软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(5):2274−2285 [doi: 10.13328/j.cnki.jos.007457] [CSTR: 32375.14.jos.007457]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 对缩减轮 SPECK 改进的差分-线性分析*

张语晗[1,2], 张 蕾[1,2], 吴文玲[1,2]

[1](中国科学院 软件研究所 可信计算与信息保障实验室, 北京 100190)

[2](中国科学院大学, 北京 100049)

通信作者: 吴文玲, E-mail: wenling@iscas.ac.cn

**摘 要**: 差分-线性分析是一种组合类分析方法, 已经被应用于许多对称密码的分析中. 特别地, 对于 ARX 类分组密码算法 SPECK, 差分-线性分析是评估其安全性的一种强有力的方式. 在最新的差分-线性分析框架中, 密码算法被分解为 3 部分: 差分部分、中间部分和线性部分, 其中差分部分、中间部分和线性部分分别包含高概率的差分特征, 高相关性的差分-线性逼近和高相关性的线性逼近, 组合 3 部分特征可以得到一个完整的差分-线性区分器. 对于 ARX 类对称密码算法, 在传统的差分-线性区分器的搜索过程中, 通常是首先借助实验方法来计算得到中间部分一个高相关性的差分-线性逼近, 然后再分别向前向后搜索线性特征和差分特征, 但是该策略容易忽视掉一些好的差分-线性区分器. 区别于传统的搜索算法, 该算法结合高相关性的差分-线性逼近中差分部分和线性部分的特点, 从高概率的差分特征和线性特征出发, 给出一个差分-线性区分器搜索算法. 将所提搜索算法应用于 SPECK 中, 得到 SPECK32 的 11 轮差分-线性区分器和 SPECK48 的 12 轮差分-线性区分器. 所提区分器都优于 SPECK32/48 目前已知最好的差分-线性区分器.

**关键词**: 密码分析; 对称密码; 差分-线性分析; SPECK; 分组密码

**中图法分类号**: TP309

中文引用格式: 张语晗, 张蕾, 吴文玲. 对缩减轮SPECK改进的差分-线性分析. 软件学报, 2026, 37(5): 2274–2285. http://www.jos.org.cn/1000-9825/7457.htm

英文引用格式: Zhang YH, Zhang L, Wu WL. Improved Differential-linear Analysis on Round-reduced SPECK. Ruan Jian Xue Bao/Journal of Software, 2026, 37(5): 2274–2285 (in Chinese). http://www.jos.org.cn/1000-9825/7457.htm

## Improved Differential-linear Analysis on Round-reduced SPECK

ZHANG Yu-Han[1,2], ZHANG Lei[1,2], WU Wen-Ling[1,2]

[1](Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

**Abstract**: Differential-linear cryptanalysis, a combined cryptanalysis method, has been applied to the analysis of many symmetric ciphers. Specifically, for the ARX block cipher SPECK, differential-linear cryptanalysis is an effective technique for evaluating its security. In the latest framework of differential-linear cryptanalysis, the cipher is divided into three components: the differential part, the middle part, and the linear part. These parts contain high-probability differential characteristics, high-correlation differential-linear approximations, and high-correlation linear approximations, respectively. For ARX ciphers, the traditional search process for differential-linear distinguishers typically involves first using experimental methods to obtain a high-correlation differential-linear approximation in the middle part. Subsequently, linear and differential characteristics are searched for forward and backward. However, this strategy may overlook some effective differential-linear distinguishers. This study proposes a search method for differential-linear distinguishers, which integrates the characteristics of the differential and linear parts in high-correlation differential-linear approximations and leverages high-probability differential and linear characteristics. The proposed search algorithm is applied to SPECK, yielding an 11-round differential-linear

* 基金项目: 国家自然科学基金 (62072445)
收稿时间: 2024-09-05; 修改时间: 2024-10-29; 采用时间: 2025-04-25; jos 在线出版时间: 2025-10-29
CNKI 网络首发时间: 2025-10-30

distinguisher for SPECK32 and a 12-round differential-linear distinguisher for SPECK48. Both outperform the best-known differential-linear distinguishers for these ciphers.

**Key words**: cryptanalysis; symmetric cipher; differential-linear cryptanalysis; SPECK; block cipher

ARX 结构的密码算法是对称密码算法中非常重要的一类密码算法. ARX 结构组成简单, 仅包含模加 (与/或)、循环移位和异或这 3 个基本操作; 易于实现, 在现在的大多数计算平台中, 如 X86、ARM、8-bit 微处理器等, 都支持模加 (与/或)、移位和异或操作, 仅需要较少的代码量就可以对基于 ARX 构造的算法进行实现; 性能优越, 无论是在软件平台还是硬件平台, 基于 ARX 构造的算法都表现良好. 结构简单、易于实现、软硬件性能优越等优点使得 ARX 结构的密码算法受到广大密码设计者的青睐. ARX 结构被广泛应用于分组密码的设计中, 如 TEA[1]、SPECK[2]、HIGHT[3]、LEA[4]、CHAM[5]、SPARX[6]等.

差分-线性分析是由 Langford 等人[7]提出的一种组合类分析方法. 假设算法 $E$ 可以被分解为 $E_0$ 和 $E_1$ 两部分, 即 $E = E_1 \circ E_0$, 组合 $E_0$ 部分一条高概率的差分和 $E_1$ 部分一条高相关性的线性逼近可以得到算法 $E$ 的一个有效的差分-线性区分器. 差分-线性分析被用于许多分组密码算法的攻击中, 特别地, 对 Serpent 和 ICEPOLE 等算法的最好的攻击是基于差分-线性分析的. 值得强调的是, Langford 等人的差分-线性分析框架依赖于两个假设: $E_0$ 和 $E_1$ 部分的独立性假设和 $E_0$ 的输出差分的随机性假设. 在后续的研究中, 许多学者指出上述两个假设在实际情况中并不一定是满足的. Biham 等人[8]指出 $E_0$ 输出差分的随机性假设在实际情况下并非一定满足. Blondeau 等人[9]进行深入研究, 给出了只满足 $E_0$ 和 $E_1$ 部分是独立的情况下差分-线性逼近相关性的精确计算公式.

关于 $E_0$ 和 $E_1$ 部分的依赖性问题, Bar-On 等人[10]将 boomerang 攻击中的 sandwich 框架的思想[11]应用于差分-线性分析框架中, 给出了 DLCT 框架来计算差分-线性逼近的相关性, 该框架考虑了 $E_0$ 和 $E_1$ 之间的依赖性. DLCT 框架基于差分线性连接表 (DLCT), 在该框架中密码算法 $E$ 可以被分解为 $E_0$、$E_m$、$E_1$ 这 3 部分, 即 $E = E_1 \circ E_m \circ E_0$. 组合 $E_0$ 部分一条高概率的差分, $E_m$ 部分一条高相关性的差分-线性逼近和 $E_1$ 部分一条高相关性的线性逼近可以得到算法 $E$ 的一个有效的差分-线性区分器. $E_m$ 部分差分-线性逼近的相关性借助 DLCT 表或者实验方法来计算得到.

ARX 类密码算法的差分-线性分析是密码分析的研究热点. 近些年来有许多相关的工作被提出. 最近, Bellini 等人[12]将混合整数线性规划 (MILP) 和混合整数二次约束规划 (MIQCP) 技术引入到 ARX 算法的差分-线性区分器搜索中并应用于 SPECK 算法. 但是该方法存在一些限制, 一方面是在差分-线性区分器搜索的建模中存在一些近似导致建模是粗糙的; 另一方面, 受求解效率的影响, 该方法只能应用于分组长度较小的算法中, 如 SPECK32. Chen 等人[13]给出了一个新的搜索 ARX 类密码算法差分-线性区分器的方法, 应用于 SPECK 和 LEA 中, 得到了许多更好的区分器.

