Modeling and Simulation 建模与仿真, 2025, 14(8), 319-325
Published Online August 2025 in Hans. https://www.hanspub.org/journal/mos
https://doi.org/10.12677/mos.2025.148570

Hans汉斯

缩减轮GRANULE算法的中间相遇分析

刘先蓓[1]，张艺昕[2]

[1]安徽财经大学统计与应用数学学院，安徽 蚌埠
[2]上海理工大学光电信息与计算机工程学院，上海

收稿日期：2025年7月26日；录用日期：2025年8月18日；发布日期：2025年8月26日

摘　要

轻量级分组密码是为计算资源受限的环境而设计的加密算法。它具有计算开销小、占用存储空间少、能耗低等优点，广泛用于嵌入式系统、智能卡以及物联网等应用场景。然而，为了追求更高的实现效率，设计轻量级分组密码算法时会牺牲部分安全性，所以有必要评估它的安全强度。本文利用中间相遇分析法对轻量级分组密码GRANULE算法的安全性进行了评估。首先，构造了7轮的GRANULE的中间相遇区分器，随后在它的前面接4轮、后面接3轮，构建了14轮的GRANULE中间相遇分析的攻击路径，并在结合GRANULE的轮密钥之间的一些线性关系之后，最终攻击需要的数据复杂度为2^{60}选择明文，时间复杂度为$2^{111.2}$次14轮GRANULE加密，存储复杂度为$2^{104.9}$个64比特块。此结果是对GRANULE安全性分析的有效补充，攻击结果表明GRANULE能较好地抵抗中间相遇攻击。

关键词

轻量级分组密码，GRANULE算法，Feistel结构，中间相遇分析

Meet-in-the-Middle Attacks on Reduced-Round GRANULE

Xianbei Liu[1], Yixin Zhang[2]

[1]School of Statistics and Applied Mathematics, Anhui University of Finance and Economics, Bengbu Anhui
[2]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai

Received: Jul. 26th, 2025; accepted: Aug. 18th, 2025; published: Aug. 26th, 2025

Abstract

Lightweight block cipher is an encryption algorithm designed for environments with limited computing resources. It has the advantages of low computational cost, less storage space occupation,

文章引用: 刘先蓓, 张艺昕. 缩减轮 GRANULE 算法的中间相遇分析[J]. 建模与仿真, 2025, 14(8): 319-325.
DOI: 10.12677/mos.2025.148570

and low energy consumption, and is widely used in application scenarios such as embedded systems, smart cards, and the Internet of Things. However, in order to pursue higher implementation efficiency, designing lightweight block cipher algorithms may sacrifice some security, so it is necessary to evaluate its security strength. In this article we evaluate the security of the lightweight block cipher GRANULE algorithm using the intermediate encounter analysis method. Firstly, a 7-round GRANULE intermediate encounter discriminator was constructed, followed by 4 rounds in front and 3 rounds behind it. 14-round GRANULE intermediate encounter analysis attack path was constructed, and after combining some linear relationships between GRANULE's round keys. Data complexity is 2^{60} chosen plaintexts, time complexity is $2^{111.2}$ 14-round GRANULE encryption, and storage complexity is $2^{104.9}$ 64-bit blocks. This result is an effective supplement to the security analysis of GRANULE, and the attack results indicate that GRANULE can resist intermediate encounter attacks well.

Keywords

Lightweight Block Ciphers, GRANULE, Feistel Structure, Meet-in-the-Middle Attack

1. 引言

随着物联网、嵌入式系统和智能设备的快速发展，越来越多的终端被部署在计算能力弱、存储空间小、能耗要求严格的环境中，而这些设备同样面临着数据加密与安全通信的需求。传统分组密码算法如AES 在这些场景中存在实现复杂、资源开销大等问题，往往难以高效运行。保证基本安全性的前提下，为了实现低功耗、小面积、高效率的加密操作，轻量级分组密码应运而生。

