Modeling and Simulation 建模与仿真, 2025, 14(4), 579-592
Published Online April 2025 in Hans. https://www.hanspub.org/journal/mos
https://doi.org/10.12677/mos.2025.144311

Hans汉斯

基于MILP的11轮INLEC的中间相遇分析

曾衡顺[1]，刘　亚[1*]，赵逢禹[2]，刘先蓓[3]，曲　博[4]

[1]上海理工大学光电信息与计算机工程学院，上海
[2]上海出版印刷高等专科学校信息与智能工程系，上海
[3]安徽财经大学统计与应用数学学院，安徽 蚌埠
[4]港专学院网络空间科技学院，香港

收稿日期：2025年3月18日；录用日期：2025年4月11日；发布日期：2025年4月21日

摘　要

INLEC是专为物联网环境设计的新的轻量级分组密码算法，为了保障它在物联网环境中的安全性，就必须研究其安全强度。本文提出了基于MILP的11轮INLEC的中间相遇分析。首先基于INLEC算法，建立了混合整数线性规划的搜索模型，找到若干条5轮INLEC中间相遇区分器；其次，根据分析过程中猜测密钥位数最少原则，选取1条中间相遇区分器，并往前扩展2轮，往后扩展4轮，构造出11轮INLEC中间相遇分析路径；最后基于该路径恢复出完全的密钥。整个分析过程的时间复杂度为$2^{115.17}$次加密，数据复杂度为2^{61}个选择明文，存储复杂度为2^{81}个64比特块。该结果为首个INLEC第三方安全性分析成果。

关键词

INLEC，中间相遇分析，混合整数线性规划，轻量级分组密码

MILP-Based for Meet-in-the-Middle Attack of 11-Round INLEC

Hengshun Zeng[1], Ya Liu[1*], Fengyu Zhao[2], Xianbei Liu[3], Bo Qu[4]

[1]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai
[2]Department of Information and Intelligence Engineering, Shanghai Publishing and Printing College, Shanghai
[3]School of Statistics and Applied Mathematics, Anhui University of Finance and Economics, Bengbu Anhui
[4]School of Cyberspace Technology, Hong Kong College of Technology, Hong Kong SAR

Received: Mar. 18th, 2025; accepted: Apr. 11th, 2025; published: Apr. 21st, 2025

Abstract

INLEC is a lightweight block cipher for resource-constrained Internet of Things environments. Thus,

*通讯作者。

文章引用: 曾衡顺, 刘亚, 赵逢禹, 刘先蓓, 曲博. 基于 MILP 的 11 轮 INLEC 的中间相遇分析[J]. 建模与仿真, 2025, 14(4): 579-592. DOI: 10.12677/mos.2025.144311

it is essential to evaluate its security boundary further. Our research proposes a meet-in-the-middle attack on 11-round INLEC based on the MILP automated search algorithm. First, by studying the structure of INLEC, we construct an automated search model of INLEC based on Mixed Integer Linear Programming to find five-round meet-in-the-middle distinguishers of INLEC. Then, we exploit the redundancy of INLEC's key schedule and select one good distinguisher according to the fewest guessed subkeys in the key recovery phase. Add one round and four rounds on the top and bottom of this distinguisher to construct a 11-round meet-in-the-middle attack path. Finally, we recover the master key with a time complexity of $2^{115.17}$ encryptions, a data complexity of 2^{61} chosen plaintexts, and a memory complexity of 2^{81} 64-bit blocks. This work represents the first third-party security analysis result for INLEC.

Keywords

INLEC, Meet-in-the-Middle Attack, Mixed Integer Linear Programming, Lightweight Block Cipher

Copyright © 2025 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着现代互联网技术的飞速发展以及现代信息泄露事件的频发，数据安全变得愈发重要。密码算法作为密码学的核心之一同样也是保护数据安全的核心技术，其被广泛应用于各种通信和存储系统当中。然而，传统的分组密码算法虽然在安全性上表现优异，但在资源受限的环境当中，它们的计算复杂性和资源消耗往往成为瓶颈。因此为了在资源受限的环境当中也能提供足够的安全性，轻量级分组密码算法应运而生。这类算法力求在能够在有限的计算能力、内存和电池寿命的使用场景中能够提供最大限度的安全性，解决了传统密码算法在低功耗、小尺寸设备中的应用瓶颈，为物联网、智能卡、RFID 标签等场景提供了高效且安全的加密解决方案。

近年来，随着技术的快速发展，轻量级分组密码算法，诸如 CRAFT [1]、LBlock [2]、PICO [3]、FESH [4]、FBC [5]等，都有着广泛的应用场景，并表现优秀。但是由于轻量级分组密码算法是应用于资源受限的环境中的分组密码，因此在设计之初往往为了追求高效的软硬件的实现效率，可能会在安全性上做出一定的妥协，寻求安全性和高效的软硬件实现的一个平衡点，而这可能成为密码算法保障数据安全的一个薄弱环节。因此对轻量级分组密码算法进行安全性分析非常重要，这是确保其在资源受限的环境中能够可靠运行的关键。只有通过严格的安全性研究评估，才能确保这些算法在实际运用中既能满足高效性需求，又能提供足够的安全保障。

混合整数线性规划(Mixed Integer Linear Programming, MILP)是一种数学优化方法，是线性规划(Linear Programming, LP)的扩展，在线性规划的基础上引入了整数变量，被广泛地应用于解决复杂的决策问题。MILP 最初于 2011 年由 Mouha 等[6]将其与密码分析方法相结合，研究者采用若干不等式来描述密码算法的各种运算，然后运用多元高次不等式求解算法来获取密钥信息，从而评估密码算法的安全性。该方法效率高且灵活，并且使用计算机辅助搜索和计算，显著提高了密码分析的效率，已成为广泛运用于密码学研究中。

INLEC 是 2024 年由 Feng 等[7]提出的基于 SPN 结构的轻量级分组密码算法，该算法分组长度为 64 比特，主密钥长度为 128 比特，目前暂未有第三方对其进行安全性分析。为了使得 INLEC 在实际系统中能够保障信息的安全，因此需要对 INLEC 算法的安全性进行深入分析。INLEC 密码设计者使用 MILP 自

动化搜索工具，对 INLEC 各个部分进行建模，证明了 15 轮的 INLEC 密码算法足以抵御不可能差分攻击、差分攻击和线性攻击，此外通过分析证明了其对代数攻击、侧信道攻击的安全性。但目前的研究尚未有针对 INLEC 的中间相遇分析，而中间相遇分析[8]已经成功地评估了众多知名分组密码算法的安全性，如 AES [9]、Midori [10]等，因此有必要进一步评估 INLEC 算法抵抗中间相遇攻击的能力。

本文利用 MILP 自动化搜索算法，提出了 11 轮 INLEC 的中间相遇分析。通过研究 INLEC 算法加密过程的各个操作，建立 MILP 自动化搜索模型，并使用 Gurobi 进行求解，得到数条 5 轮的 INLEC 中间相遇区分器，并选择需要猜测密钥位最少的一条中间相遇区分器，往前添加 2 轮，往后添加 4 轮构成 11 轮的 INLEC 中间相遇攻击路径，并根据该路径恢复出全部的密钥。整个攻击过程需要时间复杂度为 $2^{115.17}$ 次加密，数据复杂度为 2^{61} 个选择明文，存储复杂度为 2^{81} 个 64 比特块。本文评估了 INLEC 抵抗中间相遇攻击的能力，是 INLEC 的安全性研究的重要补充。