本文贡献如下. 对 ARX 类分组密码算法的差分-线性区分器搜索算法进行研究. 首先, 通过计算 SPECK32/64 在不同随机密钥下的差分-线性逼近的实际相关性, 进一步展示了样本量的选择跟实验测量的相关性的大小之间的关系. 然后, 结合高相关性的差分-线性逼近中差分部分和线性部分的特点, 我们给出了一个改进的差分-线性区分器搜索算法. 区别于传统的差分-线性区分器搜索算法, 我们从高概率的差分特征和线性特征出发, 然后实验测量中间部分差分-线性逼近的相关性, 搜索到了一些新的区分器. 具体地, 将所提差分-线性区分器搜索算法应用于 SPECK 中, 得到了 SPECK32 的 11 轮差分-线性区分器和 SPECK48 的 12 轮差分-线性区分器 (表 1 展示了与已有差分-线性区分器的结果对比). 这些区分器都优于 SPECK 已知最好的差分-线性区分器.

表 1　差分-线性区分器结果对比

| 算法 | 轮数 | 相关性 | 参考文献 |
|---|---|---|---|
| SPECK32 | 10 | $2^{-12}$ | [12] |
| SPECK32 | 10 | $2^{-11.58}$ | [13] |
| SPECK32 | 11 | $2^{-15.78}$ | 第3.2节 |
| SPECK48 | 11 | $2^{-17.55}$ | [13] |
| SPECK48 | 12 | $2^{-22.96}$ | 第3.2节 |

第 1 节为预备知识, 主要介绍符号约定、相关性的定义及差分-线性分析的框架. 第 2 节介绍本文改进的差分-线性区分器搜索算法. 第 3 节介绍所提搜索算法在 SPECK32 和 SPECK48 上的应用. 第 4 节是本文总结.

## 1 预备知识

### 1.1 符号和定义

表 2 给出了本文中用到的符号及其定义.

表 2 符号定义

| 符号 | 描述 | 符号 | 描述 |
|---|---|---|---|
| $\oplus$ | XOR操作 | $\cdot$ | 向量内积 |
| $\boxplus/\boxminus$ | 模加/模减操作 | $\circ$ | 函数复合 |
| $[i]$ | 第 $i$ 个分量取值为1, 其他分量取值为0的单位向量 | $\lll$ | 循环右移位 |
| $[i_0,i_1,\ldots,i_l]$ | 第 $i_0,i_1,\ldots,i_l$ 个分量取值为1, 其他分量取值为0的向量 | $\ggg$ | 循环左移位 |
| $\lvert S\rvert$ | 集合 $S$ 中元素的个数 | $\cup$ | 集合求并集 |

**定义 1**. 给定集合 $S\subseteq F_2^n$ 和布尔函数 $f:F_2^n\to F_2$, 相关性被定义为:

$$Cor_{x\in S}(f):=\frac{1}{|S|}\sum_{x\in S}(-1)^{f(x)}.$$

### 1.2 差分-线性分析

Langford 等人[7]结合差分分析和线性分析, 提出了差分-线性分析, 接下来我们统称 Langford 等人的框架为原始的差分-线性分析框架. 在原始的差分-线性分析框架中, 密码算法 $E$ 被分解为 $E_0$ 和 $E_1$ 两部分, 即 $E=E_1\circ E_0$, 其中 $E_0$ 和 $E_1$ 部分的轮数分别为 $r_0$ 和 $r_1$. 在 $E_0$ 部分有概率为 $p$ 的差分 $\Delta_{\text{in}}\xrightarrow{E_0}\Delta_m$, 在 $E_1$ 部分有相关性为 $q$ 的线性逼近 $\lambda_{\text{in}}\xrightarrow{E_1}\lambda_{\text{out}}$, 如图 1 所示. 密码算法 $E$ 的 $r_0+r_1$ 轮的差分-线性逼近 $\Delta_{\text{in}}\xrightarrow{E}\lambda_{\text{out}}$ 的相关性被计算为:

$$Cor(\lambda_{\text{out}}\cdot E(x)\oplus\lambda_{\text{out}}\cdot E(x\oplus\Delta_{\text{in}}))=pq^2.$$

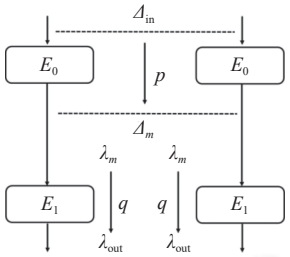



图 1 原始的差分-线性分析框架

原始的差分-线性分析框架依赖于如下的两个假设.

(1) $E_0$ 部分和 $E_1$ 部分之间是相互独立的.

(2) 当 $\Delta_X=E_0(x)\oplus E_0(x\oplus\Delta_{\text{in}})\neq\Delta_m$, $\lambda_{\text{out}}\cdot\Delta_X=0$ 成立的概率为 $\frac{1}{2}$.

然而, 后续的研究指出上述两个假设在实际中并不一定是满足的. Biham 等人[8]指出假设 (2) 在许多情况下是不满足的, 并给出了相关的实验验证. 之后, Blondeau 等人[9]给出了在不依赖于假设 (2) 的情况下的差分-线性逼近相关性的计算公式. 在假设 $E_0$ 部分和 $E_1$ 部分之间是相互独立的情况下, 差分-线性区分器的相关性被计算为:

$$Cor(\lambda_{\text{out}}\cdot E(x)\oplus\lambda_{\text{out}}\cdot E(x\oplus\Delta_{\text{in}}))=\sum_{\lambda_m}Cor(\lambda_m\cdot E_0(x)\oplus\lambda_m\cdot E_0(x\oplus\Delta_{\text{in}}))\cdot Cor(\lambda_m,\lambda_{\text{out}}).$$

关于 $E_0$ 部分和 $E_1$ 部分之间的依赖性问题很长一段时间内没有被解决. Boomerang 分析也是一种差分类的组合分析方法, 在 boomerang 分析中也存在类似的依赖性问题. Cid 等人[11]给出了 BCT 表 (boomerang 连接表) 来解

决 boomerang 区分器的两部分之间的依赖性问题. 受到 boomerang 分析中 BCT 表的启发, Bar-On 等人[10]给出了 DLCT 框架来解决差分-线性区分器 $E_0$ 部分和 $E_1$ 部分之间的依赖性问题, 接下来我们统称该框架为最新的差分-线性分析框架. 在最新的差分-线性分析框架中, 算法 $E$ 被分解为 $E_0$、$E_m$、$E_1$ 这 3 部分, 即 $E = E_1 \circ E_m \circ E_0$. 其中 $E_0$、$E_m$、$E_1$ 部分的轮数分别为 $r_0$、$r_m$、$r_1$. 在 $E_0$ 部分有概率为 $p$ 的差分 $\Delta_{\text{in}} \xrightarrow{E_0} \Delta_m$, 在 $E_m$ 部分有相关性为 $r$ 的差分-线性逼近 $\Delta_m \xrightarrow{E_m} \lambda_m$, 在 $E_1$ 部分有相关性为 $q$ 的线性逼近 $\lambda_{\text{in}} \xrightarrow{E_1} \lambda_{\text{out}}$, 如图 2 所示.

