中国科学: 信息科学 2024 年 第 54 卷 第 2 期: 335–353
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
论文

分组密码 FBC 的差分分析

刘端[1,5], 罗毅博[2,7], 贾珂婷[3,4], 张国艳[1,5,6*], 邹光南[8], 尤启迪[7], 陈颖[9]

1. 山东大学网络空间安全学院, 青岛 266237
2. 中国科学技术大学网络空间安全学院, 合肥 230022
3. 清华大学网络科学与网络空间研究院, 北京 100084
4. 中关村实验室, 北京 100095
5. 密码技术与信息安全教育部重点实验室, 青岛 266237
6. 山东区块链研究院, 济南 250101
7. 北京卫星信息工程研究所天地一体化信息技术国家重点实验室, 北京 100086
8. 清华大学计算机系, 北京 100084
9. 北京电子科技学院密码科学与技术系, 北京 100070
* 通信作者. E-mail: guoyanzhang@sdu.edu.cn

收稿日期: 2023–06–23; 修回日期: 2023–09–04; 接受日期: 2023–10–16; 网络出版日期: 2024–02–06

国家重点研发计划 (批准号: 2022YFB2702804)、国家自然科学基金 (批准号: 62072270) 和山东省重点研发计划 (批准号: 2020ZLYS09, 2019JZZY010133) 资助项目

摘要 FBC 是一种轻量级分组密码算法, 由于结构简单、软硬件实现灵活等优点成为 2018 年中国密码学会 (CACR) 举办的全国密码算法设计竞赛中晋级到第 2 轮的 10 个算法之一. FBC 密码包含 3 个版本支持 128 和 256 两种比特长度的明文分组以及 128 和 256 两种比特长度的密钥, 本文主要对分组长度 128 位的两个版本进行分析. 我们基于 SAT (Boolean satisfiability problem) 模型对 FBC 的差分特征进行自动化搜索, 得到了新的 14 轮差分路线, 概率为 $2^{-102.25}$. 基于此路线我们给出了 18 轮 FBC128-128 和 20 轮 FBC128-256 差分分析, 并且在分析过程中给出了复杂度估计. 对于 18 轮 FBC128-128 差分分析, 时间复杂度和存储复杂度分别为 $2^{101.5}$ 和 2^{52}. 对于 20 轮 FBC128-256 差分分析时间复杂度和存储复杂度分别为 2^{184} 和 2^{96}.

关键词 分组密码, 差分分析, FBC 算法, 布尔可满足性问题

1 引言

FBC 算法是 2018 年全国密码算法设计竞赛晋级到第 2 轮的 10 个算法之一, 该算法由 Feng 等 [1] 设计, 基于 4 路两重 Feistel 结构, 具有较好的软硬件实现效率. FBC 算法支持 128 和 256 比特明文分组长度以及 128 和 256 比特密钥长度, 分别表示为 FBC128-128, FBC128-256 和 FBC256-256, 对应的

引用格式: 刘端, 罗毅博, 贾珂婷, 等. 分组密码 FBC 的差分分析. 中国科学: 信息科学, 2024, 54: 335–353, doi: 10.1360/SSI-2023-0189
Liu D, Luo Y B, Jia K T, et al. Differential analysis of block cipher FBC (in Chinese). Sci Sin Inform, 2024, 54: 335–353, doi: 10.1360/SSI-2023-0189

轮数分别为 48, 64 和 80. 本文主要针对分组长度为 128 的版本进行了分析. FBC [1] 算法基于 4 路两重 Feistel 结构, 具有较好的软硬件实现效率.

基于混合整数线性规划 (mixed-integer linear programming, MILP) 的差分路线搜索技术被 Mouha 等 [2] 引入到 AES (advanced encryption standard) 的分析中, 已经成为密码自动化分析的一种基本工具. FBC 的设计者采用混合整数线性规划 (MILP) 方法搜索了差分分析中最优活跃 S 盒数, 对于 r 轮迭代, 其最优活跃 S 盒数为 $2r$, 给出 FBC128-128 达到安全界的迭代轮数为 32, 也采用 MILP 方法搜得线性分析中最优活跃 S 盒数为 $2r$ (对于 r 轮迭代), 这些估计并不是很精确. 除了差分和线性分析, 设计者也给出了 7 轮 FBC128-128 积分区分器、7 轮 FBC128-128 的零相关区分器. Zhang 等 [3] 构造了 9 轮 FBC128-128 的不可能差分区分器并发起 13 轮密钥恢复攻击, 时间复杂度为 $2^{51.39}$. Ren 等 [4] 构造了 11 轮差分区分器并给出 12 轮差分攻击, 计算复杂度为 $2^{93.41}$. 他还给出了 11 轮线性攻击, 计算复杂度为 $2^{112.54}$, 以及计算复杂度为 $2^{94.54}$ 的 11 轮不可能差分攻击. Zou 等[1)] 给出了 11 轮 FBC128-128 算法的截断差分区分器, 进行 15 轮密钥恢复攻击, 攻击的时间复杂度为 2^{100}. 对 FBC256-256 算法构造了 15 轮截断差分区分器, 进行 20 轮密钥恢复攻击, 时间复杂度为 $2^{222.6}$.

除了基于 MILP 的差分路线搜索, 布尔可满足性问题 (Boolean satisfiability problem, SAT) 也被用于差分路线搜索中. SAT 主要考虑给定布尔公式的可满足性, 并利用各种 SAT 求解器来求解, 例如 minisat [5], CaDiCaL [6] 等. Mouha 和 Preneel [7] 首次提出基于 SAT 的自动搜索方法. 在 International Cryptology Conference (CRYPTO) 2015 上, Kölbl 等 [8] 基于 SAT 搜索了 SIMON 算法的最优差分和线性路线. 在 Applied Cryptography and Network Security (ACNS) 2016 上, Liu 等 [9] 将基于 SAT 的自动搜索方法扩展到 ARX 结构的密码上. 在 FSE 2022 上, Sun 等 [10] 提出用 Matsui 边界条件来加速 SAT 方法, 并应用在多种 Feistel 结构、SPN 结构和类 SIMON 结构的密码中.

