·网络空间安全专题· 文章编号:1000-3428(2019)08-0007-07 文献标志码:A 中图分类号:TP309.2

基于优化故障定位的SIMECK密码代数故障攻击

黄长阳,王 韬,王晓晗,陈庆超,尹世庄

(陆军工程大学 装备模拟训练中心,石家庄 050003)

摘 要: 针对SIMECK密码,提出一种优化故障定位的代数故障攻击方法。通过分析SIMECK轮函数加密扩散缺陷及故障失效原因,提取故障确定性传播特征并构建确定性故障差分特性表,实现故障的精确定位。创建加密过程和故障信息等效方程组,将方程组转化为SAT问题并求解密钥。实验结果表明,该方法在SIMECK32/64第28轮左寄存器中注入随机单比特故障,仅需8次故障注入即可恢复完整64 bit主密钥,攻击成功率高达99.61%,相比已有故障攻击方法所需故障样本量更少,攻击成功率及创建方程自动化程度更高。

关键词: SIMECK密码;代数故障攻击;故障失效;轮函数缺陷;故障模型;CryptoMiniSat解析器

中文引用格式: 黄长阳,王韬,王晓晗,等. 基于优化故障定位的SIMECK密码代数故障攻击[J]. 计算机工程,2019,45(8):7-13,21.

英文引用格式: HUANG Changyang,WANG Tao,WANG Xiaohan,et al. Algebraic fault attack against SIMECK cipher based on optimized fault location[J]. Computer Engineering,2019,45(8):7-13,21.

Algebraic Fault Attack Against SIMECK Cipher Based on Optimized Fault Location

HUANG Changyang,WANG Tao,WANG Xiaohan,CHEN Qingchao,YIN Shizhuang

(Center of Equipment Simulation Training,Army Engineering University,Shijiazhuang 050003,China)

【Abstract】 This paper proposes a algebraic fault attack method based on optimized fault location against SIMECK cipher. By analyzing encryption diffusion defect of the SIMECK round function and the failure cause,the deterministic propagation characteristics of faults are extracted,and the differential characteristic table of deterministic faults is constructed to achieve accurate fault location. It creates an equivalent equation set for the encryption process and fault information,and converts the equations into SAT problems and solve the key. Experimental results show that the method can inject a random single-bit fault into the left register in the 28th round of SIMECK32/64,and only need 8 fault injections to recover the complete 64 bit master key. The attack success rate is 99.61%. Compared with the existing fault attack method,the proposed method requires fewer fault samples and has a higher attack success rate,creation of equations more automated.

【Key words】 SIMECK cipher;algebraic fault attack;fault failure;round function defect;fault model;CryptoMiniSat parser

DOI: 10.19678/j.issn.1000-3428.0051747

0 概述

代数故障攻击[1]是一种兼顾代数分析和故障攻击[2]优势的密码分析方法,并通过1 bit故障注入穷举24 bit密钥即可恢复DES全部密钥。代数故障攻击能够解决代数分析复杂度过高及故障攻击中故障信息利用率较低的问题,以其高有效性受到密码学专家的广泛关注,利用代数故障攻击成功破解PRESENT[3]、Piccolo[4]和SHA-256[5]等密码算法。

SIMECK[6]算法是一种专门运行于无源RFID标签等硬件资源有限设备的轻量级分组密码,其轮函数采用按位"与""异或"和循环移位操作组合进行加密运算,并且重用轮函数进行密钥扩展,更符合分组密码轻量化设计的要求[7],其中SIMECK32/64硬件实现仅需549GE,优于ISO标准算法PRESENT的1 570GE[8],因此对其安全性分析具有重要意义。现阶段针对SIMECK密码的分析主要集中于差分分析[9-10]和线性分析[11-12],最优能将密钥搜索空间降至2^{29},难以对其实际安全性构成威胁。在故障攻击方面,文献[13]对SIMECK进行差分故障攻击,平均28.32个故障能够恢复末轮16 bit密钥;文献[14]利用代数故障攻击在故障未知模型下需9次故障注入即可恢复完整主密钥。但是,文献[13]仅能恢复末轮16 bit密钥,无法恢复全部主密钥信息,

基金项目:国家自然科学基金(61272491,61309021,61402528)。

作者简介:黄长阳(1994—),男,硕士研究生,主研方向为网络信息安全、代数故障攻击;王 韬,教授、博士、博士生导师;王晓晗,博士研究生;陈庆超、尹世庄,硕士研究生。

收稿日期:2018-06-06 修回日期:2018-07-10 E-mail:13089998121@163.com

对密码安全性威胁较小;文献[14]对故障传播特征分析不深入,无法提取故障确定性传播信息,导致故障确定准确率仅为 82.9%,故障信息利用率较低。

本文对 SIMECK32/64 提出一种改进的代数故障攻击,在第 28 轮左寄存器任意状态注入随机单比特故障,通过分析故障失效情况,提取故障确定传播信息构建差分特性表,并阐述 SIMECK 轮函数设计缺陷判断故障注入位置。

1 相关理论与技术

1.1 SIMECK 算法

本文所使用符号及其含义说明如表 1 所示。

表 1 本文使用符号及其含义

符号	含义
$X<<a$	密码状态 X 循环左移 a bit
&	按位“与”运算
$\oplus$	按位“异或”运算
xL_r^i	密码正确加密第 r 轮左寄存器的第 i 比特
xR_r^i	密码正确加密第 r 轮右寄存器的第 i 比特
$xL_r^{*\,i}$	故障加密第 r 轮左寄存器的第 i 比特
$xR_r^{*\,i}$	故障加密第 r 轮右寄存器的第 i 比特
SK_r^i	第 r 轮加密轮子密钥的第 i 比特
n	SIMECK 加密字长度,$n=16,24,48$

1.1.1 加密轮函数

SIMECK 系列算法加密整体采用平衡型 Feistel 结构,由轮加密函数和密钥扩展策略构成。其中轮函数包括负责增强加密算法非线性的按位“&”运算,提供加密扩散性的“$\oplus$”运算和循环移位操作组合而成,迭代加密结构如图 1 所示。

