https://crad.ict.ac.cn
63(11): 2856−2870, 2026

基于代数结构的 LLLWBC 积分攻击

叶 涛[1,2,3] 焦留洋[1,3] 韦永壮[1,2,3] 李灵琛[1,3] 周尔哲[1,3]

[1]（桂林电子科技大学计算机与信息安全学院 广西桂林 541004）

[2]（密码科学技术国家重点实验室 北京 100878）

[3]（广西密码学与信息安全重点实验室（桂林电子科技大学） 广西桂林 541004）

（fendouyetao@163.com）

Integral Attack for LLLWBC Based on Algebraic Structure

Ye Tao[1,2,3], Jiao Liuyang[1,3], Wei Yongzhuang[1,2,3], Li Lingchen[1,3], and Zhou Erzhe[1,3]

[1]（*School of Computer and Information Security*, *Guilin University of Electronic Technology*, *Guilin*, *Guangxi* 541004）

[2]（*State Key Laboratory of Cryptology Science and Technology*, *Beijing* 100878）

[3]（*Guangxi Key Laboratory of Cryptography and Information Security* (*Guilin University of Electronic Technology*), *Guilin*, *Guangxi* 541004）

Abstract Automated analysis techniques for lightweight block cipher algorithms have been a hot topic in recent research. The LLLWBC algorithm, proposed at the Inscrypt 2022 conference, is a novel low-latency lightweight block cipher that demonstrates excellent encryption and decryption performance in both software and hardware implementations. However, its security against emerging cryptanalytic techniques requires further investigation. In this paper, a new technique for constructing word-based propagation trails in block cipher algorithms is introduced, an automated detection method for seven distinct algebraic structures is proposed, then an automated search approach for integral distinguishers is also proposed based on these algebraic structures. Furthermore, the construction method for key guessing trails during the key recovery process of integral attacks is investigated, and the optimal key guessing set can be obtained by using this method. Based on the proposed method, we construct a 9-round integral distinguisher for LLLWBC with a data complexity of 2^{52} plaintext blocks, which are one round longer than the distinguishers given by the designers. Subsequently, based on the obtained 9-round integral distinguisher, a 12-round key-recovery attack of LLLWBC is proposed, which requires the data complexity of about $2^{56.17}$ chosen data, the memory complexity of about 2^{72} 128b guess key and 2^{52} 64b ciphertexts, the time complexity of about $2^{122.09}$ times encryptions of 12-round LLLWBC. Experimental results demonstrate that the 12-round LLLWBC algorithm is vulnerable to integral attacks, while the full 21-round LLLWBC maintains a sufficient security margin. The construction method of integral distinguisher and key recovery technique proposed in this paper can also be applied to other word-oriented block ciphers.

Key words algebraic structure; automated analysis method; integral distinguisher; LLLWBC; key recovery

摘 要 轻量级分组密码算法的自动化分析技术是近些年研究的热点内容。LLLWBC 算法是在 Inscrypt 2022 会议上提出的新型低延迟轻量级分组密码算法，其在软件和硬件实现上具有非常优秀的加解密性能。然而，针对一些新的安全性分析技术，LLLWBC 算法是否安全还需要进一步研究。提出分组密码算法新

收稿日期：2025-10-14；**修回日期**：2026-06-12

基金项目：国家自然科学基金项目（62402132, 62162016）；桂林电子科技大学研究生教育创新计划项目（2025YCXS053）

This work was supported by the National Natural Science Foundation of China (62402132, 62162016) and Innovation Project of GUET Graduate Education (2025YCXS053).

通信作者：韦永壮（walker_wyz@guet.edu.cn）

的字传播轨迹构造技术，给出了 7 种不同布尔代数结构的自动化检测方法，并基于代数结构给出了积分区分器的自动化搜索方法。同时，研究了积分攻击密钥恢复过程中密钥猜测轨迹的构造方法，该方法能够自动地搜索到最优的密钥猜测集合。利用上述方法，构造了 LLLWBC 算法 9 轮的积分区分器，需要的数据复杂度为 2^{52} 个 LLLWBC 明文分组，相比之前的积分区分器轮数增加 1。随后，利用得到的 9 轮积分区分器，能够对 12 轮的 LLLWBC 算法进行密钥恢复攻击，该攻击需要的数据复杂度为 $2^{56.17}$ 个 64b 明文数据，存储复杂度为 2^{72} 个 128b 的猜测密钥和 2^{52} 个 64b 的密文数据，时间复杂度为 $2^{122.09}$ 次 12 轮加密操作。实验结果表明：12 轮的 LLLWBC 算法无法抵抗积分攻击，全轮为 21 轮的 LLLWBC 算法仍然具有足够的安全冗余。提出的积分区分器构造方法和密钥恢复方法同样适用于其他基于字运算的密码算法。

关键词 代数结构；自动化分析方法；积分区分器；LLLWBC；密钥恢复

中图法分类号 TP311.13；TP309

DOI: 10.7544/issn1000-1239.202550746 **CSTR:** 32373.14.issn1000-1239.202550746

分组密码算法具有加解密速度快、安全性高以及易于标准化的特点，对于保证信息的机密性和完整性等方面发挥着非常重要的作用。然而，随着物联网、RFID 以及无线通信技术的快速进步，一方面，其面临的信息安全问题越发突出；另一方面，这些应用环境除了具有硬件资源有限的特点，有时还需要具有低延迟的优势。针对这样的需求，具有低延迟优势的分组密码算法的设计成为了近些年研究的热点内容，最近提出了多种不同类型的低延迟分组密码算法，如 PRINCE, MANTIS, QARMA 等。在此基础上，Zhang 等人[1]提出了新型的低延迟轻量级分组密码算法——LLLWBC。

对于设计完成的轻量级密码算法，需要评估其安全性，包括差分密码分析[2]、线性密码分析[3]、中间相遇攻击[4]、代数攻击[5]以及积分攻击[6]等。其中，由于可分性的提出，利用可分性结合自动化分析工具，搜索密码算法的积分区分器成为了近些年研究的热点内容。然而，随着攻击轮数的增加，基于可分性的分析模型规模逐渐增大，求解需要消耗的时间也呈现指数增加。如何更快速高效地搜索积分区分器，并给出密钥恢复过程最优的密钥猜测集合，这些是迫切需要解决的问题。

1 相关工作

积分分析最初是由 Knudsen 等人[6]为了评估 Square 算法的安全性提出的分析方法，通过选取合适的明文活跃比特和常数比特构建相应的数据集，然后对这个数据集进行加密，如果得到的密文集合中一些位置的异或和恒定为 0，且这个性质与密钥的选取无关，则攻击者得到了一个积分区分器。

积分区分器的构建方法主要包括 2 种：第 1 种主要是利用代数性质构建积分区分器，即评估密码算法输出对应的代数表达式的代数次数，通过代数次数确定积分区分器的形式，这种方法随着可分性的提出被深入研究。可分性最初是由 Todo[7]在 2015 年的欧密会上提出的一种广义的积分性质，在此基础上，Todo 等人[8]提出了比特可分性。为了进一步加速构建积分区分器的效率，Xiang 等人[9]给出了可分性传播的 MILP 模型以及积分区分器的自动化搜索方法。随后，为了进一步提高攻击的轮数，三子集可分性[10]、无未知结合的三子集可分性[11]和单项式预测技术[12]先后被提出，并被应用于积分区分器的搜索[10-14]。然而，这些方法非常精细地构建了密码算法轮函数的分析模型，尤其是对于大状态的密码算法，随着攻击轮数的增加，比特级可分性的描述需要大量不等式，模型求解的效率将大幅度降低。第 2 种构建积分区分器的方法是利用积分性质的传播规则，主要是利用活跃字节、常数字节和平衡字节之间的转化关系，这种方法通常只考虑字节或者半字节的传播轨迹。在此基础上，基于置换函数的性质，Zhang 等人[15]通过统计明文字或者密文字在内部状态的表达式中出现的次数，评估密码算法抵抗积分攻击的能力。在此基础上，叶涛等人[16]引入了字传播轨迹的概念，并给出了字传播轨迹的自动化分析模型。随后，邢朝辉等人[17]给出了利用代数结构构造积分区分器的框架，并基于差分分布表的性质，给出了一种存在积分性质的新代数结构，并将其应用于对 WARP 算法的安全性分析。

本文提出了一种密码算法代数结构的新型自动化分析技术。首先，在搜索密码算法代数结构的过程中，一方面考虑了明文字在中间状态中出现的次

数, 另一方面也研究了每个明文字的分量运算至对应的密文字需要经过的 S 盒的个数, 我们针对这 2 方面提出了新的字传播轨迹概念, 给出了自动化检测代数结构的方法。同时, 基于得到的代数结构给出了积分区分器的具体形式。然后, 在构造的积分区分器的基础上, 基于轮函数结构与密钥编排原理, 进一步设计了密钥恢复的自动化建模流程, 将最优密钥猜测路径转化为 MILP 问题, 通过求解模型得到最优的密钥猜测集合, 实现了积分区分器到密钥恢复的自动衔接。

利用这个方法, 本文对 LLLWBC 抵抗积分攻击的能力进行评估, 构造了 LLLWBC 的 9 轮积分区分器, 并给出了 12 轮的密钥恢复攻击, 数据复杂度为 $2^{56.17}$ 个 64b 明文数据, 时间复杂度为 $2^{122.09}$ 次 12 轮加密, 存储复杂度为 2^{72} 个 128b 的密钥和 2^{52} 个 64b 的密文数据, 与其他文献的对比见表 1。

Table 1 LLLWBC Integral Attack Compared with the Existing Results

表 1 LLLWBC 积分攻击与已有结果对比

攻击轮数	活跃半字节数量	平衡半字节数量	时间复杂度	数据复杂度	存储复杂度	结果来源
12	13	6	$2^{122.09}$	$2^{56.17}$	2^{72}	本文
8	13	6		2^{52}		文献 [1]

Table 2 Symbolic Representation and Meaning

表 2 符号表示与含义

符号	说明
k^{wb}	预白化密钥
k^{we}	后白化密钥
M	明文
C	密文
$\boldsymbol{X}^{r-1}$	第 r 轮的输入状态
$\boldsymbol{Y}^{r-1}$	定义解密方向第 r 轮的输入
x_w^r	第 r 轮输出状态的第 w 个字
$(x_w^0)_k^{r,j}$	第 k 个出现在 x_j^r 表达式中的变量 x_w^0
$A_j^{r,w}$	x_w^0 在 x_j^r 的代数表达式中出现的次数
$B_{j,k}^{r,w}$	$(x_w^0)_k^{r,j}$ 经过运算得到 x_j^r 需要经过的 S 盒的个数
$\boldsymbol{C}^{r,w}$	密文的第 w 个字 y_w^0 经过 r 轮解密后, 内部状态 $\boldsymbol{Y}^r$ 表达式中出现的次数
$T^{0,w}\to T^{1,w}\to\cdots\to T^{r,w}$	r 轮的字传播轨迹
r_w^e	加密方向第 w 个字活跃, 最长积分区分器的轮数
ϕ^w	第 w 个字活跃, 最长积分区分器平衡字的下标集合
r_w^d	解密方向第 w 个字未达到全扩散的最大轮数
φ^w	第 w 个字活跃, 解密 r_w^d 轮未扩散到的状态字的下标集合
$\boldsymbol{a}^{r-1}$	模型变量表示第 r 轮函数的输入状态的值 $\boldsymbol{X}^{r-1}$ 是否能够被猜测确定
$\boldsymbol{K}^{r-1}$	模型变量表示第 r 轮轮函数中使用的密钥 $\boldsymbol{k}^{r-1}$ 是否被猜测
$\|$	2 个向量的拼接

2 预备知识

为清晰表述算法结构与攻击过程, 同时确保符号使用的规范性与一致性, 本文预先对核心符号、变量及数学约定在表 2 中进行了集中说明, 相关定义将贯穿全文。

2.1 LLLWBC 算法概述

LLLWBC 算法[1] 采用 64b 分组长度和 128b 密钥长度, 记作 LLLWBC-64/128, 其轮函数结构为扩展广义 Feistel 结构(GFS), 共迭代 21 轮。LLLWBC 包括密钥调度算法、加密算法和解密算法 3 部分, 且加密和解密算法结构相同, 算法轮函数的具体结构如图 1 所示。因此, 下面仅介绍 LLLWBC 算法的密钥生成算法和加密算法。