本文基于 SAT 自动搜索模型, 并对差分概率与密钥恢复过程的密钥筛选通过率进行权衡, 构造了 FBC128 的差分特征搜索工具, 得到了 FBC-128 14 轮差分区分器, 概率为 $2^{-102.25}$. 基于该区分器, 我们首次给出了 18 轮 FBC128-128 和 20 轮 FBC128-256 的差分分析. 其中, 对 18 轮 FBC128-128 分析所需的数据复杂度为 2^{105}, 时间复杂度为 $2^{101.5}$ 次 18 轮加密, 存储复杂度为 2^{52}, 与已知的最优分析结果 [1)] 提高了 3 轮, 对 20 轮 FBC128-256 分析所需的数据复杂度为 2^{105}, 时间复杂度为 2^{184} 次 20 轮加密, 存储复杂度为 2^{96}. 据我们所知, 这是目前对 FBC128-128 和 FBC128-256 最好的分析结果.

本文结构安排如下: 第 2 节简单介绍 FBC 算法, 第 3 节对基于 SAT 的分组密码差分特征自动搜索方法进行简要介绍, 第 4 节给出 FBC 算法的差分特征搜索结果, 以及如何基于差分特征搜索差分路线, 第 5 和 6 节分别给出 18 轮 FBC128-128 和 20 轮 FBC128-256 的差分分析, 第 7 节对全文进行总结.

2 分组密码 FBC 算法描述

2.1 符号说明

本文所使用的主要符号和运算符释义如下:

r: 迭代轮数, 取值为 48 或 64;

K: 主密钥, 长度为 128 比特或 256 比特;

sk_i: 第 i 个子密钥, $i = 0, 1, \ldots, 2r-1$;

1) Zou G N, Liu D, Jia K T, et al. Truncated differental cryptanalysis of FBC (accepted). J Cryptol Res.

A_i, B_i, C_i, D_i: 第 i 轮迭代的状态字, 每个字长度为 32 比特, $i = 0, 1, \ldots, r$;
ΔA_i, ΔB_i, ΔC_i, ΔD_i: 第 i 轮迭代状态字的差分, 每个字长度为 32 比特, $i = 0, 1, \ldots, r$;
$A_i^{[j]}$: 字 A_i 的第 j 比特, 从最高有效位开始计数 $j = 0, 1, \ldots, 31$;
$A_i^{[j_1, j_2, \ldots]}$: 字 A_i 的第 $j_1, j_2, \ldots$ 比特的集合;
x, y, z, u, v: 长度为 32 比特的字;
u_i, v_i: 字 u 和 v 的第 i 个长度为 8 比特的子块, $i = 1, 2, 3, 4$;
$u_i^{[j]}$, $v_i^{[j]}$: 子块 u_i 和 v_i 的第 j 比特, $j = 0, 1, \ldots, 7$;
$\mathbf{1}$: 长度为 32 比特的全 1 字;
$\mathrm{DP}_S(\boldsymbol{x}, \boldsymbol{y})$: S 盒差分模式 $(\boldsymbol{x}, \boldsymbol{y})$ 的概率;
$\mathrm{Pr}_{\mathrm{opt}}(i)$: i 轮最优差分特征的概率, $1 \leqslant i \leqslant R-1$;
$\mathrm{Pr}_{\mathrm{ini}}(R)$: 第 R 轮最优差分特征概率的下界;
$\mathcal{C}_{(r_1, r_2)}$: $r_1 + 1 - r_2$ 轮的 Matsui 边界条件式;
$\oplus$: 面向比特的异或运算;
&: 面向比特的与运算;
$\lll$: 32 比特字的循环左移运算;
$\|$: 字符串连接符.

2.2 FBC 算法的加密过程

FBC 算法基于 4 路两重 Feistel 结构设计, 如图 1 所示. 其根据明文分组长度和主密钥长度设计了 3 个算法版本: FBC128-128, FBC128-256, FBC256-256. 本文主要对 FBC128-128 和 FBC128-256 进行安全性分析, 并在不涉及密钥扩展算法时简记为 FBC-128. 下面描述 FBC128-128 和 FBC128-256 的加密过程, 若需要参考其他算法版本, 请参阅文献 [1, 11].

FBC128-128 和 FBC128-256 的迭代轮数 r 分别为 48 和 64, 每一轮的中间状态可以表示为 4 个 32 比特的字. 设 A_{i-1}, B_{i-1}, C_{i-1}, D_{i-1} 和 A_i, B_i, C_i, D_i 分别为第 i 轮加密的输入和输出, sk_{2i-2}, sk_{2i-1} 为第 i 轮的子密钥, F 为轮函数, 则 FBC-128 的 r 轮加密流程可以表示为

$$\left.\begin{aligned} A_i &= F(A_{i-1}, \mathrm{sk}_{2i-2}) \oplus B_{i-1} \\ C_i &= A_i \oplus D_{i-1} \\ D_i &= F(D_{i-1}, \mathrm{sk}_{2i-1}) \oplus C_{i-1} \\ B_i &= D_i \oplus A_{i-1} \end{aligned}\right\} 1 \leqslant i \leqslant r. \tag{1}$$

密文输出为

$$C = B_r \,\|\, A_r \,\|\, D_r \,\|\, C_r, \tag{2}$$

其中轮函数 F 是个 $\left(\mathbb{F}_2^{32}, \mathbb{F}_2^{32}\right) \to \mathbb{F}_2^{32}$ 的非线性函数, 它的输入为 2 个 32 比特的字 x 和 y, 输出为一个 32 比特的字 z, 记 $z = F(x, y)$. 具体运算过程如下:

(1) 子密钥加: $u = x \oplus y$.

(2) 列变换: 首先将 32 比特的字 u 分解为等宽的 4 个部分, 每个部分大小为 8 比特, 即 $u = u_1 \|$

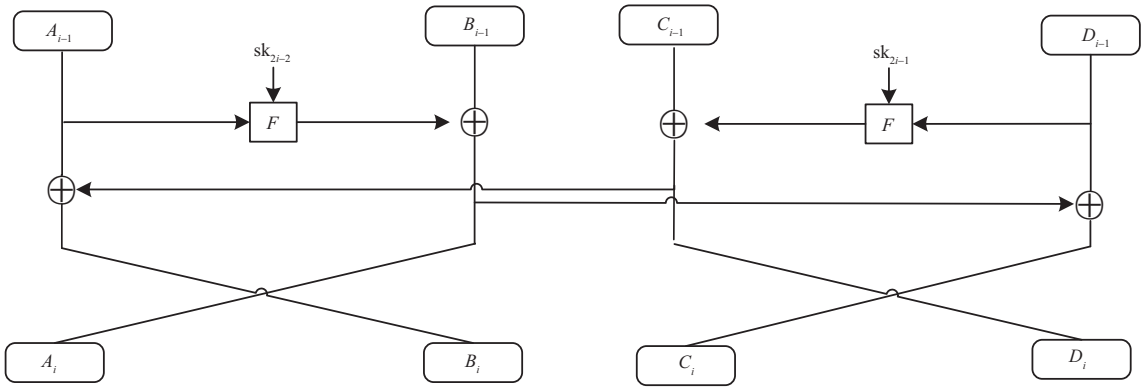

图 1　FBC 算法结构

Figure 1　Structure of FBC

表 1　FBC 算法的 S 盒

Table 1　S-box of FBC cipher

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(x)$	5	10	15	4	9	14	11	8	2	7	12	13	3	6	1	0

