doi:10.11887/j.cn.202404025 http://journal.nudt.edu.cn

积分故障分析下的Midori128密码算法安全性评估

魏悦川[1,2*],贺水喻[1],潘 峰[1,2],王湘儒[1]
(1. 武警工程大学 密码工程学院,陕西 西安 710086;
2. 网络与信息安全武警部队重点实验室,陕西 西安 710086)

摘 要:为了研究Midori128密码算法针对积分故障攻击的安全性,建立积分区分器平衡位置、故障密文与轮密钥的关系,通过密钥搜索,可以恢复出算法的最后一轮密钥,进而利用密钥扩展算法恢复出主密钥。理论分析表明,利用3轮和4轮积分区分器进行积分故障攻击时,恢复出正确密钥的时间复杂度分别为2^{21}和2^{24}。采用准确性、成功率和耗费时间对倒数第4轮注入故障的攻击过程进行仿真,成功恢复出该算法的主密钥,并且针对不同明文分组和密钥进行对比实验。通过两组故障安全性分析方案可知,Midori128算法的轮函数易受到积分故障攻击,在算法运行时至少需要对倒数6轮进行故障检测等额外防护。

关键词:轻量级分组密码;Midori128算法;积分区分器;积分故障分析
中图分类号:TP309.7 **文献标志码**:A **文章编号**:1001-2486(2024)04-229-10

Security evaluation of Midori128 cryptographic algorithm under integral fault analysis

WEI Yuechuan[1,2*], *HE Shuiyu*[1], *PAN Feng*[1,2], *WANG Xiangru*[1]
(1. College of Password Engineering, Engineering University of PAP, Xi'an 710086, China;
2. Key Laboratory of Network and Information Security of PAP, Xi'an 710086, China)

Abstract: In order to study the security of the Midori128 cryptographic algorithm against integral fault attack, the relationship between integral distinguisher balance position, fault ciphertext, and the round key was established, and the last round key of the algorithm could be recovered by key search, and then the master key could be recovered by using key extension algorithm. The theoretical analysis shows that the time complexity of recovering the correct key is 2^{21} and 2^{24} when using 3 and 4 rounds of integral distinguisher for the integral fault attack, respectively. The accuracy, success rate, and elapsed time were used to simulate the attack process of the fourth round of injection fault, and the master key of the algorithm was successfully recovered. Comparison experiments were conducted for different plaintext groups and keys. The two sets of fault security analysis schemes conclude that the round function of the Midori128 algorithm is vulnerable to integral fault attacks and requires additional protection such as fault detection for at least the last 6 rounds while the algorithm is running.

Keywords: lightweight block cipher; Midori128 algorithm; integral distinguisher; integral fault analysis

随着物联网(internet of things,IoT)技术和快速通信技术的飞速发展,各种移动终端与物联网中的网络设备变得轻量化、便捷化,将网络与人们的生活紧密关联起来,达到了万物互联、万物智能化的效果。物联网作为新形态互联网络的代表,其适配设备种类纷繁复杂,越来越多的微型嵌入式设备被广泛地应用其中,如射频识别器(radio frequency identification,RFID)、无线传感器、激光扫描器、智能卡等。由于物联网不同于传统互联网,使用的设备计算、存储等资源受限,难以使用传统的分组密码进行加密通信和传输防护,给网络环境的安全带来极大挑战,因此,保护此类受限设备进行安全的信息传输和存储的需求越来越强烈。轻量级分组密码算法因其结构简单、加解密速度快、消耗资源少和利于实现等特点,完美契合于资源受限设备的布设,在此类信息系统安全领域发挥着重要的作

收稿日期:2022-04-12
基金项目:陕西省基础研究计划资助项目(2021JM-254)
*第一作者:魏悦川(1982—),女,天津蓟州人,副教授,博士,硕士生导师,E-mail:wych004@163.com
引用格式:魏悦川,贺水喻,潘峰,等. 积分故障分析下的Midori128密码算法安全性评估[J]. 国防科技大学学报,2024,46(4):229-238.
Citation: WEI Y C, HE S Y, PAN F, et al. Security evaluation of Midori128 cryptographic algorithm under integral fault analysis[J]. Journal of National University of Defense Technology, 2024, 46(4): 229-238.

用。针对资源受限设备,密码设计者提出了一系列轻量级分组密码算法,如 PRESENT[1-2]、LED[3]、LBlock[4]、MIBS[5]、SIMON 和 SPECK[6]等,这些轻量级密码算法的安全性备受关注。

Midori 是由 Banik 等[7]于 2015 年提出的一种轻量级分组密码算法,分组规模有两个版本,分别为 Midori64 和 Midori128,密钥长度相同均为 128 bit。该算法提供的计算开销非常小,相较于类似算法 PRINCE[8]和 CRAFT[9],其能耗更加低。到目前为止,密码学者对 Midori 算法进行了许多安全性评测工作。任瑶瑶等[10]对 Midori64 进行了 14 轮的相关密钥不可能差分分析,猜测密钥数量为 84 bit。Lin 等[11]使用中间相遇攻击,得到了 12 轮 Midori64 的攻击,时间复杂度为 $2^{125.5}$ 次 12 轮加密,数据复杂度为 $2^{55.5}$ 个 64 bit 分组。于政等[12]提出了 Midori64 的 11 轮不可能差分分析,攻击的时间复杂度为 $2^{121.6}$ 次 11 轮加密,数据复杂度为 $2^{62.3}$ 个 64 bit 分组。李明明等[13]在文献[12]的基础上提出 Midori64 的 11 轮截断不可能差分分析,时间、数据复杂度更加紧实,分别为 $2^{121.42}$ 次 11 轮加密和 $2^{60.82}$ 个 64 bit 分组。程璐等[14]对 Midori64 进行了 10 轮多维零相关线性分析,时间、数据复杂度为 $2^{79.35}$ 次 10 轮加密和 $2^{62.4}$ 个 64 bit 分组。文献[15-16]分别提出了 Midori64 的不变子空间攻击和非线性不变量攻击,并以此给出该算法的全轮弱密钥攻击。文献[17-18]分别对 Midori64 进行了 8 轮、10 轮及 11 轮积分攻击,时间复杂度分别为 2^{65} 次 8 轮、$2^{67.85}$ 次 10 轮和 $2^{117.37}$ 次 11 轮加密,数据复杂度分别为 $2^{19.80}$、2^{40} 和 $2^{40.09}$ 个明文对。