1)密钥生成算法。LLLWBC 的主密钥是 128b, 用于生成 2 个 64b 的白化密钥和 21 个 32b 的轮密钥。

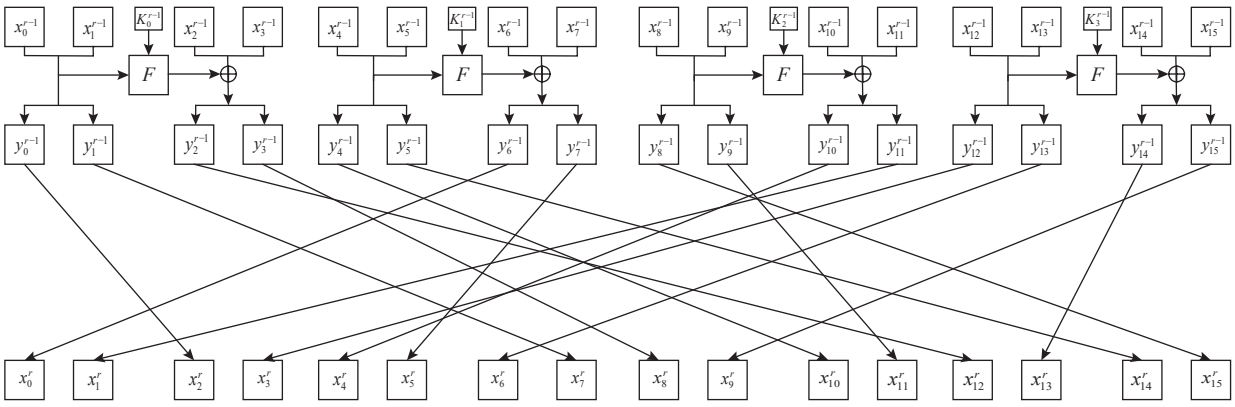


Fig. 1 The r-th round function of LLLWBC($0\leqslant r\leqslant 21$)

图 1 LLLWBC 的第 r 轮函数($0\leqslant r\leqslant 21$)

首先，128b 的主密钥 k 被分为 2 个 64b 的部分，$k=k^{\mathrm{W}}\|k^{\mathrm{e}}=\{k_0,k_1,\cdots,k_{15}\|k_{16},k_{17},\cdots,k_{31}\}$，$k_i\in F_2^4$，$0\leqslant i\leqslant 31$，定义左面为高位。预白化密钥和后白化密钥使用符号 k^{wb} 和 k^{we}，其生成方式为：

$$k^{\mathrm{wb}}=k^{\mathrm{W}},$$

$$k^{\mathrm{we}}=(k^{\mathrm{W}}\ggg 1)\oplus(k^{\mathrm{W}}\gg 63)。$$

对于奇数轮 $r=1,3,\cdots,21$，k^{e} 的高 32 位与对应的轮常数异或后生成第 r 轮的子密钥 k^{r-1}；对于偶数轮 $r=2,4,\cdots,20$，k^{e} 的低 32 位与相应的轮常数 ck^r 异或运算后生成第 r 轮子密钥 k^{r-1}。轮常数的具体细节请参考文献 [1]。

2）加密算法。令 M 和 C 分别表示为 64b 明文和密文，$\boldsymbol{X}^r=(x_0^{r-1},x_1^{r-1},\cdots,x_{15}^{r-1})$ 表示第 r 轮函数的输入状态，其中 $x_i^{r-1}\in F_2^4$，$0\leqslant i\leqslant 15$。第 r 轮的子密钥为 $\boldsymbol{k}^{r-1}=(k_0^{r-1},k_1^{r-1},k_2^{r-1},k_3^{r-1})=(k_{0,0}^{r-1}\|k_{0,1}^{r-1},k_{1,0}^{r-1}\|k_{1,1}^{r-1},k_{2,0}^{r-1}\|k_{2,1}^{r-1},k_{3,0}^{r-1}\|k_{3,1}^{r-1})$，$k_i^{r-1}\in F_2^8$，$k_{i,j}\in F_2^4$，$1\leqslant r\leqslant 21$，$0\leqslant i\leqslant 3$，$0\leqslant j\leqslant 1$，则 LLLWBC 的加密过程可以用算法 1 表示。

算法 1. LLLWBC 的加密过程。

输入：分组密码算法的明文 M，预白化密钥 k^{wb}，轮密钥 $(k_0^{r-1}\|k_1^{r-1}\|k_2^{r-1}\|k_3^{r-1})$，$1\leqslant r\leqslant 21$；

输出：密文 $(x_0^{21},x_1^{21},\cdots,x_{15}^{21})$。

① $M\oplus k^{\mathrm{wb}}=x_0^0\|x_1^0\|\cdots\|x_{15}^0$；

② for $1\leqslant r\leqslant 10$ do

③ $x_0^r,\cdots,x_{15}^r=P(x_0^{r-1},x_1^{r-1},F((x_0^{r-1}\|x_1^{r-1})\oplus K_0^{r-1})\oplus(x_2^{r-1}\|x_3^{r-1}),\cdots,x_{12}^{r-1},x_{13}^{r-1},F((x_{12}^{r-1}\|x_{13}^{r-1})\oplus K_3^{r-1})\oplus(x_{14}^{r-1}\|x_{15}^{r-1}))$；

④ end for

⑤ for $11\leqslant r\leqslant 20$ do

⑥ $x_0^r,\cdots,x_{15}^r=P^{-1}(x_0^{r-1},x_1^{r-1},F((x_0^{r-1}\|x_1^{r-1})\oplus K_0^{r-1})\oplus(x_2^{r-1}\|x_3^{r-1}),\cdots,x_{12}^{r-1},x_{13}^{r-1},F((x_{12}^{r-1}\|x_{13}^{r-1})\oplus K_3^{r-1})\oplus(x_{14}^{r-1}\|x_{15}^{r-1}))$；

⑦ end for

⑧ $x_0^r,x_1^r,\cdots,x_{15}^r=x_0^{r-1},x_1^{r-1},F((x_0^{r-1}\|x_1^{r-1})\oplus K_0^{r-1})\oplus(x_2^{r-1}\|x_3^{r-1}),\cdots,x_{12}^{r-1},x_{13}^{r-1},F((x_{12}^{r-1}\|x_{13}^{r-1})\oplus K_3^{r-1})\oplus(x_{14}^{r-1}\|x_{15}^{r-1})$；

⑨ $C=(x_0^{21}\|x_1^{21}\|\cdots\|x_{15}^{21})\oplus k^{\mathrm{we}}$；

⑩ return C。

算法 1 中，置换操作 P 及逆置换 P^{-1} 如表 3 所示。

算法 1 中，F 函数的计算方式为：

$$F:\{0,1\}^8\to\{0,1\}^8,$$

$$(u_0,u_1)\to S(\boldsymbol{A}(S(u_0,u_1)))，$$

其中 S 为 PRINCE 算法 [18] 的 4b 密码 S 盒，线性层 $\boldsymbol{A}$ 由 $GF(2^4)$ 上的 2×2 MDS 矩阵组成，使用的不可约多项式是 x^4+x+1。具体的运算为：

$$\boldsymbol{A}:\{0,1\}^8\to\{0,1\}^8,$$

$$(u_0,u_1)\to\begin{pmatrix}0\mathrm{x}2 & 0\mathrm{x}3\\ 0\mathrm{x}1 & 0\mathrm{x}1\end{pmatrix}\begin{pmatrix}u_0\\ u_1\end{pmatrix}。$$

Table 3 Substitution P and Reverse Substitution P^{-1} of LLLWBC

表 3 LLLWBC 的置换 P 及逆置换 P^{-1}

i	$P(i)$	$P^{-1}(i)$	i	$P(i)$	$P^{-1}(i)$
0	2	6	8	15	3
1	7	11	9	11	15
2	12	0	10	4	4
3	8	12	11	1	9
4	10	10	12	3	2
5	14	7	13	6	14
6	0	13	14	13	5
7	5	1	15	9	8

2.2 基于代数结构的积分区分器构造方法

在文献 [15] 中，提出了一种利用代数结构构造 SPN 结构的分组密码积分区分器的方法，其核心思想是通过将中间状态字表示为明文字或者密文字的代数表达式，并通过统计这些中间状态字在多项式中出现的次数，搜索密码算法中存在的置换函数，利用置换函数的性质，攻击者可以评估算法抵抗积分攻击的能力。置换函数的性质如下：

性质 1. 若 $F:\{0,1\}^b\to\{0,1\}^b$ 是置换函数（即每个输入唯一对应 1 个输出，且所有输出均被覆盖），且 $f:\{0,1\}^b\to\{0,1\}^b$ 是任意布尔函数，则以下结论一定成立：

$$\bigoplus_{\substack{y\in F_2^b,\\ x\text{遍历}F_2^b}}F(x\oplus f(y))=0。\tag{1}$$

性质 2. 设 $S_i(x)$ 是 F_2^b 上的置换，$i=0,1,\cdots,n-1$，c_i 为常数，则 $S_{n-1}(\cdots S_1(S_0(x\oplus c_0)\oplus c_1)\oplus\cdots\oplus c_{n-1})$ 也是关于 x 在 F_2^b 上的一个置换，根据置换函数的性质可得：

$$\bigoplus_{x\in F_2^b}S_{n-1}(\cdots S_1(S_0(x\oplus c_0)\oplus c_1\oplus\cdots\oplus c_{n-1}))=0。\tag{2}$$

基于上述性质，文献 [15] 中给出了积分区分器的构造方法，在加密方向上计算明文中的第 i 个字达到全扩散的轮数，且此时至少存在 1 个输出字对应的表达式中第 i 个字仅出现 1 次，此时对应的轮数定义为 r_{enc}^i；同时，在解密方向上计算第 i 个字第 1 次达到全扩散的轮数，并定义为 r_{dec}^i+1，且解密 r_{dec}^i 轮时，至少存在 1 个输出状态字的表达式中不出现第 i 个密文字。最后，遍历 $0\leqslant i\leqslant m-1$，按照上述方式分别计算 r_{enc}^i 和 r_{dec}^i，并计算集合 $\{r_{\mathrm{enc}}^0+r_{\mathrm{dec}}^0,r_{\mathrm{enc}}^1+r_{\mathrm{dec}}^1,\cdots,r_{\mathrm{enc}}^{m-1}+$

$r_{\mathrm{dec}}^{m-1}\}$的最大值得到$r_{\mathrm{enc}}+r_{\mathrm{dec}}$，此时可说明该密码算法存在$r_{\mathrm{enc}}+r_{\mathrm{dec}}$轮的积分区分器。

文献 [15] 在加密方向上仅考虑待测字出现 1 次的情况，在此基础上，文献 [17] 给出了另外一种存在积分性质的代数结构，具体如下。

性质 3. 假设$S(x)$是F_2^b上的置换，并且 u 和 v 是常数，则

$$\bigoplus_{x\text{遍历}F_2^b} S(S(x)\oplus S(x\oplus u)\oplus v)=0\text{。} \tag{3}$$

性质 3 表明，虽然中间状态字在密文字中出现 2 次，但是如果以式(2)(3)的形式出现，基于差分分布表的性质，当输入字 x 遍历，密文字是平衡的。

对于基于字运算的 SPN 或 Feistel 结构的密码算法，性质 2 和性质 3 给出了 2 种存在积分性质的代数结构，攻击者可以根据密码算法结构推导出中间状态的代数表达式，并判断是否满足上述代数结构，从而确定积分区分器的形式。然而，对于一些结构复杂的密码算法，手动推导其基于字的代数结构需要消耗较多的时间，并且随着攻击轮数的增加，代数表达式也越来越复杂，确定内部状态的表达式是否满足上述形式也越发困难。因此，研究能够自动化确定代数结构的分析工具，对于积分区分器的搜索具有非常重要的意义。

3 基于代数结构的积分区分器的自动化搜索方法

3.1 符号定义

对于一个内部状态为 m 个字，每个字为 n 比特的密码算法，且算法的基本操作单元为 n 比特，定义其第 r 轮的输入状态为 $\boldsymbol{X}^{r-1}$，输出为 $\boldsymbol{X}^r$，其中 $r\geqslant 1$，$\boldsymbol{X}^{r-1}=(x_0^{r-1},x_1^{r-1},\cdots,x_{m-1}^{r-1})$，$x_j^{r-1}\in F_2^b$，则明文状态为 $\boldsymbol{X}^0$。在解密方向，本文定义解密方向第 r 轮的输入为 $\boldsymbol{Y}^{r-1}=(y_0^{r-1},y_1^{r-1},\cdots,y_{m-1}^{r-1})$，输出为 $\boldsymbol{Y}^r$，则密文状态为 $\boldsymbol{Y}^0$。

定义 1. 对于明文的第 w 个字 x_w^0，加密 r 轮后，其在内部状态 $\boldsymbol{X}^r$ 中每个字的表达式中出现次数使用符号 $\boldsymbol{A}^{r,w}=(A_0^{r,w},A_1^{r,w},\cdots,A_{m-1}^{r,w})$进行表示，其中，$A_j^{r,w}$ 表示 x_w^0 在 x_j^r 的代数表达式中出现的次数，其中 $0\leqslant w\leqslant m-1$，$0\leqslant j\leqslant m-1$。同样，对于密文的第 w 个字 y_w^0，解密 r 轮后，其在内部状态 $\boldsymbol{Y}^r$ 表达式中出现的次数使用符号 $\boldsymbol{C}^{r,w}=(C_0^{r,w},C_1^{r,w},C_2^{r,w},\cdots,C_{m-1}^{r,w})$进行表示。

定义 2. 定义 $(x_w^0)_k^{r,j}$ 表示第 k 个出现在 x_j^r 表达式中的变量 x_w^0，也称为 x_w^0 在 x_j^r 中的第 k 个分量。

为了更好地理解定义 2，本文给出了如下的例子。

例 1. 假设经过 r 轮的加密后，得到 x_1^r 的代数表达式为：

$$x_1^r=S\left(x_0^0\oplus c_1\oplus S(x_0^0\oplus c_2)\right)\oplus x_0^0\text{，}$$

