性能上优于 SIMON-Masking, SIMON-CEJO 的内存占用较少, 同时 SIMON-Masking 需要的查表次数和模乘运算次数较多.

SIMON-CEJO 和 SIMON-Masking 在安全性上的对比如表 5 所示. “√” 意味着方案可以抵抗某种攻击, “—” 表示目前方案还未进行对该攻击的分析, “×” 意味着方案不可以抵抗某种攻击. SIMON-CEJO 方案不具备抵抗差分计算分析的能力, 而 SIMON-Masking 对于抵抗差分计算分析有一定的安全性. 可以看到, 在白盒实现方案中添加布尔掩码, 能提高方案在差分计算分析下的安全性. 同时, 将新型编码应用于白盒实现方案是一种新思路.

表 4 效率对比

方案	内存占用 (KB)	异或运算次数	模乘/乘法运算次数
SIMON-CEJO	369.016	88	132
SIMON-Masking	665.81	0	22 728

表 5 安全性对比

方案	BGE攻击	仿射等价算法攻击	DCA攻击	输入输出平方攻击	频率攻击
SIMON-CEJO	√	√	×	—	—
SIMON-Masking	—	—	$O(n^2k\log_2 p)$	√	√

DCA 攻击不考虑外部编码, 并且只对首轮和末轮的查找表进行分析. 由于白盒实现需要保证算法结果的正确性, 内部编码在算法执行过程中会被抵消, 攻击者假设在进入非线性运算之前的值与未加编码的值是相同的. 目前大多数 CEJO 框架的白盒实现都无法抵抗 DCA 攻击的原因是它们采用了网络化编码, 是满足这个假设的, 这是

CEJO 框架的不足之处. 第 4.5 节也表明 SIMON-CEJO 白盒实现不足以抵抗 DCA 攻击. 因此寻找新型编码对于白盒实现方案能否抵抗 DCA 攻击至关重要.

7 总 结

本文提出了 SIMON 算法的两种白盒实现方法. SIMON-CEJO 白盒实现采用经典的 CEJO 框架, 将密钥嵌入在查找表中, 通过随机的可逆仿射变换来混淆密钥. 该方案的安全性分析表明方案在 BGE 攻击和仿射等价算法攻击下是安全的, 但不能抵抗 DCA 攻击. SIMON-Masking 白盒实现基于 Battistello 等人[36]提出的编码方案, 利用 Benaloh 加密方案的同态性, 将异或运算和与运算转化为模乘和查找表操作, 算法的执行过程都是在编码状态下完成的. 方案中的随机数使得攻击者无法通过编码值来获取真实比特信息; 同时对与运算添加布尔掩码, 进一步增加 DCA 攻击的难度.

SIMON-Masking 与 SIMON-CEJO 相比, 对单比特进行编码, 具有内存小、运行快的特点, 适合用于轻量级分组密码的白盒实现. 该方法对与运算和异或运算进行白盒实现, 对其他 ARX 结构的密码算法同样适用. 添加布尔掩码是白盒实现方案应对 DCA 攻击的一种有效的手段, 研究如何在白盒实现中添加掩码是未来需要持续关注的问题.

References:

[1] Chow S, Eisen P, Johnson H, van Oorschot PC. White-box cryptography and an AES implementation. In: Proc. of the 9th Int'l Workshop on Selected Areas in Cryptography. St. John's: Springer, 2003. 250–270. [doi: 10.1007/3-540-36492-7_17]

[2] Chow S, Eisen P, Johnson H, van Oorschot PC. A white-box DES implementation for DRM applications. In: Proc. of the 2003 ACM Workshop on Digital Rights Management. Washington: Springer, 2003. 1–15. [doi: 10.1007/978-3-540-44993-5_1]

[3] Billet O, Gilbert H, Ech-Chatbi C. Cryptanalysis of a white box AES implementation. In: Proc. of the 11th Int'l Workshop on Selected Areas in Cryptography. Waterloo: Springer, 2005. 227–240. [doi: 10.1007/978-3-540-30564-4_16]

[4] Jacob M, Boneh D, Felten E. Attacking an obfuscated cipher by injecting faults. In: Proc. of the 2003 ACM Workshop on Digital Rights Management. Washington: Springer, 2003. 16–31. [doi: 10.1007/978-3-540-44993-5_2]

[5] Michiels W, Gorissen P, Hollmann HDL. Cryptanalysis of a generic class of white-box implementations. In: Proc. of the 15th Int'l Workshop on Selected Areas in Cryptography. Sackville: Springer, 2009. 414–428. [doi: 10.1007/978-3-642-04159-4_27]

[6] Bringer J, Chabanne H, Dottax E. White box cryptography: Another attempt. Cryptology ePrint Archive, 2006: 468.