故障分析作为侧信道分析方法的其中一种,在现实环境下对密码算法的安全性分析更加有效,已经成为密码算法分析领域广受关注的一个方向,密码算法抵御故障分析的能力也成为衡量密码方案设计的一项重要指标。目前,故障分析衍生出许多种攻击方法,例如差分故障分析[19]、代数故障分析[20]等。积分故障分析方法使用积分关系与故障注入相结合的方式,可以充分利用密码算法的积分性质,深入算法内部并达到更深轮数的攻击。沈煜等[21]运用积分故障分析方法对韩国国家标准密码算法 ARIA 进行了分析。

针对 Midori 算法,王艺迪等结合差分故障分析及代数故障分析进行密钥恢复[22],结果表明 Midori 算法在运行中需要对其后 5 轮进行防护。本文提出了针对 Midori128 算法的积分故障分析,通过注入随机活跃字节故障,利用正确密文和错误密文构造积分区分器,利用中间状态与轮密钥(round key, RK)之间的积分关系,根据区分器中平衡字节特性恢复密钥。本文提出了两种积分故障分析方案,分别为在倒数第 4 轮注入活跃字节故障以及在倒数第 6 轮之后或倒数第 5 轮之前注入活跃字节故障,给出了密钥恢复方法,并采用准确性、成功率和耗费时间对攻击过程进行仿真实验验证。

1 Midori128 算法介绍

1.1 算法描述

Midori128 算法采用代换-置换网络(substitution-permutation network, SPN)结构,密钥长度为 128 bit,分组长度为 128 bit,迭代轮数为 20 轮,加密流程如图 1 所示,Midori128 的明文分组长度被分成 16 个字节,状态矩阵表示如下:

$$\boldsymbol{S}=\begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[1] & S[5] & S[9] & S[13] \\ S[2] & S[6] & S[10] & S[14] \\ S[3] & S[7] & S[11] & S[15] \end{pmatrix} \tag{1}$$

其中,$S[i]$($i=0,1,\cdots,15$)表示第 i 个字节。

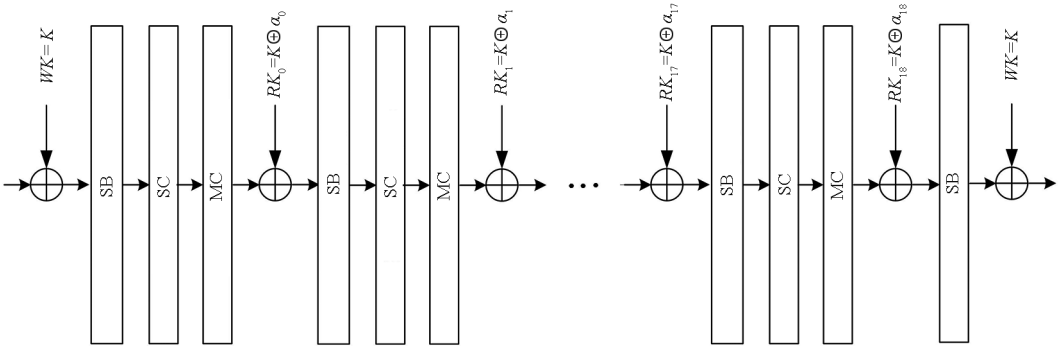

图 1 Midori128 算法加密过程

Fig. 1 Midori128 algorithm encryption process

Midori128 轮函数为包括字节替代(SubCell,SB)、置换(ShuffleCell,SC)、列混淆(MixColumn,MC)和轮密钥加(KeyAdd,KA)的复合操作,该轮函数作用在状态矩阵上。在算法加密的起始部分和结尾部分使用白化密钥进行异或,且最后一轮的轮函数只有字节替代操作,其解密流程是加密过程的逆运算。

1)字节替代:Midori128 采用 4 个 8 bit S 盒 $\boldsymbol{SSb}_0$、$\boldsymbol{SSb}_1$、$\boldsymbol{SSb}_2$、$\boldsymbol{SSb}_3$ 进行字节代换,每个 $\boldsymbol{SSb}_i$ 由输入和输出位的排列组合以及两个 $\boldsymbol{Sb}_1$ 复合得到,$\boldsymbol{Sb}_1$ 如表 1 所示。

表 1　Midori128 使用的 4 bit S 盒

Tab. 1　4 bit S-boxes used by Midori128

x	$Sb_1[x]$	x	$Sb_1[x]$
0	1	8	d
1	0	9	a
2	5	a	9
3	3	b	b
4	e	c	c
5	2	d	8
6	f	e	4
7	7	f	6

Midori128 算法的 $\boldsymbol{SSb}_i$ 中每个输出位的排列被视为相应的输入位排列的倒数,如图 2 所示,位的逆运算保持了内部关联属性,计算方式记为 $P_i(x)=y^i$。

$$\begin{cases} y^0_{[0,1,2,3,4,5,6,7]} = x_{[4,1,6,3,0,5,2,7]} \\ y^1_{[0,1,2,3,4,5,6,7]} = x_{[1,6,7,0,5,2,3,4]} \\ y^2_{[0,1,2,3,4,5,6,7]} = x_{[2,3,4,1,6,7,0,5]} \\ y^3_{[0,1,2,3,4,5,6,7]} = x_{[7,4,1,2,3,0,5,6]} \end{cases} \tag{2}$$

其中,$\boldsymbol{SSb}_i$ 分别作用于状态矩阵的各行。

2)置换:将 SB 作用之后的状态矩阵按字节进行置换,该置换及逆置换如下所示。

$$\begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[1] & S[5] & S[9] & S[13] \\ S[2] & S[6] & S[10] & S[14] \\ S[3] & S[7] & S[11] & S[15] \end{pmatrix} \xrightarrow{\text{SC}} \begin{pmatrix} S[0] & S[14] & S[9] & S[7] \\ S[10] & S[4] & S[3] & S[13] \\ S[5] & S[11] & S[12] & S[2] \\ S[15] & S[1] & S[6] & S[8] \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} S[0] & S[4] & S[8] & S[12] \\ S[1] & S[5] & S[9] & S[13] \\ S[2] & S[6] & S[10] & S[14] \\ S[3] & S[7] & S[11] & S[15] \end{pmatrix} \xrightarrow{\text{SC}^{-1}} \begin{pmatrix} S[0] & S[5] & S[15] & S[10] \\ S[7] & S[2] & S[8] & S[13] \\ S[14] & S[11] & S[1] & S[4] \\ S[9] & S[12] & S[6] & S[3] \end{pmatrix} \tag{4}$$