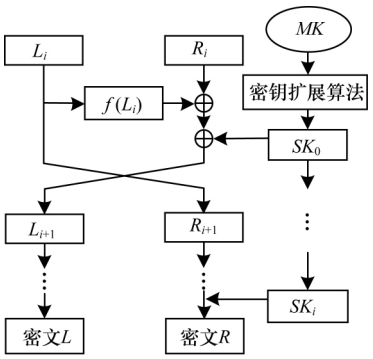

图 1 SIMECK 系列算法加密过程

以 SIMECK32/64 算法为例,其一轮加密流程如下:首先将 32 bit 明文均分成 2 个加密字,其中,低位 16 bit 送入加密左寄存器,高位 16 bit 送入加密右寄存器。启动加密算法,将两寄存器内部状态送入加密轮函数,当 32 轮迭代加密运算后寄存器内部状态即为加密密文。两寄存器内部状态更新方法如式(1)所示。

$$\begin{cases} xR_{r+1}^i = xL_r^i \\ xL_{r+1}^i = f(xL_r^i)\oplus xR_r^i \oplus SK_r^i \end{cases} \tag{1}$$

其中,f 函数运算表达式如式(2)所示。

$$f(x^i)=(x^i\&(x^i<<5))\oplus(x^i<<1) \tag{2}$$

1.1.2 密钥扩展算法

SIMECK 系列密码采用重用轮函数并引入加密参数计算轮密钥的密钥扩展策略。该设计优势在于能够使用已有的轮函数硬件加载密钥扩展算法,无需设计额外的硬件电路,减少资源消耗的同时能利用轮函数的非线性提高密钥扩展算法的安全性,更符合分组密码轻量化设计的要求。

以 SIMECK32/64 算法为例,其轮密钥扩展算法如下:

1)将 64 bit 主密钥(MK)均分成 4 个加密字长度,记为(t_2,t_1,t_0,k_0),其中,k_0 为主密钥的最低 16 bit,t_2 为主密钥的最高16 bit,分别作为密钥扩展寄存器 T_2、T_1、T_0、K_0 的初始状态。

2)将寄存器 T_0 和 K_0 分别作为加密的左寄存器和右寄存器送入加密轮函数,同时引入加密常量 C 和 LFSR 生成的 m-序列增强算法安全性。在每轮运算结束后,将寄存器 K_0 中的结果作为下一轮加密的轮子密钥,更新如式(3)所示。

$$\begin{cases} K_{0r+1}^i = T_{0r}^i \\ T_{2r+1}^i = K_{0r}^i \oplus f(T_{0r}^i)\oplus C\oplus(Z_j)_r \end{cases} \tag{3}$$

在每轮运算后,密钥反馈移位寄存器右移 n 位。在式(3)中,r 为加密轮数,C 为加密常量,$C=2^n-4$,$(Z_j)_r$ 为 m-序列 Z_j 的第 r 比特,各版本密码所用 Z_j 序列的初始状态及反馈特征多项式详见文献[6]。

1.2 攻击原理

代数故障攻击融合典型代数攻击和故障攻击各自的优势,首先构建正确加密过程等效代数方程组,通过激光束照射等方式诱导加密寄存器内部状态发生故障[15],将提取的故障信息并创建其等效方程,最终同正确加密方程联立进行密钥求解。代数故障攻击框架如图 2 所示。本文采用更适合密码方程求解的可满足性(Satisfiability,SAT)问题进行密钥方程求解。

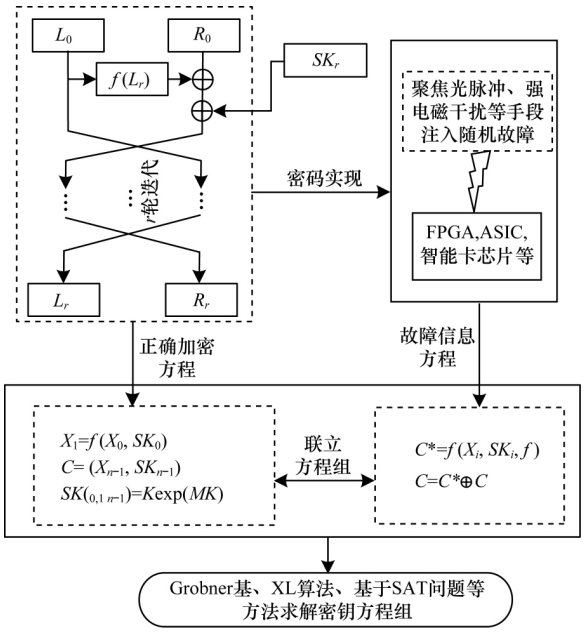

图 2 代数故障攻击框架

2　SIMECK 加密故障扩散分析

本文采用单比特翻转故障模型，即诱导故障后密码中间状态由 1 跳变为 0 跳或由 0 跳变为 1 跳。下文分析故障随迭代加密传播过程中可能出现的失效情况及确定能够扩散的情况。

2.1　单比特故障扩散路径

当诱导左寄存器第 i 比特即 $xL_r^{*\,i}$ 发生单比特翻转故障时，根据 SIMECK32/64 算法轮函数能够计算出第 $r+1$ 轮受 $xL_r^{*\,i}$ 故障影响的密码状态，如式(4)所示。

$$\begin{cases} xR_{r+1}^{*\,i} = xL_r^{*\,i} \\ xL_{r+1}^{*\,i} = (xL_r^{*\,i} \& xL_r^{(i+5)\%16}) \oplus xL_r^{(i+1)\%16} \oplus xR_r^i \oplus SK_r^i \\ xL_{r+1}^{*\,(i+15)\%16} = (xL_r^{(i+15)\%16} \& xL_r^{(i+4)\%16}) \oplus xL_r^{*\,i} \oplus \\ \qquad xR_r^{(i+15)\%16} \oplus SK_r^{(i+15)\%16} \\ xL_{r+1}^{*\,(i+11)\%16} = (xL_r^{(i+11)\%16} \& xL_r^{*\,i}) \oplus xL_r^{(i+12)\%16} \oplus \\ \qquad xR_r^{(i+11)\%16} \oplus SK_r^{(i+11)\%16} \end{cases} \tag{4}$$

因此，当故障诱导在 $xL_r^{*\,i}$ 时，影响下一轮中间状态为 $xR_{r+1}^{*\,i}$、$xL_{r+1}^{*\,i}$、$xL_{r+1}^{*\,(i+11)\%16}$ 和 $xL_{r+1}^{*\,(i+15)\%16}$。同理，当故障诱导在右寄存器第 i 比特 $xR_r^{*\,i}$ 时，影响下一轮中间状态为 $xL_{r+1}^{*\,i}$。以注入在 xL_r^4 和 xR_r^7 位置为例，故障向下一轮扩散的情况如图 3 所示。