轻量级分组密码是一类为计算资源受限的环境而设计的加密算法，具有计算开销小、占用存储空间少、能耗低等优点，因而非常适用于嵌入式系统、智能卡以及物联网等应用场景。在这一领域的发展过程中，诸如 WARP [1]、PRESENT [2]、Lblock [3]和 GRANULE [4]等密码算法被广泛应用。然而，随着各类信息安全事件的频繁发生，数据保护的重要性日益凸显。然而，为了实现高效率，设计轻量级分组密码时通常会在一定程度上牺牲安全性，从而为潜在攻击留下了空间，因此研究轻量级分组密码的安全性非常必要。

GRANULE 是 BANSOD B 和 PATIL A 等人在 2018 年提出的一款轻量级分组密码，该算法是一种 Feistel-Substitution Permutation (Feistel-SP)结构，采用 64 比特的分组长度并迭代 32 轮。算法有两个版本：GRANULE-64/80 和 GRANULE-64/128，分别对应 80 比特及 128 比特的密钥长度[4]。GRANULE 算法的轮函数设计中使用了 8 个 4 比特的 S 盒及两次循环异或和使得其满足混淆扩散原则，因此对其安全性的研究可以为选择安全高效的轻量级分组密码提供理论依据。2019 年，石淑英等[5]基于一条 GRANULE 算法 5 轮不可能差分链，扩展 6 轮，构建了 GRANULE-64/80 算法的 11 轮不可能差分分析。2020 年，武小年等[6]通过 S 盒输入/输出差分特征的规律，遍历搜索获得了几条 GRANULE 算法的 7 轮不可能差分区分器，但未进行密钥恢复攻击。2021 年，赵晨曦[7]也利用 S 盒差分特征规律得到 GRANULE 算法的 7 轮不可能差分链，分别在前、后各扩展 3 轮，对 GRANULE-64/80 发起了 13 轮不可能差分攻击。2023 年，刘先蓓等[8]利用中间相错技术构造了 GRANULE 算法 7 轮不可能差分区分器，同时将构造的区分器往前往后各扩展 3 轮，结合密钥的线性相关特征，对 GRANULE-64/80 发起了 13 轮不可能差分攻击。可以看出，研究者们主要是利用不可能差分攻击的方法对 GRANULE-64/80 算法进行了分析。

1977 年，Hellman 和 Diffie 首次提出了中间相遇攻击的概念，其核心思想是小幅提升存储复杂度来换取时间复杂度的降低[9]，即时间－空间折中策略。随后，Demirci 等人[10]分析 AES 时对此方法进行了改进，将攻击过程分为离线和在线两个阶段。离线阶段，攻击者的主要任务是构造一个中间相遇区分器，并在区分器前端部分构建一个 δ-集，计算出与之对应的输出端有序序列，存储在预计算表中，为在线阶段的攻击做好准备。在线阶段，攻击者在区分器的前端和后端分别延拓数轮形成用于恢复密钥的路径。随后攻击者选择符合路径差分的明密文对，并猜测加解密过程中前端和后端需要猜测的密钥部分，对选定的明密文对进行加解密操作，再查询预计算表，寻找是否存在与加解密结果相匹配的项。若发现匹配项，表明猜测的密钥可能是正确的密钥，否则为错误密钥，将被过滤。当排除所有错误密码后，最终恢复正确密钥。中间相遇攻击还可以和多重集技术、差分枚举技术和密钥桥技术等技术结合形成更加高效的攻击，广泛适用于分组密码，提高分析效率，目前已对许多分组密码抵抗中间相遇攻击能力的评估，如 AES [11]、MIBS [12]、Midori [13]、LBlock [14]等。因此，研究 GRANULE 抵抗中间相遇攻击的能力非常重要。

本文利用 δ-集及差分枚举技术，结合 GRANULE 算法结构特性，获得了一个 7 轮中间相遇区分器，基于该区分器，前、后分别增加 4 轮、3 轮，构建了 14 轮 GRANULE-64/128 的中间相遇攻击路径。攻击所需要数据复杂度为 2^{60} 个选择明文，时间复杂度为 $2^{111.3}$ 次 14 轮 GRANULE-64/128 加密，存储复杂度为 $2^{104.9}$ 个 64 比特块。此结果是对 GRANULE-64/128 抵抗中间相遇攻击的一个重要补充。