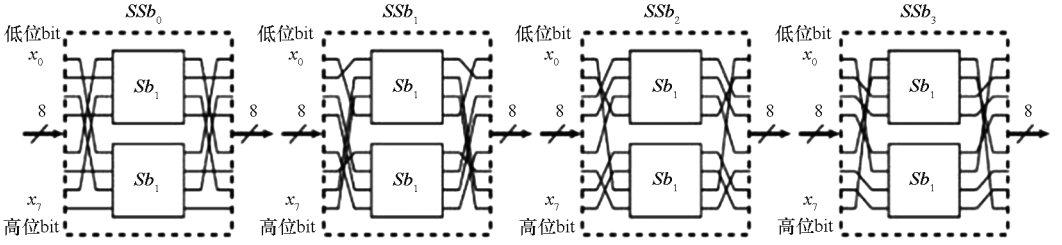

图 2　Midori128 算法的字节代替(SB)

Fig. 2　The detail of Midori128 algorithm's SB

3)列混淆:将 SC 作用之后的结果左乘矩阵 $\boldsymbol{M}$,执行按列混合变换,操作如下。

$$\begin{pmatrix} S[i] \\ S[i+1] \\ S[i+2] \\ S[i+3] \end{pmatrix} \xrightarrow{\text{MC}} \boldsymbol{M} \begin{pmatrix} S[i] \\ S[i+1] \\ S[i+2] \\ S[i+3] \end{pmatrix} \tag{5}$$

其中,$\boldsymbol{M}=\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$。

4)轮密钥加:将 MC 作用之后的状态矩阵和轮密钥进行按位异或计算。

Midori128 算法加密过程的伪代码如算法 1 所示。

1.2　密钥扩展算法

Midori128 密钥扩展算法较为简单,如算法 2 所示。主密钥由两个 64 bit 的密钥 K_0 和 K_1 级联构成,即 $K=K_0 \parallel K_1$,主密钥 K 直接作为白化密钥使用,即 $WK=K$,在首轮使用轮函数之前和最后一轮使用轮函数之后参与运算;轮密钥 $RK_i=$

算法 1 Midori128 轮函数加密过程

Alg. 1 Midori128 round function encryption process

MidoriCore:

$$\begin{cases}\{0,1\}^{16m}\times\{0,1\}^{16m}\times\{\{0,1\}^{16m}\}^{19}\to\{0,1\}^{16m}\\(X,WK,RK_0,\cdots,RK_{18})\to Y\end{cases}$$

Algorithm MidoriCore($X,WK,RK_0,\cdots,RK_{18}$):

$\boldsymbol{S}\leftarrow$KeyAdd(X,WK)

for $i=0$ to 18

$\boldsymbol{S}\leftarrow$SubCell($\boldsymbol{S}$)

$\boldsymbol{S}\leftarrow$ShuffleCell($\boldsymbol{S}$)

$\boldsymbol{S}\leftarrow$MixColumn($\boldsymbol{S}$)

end for

$\boldsymbol{S}\leftarrow$SubCell($\boldsymbol{S}$)

$Y\leftarrow$KeyAdd($\boldsymbol{S},WK$)

$K\oplus\alpha_i$($0\leqslant i\leqslant 18$),其中,$\oplus$为异或运算,α_i 为轮常数,表 2 为轮常数所选参数。

算法 2 Midori128 的密钥扩展

Alg. 2 Key extension for Midori128

$$L_{(RK_i)}:\begin{cases}\{0,1\}^{128}\times\{0,1\}^{16}\to\{0,1\}^{128}\\(K_0,K_1,\alpha_0,\alpha_1,\cdots,\alpha_{18})\to(RK_i)\end{cases}$$

$L_{(RK_i)}(K_0,K_1,\alpha_0,\alpha_1,\cdots,\alpha_{18})\to(RK_i)$:

$K\leftarrow K_0\parallel K_1$

for $i=0$ to 18

$RK_i\leftarrow K\oplus\alpha_i$

end for

$WK\leftarrow K$

表 2 轮常数 α_i

Tab. 2 The round constants α_i

轮常数	α_0	α_1	α_2	α_3
取值	0010010000111111	0110101010001000	1000010110100011	0000100011010011
轮常数	α_4	α_5	α_6	α_7
取值	0001001100011001	1000101000101110	0000001101110000	0111001101000100
轮常数	α_8	α_9	α_{10}	α_{11}
取值	1010010000001001	0011100000100010	0010100110011111	0011000111010000
轮常数	α_{12}	α_{13}	α_{14}	α_{15}
取值	0000100000101110	1111101010011000	1110110001001110	0110110010001001
轮常数	α_{16}	α_{17}	α_{18}	
取值	0100010100101000	0010000111100110	0011100011010000	

2 Midori128 算法的积分故障分析

积分故障分析方法是故障分析的一种新型方式,其主要思路是结合积分分析可以深入算法内部的特点,在密码算法任一轮之间注入随机故障,根据积分区分器的长度,推算出密码算法某一轮的轮密钥,而后反向解密该轮的状态信息,并以此类推,可获得多轮子密钥信息,最终根据密钥扩展方式推断出正确密钥。积分故障分析可以以较低的数据和存储复杂度在物联网等应用中实现,Phan 等[23]首次使用该分析方式对高级加密标准(advanced encryption standard, AES)的倒数 4 轮进行分析,并获得多轮密钥信息。

2.1 符号说明

Y、$Y^{(u)}$ 分别表示正确密文和第 u 次注入故障的错误密文,均为 128 bit;K 为 128 bit 主密钥;WK 为 128 bit 白化密钥;r 为迭代轮数。

2.2 Midori128 的故障假设和故障模型

故障假设表示攻击者在现实环境中具备的优势或能力,对于 Midori128 算法,本文基于三点假设:

1)对同一个明文,攻击者可以获得在同一个密钥作用下的正确密文 Y 和错误密文 $Y^{(u)}$。

2)攻击者可以在算法运行中注入故障,使得某个字节出错。故障字节的具体位置未知,具体的错误值未知,但是每次注入故障的位置相同,错误值不同,即错误值遍历 1 ~255。

3)在一次故障分析中,密钥的值不发生改变。

本文所使用故障模型为:随机活跃字节故障,即在算法运行的某一轮的指定位置引入随机活跃字节故障。

2.3　主要分析思路

攻击者选择任意明文进行加密操作,同时在使用相同密钥的情况下引入活跃字节故障,得到 1 组密文,其中 1 个密文为正确密文,255 个密文为错误密文;而后构造基于字节的积分区分器,根据区分器的平衡位置对最后 1 轮密钥进行筛选,找到正确轮密钥,结合密钥扩展算法,恢复出主密钥。

2.4　基于 3 轮积分区分器的故障攻击方案

本文根据 Midori128 算法轮函数特点构造了一个 3 轮积分区分器,该区分器并不是 Midori128 已知的最长区分器,却是零和效果最好的区分器。

在构造积分区分器时,需要对字节的性质作以下定义[24]。

稳定字节:对于定义在 F_{2^8} 上的字节集合 $C=\{a^{(u)}\mid 0\leqslant u\leqslant 2^8-1\}$,对于任意 $0\leqslant u\leqslant v\leqslant 2^8-1$,均有 $a^{(u)}=a^{(v)}$,则称 C 为 F_{2^8} 上的稳定字节。

活跃字节:对于定义在 F_{2^8} 上的字节集合 $A=\{a^{(u)}\mid 0\leqslant u\leqslant 2^8-1\}$,对于任意 $0\leqslant u<v\leqslant 2^8-1$,均有 $a^{(u)}\neq a^{(v)}$,则称 A 为 F_{2^8} 上的活跃字节。

平衡字节:对于定义在 F_{2^8} 上的字节集合 $B=\{a^{(u)}\mid 0\leqslant u\leqslant 2^8-1\}$ 满足 $\sum_{u=0}^{2^8-1}a^{(u)}=0$,则称 B 为 F_{2^8} 上的平衡字节。

其中,稳定/活跃字节经过字节替代和轮密钥异或运算后仍然是稳定/活跃字节;活跃字节的异或为平衡字节;稳定字节异或活跃字节后为活跃字节;平衡字节相互异或后仍为平衡字节。

下面给出 Midori128 算法的 3 轮积分区分器。

性质 1:对于 Midori128 算法,如果第 r 轮输入状态中有 1 个活跃字节而其他字节均为稳定字节,则 $r+2$ 轮输出的所有字节均为平衡字节。

以上区分器可以根据字节扩散的性质得出,区分器的一个实例如图 3 所示,该区分器可以形式化地表示为:

$$ACCC\ CCCC\ CCCC\ CCCC\xrightarrow{3\text{轮}}BBBB\ BABA\ BAAB\ BBAA$$

注意到活跃字节一定是平衡字节,因此以上区分器中,输出的所有字节均是平衡的,并且当输入字节中 A 的位置变化时,输出字节的平衡性仍然保持不变。以上区分器中输出的活跃字节的平衡性可以进行证明。

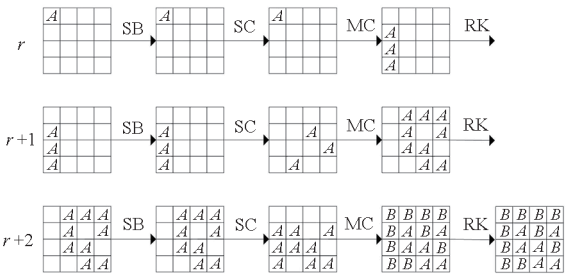

图 3　Midori128 的 3 轮积分区分器

Fig. 3　Midori128′s 3-round integral distinguisher

根据文献[25]中的两个定理证明性质 1 中区分器输出活跃字节为平衡字节的正确性。

定理 1　设多项式 $f(x)=\sum_{u=0}^{q-1}b^{(u)}x^u\in F_q[x]$,其中 q 是某个素数的幂方,则 $\sum_{x\in F_q}f(x)=-b^{q-1}$。

定理 2　若多项式 $f(x)=\sum_{u=0}^{q-1}b^{(u)}x^u\in F_q[x]$ 是置换多项式,则 $b^{q-1}=0$。

证明:$\tau_0,\tau_1,\tau_2,\cdots,\tau_{15}$ 为第 r 轮输入字节,其中 $a=\tau_0$ 为活跃字节,其他均为稳定字节,则第 r 轮的输入表示为 $(\tau_0,\tau_1,\tau_2,\cdots,\tau_{15})=(a,\tau_1,\tau_2,\cdots,\tau_{15})$,$a$ 为变量,$\tau_1,\tau_2,\cdots,\tau_{15}$ 均为常量。由图 3 可知,第 r 轮输出可以表示为 $(\tau_0,f_1(a),f_2(a),f_3(a),\tau_4,\cdots,\tau_{15})$,其中,$\tau_0,\tau_4,\cdots,\tau_{15}$ 为常量,$f_1(a),f_2(a),f_3(a)$ 为 F_{2^8} 域上的置换多项式。

第 $r+2$ 轮的输入为 $(\beta,\beta,\beta,\beta,\beta,h_5(a),h_6(a),\beta,\beta,\beta,h_{10}(a),h_{11}(a),\beta,h_{13}(a),\beta,h_{15}(a))$。其中,$h_5(a),h_6(a),h_{10}(a),h_{11}(a),h_{13}(a),h_{15}(a)$ 是 F_{2^8} 域上的置换多项式。上述的每个字节均可用 F_{2^8} 上的 8 个置换多项式的积分和表示。

假设 $r+2$ 轮输入的第一个字节为 $p_0(a)\oplus p_1(a)\oplus\cdots\oplus p_7(a)$,其中,$p_i(a)$ 是 F_{2^8} 上的置换多项式。结合定理 2 可得

$$\sum_{a\in F_{2^8}}p_0(a)=\sum_{a\in F_{2^8}}p_1(a)=\cdots=\sum_{a\in F_{2^8}}p_7(a)=0 \tag{6}$$