[7] Karroumi M. Protecting white-box AES with dual ciphers. In: Proc. of the 13th Int'l Conf. on Information Security and Cryptology. Seoul: Springer, 2011. 278–291. [doi: 10.1007/978-3-642-24209-0_19]

[8] Xiao YY, Lai XJ. A secure implementation of white-box AES. In: Proc. of the 2nd Int'l Conf. on Computer Science and its Applications. Jeju: IEEE, 2009. 1–6. [doi: 10.1109/CSA.2009.5404239]

[9] Luo R, Lai XJ, You R. A new attempt of white-box AES implementation. In: Proc. of the 2014 IEEE Int'l Conf. on Security, Pattern Analysis, and Cybernetics. Wuhan: IEEE, 2014. 423–429. [doi: 10.1109/SPAC.2014.6982727]

[10] Xiao YY, Lai XJ. White-box cryptography and a SMS4 implementation. In: Proc. of the 2009 ChinaCrypt. Guangzhou: Chinese Association for Cryptologic Research, 2009. 24–34 (in Chinese with English abstract).

[11] Shi Y, Wei WJ, He ZJ. A lightweight white-box symmetric Encryption algorithm against node capture for WSNs. Sensors, 2015, 15(5): 11928–11952. [doi: 10.3390/s150511928]

[12] Bai KP, Wu CK. A secure white-box SM4 implementation. Security and Communication Networks, 2016, 9(10): 996–1006. [doi: 10.1002/sec.1394]

[13] Pang SY, Lin TT, Lai XJ, Gong Z. A white-box implementation of IDEA. Symmetry, 2021, 13(6): 1066. [doi: 10.3390/sym13061066]

[14] De Mulder Y, Wyseur B, Preneel B. Cryptanalysis of a perturbated white-box AES implementation. In: Proc. of the 11th Int'l Conf. on Cryptology in India. Hyderabad: Springer, 2010. 293–310. [doi: 10.1007/978-3-642-17401-8_21]

[15] Lepoint T, Rivain M, De Mulder Y, Roelse P, Preneel B. Two attacks on a white-box AES implementation. In: Proc. of the 20th Int'l Workshop on Selected Areas in Cryptography. Burnaby: Springer, 2014. 265–285. [doi: 10.1007/978-3-662-43414-7_14]

[16] De Mulder Y, Roelse P, Preneel B. Cryptanalysis of the Xiao-Lai white-box AES implementation. In: Proc. of the 19th Int'l Conf. on Selected Areas in Cryptography. Windsor: Springer, 2013. 34–49. [doi: 10.1007/978-3-642-35999-6_3]

[17] Bai KP, Wu CK, Zhang ZF. Protect white-box AES to resist table composition attacks. IET Information Security, 2018, 12(4): 305–313.

[doi: 10.1049/iet-ifs.2017.0046]

[18] Lin TT, Lai XJ. Efficient attack to white-box SMS4 implementation. Ruan Jian Xue Bao/Journal of Software, 2013, 24(9): 2238–2249 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4356.htm [doi: 10.3724/SP.J.1001.2013.04356]

[19] Pan WL, Qin TH, Jia Y, Zhang LT. Cryptanalysis of two white-box SM4 implementations. Journal of Cryptologic Research, 2018, 5(6): 651–670 (in Chinese with English abstract). [doi: 10.13868/j.cnki.jcr.000274]

[20] Wang RS, Guo H, Lu JQ, Liu JW. Cryptanalysis of a white-box SM4 implementation based on collision attack. IET Information Security, 2022, 16(1): 18–27. [doi: 10.1049/ise2.12045]

[21] Bos JW, Hubain C, Michiels W, Teuwen P. Differential computation analysis: Hiding your white-box designs is not enough. In: Proc. of the 18th Int'l Conf. on Cryptographic Hardware and Embedded Systems. Santa Barbara: Springer, 2016. 215–236. [doi: 10.1007/978-3-662-53140-2_11]