其中 S 表示 S 盒运算，c_1 和 c_2 为与 x_0^0 无关的常数。x_0^0 在 x_1^r 的代数表达式中出现 3 次，根据定义 2，可以得到如下的关系：

$$x_1^r=S\left((x_0^0)_0^{r,1}\oplus c_1\oplus S\left((x_0^0)_1^{r,1}\oplus c_2\right)\right)\oplus(x_0^0)_2^{r,1}\text{。}$$

定义 3. 定义 2 维列表 $\boldsymbol{B}^{r,w}$，$\boldsymbol{B}^{r,w}=\{\boldsymbol{B}_0^{r,w},\boldsymbol{B}_1^{r,w},\boldsymbol{B}_2^{r,w},\cdots,\boldsymbol{B}_{m-1}^{r,w}\}$，1 维列表 $\boldsymbol{B}_j^{r,w}$，$\boldsymbol{B}_j^{r,w}=(B_{j,0}^{r,w},B_{j,1}^{r,w},\cdots,B_{j,|\boldsymbol{B}_j^{r,w}|-1}^{r,w})$，其中 $B_{j,k}^{r,w}$ 表示 $(x_w^0)_k^{r,j}$ 经过运算得到 x_j^r 需要经过的 S 盒的个数，$0\leqslant k<|\boldsymbol{B}_j^{r,w}|$。整数向量 $\boldsymbol{B}_j^{r,w}$ 中元素的个数也反映了 x_w^0 在 x_j^r 的代数表达式中出现的次数。

为了更好地理解定义 3，本文给出了如下的例子。

例 2. 假设经过 r 轮的加密后，得到 x_1^r 的代数表达式为：

$$x_1^r=S\left((x_0^0)_0^{r,1}\oplus c_1\oplus S\left((x_0^0)_1^{r,1}\oplus c_2\right)\right)\oplus(x_0^0)_2^{r,1}\text{。}$$

根据上述表达式，$(x_0^0)_0^{r,1}\rightarrow x_1^r$ 的运算经过 1 次 S 盒的运算，$(x_0^0)_1^{r,1}\rightarrow x_1^r$ 的运算需要经过 2 次 S 盒的运算，$(x_0^0)_2^{r,1}\rightarrow x_1^r$ 的运算经过 0 次 S 盒的运算，根据定义 2 可得 $\boldsymbol{B}_1^{r,0}=(1,2,0)$。

3.2 分组密码算法的字传播轨迹

当使用性质 2 搜索积分区分器时，需要确定明文字在密文字中是否出现 1 次；当使性质 3 搜索积分区分器时，需要确定明文字在一些密文字的表达式中出现 2 次，且由这个明文字节的每个分量计算至相应的密文字节都需要经过 2 次 S 盒的运算。因此，本文从 2 方面进行分析：一方面需要确定每个明文字在密文字的表达式中出现的次数，另一方面需要确定每个明文字的分量运算至对应的密文字需要经过的 S 盒的个数。最后，根据这 2 方面的数据，确定相应轮数的积分区分器是否存在，并给出积分区分器的具体形式。

根据对称密码算法的特点可以发现，对于每一个明文字，其在每一轮的输出状态表达式中出现的次数是固定的，且这个明文字传播到密文字经过的 S 盒的个数也是确定的。根据这个特点，本文给出了字传播轨迹的定义，相比于文献 [16] 中的字传播轨迹的定义，本文的定义增加了 $\boldsymbol{B}^{0,w}$ 的传播性质，其中，$0\leqslant w\leqslant m-1$。

定义 4. 字传播轨迹。定义迭代型分组密码算法的轮函数为 f，$T^{r,w}=\{\boldsymbol{A}^{r,w},\boldsymbol{B}^{r,w}\}$，对于 R 轮的密码算法存在如下的传播轨迹：

$$T^{0,w}\xrightarrow{f}T^{1,w}\xrightarrow{f}T^{2,w}\xrightarrow{f}\cdots\xrightarrow{f}T^{R,w}\text{，}$$

并且满足 $\boldsymbol{A}^{0,w} \xrightarrow{f} \boldsymbol{A}^{1,w} \xrightarrow{f} \boldsymbol{A}^{2,w} \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{A}^{R,w}$, $B^{0,w} \xrightarrow{f} B^{1,w} \xrightarrow{f} B^{2,w} \xrightarrow{f} \cdots \xrightarrow{f} B^{R,w}$, 则定义该轨迹 $T^{0,w} \xrightarrow{f} T^{1,w} \xrightarrow{f} T^{2,w} \xrightarrow{f} \cdots \xrightarrow{f} T^{R,w}$ 为密码算法的 R 轮的字传播轨迹。

3.3 字传播轨迹的传播性质

为了评估分组密码算法抵抗积分攻击的能力，需要构建出关于每个字的传播轨迹。然而，手动推导字传播轨迹需要耗费较长的时间，并且容易出现错误。因此，需要研究字传播轨迹的传播性质。对于基于字运算的分组密码算法来说，其基本运算包括 S 盒运算、置换运算和异或运算，下面将分别讨论这些基本运算的字传播轨迹。

性质 4. 置换运算的字传播轨迹。由于置换运算只改变了字的位置，不会影响代数表达式和代数结构。如果轮函数内存在置换运算 $x_k^r \to x_{k_1}^{r+1}$，则根据字传播轨迹的定义，能够得到如下的关系：

$$A_{k_1}^{r+1,w} = A_k^{r,w},$$

$$\boldsymbol{B}_{k_1}^{r+1,w} = \boldsymbol{B}_k^{r,w},$$

其中 $0 \leqslant w \leqslant m-1$。

性质 5. S 盒运算的字传播轨迹。S 盒运算不影响明文字出现的次数，只对明文字到密文字的运算需要经过的 S 盒个数有影响。如果轮函数内存在 S 盒运算 $x_k^{r+1} = S(x_k^r)$，根据字传播轨迹的定义，能够得到如下的关系：

$$A_k^{r+1,w} = A_k^{r,w},$$

$$B_{k,i}^{r+1,w} = B_{k,i}^{r,w} + 1,\ 0 \leqslant i < |\boldsymbol{B}_k^{r,w}|。$$

性质 6. 异或运算的字传播轨迹。在大部分的分组密码算法中，异或运算是轮函数中的主要线性操作。如果在轮函数中存在异或运算，即

$$x_{k_1}^r \oplus x_{k_2}^r \oplus x_{k_3}^r \oplus \cdots \oplus x_{k_l}^r = x_k^{r+1}。$$

根据本文字传播轨迹的定义，能够得到如下关系：

$$A_{k_1}^{r,w} + A_{k_2}^{r,w} + A_{k_3}^{r,w} + \cdots + A_{k_l}^{r,w} = A_k^{r+1,w},$$

$$\boldsymbol{B}_{k_1}^{r,w} \| \boldsymbol{B}_{k_2}^{r,w} \| \boldsymbol{B}_{k_3}^{r,w} \| \cdots \| \boldsymbol{B}_{k_l}^{r,w} = \boldsymbol{B}_k^{r+1,w}。$$

通过将密码算法的轮函数拆分成 S 盒运算、置换运算和异或运算，利用性质 4, 5, 6 构建出每个运算的传播轨迹，将这些轨迹按照算法结构组合能够得到轮函数的字传播轨迹。最后，将轮函数进行多次迭代就能够得到密码算法的 R 轮的字传播轨迹，根据明文到密文的字传播轨迹特点，密码分析者可以判断算法抵抗积分攻击的能力，并能够确定积分区分器的具体形式。

3.4 积分区分器的新自动化搜索算法

首先，我们给出字传播轨迹的初始输入如何定义。为了得到明文中第 w 个字 x_w^0 的字传播轨迹，由于 x_w^0 在明文的第 w 个字中仅出现 1 次，并且 x_w^0 在此之前没有经过 S 盒运算，则初始的字传播轨迹 $T^{0,w}$ 如下：

$$\begin{cases} A_w^{0,w} = 1, \\ A_i^{0,w} = 0,\ i \neq w,\ 0 \leqslant i \leqslant m-1。 \end{cases}$$

$$\begin{cases} \boldsymbol{B}_w^{0,w} = (0), \\ \boldsymbol{B}_i^{0,w} = (),\ i \neq w,\ 0 \leqslant i \leqslant m-1。 \end{cases}$$

随后，根据被分析算法的结构，结合性质 4, 5, 6, 计算密码算法第 w 个字的 R 轮字传播轨迹，即 $T^{0,w} \xrightarrow{f} T^{1,w} \xrightarrow{f} T^{2,w} \xrightarrow{f} \cdots \xrightarrow{f} T^{R,w}$。

最后，根据得到的 $T^{R,w}$，确定积分区分器的具体形式，主要讨论下面 3 种情况：

1）如果 $\min(\boldsymbol{A}^{R,w}) \leqslant 1$，根据置换函数的性质，说明存在关于明文的第 w 个字的积分区分器。

2）如果 $\min(\boldsymbol{A}^{R,w}) = 2$，说明明文的第 w 个字经过 R 轮的加密后，在密文字的代数表达式中至少出现 2 次。此时，需要借助 $B^{R,w}$ 的值判断是否存在积分区分器。

假设 $A_i^{R,w} = 2$, $0 \leqslant i \leqslant m-1$，也就是明文的第 w 个字经过 R 轮的加密后，在密文的第 i 个字的代数表达式中出现 2 次。下面根据列表 $\boldsymbol{B}_i^{R,w}$ 的值，讨论此时是否存在 R 轮的积分区分器，共存在下面 5 种情况：

① $\boldsymbol{B}_i^{R,w} = (0,0)$。这种情况表明 x_i^R 的代数表达式中不存在关于 x_w^0 的 S 盒运算，即 $x_i^R = x_w^0 \oplus x_w^0 \oplus C = C$, C 是与变量 x_w^0 无关的常数。此时，当 x_w^0 遍历 F_2^n, x_i^R 是平衡的，存在 R 轮的积分区分器。

② $\boldsymbol{B}_i^{R,w} = (0,1)$ 或 $\boldsymbol{B}_i^{R,w} = (1,0)$。这种情况表明 x_i^R 的代数表达式中仅出现 1 次关于 x_w^0 的 S 盒运算，即 $x_i^R = x_w^0 \oplus S(x_w^0 \oplus C_1) \oplus C_2$。此时，由于存在如下性质：

$$\bigoplus_{x_w^0 \text{遍历} F_2^n} x_w^0 = 0,$$

$$\bigoplus_{x_w^0 \text{遍历} F_2^n} S(x_w^0 \oplus C_1) = 0。$$

由此可得 $\bigoplus_{x_w^0 \text{遍历} F_2^n} x_i^R = 0$, 此时 x_i^R 是平衡的，存在 R 轮的积分区分器。

③ $\boldsymbol{B}_i^{R,w} = (1,1)$。这种情况表明 x_i^R 的代数表达式中出现了 2 次关于 x_w^0 的 S 盒运算，并且每个出现在 x_i^R 的 x_w^0 的每个分量仅经过 1 次 S 盒运算，即 $x_i^R = S(x_w^0 \oplus C_1) \oplus S(x_w^0 \oplus C_2) \oplus C_3$，可得

$$\bigoplus_{x_w^0 \text{遍历} F_2^n} x_i^R = \bigoplus_{x_w^0 \text{遍历} F_2^n} (S(x_w^0 \oplus C_1) \oplus S(x_w^0 \oplus C_2) \oplus C_3) =$$

$$0 \oplus 0 \oplus 0 = 0。$$

因此，此时 x_i^R 是平衡的，存在 R 轮积分区分器。

④ $\boldsymbol{B}_i^{R,w} = (2,1)$ 或 $\boldsymbol{B}_i^{R,w} = (1,2)$。根据 $\boldsymbol{B}_i^{R,w}$ 的定义，这

种情况表明x_i^R的代数表达式可能为:

当 $x_i^R=S(x_w^0\oplus S(x_w^0\oplus C_1)\oplus C_2)\oplus C_3$ 或 $x_i^R=S(x_w^0\oplus S(x_w^0)\oplus C_2)\oplus C_3$时,此时无法利用置换函数的性质2和性质3判断$x_i^R$是否平衡;

当$x_i^R=S(x_w^0\oplus C_1)\oplus S(S(x_w^0)\oplus C_2)\oplus C_3$时,

$$\bigoplus_{x_w^0\text{遍历}F_2^n}x_i^R=\bigoplus_{x_w^0\text{遍历}F_2^n}(S(x_w^0\oplus C_1)\oplus S(S(x_w^0)\oplus C_2)\oplus C_3)=0\oplus 0=0,$$

此时x_i^R是平衡的,存在R轮的积分区分器。

因此,对于对于④,无法直接通过字传播轨迹确定是否存在R轮的积分区分器。

⑤$\boldsymbol{B}_i^{R,w}=(2,2)$。这种情况表明$x_w^0$在$x_i^R$的代数表达式中出现了2次,且$x_w^0$的每个分量在$x_i^R$的代数表达式中都经过2次S盒运算。此时,仅存在2种满足的情况:$x_i^R=S(S(x_w^0\oplus C_1)\oplus C_2)\oplus S(S(x_w^0\oplus C_3)\oplus C_4)\oplus C_5$或$x_i^R=S(S(x_w^0\oplus C_1)\oplus S(x_w^0\oplus C_2)\oplus C_3)\oplus C_4$。