2. 预备知识

2.1. 符号说明

1) $RC_i, i \in \{1,2,3,\cdots,14\}$ 表示第 i 轮的轮常数；

2) $K\{i\}, i \in \{0,1,\cdots,127\}$ 表示将主密钥 K 的第 i 个比特；

3) $X \| Y$ 表示 X, Y 的级联；

4) $\ll i$ 表示循环向左移动 i 比特；

5) $RK_i, i \in \{1,\cdots,14\}$ 表示第 i 轮的轮密钥；

6) $RK_i[j], i \in \{1,\cdots,14\}, j \in \{0,\cdots,15\}$ 表示第 i 轮的第 j 个半字节；

7) $X_i', Y_i', Z_i', W_i', i \in \{9,\cdots,15\}$ 分别表示第 i 轮的 P 置换、单元替换、轮密钥加以及列混合之前的中间状态值的第 j 位置的状态值；

8) $X_i'[j], Y_i'[j], Z_i'[j], W_i'[j], i \in \{9,\cdots,15\}, j \in \{0,\cdots,15\}$ 分别表示第 i 轮的 P 置换、单元替换、轮密钥加以及列混合之前的中间状态值；

9) $X_i, Y_i, Z_i, W_i, i \in \{1,2,\cdots,8\}$ 分别表示第 i 轮的 P 置换、单元替换、列混合以及轮密钥加之前的中间状态值；

10) $X_i[j], Y_i[j], Z_i[j], W_i[j], i \in \{1,2,\cdots,8\}, j \in \{0,\cdots,15\}$ 分别表示第 i 轮的 P 置换、单元替换、列混合以及轮密钥加之前的中间状态值的第 j 位置的状态值。

2.2. INLEC 算法介绍

INLEC 算法采用 SPN 结构，明文长度为 64 比特，主密钥长度为 128 比特，一共 15 轮。该算法加密过程采用了两个不同的轮函数，分别为 F1 和 F2，其中第 1 至第 8 轮采用 F1 轮函数，第 9 至第 14 轮采用 F2 轮函数，最后一轮仅仅进行 P 置换和单元替换操作。INLEC 两个轮函数所包含的加密操作相同，其中轮函数 F1 加密操作顺序为 P 置换、单元替换、列混合、轮密钥加，而轮函数 F2 则为 P 置换、单元替换、轮密钥加、列混合。图 1 为轮函数 F1 的加密示意图。加密过程中涉及的操作具体定义如下所示。

1) P 置换(Permute-Nibble, PN)：将对合置换的P应用于中间状态，从而实现对中间状态的重新排序，具体的置换关系如表 1 所示。

2) 单元替换(SubCell, SC)：SC 是一种非线性变换，通常选用 S 盒来进行该操作，INLEC 选取的 S 盒是 4 × 4 的 S 盒，即 S 盒输入和输出都为 4 比特，因此将 S 盒按序应用于密码中间状态的每个单元，记为 $x[i] \to S(x[i])(0 \le i \le 15)$。S 盒具体如表 2 所示。

3) 轮密钥加(AddRoundKey, ARK)：将 64 比特的轮子密钥和密码的中间状态值进行异或。

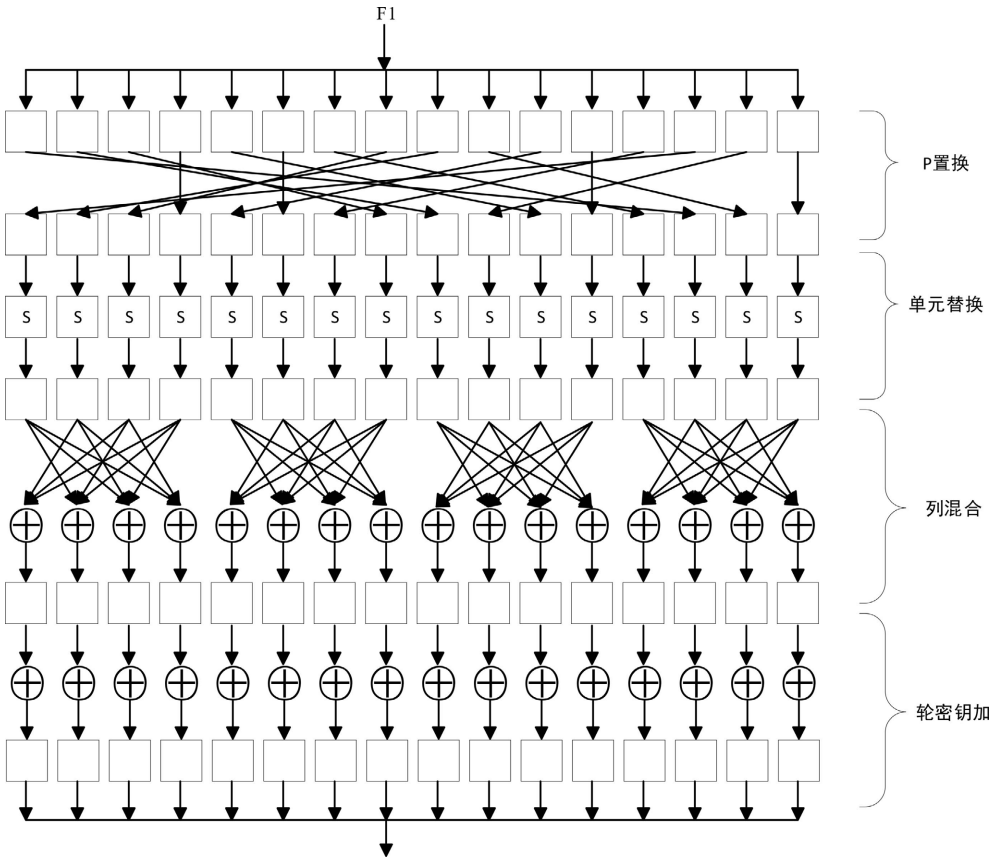

Figure 1. The round function F1 of INLEC
图 1. INLEC 加密轮函数 F1 图

Table 1. Permute-nibble of INLEC algorithm
表 1. INLEC 的 P 置换表

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$P(x)$	d	8	7	4	a		c	2	1	e	4	b	6	0	9	f

Table 2. *S*-box of INLEC algorithm
表 2. INLEC 算法的 *S* 盒

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	0	d	b	e	7	5	6	4	f	9	c	2	a	1	3	8

4) 列混合(MixColumn, MC)：将中间状态矩阵左乘二进制矩阵 M，

$$M=\begin{pmatrix}0&1&1&1\\1&0&1&1\\1&1&0&1\\1&1&1&0\end{pmatrix}$$

2.3. INLEC 的密钥编排算法

INLEC 的主密钥长度为 128 比特，记为 $IK=k_0\|k_1\|k_2\|\cdots\|k_{126}\|k_{127}$。使用主密钥通过密钥编排算法生

成 64 比特的轮子密钥，具体的密钥编排算法如算法 1 所示。

算法 1. INLEC 的密钥编排算法

输入：主密钥 IK

输出：轮密钥 RK_i，

for $i=1$ to $R-1$ do

$K \leftarrow K \ll 7$.；#循环向左移动 7 比特

$K \leftarrow K \ll 11$；#循环向左移动 11 比特

$\left[K(120)\|K(121)\|K(122)\|K(123)\right] \leftarrow S\left[K(120)\|K(121)\|K(122)\|K(123)\right]$；#第 120 至 123 比特经过 S 盒

$\left[K(124)\|K(125)\|K(126)\|K(127)\right] \leftarrow S\left[K(124)\|K(125)\|K(126)\|K(127)\right]$；#第 124 至 127 比特经过 S 盒

$\left[K(116)\|K(117)\|K(118)\|K(119)\right] \leftarrow S\left[K(116)\|K(117)\|K(118)\|K(119)\right] \oplus RC_i$；#将第 116 比特至 119 比特与轮常数进行异或运算