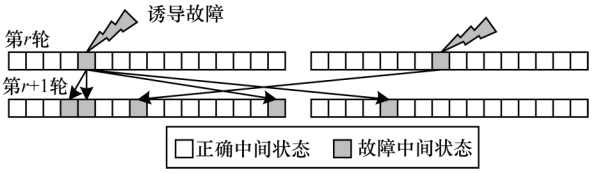

图 3　单比特故障向下一轮扩散的情况

由上述分析可得，左寄存器内故障可影响下一轮较多状态，但同时在右寄存器第 i 比特 $xR_r^{*\,i}$ 注入故障等效于在左寄存器 $xL_{r+1}^{*\,i}$ 注入故障。

2.2　故障扩散失效情况

故障实际传播过程与理论扩散路径并不完全相同，这是因为故障在随 SIMECK 算法迭代加密扩散过程中可能出现故障失效情况。

2.2.1　多个故障碰撞导致故障失效的情况

当多个故障同时扩散到密码中间状态同一比特位即多个故障发生碰撞时，可能会出现多个故障效果相互抵消的情况，导致故障加密结果与正确加密结果相同即故障扩散失效，如图 4 所示。

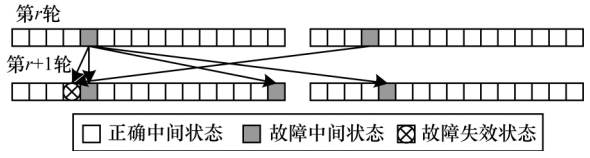

图 4　故障碰撞导致故障传播失效的情况

当第 r 轮中间状态发生故障位置为 $xL_r^{*\,4}$ 和 $xL_r^{*\,3}$ 时，经过一轮加密后 2 个故障同时影响到 $xL_{r+1}^{*\,3}$ 位置，根据轮函数计算可得:

$$\begin{aligned} xL_{r+1}^{*\,3} &= (xL_r^3 \& xL_r^8) \oplus xL_r^{*\,4} \oplus xR_r^{*\,3} \oplus SK_r^3 = \\ &\quad (xL_r^3 \& xL_r^8) \oplus (xL_r^4 \oplus 1) \oplus \\ &\quad (xR_r^3 \oplus 1) \oplus SK_r^3 = xL_{r+1}^3 \end{aligned} \tag{5}$$

若故障状态 $xL_{r+1}^{*\,3}$ 与正确状态 xL_{r+1}^3 相等，则故障效果失效。

2.2.2　加密部件数学属性导致故障失效的情况

SIMECK 算法轮函数中“&”运算的数学特性可能会导致故障扩散出现失效的情况。假设进行“&”运算的双方为 xL_r^i 和 $xL_r^{(i+5)\%16}$，经过运算后故障扩散情况如表 2 所示。

表 2　“&”运算导致故障失效的情况

xL_r^i	$xL_r^{(i+5)\%16}$	正确运算结果	故障位置	故障加密结果	故障效果
0	0	0	xL_r^i	0	失效
0	1	0	xL_r^i	1	成功
1	0	0	xL_r^i	0	失效
1	1	1	xL_r^i	0	成功

分析表 2 发现，当故障注入在 xL_r^i 时，经过“&”运算后故障是否失效仅与“&”右侧状态 $xL_r^{(i+5)\%16}$ 值有关，即若 $xL_r^{(i+5)\%16}=0$，则故障扩散失效; 否则故障能够成功扩散至下一轮状态。

2.3　故障确定扩散的情况

本节分析故障确定扩散的情况，包括故障确定能够扩散到的密码状态和故障确定不会影响到的密码状态，并依此提取故障确定性扩散特征。

2.3.1　故障确定扩散到的中间状态

在故障传播过程中，既不会发生故障相遇碰撞同时又不涉及“&”运算导致故障失效时，就能够确定故障扩散具体的比特位，即当故障仅通过“⊕”运算或“=”运算向下一轮扩散，且不发生故障相遇碰撞时，故障肯定能够扩散成功。

假设故障注入在第 r 轮 $xL_r^{*\,i}$ 位置，根据式(4)可得故障由 $xL_r^{*\,i}$ 分别依靠“⊕”运算和“=”运算扩散至 $xL_{r+1}^{*\,(i+15)\%16}$ 和 $xR_{r+1}^{*\,i}$ 且没有发生故障碰撞，因此故障确定能够扩散成功。

同理，可计算得到第 $r+2$ 轮故障确定扩散至 $xL_{r+2}^{*\,(i+14)\%16}$ 和 $xR_{r+2}^{*\,(i+15)\%16}$，第 $r+3$ 轮故障确定传播至 $xL_{r+3}^{*\,(i+13)\%16}$ 和 $xR_{r+3}^{*\,(i+14)\%16}$。以故障注入在 $xL_r^{*\,0}$ 位置为例，故障确定扩散情况如图 5 所示。

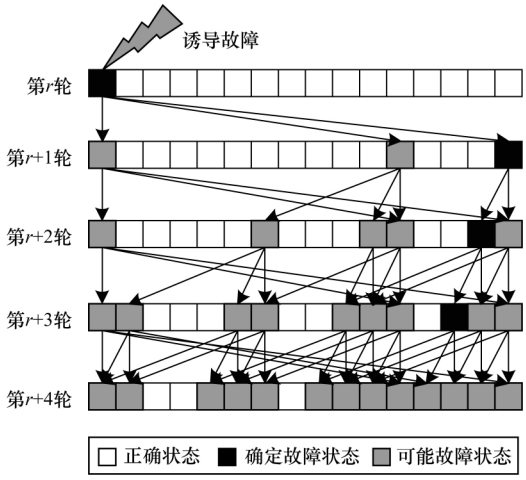

图 5 故障确定扩散的情况

当故障在第 r 轮 ~ 第 $r+3$ 轮扩散时，均有确定故障扩散状态，但是当传播至第 $r+4$ 轮时，所有可能发生故障的状态均有可能因为发生故障碰撞导致故障失效，因此第 $r+4$ 轮中没有故障能够确定扩散到的状态。

2.3.2 故障确定不会扩散到的中间状态

在去除 2.1 节中故障可能扩散的位置和 2.3.1 节中故障确定扩散成功位置的基础上，即可分析得出故障确定不会扩散，即正确/故障加密状态差分为 0 的位置。假设故障注入在第 r 轮 $xL_r^{*\,i}$ 位置，左寄存器内部故障确定不会扩散至的位置如表 3 所示。根据式(4)可得，第 $r+1$ 轮右寄存器内部故障确定不会扩散的状态与第 r 轮左寄存器情况相同，不再赘述。

