文章编号:1007－6735(2026)03－0295－13　DOI: 10.13255/j.cnki.jusst.20250221001

基于 MILP 的 10 轮 IVLBC 的中间相遇攻击

刘　亚[1]，　曾衡顺[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 200093；2. 上海出版印刷高等专科学校 信息与智能工程系，上海 200093；3. 安徽财经大学 统计与应用数学学院，蚌埠 233030；4. 广东科技学院 计算机学院，东莞 523083）

摘要： 对合型轻量级分组密码 (involutive lightweight block cipher, IVLBC) 是一种专为资源受限的物联网环境设计的轻量级分组密码算法。提出了基于混合整数线性规划（mixed-integer linear programming, MILP）改进的 10 轮 IVLBC-128 的中间相遇攻击。首先研究了 IVLBC 结构特性，建立了混合整数线性规划的自动化搜索模型，找到 10 条 5 轮 IVLBC-128 中间相遇区分器；其次，研究了该算法的密钥扩展方案的冗余性，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，并在此区分器前面接 1 轮，后面接 4 轮，构造出 10 轮 IVLBC-128 中间相遇攻击路径，最后基于该路径恢复出密钥。整个分析过程的时间复杂度为 $2^{123.26}$ 次加密，数据复杂度为 2^{61} 个选择明文，存储复杂度为 2^{77} 个 64 比特块。该结果将 IVLBC-128 此前公开的最优中间相遇攻击结果提升了两轮。

关键词： 轻量级分组密码；中间相遇攻击；混合整数线性规划；IVLBC

中图分类号： TP 309　　**文献标志码：** A

MILP-based meet-in-the-middle attack on 10-round IVLBC

LIU Ya[1]，　ZENG Hengshun[1]，　ZHAO Fengyu[2]，　LIU Xianbei[3]，　QU Bo[4]

(1. *School of Optical−Electrical And Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China*; 2. *Department of Information and Intelligence Engineering, Shang hai Publishing and Printing College, Shanghai 200093, China*; 3. *School of Statistics and Applied Mathematics, Anhui University of Finance and Economics, Bengbu, 233030, China*; 4. *School of Computer Science, Guang dong University of Science and Technology, Dongguan 523083, China*)

Abstract: Involutive lightweight block cipher (IVLBC) is a lightweight block cipher algorithm specifically designed for resource-constrained internet of things environments. An meet-in-the-middle attack of 10-round IVLBC-128 based on mixed-integer linear programming (MILP) improvement was proposed. Firstly, the structural characteristics of IVLBC were studied, and an automated search model of mixed integer linear programming was established to find 10 5-round IVLBC-128 meet-in-the-middle

收稿日期: 2025−02−21

基金项目: 国家自然科学基金资助项目（62002184）；安徽省高校自然科学重点项目（2024AH050011）

通信作者: 刘　亚（1983—），女，副教授．研究方向为信息安全、密码学等．E-mail：liuya@usst.edu.cn

引文格式: 刘亚，曾衡顺，赵逢禹，等．基于 MILP 的 10 轮 IVLBC 的中间相遇攻击[J]．上海理工大学学报，2026，48(3)：295-307.

Citation: LIU Ya, ZENG Hengshun, ZHAO Fengyu, et al. MILP-based meet-in-the-middle attack on 10-round IVLBC[J]. Journal of University of Shanghai for Science and Technology, 2026, 48(3)：295-307.

distinguishers. Secondly, the redundancy of the key expansion scheme of this algorithm was studied. According to the principle of minimizing the guessed key bit number in the analysis process, 1 meet-in-the-middle distinguisher was selected, and this distinguisher was followed by 1 round, then 4 rounds, to construct a 10-round IVLBC-128 meet-in-the-middle attack path. Finally, the key was recovered based on this path. The overall analysis requires a time complexity of $2^{123.26}$ encryptions, a data complexity of 2^{61} chosen plaintexts, and a memory complexity of 2^{77} 64-bit blocks. This result has improved the previous public optimal meet-in-the-middle attack result of IVLBC-128 by two rounds.

Keywords: *lightweight block cipher*; *meet-in-the-middle attack*; *mixed integer linear programming*; *IVLBC*

随着物联网技术的迅猛发展，智能卡等资源受限设备的应用越来越普及。尽管传统的加密算法在安全性方面表现卓越，但往往需要高的计算资源和较大的存储空间，无法满足资源受限环境的需求，因此在实际应用系统中面临部署困难的情况。针对上述问题，轻量级分组密码算法应运而生。轻量级分组密码不仅能够保障信息的保密性，而且在具体的产品或者系统中使用时需要的软硬件资源少，适应于物联网等资源受限环境的需求。为了保障这些实际系统中数据的安全，就必须研究运用其中的轻量级分组密码抵抗已有攻击的安全强度。

不可能差分分析[1]、中间相遇攻击[2]、积分分析[3]、代数故障攻击[4]、侧信道分析[5]等方法是当前非常流行的密码分析方法，已经在多种分组密码上取得了较好的分析结果。其中，中间相遇攻击需要构造出一条特定输入到特定输出的中间相遇区分器来进行密钥恢复。在构造区分器和攻击路径时，如果采用手动构造，那么存在效率低、计算量大、耗时长等诸多缺点。近年来，研究者提出了各类基于计算机辅助的自动化搜索算法，如基于混合整数线性规划(mixed-integer linear programming, MILP)[6]的自动化搜索算法、基于布尔可满足问题[7]的自动化搜索算法、基于约束规划(constraint programming, CP)的自动化搜索算法等，使用自动化搜索算法大大加快了密码分析的效率，扩大了搜索范围，改进了密码分析的效果。

基于MILP的自动化搜索算法由于模型构造简单、分析效率高等原因，近年来被广泛应用到分组密码分析中。MILP是一种优化方法，主要用于解决涉及线性目标函数和约束条件的优化问题。Mouha等[6]首次将MILP技术与密码分析方法相结合，并用来评估密码算法的安全边界。随后，Sun等[8]进一步优化MILP模型的建模方法，提出了两种更准确刻画S盒差分性质的方法：根据S盒的差分性质生成不等式；从S盒的所有可能的差分模式表达式中提取出不等式，并且通过贪心算法进行二次筛选，选取出能更精确刻画S盒性质的不等式。这种改进后的MILP模型可以求解出更为精确的最小活跃S盒数量，因此，获得了许多分组密码算法更为精准的安全边界。然而Sun等提出的该建模方法不适用于输入大于4比特的大S盒。Abdelkhalek等[9]提出了针对大S盒的建模方法，即将逻辑条件建模中生成的约束问题转换为最小化布尔函数和乘积问题。Zhang等[10]将Matsui的分支定界搜索算法与MILP自动化搜索技术相结合，通过在MILP模型中加入了由Matsui算法的边界条件得出的约束，使得模型的可行域减小，从而缩短求解时间。Zhou等[11]将分而治之的思想与MILP技术结合，通过将可行域分成若干子集，动态调整子集的搜索顺序等方式提高了MILP模型的求解速度。Sadeghi等[12]提出一种基于MILP技术识别分组密码中不兼容的差分路径的方法，利用MILP技术加速对弱密钥准确值的搜索，最后应用于SPECK和SIMECK两个密码算法中，构造出更长的SPECK相关密钥差分路径。Cao等[13]提出了一种新的基于MILP模型的不可能差分区分器搜索方法，与已有的MILP搜索不可能区分器的方式不同，该方法基于未被扰动的差分比特进行建模，克服了现有工具将该模型搜索到的不可能差分区分器个数少以及搜索到的截断不可能差分轮数短的缺点，最终应用于ASCON、SIMON、LBlock和LEA密码算法，发现了一些新的不可能差分区分器。Liu等[14]提出了一种新的基于MILP的不可能差分区分器技术，该技术引入了两种不同的类型来描述差分状态，与以往技术不