2. 预备知识

本节对文中的符号及 GRANULE 算法进行介绍。

2.1. 符号简介

(1) M,C,K：分别表示明文、密文和主密钥；

(2) L_i,R_i：分别表示第 i 轮左、右侧部分，$1\le i\le 32$；

(3) $L_i[j],R_i[j]$：分别表示第 i 轮左、右侧部分第 j 个半字节的值，$1\le i\le 32$，$0\le j\le 7$；

(4) $\Delta L_i[j],\Delta R_i[j]$：分别表示第 i 轮左、右侧部分第 j 个半字节的差分值，$1\le i\le 32$，$0\le j\le 7$；

(5) $Y_i[j]$：第 i 轮经过 P 置换后 S 盒输入的第 j 个半字节的值，$1\le i\le 32$，$0\le j\le 7$；

(6) RK_i, RK_i^j：分别表示第 i 轮使用的子密钥及它的第 j 个半字节，$1\le i\le 32$，$0\le j\le 7$；

(7) $<<<\alpha$，$>>>\beta$：分别表示循环左移 α 比特、右移 β 比特；

(8) $\oplus$：异或操作；

(9) $[i]^2$：i 的五位二进制表示；

(10) $\|$：二进制字符连接。

2.2. GRANULE 算法介绍

GRANULE 算法分组长度为 64 比特，左、右分支分别为 32 比特，整个算法经过 32 轮迭代。算法采用 Feistel 结构，见图 1。该算法轮函数中，包含 P 置换、S 盒、循环移位异或运算及轮密钥异或运算。下面具体介绍算法 P 置换、S 盒及密钥编排算法。

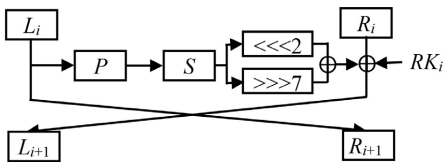

Figure 1. GRANULE algorithm structure
图 1. GRANULE 算法结构

(1) P 置换。该置换是对左分支字节重新排序。首先将左分支 L 分成 8 个半字节 $l_0l_1l_2l_3l_4l_5l_6l_7$，则经过 P 置换后将得到 $l_1l_3l_5l_2l_0l_7l_4l_6$，即

$$l_1l_3l_5l_2l_0l_7l_4l_6 = P\left(l_0l_1l_2l_3l_4l_5l_6l_7\right)$$

(2) S 盒。GRANULE 算法采用 8 个并置的 S 盒，每一个 S 盒输入输出均为 4 比特，具体如表 1 所示。

Table 1. S box of GRANULE
表 1. GRANULE 的 S 盒

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

(3) 密钥编排算法。GRANULE 算法有两个版本：GRANULE-64/80 和 GRANULE-64/128，分别对应 80 比特密钥和 128 比特密钥。本文针对 GRANULE-64/128 进行密钥恢复攻击，在此仅介绍 128 比特版本的密钥编排算法。

将 128 比特的主密钥存入密钥寄存器 K 中，$K = K_{127} \| K_{126} \| \cdots \| K_1 \| K_0$。第一轮子密钥 RK_1 为 K 最右端的 32 比特，即 $RK_1 = K_{31}K_{30} \cdots K_1K_0$。随后更新密钥寄存器 K，以 K 最右端的 32 比特作为第 r ($2 \le r \le 32$)轮子密钥 RK_r，更新算法如下所示：

$$\begin{cases} K <<< 31; \\ \left[K_3K_2K_1K_0\right] \leftarrow S\left[K_3K_2K_1K_0\right]; \\ \left[K_7K_6K_5K_4\right] \leftarrow S\left[K_7K_6K_5K_4\right]; \\ \left[K_{70}K_{69}K_{68}K_{67}K_{66}\right] \leftarrow \left[K_{70}K_{69}K_{68}K_{67}K_{66}\right] \oplus \left[i\right]^2; \end{cases}$$

3. 14 轮 GRANULE 中间相遇攻击过程

这一章节主要介绍 7 轮中间相遇区分器的构造以及 14 轮中间相遇攻击的具体过程。

3.1. 7 轮 GRANULE 中间相遇区分器

通过列举多条中间相遇区分器，找到一条在预计算过程中猜测的中间参数最少的中间相遇差分链，见图 2。图 2 详细展示了中间相遇差分链 $\left(R_1\left[7\right]; R_8\left[4\right], R_8\left[5\right]\right)$ 的具体结构，其中白色表示该半字节位置的差分为零，灰色表示此半字节位置的差分受区分器输入差分影响，黑色表示此半字节位置差分影响区分器的输出差分，同时受区分器输入差分影响。