$RK_i \leftarrow k_{64}\|k_{65}\|k_{66}\|\cdots\|k_{126}\|k_{127}$；

End for

Return RK_i；

性质 1. 对于给定的 S 盒，若其非零的输入差分和输出差分分别为 Δx 和 Δy，则平均有一个 x 值满足等式 $S(x)\oplus S(x\oplus\Delta x)=\Delta y$，该性质同样适用于 S^{-1}。其中 S^{-1} 表示 S 盒的逆。

3. 11 轮的 INLEC 中间相遇分析

3.1. INLEC 的密钥编排方案冗余性研究

根据对 INLEC 密钥扩展方案的深入研究，可以求得如下性质，这些性质将用于后续的中间相遇的攻击过程当中。

性质 2. RK_{10} [7, 8]可由主密钥 K {14, 15, 16, 17, 18, 19, 20, 21, 22, 23}确定；RK_{11} [0, 1, 2, 3, 4]由主密钥 K {6~25}确定，因此 RK_{10} [7, 8]可由 RK_{11} [0, 1, 2, 3, 4]推导而来。

性质 3. RK_9 [6]可由主密钥 K {122, 123, 124, 125}确定。且由 RK_{10} [1, 2]可由主密钥 K {118, 119, 120121, 122, 123, 125, 126, 127}确定，因此可由 RK_{10} [1, 2]推导出 RK_9 [6]；RK_9 [9]可由主密钥 K {6, 7, 8, 9, 10, 11, 12, 13}确定，RK_{11} [0, 1, 2]可由主密钥 K {6~17}确定，因此则 RK_9 [9]可由 RK_{11} [0, 1, 2]推导而来。

性质 4. RK_8 [12, 13, 14]可由主密钥 K {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}，且由先前性质可知，RK_{10}[3, 4]和 RK_{11} [0, 1, 2]可推导出主密钥 K {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17}，因此 RK_8 [12, 13, 14]可由 RK_{10} [3, 4]和 RK_{11} [0, 1, 2]推导而来。

性质 5. RK_1 [9]可由主密钥 K {118, 119, 120, 121}确定，RK_{10} [0, 1, 2]可由主密钥 K {116~127}确定，因此 RK_1 [9]可由 RK_{10} [0, 1, 2]推导而来。

3.2. INLEC 基于 MILP 的模型

INLEC 的轮函数一共有 4 种操作，分别是 P 置换、单元替换、列混合和轮密钥加，其中轮密钥加和单元替换部分并不会改变差分传播过程当中的中间状态值，因此在本次建模中轮函数的 4 种操作不考虑轮密钥加和单元替换这两个操作，仅考虑 P 置换和列混合的建模。具体的约束条件如下：

1) P 置换操作。对于 INLEC 而言，P 置换是基于半字节级别的替换，令 $x[i], y[i], i\in\{0,\cdots,15\}$ 分别表

示 P 置换的输入中间状态和输出中间状态，具体如公式(1)所示。

$$\begin{cases} x[0]-y[13]=0 \\ x[1]-y[8]=0 \\ x[2]-y[7]=0 \\ x[3]-y[3]=0 \end{cases} \begin{cases} x[8]-y[1]=0 \\ x[9]-y[14]=0 \\ x[10]-y[4]=0 \\ x[11]-y[11]=0 \end{cases} \\ \begin{cases} x[4]-y[10]=0 \\ x[5]-y[5]=0 \\ x[6]-y[12]=0 \\ x[7]-y[2]=0 \end{cases} \begin{cases} x[12]-y[6]=0 \\ x[13]-y[0]=0 \\ x[14]-y[9]=0 \\ x[15]-y[15]=0 \end{cases} \tag{1}$$

2) 列混合运算。假设，$x[i],y[i],i\in\{0,\cdots,15\}$ 分别是列混合过程的输入中间状态以及输出中间状态，则列混合的变换如公式(2)所示：

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

因此使用公式(3)来描述列混合运算的加密过程。

$$\begin{cases} 3y[0]-x[1]-x[2]-x[3]\geq 0 \\ x[1]+x[2]+x[3]-y[0]\geq 0 \\ 3y[1]-x[0]-x[2]-x[3]\geq 0 \\ x[0]+x[2]+x[3]-y[1]\geq 0 \\ 3y[2]-x[0]-x[1]-x[3]\geq 0 \\ x[0]+x[1]+x[3]-y[2]\geq 0 \\ 3y[3]-x[0]-x[1]-x[2]\geq 0 \\ x[0]+x[1]+x[2]-y[3]\geq 0 \end{cases} \begin{cases} 3y[8]-x[9]-x[10]-x[11]\geq 0 \\ x[9]+x[10]+x[11]-y[8]\geq 0 \\ 3y[9]-x[8]-x[10]-x[11]\geq 0 \\ x[8]+x[10]+x[11]-y[9]\geq 0 \\ 3y[10]-x[8]-x[9]-x[11]\geq 0 \\ x[8]+x[9]+x[11]-y[10]\geq 0 \\ 3y[11]-x[8]-x[9]-x[10]\geq 0 \\ x[8]+x[9]+x[10]-y[11]\geq 0 \end{cases} \\ \begin{cases} 3y[4]-x[5]-x[6]-x[7]\geq 0 \\ x[5]+x[6]+x[7]-y[4]\geq 0 \\ 3y[5]-x[4]-x[6]-x[7]\geq 0 \\ x[4]+x[6]+x[7]-y[5]\geq 0 \\ 3y[6]-x[4]-x[5]-x[7]\geq 0 \\ x[4]+x[5]+x[7]-y[6]\geq 0 \\ 3y[7]-x[4]-x[5]-x[6]\geq 0 \\ x[4]+x[5]+x[6]-y[7]\geq 0 \end{cases} \begin{cases} 3y[12]-x[13]-x[14]-x[15]\geq 0 \\ x[13]+x[14]+x[15]-y[12]\geq 0 \\ 3y[13]-x[14]-x[15]-x[12]\geq 0 \\ x[12]+x[14]+x[15]-y[13]\geq 0 \\ 3y[14]-x[12]-x[13]-x[15]\geq 0 \\ x[12]+x[13]+x[15]-y[14]\geq 0 \\ 3y[15]-x[12]-x[13]-x[14]\geq 0 \\ x[12]+x[13]+x[14]-y[15]\geq 0 \end{cases} \tag{3}$$

3) 差分状态的约束。假设 $IC[i],i\in\{0,\cdots,15\}$ 为加密方向的状态差分，$OC[i],i\in\{0,\cdots,15\}$ 为解密方向的状态差分，$DL[i],i\in\{0,\cdots,15\}$ 为 $IC[i]$ 和 $OC[i]$ 的交集，根据上述的描述，由于无需对轮密钥加和单元替换进行建模，因此每轮的 INLEC 的差分状态同样可以分成 3 个部分，分别是每轮的输入差分状态，经过 P 置换后的差分状态以及经过列混合之后的差分状态，具体如公式(4)所示。

3.3. 基于 MILP 的中间相遇区分器搜索算法

本文采用 Demirci 和 Selçuk 提出的中间相遇攻击方式。该分析方法具体为将加密算法E分成了三个部分，即 E_0,E_1,E_2，它们相对应的部分密钥分别为 k_0,k_1,k_2。在离线阶段，在中间的 E_1 部分构建多轮的区

分器，并且计算出中间状态的有序序列值，猜测 k_0, k_2 的部分值来进行部分加密和解密，如果求解出来的中间值与计算表中的值匹配，则猜测的可能正确。具体的攻击过程如图 2 所示。