$u_2 \parallel u_3 \parallel u_4$, 于是列变换可以表示为

$$\begin{cases} v_1^{[j]} \parallel v_2^{[j]} \parallel v_3^{[j]} \parallel v_4^{[j]} = S\left(u_1^{[j]} \parallel u_2^{[j]} \parallel u_3^{[j]} \parallel u_4^{[j]}\right),\ j = 0, 1, \ldots, 7, \\ v = v_1 \parallel v_2 \parallel v_3 \parallel v_4, \end{cases} \tag{3}$$

其中, S 是 4 比特的 S 盒代换, 如表 1 所示.

(3) 行变换 L: $z = L_{3,10}(v) = v \oplus (v \lll 3) \oplus (v \lll 10)$.

2.3　FBC 算法的密钥扩展算法

针对 FBC128-128, 将 128 比特主密钥 K 分解为 4 个 32 比特的字 sk_0, sk_1, sk_2, sk_3, 对 $i = 0, 1, \ldots, 2r-5$, 通过以下式子计算子密钥:

$$\begin{cases} \mathrm{sk}_{i+4} = [(\mathrm{sk}_i \oplus \mathbf{1}) \oplus (\mathrm{sk}_{i+1} \oplus \mathbf{1})] \,\&\mathrm{sk}_{i+2}\&\mathrm{sk}_{i+3} \oplus i, \\ \mathrm{sk}_{i+4} = L_{7,25}(\mathrm{sk}_{i+4}) = \mathrm{sk}_{i+4} \oplus (\mathrm{sk}_{i+4} \lll 7) \oplus (\mathrm{sk}_{i+4} \lll 25). \end{cases} \tag{4}$$

针对 FBC128-256, 将 256 比特主密钥 K 分解为 8 个 32 比特的字 sk_0, sk_1, $\mathrm{sk}_2, \ldots, \mathrm{sk}_7$, 对 $i = 0, 1, \ldots, 2r-9$, 通过以下式子计算子密钥:

$$\begin{cases} \mathrm{sk}_{i+8} = [(\mathrm{sk}_i \oplus \mathbf{1}) \oplus (\mathrm{sk}_{i+1} \oplus \mathbf{1})] \,\&\mathrm{sk}_{i+2}\&\mathrm{sk}_{i+3} \oplus i, \\ \mathrm{sk}_{i+8} = L_{7,25}\left(\mathrm{sk}_{i+8}\right) = \mathrm{sk}_{i+8} \oplus (\mathrm{sk}_{i+8} \lll 7) \oplus (\mathrm{sk}_{i+8} \lll 25). \end{cases} \tag{5}$$

3　基于 SAT 的差分特征自动搜索方法

布尔可满足性问题 (SAT) 是一类判断仅由与 ($\wedge$)、或 ($\vee$)、非 ($\bar{\cdot}$) 3 种逻辑运算构成的布尔公式是否存在可行解的问题. 若存在可行解, 则称该布尔公式是可满足的 (satisfiable), 否则称它为不可满足

的 (unsatisfiable). 尽管 SAT 问题被证明是 NP 完全的 [12], 但我们依然可以将需要求解的问题转化成 SAT 的形式, 并借助现代的 SAT 求解器快速求解, 如 CaDiCaL, minisat, Lingeling 等.

一般来说, 为了便于 SAT 求解器读取输入, 需要将问题等价转化为布尔公式的标准形式 – 合取范式 (conjunctive normal form, CNF), 形如 $\bigwedge_{i=0}^{m}\bigvee_{j=0}^{n_i} c_{ij}$. 其中 c_{ij} 表示一个布尔变量或常数 (的反), 每一项 $\bigvee_{j=0}^{n_i} c_{ij}$ 称为一条子句 (clause). 因此, 首先要将搜索差分特征的原问题转化为用 CNF 表示的 SAT 模型.

3.1 基本部件的 SAT 模型

本小节将介绍 FBC-128 中使用的几种基本部件 (XOR, 3-XOR 和 S 盒代换) 的 SAT 模型, 用于约束它们的差分传播. 对于其他部件的 SAT 模型, 请参阅文献 [10].

差分模型 1 (XOR) [10]. 对于 n 比特的 XOR 运算, 设 $a=(a_0,a_1,\ldots,a_{n-1})$, $b=(b_0,b_1,\ldots,b_{n-1})$ 为 n 比特的输入差分, $y=(y_0,y_1,\ldots,y_{n-1})$ 为 n 比特的输出差分, 则 $(a,b)\to y$ 为一条有效差分路径当且仅当

$$\left.\begin{array}{l} a_i\vee b_i\vee \overline{y_i}=1\\ a_i\vee \overline{b_i}\vee y_i=1\\ \overline{a_i}\vee b_i\vee y_i=1\\ \overline{a_i}\vee \overline{b_i}\vee \overline{y_i}=1 \end{array}\right\} 0\leqslant i\leqslant n-1. \tag{6}$$

差分模型 2 (3-XOR) [10]. 对于 n 比特的 3-XOR 运算, 设

$$a=(a_0,a_1,\ldots,a_{n-1}),\ b=(b_0,b_1,\ldots,b_{n-1}),\ c=(c_0,c_1,\ldots,c_{n-1})$$

为它的输入差分, $y=(y_0,y_1,\ldots,y_{n-1})$ 为输出差分, 则 $(a,b,c)\to y$ 为一条有效差分路径当且仅当

$$\left.\begin{array}{l} a_i\vee b_i\vee c_i\vee \overline{y_i}=1\\ a_i\vee b_i\vee \overline{c_i}\vee y_i=1\\ a_i\vee \overline{b_i}\vee c_i\vee y_i=1\\ \overline{a_i}\vee b_i\vee c_i\vee y_i=1\\ \overline{a_i}\vee \overline{b_i}\vee \overline{c_i}\vee y_i=1\\ \overline{a_i}\vee \overline{b_i}\vee c_i\vee \overline{y_i}=1\\ \overline{a_i}\vee b_i\vee \overline{c_i}\vee \overline{y_i}=1\\ a_i\vee \overline{b_i}\vee \overline{c_i}\vee \overline{y_i}=1 \end{array}\right\} 0\leqslant i\leqslant n-1. \tag{7}$$