同，通过几个约束来表示差分矛盾点，随后从相应模型的解中得出不可能差分链，最终应用于 Midori-64 和 CRAFT 等密码算法中。目前，基于 MILP 的自动化搜索算法可以有效地提高密码分析的效率，得到了许多非常好的分析结果。

对合型轻量级分组密码 (involutive lightweight block cipher, IVLBC) 是 Huang 等[15] 提出的一种轻量级分组密码算法，该算法是一种基于替换和排列的置换−代换网络（substitution-permutation network, SPN）结构的分组密码算法，该密码具有低延迟和低功耗等特性，旨在为物联网环境提供适合的加密解决方案。Murat 等[16] 利用 MILP 模型对 IVLBC 算法进行了评估，并且构造了 7 轮的 IVLBC 差分区分器和 7 轮的线性区分器。Yuki 等[17] 首次评估了 IVLBC 抗中间相遇攻击的能力，证明了 8 轮的 IVLBC-128 和 IVLBC-80 都是容易受到中间相遇攻击，攻击需要的时间复杂度分别为 $2^{75.547}$ 和 $2^{116.541}$ 次加密。Kaur 等[18] 利用融入差分分布表（differential distribution table, DDT）概率的 MILP 模型对 IVLBC 进行差分密码分析，找到了 7 轮的 IVLBC 差分区分器，并实现了 8 轮 IVLBC 的密钥恢复攻击，攻击的数据复杂度为 2^{49} 个选择明文，内存复杂度为 $2^{50.59}$ 字节，时间复杂度为 2^{49} 次加密。Yu 等[19] 改进复杂线性层的 MILP 建模方法，首次提出 9 轮 IVLBC 积分区分器，又利用二元扩散层的结构特性将其扩展为 10 轮积分区分器，并基于该区分器实现 14 轮密钥恢复攻击，攻击的数据复杂度为 2^{63} 个选择明文，时间复杂度为 $2^{71.197}$ 次 14 轮加密，这是目前针对 IVLBC 最长的积分密码分析结果，但是该攻击的数据复杂度非常大，需要半个电码本。

本文利用 MILP 自动化搜索算法，提出了 10 轮 IVLBC-128 改进的中间相遇攻击。具体来说，首先结合 IVLBC 算法的各个模块的结构特性，建立半字节级的 MILP 自动化搜索模型，该模型包含 1 200 个约束不等式。接着结合自动化搜索算法找到了 10 条 5 轮中间相遇区分器；随后，研究该算法的密钥扩展方案的冗余性，找到轮密钥之间的线性关系，基于此选取需要猜测密钥位最少的一条中间相遇区分器，并在此区分器的前面增加 1 轮，后面增加 4 轮，构成 10 轮 IVLBC-128 的中间相遇攻击路径；最后根据该路径恢复出主密钥。整个攻击需要时间复杂度为 $2^{123.26}$ 次加密，数据复杂度为 2^{61} 个选择明文，存储复杂度为 2^{77} 个 64 比特块。表 1 列出了近年来对于 IVLCB-80/128 的分析结果。从表 1 可知，本文的结果将之前最好的中间相遇攻击的轮数提高了两轮，是对轻量级分组密码 IVLBC 安全性评估的重要补充。

表 1　IVLBC-80/128 分析总结

Tab.1　Summary of the cryptanalysis on IVLBC-80/128

密码长度	轮数	时间/次加密	数据/比特	存储/比特	攻击	参考文献
80/128	7	—	—	—	差分分析	[16]
80/128	7	—	—	—	线性分析	[16]
80	8	$2^{75.47}$	4	—	中间相遇攻击	[17]
128	8	$2^{116.541}$	6	—	中间相遇攻击	[17]
80/128	8	2^{49}	2^{49}	$2^{50.59}$	差分分析	[18]
128	10	$2^{123.26}$	2^{61}	2^{77}	中间相遇攻击	本文
80/128	14	2^{75}	2^{63}	2^{36}	积分攻击	[19]

1　预备知识

1.1　符号介绍

a. RC_i，$i\in\{1,2,3,\cdots,29\}$表示第i轮的轮常数；

b. $K\{i\}$，$i\in\{0,1,2,3\}$表示将主密钥的 4 部分，每个 32 比特；

c. $K_0\|K_1$表示K_0、K_1的级联；

d. $<<7$表示循环向左移动 7 比特；

e. RK_i，$i\in\{1,\cdots,28\}$表示第i轮的轮密钥；

f. $K(i)$，$i\in\{0,\cdots,127\}$表示主密钥K的第i个比特；

g. $RK_i[j]$，$i\in\{1,\cdots,28\}$，$j\in\{0,\cdots,15\}$表示第i轮的第j个半字节；

h. $RK_i[j](k)$，$i\in\{1,\cdots,28\}$，$j\in\{0,\cdots,15\}$，$k\in\{0,\cdots,4\}$表示第i轮的第j个半字节的第k个比特；

i. $RC_i(j)$，$i\in\{1,\cdots,28\}$，$j\in\{0,1,2,3,4\}$表示第i轮的轮常数的第j个比特；

j. $FX_i, FY_i, FZ_i, FW_i, i \in \{1,\cdots,29\}$分别表示第$i$轮的轮密钥加(AddRoundKey, ARK)、S盒替换(SubCell, SC)、P置换(Permute-Nibble, PN)以及列混合(MixColumn, MC)之前的中间状态;

k. $FX_i[j]$, $FY_i[j]$, $FZ_i[j]$, $FW_i[j]$, $i \in \{1,\cdots,29\}$, $j \in \{0,\cdots,15\}$分别表示第i轮的ARK、SC、PN以及MC之前的中间状态值的第j半字节;

l. $FX'_i[j]$, $FY'_j[j]$, $FZ'_i[j]$, $FW'_i[j]$, $i \in \{1,\cdots,29\}$, $j \in \{0,\cdots,15\}$分别表示交换MC和ARK之后第i轮的MC、SC、PN以及ARK之前的中间状态值的第j半字节;

m. $ERK_i[j]$表示MC和ARK操作交换后的第i轮的第j个等效轮子密钥单元。

1.2 轻量级分组密码算法 IVLBC 的描述

IVLBC是一种基于SPN结构的轻量级分组密码,其明密文长度均为64比特。密钥有两种选择,即80比特和128比特,分别记为IVLBC-80/128,轮数为29。本文主要研究IVLBC-128抵抗中间相遇攻击的能力。IVLBC-128算法的整体结构如图1所示。

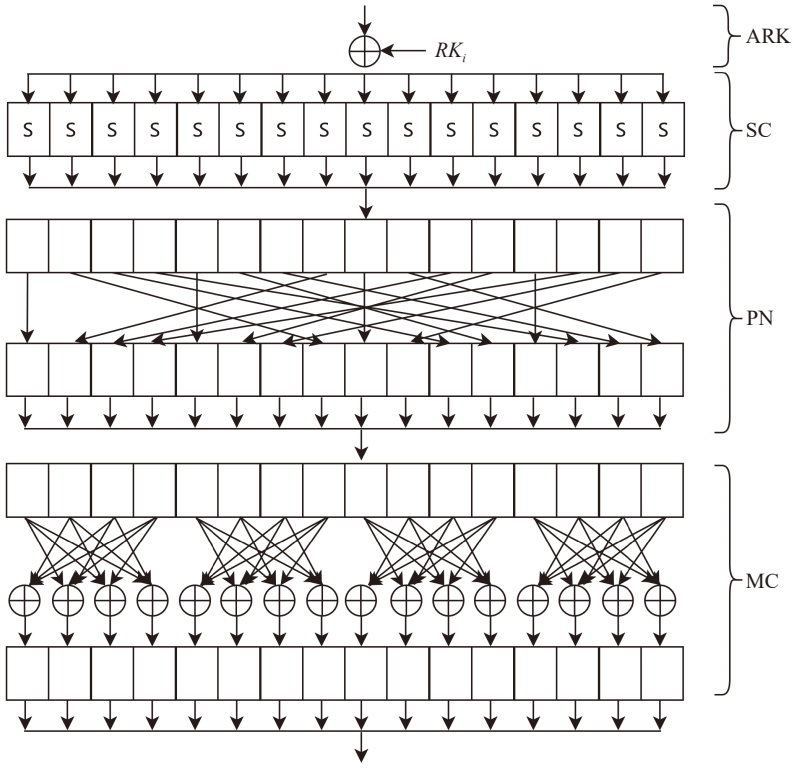

图 1 IVLBC 加密流程图

Fig.1 IVLBC encryption flow chart

IVLBC-128的分组长度为64比特,其内部状态可以表示成4×4的矩阵:

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

1.2.1 加密算法

IVLBC-128共29轮,每轮包含4个运算:ARK、SC、PN以及MC,具体定义如下:

a. ARK:通过密钥编排算法生成每轮的轮密钥,并将轮密钥与中间状态矩阵按相应比特进行异或操作。

b. SC:SC是一种非线性变换,将S盒按照顺序运用于密码中间状态的每个半字节单元,即$x[i] \to S(x[i])$, $(0 \leqslant i \leqslant 15)$,S盒的输入和输出均为4比特,具体如表2所示。

c. 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,如表3所示。

d. MC:将中间状态矩阵左乘二进制矩阵M:

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \tag{2}$$

此外,在第28轮加密完成之后,最后一轮仅

表 2　IVLBC 的 S 盒

Tab.2　S-box of IVLBC

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

表 3　IVLBC 的 PN

Tab.3　Permute-nibble of IVLBC

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

需要进行一个 ARK 即可。

1.2.2　密钥编排方案

IVLBC-128 使用的密钥编排方案如算法 1 所示。该密钥编排方案使运用 128 比特的主密钥 K 生产轮子密钥，并且每次取 128 比特的高位 64 比特作为轮密钥。

算法1　IVLBC-128的密钥编排算法

输入: 主密钥 K

输出: 轮密钥 RK_i

for $i = 1$ to R do

$K\{0\}\|K\{1\}\|K\{2\}\|K\{3\} \leftarrow K$;

$K\{0\}\|K\{1\} \leftarrow (K\{0\}\|K\{1\}) <<< 7$;
$[K(120)\|K(121)\|K(122)\|K(123)] \leftarrow S[K(120)\|K(121)\|K(122)\|K(123)]$;
$[K(124)\|K(125)\|K(126)\|K(127)] \leftarrow S[K(124)\|K(125)\|K(126)\|K(127)]$;
$[K(64)\|K(65)\|K(66)\|K(67)\|K(68)] \leftarrow [K(64)\|K(65)\|K(66)\|K(67)\|K(68)] \oplus RC_i$;

$RK_i \leftarrow K\{0\}\|K\{1\}$;

$K \leftarrow K\{2\}\|K\{3\}\|K\{0\}\|K\{1\}$;

end for

return RK_i

性质 1　（S 盒性质）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。

2　10 轮的 IVLBC-128 中间相遇攻击

2.1　IVLBC-128 的密钥编排方案冗余性研究

本文研究了 IVLBC-128 的密钥编排方案的冗余性，利用计算机辅助得到轮密钥的线性关系如下：

性质 2　若轮子密钥 $RK_i[j]$ 的值已知，则 $RK_{i+2}[j]$ 可以由 $RK_i[j]$ 计算出来，具体的对应关系如表 4 所示。譬如：$RK_{i+2}[0](0 \sim 3) = RK_i[1](3)\|RK_i[2](0 \sim 2)$。同时，由于 $ERK_i[j] = MC^{-1}(RK_i[j])$，即可以由 $RK_i[j]$ 推导出 $ERK_i[j]$ 的值，因此该性质也可被应用于等效轮子密钥 $ERK_i[j]$ 中。其中：$S[RK_i[14]](0 \sim 2)$ 表示第 i 轮的第 14 个半字节经过 S 盒之后的第 0～2 比特。表 4 中展现了 IVLBC 的轮密钥的线性关系，在接下来的中间相遇区分器的搜索和中间相遇攻击的构造中，将基于表 4 来提高分析的效率。

表 4　轮密钥 $RK_i[j]$ 与 $RK_{i+2}[j]$ 的关系表

Tab.4　Relationship between $RK_i[j]$ and $RK_{i+2}[j]$

$RK_{i+2}[j](k)$	$RK_i[j](k)$
$RK_{i+2}[0](0 \sim 3)$	$RK_i[1](3)\|RK_i[2](0 \sim 2)$
$RK_{i+2}[1](0 \sim 3)$	$RK_i[2](3)\|RK_i[3](0 \sim 2)$
$RK_{i+2}[2](0 \sim 3)$	$RK_i[2](3)\|RK_i[3](0 \sim 2)$
$RK_{i+2}[3](0 \sim 3)$	$RK_i[3](3)\|RK_i[4](0 \sim 2)$
$RK_{i+2}[4](0 \sim 3)$	$RK_i[4](3)\|RK_i[5](0 \sim 2)$
$RK_{i+2}[5](0 \sim 3)$	$RK_i[5](3)\|RK_i[6](0 \sim 2)$
$RK_{i+2}[6](0 \sim 3)$	$RK_i[6](3)\|RK_i[7](0 \sim 2)$
$RK_{i+2}[7](0 \sim 3)$	$RK_i[7](3)\|RK_i[8](0 \sim 2)$
$RK_{i+2}[8](0 \sim 3)$	$RK_i[8](3)\|RK_i[9](0 \sim 2)$
$RK_{i+2}[9](0 \sim 3)$	$RK_i[9](3)\|RK_i[10](0 \sim 2)$
$RK_{i+2}[10](0 \sim 3)$	$RK_i[10](3)\|RK_i[11](0 \sim 2)$
$RK_{i+2}[11](0 \sim 3)$	$RK_i[11](3)\|RK_i[12](0 \sim 2)$
$RK_{i+2}[12](0 \sim 3)$	$RK_i[12](3)\|RK_i[13](0 \sim 2)$
$RK_{i+2}[13](0 \sim 3)$	$RK_i[13](3)\|S[RK_i[14]](0 \sim 2)$
$RK_{i+2}[14](0 \sim 3)$	$S[RK_i[14]](3)\|S[RK_i[15]](0 \sim 2)$
$RK_{i+2}[15](0 \sim 3)$	$S[RK_i[15]](3)\|(RK_i[0](0 \sim 2) \oplus RC_i(0 \sim 2))$

性质 3　根据密钥扩展算法可得，$RK_{11}[0 \sim 3]$ 可由 $ERK_{11}[0 \sim 3]$ 推导而来，且 $RK_{11}[0 \sim 3]$ 可由主密钥 $K(42 \sim 57)$ 确定。同时 $RK_1[9]$ 可由 $K(43 \sim 46)$ 确定，因此，$RK_1[9]$ 可由 $RK_{11}[0 \sim 3]$ 推导而来，即 $RK_1[9]$ 可由 $ERK_{11}[0 \sim 3]$ 推导而来。

性质 4　根据密钥扩展算法可得，$RK_{11}[4 \sim 7]$ 可由 $ERK_{11}[4 \sim 7]$ 推导而来，且 $RK_{11}[4 \sim 7]$ 可由主密钥 $K(58 \sim 63, 0 \sim 9)$ 确定。$RK_1[15]$ 可由 $K(3 \sim 6)$ 确定，因此 $RK_1[15]$ 可由 $RK_{11}[4 \sim 7]$ 推导而来；同时 $RK_1[12]$ 可由 $K(55 \sim 58)$ 确定，且由性质 3 可知，