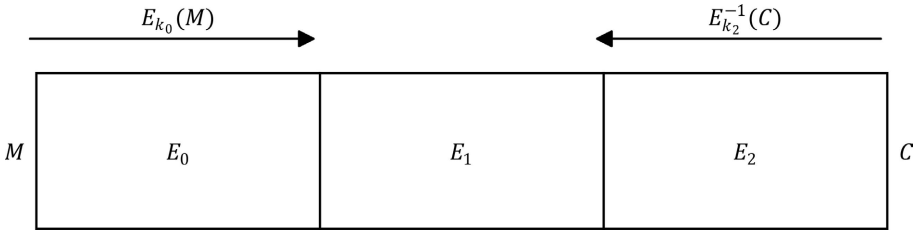

Figure 2. DS-MITM attack process
图 2. DS-MITM 攻击过程

$$
\begin{cases}
IC[0]-DL[0]\geq 0\\ OC[0]-DL[0]\geq 0\\ DL[0]-OC[0]-IC[0]\geq -1\\ IC[1]-DL[1]\geq 0\\ OC[1]-DL[1]\geq 0\\ DL[1]-OC[1]-IC[1]\geq -1\\ IC[2]-DL[2]\geq 0\\ OC[2]-DL[2]\geq 0\\ DL[2]-OC[2]-IC[2]\geq -1\\ IC[3]-DL[3]\geq 0\\ OC[3]-DL[3]\geq 0\\ DL[3]-OC[3]-IC[3]\geq -1\\ IC[4]-DL[4]\geq 0\\ OC[4]-DL[4]\geq 0\\ DL[4]-OC[4]-IC[4]\geq -1\\ IC[5]-DL[5]\geq 0
\end{cases}
\begin{cases}
OC[5]-DL[5]\geq 0\\ DL[5]-OC[5]-IC[5]\geq -1\\ IC[6]-DL[6]\geq 0\\ OC[6]-DL[6]\geq 0\\ DL[6]-OC[6]-IC[6]\geq -1\\ IC[7]-DL[7]\geq 0\\ OC[7]-DL[7]\geq 0\\ DL[7]-OC[7]-IC[7]\geq -1\\ IC[8]-DL[8]\geq 0\\ OC[8]-DL[8]\geq 0\\ DL[8]-OC[8]-IC[8]\geq -1\\ IC[9]-DL[9]\geq 0\\ OC[9]-DL[9]\geq 0\\ DL[9]-OC[9]-IC[9]\geq -1\\ IC[10]-DL[10]\geq 0\\ OC[10]-DL[10]\geq 0
\end{cases}
\begin{cases}
DL[10]-OC[10]-IC[10]\geq -1\\ IC[11]-DL[11]\geq 0\\ OC[11]-DL[11]\geq 0\\ DL[11]-OC[11]-IC[11]\geq -1\\ IC[12]-DL[12]\geq 0\\ OC[12]-DL[12]\geq 0\\ DL[12]-OC[12]-IC[12]\geq -1\\ IC[13]-DL[13]\geq 0\\ OC[13]-DL[13]\geq 0\\ DL[13]-OC[13]-IC[13]\geq -1\\ IC[14]-DL[14]\geq 0\\ OC[14]-DL[14]\geq 0\\ DL[14]-OC[14]-IC[14]\geq -1\\ IC[15]-DL[15]\geq 0\\ OC[15]-DL[15]\geq 0\\ DL[15]-OC[15]-IC[15]\geq -1
\end{cases}
\tag{4}
$$

本文首先采用 3.2 小节中所建立的 INLEC 的模型，并且结合 INLEC 自动化搜索中间相遇区分器算法，即算法 2 生成自动化搜索模型，并采用相应的求解器进行求解，最终得到数条 5 轮的中间相遇区分器。对搜索到的中间相遇区分器分别往前和往后添加若干轮形成相应的攻击路径，记录下各个区分器所能攻击到的最长轮数和相应复杂度的情况，构成有序序列所需的半字节个数等等信息，最终对这些区分器进行筛选，选取出攻击效果最好的区分器作为本次攻击采用的区分器，表 3 中列出了搜索到的效果最好的 4 条中间相遇区分器的具体情况。根据表 3 的内容可知 Δ_1 的效果最好，因此选取 Δ_1 为本文中间相遇攻击的区分器。

算法 2. INLEC 自动化搜索中间相遇区分器算法

输入：输入差分集合 ΔI，输出差分集合 ΔO，INLEC 算法的不等式约束

输出：r 轮的可能的中间相遇区分器集合 Distlist 列表 IC；存储将输入差分部分加密后每轮的中间状态差分列表 OC；存储将输出差分部分解密后每轮的中间状态差分

for $i=1$ to R do

续表

输入 INLEC 算法的不等式约束

将目标函数设置为 IC 和 OC 的交集

end for

for $i \in \Delta I$ do

$IC = getIC(i)$ #将输入差分部分加密，记录每轮的中间状态差分，最后返回并且存储在列表 IC 当中

for $j \in \Delta O$ do

$OC = getOC(j)$ #将输出差分部分解密，记录每轮的中间状态差分，最后返回并且存储在列表 OC 当中。

Distlist = IC AND OC#选取 IC 和 OC 中保存的每轮中的中间状态差分值的交集

end for

end for

Return 中间相遇区分器集合 Distlist

Table 3. IVLBC meet-in-the-middle distinguishers (part)
表 3. 搜索到的部分 INLEC 中间相遇区分器(部分)

编号	中间相遇区分器	轮数	有序序列需要猜测半字节个数	攻击轮数
Δ_1	$\left[1,0_{(14)},1\right]\to\left[0_{(15)},1\right]$	5	20	11
Δ_2	$\left[0_{(8)},1,0_{(4)},1,0_{(2)}\right]\to\left[1,0_{(15)}\right]$	5	19	10
Δ_3	$\left[1,0_{(5)},1,0_{(9)}\right]\to\left[0_{(9)},1,0_{(6)}\right]$	5	15	10
Δ_4	$\left[0_{(6)},1,0_{(9)}\right]\to\left[0_{(8)},1,0,1,0_{(5)}\right]$	5	16	10

在表 3 中 0 表示该半字节的差分为零，1 则表示差分为非零，$0_{(i)}$ 表示连续 i 个比特为 0。

3.4. 5 轮的 INLEC 中间相遇区分器

根据上一小节中的描述，选取表 3 中的 Δ_1 作为本次中间相遇攻击的区分器，该区分器的具体情况如图 3 所示。为了描述搜索到的区分器，需要定义一个 δ 集，记为 $\left(W_2(0),W_2(1),\cdots,W_2\left(2^8-1\right)\right)$，选取其中 33 个 $W_i(i),i\in\{0,\cdots,32\}$，经过 5 轮的 INLEC 加密过后得到相应的有序序列。如果该集合中该集合中有一个元素满足图 3 所示的截断差分，则该有序序列共 2^{80} 个值。将这些有序序列存放在一个预计算表 H_0 中。该 5 轮中间相遇区分器的性质如下：

性质 6 将 $W_2[0,15]$ 处的 δ 集 $\left(W_2(0),W_2(1),\cdots,W_2\left(2^8-1\right)\right)$ 进行 5 轮的 INLEC 部分加密，如果该集合中有一个满足图 3 所示的截断差分特征，则与该 δ 集相关的有序序列 $\Delta Z_8[15]$ 的可能性为 2^{80} 个，由 $Y_3[13,15],Y_4[0,6,9,15],\Delta Z_8[15],Z_8[15],Z_7[12,13,14],Z_6[1\sim5,7,8,10,11]$ 共计 20 个半字节确定。