差分模型 3 (S 盒) [13]. 对于差分均匀度为 4 的 4 比特 S 盒, 它的差分分布表 (differential distribution table, DDT) 中的项仅包含 0, 2, 4, 16 这 4 种取值, 对应的概率分别为 0, 2^{-3}, 2^{-2}, 1. 因此可以引入 3 个辅助变量 p_0, p_1, p_2, 用 $p_0+p_1+p_2$ 表示有效差分模式对应概率的权重. 设 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 分别为 4 比特 S 盒的输入和输出差分, $\boldsymbol{p}=(p_0,p_1,p_2)$, $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})\in\mathbb{F}_2^{11}$, 通过如下布尔方程 $f(\boldsymbol{x}\parallel\boldsymbol{y}\parallel\boldsymbol{p})$ 来刻画 S 盒的差分传播模式:

$$\text{if } \mathrm{DP}_S(\boldsymbol{x},\boldsymbol{y})=0 \text{ then}: \qquad f(\boldsymbol{x}\parallel\boldsymbol{y}\parallel\boldsymbol{p})=0,$$

$$\text{if } \mathrm{DP}_S(\boldsymbol{x},\boldsymbol{y})=2^{-3} \text{ then}: \quad f(\boldsymbol{x}\parallel\boldsymbol{y}\parallel\boldsymbol{p})=\begin{cases}1 \text{ if } \boldsymbol{p}=(1,1,1),\\ 0 \text{ else},\end{cases}$$

$$\text{if } \mathrm{DP}_S(\boldsymbol{x},\boldsymbol{y})=2^{-2} \text{ then}: \quad f(\boldsymbol{x}\parallel\boldsymbol{y}\parallel\boldsymbol{p})=\begin{cases}1 \text{ if } \boldsymbol{p}=(0,1,1),\\ 0 \text{ else},\end{cases}$$

$$\text{if } \mathrm{DP}_S(\boldsymbol{x},\boldsymbol{y})=1 \text{ then}: \quad f(\boldsymbol{x}\parallel\boldsymbol{y}\parallel\boldsymbol{p})=\begin{cases}1 \text{ if } \boldsymbol{p}=(0,0,0),\\ 0 \text{ else},\end{cases}$$

其中, $\mathrm{DP}_S(\boldsymbol{x},\boldsymbol{y})$ 是差分模式 $(\boldsymbol{x},\boldsymbol{y})$ 的概率. 接下来, 只需要将此布尔方程通过启发式算法 Espresso [14] 化简, 即可得到表达 S 盒差分概率传播的 SAT 模型, 可以借助软件 Logic Friday[2)] 完成该步骤.

3.2 序列编码方法

为了搜索高概率的差分特征, 需要将活跃 S 盒数或差分概率限制在一定范围内, 这种约束可以归纳为布尔基数约束 $\sum_{i=0}^{n-1} x_i \leqslant k$, 其中 x_i 是布尔变量, k 是一非负整数. 为了将这种基数约束转化为 CNF 子句, Liu 等 [9] 使用了序列编码方法 [15] 对它进行编码.

序列编码方法通过引入 $(n-1)\cdot k$ 个辅助变量 $s_{i,j}$ $(0 \leqslant i \leqslant n-2,\ 0 \leqslant j \leqslant k-1)$ 计算基数约束的部分和 $s_i=\sum_{j=0}^{i} x_j$. 具体来说, 对每一个部分和 $s_i \leqslant k$, 利用 k 个辅助变量将其表示为一进制 $s_{i,0}\parallel s_{i,1}\parallel\cdots\parallel s_{i,k-1}$, 即 $s_i=m$ 时, $s_{i,0}=\cdots=s_{i,m-1}=1$, $s_{i,m}=\cdots=s_{i,k-1}=0$, 从而 $s_i=\sum_{j=0}^{k-1} s_{i,j}=m$. 这样一来, 就能够通过 s_{i-1} 和 x_i 约束 s_i 的取值, 最终得到基数约束 $\sum_{i=0}^{n-1} x_i \leqslant k$ 的 SAT 模型. 实现序列编码方法需要添加如下 $2kn+n-3k-1$ 条子句:

$$\left.\begin{array}{l}\overline{x_0}\vee s_{0,0}=1\\ \overline{s_{0,j}}=1,\ \ 1\leqslant j\leqslant k-1\\ \overline{x_i}\vee s_{i,0}=1\\ \overline{s_{i-1,0}}\vee s_{i,0}=1\\ \left.\begin{array}{l}\overline{x_i}\vee\overline{s_{i-1,j-1}}\vee s_{i,j}=1\\ \overline{s_{i-1,j}}\vee s_{i,j}=1\end{array}\right\}1\leqslant j\leqslant k-1\\ \overline{x_i}\vee\overline{s_{i-1,k-1}}=1\\ \overline{x_{n-1}}\vee\overline{s_{n-2,k-1}}=1\end{array}\right\}1\leqslant i\leqslant n-2. \tag{8}$$

3.3 Matsui 边界条件引入

在 European Cryptology Conference (EUROCRYPT) 1994 上, Matsui [16] 提出了一种深度优先的分枝定界算法来搜索 DES 算法的最优差分特征, 这是第一次借助程序对分组密码区分器进行自动化搜索, 并由此衍生出了一系列的 Matsui-like 自动搜索算法 [17~19]. 在 Matsui 的算法中, 为了给搜索过程定界以达到剪枝效果, 引入了一系列基于已知最优差分特征概率构造的边界条件, 称为 Matsui 边界条件. Sun 等 [10] 将这些边界条件编码并加入到搜索差分特征的 SAT 模型中, 实现了显著的加速效果. 本小节将简要介绍 Matsui 边界条件, 以及与序列编码方法结合的 Matsui 边界条件 SAT 模型.

2) Logic friday. http://windows.dailydownloaded.com/en/educational-software/student-tools/44924-logic-friday-download-install.