当$x_i^R=S(S(x_w^0\oplus C_1)\oplus C_2)\oplus S(S(x_w^0\oplus C_3)\oplus C_4)\oplus C_5$时,由性质2可得

$$\bigoplus_{x_w^0\text{遍历}F_2^n}x_i^R=\bigoplus_{x_w^0\text{遍历}F_2^n}(S(S(x_w^0\oplus C_1)\oplus C_2\oplus S(S(x_w^0\oplus C_3)\oplus C_4)\oplus C_5))=0\oplus 0\oplus 0=0。$$

当$x_i^R=S(S(x_w^0\oplus C_1)\oplus S(x_w^0\oplus C_2)\oplus C_3)\oplus C_4$时,此时由性质3可以证明$x_i^R$关于$x_w^0$是仍然是平衡的。因此,对于第5种情况,此时$x_i^R$是平衡的,存在$R$轮积分区分器。

3)如果$\min(\boldsymbol{A}^{R,w})=h$,且$h>2$。假设$A_i^{R,w}=h$,$0\leqslant i<m$,如果$\max(\boldsymbol{B}_i^{R,w})\leqslant 1$,说明$x_w^0$的每一个分量在$x_i^R$的代数表达式中只涉及1个S盒运算,此时$x_i^R$的代数表达式只能为如下的情况:

$$x_i^R=S(x_w^0\oplus C_1)\oplus S(x_w^0\oplus C_2)\oplus\cdots\oplus S(x_w^0\oplus C_h)\oplus C_{h+1},$$

根据性质2可得

$$\bigoplus_{x_w^0\text{遍历}F_2^n}x_i^R=\bigoplus_{x_w^0\text{遍历}F_2^n}(S(x_w^0\oplus C_1)\oplus S(x_w^0\oplus C_2)\oplus\cdots\oplus S(x_w^0\oplus C_h)\oplus C_{h+1})=0\oplus 0\oplus 0\oplus\cdots\oplus 0=0,$$

由此,x_i^R仍然是平衡字,此时存在R轮的积分区分器。

对上述讨论进行总结,当$\min(\boldsymbol{A}^{R,w})\leqslant 1$,此时一定存在关于明文第$w$个字$x_w^0$的$R$轮积分区分器;当$\min(\boldsymbol{A}^{R,w})=2$时,存在$A_i^{R,w}=2$且$\max(\boldsymbol{B}_i^{R,w})\leqslant 1$或者$\boldsymbol{B}_i^{R,w}=(2,2)$,此时存在关于明文第$w$个字$x_w^0$的$R$轮积分区分器;当$\min(\boldsymbol{A}^{R,w})\geqslant 2$且存在$\max(\boldsymbol{B}_i^{R,w})\leqslant 1$,此时存在关于明文第$w$个字$x_w^0$的$R$轮积分区分器。

为了搜索最长轮数的积分区分器,本文给出算法2和算法3。

算法2. 搜索加密方向最长轮数的积分区分器。

输入:分组密码算法的轮函数f,轮函数的输入为m个n比特的字,加密总轮数R,活跃字的下标w;

输出:第w个字活跃,加密方向最长的积分区分器轮数r_w^e,平衡字的下标集合ϕ^w。

① 初始化$r_w^e=0$,$\phi^w=\varnothing$,定义1维$\boldsymbol{A}^{0,w}$和2维列表$B^{0,w}$,$A_w^{0,w}=1$,$\boldsymbol{B}_w^{0,w}=(0)$,$A_i^{0,w}=0$,$\boldsymbol{B}_i^{0,w}=()$,$i\neq w$,$0\leqslant i\leqslant m-1$;
② for $0\leqslant i\leqslant R$ do
③ 根据算法结构计算r轮的字传播轨迹:$T^{0,w}\xrightarrow{f}T^{1,w}\xrightarrow{f}T^{2,w}\xrightarrow{f}\cdots\xrightarrow{f}T^{r,w}$;
④ if $\min(\boldsymbol{A}^{r,w})\leqslant 1$then
⑤ $r_w^e=r$;
⑥ else if $\min(\boldsymbol{A}^{r,w})==2$ and(当$A_i^{r,w}=2$时,$\max(\boldsymbol{B}_i^{r,w})\leqslant 1$ or $B_i^{r,w}==[2,2]$)
⑦ $r_w^e=r$;
⑧ else if $\min(\boldsymbol{A}^{r,w})\geqslant 2$ and $\max(\boldsymbol{B}_i^{r,w})\leqslant 1$
⑨ $r_w^e=r$;
⑩ else
⑪ break;
⑫ end if
⑬ end for
⑭ if $\min(\boldsymbol{A}^{r_w^e,w})\leqslant 1$ then
⑮ for $0\leqslant i\leqslant m-1$ do
⑯ if $A_i^{r_w^e,w}\leqslant 1$ then
⑰ $\phi^w=\phi^w\cup\{i\}$;
⑱ end if
⑲ end for
⑳ else if $\min(\boldsymbol{A}^{r_w^e,w})==2$ then
㉑ for $0\leqslant i\leqslant m-1$ do
㉒ if $A_i^{r_w^e,w}==2$ and($\max(\boldsymbol{B}_i^{r_w^e,w})\leqslant 1$ or $\boldsymbol{B}_i^{r_w^e,w}=(2,2)$) then
㉓ $\phi^w=\phi^w\cup\{i\}$;
㉔ end if
㉕ end for
㉖ else $\min(\boldsymbol{A}^{r_w^e,w})>2$
㉗ for $0\leqslant i\leqslant m-1$ do
㉘ if $\max(\boldsymbol{B}_i^{r_w^e,w})\leqslant 1$ then
㉙ $\phi^w=\phi^w\cup\{i\}$;
㉚ end if
㉛ end for
㉜ end if
㉝ return r_w^e, ϕ^w。

在算法2中,行①为初始化过程,主要是根据字传播轨迹的定义对$\boldsymbol{A}^{0,w}$和$B^{0,w}$进行初始化;行②~⑬表示利用字传播轨迹,确定加密方向上关于第w个明文

字的最长积分区分器轮数r_w^{e}；行⑭~㉝的作用是得到加密r_w^{e}轮后关于第 w 个明文字的平衡密文字的位置。

为了进一步扩展积分区分器的轮数，需要利用文献 [15, 17] 中的方法推导密文字在解密方向上的字传播轨迹。根据算法解密方向的轮函数结构，判断每一个密文字在解密方向是否达到全扩散。如果解密 r+1 轮后，密文的第 w 个字第 1 次达到全扩散，并且解密 r 轮后，第 w 个字没有达到全扩散，即存在部分状态字的表达式中不出现y_w^0，则将此时解密的轮数定义为r_w^{d}。根据文献 [15, 17] 给出的证明，此时密码算法一定存在$r_w^{\mathrm{e}}+r_w^{\mathrm{d}}$轮的积分区器。

算法 3. 向解密方向扩展积分区分器。

输入：分组密码算法的解密轮函数f^{-1}，轮函的输入为 m 个 n 比特的字，解密总轮数 R，活跃密文字的下标 w；

输出：第 w 个密文字活跃时，解密方向第 w 个字未达到全扩散的最大轮数r_w^{d}，未扩散到的状态字的下标集合φ^w。

① 初始化$r_w^{\mathrm{d}}=0$, $\varphi^w=\varnothing$;

② for $1\leqslant r<R$ do

③ $\boldsymbol{C}^{0,w}\xrightarrow{f^{-1}}\boldsymbol{C}^{1,w}\xrightarrow{f^{-1}}\boldsymbol{C}^{2,w}\xrightarrow{f^{-1}}\cdots\xrightarrow{f^{-1}}\boldsymbol{C}^{r,w}$;

④ if $\min(\boldsymbol{C}^{r,w})==0$ then

⑤ $r_w^{\mathrm{d}}=r$;

⑥ else

⑦ break;

⑧ end if

⑨ end for

⑩ for $0\leqslant i<m-1$ do

⑪ if $C_i^{r_w^{\mathrm{d}},w}==0$ then

⑫ $\varphi^w=\varphi^w\cup\{i\}$;

⑬ end if

⑭ end for

⑮ return r_w^{d}, φ^w。

对于迭代型的分组密码算法，我们给出如下搜索积分区分器的步骤：

第 1 步. 遍历所有的$0\leqslant w\leqslant m-1$，利用算法 2 计算对应的$r_w^{\mathrm{e}}$和$\phi^w$；

第 2 步. 遍历所有的$0\leqslant w\leqslant m-1$，利用算法 3 计算对应的$r_w^{\mathrm{d}}$和$\varphi^w$；

第 3 步. 计算$r^{\mathrm{ed}}=\max(r_0^{\mathrm{e}}+r_0^{\mathrm{d}},r_1^{\mathrm{e}}+r_1^{\mathrm{d}},\cdots,r_{m-1}^{\mathrm{e}}+r_{m-1}^{\mathrm{d}})$以及最大值$r^{\mathrm{ed}}$在列表$(r_0^{\mathrm{e}}+r_0^{\mathrm{d}},r_1^{\mathrm{e}}+r_1^{\mathrm{d}},r_2^{\mathrm{e}}+r_2^{\mathrm{d}},\cdots,r_{m-1}^{\mathrm{e}}+r_{m-1}^{\mathrm{d}})$中对应的下标集合$\beta$；

第 4 步. 根据ϕ^i和φ^i确定积分区分器的具体形式，$i\in\beta$。

4 对 LLLWBC 算法的积分攻击

4.1 LLLWBC 算法字传播轨迹的构建

为了分析 LLLWBC 算法抵抗积分攻击的能力，需要构建出其对应的字传播轨迹。由于 LLLWBC 的基本操作单元为 4b，本节主要讨论 4b 的明文字在 LLLWBC 中字传播轨迹的构建方法。

首先，在加密方向上，64b 的明文输入为$\boldsymbol{X}^0=(x_0^0,x_1^0,\cdots,x_{15}^0)$，当第 w 个字为活跃，即x_w^0活跃，其余字固定为常数，根据新型字传播轨迹的定义，可以得到如下的初始化向量：

$$\boldsymbol{A}^{0,w}=(A_0^{0,w},A_1^{0,w},A_2^{0,w},\cdots,A_w^{0,w},\cdots,A_{15}^{0,w})=(0,0,0,\cdots,0,1,0,\cdots,0),\tag{4}$$

$$B^{0,w}=\{\boldsymbol{B}_0^{0,w},\boldsymbol{B}_1^{0,w},\boldsymbol{B}_2^{0,w},\cdots,\boldsymbol{B}_w^{0,w},\cdots,\boldsymbol{B}_{15}^{0,w}\}=\{(),(),(),\cdots,(),(0),(),\cdots,()\},\tag{5}$$

其中，() 表示该位置对应的内部状态不存在关于x_w^0可行的传播路径，因此相应的内部状态的表达式中x_w^0经过 S 盒次数构成的向量为空。

随后，根据 LLLWBC 算法的轮函数结构，为了更好地构建其字传播轨迹，我们首先给出 LLLWBC 算法中第 i 个 T 函数的定义，其具体结构如图 2 所示。其表达式如下：

$$y_{4\times i}^{r-1}=x_{4\times i}^{r-1},\tag{6}$$

$$y_{4\times i+1}^{r-1}=x_{4\times i+1}^{r-1},\tag{7}$$

$$y_{4\times i+2}^{r-1}=x_{4\times i+2}^{r-1}\oplus S(0\mathrm{x}2\cdot S(x_{4\times i}^{r-1})\oplus 0\mathrm{x}3\cdot S(x_{4\times i+1}^{r-1})),\tag{8}$$

$$y_{4\times i+3}^{r-1}=x_{4\times i+3}^{r-1}\oplus S(0\mathrm{x}1\cdot S(x_{4\times i}^{r-1})\oplus 0\mathrm{x}1\cdot S(x_{4\times i+1}^{r-1}))。\tag{9}$$

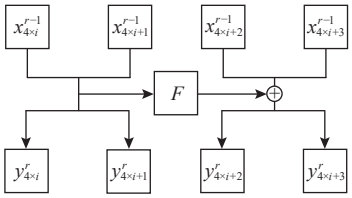


Fig. 2 T-function of LLLWBC algorithm

图 2 LLLWBC 算法的 T 函数

根据式(4)~(9)，本文给出了算法 4，用于描述 LLLWBC 算法中第 r 轮加密第 i 个 T 函数的字传播轨迹。

算法 4. LLLWBC 算法中第 r 轮加密第 i 个 T 函数字传播轨迹 *T_Word_Trail*(加密轮数 r, $A_{4\times i}^{r-1,w}$,$A_{4\times i+1}^{r-1,w}$, $A_{4\times i+2}^{r-1,w}$,$A_{4\times i+3}^{r-1,w}$,$\boldsymbol{B}_{4\times i}^{r-1,w}$,$\boldsymbol{B}_{4\times i+1}^{r-1,w}$,$\boldsymbol{B}_{4\times i+2}^{r-1,w}$,$\boldsymbol{B}_{4\times i+3}^{r-1,w}$)。