$RK_{11}[0\sim3]$可由主密钥$K(42\sim57)$确定，因此$RK_1[12]$可由$RK_{11}[4\sim7]$和$RK_{11}[0\sim3]$推导而来。

性质5 根据密钥扩展算法可得，$RK_{11}[8\sim11]$可由$ERK_{11}[8\sim11]$推导而来，且$RK_{11}[8\sim11]$可推导出主密钥$K(10\sim24)$。$RK_1[3]$可由$K(19\sim22)$确定，因此$RK_1[3]$可由$RK_{11}[8\sim11]$推导而来；同时$RK_1[2]$可由$K(15\sim18)$确定，因此$RK_1[2]$可由$RK_{11}[8\sim11]$推导而来。

性质6 根据密钥扩展算法可得，$RK_{11}[12\sim15]$可由$ERK_{11}[12\sim15]$推导而来，且$RK_{11}[12\sim15]$可推导出主密钥$K(27,28,31,32,35\sim41)$。$RK_1[8]$可由$K(39\sim42)$确定，且由性质3可知，$RK_{11}[0\sim3]$可由主密钥$K(42\sim57)$确定，因此$RK_1[8]$可由$RK_{11}[12\sim15]$和$RK_{11}[0\sim3]$推导而来。

2.2 基于MILP的IVLBC-128模型

IVLBC的加密过程由ARK、SC、PN以及MC组成，其中ARK不会改变差分，且由于本文中的区分器是基于半字节建立的，SC并不会改变差分值状态，因此在本小节中对ARK以及SC不进行建模。

a. PN。假设$x[i],y[i],i\in\{0,\cdots,15\}$分别表示IVLBC中PN过程的输入和输出，则每一轮的PN的约束不等式共有16个，具体如式(3)所示：

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

b. MC运算。在IVLBC加密算法中，MC运算是通过左乘一个数字矩阵完成MC变换，变换过程中涉及的加法和乘法变换均在有限域$GF(2)$上执行。假设$(x[0],\cdots,x[15])$以及$(y[0],\cdots,y[15])$分别是MC过程的输入以及输出，则MC的变换可以表示为

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

以$y[0]$为例，$y[0]=x[1]+x[2]+x[3]$，因此可以用如下的不等式描述MC前后的状态值的逻辑变化：

$$\begin{cases} 3y[0]-x[1]-x[2]-x[3]\geqslant 0 \\ x[1]+x[2]+x[3]-y[0]\geqslant 0 \end{cases} \tag{5}$$

综上，每轮的MC的计算过程用如下约束不等式共32个，具体如式(6)所表示。

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

c. 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$IC[i]$和的$OC[i]$交集，根据上述的描述，每轮的IVLBC的差分状态可以分成3个部分，分别是每轮的输入差分状态，经过PN后的差分状态以及经过MC之后的差分状态，每个部分的不等式约束共48个，具体如式(7)所示。

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

综上，考虑加密方向和解密方向的差分状态，每轮的 PN 一共 16+16=32 个约束条件；每轮的 MC 一共 32+32=64 个约束条件；每轮关于差分状态的约束一共 48×3=144 个约束条件。因此搜索 5 轮的中间相遇一共需要 (64+144+32) ×5=1 200 个约束条件。

2.3　基于 MILP 的区分器搜索算法

采用 Demirci[20] 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 3 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，则这对密钥猜测可能正确。具体的攻击过程如图 2 所示。

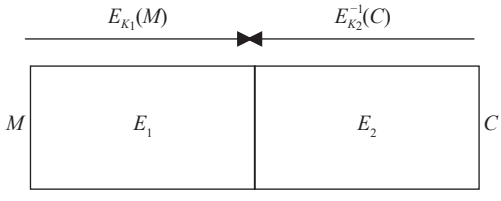

图 2　DS-MITM 攻击过程

Fig.2　DS-MITM attack process

在 DS-MITM 中需要构建出相应密码算法的中间相遇区分器，并计算出区分器相应的有序序列值，并且有序序列的取值越少，相应攻击的复杂度也就越低。算法 2 展示了基于 MILP 的中间相遇区分器的搜索算法，该算法遍历所有可能的输入差分集合ΔI和输出差分集合ΔO，将输入差分向加密方向扩散，且将输出差分向解密方向扩散，最终若扩散若干轮后，二者的中间状态相同则视为搜索到一条中间相遇区分器。因此，本文首先采用 2.2 小节中所列出的不等式约束条件构建出 IVLBC-128 的中间相遇搜索模型，随后根据算法 2，求解得出 10 条 IVLBC-128 中间相遇区分器。对搜索到的中间相遇区分器，在其基础上，往前和往后各添加若干轮构成中间相遇攻击，并记录下能够攻击的最长轮数以及相应的复杂度情况，最后对这些区分器进行筛选，根据有序序列猜测值最少，和在线阶段所需密钥位最少的原则进行筛选，选取出效果最好的区分器作为本文的攻击区分器。表 5 中列出了搜索到的区分器中攻击效果最好的 3 条中间相遇区分器，以及这些中间相遇区分器所最多能够攻击的轮数和有序序列需要猜测半字节个数。根据表 5 的内容可知Δ_3的攻击

算法2 IVLBC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,IVLBC算法的不等式约束,总轮数R

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

将目标函数设置为IC和OC的交集

for $i = 1$ to R do

输入IVLBC算法的不等式约束

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 $i \in \Delta O$ 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$Distlist$

表 5 搜索到的部分 IVLBC 中间相遇区分器

Tab.5 Meet-in-the-middle distinguishers of IVLBC

编号	中间相遇区分器	区分器轮数	有序序列需要猜测半字节个数	攻击轮数
Δ_1	$[0,1,0,0,1,0_{(11)}] \rightarrow [0_{(5)},1,0,1,0_{(8)}]$	5	23	9
Δ_2	$[0_{(4)},1,0_{(11)}] \rightarrow [0_{(6)},1,1,0_{(8)}]$	5	21	9
Δ_3	$[0_{(11)},1,0_{(2)},1,0] \rightarrow [0_{(7)},1,0_{(8)}]$	5	19	10

轮数最长，构成有序序列需要猜测的半字节数最少，因此在文章后续内容中作为中间相遇攻击的区分器。

在表 5 中 0 表示该半字节的差分为 0，1 则表示差分为非零， $0_{(i)}$表示连续 i 个比特为 0。本文共搜索到 10 条 5 轮 IVLBC-128 中间相遇区分器。表 5 列出了其中攻击效果最优的 3 条，其余区分器在攻击轮数、猜测半字节数或有序序列复杂度方面均不如所列 3 条，故未全部列出。