因此

$$\begin{aligned}&\sum_{a\in F_{2^8}}(p_0(a)\oplus p_1(a)\oplus\cdots\oplus p_7(a))\\&=\sum_{a\in F_{2^8}}p_0(a)=\cdots=\sum_{a\in F_{2^8}}p_7(a)\\&=0\end{aligned} \tag{7}$$

□

通过在第 r 轮的状态矩阵任意字节引入随机活跃字节故障，依据以上区分器，可以得出 $r+2$ 轮的输出均为平衡字节。本文选择以上区分器进行分析，是因为所有输出字节均为平衡字节，能够以较高的效率恢复轮密钥。

基于以上积分区分器，可以给出在倒数第 4 轮（第 r 轮）注入活跃字节故障的攻击方案。

Step 1：选择任意明文 X，使用主密钥 K 进行加密操作，得到正确密文 Y。

Step 2：重复加密操作，攻击者在加密过程的倒数第 5 轮 MC 之后、倒数第 4 轮 MC 之前的任何一个时刻和位置注入活跃字节故障，取值在 [0,255] 之间，获得故障密文，如图 4 所示。在倒数第 5 轮 MC 之后、倒数第 4 轮 MC 之前的任何一个时刻和位置注入活跃字节故障，不会影响活跃字节的属性。后续各状态矩阵的积分值随着故障注入发生变化，密文 $Y^{(u)}$ 存在如下关系。

$$\begin{aligned}\sum_{u=0}^{255} \boldsymbol{S}_{r+3}^{(u)} &= \sum_{u=0}^{255} [\mathrm{SB}^{-1}(Y^{(u)} \oplus WK^{(u)})] \\ &= \sum_{u=0}^{255} [\mathrm{SB}^{-1}(Y^{(u)} \oplus WK)] \\ &= \sum_{u=0}^{255} [\mathrm{SB}(Y^{(u)} \oplus WK)] \end{aligned} \tag{8}$$

以上关系式利用了 SB 变换的对合性，其中，$\boldsymbol{S}_{r+3}^{(u)}$、$WK^{(u)}$ 分别表示第 u 次注入故障时 $\boldsymbol{S}_{r+3}$、WK 的值，$u=0$ 对应正确的状态矩阵和密文。

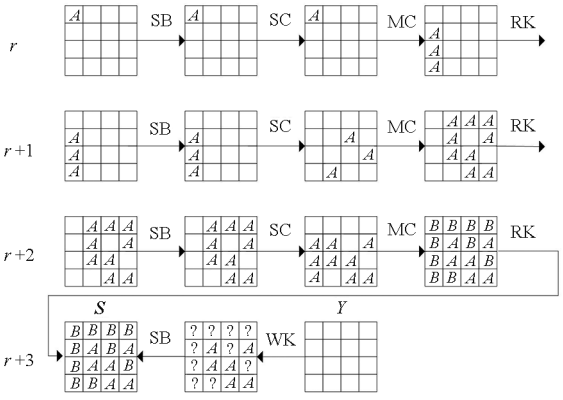

图 4 基于 3 轮积分区分器的故障分析过程

Fig. 4 Fault analysis process based on 3-round integral distinguisher

Step 3：基于 3 轮积分区分器，$\boldsymbol{S}_{r+3}$ 中的每个字节都是平衡的，即：

$$\sum_{u=0}^{255} S_{r+3}^{(u)}[i] = 0 (0 \leqslant i \leqslant 15) \tag{9}$$

其中，$S_{r+3}^{(u)}[i]$ 是 4 轮加密操作后状态矩阵在位置 i 处的值。攻击者逐字节搜索白化密钥 $WK[i]$，计算 $S_{r+3}^{(u)}[i] = \mathrm{SB}(Y^{(u)}[i] \oplus WK[i])$，$Y^{(u)}[i]$ 是故障密文中对应位置 i 的故障字节值。然后判断是否满足 $\sum_{u=0}^{255} S_{r+3}^{(u)}[i] = 0$，若满足，则将 $WK[i]$ 记为候选密钥，不满足则淘汰。

Step 4：保持加密过程中使用的主密钥不变，利用不同明文加密获得对应密文集，包含 1 个正确密文，255 个错误密文。重复进行步骤 1 ~ 3，再次筛选候选密钥值，直到候选密钥被唯一确定，即恢复出正确的白化密钥。

Step 5：根据 Midori128 密钥扩展算法可知，$WK = K$，所恢复出的白化密钥即为主密钥。

2.5 基于 4 轮积分区分器的故障攻击方案

基于 4 轮积分区分器进行故障攻击所使用的区分器，是对连闯[17]给出的 Midori64 的 4 轮区分器的扩展，也是目前已知的 Midori 算法最长的基于字节的积分区分器，性质 2 是文献[17]给出的区分器，性质 3 是本文扩展后的区分器。

性质 2[17]：对于 Midori128 算法，如果第 r 轮明文输入状态中第 1 个字节是活跃字节，其他字节均为稳定字节，则 $r+3$ 轮输出的第 1 个字节为平衡字节。

以上字节状态的扩散过程如图 5 所示，该积分区分器可以形式化地表示为：

$$ACCC\ \ CCCC\ \ CCCC\ \ CCCC \xrightarrow{4\text{轮}} B??\ ????\ ????\ ????$$

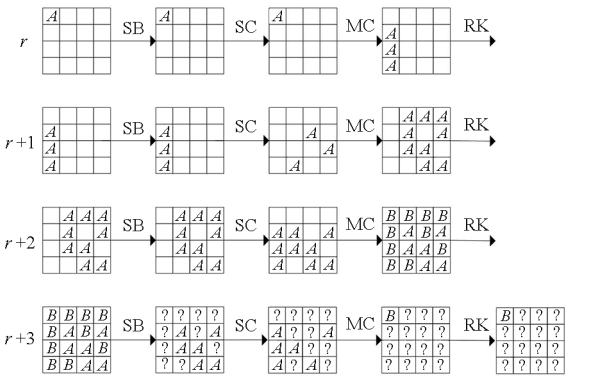

图 5 Midori128 的 4 轮积分区分器

Fig. 5 Midori128's 4-round integral distinguisher

经过深入分析，我们发现以上积分区分器可以扩展为以下形式。

性质 3：对于 Midori128 算法，如果第 r 轮明文输入状态中第 i 个字节是活跃字节，其他字节均为稳定字节，则 $r+3$ 轮输出的第 i 个字节为平衡字节。即：$C\cdots C\overset{i}{A}C\cdots C \xrightarrow{4\text{轮}} ?\cdots ?\overset{i}{B}?\ \cdots ?$。