表 3 左寄存器内确定不会发生故障的状态

轮数	左寄存器内部故障确定不会扩散到的中间状态
$r+1$	xL_{r+1}^{j}，$0\leqslant j\leqslant 15$，$j\neq i$，$(i+11)\%16$，$(i+15)\%16$
$r+2$	xL_{r+2}^{j}，$0\leqslant j\leqslant 15$，$j\neq i$，$(i+6)\%16$，$(i+10)\%16$，$(i+11)\%16$，$(i+14)\%16$，$(i+15)\%16$
$r+3$	$xL_{r+3}^{(i+2)\%16}$，$xL_{r+3}^{(i+3)\%16}$，$xL_{r+3}^{(i+4)\%16}$，$xL_{r+3}^{(i+7)\%16}4$，$xL_{r+3}^{(i+8)\%16}$，$xL_{r+3}^{(i+12)\%16}$
$r+4$	$xL_{r+3}^{(i+2)\%16}$，$xL_{r+3}^{(i+3)\%16}$，$xL_{r+3}^{(i+7)\%16}$
$r+5$	$xL_{r+3}^{(i+2)\%16}$

3 改进的 SIMECK32/64 代数故障攻击

准确判定发生故障的状态位置是利用故障信息的前提，如果故障位置判定错误，则会导致故障信息等效方程创建错误致使故障攻击失败。针对以上故障失效导致故障攻击成功率下降的问题，本文提出 2 种方法提高故障定位成功率，并给出通用的加密等效方程组创建方法。

3.1 故障注入位置计算

分析 SIMECK32/64 算法轮函数加密扩散缺陷，根据获取正确密文和故障密文即可确定 $r-2$ 轮左寄存器故障诱导位置和 $r-3$ 轮右寄存器故障诱导位置。

根据式(1)和式(2)运算可得：

$$xL_{r+1}^{i}=f(xR_{r+1}^{i})\oplus xL_{r-1}^{i}\oplus sk_{r}^{i} \tag{6}$$

即：

$$xL_{r}^{i}=f(xR_{r}^{i})\oplus xL_{r-2}^{i}\oplus sk_{r-1}^{i}$$

所以：

$$xL_{r-2}^{i}=xL_{r}^{i}\oplus f(xR_{r}^{i})\oplus sk_{r-1}^{i} \tag{7}$$

假设故障注入在 $xL_{r-2}^{*\,i}$ 状态，即 $xL_{r-2}^{*\,i}=xL_{r-2}^{i}\oplus 1$，则可得：

$$xL_{r-2}^{*\,i}=xL_{r}^{*\,i}\oplus f(xR_{r}^{*\,i})\oplus sk_{r-1}^{i} \tag{8}$$

根据式(7)和式(8)，可计算得出 $r-2$ 轮左寄存器内部正确加密和故障加密状态差分 ΔxL_{r-2}^{i} 如式(9)所示。

$$\begin{aligned}\Delta xL_{r-2}^{i}=&xL_{r-2}^{i}\oplus xL_{r-2}^{*\,i}=xL_{r}^{i}\oplus f(xR_{r}^{i})\oplus\\&sk_{r}^{i}\oplus xL_{r}^{*\,i}\oplus f(xR_{r}^{*\,i})\oplus sk_{r}^{i}=xL_{r}^{i}\oplus\\&xL_{r}^{*\,i}\oplus(xR_{r}^{i}\&(xR_{r}^{i}<<5))\oplus(xR_{r}^{i}<<1)\oplus\\&(xR_{r}^{*\,i}\&(xR_{r}^{*\,i}v5))\oplus(xR_{r}^{*\,i}<<1)\end{aligned} \tag{9}$$

根据式(1)和式(9)可计算得第 $r-3$ 轮右寄存器内部正确加密与故障加密状态差分 ΔxL_{r-1}^{i} 如式(10)所示。

$$\Delta xR_{r-1}^{i}=xR_{r-1}^{i}\oplus xR_{r-1}^{*\,i}=xL_{r-2}^{i}\oplus xL_{r-2}^{*\,i}=\Delta xL_{r-2}^{i} \tag{10}$$

当 $r=32$ 时，xL_{32}^{i} 和 xR_{32}^{i} 是正确加密密文，$xL_{32}^{*\,i}$ 和 $xR_{32}^{*\,i}$ 是故障加密密文，即式(9)中等号右边所有变量均为已知，可准确计算出第 30 轮左寄存器内故障注入位置。若通过正确密文和故障密文计算得到 $\Delta xL_{r-2}^{i}=1$，则表明 ΔxL_{r-2}^{i} 为故障注入位置，若 $\Delta xL_{r-2}^{i}=0$，则 ΔxL_{r-2}^{i} 不是故障注入位置。同理，通过正确密文和故障密文能够计算出第 31 轮右寄存器内故障注入位置。

3.2 故障注入位置判定

根据 2.3 节分析所得结论，当故障注入在第 29 轮 $xL_{29}^{*\,i}$ 位置时，可准确计算得出 xL_{32}^{i} 和 xR_{32}^{i}，即故障密文输出中一定不会发生故障的状态位置和确定发生故障的状态位置。通过提取上述故障确定性扩散特征，构建故障差分特性表判定故障注入位置。

设正确密文与故障密文差分值为 ΔC，则：

$$\Delta C_j=\begin{cases}xL_{32}^{j}\oplus xL_{32}^{*\,j}, & 0\leqslant j<16\\ xR_{32}^{j-16}\oplus xR_{32}^{*\,j-16}, & 16\leqslant j<32\end{cases} \tag{11}$$

基于上述理论分析，遍历计算故障注入在 $xL_{29}^{*\,i}$ $(0\leqslant i<16)$ 时密文差分 $\Delta C_j(0\leqslant j<32)$ 的值，若故障确定扩散至密文第 j 比特位，则 $\Delta C_j=1$；若故障确定没有扩散至密文第 j 比特位，则 $\Delta C_j=0$；其余故障不确定扩散情况的位置 k 时，令 $\Delta C_k=x$，$0\leqslant k<32$，$k\neq i,j$。基于上述理论，构建故障注入在 SIMECK32/64 算法第 29 轮时正确/故障密文差分 $(\Delta C_j, j=0,1,\cdots,30)$ 特性，如表 4 所示。