输入：轮数 r, $A_{4\times i}^{r-1,w}$,$A_{4\times i+1}^{r-1,w}$,$A_{4\times i+2}^{r-1,w}$,$A_{4\times i+3}^{r-1,w}$,$\boldsymbol{B}_{4\times i}^{r-1,w}$,$\boldsymbol{B}_{4\times i+1}^{r-1,w}$, $\boldsymbol{B}_{4\times i+2}^{r-1,w}$,$\boldsymbol{B}_{4\times i+3}^{r-1,w}$;

输出: $A^{r,w}_{4\times i}, A^{r,w}_{4\times i+1}, A^{r,w}_{4\times i+2}, A^{r,w}_{4\times i+3}, \boldsymbol{B}^{r,w}_{4\times i}, \boldsymbol{B}^{r,w}_{4\times i+1}, \boldsymbol{B}^{r,w}_{4\times i+2}, \boldsymbol{B}^{r,w}_{4\times i+3}$。

① $A^{r,w}_{4\times i} = A^{r-1,w}_{4\times i}$;

② $A^{r,w}_{4\times i+1} = A^{r-1,w}_{4\times i+1}$;

③ $A^{r,w}_{4\times i+2} = A^{r-1,w}_{4\times i} + A^{r-1,w}_{4\times i+1} + A^{r-1,w}_{4\times i+2}$;

④ $A^{r,w}_{4\times i+3} = A^{r-1,w}_{4\times i} + A^{r-1,w}_{4\times i+1} + A^{r-1,w}_{4\times i+3}$;

⑤ $\boldsymbol{B}^{r,w}_{4\times i} = \boldsymbol{B}^{r-1,w}_{4\times i}$;

⑥ $\boldsymbol{B}^{r,w}_{4\times i+1} = \boldsymbol{B}^{r-1,w}_{4\times i+1}$;

⑦ for $0 \leqslant j < |\boldsymbol{B}^{r-1,w}_{4\times i}|$ do

⑧ $\quad \boldsymbol{B}^{r-1,w}_{4\times i,j} = \boldsymbol{B}^{r-1,w}_{4\times i,j} + 1$;

⑨ end for

⑩ for $0 \leqslant j < |\boldsymbol{B}^{r-1,w}_{4\times i+1}|$ do

⑪ $\quad \boldsymbol{B}^{r-1,w}_{4\times i+1,j} = \boldsymbol{B}^{r-1,w}_{4\times i+1,j} + 1$;

⑫ end for

⑬ $\boldsymbol{B}^{*} = \boldsymbol{B}^{r-1,w}_{4\times i} \| \boldsymbol{B}^{r-1,w}_{4\times i+1}$;

⑭ for $0 \leqslant j < |B^{*}|$ do

⑮ $\quad \boldsymbol{B}^{*}_{j} = \boldsymbol{B}^{*}_{j} + 1$;

⑯ end for

⑰ $\boldsymbol{B}^{r,w}_{4\times i+2} = \boldsymbol{B}^{*} \| \boldsymbol{B}^{r-1,w}_{4\times i+2}$;

⑱ $\boldsymbol{B}^{r,w}_{4\times i+3} = \boldsymbol{B}^{*} \| \boldsymbol{B}^{r-1,w}_{4\times i+3}$;

⑲ return $A^{r,w}_{4\times i}, A^{r,w}_{4\times i+1}, A^{r,w}_{4\times i+2}, A^{r,w}_{4\times i+3}, \boldsymbol{B}^{r,w}_{4\times i}, \boldsymbol{B}^{r,w}_{4\times i+1}, \boldsymbol{B}^{r,w}_{4\times i+2}, \boldsymbol{B}^{r-1,w}_{4\times i+2}, \boldsymbol{B}^{r-1,w}_{4\times i+3}$。

在算法 4 中, 行①~④用于描述明文的第 w 个字在第 r 轮的第 i 个 T 函数字传播轨迹的变化情况, 即明文中第 w 个字在 T 函数的代数表达式中出现的次数的变化。行⑤⑥主要用于描述式(6)(7)的字传播轨迹。行⑦~⑨主要用于表示 $0\text{x}2\cdot S(x^{r-1}_{4\times i})$ 的字传播轨迹, 根据字传播轨迹的定义, 由于 $x^{r-1}_{4\times i}$ 经过 1 次 S 盒运算, 此时明文的第 w 个字 x^{0}_{w} 在 $S(x^{r-1}_{4\times i})$ 的表达式中的分量经过 S 盒的个数比 $x^{r-1}_{4\times i}$ 增加 1 个, 然后, 执行 $0\text{x}2\cdot S(x^{r-1}_{4\times i})$, 这个乘法操作对字传播轨迹无影响。算法的行⑩~⑫的原理与行⑦~⑨类似。算法的行⑬~⑯用于描述 $S(0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1}))$ 的字传播轨迹, x^{0}_{w} 在表达式 $0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1})$ 中分量的个数为 $0\text{x}2\cdot S(x^{r-1}_{4\times i})$ 中 x^{0}_{w} 分量的个数与 $0\text{x}3\cdot S(x^{r-1}_{4\times i+1})$ 中 x^{0}_{w} 分量个数之和, 每个分量经过 S 盒的个数没有变化, 所以使用 $\boldsymbol{B}^{*} = \boldsymbol{B}^{r-1,w}_{4\times i} \| \boldsymbol{B}^{r-1,w}_{4\times i+1}$ 表示此时 x^{0}_{w} 的字传播轨迹。然后, $0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1})$ 的输出经过 S 盒操作, x^{0}_{w} 每个分量经过 S 盒的个数比经过 $0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1})$ 的个数增加 1。算法的行⑰用于描述 $y^{r-1}_{4\times i+2} = x^{r-1}_{4\times i+2}\oplus S(0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1}))$ 的字传播轨迹, 此时 $y^{r-1}_{4\times i+2}$ 中 x^{0}_{w} 分量的个数为 $x^{r-1}_{4\times i+2}$ 中分量的个数与 $S(0\text{x}2\cdot S(x^{r-1}_{4\times i})\oplus 0\text{x}3\cdot S(x^{r-1}_{4\times i+1}))$ 分量个数的和, 且每个分量经过 S 盒的个数不变, 所以使用 $\boldsymbol{B}^{r,w}_{4\times i+2} = \boldsymbol{B}^{r-1,w}_{4\times i+2} \| \boldsymbol{B}^{*}$ 描述其字传播轨迹, 与算法的行⑱原理类似。最后, 算法返回 $A^{r,w}_{4\times i}, A^{r,w}_{4\times i+1}, A^{r,w}_{4\times i+2}, A^{r,w}_{4\times i+3}$ 和 $\boldsymbol{B}^{r,w}_{4\times i}, \boldsymbol{B}^{r,w}_{4\times i+1}, \boldsymbol{B}^{r,w}_{4\times i+2}, \boldsymbol{B}^{r,w}_{4\times i+3}$。

基于算法 4, 结合 LLLWBC 算法的基本结构, 本文设计了算法 5 用于描述 LLLWBC 算法加密方向轮函数的字传播轨迹。

算法 5. LLLWBC 算法加密方向轮函数的字传播轨迹。

输入: w, 加密轮数 $r(r>0)$, $\boldsymbol{A}^{r-1,w}$, $B^{r-1,w}$;

输出: 第 r 轮的 $\boldsymbol{A}^{r,w}$, $B^{r,w}$。

① 定义 $\boldsymbol{A}^{*} = (A^{*}_{0}, A^{*}_{1}, A^{*}_{2}, \cdots, A^{*}_{15})$;

② 定义 $B^{*} = \{\boldsymbol{B}^{*}_{0}, \boldsymbol{B}^{*}_{1}, \boldsymbol{B}^{*}_{2}, \cdots, \boldsymbol{B}^{*}_{15}\}$;

③ for $0 \leqslant i < 4$ do

④ $\quad A^{*}_{4\times i}, A^{*}_{4\times i+1}, A^{*}_{4\times i+2}, A^{*}_{4\times i+3}, \boldsymbol{B}^{*}_{4\times i}, \boldsymbol{B}^{*}_{4\times i+1}, \boldsymbol{B}^{*}_{4\times i+2}, \boldsymbol{B}^{*}_{4\times i+3} = T_Word_Trail(A^{r-1,w}_{4\times i}, A^{r-1,w}_{4\times i+1}, A^{r-1,w}_{4\times i+2}, A^{r-1,w}_{4\times i+3}, \boldsymbol{B}^{r-1,w}_{4\times i}, \boldsymbol{B}^{r-1,w}_{4\times i+1}, \boldsymbol{B}^{r-1,w}_{4\times i+2}, \boldsymbol{B}^{r-1,w}_{4\times i+3})$

⑤ end for

⑥ for $0 \leqslant i < 16$ do

⑦ $\quad A^{r,w}_{P[i]} = A^{*}_{i}$, $\boldsymbol{B}^{r,w}_{P[i]} = \boldsymbol{B}^{*}_{i}$;

⑧ end for

⑨ return $T^{r,w} = \{\boldsymbol{A}^{r,w}, B^{r,w}\}$。

在解密方向, 由于 LLLWBC 算法的结构是广义 Feistel 结构, 设计了算法 6 用于描述 LLLWBC 算法在解密方向轮函数的字传播轨迹。

算法 6. LLLWBC 算法解密方向轮函数的字传播轨迹。

输入: w, 加密轮数 $r(r>0)$, $\boldsymbol{C}^{r-1,w}$; /*$\boldsymbol{C}^{r-1,w}$ 表示解密 $r-1$ 轮后, 密文的第 w 个字在内部状态 Y^{r-1} 的代数表达式中出现的次数*/

输出: $\boldsymbol{C}^{r,w}$。

① 定义 $\boldsymbol{C}^{*} = (C^{*}_{0}, C^{*}_{1}, C^{*}_{2}, \cdots, C^{*}_{15})$;

② for $0 \leqslant i < 16$ do

③ $\quad C^{*}_{P^{-1}[i]} = C^{r-1}_{i}$;

④ end for

⑤ for $0 \leqslant i < 4$ do

⑥ $\quad C^{r,w}_{4\times i} = C^{*}_{4\times i}$;

⑦ $\quad C^{r,w}_{4\times i+1} = C^{*}_{4\times i+1}$;

⑧ $\quad C^{r,w}_{4\times i+2} = C^{*}_{4\times i} + C^{*}_{4\times i+1} + C^{*}_{4\times i+2}$;

⑨ $\quad C^{r,w}_{4\times i+3} = C^{*}_{4\times i} + C^{*}_{4\times i+1} + C^{*}_{4\times i+3}$;

⑩ end for

⑪ return $\boldsymbol{C}^{r,w}$。

4.2 LLLWBC 算法的积分区分器

将算法 5 和算法 6 迭代 R 次, 即可得到加密和解密方向 R 轮的 LLLWBC 算法的字传播轨迹, 并分别

将得到的轨迹 $T^{0,w}\xrightarrow{f}T^{1,w}\xrightarrow{f}T^{2,w}\xrightarrow{f}\cdots\xrightarrow{f}T^{R,w}$ 和 $\boldsymbol{C}^{0,w}\xrightarrow{f^{-1}}\boldsymbol{C}^{1,w}\xrightarrow{f^{-1}}\boldsymbol{C}^{2,w}\xrightarrow{f^{-1}}\cdots\xrightarrow{f^{-1}}\boldsymbol{C}^{R,w}$ 代入到算法 2 和算法 3，得到 r_w^{e}，ϕ^w，r_w^{d}，φ^w 并遍历 $0\leqslant w\leqslant 15$，计算 $r^{\mathrm{ed}}=\max(r_0^{\mathrm{e}}+r_0^{\mathrm{d}},r_1^{\mathrm{e}}+r_1^{\mathrm{d}},\cdots,r_{m-1}^{\mathrm{e}}+r_{m-1}^{\mathrm{d}})$ 确定积分区分器的最长轮数以及积分区分器的具体形式。

对于 LLLWBC 算法，以 $w=2$ 为例，具体的积分区分器搜索步骤如下：

1）在 LLLWBC 算法的加密方向，利用算法 5 计算 LLLWBC 算法每一轮的字传播轨迹为

$\boldsymbol{A}^{0,2}=(0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$，

$B^{0,2}=\{(),(),(0),(),(),(),(),(),(),(),(),(),(),(),()\}$，

$\boldsymbol{A}^{1,2}=(0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0)$，

$B^{1,2}=\{(),(),(),(),(),(),(),(),(),(),(),(),(0),(),(),()\}$，

$\boldsymbol{A}^{2,2}=(0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0)$，

$B^{2,2}=\{(),(),(),(0),(),(),(),(),(),(2),(),(),(),(2),(),()\}$，

$\boldsymbol{A}^{3,2}=(0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,0)$，

$B^{3,2}=\{(),(4),(),(),(4),(),(2),(),(0),(4),(),(2),(),(4),(),()\}$，

$\boldsymbol{A}^{4,2}=(2,3,0,0,2,1,1,1,1,1,1,1,1,1,0,1)$，

$B^{4,2}=\{(6,2),(2,6,2),(),(),(2,6),(6),(4),(4),(6),$
$(6),(4),(4),(6),(6),(),(0)\}$，

$\boldsymbol{A}^{5,2}=(4,3,2,1,3,4,1,3,5,3,2,1,5,2,1,1)$，

$B^{5,2}=\{(4,8,8,4),(8,8,4),(6,2),(6),(8,8,4),(4,8,8,4),(6),$
$(2,6,2),(8,4,4,8,4),(8,8,0),(2,6),(6),(8,4,4,8,4),$
$(8,8),(6),(6)\}$。