定理 1：若 δ-集满足 $R_1\left[7\right]$ 活跃，其他位置非活跃，则经过 7 轮 GRANULE 加密后输出差分序列：$R_8^0\left[4\right] + R_8^1\left[4\right]$，$R_8^0\left[4\right] + R_8^2\left[4\right]$，…，$R_8^0\left[4\right] + R_8^{15}\left[4\right]$，$R_8^0\left[5\right] + R_8^1\left[5\right]$，$R_8^0\left[5\right] + R_8^2\left[5\right]$，…，$R_8^0\left[5\right] + R_8^{15}\left[5\right]$ 可由 $Y_2\left[5\right]$，$Y_3\left[7,6,2,1\right]$，$Y_4\left[7 \sim 0\right]$，$Y_5\left[7 \sim 0\right]$，$Y_6\left[7 \sim 3\right]$ 共 26 个参数表示。

证明：如图 2 所示，假设区分器输入差分满足 $\Delta R_1\left[7\right]$ 为非零差分，其他位置均为零差分。

(1) 猜测 $Y_2\left[5\right]$，则第二轮 S 盒的输入值已知，经过 S 盒后与密钥进行异或，因为异或操作、移位操作、分组置换不会改变差分值，因此第二轮右侧的输出差分 ΔR_2 已知，也即第三轮左侧输入差分 ΔL_3 已知；

(2) 因为 $\Delta L_3\left[6,5,4,3\right]$ 非零，经 P 置换后，$\Delta Y_3\left[7,6,2,1\right]$ 非零，因此猜测 $Y_3\left[7,6,2,1\right]$，则第三轮右侧的输出差分 ΔR_3 已知；

(3) 因为 $\Delta L_4 = \Delta R_3$ 已知，猜测 $Y_4\left[7 \sim 0\right]$，则第四轮右侧的输出差分 ΔR_4 已知；

(4) 同理，猜测 $Y_5\left[7 \sim 0\right]$ 的值，则第五轮右侧的输出差分 ΔR_5 已知；

(5) 猜测 $Y_6\left[7 \sim 3\right]$ 的值，则获得第六轮右侧的输出差分 ΔR_6，即第 7 轮左侧的 $\Delta L_7\left[5\right], \Delta L_7\left[4\right]$ 已知；

此时，不需要再另外猜测 Y，即可得到第 8 轮右侧的 $\Delta R_8[5],\Delta R_8[4]$，即 $R_8[5],R_8[4]$ 的 30 个 4 比特差分序列可以由 $Y_2[5]$，$Y_3[7,6,2,1]$，$Y_4[7\sim 0]$，$Y_5[7\sim 0]$，$Y_6[7\sim 3]$ 这 26 个参数表示。

由定理 1 可知，$R_8[5],R_8[4]$ 处，这 15×2 个 4 比特差分序列仅有 $2^{26\times 4}=2^{104}$ 种取值，而非 $2^{30\times 4}=2^{120}$ 种取值。