表 4　故障注入在第 29 轮时正确/故障密文差分特性表

故障注入位置	正确/故障密文差分值
$xL_{29}^{*\,0}$	x,x,0,0,0,x,x,0,0,x,x,x,0,1,x,x,x,0,0,0,0,0,x,0,0,0,x,x,0,0,1,x
$xL_{29}^{*\,1}$	x,x,x,0,0,0,x,x,0,0,x,x,x,0,1,x,x,x,0,0,0,0,0,x,0,0,0,x,x,0,0,1
$xL_{29}^{*\,2}$	x,x,x,x,0,0,0,x,x,0,0,x,x,x,0,1,1,x,x,0,0,0,0,0,x,0,0,0,x,x,0,0
$xL_{29}^{*\,3}$	1,x,x,x,x,0,0,0,x,x,0,0,x,x,x,0,0,1,x,x,0,0,0,0,x,0,0,0,x,x,0
$xL_{29}^{*\,4}$	0,1,x,x,x,x,0,0,0,x,x,0,0,x,x,x,0,0,1,x,x,0,0,0,0,x,0,0,0,x,x
$xL_{29}^{*\,5}$	x,0,1,x,x,x,x,x,0,0,0,x,x,0,0,x,x,x,0,0,1,x,x,0,0,0,0,x,0,0,0,x
$xL_{29}^{*\,6}$	x,x,0,1,x,x,x,x,0,0,0,x,x,0,0,x,x,x,0,0,1,x,x,0,0,0,0,0,x,0,0,0
$xL_{29}^{*\,7}$	x,x,x,0,1,x,x,x,x,0,0,0,x,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0,0,x,0,0
$xL_{29}^{*\,8}$	0,x,x,x,0,1,x,x,x,x,0,0,0,x,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0,0,x,0
$xL_{29}^{*\,9}$	0,0,x,x,x,0,1,x,x,x,x,0,0,0,x,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0,0,x
$xL_{29}^{*\,10}$	x,0,0,x,x,x,0,1,x,x,x,x,0,0,0,x,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0,0
$xL_{29}^{*\,11}$	x,x,0,0,x,x,x,0,1,x,x,x,x,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0
$xL_{29}^{*\,12}$	0,x,x,0,0,x,x,x,0,1,x,x,x,x,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x,0,0,0
$xL_{29}^{*\,13}$	0,0,x,x,0,0,x,x,x,0,1,x,x,x,x,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x,0,0
$xL_{29}^{*\,14}$	0,0,0,x,x,0,0,x,x,x,0,1,x,x,x,x,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x,0
$xL_{29}^{*\,15}$	x,0,0,0,x,x,0,0,x,x,x,0,1,x,x,x,0,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x

在表 4 中，第 i 行表示故障注入在 $xL_{29}^{*\,i}$ 时，第 j 列表示正确/故障密文差分第 j 比特位 ΔC_j。差分特性表中 a [ ] j] 等于“0”或“1”表明当故障注入在 $xL_{29}^{*\,i}$ 时，ΔC_j 的值始终为“0”或“1”。基于上述理论，攻击者通过计算正确/故障密文差分 ΔC，并与表 4 进行匹配即能够确定故障注入状态位置。为验证上述理论的正确性，本文进行左寄存器仿真故障注入实验，共进行 10 000 次，其中正确判定故障注入位置10 000 次，成功率为 100%。

3.3　故障注入位置融合判定

上述 2 种方法虽然能够正确计算出故障注入位置，但是由于故障注入轮数较浅，没有涉及到全部主密钥信息，因此仅能够恢复部分密钥。本节将融合 2 种方法，利用轮函数加密扩散缺陷加深差分特性表构建深度，以达到利用更多故障信息恢复全部密钥的目的。

1) 根据 3.1 节分析轮函数加密扩散缺陷以及所获正确密文和故障密文，通过式(9) 能够遍历计算得出 $\Delta xL_{30}^{i}(0\leqslant i<16)$。

2) 利用 3.2 节方法能够构建故障注入在第 $r=27$ 轮左寄存器时，第 30 轮中间状态差分特性表，由于步骤 1) 中计算得出 16 bit $\Delta xL_{30}^{i}(0\leqslant i<16)$，因此仅需构建第 30 轮左寄存器内部差分特性表。

3) 将步骤 1) 中所得 16 bit $\Delta xL_{30}^{i}(0\leqslant i<16)$ 与步骤 2) 中所构建的第 30 轮左寄存器内部状态差分特性表匹配，即能够确定第 $r=27$ 轮左寄存器内故障注入位置。

由于差分特性表中的故障确定状态较少，因此实际攻击中确定故障成功率为89.4%。为能够在差分特性表中有更多确定信息，将步骤 2) 改进为在第 28 轮左寄存器 xL_{28}^{i} 注入单比特随机故障，通过计算 3 轮故障确定信息构建第 30 轮中间状态差分($xL_{28}^{i}, i=0,1,\cdots,15$) 特性，如表 5 所示。

表 5　第 30 轮中间状态差分特性表

故障注入位置	第 30 轮中间状态差分值
$xL_{28}^{*\,0}$	x,0,0,0,0,0,x,0,0,0,x,x,0,0,1,x
$xL_{28}^{*\,1}$	x,x,0,0,0,0,0,x,0,0,0,x,x,0,0,1
$xL_{28}^{*\,2}$	1,x,x,0,0,0,0,0,x,0,0,0,x,x,0,0
$xL_{28}^{*\,3}$	0,1,x,x,0,0,0,0,0,x,0,0,0,x,x,0
$xL_{28}^{*\,4}$	0,0,1,x,x,0,0,0,0,0,x,0,0,0,x,x
$xL_{28}^{*\,5}$	x,0,0,1,x,x,0,0,0,0,0,x,0,0,0,x
$xL_{28}^{*\,6}$	x,x,0,0,1,x,x,0,0,0,0,0,x,0,0,0
$xL_{28}^{*\,7}$	0,x,x,0,0,1,x,x,0,0,0,0,0,x,0,0
$xL_{28}^{*\,8}$	0,0,x,x,0,0,1,x,x,0,0,0,0,0,x,0
$xL_{28}^{*\,9}$	0,0,0,x,x,0,0,1,x,x,0,0,0,0,0,x
$xL_{28}^{*\,10}$	x,0,0,0,x,x,0,0,1,x,x,0,0,0,0,0
$xL_{28}^{*\,11}$	0,x,0,0,0,x,x,0,0,1,x,x,0,0,0,0
$xL_{28}^{*\,12}$	0,0,x,0,0,0,x,x,0,0,1,x,x,0,0,0
$xL_{28}^{*\,13}$	0,0,0,x,0,0,0,x,x,0,0,1,x,x,0,0
$xL_{28}^{*\,14}$	0,0,0,0,x,0,0,0,x,x,0,0,1,x,x,0
$xL_{28}^{*\,15}$	0,0,0,0,0,x,0,0,0,x,x,0,0,1,x,x

在表 5 中的差分值确定状态较多，能够有效提高故障注入位置判定准确率。因此，本文选取第 28 轮左寄存器注入随机单比特故障，计算故障注入位置的算法具体如下:

输入　正确密文 C 和故障密文 C^*

输出　第 28 轮故障注入索引值 $index$

初始化　加密字 $n\leftarrow 16$; σ [n] [n] $\leftarrow 0$; //n 行×n 列差分//特性表

1. 随机生成明文 P 和主密钥 MK，正确运行 SIMECK32/64 算法得到密文 C;

2. 在同一明文 P 和密钥 MK 下重启加密，当加密运行至第 28 轮时，向左寄存器内部随机注入单比特故障，加密至故障密文输出 C^*;

for i = 0 to n - 1

for j = 0 to n - 1

根据 2.3 节原理计算 xL_{30}^{j} 是否为故障传播状态

if 故障确定能传播至 xL_{30}^{j} then

σ[i][j] = 1;

else if 故障确定不会传播至 xL_{30}^{j} then

σ[i][j] = 1;

else

σ[i][j] = x;

end if

end for

for i = 0 to n - 1

$\Delta xL_{30}[i] = xL_{32}^{i} \oplus xL_{32}^{*i} \oplus (xR_{32}^{i} \& (xR_{32}^{i} << 5)) \oplus (xR_{32}^{i} << 1) \oplus (xR_{32}^{*i} \& (xR_{32}^{*i} << 5)) \oplus (xR_{32}^{*i} << 1)$

end for

if ΔxL_{30}^{i} ($0 \leqslant i < 16$) 中 '0' 和 '1' 的位置与差分特性表中第 i 行匹配

index = i;

end if

为充分验证算法的有效性，进行 160 000 次仿真随机单比特故障注入实验，其中正确判定故障注入位置 159 368 次，成功率为 99.61%。

3.4 等效代数方程自动化创建

3.4.1 基于正确加密的等效方程创建

根据式(1)和式(2)可创建加密轮函数等效方程如式(12)所示。

$$\begin{cases} xR_{r+1}^{i} \oplus xL_{r}^{i} = 0 \\ xL_{r+1}^{i} \oplus (xL_{r}^{i} \& xL_{r}^{(i+5)\%16}) \oplus \\ \qquad xL_{r}^{(i+1)\%16} \oplus xR_{r}^{i} \oplus SK_{r}^{i} = 0 \end{cases} \tag{12}$$

其中，r 为加密轮数，$0 \leqslant r < 32$，i 为中间状态位置，$0 \leqslant i < 16$。

同理，密钥扩展算法等效代数方程组构建如式(13)所示。

$$SK_{r}^{i} = SK_{r-4}^{i} \oplus (SK_{r-3}^{i} \& SK_{r-3}^{(i+5)\%16}) \oplus SK_{r-3}^{(i+1)\%16} \oplus C \oplus Z_{j} \tag{13}$$

由于 SIMECK32/64 前 4 轮密钥是由 64 bit 主密钥 MK 直接分解而来，因此只需创建第 4 轮～第 31 轮的轮密钥等效方程。其中，r 为加密轮数，$4 \leqslant r < 32$，i 为 SK_r 第 i 比特，$0 \leqslant i < 16$。

3.4.2 基于故障信息的等效方程创建

故障信息利用可分为故障加密等效方程和故障传播信息等效方程两部分，其中故障加密等效方程创建原理及形式与式(12)类似，下文主要阐述如何自动化创建故障扩散信息等效方程。

故障扩散信息等效方程可分为 3 类：故障确定能扩散到的状态方程，故障可能扩散到的状态方程以及故障确定不会扩散到的状态方程。

随机单比特故障注入在左寄存器内 xL_{28}^{i} 状态，则故障确定能传播至的状态等效方程如式(14)所示。

$$\begin{cases} xL_{28}^{*j} \oplus xL_{28}^{j} \oplus 1 = 0, \ j = i \\ xL_{29}^{*(j+15)\%16} \oplus xL_{29}^{(j+15)\%16} \oplus 1 = 0, \ j = i \\ xL_{30}^{*(j+14)\%16} \oplus xL_{30}^{(j+14)\%16} \oplus 1 = 0, \ j = i \\ xL_{31}^{*(j+13)\%16} \oplus xL_{31}^{(j+13)\%16} \oplus 1 = 0, \ j = i \end{cases} \tag{14}$$

故障可能扩散至的状态等效方程如式(15)所示。

$$\begin{cases} xL_{29}^{*j} \oplus (xL_{28}^{*i} \& xL_{28}^{(i+5)\%16}) \oplus xL_{28}^{(i+1)\%16} \oplus \\ \qquad xR_{28}^{i} \oplus SK_{28}^{i} = 0 \\ xL_{29}^{*(i+11)\%16} \oplus (xL_{28}^{(i+11)\%16} \& xL_{28}^{*i}) \oplus xL_{28}^{(i+12)\%16} \oplus \\ \qquad xR_{28}^{(i+11)\%16} \oplus SK_{28}^{(i+11)\%16} = 0 \\ xL_{30}^{*i} \oplus (xL_{29}^{*i} \& xL_{29}^{(i+5)\%16}) \oplus xL_{28}^{(i+1)\%16} \oplus xR_{29}^{*i} \oplus \\ \qquad SK_{29}^{i} = 0 \\ xL_{30}^{(i+6)\%16} \oplus (xL_{29}^{(i+6)\%16} \& xL_{29}^{*(i+11)\%16}) \oplus \\ \qquad xL_{29}^{(i+7)\%16} \oplus xR_{29}^{(i+6)\%16} \oplus SK_{29}^{(i+6)\%16} = 0 \\ xL_{30}^{*(i+10)\%16} \oplus (xL_{29}^{(i+10)\%16} \& xL_{29}^{*(i+15)\%16}) \oplus \\ \qquad xL_{29}^{*(i+11)\%16} \oplus \\ \qquad xR_{29}^{(i+10)\%16} \oplus SK_{29}^{(i+10)\%16} = 0 \\ xL_{30}^{*(i+11)\%16} \oplus (xL_{30}^{*(i+11)\%16} \& xL_{29}^{*i}) \oplus \\ \qquad xL_{29}^{(i+12)\%16} \oplus \\ \qquad xR_{29}^{(i+1)\%16} \oplus SK_{29}^{(i+11)\%16} = 0 \\ xL_{30}^{*(i+15)\%16} \oplus (xL_{29}^{*(i+15)\%16} \& xL_{29}^{(i+4)\%16}) \oplus \\ \qquad xL_{29}^{*i} \oplus xR_{29}^{(i+15)\%16} \oplus \\ \qquad SK_{29}^{(i+15)\%16} = 0 \end{cases} \tag{15}$$