[22] Su S, Dong H, Fu G, Zhang CP, Zhang M. A white-box CLEFIA implementation for mobile devices. In: Proc. of the Communications Security Conf. London: IET, 2014. 1–8. [doi: 10.1049/cp.2014.0752]

[23] Gong YT. Security analysis and improvement of white-box CLEFIA algorithm [MS. Thesis]. Xi'an: Xidian University, 2019 (in Chinese with English abstract). [doi: 10.27389/d.cnki.gxadu.2019.002505]

[24] Zhou L, Su CH, Wen YM, Li WJ, Gong Z. Towards practical white-box lightweight block cipher implementations for IoTs. Future Generation Computer Systems, 2018, 86: 507–514. [doi: 10.1016/j.future.2018.04.011]

[25] Yao S. The design and analysis of white-box implementation [MS. Thesis]. Xi'an: Xidian University, 2021 (in Chinese with English abstract). [doi: 10.27389/d.cnki.gxadu.2021.002077]

[26] Chen J, Tong P, Yao S. A white-box implementation scheme of lightweight block cipher GIFT. Netinfo Security, 2021, 21(2): 16–23 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-1122.2021.02.003]

[27] Ranea A, Vandersmissen J, Preneel B. Implicit white-box implementations: White-boxing ARX ciphers. In: Proc. of the 13th Annual Int'l Cryptology Conf. Seoul: Springer, 2022. 33–63. [doi: 10.1007/978-3-031-15802-5_2]

[28] Vandersmissen J, Ranea A, Preneel B. A white-box speck implementation using self-equivalence encodings. In: Proc. of the 20th Int'l Conf. on Applied Cryptography and Network Security. Rome: Springer, 2022. 771–791. [doi: 10.1007/978-3-031-09234-3_38]

[29] Biryukov A, Udovenko A. Attacks and countermeasures for white-box designs. In: Proc. of the 24th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Brisbane: Springer, 2018. 373–402. [doi: 10.1007/978-3-030-03329-3_13]

[30] Alpirez Bock E, Brzuska C, Michiels W, Treff A. On the ineffectiveness of internal encodings—Revisiting the DCA attack on white-box cryptography. In: Proc. of the 16th Int'l Conf. on Applied Cryptography and Network Security. Leuven: Springer, 2018. 103–120. [doi: 10.1007/978-3-319-93387-0_6]

[31] Goubin L, Rivain M, Wang JW. Defeating state-of-the-art white-box countermeasures with advanced gray-box attacks. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2020, 2020(3): 454–482. [doi: 10.13154/tches.v2020.i3.454-482]

[32] Rivain M, Wang JW. Analysis and improvement of differential computation attacks against internally-encoded white-box implementations. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2019, 2019(2): 225–255. [doi: 10.13154/tches.v2019.i2.225-255]

[33] Sasdrich P, Moradi A, Güneysu T. White-box cryptography in the gray box. In: Proc. of the 23rd Int'l Conf. on Fast Software Encryption. Bochum: Springer, 2016. 185–203. [doi: 10.1007/978-3-662-52993-5_10]

[34] Lee S, Kim M. Improvement on a masked white-box cryptographic implementation. IEEE Access, 2020, 8: 90992–91004. [doi: 10.1109/ACCESS.2020.2993651]

[35] Seker O, Eisenbarth T, Liskiewicz M. A white-box masking scheme resisting computational and algebraic attacks. IACR Trans. on Cryptographic Hardware and Embedded Systems, 2021, 2021(2): 61–105. [doi: 10.46586/tches.v2021.i2.61-105]

[36] Battistello A, Castelnovi L, Chabrier T. Enhanced encodings for white-box designs. In: Proc. of the 20th Int'l Conf. on Smart Card Research and Advanced Applications. Lübeck: Springer, 2022. 254–274. [doi: 10.1007/978-3-030-97348-3_14]

[37] Benaloh J. Dense probabilistic encryption. In: Proc. of the 1999 Int'l Workshop on Selected Areas in Cryptography. 1999. 120–128.

[38] Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, 2013: 404.

[39] AIkhzaimi H, Lauridsen MM. Cryptanalysis of the SIMON family of block ciphers. Cryptography ePrint Archive, 2013: 543.