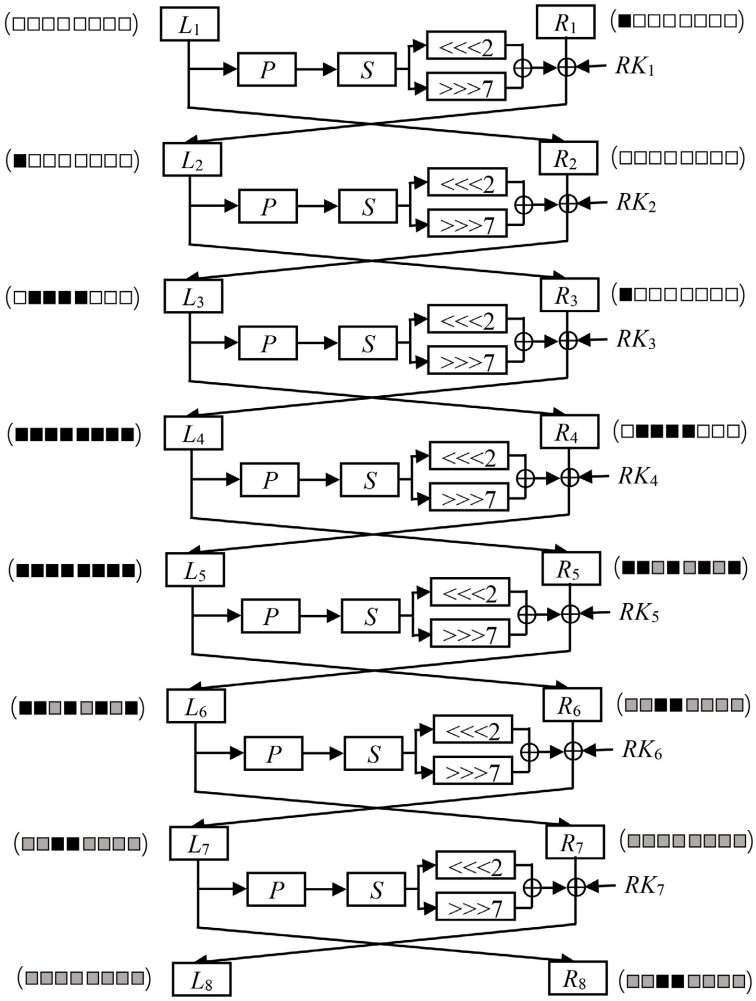

Figure 2. The specific structure of the intermediate encounter differential chain $(R_1[7];R_8[4],R_8[5])$

图 2. 中间相遇差分链 $(R_1[7];R_8[4],R_8[5])$ 的具体结构

3.2. 14 轮 GRANULE 中间相遇攻击

基于上述的 7 轮区分器，在前端加 4 轮，后面接 3 轮，对 14 轮 GRANULE 进行中间相遇攻击，攻击路径见图 3。攻击分为离线和在线两个部分，具体如下。

离线阶段：计算 120 比特差分序列的所有 2^{104} 种可能值，并存储于哈希表中。

在线阶段：选择符合图 3 路径的明文对，构造对应的 δ -集，猜测部分轮密钥得到差分序列 $\Delta R_8[5],\Delta R_8[4]$，检查是否与离线阶段建立的哈希表中的项匹配。若没有相匹配的项，则猜测密钥为错误。删除错误密钥后，在剩余密钥基础上穷搜恢复完整主密钥。具体过程如下：

(1) 选择符合图 3 的明文 $M^0=L_1^0\|R_1^0$，猜测密钥 $RK_1^{7\sim 5}$、$RK_1^{3\sim 0}$，部分加密 M^0 得到 $L_2^0\|R_2^0$；

(2) 猜测密钥 RK_2^7、RK_2^5、RK_2^4、RK_2^0，部分加密 $L_2^0 \| R_2^0$ 得 $L_3^0 \| R_3^0$；

(3) 猜测密钥 RK_3^7，部分加密 $L_3^0 \| R_3^0$ 得 $L_4^0 \| R_4^0$。因为 $R_5^0 = L_4^0$，所以可得区分器的输入端 $L_5^0 \| R_5^0$；

(4) 已知 $L_{15} \| R_{15}$ 及其差分，需要猜测 $RK_{14}^{7\sim0}$，对 $L_{15} \| R_{15}$ 进行解密即可得到 $L_{14} \| R_{14}$ 和 $\Delta L_{14} \| \Delta R_{14}$；

(5) 猜测 $RK_{13}^{7\sim0}$，对 $L_{14} \| R_{14}$ 进行解密即可得到 $L_{13} \| R_{13}$ 和 $\Delta L_{13} \| \Delta R_{13}$；

(6) 对于 $L_{13} \| R_{13}$，L_{13} 经过 S 盒之后得到的输出差分与 ΔR_{13} 异或即可得到区分器的输出差分 ΔR_{12} [5]，ΔR_{12} [4]；