2.4 5 轮的 IVLBC-128 中间相遇区分器

按照上节所述，选取表 5 的Δ_3作为本文的中间相遇区分器。该区分器的具体情况如图 3 所示。

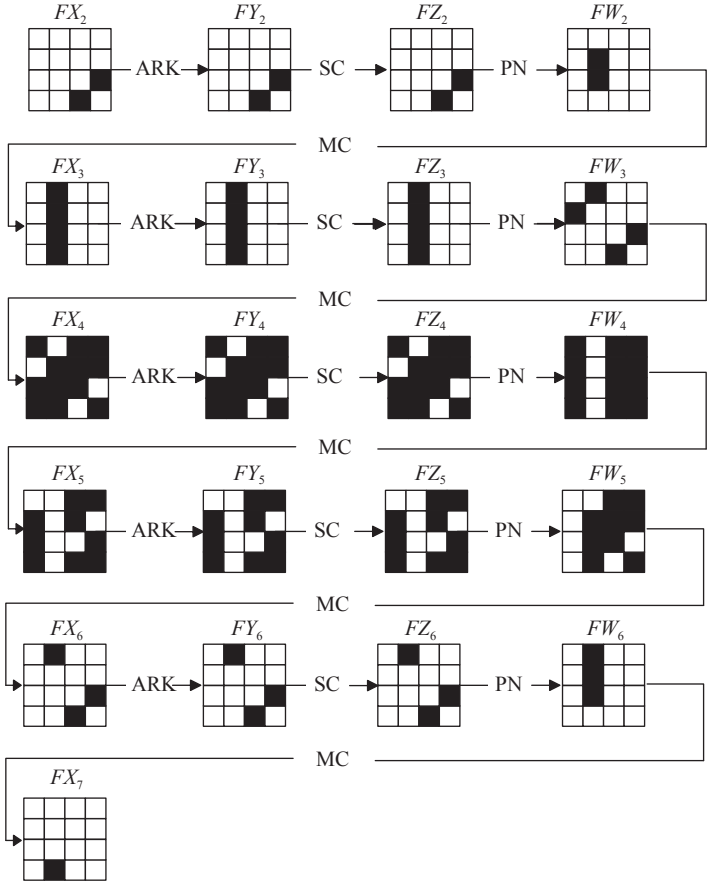

图 3 5 轮 IVLBC-128 的中间相遇区分器

Fig.3 A middle-in-the-middle distinguisher of IVBLC-128

对于选取中间相遇区分器，首先在$FX_2[11,14]$处构造了一个δ集，即选取$FX_2[11,14]$为所有可能值从而形成一个δ集，记为$(FX_2(0),FX_2(1),\cdots,FX_2(2^8-1))$，选取32个$FX_2(i),i\in\{0,\cdots,31\}$通过5轮的IVLBC-128加密，得到有序序列。如果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^{76}个值。将这些有序序列存放在一个预计算表中。该5轮的IVLBC-128中间相遇区分器的特性如性质7所述。

性质 7　将$FX_2[11,14]$处的$[FX_2(0),FX_2(1),\cdots,FX_2(2^8-1)]$集$[(FX_2(0),FX_2(1),\cdots,FX_2(2^8-1)]$进行5轮的IVLBC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，则与该集相关的有序序列$\Delta FX_7[7]$的可能性为2^{76}个，由$FY_2[11,14]$、$FY_3[4\sim7]$、$\Delta FX_7[7]$、$FZ_6[4,11,14]$、FZ_5 [1,2,3,8,9,11,12,14,15] 共计19个半字节确定。

证明　首先根据$\Delta FX_2[11,14]$已知可以推导出$\Delta FY_2[11,14]$，并猜测$FY_2[11,14]$的值，可以计算出$\Delta FZ_2[11,14]$以及$FZ_2[11,14]$的值。由于PN和列混淆是线性操作不会改变中间状态的差分值，所以可以推导出$\Delta FY_3[4,5,6,7]$的值，类似地猜测$FY_3[4,5,6,7]$的值，可以推导出$\Delta FZ_3[4,5,6,7]$以及$FZ_3[4,5,6,7]$的值，随后可以计算出$\Delta FY_4[0,2,3,5,6,7,8,9,10,12,13,15]$的值。然后从输出端进行反向推导，猜测$\Delta FX_7[7]$、$FZ_6[4,11,14]$的值，可以推导出$FY_6[4,11,14]$和$\Delta FY_6[4,11,14]$的值，再继续往前计算可以求出$\Delta FZ_5[1,2,3,8,9,11,12,14,15]$的值，继续猜测$FZ_5[1,2,3,8,9,11,12,14,15]$的值，所以可以推导出$\Delta FZ_4[0,2,3,5,6,7,8,9,10,12,13,15]$的值。根据性质1，平均可以得到一个值$X$满足$\Delta FZ_4[0,2,3,5,6,7,8,9,10,12,13,15] = S(X)\oplus S(\Delta FY_4[0,2,3,5,6,7,8,9,10,12,13,15]\oplus X)$，证毕。

2.5　10 轮的 IVLBC-128 攻击过程

在2.4节所示的区分器的基础上，往前接1轮，往后接4轮，并且通过调整密钥异或操作ARK与列混淆MC操作的位置，使用等效密钥ERK减少分析轮中需要猜测的相关轮子密钥的数量，最终构成10轮的中间相遇攻击。具体的攻击过程如图4所示。

该攻击过程的具体流程包括两个阶段：预计

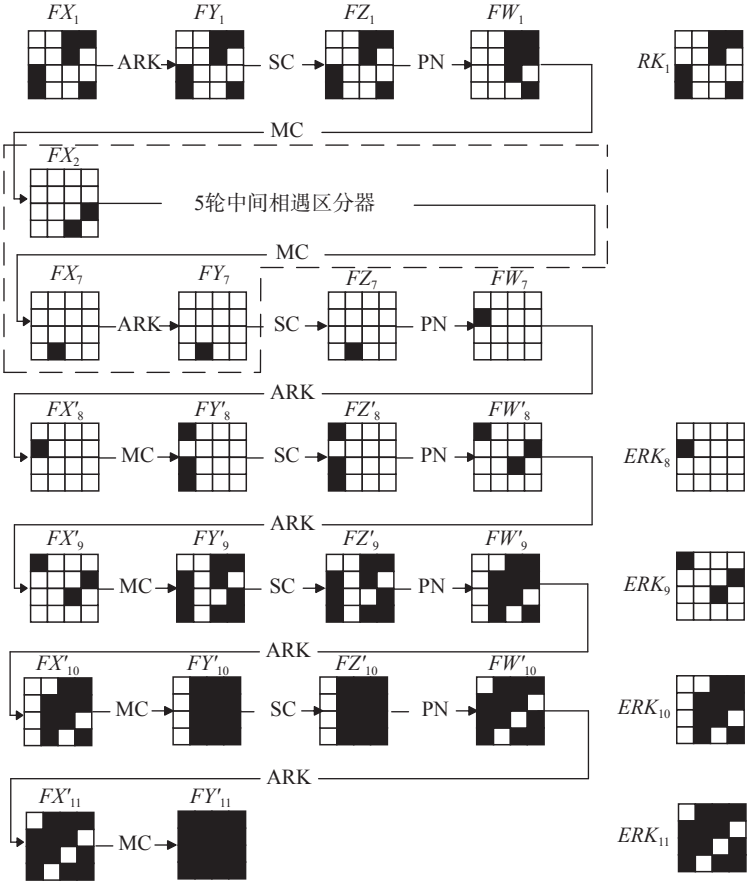

图 4　10 轮 IVBLC-128 的中间相遇攻击过程

Fig.4　A meet-in-the-middle attack on 10-round IVBLC-128

算阶段和在线阶段。

2.5.1 预计算阶段

a. 构建一个预计算表H_0来存储有序序列$\Delta FX_7[7]$的值。此外还建立了分析轮相关的预计算表，以便于在线阶段提取相关的轮子密钥。

b. 通过猜测第10轮中$\Delta FX'_{10}$不为0的状态差分以及FY'_{10}的值，同时结合密文对差分以及密文值计算出满足要求的轮子密钥ERK_{11}的数量。将计算轮子密钥时涉及到的差分以及状态值按需根据索引和内容的形式继续存储，以减少密钥恢复阶段的操作和实时计算。

表$H_1\{ERK_{11}[1,4,11,14]\}$：猜测$\Delta FX'_{10}[5,6,7]$和$FY'_{10}[4,5,6,7]$的全部$2^{28}$个可能值，由此计算出$FX'_{10}[5,6,7]$、$\Delta FX'_{11}[1,4,11,14]$、$FZ'_{10}[4,5,6,7]$、$\Delta FW'_{10}[1,4,11,14]$、$FW'_{10}[1,4,11,14]$，以$\Delta FX'_{11}[1,4,11,14]$为索引，将$\Delta FX'_{10}[5,6,7]$、$FW'_{10}[1,4,11,14]$、$FX'_{10}[5,6,7]$存储在$H_1$中。表$H_1$中有$2^{16}$行，平均每行有$2^{12}$个值。