设 $\mathrm{Pr}_{\mathrm{opt}}(i)$ $(1 \leqslant i \leqslant R-1)$ 是已知的前 $R-1$ 轮最优差分特征概率, $\mathrm{Pr}_{\mathrm{ini}}(R)$ 是对第 R 轮最优差分特征概率估计的下界. 算法目标是通过搜索得到第 R 轮的最优差分特征概率 $\mathrm{Pr}_{\mathrm{opt}}(R)$, 即寻找一条差分特征 $(\alpha^0, \ldots, \alpha^R)$, 满足

$$\prod_{i=0}^{R-1} \Pr\left(\alpha^i \to \alpha^{i+1}\right) \geqslant \mathrm{Pr}_{\mathrm{ini}}(R). \tag{9}$$

以此更新 $\mathrm{Pr}_{\mathrm{ini}}(R)$ 的值, 直至遍历完整个搜索空间. 在这个过程中, 假设已经搜索到了第 r_0 轮的节点, 则不被剪枝的节点一定满足条件

$$\prod_{i=0}^{r_0-1} \Pr\left(\alpha^i \to \alpha^{i+1}\right) \cdot \mathrm{Pr}_{\mathrm{opt}}\left(R-r_0\right) \geqslant \mathrm{Pr}_{\mathrm{int}}(R). \tag{10}$$

上式即为 Matsui 边界条件的一般表达式, 目的是判断当前节点的概率与未搜索轮数的最优概率相结合得到的理想差分特征是否优于下界 $\mathrm{Pr}_{\mathrm{ini}}(R)$ 对应的差分特征. 显然, 不满足此条件的节点不可能扩展出优于 $\mathrm{Pr}_{\mathrm{ini}}(R)$ 的新差分特征.

为了将 Matsui 边界条件与基于 SAT 的自动搜索方法结合, 首先设 $k = -\log_2\left(\mathrm{Pr}_{\mathrm{ini}}(R)\right)$, R 轮特征共含 n 个权重变量 $x_1, \ldots, x_{n-1}$, 并利用序列编码方法构造 $\sum_{i=0}^{n-1} x_i \leqslant k$ 的 SAT 模型. 然后考虑 Matsui 边界条件的推广, 即约束的轮数由 1–r_0 推广为 (r_1+1)–r_2 $(0 \leqslant r_1 < r_2 \leqslant R)$, 此时应满足条件

$$\mathrm{Pr}_{\mathrm{opt}}\left(r_1\right) \cdot \prod_{i=r_1}^{r_2-1} \Pr\left(\alpha^i \to \alpha^{i+1}\right) \cdot \mathrm{Pr}_{\mathrm{opt}}\left(R-r_2\right) \geqslant \mathrm{Pr}_{\mathrm{int}}(R). \tag{11}$$

将式 (11) 取负对数并移项得

$$-\sum_{i=r_1}^{r_2-1} \log_2\left(\Pr\left(\alpha^i \to \alpha^{i+1}\right)\right) \leqslant -\log_2\left(\mathrm{Pr}_{\mathrm{tnt}}\left(R\right)\right) + \log_2\left(\mathrm{Pr}_{\mathrm{opt}}\left(r_1\right)\right) + \log_2\left(\mathrm{Pr}_{\mathrm{opt}}\left(R-r_2\right)\right). \tag{12}$$

式 (12) 左边为 (r_1+1)–r_2 轮概率权重累加, 右边 3 项均为常数, 且和不大于 k, 设为 m, 因此可以改写为

$$\sum_{i=e_1}^{e_2} x_i \leqslant m, \tag{13}$$

其中, $e_1 \geqslant 0, e_2 \leqslant n-1$ 且 $m \leqslant k$.

注意到式 (13) 左边可以表示为两个部分和之差, 即 $\sum_{i=e_1}^{e_2} x_i = \sum_{i=0}^{e_2} x_i - \sum_{i=0}^{e_1} x_i = s_{e_2} - s_{e_1}$, 而这两个部分和已经由式 (8) 引入的辅助变量编码, 因此可以基于序列编码方法给出如下的 Matsui 边界条件 SAT 模型:

(1) 当 $e_1 = 0$, $e_2 < n-1$ 时, 边界条件 $\sum_{i=0}^{e_2} x_i \leqslant m$ 可由如下 e_2 条子句表示:

$$\overline{x_i} \vee \overline{s_{i-1,m-1}} = 1,\ 1 \leqslant i \leqslant e_2. \tag{14}$$

(2) 当 $e_1 > 0$, $e_2 < n-1$ 时, 边界条件 $\sum_{i=e_1}^{e_2} x_i \leqslant m$ 可由如下 $k-m$ 条子句表示:

$$s_{e_1-1,j} \vee \overline{s_{e_2,j+m}} = 1,\ 0 \leqslant j \leqslant k-m-1. \tag{15}$$

(3) 当 $e_1 > 0$, $e_2 = n-1$ 时, 边界条件 $\sum_{i=e_1}^{n-1} x_i \leqslant m$ 可由如下 $2 \cdot (k-m)+1$ 条子句表示:

$$\begin{cases} s_{e_1-1,j} \vee \overline{s_{n-2,j+m}} = 1,\ 0 \leqslant j \leqslant k-m-1, \\ s_{e_1-1,j} \vee \overline{x_{n-1}} \vee \overline{s_{n-2,j+m-1}} = 1,\ 0 \leqslant j \leqslant k-m. \end{cases} \tag{16}$$

此模型没有添加任何的辅助变量, 且每条 Matsui 边界条件的子句数不超过 $\max(e_2, 2\cdot(k-m)+1)$ 条, 与原基数约束 $\sum_{i=0}^{n-1} x_i \leqslant k$ 的 SAT 模型中的子句数相比可以忽略不计.

Sun 等 [10] 根据此方法对若干轻量级分组密码进行了实验, 结果表明, 使用 Matsui 边界条件集合 $\mathcal{C}_{(0,*)} = \{\mathcal{C}_{(0,x)} \mid 1 \leqslant x \leqslant R-1\}$ 或 $\mathcal{C}_{(*,R-1)} = \{\mathcal{C}_{(x,R-1)} \mid 0 \leqslant x \leqslant R-2\}$ 可以实现较为理想的加速效果, 其中 $\mathcal{C}_{(r_1,r_2)}$ 是 (r_1+1)–r_2 轮对应的 Matsui 边界条件 (式 (12)).