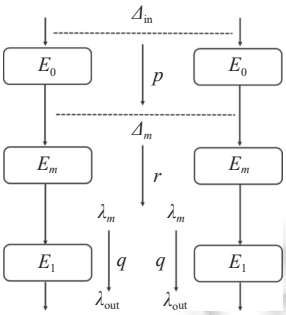



图 2 最新的差分-线性分析框架

当 $\Delta_m$ 和 $\lambda_m$ 是一个好的选择时, 我们可以通过如下的公式来得到差分-线性逼近 $\Delta_{\text{in}} \xrightarrow{E} \lambda_{\text{out}}$ 相关性的一个好的估计.

$$Cor(\lambda_{\text{out}} \cdot E(x) \oplus \lambda_{\text{out}} \cdot E(x \oplus \Delta_{\text{in}})) = prq^2.$$

精确的相关性被计算为:

$$Cor(\lambda_{\text{out}} \cdot E(x) \oplus \lambda_{\text{out}} \cdot E(x \oplus \Delta_{\text{in}})) = \sum_{\Delta_m, \lambda_m} \Pr(\Delta_{\text{in}}, \Delta_m) \cdot Cor(\lambda_m \cdot E_m(x) \oplus \lambda_m \cdot E_m(x \oplus \Delta_m)) \cdot Cor(\lambda_m, \lambda_{\text{out}}).$$

本文的分析都是基于最新的差分-线性分析框架的.

## 2 改进的差分-线性区分器搜索

在最新的差分-线性分析框架中, 当 $\Delta_m$ 和 $\lambda_m$ 是一个好的选择时, 差分-线性逼近 $\Delta_{\text{in}} \xrightarrow{E} \lambda_{\text{out}}$ 的相关性被估计为 $prq^2$. $E_0$ 部分的差分概率 $p$ 和 $E_1$ 部分的线性相关性 $q$ 基于现有的差分分析技术和线性分析技术就可以求解得到. 然而, 对于 ARX 类密码算法, 关于 $E_m$ 的差分-线性相关性 $r$ 的计算常常是借助实验方法来计算得到的. 但是, 当通过实验方法来计算 $r$ 的值时存在一些未解释的现象[13]. 在本节中, 我们深入研究了 SPECK32/64 的差分-线性分析, 对这些现象给出了一些说明. 然后, 我们给出了一个改进的差分-线性区分器搜索算法.

### 2.1 解释说明

在文献 [13] 中, Chen 等人借助实验方法来估计 $E_m$ 部分差分-线性相关性 $r$ 的值. 通过多次的实验我们发现当借助实验方法来估计 $r$ 的值时, 存在一些未解释的现象, 接下来我们主要对如下两个未解释的现象展开讨论, 并给出解释说明.

(1) 以 SPECK32/64 为例, 为什么 $E_m$ 部分的轮数 $r_m$ 的选取不超过 6 轮?

(2) 当采用实验方法估计 $r$ 的值时, 样本集合的大小是否会影响 $r$ 的估计?

在最新的差分-线性分析框架中, $E_m$ 部分的主要作用是解决 $E_0$ 部分和 $E_1$ 部分之间的依赖性问题, 当 $E_m$ 部分的轮数 $r_m$ 较小时, 很有可能是无法保证 $E_0$ 部分和 $E_1$ 部分是相互独立的, 从这个角度来看 $E_m$ 部分的轮数 $r_m$ 的选取越大越好. 但是在文献 [13] 中, 对轮数 $r_m$ 的选取似乎是有一些限制的, 例如对于 SPECK32/64, $r_m$ 选取的最大值为 6, 但是为什么这么选取作者并没有给出详细的解释.

根据统计理论, 当使用实验方法估计中间部分的相关性时, 样本量的选择跟实验测量的相关性的大小之间存在着紧密的关系. 例如, 在文献 [14−16] 中样本量的选择都满足如下的情况: 如果使用 $2^n$ 个随机数据来实验测量相关性, 那么只能较准确地测量到大于 $c \cdot 2^{-\frac{n}{2}}$ ($c \approx 10$) 的偏差.

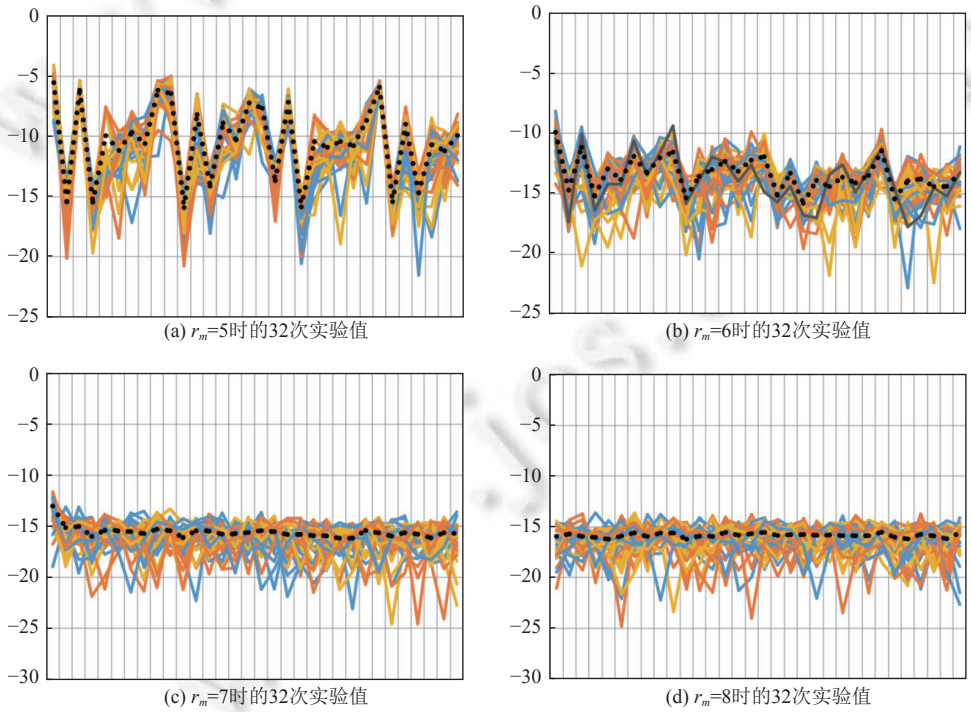
在本节中, 我们遍历了 SPECK32/64 的 32 比特的明文空间, 测试了 32 个随机密钥下 SPECK32/64 实际的差分-线性逼近的相关性. 具体地, 我们测试了当轮数 $r_m = 5,6,7,8$ 时, 不同密钥下 SPECK32/64 如下形式的差分-线性逼近的相关性.

$$[j] \xrightarrow{r_m} [i],\ i,j = 0,1,\ldots,31.$$

例如, 当 $[j] = [0]$ 时, 差分-线性逼近 $\Delta_m \xrightarrow{E_m} \lambda_m$ 在 32 个随机密钥下的平均相关性如表 3 所示, 32 次实验值如图 3 所示 (黑色虚线表示平均相关性的取值).