表$H_2\{ERK_{11}[2,5,8,15]\}$：猜测$\Delta FX'_{10}[8,9,10]$和$FY'_{10}[7,8,9,10]$的全部$2^{28}$个可能值，由此计算出$FX'_{10}[8,9,10]$、$\Delta FX'_{11}[2,5,8,15]$、$FZ'_{10}[7,8,9,10]$、$\Delta FW'_{10}[2,5,8,15]$、$FW'_{10}[2,5,8,15]$，以$\Delta FX'_{11}[2,5,8,15]$为索引，将$\Delta FX'_{10}[8,9,10]$、$FW'_{10}[2,5,8,15]$、$FX'_{10}[8,9,10]$存储在表$H_2$中。表$H_2$中有$2^{16}$行，平均每行有$2^{12}$个值。

表$H_3\{ERK_{11}[3,6,9,12]\}$：猜测$\Delta FX'_{10}[12,13,15]$和$FY'_{10}[12,13,14,15]$的全部$2^{28}$个可能值，由此计算出$FX'_{10}[12,13,15]$、$\Delta FX'_{11}[3,6,9,12]$、$\Delta FW'_{10}[3,6,9,12]$、$FW'_{10}[3,6,9,12]$，以$\Delta FX'_{11}[3,6,9,12]$为索引，将$\Delta FX'_{10}[12,13,15]$、$FW'_{10}[3,6,9,12]$、$FX'_{10}[12,13,15]$存储在$H_3$中。表中有$2^{16}$行，平均每行有$2^{12}$个值。

c. 猜测第9轮中$\Delta FX'_9$不为0的状态差分以及FY'_9的值，同时结合密文对差分以及密文值计算出满足要求的轮子密钥ERK_{10}的数量。

表$H_4\{ERK_{10}[5,8,15]\}$：猜测$\Delta FX'_9[10]$和$FY'_9[8,9,11]$的全部2^{16}个可能值，由此计算出$FX'_9[10]$、$\Delta FX'_{10}[5,8,15]$、$\Delta FW'_9[5,8,15]$、$FW'_9[5,8,15]$，以$\Delta FX'_{10}[5,8,15]$作为索引，将$\Delta FX'_9[10]$、$FW'_9[5,8,15]$、$FX'_9[10]$存储在H_4中。表H_4中有2^{12}行，平均每行有2^4个值。

表$H_5\{ERK_{10}[7,10,13]\}$：猜测$\Delta FX'_9[0]$和$FY'_9[1,2,3]$的全部2^{16}个可能值，由此计算出$FX'_9[0]$、$\Delta FX'_{10}[7,10,13]$、$\Delta FW'_9[7,10,13]$、$FW'_9[7,10,13]$，以$\Delta FX'_{10}[7,10,13]$作为索引，将$\Delta FX'_9[0]$、$FW'_9[7,10,13]$、$FX'_9[0]$存储在表H_5中。表H_5中有2^{12}行，平均每行有2^4个值。

表$H_6\{ERK_{10}[6,9,12]\}$：猜测$\Delta FX'_9[13]$和$FY'_9[12,14,15]$的全部2^{16}个可能值，由此计算出$FX'_9[13]$、$\Delta FX'_{10}[6,9,12]$、$\Delta FW'_9[6,9,12]$、$FW'_9[6,9,12]$，以$\Delta FX'_{10}[6,9,12]$作为索引，将$\Delta FX'_9[13]$、$FW'_9[6,9,12]$以及$FX'_9[13]$存储在H_6中。表H_6中有2^{12}行，平均每行有2^4个值。

d. 由性质2可以得出$ERK_9[0,10,13]$可以由$ERK_{11}[8,9,10,11,12,14,15]$推出，随后通过猜测第8轮$\Delta FX'_8$中不为0的状态差分以及$FY'_8$的值，同时结合密文对差分以及密文值计算出满足要求的轮子密钥的数量。

表$H_7\{ERK_9[0,10,13]\}$：猜测$\Delta FX'_8[1]$和$FY'_8[0,2,3]$的全部2^{16}个可能值，由此计算出$FX'_8[1]$、$\Delta FW'_8[0,10,13]$、$FW'_8[0,10,13]$、$\Delta FX'_9[0,10,13]$，并且以$FW'_8[0,10,13]$和$\Delta FX'_9[0,10,13]$为索引，将$FW'_8[0,10,13]FX'_8[1]$、$\Delta FX'_8[1]$存储在表H_7中。表H_7中有2^{24}行，平均每2^8行有1个值。

e. 通过猜测第7轮中ΔFZ_7不为0的状态差分以及FW_7的值，同时结合密文对差分以及密文值计算出满足要求的轮子密钥ERK_8的数量。

表$H_8\{ERK_8[1]\}$：猜测$\Delta FZ_7[7]$和$FZ_7[7]$的全部2^8个可能值，由此计算出$\Delta FW_7[1]$、$FW_7[1]$、$\Delta FX_8[1]$，并且以$\Delta FX_8[1]$为索引，将$\Delta FW_7[1]$和$FW_7[1]$存储在表H_8中。表H_8中有2^4行，平均每行有2^4个值。

2.5.2 在线阶段

本文中描述的截断差分路径的概率为2^{-84}，具体分析如下：差分$\Delta FW_4[0,1,2,3,8,9,10,11,12,13,14,15]$经过列混淆操作之后生成了$\Delta FX_5[1,2,3,8,9,11,12,14,15]$（记作$\Delta FW_4[0,1,2,3,8,9,10,11,12,13,14,15]\rightarrow\Delta FX_5[1,2,3,8,9,11,12,14,15]$），该操作会产生一定的概率，大小为$2^{-12}$；当$\Delta FW_5[5,6,7,8,9,10,12,13,15]\rightarrow\Delta FX_6[4,11,14]$时，所产生的概率大小为$2^{-36}$；当$\Delta FW_6[4,5,6]\rightarrow\Delta FX_7[7]$时，所产生的概率为$2^{-12}$；当$\Delta FW_1[8,9,10,12,13,15]\rightarrow\Delta FX_2[11,14]$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$2^{-12-36-12-24}=2^{-84}$。

为了找到正确的消息对，并没有猜测轮子密钥和检查明密文对是否满足截断差分特征，而是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$\Delta FX_7[7]$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。在线阶段的具体攻击过程描述如下：

a. 定义一个包含2^{24}个明文的结构，其中$FX_1[2,3,8,9,12,15]$取所有可能的2^{24}个值，其余的10个半字节被固定为一些常量。因此，一个结构可以生成$2^{24}\times(2^{24}-1)/2\approx 2^{47}$个明文对，且每个对都满足明文差分。现需使用$2^{37}$个这样的结构来生成$2^{37}\times 2^{47}=2^{84}$个明文对。在这个$2^{84}$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^{-84}。

b. 对选择的2^{84}个明文对进行加密并且筛选出满足$\Delta FX'_{11}[0,7,10,13]$这4个位置为固定值的密文对，由此共有$2^{84-16}=2^{68}$个明密文对，并且利用这些密文对来计算出密文对差分值。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（$RK_7[7]$、$ERK_9[0,10,13]$、$ERK_{11}[1\sim 6,8,9,11,12,14,15]$、$ERK_{10}[5\sim 10,12,13,15]$ $ERK_8[1]$、$RK_1[2,3,8,9,12,15]$）

a) 首先，对筛选后的明文−密文对按照以下步骤进行处理，以获得轮子密钥$ERK_{11}[1\sim 6,8,9,11,12,14,15]$的所有可能值。

(a) 根据密文对状态值，可以计算到$FX'_{11}[1,4,11,14]$和$\Delta FX'_{11}[1,4,11,14]$的值，以$\Delta FX'_{11}[1,4,11,14]$为索引访问$H_1$，得到$FW'_{10}[1,4,11,14]$，并计算等式$ERK_{11}[1,4,11,14]\oplus FW'_{10}[1,4,11,14]=FX'_{11}[1,4,11,14]$。因此，$ERK_{11}[1,4,11,14]$有$2^{12}$个可能值。