证明：根据 $\Delta W_2[0,15]$ 已知，可以推导出 $\Delta Y_3[13,15]$ 的值，并猜测 $Y_3[13,15]$ 的值，可以计算出 $\Delta Z_3[13,15]$ 以及 $Z_3[13,15]$ 的值。由于轮密钥加和列混合是线性操作不会改变中间状态的差分值，所以可以推导出 $\Delta Y_4[0,6,9,15]$ 的值，类似地猜测 $Y_4[0,6,9,15]$ 的值，可以推导出 $\Delta Z_4[0,6,9,15]$ 和 $Z_4[0,6,9,15]$ 的值，随后可以计算出 $\Delta Y_5[0\sim11]$ 的值。然后从输出端进行反向推导，猜测 $\Delta Z_8[15],Z_8[15]$，由此根据性质 1 可以推导出 $\Delta X_8[15]$，随后猜测 $Z_7[12,13,14]$ 的值，可以推导出 $Y_7[12,13,14]$ 和 $\Delta Y_7[12,13,14]$ 的值，再继续往前计算可以求出 $\Delta Z_6[1\sim5,7,8,10,11]$ 的值，继续猜测 $Z_6[1\sim5,7,8,10,11]$ 的值，所以可以推导出 $\Delta Z_5[0\sim11]$ 的值。根据性质 1，平均可以得到一个值 X 满足 $\Delta Z_5[0\sim11]=S(X)\oplus S\left(\Delta Y_5[0\sim11]\oplus X\right)$，证毕。

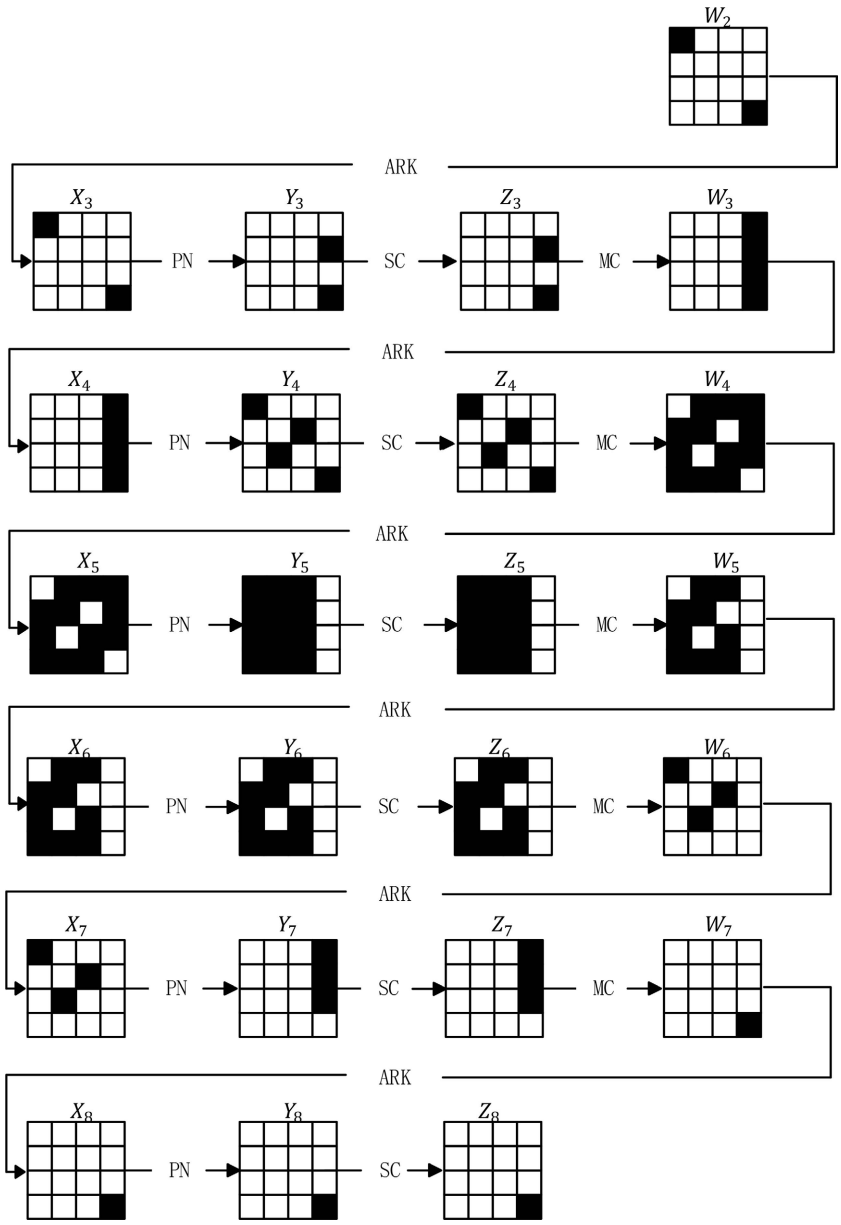

Figure 3. 5-Round meet in the middle distinguisher
图 3. 5 轮 INLEC 中间相遇区分器

3.5. 11 轮的 INLEC 中间相遇攻击过程

在图 3 所示的 5 轮中间相遇区分器的基础上往前添加 2 轮，往后添加 4 轮，构成 11 轮的 INLEC 中间相遇攻击路径，该路径具体情况如图 4 所示。整个攻击包含预计算阶段和在线阶段两部分，其中预计算阶段通过构建预计算表来方便在线阶段提取相对应的密钥。

3.5.1. 预计算阶段

构建一个预计算表 H_0 来存储 2^{80} 个有序序列 $\Delta Z_8[15]$ 的值。并且还对分析轮当中的部分轮数建立起预计算表，以便在线阶段获得相关的轮子密钥，降低攻击的时间复杂度。

1) 表 $H_1\{RK_{11}[3,7,8,13]\}$：猜测 $\Delta W'_{10}[1,2,3]$ 和 $X'_{11}[0,1,2,3]$ 的全部 2^{28} 个可能值，由此计算出

$\Delta X'_{11}[0,1,2,3]$，$Y'_{11}[3,7,8,13]$，$\Delta Y'_{11}[3,7,8,13], Z'_{11}[3,7,8,13], \Delta W'_{11}[3,7,8,13], W'_{10}[1,2,3]$，以 $\Delta W'_{11}[3,7,8,13]$ 为索引，将 $\Delta W'_{10}[1,2,3] \| W'_{10}[1,2,3] \| Z'_{11}[3,7,8,13]$ 存储在 H_1 中。表 H_1 中有 2^{16} 行，平均每行有 2^{12} 个值。