表 3 $r_m = 5,6,7,8$ 时, $E_m$ 部分差分-线性逼近的相关性 $\log_2 Cor(\Delta_m \xrightarrow{E_m} \lambda_m)$ 在 32 个随机密钥下的平均相关性

| $[i]$ | $r_m=5$ | $r_m=6$ | $r_m=7$ | $r_m=8$ | $[i]$ | $r_m=5$ | $r_m=6$ | $r_m=7$ | $r_m=8$ |
|---|---|---|---|---|---|---|---|---|---|
| [0] | −5.5394 | −9.9000 | −12.99689894 | −15.9452 | [16] | −7.6663 | −11.9086 | −15.38938581 | −15.5435 |
| [1] | −15.4292 | −14.6925 | −15.10391992 | −15.7123 | [17] | −13.6780 | −15.1131 | −15.49581395 | −15.8619 |
| [2] | −6.1737 | −11.1484 | −14.99355333 | −15.9597 | [18] | −7.1606 | −13.2943 | −15.7799793 | −15.7807 |
| [3] | −15.4615 | −15.1770 | −15.95371775 | −16.0540 | [19] | −15.8611 | −15.7884 | −15.76208401 | −15.7593 |
| [4] | −9.9501 | −12.9326 | −15.37358988 | −16.2166 | [20] | −10.4416 | −13.8920 | −15.88734698 | −15.8400 |
| [5] | −11.1612 | −13.8246 | −15.42270327 | −15.9114 | [21] | −10.9130 | −14.3982 | −15.91515767 | −15.7922 |
| [6] | −9.6737 | −11.8934 | −15.48989365 | −15.5953 | [22] | −10.4377 | −13.6205 | −16.13804963 | −15.8754 |
| [7] | −10.6030 | −13.2615 | −15.69323659 | −15.9676 | [23] | −10.7732 | −14.0968 | −16.13115267 | −15.8874 |
| [8] | −6.1734 | −12.0252 | −15.25886998 | −15.6205 | [24] | −8.6727 | −13.4747 | −15.38750188 | −15.8829 |
| [9] | −6.4455 | −11.5373 | −15.40585711 | −15.6163 | [25] | −5.9334 | −11.4262 | −15.62241014 | −15.5418 |
| [10] | −15.9226 | −15.3312 | −16.07323483 | −16.2170 | [26] | −15.4266 | −15.4253 | −15.92693883 | −16.0341 |
| [11] | −8.1954 | −13.5437 | −15.40519983 | −15.9190 | [27] | −8.9989 | −13.9208 | −15.60298064 | −16.1945 |
| [12] | −14.1290 | −12.9866 | −15.45124118 | −15.9499 | [28] | −14.6921 | −13.7726 | −15.73775607 | −15.7497 |
| [13] | −8.8852 | −12.3095 | −15.73757125 | −15.5523 | [29] | −10.6924 | −14.3927 | −16.09294605 | −15.9012 |
| [14] | −10.2927 | −13.1264 | −15.67638158 | −15.8337 | [30] | −11.2060 | −14.3788 | −15.61748001 | −16.1772 |
| [15] | −7.2194 | −12.2109 | −15.58155542 | −15.6851 | [31] | −9.8963 | −13.2542 | −15.68419689 | −15.6044 |




(a) $r_m$=5时的32次实验值　(b) $r_m$=6时的32次实验值

(c) $r_m$=7时的32次实验值　(d) $r_m$=8时的32次实验值

图 3 $r_m = 5,6,7,8$ 时, $E_m$ 部分差分-线性逼近的相关性 $\log_2 Cor(\Delta_m \xrightarrow{E_m} \lambda_m)$ 的 32 次实验值

根据实验结果我们得到了如下的观察结果.

● 观察 1. 对于 SPECK32/64, $E_m$ 部分的轮数 $r_m$ 的选取不超过 6 轮是因为当轮数不超过 6 轮时, 有明显大于 $2^{-16}$ 的相关性, 并且不同密钥下的相关性的值都大于 $2^{-16}$.

当利用实验方法计算差分-线性逼近的相关性时, 因为计算资源有限, 一般采用一个小的样本集合, 如 $2^{22}$–$2^{30}$. 通过实验我们发现, 如果选择的样本集合的大小为 $2^{22}$ ($2^{30}$) 时, 在实验计算得到的相关性的值趋近于 $2^{-11}$ ($2^{-15}$) 时, 实际的相关性的值其实已经很接近随机情况了. 这也侧面反映了当采用实验方法计算 $E_m$ 部分的差分-线性逼近的相关性的值时, 对于 $E_m$ 部分差分-线性逼近的选择, 应该选择相关性的值尽可能大的差分-线性逼近. 但是, 相关性的值越大, 也就意味着轮数 $r_m$ 的选择不能太大, 这也将会影响 $E_0$ 部分和 $E_1$ 部分的依赖性. 因此, 当选择轮数 $r_m$ 的值的时候, 一方面应该选择尽可能长的轮数来保证 $E_0$ 部分和 $E_1$ 部分的独立性; 另一方面, 不能选择太长的轮数来保证实验方法计算的相关性的值的准确性. 基于上述实验结果, 我们得到了如下的观察.

● 观察 2. 当通过实验方法计算 $E_m$ 部分差分-线性逼近的相关性的值时, 为了得到更加接近实际值的相关性估计, 在选择 $E_m$ 部分差分-线性逼近的时候, 我们应该尽可能选择相关性的取值远大于 $2^{-\frac{N}{2}}$ 的差分-线性逼近, 其中样本集合的大小为 $2^N$.

在本节中, 我们参考文献 [14−16] 并通过遍历 SPECK32/64 的 32 比特明文空间, 测试差分-线性逼近的实际相关性. 随后, 对文献 [13] 中未解释的现象给出了一些说明.

### 2.2 差分-线性区分器搜索算法

目前已有的差分-线性区分器搜索算法往往先从搜索相关性高的中间部分的差分-线性逼近出发, 然后再分别向后和向前搜索差分部分的特征和线性部分的特征, 基于该策略搜索到的差分-线性特征的差分部分和线性部分的特征往往不是最优或者次优的, 容易错过一些高相关性的差分-线性区分器. 在本节中我们首先从高概率且输出差分汉明重量低的差分特征和高相关性且输入掩码汉明重量低的线性特征出发, 然后再测量中间部分的相关性, 得到了一些新的差分-线性区分器.

在文献 [13] 中, 作者给出了一个基于 Meet-in-the-Middle 思想的搜索方法来搜索 ARX 类密码算法的差分-线性区分器, 得到了 SPECK 和 LEA 算法概率更高轮数更长的差分-线性区分器. 在本节中, 我们借鉴了该 Meet-in-the-Middle 的搜索思想, 给出了一个改进的差分-线性区分器搜索算法. 我们的搜索算法考虑了差分概率 $p$, 差分-线性相关性 $r$ 和线性相关性 $q$ 三者的折中. 首先, 给出如下两个观察结果.

● 观察 3. 在差分-线性区分器搜索中, 当差分 (线性) 部分的特征为次优、次次优特征时更容易得到高相关性的差分-线性区分器.

● 观察 4. 在差分-线性区分器搜索中, $\Delta_m$ 中的差分活跃比特个数较少且 $\lambda_m$ 中的线性活跃比特数较少时更容易得到高相关性的差分-线性区分器.

基于观察 3 和观察 4, 我们通过如下的步骤来搜索密码算法 $E$ 的 $r_{DL}$ 轮差分-线性区分器.

(1) 首先, 确定 $E_0$、$E_m$ 和 $E_1$ 部分的轮数 $r_0$、$r_m$ 和 $r_1$.

(2) 然后, 搜索得到密码算法 $E$ 的 $r_0$ 轮最优、次优、次次优差分特征的输出差分集合 $\mathcal{S}_{\Delta}^{r_0} = \bigcup_p \bigcup_b S_{\Delta}^{r_0}(p,b)$, 其中 $S_{\Delta}^{r_0}(p,b)$ 表示概率为 $p$, 输出差分的活跃比特个数为 $b$ 的 $r_0$ 轮差分特征的输出差分的集合.