(b) 根据密文对状态值，可以计算得到$FX'_{11}[2,5,8,15]$和$\Delta FX'_{11}[2,5,8,15]$的值，以$\Delta FX'_{11}[2,5,8,15]$为索引访问$H_2$，得到$FW'_{10}[2,5,8,15]$，并计算等式$ERK_{11}[2,5,8,15]\oplus FW'_{10}[2,5,8,15]=FX'_{11}[2,5,8,15]$。因此，$ERK_{11}[1,2,4,5,8,11,14,15]$有$2^{12+12}=2^{24}$个可能值。

(c) 根据密文对状态值，可以计算得到$FX'_{11}[3,6,9,12]$和$\Delta FX'_{11}[3,6,9,12]$的值，以$\Delta FX'_{11}[3,6,9,12]$为索引访问$H_3$，得到$FW'_{10}[3,6,9,12]$，并计算等式$ERK_{11}[3,6,9,12]\oplus FW'_{10}[3,6,9,12]=FX'_{11}[3,6,9,12]$。因此，$ERK_{11}[1\sim 6,8,9,11,12,14,15]$有$2^{24+12}=2^{36}$个可能值。

b) 然后，对筛选后的明文−密文对按照以下步骤进行处理，以获得轮子密钥$ERK_{10}[5\sim 10,12,13,15]$的所有可能值。

(a) 由步骤a) 中对H_1、H_2、H_3查表的结果，可以得到$\Delta FX'_{10}[5,8,15]$以及$FX'_{10}[5,8,15]$，随后以$\Delta FX'_{10}[5,8,15]$为索引查询表H_4，可以得到$FW'_9[5,8,15]$，并且通过利用该等式$ERK_{10}[5,8,15]\oplus FW'_9[5,8,15]=FX'_{10}[5,8,15]$求得$ERK_{10}[5,8,15]$，由此可以获得$ERK_{11}[1\sim 6,8,9,11,12,14,15],ERK_{10}[5,8,15]$共计$2^{36+4}=2^{40}$个可能值。

(b) 由步骤a) 中对H_1、H_2、H_3查表的结果，可以得到$\Delta FX'_{10}[7,10,13]$和$FX'_{10}[7,10,13]$，随后以$\Delta FX'_{10}[7,10,13]$为索引查询表H_5，可以得到$FW'_9[7,10,13]$，并且通过等式$ERK_{10}[7,10,13]\oplus FW'_9[7,10,13]=FX'_{10}[7,10,13]$，求得$ERK_{10}[7,10,13]$，由此可以获得$ERK_{11}[1\sim 6,8,9,11,12,14,15],ERK_{10}[5,7,8,10,13,15]$共计$2^{40+4}=2^{44}$个可能值。

(c) 由步骤a) 中对H_1、H_2、H_3查表的结果，可以得到$\Delta FX'_{10}[6,9,12]$和$FX'_{10}[6,9,12]$，随后以$\Delta FX'_{10}[6,9,12]$为索引查询表H_6，可以得到$FW'_9[6,9,12]$，并且通过等式$ERK_{10}[6,9,12]\oplus FW'_9[6,9,12]=FX'_{10}[6,9,12]$，求得$ERK_{10}[6,9,12]$，由此可以获得$ERK_{11}[1\sim 6,8,9,11,12,14,15]$、$ERK_{10}[5\sim 10,12,13,15]$，共计$2^{44+4}=2^{48}$个可能值。

c) 由于$ERK_9[0,10,13]$可以通过$ERK_{11}[8,9,11,12,14,15]$推导出来。继续对筛选后的明文−密文对按照以下步骤进行处理。

由步骤b) 的H_4、H_5、H_6对查询的过程可以得到$\Delta FW'_9[5\sim 10,12,13,15]$和$FW'_9[5\sim 10,12,13,15]$，通过计算推导可以求出$\Delta FX'_9[0,10,13]$和$FX'_9[0,10,13]$，随后通过计算出$FW'_8[0,10,13]=FX'_9[0,10,13]\oplus ERK_9[0,10,13]$，并且以$\Delta FX'_9[0,10,13]$和$FW'_8[0,10,13]$为索引，查询表$H_7$，每$2^8$行中可以找到1个值。因此对于$ERK_{11}[1\sim 6,8,9,11,12,14,15]$、$ERK_{10}[5\sim 10,12,13,15]$、$ERK_9[0,10,13]$有$2^{48-8}=2^{40}$个值。

d) 继续对经过了上述操作筛选过后的明文−密文对进行下列的操作以获得轮子密钥$ERK_8[1]$。

由步骤c) 中对表H_7的查询的结果可以获得$\Delta FX'_8[1]$和$FX'_8[1]$，随后以$\Delta FX'_8[1]$为索引查询表H_8，可以获得$FW_7[1]$，并且通过等式$ERK_8[1]\oplus FW_7[1]=FX'_8[1]$，求得$ERK_8[1]$，由此可以获得$ERK_{11}[1\sim 6,8,9,11,12,14,15]$、$ERK_{10}[5\sim 10,12,13,15]$ $ERK_9[0,10,13]$、$ERK_8[1]$，共计$2^{40+4}=2^{44}$个可能值。

e) 继续对经过了上述操作筛选过后的明文−密文对进行下列的操作以获得轮子密钥$RK_1[2,3,8,9,12,15]$。

猜测$ERK_{11}[0]$、$ERK_{11}[7]$、$ERK_{11}[10]$的全部2^{12}的可能值，在先前的步骤中已经猜测了ERK_{11}

[1 ~ 6,8,9,11,12,14,15]的可能值，因此由性质3、性质4、性质5可以求出$RK_1[3,9,12]$，由此可以获得 $ERK_{11}[1 \sim 6,8,9,11,12,14,15]$、$ERK_{10}[5 \sim 10,12,13,15]$、$ERK_9[0,10,13]$、$ERK_8[1]$、$RK_1[3,9,12]$共计$2^{44+12} = 2^{56}$个可能值。同时，根据$RK_1[3,9,12]$的值，可以由明文$FX_1[3,9,12]$和明文对差分$\Delta FX_1[3,9,12]$求出$FW_1[8,9,10]$，且由$FW_1[8,9,10]$计算出满足$FX_2[8,9,10]$处为0的概率为$2^{-12}$，则此时满足该情况的密钥$ERK_{11}[1 \sim 6,8,9,11,12,14,15]$、$ERK_{10}[5 \sim 10,12,13,15]$、$ERK_9[0,10,13]$、$ERK_8[1]$、$RK_1[3,9,12]$有$2^{56-12} = 2^{44}$个可能值。

猜测$ERK_{11}[13]$的全部的2^4可能值，在先前的步骤中已经猜测了$ERK_{11}[1 \sim 6,8,9,11,12,14,15]$的可能值，因此由性质3、性质4、性质5和性质6可以求出$RK_1[2,8,15]$，由此可以获得轮密钥$ERK_{11}[1 \sim 6,8,9,11,12,14,15]$、$ERK_{10}[5 \sim 10,12,13,15]$ $ERK_9[0,10,13]$、$ERK_8[1]$、$RK_1[2,3,8,9,12,15]$共计$2^{44+4} = 2^{48}$个可能值。同时，根据$FX_1[2,8,15]$的值，可以由明文$FX_1[2,8,15]$和明文对差分$\Delta FX_1[2,8,15]$可以求出$FW_1[12,13,15]$，且由$FW_1[12,13,15]$计算出满足$FX_2[12,13,15]$处为0的概率为2^{-12}，则此时满足该情况的密钥$ERK_{11}[1 \sim 6,8,9,11,12,14,15]$、$ERK_{10}[5 \sim 10,12,13,15]$、$ERK_9[0,10,13]$、$ERK_8[1]$、$RK_1[2,3,8,9,12,15]$有$2^{48-12} = 2^{36}$个可能值。

c. 对于上述操作所得到的有序序列$\Delta FX_7[7]$，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$2^{76-128} = 2^{-52}$，因此大约还有$1+2^{76-128} = 2^{-52}$个候选轮子密钥剩余。根据密钥编排，奇数轮的子密钥可由主密钥K的前64比特确定，偶数轮的子密钥可由主密钥K的后64比特确定，且由于$ERK_{11}[0 \sim 15]$的全部64比特都已被猜测，因此对于主密钥还剩余除了$ERK_{10}[5 \sim 10,12,13,15]$和$ERK_8[1]$外的6个未知半字节，即24个未知比特未被猜测。最后穷举搜索2^{52}个候选轮子密钥和24个未知比特来恢复主密钥。