4 FBC-128 的差分特征搜索

4.1 约束首尾两轮的活跃 S 盒数

在进行差分密码分析时, 除差分特征概率外, 密钥恢复过程中的明文对筛选通过率也是影响信噪比的一个重要因素. 筛选通过率越低, 则区分器的筛选能力越强, 对应信噪比的值也越大. 一般来说, 筛选通过率主要受差分分析时首尾两轮的活跃比特数影响, 而这可以基于差分特征的首轮 (尾轮) 向前 (向后) 扩展一轮时的活跃 S 盒数进行估计. 因此, 在搜索差分特征时, 需要对首尾两轮的活跃 S 盒数进行额外约束.

首先引入 32 个布尔辅助变量 $y_r^{[i]}$ $(0 \leqslant i \leqslant 15,\ r = 0, R)$ 表示首尾两轮 S 盒的活跃情况并设 $(\Delta A_0, \Delta B_0, \Delta C_0, \Delta D_0) = (\Delta A_0^{[0,1,\ldots,31]}, \Delta B_0^{[0,1,\ldots,31]}, \Delta C_0^{[0,1,\ldots,31]}, \Delta D_0^{[0,1,\ldots,31]})$, $(\Delta A_R, \Delta B_R, \Delta C_R, \Delta D_R) = (\Delta A_R^{[0,1,\ldots,31]}, \Delta B_R^{[0,1,\ldots,31]}, \Delta C_R^{[0,1,\ldots,31]}, \Delta D_R^{[0,1,\ldots,31]})$ 分别表示 FBC-128 R 轮差分特征的输入和输出差分. 则它们应满足如下关系:

$$
\begin{aligned}
y_0^{[i]} &= \begin{cases}
\left(\Delta B_0^{[i]} \oplus \Delta D_0^{[i]}\right) \vee \left(\Delta B_0^{[i+8]} \oplus \Delta D_0^{[i+8]}\right) \vee \left(\Delta B_0^{[i+16]} \oplus \Delta D_0^{[i+16]}\right) \vee \left(\Delta B_0^{[i+24]} \oplus \Delta D_0^{[i+24]}\right), & 0 \leqslant i \leqslant 7, \\
\left(\Delta A_0^{[i-8]} \oplus \Delta C_0^{[i-8]}\right) \vee \left(\Delta A_0^{[i]} \oplus \Delta C_0^{[i]}\right) \vee \left(\Delta A_0^{[i+8]} \oplus \Delta C_0^{[i+8]}\right) \vee \left(\Delta A_0^{[i+16]} \oplus \Delta C_0^{[i+16]}\right), & 8 \leqslant i \leqslant 15,
\end{cases} \\
y_R^{[i]} &= \begin{cases}
\Delta A_R^{[i]} \vee \Delta A_R^{[i+8]} \vee \Delta A_R^{[i+16]} \vee \Delta A_R^{[i+24]}, & 0 \leqslant i \leqslant 7, \\
\Delta D_R^{[i-8]} \vee \Delta D_R^{[i]} \vee \Delta D_R^{[i+8]} \vee \Delta D_R^{[i+16]}, & 8 \leqslant i \leqslant 15.
\end{cases}
\end{aligned} \tag{17}
$$

而布尔函数 $\mu = \alpha \vee \beta \vee \gamma \vee \delta$ 和 $\nu = (\alpha \oplus \alpha') \vee (\beta \oplus \beta') \vee (\gamma \oplus \gamma') \vee (\delta \oplus \delta')$ 对应的 CNF 子句分别为式 (18) 和 (19). 因此, 将式 (17) 中对应的布尔变量代入式 (18) 和 (19), 得到的 SAT 模型包含 $16 \times (24+5) = 464$ 个 CNF 子句. 然后再使用 2.2 小节的序列编码方法将基数约束 $\sum_{i=0}^{15}(y_0^{[i]} + y_R^{[i]}) \leqslant l$ $(0 \leqslant l \leqslant 32)$ 编码即可, 其中 l 是预设的首尾两轮活跃 S 盒数的最大值. 对于固定的差分概率, 应找到使 l 尽可能小的差分特征.

$$
\begin{cases}
\overline{\mu} \vee \alpha \vee \beta \vee \gamma \vee \delta = 1, \\
\mu \vee \overline{\alpha} = 1, \\
\mu \vee \overline{\beta} = 1, \\
\mu \vee \overline{\gamma} = 1, \\
\mu \vee \overline{\delta} = 1.
\end{cases} \tag{18}
$$

表 2 FBC-128 的最优差分特征概率

Table 2 The maximum probability of differential characteristics of FBC-128

Round	1	2	3	4	5	6	7	8	9	10	11
Probability	0	2^{-2}	2^{-4}	2^{-12}	2^{-18}	2^{-26}	2^{-32}	2^{-46}	2^{-52}	2^{-64}	2^{-72}

$$
\begin{cases}
\nu \vee \alpha \vee \overline{\alpha'} = 1, \\
\nu \vee \overline{\alpha} \vee \alpha' = 1, \\
\nu \vee \beta \vee \overline{\beta'} = 1, \\
\nu \vee \overline{\beta} \vee \beta' = 1, \\
\nu \vee \gamma \vee \overline{\gamma'} = 1, \\
\nu \vee \overline{\gamma} \vee \gamma' = 1, \\
\nu \vee \delta \vee \overline{\delta'} = 1, \\
\nu \vee \overline{\delta} \vee \delta' = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \overline{\beta} \vee \overline{\beta'} \vee \overline{\gamma} \vee \overline{\gamma'} \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \overline{\beta} \vee \overline{\beta'} \vee \overline{\gamma} \vee \overline{\gamma'} \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \beta \vee \beta' \vee \overline{\gamma} \vee \overline{\gamma'} \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \overline{\beta} \vee \overline{\beta'} \vee \gamma \vee \gamma' \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \overline{\beta} \vee \overline{\beta'} \vee \overline{\gamma} \vee \overline{\gamma'} \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \beta \vee \beta' \vee \overline{\gamma} \vee \overline{\gamma'} \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \overline{\beta} \vee \overline{\beta'} \vee \gamma \vee \gamma' \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \overline{\beta} \vee \overline{\beta'} \vee \overline{\gamma} \vee \overline{\gamma'} \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \beta \vee \beta' \vee \gamma \vee \gamma' \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \beta \vee \beta' \vee \overline{\gamma} \vee \overline{\gamma'} \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \overline{\beta} \vee \overline{\beta'} \vee \gamma \vee \gamma' \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \beta \vee \beta' \vee \gamma \vee \gamma' \vee \overline{\delta} \vee \overline{\delta'} = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \beta \vee \beta' \vee \overline{\gamma} \vee \overline{\gamma'} \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \overline{\beta} \vee \overline{\beta'} \vee \gamma \vee \gamma' \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \overline{\alpha} \vee \overline{\alpha'} \vee \beta \vee \beta' \vee \gamma \vee \gamma' \vee \delta \vee \delta' = 1, \\
\overline{\nu} \vee \alpha \vee \alpha' \vee \beta \vee \beta' \vee \gamma \vee \gamma' \vee \delta \vee \delta' = 1.
\end{cases}
\tag{19}
$$