2）在 LLLWBC 算法的解密方向，利用算法 6 计算 LLLWBC 算法解密方向每一轮的字传播轨迹为

$\boldsymbol{C}^{0,2}=(0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$，

$\boldsymbol{C}^{1,2}=(0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0)$，

$\boldsymbol{C}^{2,2}=(0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0)$，

$\boldsymbol{C}^{3,2}=(0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,0)$，

$\boldsymbol{C}^{4,2}=(2,3,0,0,2,1,1,1,1,1,1,1,1,1,0,1)$。

3）将由 1）得到的字传播轨迹代入算法 2，可得 $r_2^{\mathrm{e}}=5$，$\phi^w=\{3,6,11,14,15\}$；

4）将由 2）得到的字传播轨迹代入算法 3，可得 $r_2^{\mathrm{d}}=4$，$\varphi^2=\{2,3,14\}$；

5）根据 LLLWBC 算法结构，可得

$$x_2^5=x_0^4=F(x_4^3,x_5^3)\oplus x_6^3=0\mathrm{x}2\cdot x_4^3\oplus 0\mathrm{x}3\cdot x_5^3\oplus x_6^3。\tag{10}$$

由于在加密方向，$\boldsymbol{A}^{3,2}=(0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,0)$，可得 x_2^0 在 x_4^3，x_5^3，x_6^3 中分别出现 1 次、0 次和 1 次，根据置换函数的性质可得 x_6^3 和 x_4^3 是关于 x_2^0 的置换函数，且 x_5^3 的值与 x_2^0 的值无关，可得如下关系：

$$\bigoplus_{\substack{x_2^0\text{遍历}F_2^4\\ x_0^0\|x_1^0\|x_3^0\|,\cdots,\|x_{15}^0\in F_2^{60}}}x_4^3=0^4,\tag{11}$$

$$\bigoplus_{\substack{x_2^0\text{遍历}F_2^4\\ x_0^0\|x_1^0\|x_3^0\|,\cdots,\|x_{15}^0\in F_2^{60}}}x_5^3=0^4,\tag{12}$$

$$\bigoplus_{\substack{x_2^0\text{遍历}F_2^4\\ x_0^0\|x_1^0\|x_3^0\|,\cdots,\|x_{15}^0\in F_2^{60}}}x_6^3=0^4。\tag{13}$$

由此可得 x_2^5 关于 x_2^0 也是平衡的，$\phi^w=\{3,6,11,14,15\}\cup\{2\}=\{2,3,6,11,14,15\}$。

6）最后得到关于 x_2^0 的具体积分区分器形式为

$$\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{c}^4\boldsymbol{c}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{c}^4\boldsymbol{a}^4\xrightarrow{9\text{轮}}$$
$$\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{b}^4。\tag{14}$$

根据上述步骤，我们分别遍历 $0\leqslant w\leqslant 15$，得到最长的 9 轮积分区分器轮数，得到的 r_w^{d}，r_w^{e}，ϕ^w，φ^w 的数值以及对应的积分区分器的形式如表 4 和表 5 所示。

Table 4 Values of w, r_w^{d}, φ^w, r_w^{e} and ϕ^w

表 4 w，r_w^{d}，φ^w，r_w^{e}，ϕ^w 的值

w	r_w^{d}	φ^w	r_w^{e}	ϕ^w
2	4	{2，3，14}	5	{2，3，6，11，14，15}
3	4	{6，7，10}	5	{2，3，6，11，14，15}
6	4	{2，14，15}	5	{2，3，7，10，14，15}
7	4	{7，10，11}	5	{2，3，7，10，14，15}
10	4	{6，7，10}	5	{3，6，7，10，11，15}
11	4	{2，3，15}	5	{3，6，7，10，11，15}
14	4	{3，14，15}	5	{2，6，7，10，11，14}
15	4	{6，10，11}	5	{2，6，7，10，11，14}

Table 5 The 9-Round Integral Distinguisher of LLLWBC Algorithm

表 5 LLLWBC 算法的 9 轮积分区分器

w	输入形式	输出形式
2	$a^4a^4c^4c^4a^4a^4a^4a^4a^4a^4a^4a^4c^4a^4$	$u^4u^4b^4b^4u^4u^4b^4u^4u^4u^4b^4u^4u^4b^4b^4$
3	$a^4a^4a^4a^4a^4a^4c^4c^4a^4a^4c^4a^4a^4a^4a^4$	$u^4u^4b^4b^4u^4u^4b^4u^4u^4u^4b^4u^4u^4b^4b^4$
6	$a^4a^4c^4a^4a^4a^4a^4a^4a^4a^4a^4a^4c^4c^4$	$u^4u^4b^4b^4u^4u^4b^4u^4u^4b^4u^4u^4u^4b^4b^4$
7	$a^4a^4a^4a^4a^4a^4c^4a^4a^4c^4c^4a^4a^4a^4a^4$	$u^4u^4b^4b^4u^4u^4u^4b^4u^4u^4b^4u^4u^4u^4b^4b^4$
10	$a^4a^4a^4a^4a^4a^4c^4c^4a^4a^4a^4c^4a^4a^4a^4$	$u^4u^4u^4b^4u^4u^4b^4b^4u^4u^4b^4b^4u^4u^4u^4b^4$
11	$a^4a^4c^4c^4a^4a^4a^4a^4a^4a^4a^4a^4a^4a^4c^4$	$u^4u^4u^4b^4u^4u^4b^4b^4u^4u^4b^4b^4u^4u^4u^4b^4$
14	$a^4a^4a^4c^4a^4a^4a^4a^4a^4a^4a^4a^4c^4c^4$	$u^4u^4b^4u^4u^4u^4b^4b^4u^4u^4b^4b^4u^4u^4b^4u^4$
15	$a^4a^4a^4a^4a^4a^4c^4a^4a^4a^4c^4c^4a^4a^4a^4a^4$	$u^4u^4b^4u^4u^4u^4b^4b^4u^4u^4b^4b^4u^4u^4b^4u^4$

4.3 对 LLLWBC 算法的密钥恢复攻击

在对密码算法进行密钥恢复时，首先需要构建出 r 轮的积分区分器，假设区分器输出中平衡字下标构成的集合为 B。随后，构建与区分器相关的明文数据集 M，加密 $r+R$ 轮后得到内部状态 $\boldsymbol{X}^{r+R}$ 构成的集合 $\boldsymbol{C}^{r+R}$。然后，根据算法结构猜测与扩展的 R 轮相关的

密钥, 对$\boldsymbol{X}^{r+R}$中的部分字在$\boldsymbol{C}^{r+R}$上进行解密R轮, 能够计算得到$\boldsymbol{X}^{r}$中部分位置的值, 构成的数据集为$\boldsymbol{C}^{r}$。如果数据集$\boldsymbol{C}^{r}$中每个字都是平衡的, 则猜测的密钥可能是正确的, 否则, 猜测的密钥一定是错误的。假设$\boldsymbol{C}^{r}$中每个元素由m个n比特的字构成, 则上述过程执行1次可以恢复出$m\times n$比特的密钥信息。上述过程是积分攻击中最基本的密钥恢复方法, 然而, 需要猜测哪些位置的密钥、扩展的R轮中需要确定哪些内部状态的值以及利用积分区分器哪些平衡的位置, 这些因素对积分攻击的密钥恢复的复杂度以及最终攻击的轮数具有较大的影响。因此, 如何选取出最优的猜测路径是需要解决的问题。针对这个问题, 本节主要研究如何根据算法的结构搜索最优的密钥猜测路径。

以LLLWBC为例, 当轮数$1\leqslant r\leqslant 10$时, 第r轮解密方向的代数表达式为

$$\begin{cases} x_0^{r-1}=x_2^r,\ x_1^{r-1}=x_7^r,\\ x_2^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_7^r))\oplus k_{0,0}^{r-1}\oplus x_{12}^r,\\ x_3^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_7^r))\oplus k_{0,1}^{r-1}\oplus x_8^r,\\ x_4^{r-1}=x_{10}^r,\ x_5^{r-1}=x_{14}^r,\\ x_6^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{1,0}^{r-1}\oplus x_0^r,\\ x_7^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{1,1}^{r-1}\oplus x_5^r,\\ x_8^{r-1}=x_{15}^r,\ x_9^{r-1}=x_{11}^r,\\ x_{10}^{r-1}=S(2\cdot S(x_{15}^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{2,0}^{r-1}\oplus x_4^r,\\ x_{11}^{r-1}=S(2\cdot S(x_{15}^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{2,1}^{r-1}\oplus x_1^r,\\ x_{12}^{r-1}=x_3^r,\ x_{13}^{r-1}=x_6^r,\\ x_{14}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_6^r))\oplus k_{3,0}^{r-1}\oplus x_{13}^r,\\ x_{15}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_6^r))\oplus k_{3,1}^{r-1}\oplus x_9^r。 \end{cases} \tag{15}$$

当$11\leqslant r\leqslant 20$时, 第$r$轮解密方向的代数表达式为

$$\begin{cases} x_0^{r-1}=x_6^r,\ x_1^{r-1}=x_{11}^r,\\ x_2^{r-1}=S(2\cdot S(x_6^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{0,0}^{r-1}\oplus x_0^r,\\ x_3^{r-1}=S(2\cdot S(x_6^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{0,1}^{r-1}\oplus x_{12}^r,\\ x_4^{r-1}=x_{10}^r,\ x_5^{r-1}=x_7^r,\\ x_6^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_7^r))\oplus k_{1,0}^{r-1}\oplus x_{13}^r,\\ x_7^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_7^r))\oplus k_{1,1}^{r-1}\oplus x_1^r,\\ x_8^{r-1}=x_3^r,\ x_9^{r-1}=x_{15}^r,\\ x_{10}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_{15}^r))\oplus k_{2,0}^{r-1}\oplus x_4^r,\\ x_{11}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_{15}^r))\oplus k_{2,1}^{r-1}\oplus x_9^r,\\ x_{12}^{r-1}=x_2^r,\ x_{13}^{r-1}=x_{14}^r,\\ x_{14}^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{3,0}^{r-1}\oplus x_5^r,\\ x_{15}^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{3,1}^{r-1}\oplus x_8^r。 \end{cases} \tag{16}$$

在第10轮解密方向上, 存在如下的表达式:

$$x_2^9=x_{12}^{10}\oplus S(2\cdot S(x_2^{10})\oplus 3\cdot S(x_7^{10}))\oplus k_{0,0}^9。 \tag{17}$$

假设攻击者通过猜测密钥能够得到中间状态x_{12}^{10}, x_2^{10}, x_7^{10}的值, 此时再次猜测密钥$k_{0,0}^9$的值, 根据式(17), 能够猜测确定x_2^9的值。然而, 当积分区分器扩展的轮数较多, 直接通过手动推导上述的解密过程较为繁琐, 且容易出现错误。为此, 本节重点研究了基于MILP的密钥猜测路径搜索的问题。

主要以LLLWBC算法为例。首先, 定义模型变量$\boldsymbol{a}^{r-1}=(a_0^{r-1},a_1^{r-1},a_2^{r-1},\cdots,a_{14}^{r-1},a_{15}^{r-1})$表示第$r$轮函数的输入状态的值$X^{r-1}$是否能够被猜测确定, 其中模型变量$a_i^{r-1}$与$x_i^{r-1}$相对应$(0\leqslant i\leqslant 15)$。$a_i^{r-1}$为二元模型变量, 当$a_i^{r-1}$的值为1时, 说明$x_i^{r-1}$的数值能够被猜测确定; 否则, x_i^{r-1}的值不能被猜测确定。定义模型变量$\boldsymbol{K}^{r-1}=(K_{0,0}^{r-1},K_{0,1}^{r-1},K_{1,0}^{r-1},K_{1,1}^{r-1},K_{2,0}^{r-1},K_{2,1}^{r-1},K_{3,0}^{r-1},K_{3,1}^{r-1})$表示第$r$轮函数中使用的密钥$\boldsymbol{k}^{r-1}=(k_{0,0}^{r-1},k_{0,1}^{r-1},k_{1,0}^{r-1},k_{1,1}^{r-1},k_{2,0}^{r-1},k_{2,1}^{r-1},k_{3,0}^{r-1},k_{3,1}^{r-1})$是否被猜测, $k_{i,0}^{r-1}\in F_2^4$表示k_i^{r-1}的高4位, $k_{i,1}^{r-1}\in F_2^4$表示k_i^{r-1}的低4位。$K_{i,j}^{r-1}$为二元模型变量, 当$K_{i,j}^{r-1}$的值为1时, 表示密钥比特$k_{i,j}^{r-1}$的数值需要被猜测$(0\leqslant i\leqslant 3,\ 0\leqslant j\leqslant 1)$; 当$K_{i,j}^{r-1}$的值为0时, 表示密钥比特$k_{i,j}^{r-1}$的数值不需要被猜测。在对LLLWBC算法进行密钥恢复攻击时, 考虑简化轮的LLLWBC算法的最后1轮需要经过白化密钥, 定义白化密钥对应的模型变量为$\boldsymbol{K}^{\mathrm{b}}=(K_0^{\mathrm{b}},K_1^{\mathrm{b}},\cdots,K_{15}^{\mathrm{b}})$。