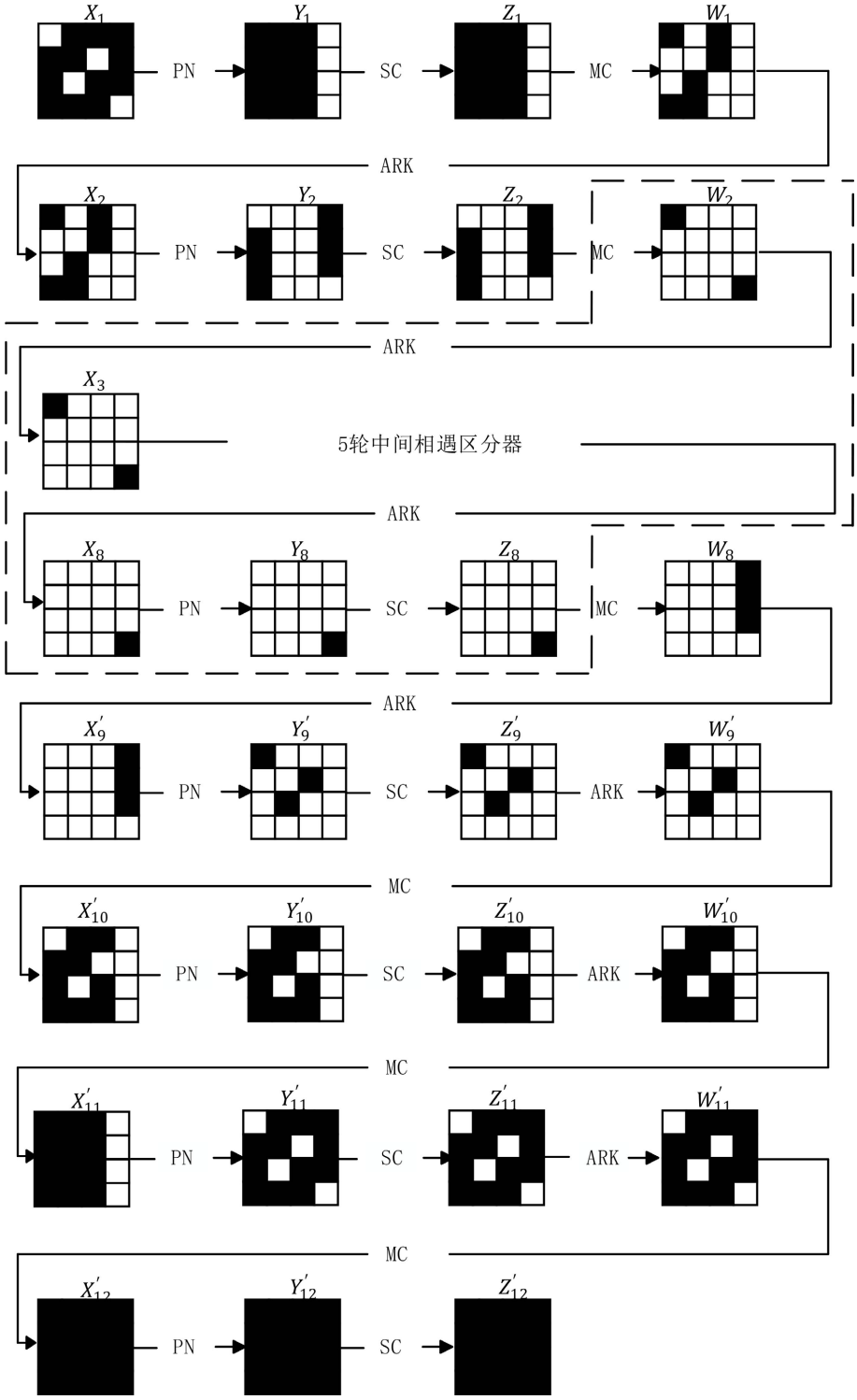

Figure 4. 11 rounds of meet-in-the-middle attack process
图 4. 11 轮 INLEC 中间相遇攻击过程

2) 表 $H_2\{RK_{11}[2,5,10,12]\}$：猜测 $\Delta W'_{10}[4,5,7]$ 和 $X'_{11}[4,5,6,7]$ 的全部 2^{28} 个可能值，由此计算出 $\Delta X'_{11}[4,5,6,7]$，$Y'_{11}[2,5,10,12]$，$\Delta Y'_{11}[2,5,10,12], Z'_{11}[2,5,10,12], \Delta W'_{11}[2,5,10,12], W'_{10}[4,5,7]$ 以 $\Delta W'_{11}[2,5,10,12]$ 为索引，将 $\Delta W'_{10}[4,5,7]\|W'_{10}[4,5,7]\|Z'_{11}[2,5,10,12]$ 存储在 H_2 中。表 H_2 中有 2^{16} 行，平均每行有 2^{12} 个值。

3) 表 $H_3\{RK_{11}[1,4,11,14]\}$：猜测 $\Delta W'_{10}[8,10,11]$ 和 $X'_{11}[8,9,10,11]$ 的全部 2^{28} 个可能值，由此计算出 $\Delta X'_{11}[8,9,10,11]$，$Y'_{11}[1,4,11,14]$，$\Delta Y'_{11}[1,4,11,14], Z'_{11}[1,4,11,14], \Delta W'_{11}[1,4,11,14], W'_{10}[8,10,11]$, 以 $\Delta W'_{11}[1,4,11,14]$ 为索引，将 $\Delta W'_{10}[8,10,11]\|W'_{10}[8,10,11]\|Z'_{11}[1,4,11,14]$ 存储在 H_3 中。表 H_3 中有 2^{16} 行，平均每行有 2^{12} 个值。

4) 表 $H_4\{RK_{10}[2,5,10]\}$：猜测 $\Delta W'_9[6]$ 和 $X'_{10}[4,5,7]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X'_{10}[4,5,7]$，$Y'_{10}[2,5,10]$，$\Delta Y'_{10}[2,5,10], Z'_{10}[2,5,10], \Delta W'_{10}[2,5,10], W'_9[6]$ 以 $\Delta W'_{10}[2,5,10]$ 为索引，将 $\Delta W'_9[6]\|W'_9[6]\|Z'_{10}[2,5,10]$ 存储在 H_4 中。表 H_4 中有 2^{12} 行，平均每行有 2^4 个值。

5) 表 $H_5\{RK_{10}[1,4,11]\}$：猜测 $\Delta W'_9[9]$ 和 $X'_{10}[8,10,11]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X'_{10}[8,10,11]$，$Y'_{10}[1,4,11]$，$\Delta Y'_{10}[1,4,11], Z'_{10}[1,4,11], \Delta W'_{10}[1,4,11], W'_9[9]$, 以 $\Delta W'_{10}[1,4,11]$ 为索引，将 $\Delta W'_9[9]\|W'_9[9]\|Z'_{10}[1,4,11]$ 存储在 H_5 中。表 H_5 中有 2^{12} 行，平均每行有 2^4 个值。

6) 表 $H_6\{RK_{10}[3,7,8]\}$：猜测 $\Delta W'_9[0]$ 和 $X'_{10}[1,2,3]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X'_{10}[1,2,3]$，$Y'_{10}[3,7,8]$，$\Delta Y'_{10}[3,7,8], Z'_{10}[3,7,8], \Delta W'_{10}[3,7,8], W'_9[0]$，以 $\Delta W'_{10}[3,7,8]$ 和 $Z'_{10}[7,8]$ 为索引，将 $\Delta W'_9[0]\|W'_9[0]\|Z'_{10}[7,8]$ 存储在 H_6 中。表 H_6 中有 2^{20} 行，平均每 2^4 行有 1 个值。

7) 表 $H_7\{RK_9[0,6,9]\}$：猜测 $\Delta X'_9[12,13,14]$ 和 $X'_9[12,13,14]$ 的全部 2^{24} 个可能值，由此计算出 $Y'_9[0,6,9]$，$\Delta Y'_9[0,6,9], Z'_9[0,6,9], \Delta W'_9[0,6,9], W_8[12,13,14]$，以 $\Delta W'_9[0,6,9]$ 和 $Z'_9[6,9]$。为索引，将 $\Delta X'_9[12,13,14]\|\Delta W_8[12,13,14]\|Z'_9[0,6,9]\|X'_9[12,13,14]$ 存储在 H_7 中。表 H_7 中有 2^{20} 行，平均每行有 2^4 个值。

8) 表 $H_8\{RK_8[12,13,14]\}$：猜测 $\Delta Z_8[15]$ 和 $W_8[12,13,14]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X'_9[12,13,14], \Delta W_8[12,13,14]$, 以 $W_8[12,13,14]$ 和 $\Delta X'_9[12,13,14]$ 为索引，将 $W_8[12,13,14]$ 存储在 H_8 中。表 H_8 中有 2^{24}。行，平均每 2^8 行有 1 个值。

9) 表 $H_9\{RK_1[3,7,8]\}$：猜测 $\Delta W_2[0]$ 和 $Z_2[1,2,3]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X_2[3,7,8]$，$X_2[3,7,8]$，$\Delta Y_2[1,2,3], W_2[0], \Delta W_1[3,7,8]$ 以 $\Delta W_1[3,7,8]$ 为索引，将 $X_2[3,7,8]\|\Delta W_2[0]\|W_2[0]$ 存储在 H_9 中。表 H_9 中有 2^{12} 行，平均每行有 2^4 个值。