基于性质 3 中的区分器，本节给出以下攻击方案：在倒数第 6 轮 MC 之后或倒数第 5 轮 MC 之前的任何一个时刻注入随机活跃字节故障，使

用与 3 轮攻击方案相同的思路进行攻击,则可以恢复出 1 个字节的密钥,故障分析过程见图 6。要恢复出最后一轮密钥的全部字节,需要将上述过程重复 16 次。

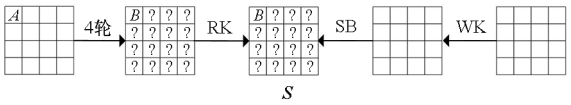

图 6　基于 4 轮积分区分器的故障分析过程

Fig. 6　Fault analysis process based on 4-round integral distinguisher

根据性质 3 所给出的区分器实施攻击方案如下:

Step 1:选择任意明文 X,使用主密钥 K 进行加密操作,得到正确密文 Y。

Step 2:重复加密操作,攻击者在加密过程的倒数第 6 轮 MC 之后或倒数第 5 轮 MC 之前任何一个时刻注入活跃字节故障,故障位于第 i 个字节($0 \leqslant i \leqslant 15$),取值在[0,255]之间,并得到故障密文。根据性质 3 中的结论可知,在引入活跃字节故障之后,经过 4 轮加密操作可让状态矩阵相应位置的字节达到平衡,并且存在如下关系:

$$\sum_{u=0}^{255} S_{r+3}^{(u)}[i] = \sum_{u=0}^{255} [\mathrm{SB}(Y^{(u)}[i] \oplus WK[i])] \tag{10}$$

Step 3:攻击者通过穷尽搜索对应的密钥字节(取值[0,255]),即验证是否存在密钥字节使得 $\sum_{u=0}^{255} S_{r+3}^{(u)}[i] = 0$ 成立,若存在则保留其为候选密钥字节。

Step 4:保持加密过程的主密钥不变,依次更换活跃字节注入位置,直到恢复出 WK 全部的候选密钥字节 $WK[i]$(此时 $WK[i]$可能并不唯一)。

Step 5:保持加密过程中使用的主密钥不变,利用不同明文加密获得对应密文集,包含 1 个正确密文,255 个错误密文。重复进行步骤 1 ~ 4,再次筛选 $WK[i]$的候选密钥值,直到候选密钥被唯一确定,即得到正确的白化密钥。

Step 6:根据 Midori128 密钥扩展算法可知,$WK = K$,所恢复出的白化密钥即为主密钥。

3　理论分析与实验结果

3.1　理论时间复杂度

在攻击方案实施过程中,为了计算平衡字节的积分和,攻击者至少需要 1 组密文,包含 1 个正确的密文和 255 个故障的密文。设 δ 表示 1 个错误轮密钥满足积分和为零的概率,Γ 表示穷尽搜索轮密钥的空间。当攻击者诱导 τ 组密文集时,轮密钥候选值中剩余错误密钥数为$(\Gamma - 1)\delta^{\tau}$。若$(\Gamma - 1)\delta^{\tau} < 1$,则表示轮密钥候选值可以被唯一确定,即正确的轮密钥。

3 轮积分故障攻击过程中有:

$$\begin{cases} \Gamma = 2^8 \\ \delta = 2^{-8} \end{cases} \tag{11}$$

如果 $\tau > 2$,则攻击者可以找到正确的密钥,此时穷尽搜索 1 组密文集的时间复杂度为$2^8 \times 2^8 \times 16 = 2^{20}$,即恢复 Midori128 算法主密钥的最少理论时间复杂度为 2^{21}。4 轮积分故障攻击时,由于单次实验平衡字节数仅为 1,需要进行 16 次不同字节位置故障导入,所以时间复杂度为 $2^8 \times 2^8 \times 16 \times 16 = 2^{24}$。

3.2　实验结果分析与对比

本文的实验环境为 Inter Core I5 - 10800H@ 4.6 GHz、内存为 16 GB 的计算机,使用系统环境 C ++ 语言程序(Code ∶∶ Blocks 20.03)编写 Midori128 算法加解密和积分故障分析计算机仿真过程,通过程序模拟故障注入并生成错误密文,解密一轮后可得到候选密钥值。使用的明密文分组及密钥如表 3 所示。以在倒数第 4 轮任意字节注入故障的方案为例,单字节密钥恢复实验中,选定字节位置引入故障,可恢复对应平衡字节所使用的密钥字节,表 4 为进行 1 000 次实验时恢复的各个字节位置平均候选密钥个数。

在进行多字节密钥恢复实验时,本节通过采用准确性、成功率和耗费时间对实验结果进行详细描述。候选密钥值与真实密钥值之间的近似程度通过准确性进行衡量,本文采用均方根误差(root mean square error,RMSE)公式来进行计算。

$$RESM = \sqrt{\frac{\sum_{e=1}^{n} (\phi(e) - \theta)}{n}} \tag{12}$$

式中,n 为实验次数,$\phi(e)$为在第 e 次实验时恢复的候选密钥的字节数,θ 为真正密钥的字节数,其中 $n = 200$,$\theta = 16$。本文共进行 1 000 次实验,平均分为 5 组,分别为 G_1、G_2、G_3、G_4 和 G_5。均方根误差越接近于 0 则实验结果越准确,候选密钥的均方根误差如表 5 所示,实验结果表明,在倒数第 4 轮注入故障攻击方案中,使用 3 组密文集合进行密钥恢复操作即可恢复出正确密钥。图 7 展示了使用 2 组密文集合进行密钥恢复后,筛选出正确密钥的效率波形,图 8 显示表 5 中对应的使用 2 组和 3 组密文集时,成功概率平均值分别为 94.7% 和 100%。