(3) 接着, 搜索得到密码算法 $E$ 的 $r_1$ 轮最优、次优、次次优线性特征的输入线性掩码集合 $\mathcal{S}_{\Lambda}^{r_1} = \bigcup_q \bigcup_b S_{\Lambda}^{r_1}(q,b)$, 其中 $S_{\Lambda}^{r_1}(q,b)$ 表示相关性为 $q$, 输入线性掩码的活跃比特个数为 $b$ 的 $r_1$ 轮线性特征的输入线性掩码的集合.

(4) 从集合 $\mathcal{S}_{\Delta}^{r_0}$ 中挑选差分概率高且活跃比特个数少的差分作为 $\Delta_m$ 的候选, 从集合 $\mathcal{S}_{\Lambda}^{r_1}$ 中选择相关性高且活跃比特个数少的线性掩码作为 $\lambda_m$ 的候选, 并计算差分-线性逼近 $\Delta_m \xrightarrow{E_m} \lambda_m$ 的相关性 $r$.

(5) $r_{DL} = r_0 + r_m + r_1$ 轮的差分-线性逼近的相关性被计算为 $cor = prq^2$.

(6) 重复步骤 (1)–(5), 直到找到一个高相关性的差分-线性逼近.

注意到, 对于步骤 (2) 和 (3) 可以通过预计算得到:

$$\begin{cases} S_{\Delta}^{r_0}(p,b), & p>0, b>0, r_0>0 \\ S_{\Lambda}^{r_1}(q,b), & q>0, b>0, r_1>0 \end{cases}$$

从而对搜索过程进行加速. 具体的搜索过程如算法 1 所示.

**算法 1**. 改进的差分-线性区分器搜索算法.

输入: 轮数 $r_{DL}$;

输出: 高相关性的 $r_{DL}$ 轮差分-线性逼近.

$cor \leftarrow 0$;
WHILE TRUE:
  确定 $r_0, r_m, r_1$ 的值使得 $r_0 + r_m + r_1 = r_{DL}$;
  FOR $b_{\Delta}$ in range $(1, B)$: /*$B$ 是预先设置的活跃比特个数的阈值*/
    FOR $b_{\Lambda}$ in range $(1, B)$:
      FOR $k_{\Delta}$ in range $(1, K)$: /*$K$ 是预先设置的与差分概率和线性相关性有关的阈值*/
        $p = 2^{-k_{\Delta}}$;
        $\Delta_m \leftarrow S_{\Delta}^{r_0}(p, b_{\Delta})$; /* $S_{\Delta}^{r_0}(p, b_{\Delta})$ 可以通过预计算得到*/
        FOR $k_{\Lambda}$ in range $(1, K)$:
          $q = 2^{-k_{\Lambda}}$;
          $\lambda_m \leftarrow S_{\Lambda}^{r_1}(q, b_{\Lambda})$; /* $S_{\Lambda}^{r_1}(q, b_{\Lambda})$ 可以通过预计算得到*/
          $r = Cor(\Delta_m \xrightarrow{r_m} \lambda_m)$;
          $\overline{cor} \leftarrow prq^2$;
          IF $\overline{cor} > cor$:
            $cor = \overline{cor}$;
          END IF
        END FOR
      END FOR
    END FOR
  END FOR
END WHILE

## 3 应用于 SPECK

### 3.1 SPECK 算法简介

SPECK 是 NSA 在 2013 年设计的一族轻量级分组密码算法[2], SPECK 的参数如表 4 所示.

表 4 SPECK 的参数

| 版本 | 分组长度$2n$ | 字大小$n$ | 密钥长度$mn$ | 密钥字个数$m$ | 轮数 | $a$ | $b$ | 版本 | 分组长度$2n$ | 字大小$n$ | 密钥长度$mn$ | 密钥字个数$m$ | 轮数 | $a$ | $b$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPECK32/64 | 32 | 16 | 64 | 4 | 22 | 7 | 2 | SPECK96/96 | 96 | 48 | 96 | 2 | 28 | 8 | 3 |
| SPECK48/72 | 48 | 24 | 72 | 3 | 22 | 8 | 3 | SPECK96/144 | 96 | 48 | 144 | 3 | 29 | 8 | 3 |
| SPECK48/96 | 48 | 24 | 96 | 4 | 23 | 8 | 3 | SPECK128/128 | 128 | 64 | 128 | 2 | 32 | 8 | 3 |
| SPECK64/96 | 64 | 32 | 96 | 3 | 26 | 8 | 3 | SPECK128/192 | 128 | 64 | 192 | 3 | 33 | 8 | 3 |
| SPECK64/128 | 64 | 32 | 128 | 4 | 27 | 8 | 3 | SPECK128/256 | 128 | 64 | 256 | 4 | 34 | 8 | 3 |

SPECK2$n$ 的轮函数 $E_k: F_2^n \times F_2^n \to F_2^n \times F_2^n$ 被定义为:

$$E_k(x,y) = ((x \ggg a \boxplus y) \oplus k, y \lll b \oplus (x \ggg a \boxplus y) \oplus k.$$

SPECK2$n$ 的密钥编排复用轮函数来生成轮密钥. 令 $K = (l_{m-2}, \ldots, l_0, k_0)$, $l_i, k_i \in F_2^n$, $m \in \{2,3,4\}$ 是 SPECK2$n$ 的一个密钥, $l_i, k$ 满足:

$$\begin{cases} l_{i+m-1} = (k_i \boxplus l_i \ggg a) \oplus i \\ k_{i+1} = k_i \lll b \oplus l_{i+m-1} \end{cases}.$$

SPECK2$n$ 的轮函数和密钥编排的分解图如图 4 所示.

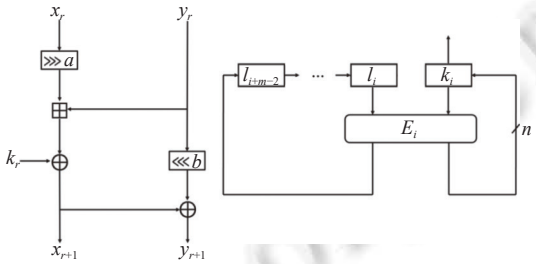



图 4 SPECK2$n$ 的轮函数和密钥编排

### 3.2 应 用

我们将算法 1 应用于 SPECK 算法中, 首次得到了 SPECK32 的 11 轮差分-线性区分器和 SPECK48 的 12 轮差分-线性区分器.

为了能够在较短的时间内返回 SPECK 的好的差分-线性区分器. 参考 Chen 等人[13]在 ASIACRYPT 2023 中的测试方法, 对于中间部分相关性的测量, 我们首先选择 $2^{26}$ 大小的样本量来测试, 然后当算法 1 返回一个好的差分-线性特征时, 我们再使用 $2^{30}$ 大小的样本量和 100 个随机密钥来进一步确认中间部分相关性的准确性, 基于该策略我们能够快速地搜索到 SPECK 新的差分-线性区分器. 事实上, 在 SPECK32 的 11 轮差分-线性区分器和 SPECK48 的 12 轮差分-线性区分器中中间部分的相关性均是不小于 $2^{-10}$ 的. 基于统计理论, 无论是 $2^{26}$ 大小的样本量还是 $2^{30}$ 大小的样本量都能够得到较准确的测量值.