当K_i^{b}的值为1时, 表示后白化密钥中的第i个字$k_{\mathrm{e},i}^{w}$的数值需要被猜测$(0\leqslant i\leqslant 15)$; 否则, 不需要被猜测。

根据上述定义的模型变量及其含义, 结合式(15)(16), 可以使用如下的限制条件构建LLLWBC算法的密钥猜测轨迹:

当$1\leqslant r\leqslant 10$时, 基于式(15)构建MILP模型。对于第r轮内部状态存在的如下等式:

$$\begin{cases} x_0^{r-1}=x_2^r,\\ x_1^{r-1}=x_7^r,\\ x_4^{r-1}=x_{10}^r,\\ x_5^{r-1}=x_{14}^r,\\ x_8^{r-1}=x_{15}^r,\\ x_9^{r-1}=x_{11}^r,\\ x_{12}^{r-1}=x_3^r,\\ x_{13}^{r-1}=x_6^r。 \end{cases} \tag{18}$$

可得到如式(19)的模型变量之间的关系。

$$\begin{cases} a_0^{r-1}=a_2^r,\\ a_1^{r-1}=a_7^r,\\ a_4^{r-1}=a_{10}^r,\\ a_5^{r-1}=a_{14}^r,\\ a_8^{r-1}=a_{15}^r,\\ a_9^{r-1}=a_{11}^r,\\ a_{12}^{r-1}=a_3^r,\\ a_{13}^{r-1}=a_6^r。 \end{cases} \tag{19}$$

对于第 r 轮内部状态存在的如下等式：

$$\begin{cases} x_2^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_7^r))\oplus k_{0,0}^{r-1}\oplus x_{12}^r, \\ x_3^{r-1}=S(2\cdot S(x_2^r)\oplus 3\cdot S(x_7^r))\oplus k_{0,1}^{r-1}\oplus x_8^r, \\ x_6^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{1,0}^{r-1}\oplus x_0^r, \\ x_7^{r-1}=S(2\cdot S(x_{10}^r)\oplus 3\cdot S(x_{14}^r))\oplus k_{1,1}^{r-1}\oplus x_5^r, \\ x_{10}^{r-1}=S(2\cdot S(x_{15}^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{2,0}^{r-1}\oplus x_4^r, \\ x_{11}^{r-1}=S(2\cdot S(x_{15}^r)\oplus 3\cdot S(x_{11}^r))\oplus k_{2,1}^{r-1}\oplus x_1^r, \\ x_{14}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_6^r))\oplus k_{3,0}^{r-1}\oplus x_{13}^r, \\ x_{15}^{r-1}=S(2\cdot S(x_3^r)\oplus 3\cdot S(x_6^r))\oplus k_{3,1}^{r-1}\oplus x_9^r。 \end{cases} \tag{20}$$

在式(20)中，当每一个等式等号右面的变量全部猜测确定后，等号左面变量的值才能确定。以式(20)中的第 1 个等式为例，当 x_2^r, x_7^r, x_{12}^r的值被猜测确定，并且猜测了子密钥$k_{0,0}^{r-1}$后，x_2^{r-1}的值才能够被确定。对应的模型变量则存在如下的关系：

如果a_2^r, a_7^r, a_{12}^r, $K_{0,0}^{r-1}$同时为 1，则a_2^{r-1}一定为 1；否则a_2^{r-1}为 0。

上述关系表示如下：

$$guess(a_2^r,a_7^r,a_{12}^r,K_{0,0}^{r-1},a_2^{r-1})=\begin{cases} a_2^{r-1}\geqslant a_2^r+a_7^r+a_{12}^r+K_{0,0}^{r-1}-3, \\ a_2^{r-1}\leqslant a_2^r, \\ a_2^{r-1}\leqslant a_7^r, \\ a_2^{r-1}\leqslant a_{12}^r, \\ a_2^{r-1}\leqslant K_{0,0}^{r-1}。 \end{cases} \tag{21}$$

在式(21)中，第 1 行表示如果$a_2^r,a_7^r,a_{12}^r,K_{0,0}^{r-1}$同时为 1，则$a_2^{r-1}$一定为 1；第 2~4 行表示当$a_2^r,a_7^r,a_{12}^r,K_{0,0}^{r-1}$其中 1 个为 0 时，$a_2^{r-1}$一定为 0。式(21)中的限制关系本文使用函数$guess(a_2^r,a_7^r,a_{12}^r,K_{0,0}^{r-1},a_2^{r-1})$表示。根据上述描述，可以设计出 LLLWBC 算法的第 r 轮密钥猜测轨迹模型，具体如算法 7 所示。

算法 7. LLLWBC 算法的第 r 轮的密钥猜测轨迹模型：*guess_key*(*M*, *r*)。

输入：MILP 模型 *M*，轮数 r。

① 定义二元 MILP 模型变量：

$M.var\leftarrow a^{r-1}=\{a_0^{r-1},a_1^{r-1},a_2^{r-1},\cdots,a_{14}^{r-1},a_{15}^{r-1}\}$;

② 定义二元 MILP 模型变量：

$M.var\leftarrow a^{r}=\{a_0^{r},a_1^{r},a_2^{r},\cdots,a_{14}^{r},a_{15}^{r}\}$;

③ 定义二元 MILP 模型变量：

$M.var\leftarrow K^{r-1}=\{K_{0,0}^{r-1},K_{0,1}^{r-1},K_{1,0}^{r-1},K_{1,1}^{r-1},K_{2,0}^{r-1},K_{2,1}^{r-1},K_{3,0}^{r-1},K_{3,1}^{r-1}\}$;

④ for $1\leqslant r\leqslant 10$ do

⑤ $M.con\leftarrow a_0^{r-1}=a_2^r;a_1^{r-1}=a_7^r$;

⑥ $M.con\leftarrow guess(a_2^r,a_7^r,a_{12}^r,K_{0,0}^{r-1},a_2^{r-1})$;

⑦ $M.con\leftarrow guess(a_2^r,a_7^r,a_8^r,K_{0,1}^{r-1},a_3^{r-1})$;

⑧ $M.con\leftarrow a_4^{r-1}=a_{10}^r;a_5^{r-1}=a_{14}^r$;

⑨ $M.con\leftarrow guess(a_{10}^r,a_{14}^r,a_0^r,K_{1,0}^{r-1},a_6^{r-1})$;

⑩ $M.con\leftarrow guess(a_{10}^r,a_{14}^r,a_5^r,K_{1,1}^{r-1},a_7^{r-1})$;

⑪ $M.con\leftarrow a_8^{r-1}=a_{15}^r;a_9^{r-1}=a_{11}^r$;

⑫ $M.con\leftarrow guess(a_{15}^r,a_{11}^r,a_4^r,K_{2,0}^{r-1},a_{10}^{r-1})$;

⑬ $M.con\leftarrow guess(a_{15}^r,a_{11}^r,a_1^r,K_{2,1}^{r-1},a_{11}^{r-1})$;

⑭ $M.con\leftarrow a_{12}^{r-1}=a_3^r;a_{13}^{r-1}=a_6^r$;

⑮ $M.con\leftarrow guess(a_3^r,a_6^r,a_{13}^r,K_{3,0}^{r-1},a_{14}^{r-1})$;

⑯ $M.con\leftarrow guess(a_3^r,a_6^r,a_9^r,K_{3,1}^{r-1},a_{15}^{r-1})$;

⑰ end for

⑱ for $11\leqslant r\leqslant 20$ do

⑲ $M.con\leftarrow a_0^{r-1}=a_6^r;a_1^{r-1}=a_{11}^r$;

⑳ $M.con\leftarrow guess(a_6^r,a_{11}^r,a_0^r,K_{0,0}^{r-1},a_2^{r-1})$;

㉑ $M.con\leftarrow guess(a_6^r,a_{11}^r,a_{12}^r,K_{0,1}^{r-1},a_3^{r-1})$;

㉒ $M.con\leftarrow a_4^{r-1}=a_{10}^r;a_5^{r-1}=a_7^r$;

㉓ $M.con\leftarrow guess(a_{10}^r,a_7^r,a_{13}^r,K_{1,0}^{r-1},a_6^{r-1})$;

㉔ $M.con\leftarrow guess(a_{10}^r,a_7^r,a_1^r,K_{1,1}^{r-1},a_7^{r-1})$;

㉕ $M.con\leftarrow a_8^{r-1}=a_3^r;a_9^{r-1}=a_{15}^r$;

㉖ $M.con\leftarrow guess(a_3^r,a_{15}^r,a_4^r,K_{2,0}^{r-1},a_{10}^{r-1})$;

㉗ $M.con\leftarrow guess(a_3^r,a_{15}^r,a_9^r,K_{2,1}^{r-1},a_{11}^{r-1})$;

㉘ $M.con\leftarrow a_{12}^{r-1}=a_2^r;a_{13}^{r-1}=a_{14}^r$;

㉙ $M.con\leftarrow guess(a_2^r,a_{14}^r,a_5^r,K_{3,0}^{r-1},a_{14}^{r-1})$;

㉚ $M.con\leftarrow guess(a_2^r,a_{14}^r,a_8^r,K_{3,1}^{r-1},a_{15}^{r-1})$;

㉛ end for

在算法 7 中，行①②分别定义了第 r 轮的 LLLWBC 算法输入和输出对应的模型变量，行③定义了 LLLWBC 算法第 r 轮密钥对应的模型变量。当$1\leqslant r\leqslant 10$时，执行算法的行④~⑰来构建 LLLWBC 算法的第 r 轮函数的密钥猜测轨迹模型，这部分算法的设计主要是基于式(15)并结合式(19)(21)。对于 LLLWBC 算法，当$11\leqslant r\leqslant 20$时，轮函数的线性部件使用的是 P^{-1}置换，此时执行算法的行⑱~㉛，构建 LLLWBC 算法的密钥猜测轨迹。

如果得到的积分区分器为r_1轮，并向加密方向继续扩展了r_2轮。此时，攻击者的目标是对r_1+r_2轮加密的 LLLWBC 算法进行密钥恢复攻击。为了进行密钥恢复，攻击者需要猜测白化密钥和扩展的r_2轮中涉及到的密钥，并从第r_1+r_2轮解密到r_1轮，根据解密后数据集的性质是否与区分器的输出匹配，确定猜测的密钥是否正确。同时，由于第r_1+r_2轮的输出与白化密钥异或后得到的是简化轮的密码算法的输出，且简化轮的输出在攻击的过程中是已知的，当攻击者猜测白化密钥第 i 个字的值时，即模型变量K_i^b的值为 1，此时加密r_1+r_2轮后内部状态的第 i 个字节的值

也一定是能够通过猜测确定，即模型变量$a_i^{r_1+r_2}$的值为1。因此，存在如下的关系：

$$K_i^{\mathrm{b}} = a_i^{r_1+r_2},\ 0 \leqslant i \leqslant 15。\tag{22}$$

在进行密钥恢复的过程中，攻击者通常希望猜测的密钥比特最少，这样密钥恢复需要的时间复杂度也是最小。为了达到这样的目的，在MILP模型中需要加入目标函数：

$$M.optimize \leftarrow \min\left(\sum_{r=r_1+1}^{r=r_1+r_2}\sum_{i=0}^{i=3}\sum_{j=0}^{j=1}K_{i,j}^{r-1}+\sum_{i=0}^{15}K_i^{\mathrm{b}}\right),$$

结合式(22)，可得

$$M.optimize \leftarrow \min\left(\sum_{r=r_1+1}^{r=r_1+r_2}\sum_{i=0}^{i=3}\sum_{j=0}^{j=1}K_{i,j}^{r-1}+\sum_{i=0}^{15}a_i^{r_1+r_2}\right)。$$

根据上述讨论，基于算法7，我们设计了算法8用于检测对r_1+r_2轮的LLLWBC算法进行密钥恢复攻击需要猜测的最少的密钥比特。

算法8. 检测对r_1+r_2轮的LLLWBC算法进行密钥恢复攻击需要猜测的最少的密钥比特。

输入：r_1轮积分区分器中用于密钥恢复攻击的平衡字的下标集合BT；

输出：目标函数的值d，模型变量$\boldsymbol{K}^{r_1},\boldsymbol{K}^{r_1+1},\boldsymbol{K}^{r_1+2},\cdots,\boldsymbol{K}^{r_1+r_2-1},\boldsymbol{a}^{r_1+r_2}$的数值。

① 定义空的MILP模型M；

② for i in BT

③ $a_i^{r_1}=1$；

④ end for

⑤ for ($r=r_1+1$; $r=r+1$; $r\leqslant r_1+r_2$)

⑥ *guess_key*(M, r); /*算法7*/

⑦ end for

⑧ 向模型M中添加目标函数

$$\min\left(\sum_{r=r_1+1}^{r=r_1+r_2}\sum_{i=0}^{i=3}\sum_{j=0}^{j=1}K_{i,j}^{r-1}+\sum_{i=0}^{15}a_i^{r_1+r_2}\right);$$

⑨ 利用求解工具对MILP模型进行求解；

⑩ return 目标函数的值d和此时模型变量$\boldsymbol{K}^{r_1}$，$\boldsymbol{K}^{r_1+1},\boldsymbol{K}^{r_1+2},\cdots,\boldsymbol{K}^{r_1+r_2-1},\boldsymbol{a}^{r_1+r_2}$的数值。