同理，可创建后续加密轮等效方程，不再赘述。

故障确定不会扩散至状态等效方程，如式(16)所示。

$$xL_{r}^{*j} \oplus xL_{r}^{j} = 0, 0 \leqslant j < 16 \tag{16}$$

其中，当 $r = 28$ 时，$0 \leqslant j < 16$ 且 $j \neq i$；当 $r = 29$ 时，$0 \leqslant j < 16$ 且 $j \neq i$，$(i+11)\%16$，$(i+15)\%16$；当 $r = 30$ 时，$0 \leqslant j < 16$ 且 $j \neq i$，$(i+6)\%16$，$(i+10)\%16$，$(i+11)\%16$，$(i+14)\%16$，$(i+15)\%16$；当 $r = 31$ 时，$j = (i+2)\%16$，$(i+3)\%16$，$(i+4)\%16$，$(i+7)\%16$，$(i+8)\%16$，$(i+12)\%16$；当 $r = 32$ 时，$j = (i+2)\%16$，$(i+3)\%16$，$(i+7)\%16$。至此，所有方程均已创建完毕，且不需要为 $i = 0, 1, \cdots, 15$ 分别创建方程，自动化程度更高。

4 实验与结果分析

在普通 PC 机(CPU：Intel(R) Core(TM) i7-4790 @ 3.20 GHz，内存 8 GB，64 位 Windows 7 系统)上，使用 C++ 语言编程仿真故障注入过程，使用 CryptoMinisat 解析器(4.4 版本)求解密钥方程。

4.1 改进的 SIMECK32/64 代数故障攻击实例

改进的 SIMECK32/64 代数故障攻击实例具体步骤如下：

1）开始随机生成 32 bit 明文 mP 为 0x a78b 66ae，64 bit 主密钥 MK 为 0x 8df3 629d 2671 cb13，运行加密算法得到正确密文 C 为 0x ea6f 0387，并使用 3.4.1 节方法创建正确加密等效代数方程组。

2）同一明文和密钥下重启加密算法，向加密第 28 轮左寄存器任意状态注入随机单比特故障，设定故障样本量为 8；利用 3.3 节原理判定故障注入索引值，并利用 3.4.2 节方法创建故障信息方程组。

3）使用 CryptoMinisat 解析器求解步骤 1）和步骤 2）所构建方程组。64 bit 主密钥编号为 46～109，其中，正数表示该编号所代表变量值为 1，负数表示该编号变量所代表的变量值为 0。求解出的密钥信息为：1000 1101 1111 0011 0110 0010 1001 1101 0010 0110 0111 0001 1100 1011 0001 0011，转换为十六进制为 0x 8df3 629d 2671 cb13，与正确密钥信息一致，攻击成功。

4.2　结果分析

为验证实验结果有效性，本文在故障样本量 $N=8$ 下进行 100 次攻击实验，结果如图 6 所示。

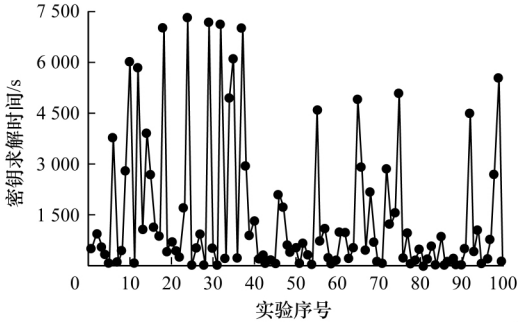

图 6　故障样本量 $N=8$ 时 100 组实验攻击结果

从图 6 发现求解时间具有抖动性，分析其原因为：1）不同样本之间故障失效位置和数量有差异，故障失效状态位置多的样本有效故障信息方程较少，导致求解时间增加；2）CryptoMinisat 解析器自身稳定性不足，当方程组数量过多时可能导致求解时间具有波动性。因此为减少求解时间抖动性对实验结果的影响，本文对每种故障样本量下进行 100 组攻击实验，取平均值作为对应的密钥求解时间。

为充分验证本文方法的正确性，对不同故障样本量 N 下进行攻击实验，每个故障样本量均进行 100 次攻击实验取平均值作为结果。密钥求解时间随故障注入数量的变化情况如图 7 所示。

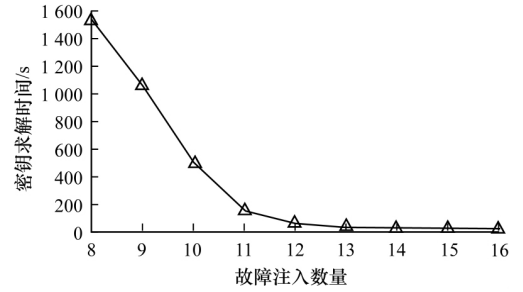

图 7　密钥求解时间随故障注入数量的变化情况

最少 8 次故障注入即可破解完整 64 bit 主密钥，平均求解时间约为 1 532 s。随着故障样本量增大，密钥求解时间逐渐减小，并且故障样本量越少，求解时间减小趋势越明显；当故障样本量增加至 15 时，密钥求解时间最短，约为 18.43 s。文献［13］差分故障攻击仅能破解最后一轮 16 bit 轮密钥，而本文方法能够将故障注入更深的加密轮，从而使得故障信息涉及全部主密钥，并能够恢复完整 64 bit 主密钥信息，且所需故障注入数量更少，对密码安全性威胁更大。与文献［14］代数故障攻击相比，本文通过深入分析故障确定传播状态，结合 SIMECK 算法轮函数加密扩散缺陷，将故障注入位置判定成功率由 82.9% 提高至 99.61%，并且本文在利用故障信息时，增加了故障确定状态等效方程，因此能够创建更多的有效故障信息方程，将所需故障注入数量由 9 个减少至 8 个，故障信息利用率更高。文献［14］需要离线构建故障注入在第 0 比特～第 15 比特全部方程组再进行选择，而本文通过创建通用方程，判定故障注入位置后进行动态创建方程，自动化程度更高。

5　结束语

本文针对轻量级分组密码 SIMECK32/64 提出一种改进的代数故障攻击方法，通过分析轮函数扩散缺陷和提取故障确定传播状态构建差分特性表，将故障攻击成功率提高至 99.61%，且仅需 8 次故障注入即可恢复完整 64 bit 主密钥信息。下一步可将本文方法扩展至其他轻量级分组密码算法，并使故障注入更深轮，提高故障信息利用率。

参考文献

[1] COURTOIS N，WARE D，JACKSON K. Fault-algebraic attacks on inner rounds of DES［EB/OL］.［2018-06-05］. https://www.researchgate.net/publication/264885998_Fault-Algebraic_Attacks_on_Inner_Rounds_of_DES.