表 3 实验明密文实例

Tab. 3 Examples of experimental plaintexts

分组	密钥 1:687ded3b3c85b3f35b1009863e2a8cbf	
	明文	密文
分组 1	51084ce6e73a5ca2ec87d7babc297543	1e0ac4fddff71b4c1801b73ee4afc83d
分组 2	f53ee29eb1f337a966b5cde43d047fdc	5519e60051e371cf76eaf5bc74fcbd07
分组 3	55cbb95996d14902b60574d5e728d6ac	e0d59ebf8731af78d1d5509ed268bf
分组	密钥 2:2a4de36b4f77c5d32b1109369c1a6cad	
	明文	密文
分组 1	51084ce6e73a5ca2ec87d7babc297543	2a0b3dade80a8f35d6c65d0e6f6a44ce
分组 2	f53ee29eb1f337a966b5cde43d047fdc	59d8d1dbba0924cc4c02216c9803d63d
分组 3	55cbb95996d14902b60574d5e728d6ac	a1ed8e1bece4f044171409e5ef90dc75

表 4 各字节实验 1 000 次的平均候选密钥个数

Tab. 4 Average number of candidate keys for each byte experiment 1 000 times

密钥	$S[0]$	$S[1]$	$S[2]$	$S[3]$	$S[4]$	$S[5]$	$S[6]$	$S[7]$	$S[8]$	$S[9]$	$S[10]$	$S[11]$	$S[12]$	$S[13]$	$S[14]$	$S[15]$
密钥 1	1.81	1.05	1.22	1.17	1.05	1.06	1.31	1.04	1.73	1.19	1.26	1.86	1.01	1.10	1.18	1.21
密钥 2	1.98	2.26	1.23	1.13	2.55	2.08	1.47	2.14	1.75	1.15	1.33	1.61	1.41	1.91	1.16	2.22

表 5 恢复密钥的均方根误差

Tab. 5 RMSE of recovery keys

集合	G_1	G_2	G_3	G_4	G_5
1	4.12	4.69	3.87	3.95	4.24
2	0.50	0.67	0.71	0.44	0.84
3	0	0	0	0	0

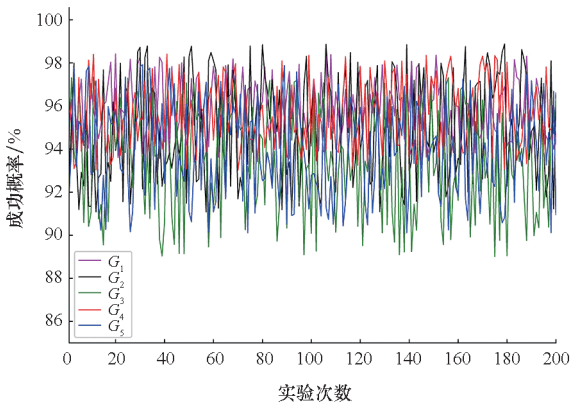

图 7 两次密钥筛选效率

Fig. 7 Twice key screening efficiency

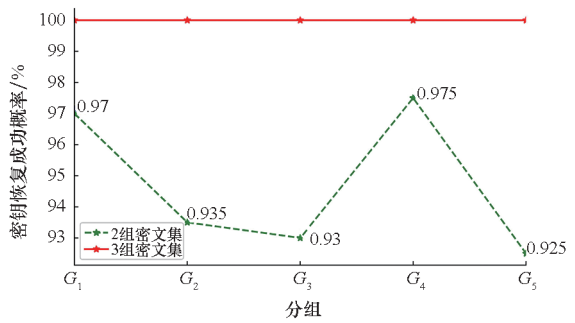

图 8 成功恢复密钥概率图

Fig. 8 Probability of successful key recovery

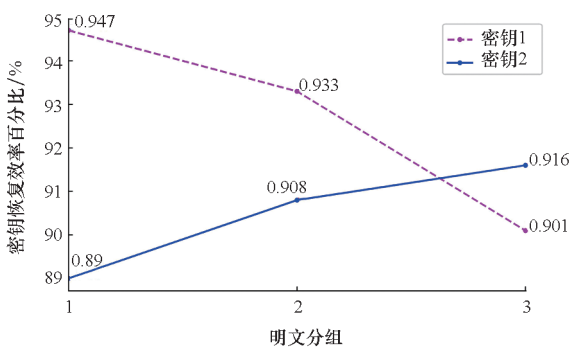

图 9 密钥筛选对比

Fig. 9 Key filter comparison

通过更换密钥对应不同明文分组,进行多次对比实验后,总体密钥恢复的效率较高,结果如图 9所示。从图 9 中可以看出,进行两次筛选后可以以高概率得到正确的密钥信息。

本文基于 3 轮积分区分器攻击方案中单次实验耗费的时间处于 12 ~ 21 s;在使用 4 轮积分区分器进行故障攻击时,单次实验耗费时间处于 5 ~ 14 s,由于单次实验仅可以恢复一个密钥字

节,密钥恢复的效率有所下降,恢复密钥字节的平均时间约为 9.2 s。

传统的差分故障分析利用的是字节的差分传播特性,本文的积分故障分析利用的是字节的积分特性,两者恢复密钥的原理是不同的。传统的差分故障分析所需的明文量较少,但是故障的注入轮相对较浅,积分故障分析中,由于 Midori128 算法是基于字节设计的,能够获得较好的字节零和性质,因此积分故障分析更适合对算法进行深度的故障注入分析,并且具有更高的恢复效率。

4　结论

本文提出了基于 Midori128 算法的积分故障分析方法,构造了 Midori128 算法的 3 轮和 4 轮积分区分器,并采取随机活跃字节故障注入的方式进行分析。使用基于 3 轮和 4 轮积分区分器进行故障分析分别可深入 Midori128 算法的 4 轮和 6 轮。理论分析表明了 Midori128 算法恢复主密钥的可行性,而后通过实验结果进行验证。相较于传统的故障分析方法,本文方案能够对算法的攻击轮数更高,故障注入更为深入。由本文提出的两种方案可知,使用 Midori 密码算法加密的同时,至少应当对倒数 6 轮进行故障检测等硬件防护,避免引起密钥泄露。

参考文献(References)

[1] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: an ultra-lightweight block cipher [C]// Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, 2007.