10) 表 $H_{10}\{RK_1[0,6,9]\}$：猜测 $\Delta W_2[15]$ 和 $Z_2[12,13,14]$ 的全部 2^{16} 个可能值，由此计算出 $\Delta X_2[0,6,9]$，$X_2[0,6,9]$，$\Delta Y_2[12,13,14], W_2[15], \Delta W_1[0,6,9]$，和 $X_2[9]$ 为索引，将 $X_2[0,6,9]\|\Delta W_2[15]\|W_2[15]$ 存储在 H_{10} 中。表 H_{10} 中有 2^{16} 行，平均每行有 1 个值。

3.5.2. 在线阶段

本节中构成的中间相遇攻击路径的概率为 2^{-108}，具体分析如下：第一，差分 $\Delta Z_5[0\sim11]$ 经过列混淆操作之后生成了 $\Delta W_5[1\sim5,7,8,10,11]$ (记作 $\Delta Z_5[0\sim11]\to\Delta W_5[1\sim5,7,8,10,11]$)，该操作的概率，大小为 2^{-12}；第二，当 $\Delta Z_6[1\sim5,7,8,10,11]\to\Delta W_6[0,6,9]$ 时产生的概率大小为 2^{-36}；第三，当 $\Delta Z_7[12,13,14]\to\Delta W_7[15]$ 时，所产生的概率为 2^{-12}；第四，当 $\Delta Z_2[1,2,3,12,13,15]\to\Delta W_2[0,15]$ 时，所产生的概率为 2^{-24}；第五，当 $\Delta Z_1[0\sim11]\to\Delta W_1[0,3,6,7,8,9]$ 时，所产生的概率为 2^{-24}。因此，该截断差分特征所对应的概率大小为 $2^{-12-36-12-24-24}=2^{-108}$。

为了找到正确的消息对，推导出使其满足每对截断差分特征的轮子密钥。然后根据 δ 集，计算得到有序序列 $\Delta X_8[15]$ 的值，并判断该序列是否与预计算阶段构建的表 H_0 内的条目匹配。在线阶段的攻击过程描述如下：

1. 定义一个包含 2^{48} 个明文的结构，其中 $X_1[1,2,3,4,5,7,8,10,11,12,13,14]$ 取所有可能的 2^{48} 个值，其余的 4 个半字节被固定为一些常量。因此，一个结构可以生成 $2^{48}\times\left(2^{48}-1\right)/2\approx 2^{95}$ 个明文对，且每个对都满足明文差分。现需使用 2^{13} 个这样的结构来生成 $2^{95}\times 2^{13}=2^{108}$ 个明文对。在这 2^{108} 个明文对中，约有 1 个明文对满足其截断差分特征，因为该截断差分特征的概率为 2^{-108}。

2. 对选择的 2^{108} 个明文对进行加密并且筛选出满足 $W'_{11}[0,6,9,15]$ 这 4 个位置为固定值的密文对，由此共有 $2^{108-16}=2^{92}$ 个明密文对，利用这些密文对来计算出密文对差分值；同时对明文对继续进行筛选，筛选出满足 $W_1[1,2,4,5,10,11]$ 这 6 个位置为固定值的明文对，最后共有 $2^{92-24}=2^{68}$ 个明密文对。然后对这些明文－密文对执行以下的操作，来恢复可能的候选密钥。

1) 根据密文对状态值，计算得到 $W'_{11}[3,7,8,13]$ 和 $\Delta W'_{11}[3,7,8,13]$ 的值，以 $\Delta W'_{11}[3,7,8,13]$ 为索引访问 H_1，得到 $Z'_{11}[3,7,8,13]$，并通过计算可以获得 $RK_{11}[3,7,8,13]$ 共计 2^{12} 个可能值。

2) 根据密文对状态值，可以计算得到 $W'_{11}[2,5,10,12]$ 和 $\Delta W'_{11}[2,5,10,12]$ 的值，以 $\Delta W'_{11}[2,5,10,12]$ 为索引访问 H_2，得到 $Z'_{11}[2,5,10,12]$，并通过计算可以获得 $RK_{11}[2,3,5,7,8,10,12,13]$ 共计 $2^{12+12}=2^{24}$ 个可能值。

3) 根据密文对状态值，可以计算得到 $W'_{11}[1,4,11,14]$ 和 $\Delta W'_{11}[1,4,11,14]$ 的值，以 $\Delta W'_{11}[1,4,11,14]$ 为索引访问 H_3，得到 $Z'_{11}[1,4,11,14]$，并通过计算可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]$ 共计 $2^{24+12}=2^{36}$ 个可能值

4) 根据先前对表 H_1,H_2,H_3 的查询结果，可以计算得到 $W'_{10}[2,5,10]$ 和 $\Delta W'_{10}[2,5,10]$ 的值，以 $\Delta W'_{10}[2,5,10]$ 为索引访问 H_4，得到 $Z'_{10}[2,5,10]$，并通过计算可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[2,5,10]$ 共 $2^{36+4}=2^{40}$ 个可能值。

5) 根据先前对表 H_1,H_2,H_3 的查询结果，可以计算得到 $W'_{10}[1,4,11]$ 和 $\Delta W'_{10}[1,4,11]$ 的值，以 $\Delta W'_{10}[1,4,11]$ 为索引访问 H_5，得到 $Z'_{10}[1,4,11]$，并通过计算可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[1,2,4,5,10,11]$ 共 $2^{40+4}=2^{44}$ 个可能值。

6) 根据先前对表 H_1,H_2,H_3 的查询结果，可以得到 $W'_{10}[3,7,8]$ 和 $\Delta W'_{10}[3,7,8]$ 的值，并且根据性质 2 可知 $RK_{10}[7,8]$ 可由 $RK_{11}[0,1,2,3,4]$，且在先前的步骤中猜测了 $RK_{11}[1,2,3,4]$，因此猜测 $RK_{11}[0]$ 的全部 2^4 个可能值，随后根据等式 $RK_{10}[7,8]\oplus W'_{10}[7,8]=Z'_{10}[7,8]$，得到 $Z'_{10}[7,8]$，以 $\Delta W'_{10}[3,7,8]$ 和 $Z'_{10}[7,8]$ 为索引访问 H_6，每 2^4 行得到 1 个值，最终可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[1\sim 5,7,8,10,11]$ 共 $2^{44+4-4}=2^{44}$ 个可能值。

7) 根据先前对表 H_4，H_7 和 H_5 的查询可以得到 $\Delta W'_9[0,6,9]$ 和 $W'_9[0,6,9]$，根据性质 3 可知，$RK_{10}[1,2]$ 可推导出 $RK_9[6]$，$RK_9[9]$ 可由 $RK_{11}[0,1,2]$ 推导出，因为 $RK_{11}[0,1,2]$ 在先前的步骤中都已猜测，由此可以获得 $RK_9[6,9]$ 的值，根据 $RK_9[6,9]\oplus W'_9[6,9]=Z'_9[6,9]$，计算出 $Z'_9[6,9]$，以 $\Delta W'_9[0,6,9]$ 和 $Z'_9[6,9]$ 为索引访问 H_7，每行有 2^4 个值，最终可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[1\sim 5,7,8,10,11]\|RK_9[0,6,9]$ 共 $2^{44+4}=2^{48}$ 个可能值。