2.5.3 复杂度分析

根据在线阶段的步骤a，可以清楚地发现攻击的数据复杂度为$2^{37+24} = 2^{61}$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$2^{76} \times 128/64 \approx 2^{77}$个64比特块。预计算阶段的时间复杂度大约为$2^{76} \times 2^5 \times 30/(16 \times 10) \approx 2^{78.58}$次加密，在线阶段的时间复杂度为$2^{68} \times 2^5 \times 2^{56} \times 3/(16 \times 10) \approx 2^{123.26}$次加密。

对于剩下的2^{52}个候选子密钥以及24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$2^{52} \times 2 \times 2^{24} = 2^{77}$加密。

综上所述，该攻击的时间复杂度大约为$2^{123.26}$次加密，数据复杂度为2^{61}个选择明文，存储复杂度为2^{77}个64比特块。

3 结 论

本文主要分析了IVLBC密码算法抵抗中间相遇攻击的安全性。首先利用IVLBC密码算法的结构特性，针对它的PN，用一个约束条件描述一个PN的过程，针对异或运算采用3个约束条件来描述一个异或运算的过程；随后结合算法3和不等式约束条件，构建出基于MILP的自动化搜索模型，该模型共计1 200条约束条件，通过Gurobi求解器，寻找出10条5轮的中间相遇区分器；然后结合IVLBC密钥编排的特点，对10条中间相遇区分器分别在前后添加若干轮进行密码攻击；最后选取需要猜测密钥位最少，且攻击轮数最长的5轮中间相遇区分器（即表5中的Δ_3），在该区分器的基础上，往前添加1轮，往后添加4轮，构成10轮的IVLBC-128中间相遇攻击，在攻击过程当中，使用预计算表、等效密钥和轮子密钥之间的线性关系等技巧提升攻击的效率，降低攻击的复杂度。该攻击过程的时间复杂度为$2^{123.26}$次加密，数据复杂度为2^{61}个选择明文，存储复杂度为2^{77}个64比特块，比已有的结果提高了2轮。该结果是目前对IVLBC最好的中间相遇攻击结果。研究表明，结合MILP模型进行中间相遇攻击的方法可以达到很好的攻击效果，值得深入研究，进一步挖掘基于MILP搜索进行中间相遇攻击抑或是与其他攻击方式相结合的潜力，如进行更为精确的建模，减少搜索到有效区分器所需的时间；更加深入研究IVLBC算法构造，寻找出更长轮数的区分器等将是下一步继续研究的问题。

参考文献：

[1] BIHAM E, BIRYUKOV A, SHAMIR A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible

differentials[C]//Proceedings of International Conference on the Theory and Application of Cryptographic Techniques on Advances in Cryptology – EUROCRYPT ’99. Prague: Springer, 1999: 12–23.

[2] DIFFIE W, HELLMAN M E. Special feature exhaustive cryptanalysis of the NBS data encryption standard[J]. Computer, 1977, 10(6): 74−84.

[3] 袁征, 朱亮, 赵晨曦, 等. 基于 MILP 搜索的 ANU 算法积分分析 [J]. 计算机应用研究, 2021, 38(4): 1171−1174.

[4] 黄长阳, 王韬, 陈浩, 等. SIMECK 密码代数故障攻击研究 [J]. 计算机应用研究, 2019, 36(7): 2184−2189.

[5] 王恺, 蔡爵嵩, 严迎建. 基于 MLP 神经网络的分组密码算法能量分析研究 [J]. 计算机应用研究, 2021, 38(3): 881−885,892.

[6] MOUHA N, WANG Q J, GU D W, et al. Differential and linear cryptanalysis using mixed-integer linear programming[C]//Proceedings of 7th International Conference on Information Security and Cryptology. Beijing: Springer, 2012: 57–76.

[7] Mouha N, Preneel B. Towards finding optimal differential characteristics for ARX: Application to Salsa20[PP/OL]. Cryptology ePrint Archive, 2013. https://eprint.iacr.org/2013/468.

[8] SUN S W, HU L, WANG P, et al. Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers[C]//Proceedings of 20th International Conference on the Theory and Application of Cryptology and Information Security on Advances in Cryptology -- ASIACRYPT 2014. Taiwan, China: Springer, 2014: 158–178.

[9] ABDELKHALEK A, SASAKI Y, TODO Y, et al. MILP modeling for (large) s-boxes to optimize probability of differential characteristics[J]. IACR Transactions on Symmetric Cryptology, 2017, 2017(4): 99−129.

[10] ZHANG Y J, SUN S W, CAI J H, et al. Speeding up MILP aided differential characteristic search with Matsui’ s strategy[C]//Proceedings of 21st International Conference on Information Security. Guildford: Springer, 2018: 101–115.

[11] ZHOU C N, ZHANG W T, DING T Y, et al. Improving the MILP-based security evaluation algorithm against differential/linear cryptanalysis using a divide-and-conquer approach[J]. IACR Transactions on Symmetric Cryptology, 2020, 2019(4): 438−469.

[12] SADEGHI S, RIJMEN V, BAGHERI N. Proposing an MILP-based method for the experimental verification of difference-based trails: application to SPECK, SIMECK[J]. Designs, Codes and Cryptography, 2021, 89(9): 2113−2155.

[13] CAO W W, ZHANG W T, ZHOU C N. New automatic search tool for searching for impossible differentials using undisturbed bits[C]//Proceedings of 18th International Conference on Information Security and Cryptology. Beijing: Springer, 2022: 43–63.

[14] LIU Y, XIANG Z J, CHEN S W, et al. A novel automatic technique based on MILP to search for impossible differentials[C]//Proceedings of 21st International Conference on Applied Cryptography and Network Security. Kyoto: Springer, 2023: 119–148.

[15] HUANG X T, LI L, YANG J L. IVLBC: an involutive lightweight block cipher for internet of things[J]. IEEE Systems Journal, 2023, 17(2): 3192−3203.

[16] İLTER M B, SELÇUK A A. Differential and linear cryptanalysis of IVLBC via MILP modeling[C]//2023 16th International Conference on Information Security and Cryptology (ISCTürkiye). Ankara: IEEE, 2023: 1–5.

[17] UCHIYAMA Y, IGARASHI Y. Meet-in-the-middle Cryptanalysis of IVLBC[C]//2024 Tenth International Conference on Communications and Electronics (ICCE). Danang: IEEE, 2024: 445–450.

[18] KAUR M, DEY D. MILP based differential cryptanalysis on IVLBC and Eslice 64[J]. Defence Science Journal, 2024, 74(5): 651−661.

[19] YU B, CHEN W, LIU W F, et al. Integral cryptanalysis of IVLBC suitable for IoT sensors[J]. IEEE Sensors Journal, 2025, 25(22): 42221−42230.

[20] DEMIRCI H, SELÇUK A A. A meet-in-the-middle attack on 8-round AES[C]//Proceedings of 15th International Workshop on Fast Software Encryption. Lausanne: Springer, 2008: 116–126.

（编辑：何代华）