[2] 黄湘蜀, 王敏, 杜之波, 等. 针对轻量级分组密码算法 PRESENT 的随机差分故障攻击[J]. 成都信息工程大学学报, 2022, 37(1): 8 – 15.
HUANG X S, WANG M, DU Z B, et al. Random differential fault attack against the lightweight block cipher algorithm PRESENT[J]. Journal of Chengdu University of Information Technology, 2022, 37(1): 8 – 15. (in Chinese)

[3] GUO J, PEYRIN T, POSCHMANN A, et al. The LED block cipher[C]//Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, 2011.

[4] CUI Y X, XU H, QI W F. Improved integral attacks on 24-round LBlock and LBlock-s[J]. IET Information Security, 2020, 14(5): 505 – 512.

[5] 任炯炯, 侯泽洲, 李曼曼, 等. 改进的减轮 MIBS-80 密码的中间相遇攻击[J]. 电子与信息学报, 2022, 44(8): 2914 – 2923.
REN J J, HOU Z Z, LI M M, et al. Improved meet-in-the-middle attacks on reduced-round MIBS-80 cipher[J]. Journal of Electronics & Information Technology, 2022, 44(8): 2914 – 2923. (in Chinese)

[6] SUSANTI B H, PERMANA O J, AMIRUDDIN. Robustness test of SIMON-32, SPECK-32, and SIMECK-32 algorithms using fixed-point attacks[J]. Journal of Physics: Conference Series, 2021, 1836(1): 012006.

[7] BANIK S, BOGDANOV A, ISOBE T, et al. Midori: a block cipher for low energy[C]//Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, 2015.

[8] MÜLLER N, MOOS T, MORADI A. Low-latency hardware masking of PRINCE[C]//Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design, 2021.

[9] HADIPOUR H, SADEGHI S, NIKNAM M M, et al. Comprehensive security analysis of CRAFT[J]. IACR Transactions on Symmetric Cryptology, 2020, 2019(4): 290 – 317.

[10] 任瑶瑶, 张文英. Midori64 的相关密钥不可能差分分析[J]. 计算机应用研究, 2018, 35(6): 1800 – 1802.
REN Y Y, ZHANG W Y. Related-key differential analysis of Midori64[J]. Application Research of Computers, 2018, 35(6): 1800 – 1802. (in Chinese)

[11] LIN L, WU W L. Meet-in-the-middle attacks on reduced-round Midori64[J]. IACR Transactions on Symmetric Cryptology, 2017, 2017(1): 215 – 239.

[12] 于政, 毛明, 李艳俊. 基于轮密钥分步猜测方法的 Midori64 算法 11 轮不可能差分分析[J]. 计算机应用研究, 2018, 35(9): 2777 – 2780.
YU Z, MAO M, LI Y J. Impossible differential analysis of 11-round Midori64 based on method of step-key-guessing[J]. Application Research of Computers, 2018, 35(9): 2777 – 2780. (in Chinese)

[13] 李明明, 郭建胜, 崔竞一, 等. Midori-64 算法的截断不可能差分分析[J]. 软件学报, 2019, 30(8): 2337 – 2348.
LI M M, GUO J S, CUI J Y, et al. Truncated impossible differential cryptanalysis of Midori-64[J]. Journal of Software, 2019, 30(8): 2337 – 2348. (in Chinese)

[14] 程璐, 魏悦川, 李安辉, 等. Midori 算法的多维零相关线性分析[J]. 山东大学学报(理学版), 2018, 53(2): 88 – 94.
CHENG L, WEI Y C, LI A H, et al. Multidimensional zero-correlation linear cryptanalysis on Midori[J]. Journal of Shandong University (Natural Science), 2018, 53(2): 88 – 94. (in Chinese)

[15] GUO J, JEAN J, NIKOLIC I, et al. Invariant subspace attack against Midori64 and the resistance criteria for S-box designs[J]. IACR Transactions on Symmetric Cryptology, 2016, 2016(1): 33 – 56.

[16] TODO Y, LEANDER G, SASAKI Y. Nonlinear invariant attack: practical attack on full SCREAM, iSCREAM, and Midori64[J]. Journal of Cryptology, 2019, 32(4): 1383 – 1422.

[17] 连闯. 轻量级分组密码的积分分析[D]. 西安: 西安电子科技大学, 2018.

LIAN C. Integral cryptanalysis of lightweight block ciphers[D]. Xi'an: Xidian University, 2018. (in Chinese)

[18] 王超, 陈怀凤. Midori64 分组密码算法的积分攻击[J]. 计算机工程, 2021, 47(5): 117 – 123.
WANG C, CHEN H F. Integral attacks on Midori64[J]. Computer Engineering, 2021, 47(5): 117 – 123. (in Chinese)

[19] GUO H, SUN S W, SHI D P, et al. Differential attacks on CRAFT exploiting the involutory S-boxes and tweak additions[J]. IACR Transactions on Symmetric Cryptology, 2020, 2020(3): 119 – 151.

[20] VIELHABER M. Breaking one. Fivium by AIDA an algebraic IV differential attack[EB/OL]. [2022 – 04 – 01]. https://eprint. iacr. org/2007/413. pdf.

[21] 沈煜, 李玮, 谷大武, 等. ARIA 密码的积分故障分析[J]. 通信学报, 2019, 40(2): 164 – 173.
SHEN Y, LI W, GU D W, et al. Integral fault analysis of the ARIA cipher[J]. Journal on Communications, 2019, 40(2): 164 – 173. (in Chinese)

[22] 王艺迪, 赵新杰, 张帆, 等. Midori 算法抗故障攻击安全性评估[J]. 密码学报, 2017, 4(1): 58 – 78.
WANG Y D, ZHAO X J, ZHANG F, et al. Security evaluation for fault attacks on lightweight block cipher Midori[J]. Journal of Cryptologic Research, 2017, 4(1): 58 – 78. (in Chinese)

[23] PHAN R C W, YEN S M. Amplifying side-channel attacks with techniques from block cipher cryptanalysis[C]// Proceedings of the International Conference on Smart Card Research and Advanced Applications, 2006.

[24] DAEMEN J, KNUDSEN L, RIJMEN V. The block cipher Square[C]//Proceedings of the International Workshop on Fast Software Encryption, 1997.

[25] LIDL R, NIEDERREITER H. Finite fields[M]. 2nd ed. Cambridge, UK: Cambridge University Press, 1997.