(7) 检查解密所得差分序列是否存在于离线阶段建立的哈希表中。若不存在，否则删除猜测密钥。在对剩余密钥穷举搜索进而恢复完整主密钥。

此时，在整个攻击过程中，一共猜测了轮密钥 $RK_1^{7\sim5}$、$RK_1^{3\sim0}$、RK_2^7、RK_2^5、RK_2^4、RK_2^0、RK_3^7、$RK_{13}^{7\sim0}$、$RK_{14}^{7\sim0}$，由密钥关系可知，RK_1^0 的最低位与 RK_2^7 的最高位一样，所以在此处可少猜一位密钥，共 111 个密钥。

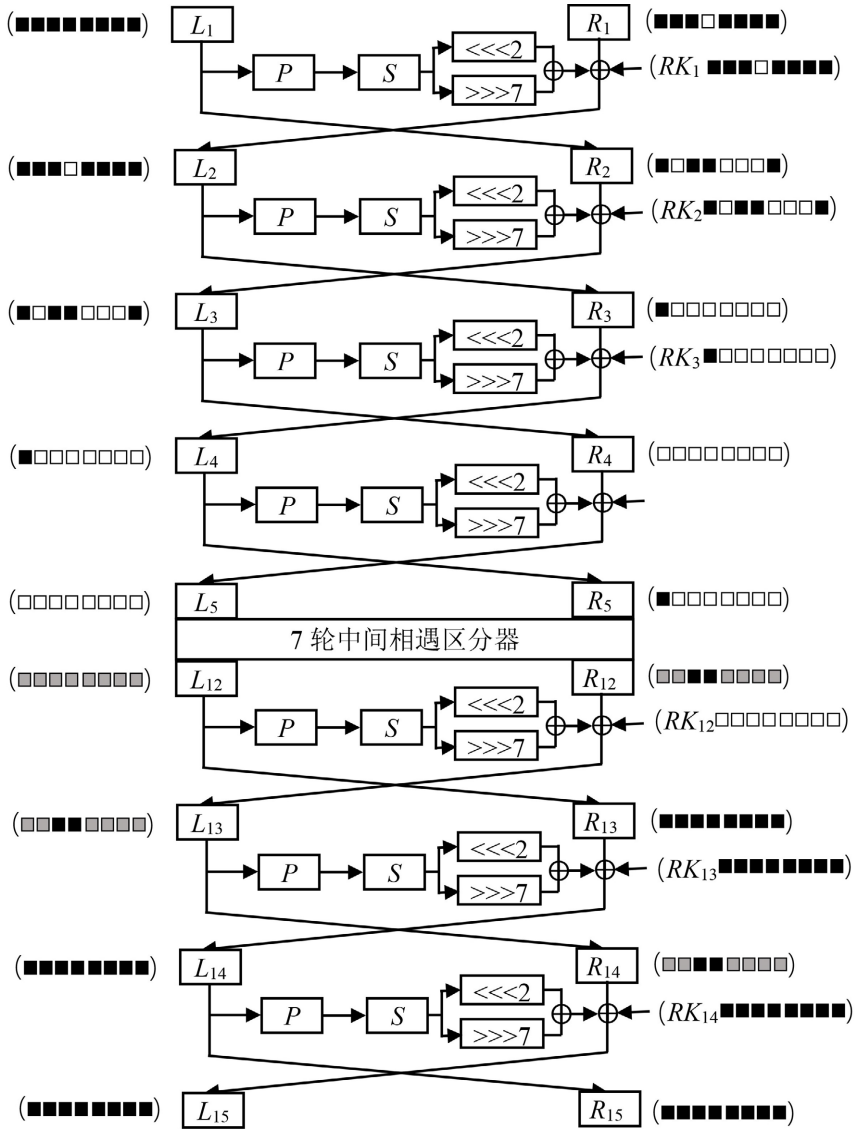

Figure 3. Attack path encountered in the middle of round 14 GRANULE
图 3. 14 轮 GRANULE 中间相遇攻击路径

3.3. 复杂度分析

由于选择明文 M^0 要求 1 个半字节差分为零，15 个半字节差分非零，故攻击所需数据量为 $2^{15\times4}=2^{60}$ 个选择明文。

离线阶段时间复杂度为 $2^4\times2^{26\times4}$ 次部分加密，约为 $2^4\times2^{26\times4}/14\approx2^{104.2}$ 次 14 轮 GRANULE 加密；存储为 $2^{26\times4}=2^{104}$ 个 120 比特差分序列，$120/64=1.875\approx2^{0.9}$，故存储复杂度约为 $2^{104.9}$ 个 64 比特块。