4.2 FBC-128 的差分特征搜索结果

本小节将采用第 3 节及 4.1 小节介绍的基于 SAT 的差分特征自动搜索方法, 对 FBC-128 的差分特征进行搜索. 本文采用的实验设备为一台笔记本电脑, CPU 为 Intel(R) Core(TM) i7-11800H @ 2.30 GHz, 内存 16 GB. 本小节及 4.3 小节的搜索程序均可在一周内得到结果.

首先搜索 FBC-128 各轮的最优差分特征概率, 得到 1–11 轮的结果如表 2 所示.

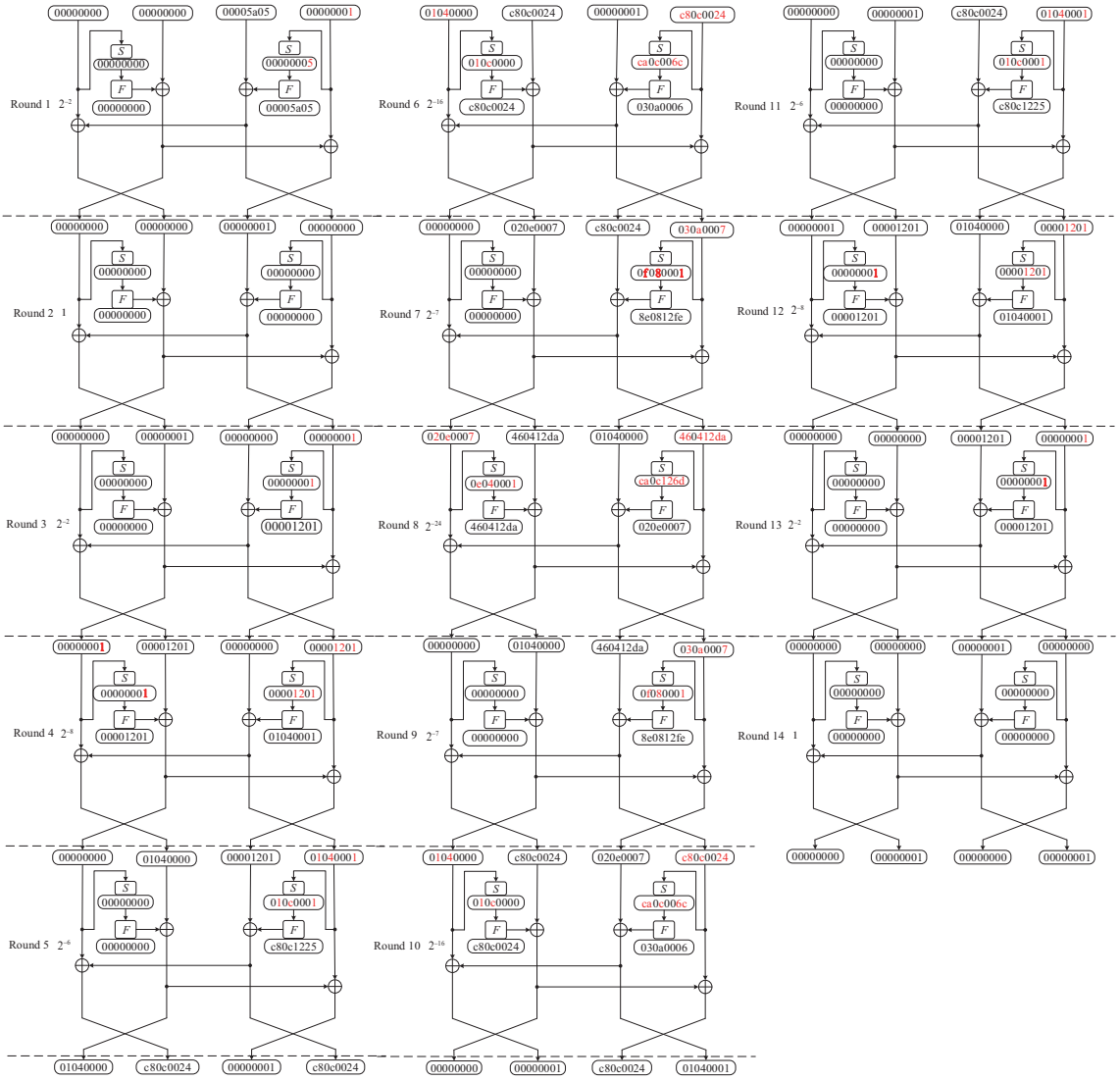

图 2　14 轮差分特征

Figure 2　14-round differential characteristic

表 2 中, 10 和 11 轮的 FBC-128 最优差分特征概率分别为 2^{-64} 和 2^{-72}, 而 $2^{-64}\times 2^{-72}=2^{-136}<2^{-128}$, 因此 FBC-128 不存在 21 轮的非平凡差分特征, 远小于文献 [1] 中估计的差分安全界 32 轮. 此外, 对于 12 和 13 轮, 搜索得到了概率分别为 2^{-83} 和 2^{-89} 的差分特征.