[2] BIHAM E，SHAMIR A. Differential fault analysis of secret key cryptosystems［C］//Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology. New York，USA：ACM Press，1997：513-525.

[3] 吴克辉，赵新杰，王韬，等. PRESENT 密码代数故障攻击［J］. 通信学报，2012，33（8）：85-92.

[4] 赵新杰，郭世泽，王韬，等. Piccolo 密码代数故障分析研究［J］. 计算机学报，2013，36（4）：882-894.

[5] HAO Ronglin，LI Bao，MA Bingke，et al. Algebraic fault attack on the SHA-256 compression function［J］. International Journal of Research in Computer Science，2014，4（2）：1-7.

[6] YANG Gangqiang，ZHU Bo，SUDER V，et al. The SIMECK family of lightweight block ciphers［C］//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin，Germany：Springer，2015：307-329.

（下转第 21 页）

[5] WANG Jun, WANG Shaoping, WANG Xingjian, et al. Active fault tolerant control for vertical tail damaged aircraft with dissimilar redundant actuation system [J]. Chinese Journal of Aeronautics, 2016, 29(5): 1313-1325.
[6] OKHRAVI H, HOBSON T, BIGELOW D, et al. Finding focus in the blur of moving-target techniques [J]. IEEE Security and Privacy, 2014, 12(2): 16-26.
[7] 邬江兴. 网络空间拟态防御导论 [M]. 北京: 科学出版社, 2017.
[8] LOPES R V, MENASCE D. A taxonomy of job scheduling on distributed computing systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3412-3428.
[9] KALEEM R, BARIK R, SHPEISMAN T, et al. Adaptive heterogeneous scheduling for integrated GPUs [C]//Proceedings of the 23rd International Conference on Parallel Architecture and Compilation Techniques. Washington D. C., USA: IEEE Press, 2014: 151-162.
[10] LIU Jinwei, SHEN Haiying. Dependency-aware and resource-efficient scheduling for heterogeneous jobs in clouds [C]//Proceedings of 2016 IEEE International Conference on Cloud Computing Technology and Science. Washington D. C., USA: IEEE Press, 2016: 110-117.
[11] DAS A, KUMAR A, VEERAVALLI B. Reliability and energy-aware mapping and scheduling of multimedia applications on multiprocessor systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(3): 869-884.
[12] 彭浩，陆阳，孙峰，等. 副版本不可抢占的全局容错调度算法 [J]. 软件学报, 2016, 27(12): 3158-3171.
[13] 朱晓敏，祝江汉，马满好. 异构集群系统中具有 QoS 需求的实时任务容错调度 [J]. 软件学报, 2011, 22(7): 1440-1456.
[14] EDWARD N, ELCOCK J. An efficient task scheduling algorithm for heterogeneous multiprocessing environments [C]//Proceedings of International Conference on Information and Computer Technologies. Washington D. C., USA: IEEE Press, 2018: 101-106.
[15] COLOM J F, GIL D, MORD H, et al. Scheduling framework for distributed intrusion detection systems over heterogeneous network architectures [J]. Journal of Network and Computer Applications, 2018, 108: 76-86.
[16] 邬江兴，李军飞. 一种异构功能等价体调度装置及方法: 中国, CN106161417A [P]. 2016-11-23.
[17] 刘勤让，林森杰，顾泽宇. 面向拟态安全防御的异构功能等价体调度算法 [J]. 通信学报, 2018, 39(7): 188-198.
[18] 仝青，张铮，张为华，等. 拟态防御 Web 服务器设计与实现 [J]. 软件学报, 2017, 28(4): 883-897.
[19] TWU P, MOSTOFI Y, EGERSTEDT M. A measure of heterogeneity in multi-agent systems [C]//Proceedings of American Control Conference. Washington D. C., USA: IEEE Press, 2014: 3972-3977.
[20] RAO C R. Diversity and dissimilarity coefficients: a unified approach [J]. Theoretical Population Biology, 1982, 21(1): 24-43.
[21] YOUNIS A, MALAIYA Y K, RAY I. Evaluating CVSS base score using vulnerability rewards programs [C]//Proceedings of IFIP International Conference on ICT Systems Security and Privacy Protection. Berlin, Germany: Springer, 2016: 62-75.
[22] 张铮，马博林，邬江兴. Web 服务器拟态防御原理验证系统测试与分析 [J]. 信息安全学报, 2017, 2(1): 13-28.
[23] PETER L J, HULL R. The peter principle [M]. London, UK: Souvenir Press, 1969.
[24] 姚文斌，杨孝宗. 相异性软件组件选择算法设计 [J]. 哈尔滨工业大学学报, 2003, 35(3): 261-264.

编辑　索书志

(上接第 13 页)

[7] BIRYUKOV A, PERRIN L P. State of the art in lightweight symmetric cryptography [EB/OL]. [2018-06-05]. https://eprint.iacr.org/2017/511.
[8] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: an ultra-lightweight block cipher [C]//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Berlin, Germany: Springer, 2007: 450-466.
[9] QIAO Kexin, HU Lei, SUN Siwei. Differential security evaluation of SIMECK with dynamic key-guessing techniques [C]//Proceedings of International Conference on Information Systems Security and Privacy. Berlin, Germany: Springer, 2015: 74-84.
[10] 陈彦琴，张文英. SIMECK32/64 算法的不可能差分分析 [J]. 计算机工程, 2017, 43(4): 141-144.
[11] ZHANG Kai, GUAN Jie, HU Bin, et al. Security evaluation on SIMECK against zero-correlation linear cryptanalysis [EB/OL]. [2018-06-05]. https://eprint.iacr.org/2015/911.
[12] BAGHERI N. Linear cryptanalysis of reduced-round SIMECK variants [C]//Proceedings of INDOCRYPT'15. Berlin, Germany: Springer, 2015: 140-152.
[13] SARASWAT V. Differential fault attack on SIMECK [C]//Proceedings of Workshop on Cryptography and Security in Computing Systems. New York, USA: ACM Press, 2016: 45-48.
[14] NALLA V, SAHU, R A, SARASWAT V. Differential fault attack on SIMECK [C]//Proceedings of Workshop on Cryptography and Security in Computing Systems. New York, USA: ACM Press, 2016: 45-48.
[15] JOYE M, TUNSTALL M. 密码故障分析与防护 [M]. 赵新杰，郭世泽，张帆，等译. 北京: 科学出版社, 2015.

编辑　陆燕菲