在线阶段经过 $2^4\times2^{111}$ 次部分加解密，约为 $2^4\times2^{111}/14\approx2^{111.2}$ 次 14 轮 GRANULE 加密。

综上，对 14 轮 GRANULE 中间相遇攻击所需的最终攻击需要的数据复杂度为 2^{60} 选择明文，时间复杂度为 $2^{111.2}$ 次 14 轮 GRANULE 加密，存储复杂度为 $2^{104.9}$ 个 64 比特块。

4. 结论

本文利用 δ -集及差分枚举技术，结合 GRANULE 算法结构特性，获得了一个 7 轮中间相遇区分器，基于该区分器，前、后分别增加 4 轮、3 轮，构建了 14 轮 GRANULE-64/128 的中间相遇攻击路径。攻击所需要数据复杂度为 2^{60} 个选择明文，时间复杂度为 $2^{111.2}$ 次 14 轮 GRANULE-64/128 加密，存储复杂度为 $2^{104.9}$ 个 64 比特块。此结果是对 GRANULE-64/128 抵抗中间相遇攻击的一个重要补充。

基金项目

安徽省高校自然科学重点项目(2024AH050011)。

参考文献

[1] Banik, S., Bao, Z., Isobe, T., Kubo, H., Liu, F., Minematsu, K., *et al.* (2021) WARP: Revisiting GFN for Lightweight 128-Bit Block Cipher. In: Dunkelman, O., Jacobson Jr., M.J. and O'Flynn, C., Eds., *Selected Areas in Cryptograph*, Springer, 535-564. https://doi.org/10.1007/978-3-030-81652-0_21

[2] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., *et al.* (2007) PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P. and Verbauwhede, I., Eds., *Cryptographic Hardware and Embedded Systems—CHES* 2007, Springer, 450-466. https://doi.org/10.1007/978-3-540-74735-2_31

[3] Wu, W. and Zhang, L. (2011) LBlock: A Lightweight Block Cipher. In: Lopez, J. and Tsudik, G., Eds., *Applied Cryptography and Network Security*, Springer, 327-344. https://doi.org/10.1007/978-3-642-21554-4_19

[4] Bansod, B., Patil, A. and Pisharoty, N. (2018) Granule: An Ultra Lightweight Cipher Design for Embedded Security. https://eprint.iacr.org/2018/600

[5] 石淑英, 何骏. GRANULE 算法的不可能差分分析[J]. 计算机工程, 2019, 45(10): 134-138.

[6] 武小年, 李迎新, 韦永壮, 等. GRANULE 和 MANTRA 算法的不可能差分区分器分析[J]. 通信学报, 2020, 41(1): 94-101.

[7] 赵晨曦. 轻量级分组密码的不可能差分分析[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2021: 41-51.

[8] 刘先蓓, 刘亚. GRANULE 算法的截断不可能差分分析[J]. 山西师范大学学报(自然科学版), 2023, 37(1): 41-51.

[9] 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

[10] Demirci, H. and Selçuk, A.A. (2008) A Meet-In-The-Middle Attack on 8-Round AES. In: Nyberg, K., Ed., *Fast Software Encryption*, Springer, 116-126. https://doi.org/10.1007/978-3-540-71039-4_7

[11] 李超, 孙兵, 李瑞林. 分组密码的攻击方法与实例分析[M]. 北京: 科学出版社, 2010.

[12] 任炯炯, 侯泽洲, 李曼曼, 等. 改进的减轮 MIBS-80 密码的中间相遇攻击[J]. 电子与信息学报, 2022, 44(8): 2914-2923.

[13] 刘亚, 刁倩倩, 李玮, 等. 10 轮 Midori128 的中间相遇攻击[J]. 计算机应用研究, 2019, 36(1): 230-234, 238.

[14] 郑雅菲, 吴文玲. LBlock 算法的改进中间相遇攻击[J]. 计算机学报, 2017, 40(5): 1080-1091.