#### 3.2.1 应用于 SPECK32

我们首先预计算了 SPECK32 的 2 轮、3 轮和 4 轮最优、次优、次次优差分特征的输出差分集合 $\mathcal{S}_\Delta^2$、$\mathcal{S}_\Delta^3$、$\mathcal{S}_\Delta^4$ 和 SPECK32 的 2 轮、3 轮和 4 轮最优、次优、次次优线性特征的输入线性掩码集合 $\mathcal{S}_\Lambda^2$、$\mathcal{S}_\Lambda^3$、$\mathcal{S}_\Lambda^4$. 集合 $S_\Delta^{r_0}(p,b)$ 表示概率为 $p$, 输出差分的活跃比特个数为 $b$ 的 $r_0$ 轮差分特征的输出差分的集合, $S_\Lambda^{r_1}(q,b)$ 表示相关性为 $q$, 输入线性掩码的活跃比特个数为 $b$ 的 $r_1$ 轮线性特征的输入线性掩码的集合. 我们使用基于 SAT 的自动化搜索方法来获得集合 $S_\Delta^{r_0}(p,b)$ 和集合 $S_\Lambda^{r_1}(q,b)$. 表 5 中列出了 $\mathcal{S}_\Delta^2$ 的部分元素. 本文中, 在计算 $E_m$ 部分的差分-线性逼近的相关性时, 我们使用 $2^{30}$ 大小的样本集合.

表 5 SPECK32 的 $\mathcal{S}_\Delta^2$ 的部分元素

| $b$的取值 | $p = 2^{-1}$ | $p = 2^{-2}$ | $p = 2^{-3}$ |
|---|---|---|---|
| $b = 1$ | — | — | $S_\Delta^2(2^{-3},1) = \{[3,10,1,2]\}$ |
| $b = 2$ | — | $S_\Delta^2(2^{-2},2) = \{[8,24],[15,31],[10,26]\}$ | $S_\Delta^2(2^{-3},2) = \{[1,17],\ldots,[3,19]\}$ |
| $b = 3$ | $S_\Delta^2(2^{-1},3) = \{[1,15,31],[1,3,17]\}$ | $S_\Delta^2(2^{-2},3) = \{[1,15,31],\ldots,[10,12,26]\}$ | $S_\Delta^2(2^{-3},3) = \{[1,15,31],\ldots,[10,12,26]\}$ |

表 6 中列出了 $\mathcal{S}_\Lambda^4$ 的部分元素.

基于算法 1, 结合预计算得到 SPECK32 的 2 轮、3 轮和 4 轮最优、次优、次次优差分特征的输出差分集合

$\mathcal{S}_{\Delta}^{2}$、$\mathcal{S}_{\Delta}^{3}$、$\mathcal{S}_{\Delta}^{4}$ 和 SPECK32 的 2 轮、3 轮和 4 轮最优、次优、次次优线性特征的输入线性掩码集合 $\mathcal{S}_{\Lambda}^{2}$、$\mathcal{S}_{\Lambda}^{3}$、$\mathcal{S}_{\Lambda}^{4}$, 我们首次搜索得到了 SPECK32 的 11 轮差分-线性区分器. 11 轮区分器的具体细节如下所示:

$$\begin{cases} r_0=3,\ r_m=4,\ r_1=4 \\ \Delta_m=[10,26],\ \lambda_m=[3] \\ p=\Pr(\Delta_{\text{in}} \xrightarrow{r_0=3} \Delta_m)=2^{-6} \\ r=Cor(\Delta_m \xrightarrow{r_m=4} \lambda_m)=2^{-1.78} \\ q=Cor(\lambda_m \xrightarrow{r_1=4} \lambda_{\text{out}})=2^{-4} \\ Cor(\Delta_{\text{in}} \xrightarrow{r_{DL}=11} \lambda_{\text{out}})=2^{-15.79} \end{cases}.$$

最终我们得到了如下的 2 个 SPECK32 的 11 轮区分器:

$$\begin{cases} D_1:(8024,8500) \xrightarrow{r_{DL}=11} (387\text{a},3864) \\ D_2:(8024,8500) \xrightarrow{r_{DL}=11} (3854,3844) \end{cases}.$$

表 6 SPECK32 的 $\mathcal{S}_{\Lambda}^{4}$ 的部分元素

| $b$的取值 | $q=2^{-3}$ | $q=2^{-4}$ |
|---|---|---|
| $b=1$ | — | $S_{\Lambda}^{4}(2^{-4},1)=\{[3]\}$ |
| $b=2$ | — | — |
| $b=3$ | — | $S_{\Lambda}^{4}(2^{-4},3)=\{[0,14,23],\ldots,[0,15,23]\}$ |
| $b\geqslant 4$ | $S_{\Lambda}^{4}(2^{-3},1)=\{[0, 5, 13, 14, 20, 23],\ldots,$ [0, 4, 5, 12, 14, 23, 28, 29]$\}$ | $S_{\Lambda}^{4}(2^{-4},\geqslant 4)=\{[10, 11, 12, 19],\ldots,$ [0, 3, 6, 7, 10, 11, 12, 13, 14, 19, 20, 23, 30]$\}$ |

● SPECK32/64 的 13 轮密钥恢复: 我们以区分器 $D_1$ 下的密钥恢复攻击为例进行介绍, 其他区分器下的攻击过程是类似的. 在 SPECK32 的 11 轮差分-线性区分器 $D_1$ 后添加 2 轮, 我们给出了 SPECK32/64 的 13 轮密钥恢复攻击. 令 $x^r=(x_{15}^r,x_{14}^r,\ldots,x_0^r)$, $y^r=(y_{15}^r,y_{14}^r,\ldots,y_0^r)$ 表示 SPECK32/64 第 $r$ 轮的输入, $k^r=(k_{15}^r,k_{14}^r,\ldots,k_0^r)$ 表示第 $r$ 轮的轮密钥, $x^{r+1}=(x_{15}^{r+1},x_{14}^{r+1},\ldots,x_0^{r+1})$, $y^{r+1}=(y_{15}^{r+1},y_{14}^{r+1},\ldots,y_0^{r+1})$ 表示 SPECK32/64 第 $r$ 轮的输出. 根据 SPECK32/64 的轮函数有:

$$\begin{cases} y^r=(x^{r+1}\oplus y^{r+1})\ggg 2 \\ x^r=((x^{r+1}\oplus k^r)\boxminus y^r)\lll 7 \end{cases}.$$

令 $t^r=(x^{r+1}\oplus k^r)\boxminus y^r$, 根据模加操作的定义可以得到:

$$\begin{cases} t_i^r=f_i^r(x_i^{r+1},x_{i-1}^{r+1},\ldots,x_0^{r+1},k_i^r,k_{i-1}^r,\ldots,k_0^r,y_i^r,y_{i-1}^r,\ldots,y_0^r) \\ x_i^r=t_{(i-7)\%16}^r \\ y_i^r=x_{(i+2)\%16}^{r+1}\oplus y_{(i+2)\%16}^{r+1} \end{cases},$$

其中, $f_i^r$ 是关于 $x_i^{r+1},x_{i-1}^{r+1},\ldots,x_0^{r+1},k_i^r,k_{i-1}^r,\ldots,k_0^r,y_i^r,y_{i-1}^r,\ldots,y_0^r$ 的非线性函数. 将 $i=2,5,6,11,12,13,18,19,20,21,22,27,28,29$ 代入 $x_i^{12}$, 可知对 SPECK32/64 实施 13 轮的密钥恢复攻击需要猜测 $k^{12}$和$k^{13}$ 共 32 个密钥比特. 此时攻击需要的数据复杂度为 $2^{15.78\times 2}=2^{31.56}$, 需要的时间复杂度为 $2^{31.56+32}=2^{63.56}$.