在算法8中输入的参数BT为r_1轮积分区分器中用于密钥恢复攻击的平衡字的下标集合，这主要是由于在进行密钥恢复攻击的过程中，攻击者为了达到最大的分析轮数且复杂度不超过穷举攻击，可能会只选取积分区分器中的部分平衡的输出位置进行密钥恢复攻击。例如，攻击者得到r_1轮的积分区分器中平衡字的下标位置为0, 1, 2, …, 9，攻击者可以从集合{0, 1, 2, …, 9}中任选出w个数作为密钥恢复攻击过程需要利用到的平衡的位置，此时攻击者需要在第r_1+1轮至第r_1+r_2轮中猜测合适的子密钥，使得加密后的密文能够部分解密r_2轮，并得到攻击者选取的平衡中间状态。因此，算法8在MILP模型M中加入了行②~④的限制条件，这些限制条件用于确保通过猜测$\boldsymbol{K}^{r_1},\boldsymbol{K}^{r_1+1},\boldsymbol{K}^{r_1+2},\cdots,\boldsymbol{K}^{r_1+r_2-1},\boldsymbol{a}^{r_1+r_2}$中为1的位置对应的密钥，密码分析者能够确定满足$a_i^{r_1}=1$对应的内部状态$x_i^{r_1}$的值。

利用表5中的第1个9轮积分区分器进行密钥恢复攻击，具体的区分器形式为：

$$\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{c}^4\boldsymbol{c}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{a}^4\boldsymbol{c}^4\boldsymbol{a}^4 \xrightarrow{9\text{轮}} \boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{u}^4\boldsymbol{u}^4\boldsymbol{b}^4\boldsymbol{b}^4。\tag{23}$$

通过测试发现，当只使用该区分器输出的第2个平衡字节对12轮的LLLWBC算法进行密钥恢复攻击需要的复杂度小于穷举密钥需要的时间时，其他情况将超过穷举密钥的时间复杂度。因此，在后续的密钥恢复攻击中，我们只关注式(23)中积分区分器的第2个平衡字。此时，需要按照算法8中的行②~④在MILP模型中加入如下的限制条件

$$M.con \leftarrow a_2^9=1。\tag{24}$$

然后，利用算法8，可得目标函数的值为19，最优解对应的模型变量的值为

$$\boldsymbol{K}^9=(1,0,0,0,0,0,0,0),$$
$$\boldsymbol{K}^{10}=(1,0,0,1,0,0,0,0),$$
$$\boldsymbol{K}^{11}=(1,0,1,1,1,1,0,0),$$
$$\boldsymbol{a}^{12}=(1,1,0,1,1,0,1,1,0,1,1,1,0,1,0,1)。$$

根据上述论述得到的结果为：

对于第10轮，需要猜测的轮密钥字为$k_{0,0}^9$，$k_{0,0}^9$只与主密钥中的k_{24}相关；

对于第11轮，需要猜测的轮密钥字为$k_{0,0}^{10}$和$k_{1,1}^{10}$，$k_{0,0}^{10}$只与主密钥中的k_{16}相关，$k_{1,1}^{10}$只与主密钥中的k_{19}相关；

对于第12轮，需要猜测的轮密钥字为$k_{0,0}^{11}$，$k_{1,0}^{11}$，$k_{1,1}^{11}$，$k_{2,0}^{11}$，$k_{2,1}^{11}$，这些密钥字只与主密钥中的k_{24}，k_{26}，k_{27}，k_{28}，k_{29}相关。

因此和轮密钥相关的主密钥集合为{k_{16}, k_{19}, k_{24}, k_{26}, k_{27}, k_{28}, k_{29}}，需要猜测的主密钥是28b，而对于后白化密钥，根据模型变量$\boldsymbol{a}^{12}$的值，可以确定需要猜测的后白化密钥密钥为{$k_{\mathrm{e},0}^{\mathrm{w}},k_{\mathrm{e},1}^{\mathrm{w}},k_{\mathrm{e},3}^{\mathrm{w}},k_{\mathrm{e},4}^{\mathrm{w}},k_{\mathrm{e},6}^{\mathrm{w}},k_{\mathrm{e},7}^{\mathrm{w}},k_{\mathrm{e},9}^{\mathrm{w}},k_{\mathrm{e},10}^{\mathrm{w}},k_{\mathrm{e},11}^{\mathrm{w}},k_{\mathrm{e},13}^{\mathrm{w}},k_{\mathrm{e},15}^{\mathrm{w}}$}，由$k_{\mathrm{e}}^{\mathrm{w}}=(k^{\mathrm{w}}\ggg 1)\oplus(k^{\mathrm{w}}\gg 63)$可得，需要猜测的主密钥是44b。

综上所述，需要猜测的主密钥是72b，即猜测的密钥空间为2^{72}。然后，利用式(23)中的区分器形式构建明文集合，并进行12轮的LLLWBC算法的加密，

得到简化轮的密文集合。随后，猜测 72b 的密钥信息，对简化轮的密文集合进行解密。最后，判断第 9 轮的第 2 个字是否平衡。如果是不平衡的，说明猜测的密钥错误；如果是平衡的，说明猜测的密钥可能是正确的。由于错误的密钥使得x_2^9是平衡的概率为2^{-4}，上述过程执行 1 次能够恢复 4b 的密钥信息。经过 18 组的明文筛选后，能够恢复出 72b 的密钥信息。剩余的 56b 密钥信息可以通过穷举搜索得到。LLLWBC 算法解密 3 轮的时间复杂度是$2^{52}\times(2^{72}+2^{68}+2^{64}+2^{60}+\cdots+2^8+2^4)\approx 2^{124.09}$次 3 轮 LLLWBC 加密，等价于进行$2^{124.09}/4=2^{122.09}$次 12 轮 LLLWBC 加密。因此，该密钥恢复需要的总时间复杂度为$2^{56}+2^{122.09}\approx 2^{122.09}$次 12 轮 LLLWBC 加密，数据复杂度为$18\times 2^{52}=2^{56.17}$个 64b 明文数据，存储复杂度为$2^{72}$个 128b 的密钥和为$2^{52}$个 64b 的密文数据。

5 结　论

本文围绕新型低延迟轻量级分组密码 LLLWBC 的安全性展开系统研究，提出了一种基于代数结构的新型自动化积分分析框架。与传统依赖人工推导或单一可分性建模的方法相比，本文在代数结构搜索过程中同时统计明文字出现的次数和分量传播至密文需经过的 S 盒个数，据此定义并构造新型字传播轨迹，实现了对候选代数结构的自动化检测与筛选，并能够直接生成可用的积分区分器，有效缓解了在高轮数下代数结构复杂度急剧增长所带来的难题。在实验评估中，本文在 13 个活跃字条件下成功构造出 9 轮积分区分器，相较于设计者提出的 8 轮区分器提升了 1 轮。

进一步地，本文针对密钥恢复问题提出了一种基于 MILP 的密钥猜测路径优化建模方法。该方法依据轮函数结构与密钥编排原理构建密钥恢复的 MILP 模型，用代数约束刻画中间状态与子密钥的依赖关系，最后通过求解模型得到最优的密钥猜测路径。基于该模型，本文实现了对 12 轮 LLLWBC 的密钥恢复攻击，其数据复杂度为$2^{56.17}$个 64b 明文数据，存储复杂度为2^{72}个 128b 的密钥和2^{52}个 64b 的密文数据，时间复杂度为$2^{122.09}$次 12 轮加密操作。

本文提出了积分区分器自动化搜索方法与基于 MILP 建模的密钥恢复策略，能够为广义 Feistel 结构与大状态轻量级分组密码的安全评估提供高效、通用的分析工具。未来工作可进一步探索该方法在其他轻量级密码算法上的应用。

作者贡献声明：叶涛、韦永壮负责方案整体设计并撰写论文；焦留洋和周尔哲负责算法思路和实验方法；李灵琛提出指导意见并修改论文。

参　考　文　献

[1] Zhang Lei, Wu Ruichen, Zhang Yuhan, et al. LLLWBC: A new low-latency light-weight block cipher[C]//Proc of the 19th Int Conf on Information Security and Cryptology. Berlin: Springer, 2023: 23−42

[2] Biham E, Shamir A. Differential cryptanalysis of DES-like cryptosystems[J]. Journal of Cryptology, 1991, 4(1): 3−72

[3] Matsui M. Linear cryptanalysis method for DES cipher[C]//Proc of the 12th Workshop on the Theory and Application of Cryptographic Techniques. Berlin: Springer, 1993: 386−397

[4] Demirci H, Selcuk A. A meet-in-the-middle attack on 8-round AES[C]//Proc of the 15th Int Workshop on Fast Software Encryption. Berlin: Springer, 2008: 116−126

[5] Armknecht F, Krause M. Algebraic attacks on combiners with memory[C]//Proc of the 23rd Annual Int Cryptology Conf. Berlin: Springer, 2003: 162−175

[6] Knudsen L, Wagner D. Integral cryptanalysis[C]//Proc of the 9th Int Workshop on Fast Software Encryption. Berlin: Springer, 2002: 112−127

[7] Todo Y. Structural evaluation by generalized integral property[C]//Proc of the 34th Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2015: 287−314

[8] Todo Y, Morii M. Bit-based division property and application to SIMON family[C]//Proc of the 23rd Int Conf on Fast Software Encryption. Berlin: Springer, 2016: 357−377

[9] Xiang Zejun, Zhang Wentao, Bao Zhenzhen, et al. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers[C]//Proc of the 23rd Int Conf on Fast Software Encryption. Berlin: Springer, 2016: 357−377

[10] Hu Kai, Wang Meiqin. Automatic search for a variant of division property using three subsets[C]//Proc of the 19th Cryptographers' Track at the RSA Conf. Berlin: Springer, 2019: 412−432

[11] Hao Yonglin, Leader G, Meier W, et al. Modeling for three-subset division property without unknown subsets[C]//Proc of the 39th Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2020: 466−495

[12] Hu Kai, Sun Siwei, Wang Meiqin, et al. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums[C]//Proc of the 26th Advances in Cryptology (ASIACRYPT 2020). Berlin: Springer, 2020: 446−476

[13] Cui Jiamin, Hu Kai, Wang Qingju, et al. Integral attacks on Pyjamask-96 and round-reduced Pyjamask-128[C]//Proc of the 22nd Cryptographers' Track at the RSA Conf. Berlin: Springer, 2022: 223−246

[14] Chakraborty D. Finding three-subset division property for ciphers with

complex linear layers[C]//Proc of the 23rd Progress in Cryptology (INDOCRYPT 2022). Berlin: Springer, 2023: 398−421

[15] Zhang Wenying, Cao Meichun, Guo Jian, et al. Improved security evaluation of SPN block ciphers and its applications in the single-key attack on SKINNY[J]. IACR Transactions on Symmetric Cryptology, 2019, 2019(4): 171−191

[16] Ye Tao, Wei Yongzhuang, Li Lingchen. Integral cryptanalysis of ACE encryption algorithm[J]. Journal of Electronics & Information Technology, 2021, 43(4): 908−914 (in Chinese)
(叶涛, 韦永壮, 李灵琛. ACE 密码算法的积分分析[J]. 电子与信息学报, 2021, 43(4): 908−914)

[17] Xing Zhaohui, Zhang Wenying, Cao Meichun. Integral cryptanalysis on lightweight block cipher WARP based on the algebraic structure perspective[J]. Journal of Computer Research and Development, 2023, 60(4): 860−872 (in Chinese)
(邢朝辉, 张文英, 曹梅春. 基于代数结构视角对轻量分组密码 WARP 的积分分析[J]. 计算机研究与发展, 2023, 60(4): 860−872)

[18] Borghoff J, Canteaut A, Güneysu T, et al. PRINCE—A low-latency block cipher for pervasive computing applications[C]//Proc of the 32nd Workshop on the Theory and Application of Cryptographic Techniques. Berlin: Springer, 2012: 208−225



Ye Tao, born in 1991. PhD, lecturer, master supervisor. His main research interest includes design and analysis of symmetric cryptographic algorithms.

叶 涛, 1991 年生。博士, 讲师, 硕士生导师。主要研究方向为对称密码算法的设计与分析。



Jiao Liuyang, born in 1999. Master. Her main research interest includes design and analysis of symmetric cryptographic algorithms.

焦留洋, 1999 年生。硕士。主要研究方向为对称密码算法的设计与分析。



Wei Yongzhuang, born in 1976. PhD, professor, PhD supervisor. His main research interests include the design of symmetric cryptographic algorithms, intelligent analysis of cryptographic algorithms, and side-channel cryptanalysis.

韦永壮, 1976 年生。博士, 教授, 博士生导师。主要研究方向为对称密码算法的设计、密码算法智能化分析、侧信道密码分析。



Li Lingchen, born in 1988. PhD, lecturer, master supervisor. Her main research interest includes design and analysis of symmetric cryptographic algorithms.

李灵琛, 1988 年生。博士, 讲师, 硕士生导师。主要研究方向为对称密码算法的设计与分析。



Zhou Erzhe, born in 1999. Master. His main research interest includes design and analysis of symmetric cryptographic algorithms.

周尔哲, 1999 年生。硕士。主要研究方向为对称密码算法的设计与分析。