8) 根据先前对表 H_7 的查询可以得到 $\Delta X'_9[12,13,14]$ 和 $X'_9[12,13,14]$，根据性质 4 可知，$RK_8[12,13,14]$ 可由 $RK_{10}[3,4]$ 和 $RK_{11}[0,1,2]$ 推导而来，根据 $RK_8[12,13,14]\oplus X'_9[12,13,14]=W_8[12,13,14]$ 计算出 $W_8[12,13,14]$，以 $W_8[12,13,14]$ 和 $\Delta X'_9[12,13,14]$ 为索引访问 H_8，每 2^8 行有 1 个值，最终可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[1\sim 5,7,8,10,11]\|RK_9[0,6,9]\|RK_8[12,13,14]$ 共 $2^{48-8}=2^{40}$ 个可能值。

9) 根据明文对状态值，可以计算得到 $W_1[3,7,8]$ 和 $\Delta W_1[3,7,8]$ 的值，以 $\Delta W_1[3,7,8]$ 为索引访问 H_9，得到 $X_2[3,7,8]$，并通过计算可以获得 $RK_{11}[1\sim 5,7,8,10\sim 14]\|RK_{10}[1\sim 5,7,8,10\sim 14]\|RK_9[3,6,9]\|RK_8[15]\|RK_1[3,7,8]$ 共 $2^{40+4}=2^{44}$ 个可能值。

10) 根据明文对状态值，可以计算得到 $W_1[0,6,9]$ 和 $\Delta W_1[0,6,9]$ 的值，根据性质 5，可知 $RK_1[9]$ 可由 $RK_{10}[0,1,2]$ 推导而来，且 $RK_{10}[1,2]$ 在先前步骤已经获得，因此猜测 $RK_{10}[0]$ 的全部 2^4 个可能值，可以获得 2^4 个 $RK_1[9]$ 的可能值，因此通过计算等式 $RK_1[9]\oplus W_1[9]=X_2[9]$，因此以 $\Delta W_1[0,6,9]$ 和 $X_2[9]$ 为索引

访问 H_{10}，平均每行有一个值，因此可以获得 $RK_{11}[1\sim5,7,8,10\sim14]\|RK_{10}[1\sim5,7,8,10\sim14]\|RK_9[3,6,9]\|RK_8[15]\|RK_1[0,3,6,7,8,9]$ 共 $2^{44+4+0}=2^{48}$ 个可能值。

3. 对于上述操作所得到的有序序列 $\Delta Z_8[15]$，需要判断是否与表 H_0 中的条目匹配。如果匹配则上述轮子密钥可能为正确的；否则，就是错误的轮子密钥，直接筛选出对应的密钥空间。一个错误的密钥被筛选的概率为 $2^{80-128}=2^{-48}$，因此大约还有 $1+2^{-48}\times2^{68}\times2^{48}\approx2^{68}$ 个候选轮子密钥剩余。根据密钥编排可知 $RK_{10}[7,8]$ 可由 $RK_{11}[0,1,2,3,4]$ 推导；$RK_9[6]$ 可由 $RK_{10}[1,2]$ 推导出；$RK_9[9]$ 可由 $RK_{11}[0,1,2]$ 推导而来；$RK_8[12,13,14]$ 可由 $RK_{10}[3,4]$ 和 $RK_{11}[1,2]$ 推导而来；$RK_1[9]$ 可由 $RK_{10}[0,1,2]$ 推导而来，由此在在线阶段一共猜测 100 个主密钥相关比特位，剩余 28 个未知比特。最后穷举搜索 2^{68} 个候选轮子密钥和 28 个未知比特来恢复主密钥。

3.5.3. 复杂度分析

根据在线阶段的步骤(1)，可以清楚地发现攻击的数据复杂度为 $2^{13+48}=2^{61}$ 个选择明文。同时攻击的存储复杂度主要由预计算阶段的预计算表 H_0 的大小决定，大约为 $2^{80}\times128/64\approx2^{81}$ 个 64 比特块。预计算阶段的时间复杂度大约为 $2^{80}\times33\times30/(16\times11)\approx2^{82.49}$ 次加密，由上文可知，在线阶段复杂度最高于步骤 10)，因此在线阶段的时间复杂度为 $2^{68}\times33\times2^{48}\times3/(16\times11)\approx2^{115.17}$ 次加密。对于剩下的 2^{68} 个候选子密钥以及 28 个未知比特，可以使用两个明密文对进行穷尽搜索恢复主密钥的值，该过程对应的时间复杂度为 $2^{68}\times2\times2^{28}=2^{97}$ 加密。

综上所述，该攻击的时间复杂度为 $2^{115.17}$ 次加密，数据复杂度为 2^{61} 个选择明文，存储复杂度为 2^{81} 个 64 比特块。

4. 结束语

本文研究了 INLEC 轻量级分组密码算法抵抗中间相遇攻击的能力。首先研究了 INLEC 密码算法的结构，结合算法 2 构建出基于 MILP 的自动化搜索模型，并使用 Gurobi 求解器，寻找出了多条 5 轮的 INLEC 中间相遇区分器，并且对搜索到的区分器往前和往后添加若干轮，记录其攻击的最大轮数和构建出有序序列所需的半字节个数，最终筛选出效果最好的区分器作为本文攻击的中间相遇区分器。根据上述策略，选取出效果最优的区分器，并在其基础上往前扩展 2 轮，往后扩展 4 轮，构成了针对 INLEC 密码算法的 11 轮中间相遇攻击，在攻击的过程当中通过研究其密钥编排算法，使用预计算表技巧提升了攻击的效率，降低了攻击所需的时间复杂度。该攻击所需的时间复杂度为 $2^{115.17}$ 次加密，数据复杂度为 2^{61} 个选择明文，存储复杂度为 2^{81} 个 64 比特块。此结果是对 INLEC 安全性分析的重要补充。

参考文献

[1] Beierle, C., Leander, G., Moradi, A. and Rasoolzadeh, S. (2019) CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection against DFA Attacks. *IACR Transactions on Symmetric Cryptology*, **1**, 5-45. https://doi.org/10.46586/tosc.v2019.i1.5-45

[2] Wu, W. and Zhang, L. (2011) Lblock: A Lightweight Block Cipher. In: *Lecture Notes in Computer Science*, Springer, 327-344. https://doi.org/10.1007/978-3-642-21554-4_19

[3] Bansod, G., Pisharoty, N. and Patil, A. (2016) PICO: An Ultra Lightweight and Low Power Encryption Design for Ubiquitous Computing. *Defence Science Journal*, **66**, Article 259. https://doi.org/10.14429/dsj.66.9276

[4] 贾珂婷, 董晓阳, 魏淙洺, 等. 分组密码算法 FESH [J]. 密码学报, 2019, 6(6): 713-726.

[5] 冯秀涛, 曾祥勇, 张凡, 等. 轻量级分组密码算法 FBC [J]. 密码学报, 2019, 6(6): 768-785.

[6] Mouha, N., Wang, Q., Gu, D. and Preneel, B. (2012) Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: *Lecture Notes in Computer Science*, Springer, 57-76. https://doi.org/10.1007/978-3-642-34704-7_5

[7] Feng, J., Li, L., Yan, L. and Deng, C. (2024) INLEC: An Involutive and Low Energy Lightweight Block Cipher for Internet of Things. *Pervasive and Mobile Computing*, **105**, Article 101991. https://doi.org/10.1016/j.pmcj.2024.101991

[8] Diffie, W. and Hellman, M.E. (1977) Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, **10**, 74-84. https://doi.org/10.1109/c-m.1977.217750

[9] Joan, D. and Vincent, R. (2002) The Design of Rijndael: AES-the Advanced Encryption Standard. In: *Information Security and Cryptography*, Springer, 126.

[10] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., *et al*. (2015) Midori: A Block Cipher for Low Energy. In: *Lecture Notes in Computer Science*, Springer, 411-436. https://doi.org/10.1007/978-3-662-48800-3_17