3.2.2 应用于 SPECK48

与 SPECK32 的差分-线性区分器搜索过程类似, 我们首先预计算了 SPECK48 的 2 轮、3 轮和 4 轮最优、次优、次次优差分特征的输出差分集合 $\mathcal{S}_{\Delta}^{2}$、$\mathcal{S}_{\Delta}^{3}$、$\mathcal{S}_{\Delta}^{4}$ 和 SPECK48 的 2 轮、3 轮和 4 轮最优、次优、次次优线性特征的输入线性掩码集合 $\mathcal{S}_{\Lambda}^{2}$、$\mathcal{S}_{\Lambda}^{3}$、$\mathcal{S}_{\Lambda}^{4}$. 然后利用算法 1 来搜索 SPECK48 的差分-线性区分器. 表 7 中列出了 $\mathcal{S}_{\Delta}^{4}$ 的部分元素. 本文中, 在计算 $E_m$ 部分的差分-线性逼近的相关性时, 我们使用 $2^{30}$ 大小的样本集合.

表 7 SPECK48的 $\mathcal{S}_{\Delta}^{4}$ 的部分元素

| $b$的取值 | $p=2^{-6}$ | $p=2^{-7}$ | $p=2^{-8}$ | $p=2^{-9}$ |
|---|---|---|---|---|
| $b=3$ | — | — | $S_{\Delta}^{4}(2^{-8},3)=\{[0,21,45]\}$ | $S_{\Delta}^{4}(2^{-9},3)=\{[2,23,47],[13,16,37],[0,21,45],[8,11,32],[10,13,34],[16,19,40],[2,5,26],[18,21,42],[5,8,29]\}$ |
| $b=4$ | — | — | — | — |
| $b=5$ | — | $S_{\Delta}^{4}(2^{-7},5)=\{[2,15,23,39,47],[7,10,23,31,47]\}$ | $S_{\Delta}^{4}(2^{-8},5)=\{[0,13,21,37,45],\ldots,[7,15,18,31,39]\}$ | $S_{\Delta}^{4}(2^{-9},5)=\{[6,14,17,30,38],\ldots,[3,6,19,27,43]\}$ |
| $b=6$ | $S_{\Delta}^{4}(2^{-6},6)=\{[5,15,23,26,39,47]\}$ | $S_{\Delta}^{4}(2^{-7},6)=\{[0,10,18,34,42,45],\ldots,[2,12,20,36,44,47]\}$ | $S_{\Delta}^{4}(2^{-8},6)=\{[1,11,19,35,43,46],\ldots,[3,9,19,27,30,43]\}$ | $S_{\Delta}^{4}(2^{-9},6)=\{[6,12,22,30,33,46],\ldots,[5,11,21,29,32,45]\}$ |
| $b=7$ | $S_{\Delta}^{4}(2^{-6},7)=\{[5,7,18,23,26,31,47]\}$ | $S_{\Delta}^{4}(2^{-7},7)=\{[2,7,13,15,31,34,39],\ldots,[10,15,21,23,39,42,47]\}$ | $S_{\Delta}^{4}(2^{-8},7)=\{[5,7,18,23,26,31,47],\ldots,[2,4,15,20,28,44,47]\}$ | $S_{\Delta}^{4}(2^{-9},7)=\{[9,14,20,22,38,41,46],\ldots,[1,7,9,20,25,28,33]\}$ |

表 8 中列出了 $\mathcal{S}_{\Lambda}^{3}$ 的部分元素.

表 8 SPECK48的 $\mathcal{S}_{\Lambda}^{3}$ 的部分元素

| $b$的取值 | $q=2^{-1}$ | $q=2^{-2}$ | $q=2^{-3}$ |
|---|---|---|---|
| $b=1$ | — | — | $S_{\Lambda}^{3}(2^{-3},1)=\{[5,10],[2,3]\}$ |
| $b=2$ | — | — | $S_{\Lambda}^{3}(2^{-3},2)=\{[9,10],\ldots,[16,24]\}$ |
| $b=3$ | $S_{\Lambda}^{3}(2^{-1},3)=\{[0,21,32]\}$ | $S_{\Lambda}^{3}(2^{-2},3)=\{[0,21,32],[1,22,33]\}$ | $S_{\Lambda}^{3}(2^{-3},3)=\{[0,21,32],\ldots,[15,16,24]\}$ |
| $b=4$ | $S_{\Lambda}^{3}(2^{-1},4)=\{[2,4,5,37],[2,5,36,37]\}$ | $S_{\Lambda}^{3}(2^{-2},4)=\{[5,7,8,40],\ldots,[4,5,8,40]\}$ | $S_{\Lambda}^{3}(2^{-3},4)=\{[7,8,10,42],\ldots,[7,10,41,42]\}$ |

基于算法 1, 我们首次搜索到了 SPECK48 的两条 12 轮差分-线性区分器, 两条 12 轮差分-线性区分器的具体细节如下所示.

● SPECK48 的 12 轮差分-线性区分器 1:

$$\begin{cases} r_0=4,\ r_m=5,\ r_1=3 \\ \Delta_m=[0,21,45],\ \lambda_m=[5] \\ p=\Pr(\Delta_{\text{in}}\xrightarrow{r_0=4}\Delta_m)=2^{-8} \\ r=Cor(\Delta_m\xrightarrow{r_m=5}\lambda_m)=2^{-9.73} \\ q=Cor(\lambda_m\xrightarrow{r_1=3}\lambda_{\text{out}})=2^{-3} \\ Cor(\Delta_{\text{in}}\xrightarrow{r_{DL}=12}\lambda_{\text{out}})=2^{-23.73} \end{cases}.$$

最终我们得到了如下 8 个 SPECK48 的 12 轮区分器:

$$\begin{cases} D_3:(148090,101080)\xrightarrow{r_{DL}=12}(65400\text{c},6\text{c}4000),\ D_4:(148090,101080)\xrightarrow{r_{DL}=12}(41400\text{c},4\text{c}4000) \\ D_5:(148090,101080)\xrightarrow{r_{DL}=12}(45\text{c}00\text{c},484000),\ D_6:(148090,101080)\xrightarrow{r_{DL}=12}(61\text{c}00\text{c},684000) \\ D_7:(148090,101080)\xrightarrow{r_{DL}=12}(61\text{c}028,684020),\ D_8:(148090,101080)\xrightarrow{r_{DL}=12}(45\text{c}028,484020) \\ D_9:(148090,101080)\xrightarrow{r_{DL}=12}(414028,4\text{c}4020),\ D_{10}:(148090,101080)\xrightarrow{r_{DL}=12}(654028,6\text{c}4020) \end{cases}.$$

● SPECK48 的 12 轮差分-线性区分器 2:

$$\begin{cases} r_0=4,\ r_m=5,\ r_1=3 \\ \Delta_m=[18,21,42],\ \lambda_m=[5] \\ p=\Pr(\Delta_{\text{in}} \xrightarrow{r_0=4} \Delta_m)=2^{-9} \\ r=Cor(\Delta_m \xrightarrow{r_m=5} \lambda_m)=2^{-7.96} \\ q=Cor(\lambda_m \xrightarrow{r_1=3} \lambda_{\text{out}})=2^{-3} \\ Cor(\Delta_{\text{in}} \xrightarrow{r_{DL}=12} \lambda_{\text{out}})=2^{-22.96} \end{cases}.$$