[40] Biryukov A, Roy A, Velichkov V. Differential analysis of block ciphers SIMON and SPECK. In: Proc. of the 21st Int'l Workshop on Fast Software Encryption. London: Springer, 2015. 546–570. [doi: 10.1007/978-3-662-46706-0_28]

[41] Boura C, Naya-Plasencia M, Suder V. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and SIMON. In: Proc. of the 20th Int'l Conf. on the Theory and Application of Cryptology and Information Security. Kaoshiung,

China: Springer, 2014. 179–199. [doi: 10.1007/978-3-662-45611-8_10]

[42] Alizadeh J, Bagheri N, Gauravaram P, Kumar A, Sanadhya SK. Linear cryptanalysis of round reduced SIMON. Cryptography ePrint Archive, 2013: 663.

[43] Abdelraheem MA, Alizadeh J, AlKhzaimi HA, Reza Aref M, Bagheri N, Gauravaram P. Improved linear cryptanalysis of reduced-round SIMON-32 and SIMON-48. In: Proc. of the 16th Int'l Conf. on Cryptology in India. Bangalore: Springer, 2015. 153–179. [doi: 10.1007/978-3-319-26617-6_9]

[44] Biryukov A, De Cannière C, Braeken A, Preneel B. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: Proc. of the 2003 Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Warsaw: Springer, 2003. 33–50. [doi: 10.1007/3-540-39200-9_3]

[45] Mangard S, Oswald E, Popp T. Power Analysis Attacks—Revealing the Secrets of Smart Cards. New York: Springer, 2007. 223–244. [doi: 10.1007/978-0-387-38162-6]

[46] Biryukov A, Dinu D, Le Corre Y, Udovenko A. Optimal first-order boolean masking for embedded IoT devices. In: Proc. of the 16th Int'l Conf. on Smart Card Research and Advanced Applications. Lugano: Springer, 2018. 22–41. [doi: 10.1007/978-3-319-75208-2_2]

[47] Trichina E. Combinational logic design for AES SubByte transformation on masked data. Cryptography ePrint Archive, 2003: 236.

[48] Cheng BQ, Liu GZ, Xiao H. Improved montgomery modular multiplication algorithm and FPGA implementation. Electronic Science and Technology, 2022, 35(7): 58–63 (in Chinese with English abstract). [doi: 10.16180/j.cnki.issn1007-7820.2022.07.010]

附中文参考文献:

[10] 肖雅莹, 来学嘉. 白盒密码及 SMS4 算法的白盒实现. 见: 中国密码学会 2009 年会论文集. 广州: 中国密码学会, 2009. 24–34.

[18] 林婷婷, 来学嘉. 对白盒 SMS4 实现的一种有效攻击. 软件学报, 2013, 24(9): 2238–2249. http://www.jos.org.cn/1000-9825/4356.htm [doi: 10.3724/SP.J.1001.2013.04356]

[19] 潘文伦, 秦体红, 贾音, 张立廷. 对两个 SM4 白盒方案的分析. 密码学报, 2018, 5(6): 651–670. [doi: 10.13868/j.cnki.jcr.000274]

[23] 宫雅婷. 白盒 CLEFIA 算法的安全性分析与改进 [硕士学位论文]. 西安: 西安电子科技大学, 2019. [doi: 10.27389/d.cnki.gxadu.2019.002505]

[25] 姚思. 白盒实现的设计与分析 [硕士学位论文]. 西安: 西安电子科技大学, 2021. [doi: 10.27389/d.cnki.gxadu.2021.002077]

[26] 陈杰, 童鹏, 姚思. 轻量级分组密码 GIFT 的一种白盒实现方案. 信息网络安全, 2021, 21(2): 16–23. [doi: 10.3969/j.issn.1671-1122.2021.02.003]

[48] 程碧倩, 刘光柱, 肖昊. 改进的蒙哥马利模乘算法及 FPGA 实现. 电子科技, 2022, 35(7): 58–63. [doi: 10.16180/j.cnki.issn1007-7820.2022.07.010]

罗一诺(1998－), 女, 硕士生, 主要研究领域为白盒密码的设计与安全性分析.

王超(1997－), 男, 硕士生, 主要研究领域为白盒密码的安全性分析.

陈杰(1979－), 女, 博士, 副教授, 主要研究领域为密码算法的设计与安全性分析.