最终我们得到了如下 8 个 SPECK48 的 12 轮区分器:

$$\begin{cases} D_{11}:(028210,000212) \xrightarrow{r_{DL}=12} (65400c,6c4000),\ D_{12}:(028210,000212) \xrightarrow{r_{DL}=12} (41400c,4c4000) \\ D_{13}:(028210,000212) \xrightarrow{r_{DL}=12} (45c00c,484000),\ D_{14}:(028210,000212) \xrightarrow{r_{DL}=12} (61c00c,684000) \\ D_{15}:(028210,000212) \xrightarrow{r_{DL}=12} (61c028,684020),\ D_{16}:(028210,000212) \xrightarrow{r_{DL}=12} (45c028,484020) \\ D_{17}:(028210,000212) \xrightarrow{r_{DL}=12} (414028,4c4020),\ D_{18}:(028210,000212) \xrightarrow{r_{DL}=12} (654028,6c4020) \end{cases}.$$

● SPECK48/96 的 14 轮密钥恢复: 我们以区分器 $D_{12}$ 下的密钥恢复攻击为例进行介绍, 其他区分器下的攻击过程是类似的. 在 SPECK48 的 12 轮差分-线性区分器 $D_{12}$ 后添加 2 轮, 我们给出了 SPECK48/96 的 14 轮密钥恢复攻击. 类似 SPECK32/64 的密钥恢复过程, 将 $i=2$, 3, 14, 16, 22 代入 $x_i^{13}$, 可知对 SPECK48/96 实施 14 轮的密钥恢复攻击需要猜测 $(k_{19}^{13},k_{18}^{13},\ldots,k_0^{13})$, $k^{14}$ 共 44 个密钥比特. 此时攻击需要的数据复杂度为 $2^{22.96\times 2}=2^{45.92}$, 需要的时间复杂度为 $2^{45.92+44}=2^{91.92}$.

## 4 总 结

在本文中, 我们对 ARX 类密码算法的差分-线性区分器搜索算法进行研究. 首先, 通过遍历 SPECK32/64 的 32 比特明文空间, 测试不同随机密钥下的差分-线性逼近的相关性, 进一步展示了样本量的大小与实验测量的相关性的准确性之间的关系. 然后, 研究高相关性的差分-线性逼近差分部分和线性部分的特点, 进而基于这些特点给出了一个 ARX 类对称密码算法差分-线性区分器的搜索算法; 基于所提搜索算法, 我们得到了 SPECK32 的 11 轮差分-线性区分器和 SPECK48 的 12 轮差分-线性区分器.

## References

[1] Wheeler DJ, Needham RM. TEA, a tiny encryption algorithm. In: Proc. of the 2nd Int'l Workshop on Fast Software Encryption. Leuven, Belgium: Springer, 1994. 363–366. [doi: 10.1007/3-540-60590-8_29]

[2] Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L. The SIMON and speck block ciphers on AVR 8-bit microcontrollers. In: Proc. of the 3rd Int'l Workshop on Lightweight Cryptography for Security and Privacy. Istanbul: Springer, 2014. 3–20. [doi: 10.1007/978-3-319-16363-5_1]

[3] Hong D, Sung J, Hong S, Lim J, Lee S, Koo BS, Lee C, Chang D, Lee J, Jeong K, Kim H, Kim J, Chee S. HIGHT: A new block cipher suitable for low-resource device. In: Proc. of the 8th Int'l Workshop on Cryptographic Hardware and Embedded Systems. Yokohama: Springer, 2006. 46–59. [doi: 10.1007/11894063_4]

[4] Hong D, Lee J, Kim DC, Kwon D, Ryu KH, Lee DG. LEA: A 128-bit block cipher for fast encryption on common processors. In: Proc. of the 14th Int'l Workshop on Information Security Applications. Jeju Island: Springer, 2013. 3–27. [doi: 10.1007/978-3-319-05149-9_1]

[5] Koo B, Roh D, Kim H, Jung Y, Lee DG, Kwon D. CHAM: A family of lightweight block ciphers for resource-constrained devices. In: Proc. of the 20th Int'l Conf. on Information Security and Cryptology. Seoul: Springer, 2017. 3–25. [doi: 10.1007/978-3-319-78556-1_1]

[6] Dinu D, Perrin L, Udovenko A, Velichkov V, Großschädl J, Biryukov A. Design strategies for ARX with provable bounds: SPARX and LAX. In: Proc. of the 22nd Int'l Conf. on the Theory and Application of Cryptology and Information Security on Advances in Cryptology. Hanoi: Springer, 2016. 484–513. [doi: 10.1007/978-3-662-53887-6_18]

[7] Langford SK, Hellman ME. Differential-linear cryptanalysis. In: Proc. of the 14th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 1994. 17–25. [doi: 10.1007/3-540-48658-5_3]

[8] Biham E, Dunkelman O, Keller N. Enhancing differential-linear cryptanalysis. In: Proc. of the 2002 Int'l Conf. on the Theory and

Application of Cryptology and Information Security. Berlin: Springer, 2002. 254–266. [doi: 10.1007/3-540-36178-2_16]

[9] Blondeau C, Leander G, Nyberg K. Differential-linear cryptanalysis revisited. In: Proc. of the 21st Int'l Conf. on Fast Software Encryption. London: Springer, 2014. 411–430. [doi: 10.1007/978-3-662-46706-0_21]

[10] Bar-On A, Dunkelman O, Keller N, Weizman A. DLCT: A new tool for differential-linear cryptanalysis. In: Proc. of the 38th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology. Darmstadt: Springer, 2019. 313–342. [doi: 10.1007/978-3-030-17653-2_11]

[11] Cid C, Huang T, Peyrin T, Sasaki Y, Song L. Boomerang connectivity table: A new cryptanalysis tool. In: Proc. of the 2018 Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2018. 683–714. [doi: 10.1007/978-3-319-78375-8_22]

[12] Bellini E, Gérault D, Grados J, Makarim RH, Peyrin T. Fully automated differential-linear attacks against ARX ciphers. IACR Cryptology ePrint Archive, 2023. 252–276. [doi: 10.1007/978-3-031-30872-7_10]

[13] Chen Y, Bao ZZ, Yu HB. Differential-linear approximation semi-unconstrained searching and partition tree: Application to LEA and speck. In: Proc. of the 29th Int'l Conf. on the Theory and Application of Cryptology and Information Security on Advances in Cryptology. Guangzhou: Springer, 2023. 223–255. [doi: 10.1007/978-981-99-8727-6_8]

[14] Beierle C, Leander G, Todo Y. Improved differential-linear attacks with applications to ARX ciphers. In: Proc. of the 40th Annual Int'l Cryptology Conf. on Advances in Cryptology. Santa Barbara: Springer, 2020. 329–358. [doi: 10.1007/978-3-030-56877-1_12]

[15] Dunkelman O, Indesteege S, Keller N. A differential-linear attack on 12-round serpent. In: Proc. of the 9th Int'l Conf. on Progress in Cryptology. Kharagpur: Springer, 2008. 308–321. [doi: 10.1007/978-3-540-89754-5_24]

[16] Aumasson JP, Fischer S, Khazaei S, Meier W, Rechberger C. New features of Latin dances: Analysis of Salsa, ChaCha, and Rumba. In: Proc. of the 15th Int'l Conf. on Fast Software Encryption. Lausanne: Springer, 2008. 470–488. [doi: 10.1007/978-3-540-71039-4_30]

## 作者简介

**张语晗**, 博士生, 主要研究领域为对称密码算法的安全性分析.

**张蕾**, 博士, 副研究员, 主要研究领域为对称密码.

**吴文玲**, 博士, 研究员, 主要研究领域为密码学.