利用表 2 中的 1–11 轮最优差分特征概率, 以及我们搜索到的 12 和 13 轮的概率分别为 2^{-83} 和 2^{-89} 的差分特征, 结合 4.1 小节限制首尾两轮 S 盒数量的方法, 搜索 FBC-128 的 14 轮差分特征, 可以得到一条概率为 2^{-104}, 首尾两轮的扩展轮活跃 S 盒数为 5 的差分特征, 如图 2 所示. 其输入差分和输出差分分别为 (00000000, 00000000, 00005a05, 00000001) 和 (00000000, 00000001, 00000000, 00000001), 每个十六进制数表示对应 32 比特的字中第 $i, i+8, i+16, i+24$ $(0\leqslant i\leqslant 7)$ 比特的级联. 该差分特征共有 47 个活跃 S 盒, 其中大部分位于中间轮, 图 2 中以红色加粗数字标记了所有活跃 S 盒的输入和输出差分. 此外, 为了验证差分特征中不存在错误, 我们编写程序分别计算了差分特征中线性层和非

表 3 FBC-128 的 14 轮差分特征数量

Table 3 The numbers of differential characteristics of 14-round FBC-128

Probability	2^{-104}	2^{-105}	2^{-106}	2^{-107}	2^{-108}	2^{-109}	2^{-110}	2^{-111}	2^{-112}	2^{-113}	2^{-114}	2^{-115}	2^{-116}	2^{-117}	2^{-118}	2^{-119}	2^{-120}
Quantity	2	0	0	2	11	2	6	11	24	14	30	29	42	51	73	77	109

表 4 一条概率为 2^{-107} 的 14 轮差分特征

Table 4 One 14-round 2^{-107} differential characteristic

Round	Differential				Probability
1	00000000	00000000	00005a05	00000001	2^{-2}
2	00000000	00000000	00000001	00000000	1
3	00000000	00000001	00000000	00000001	2^{-2}
4	00000001	00001201	00000000	00001201	2^{-8}
5	00000000	01040000	00001201	01040001	2^{-6}
6	01040000	c80c0024	00000001	c80c0024	2^{-16}
7	00000000	020e0007	c80c0024	030a0007	2^{-7}
8	020e0007	460412da	01040000	460412da	2^{-24}
9	00000000	01040000	460412da	030a0007	2^{-6}
10	01040000	c20c007e	020e0007	c20c007e	2^{-19}
11	00000000	00000001	c20c007e	01040001	2^{-7}
12	00000001	00001201	01040000	00001201	2^{-8}
13	00000000	00000000	00001201	00000001	2^{-2}
14	00000000	00000000	00000001	00000000	1
Output	00000000	00000001	00000000	00000001	2^{-107}

线性层函数的输出值. 结果表明, 搜索到的差分特征是有效的差分特征.

4.3 从差分特征到差分

在得到 4.2 小节中 FBC-128 的 14 轮差分特征后, 可以将输入和输出差分固定, 继续搜索其他差分特征, 并将它们结合为差分, 从而提高差分概率. 采用类似 Sun 等 [13] 的方法, 每搜索到一条新的差分特征, 就将其作为一条新的 CNF 子句加入到 SAT 模型中, 从而将它排除, 直至整个 SAT 模型不满足为止.

对符合 $(00000000, 00000000, 00005a05, 00000001) \rightarrow (00000000, 00000001, 00000000, 00000001)$ 的所有概率不低于 2^{-120} 的差分特征进行搜索, 得到的结果如表 3 所示. 表 4 和 5 展示了其中两条概率为 2^{-107} 的差分特征.

由此可以计算出

$$(00000000, 00000000, 00005a05, 00000001) \rightarrow (00000000, 00000001, 00000000, 00000001).$$

的差分概率约为 $2^{-102.25}$.

此外, 为了验证差分概率的准确性, 使用改进的 Matsui 算法 [20] 对该差分中包含的差分特征重新进行搜索, 并利用中间相遇的时空折衷思想提高效率. 由于计算复杂度限制, 只搜索了差分概率不低于 2^{-108} 的特征, 并成功得到了概率为 2^{-104} 的 2 条特征, 概率为 2^{-107} 的 2 条特征和概率为 2^{-108} 的

表 5　另一条概率为 2^{-107} 的 14 轮差分特征

Table 5　Another 14-round 2^{-107} differential characteristic

Round	Differential				Probability
1	00000000	00000000	00005a05	00000001	2^{-2}
2	00000000	00000000	00000001	00000000	1
3	00000000	00000001	00000000	00000001	2^{-2}
4	00000001	00001201	00000000	00001201	2^{-8}
5	00000000	01040000	00001201	01040001	2^{-7}
6	01040000	c20c007e	00000001	c20c007e	2^{-19}
7	00000000	020e0007	c20c007e	030a0007	2^{-6}
8	020e0007	460412da	01040000	460412da	2^{-24}
9	00000000	01040000	460412da	030a0007	2^{-7}
10	01040000	c80c0024	020e0007	c80c0024	2^{-16}
11	00000000	00000001	c80c0024	01040001	2^{-6}
12	00000001	00001201	01040000	00001201	2^{-8}
13	00000000	00000000	00001201	00000001	2^{-2}
14	00000000	00000000	00000001	00000000	1
Output	00000000	00000001	00000000	00000001	2^{-107}

11 条特征, 与本文结果相同.

5　18 轮 FBC-128 的差分分析

5.1　密钥恢复攻击

基于 14 轮差分路线开展 FBC-128 的 18 轮差分攻击. 为了降低时间复杂度 (图 3), 预先计算以下表:

表 T_1: S 代表 FBC 算法的 S 盒, Δ_{in}, Δ_{out} 是 S 盒的输入和输出差分, 当 Δ_{in} 和 Δ_{out} 非零时, 平均有 1 个 x 值满足等式 $S(\Delta_{\text{in}} \oplus x) \oplus S(\Delta_{\text{in}}) = \Delta_{\text{out}}$. 建立表 T_1 存储 $(x, S(x))$ 的值, 并由 Δ_{in}, Δ_{out} 索引.

数据收集. 进行选择明文攻击, 收集满足输入和输出差分的明密文对. 定义 1 个明文结构, 满足

$$\Lambda = \{L(c_0c_1c_2c_3c_4c_5c_6a_0), L(c_7c_8c_9c_{10}a_1a_2c_{11}a_3), L(c_{12}a_4a_5a_6a_7a_8c_{13}a_9),$$
$$L(c_{14}c_{15}c_{16}c_{17}a_{10}a_{11}c_{18}a_{12})\} \in \left(\mathbb{F}_2^4\right)^{64},$$

其中 $c_0, \ldots, c_{18}$ 为固定常数, 遍历 $a_0, \ldots, a_{12}$ 的所有取值, 共 2^{52} 种取值. 选取 2^N 个结构体, 询问其 18 轮加密的密文, 用密文差分为 0 的半字节作为索引, 存储相应的明文值, 并对每个索引下的明密文进行两两组对并异或固定的差分值, 平均有 $2^{N+103-24\times4} = 2^{N+7}$ 对差分对, 对应的输出差分值 $(\Delta A_{18}, \Delta B_{18}, \Delta C_{18}, \Delta D_{18})$ 满足

$$(\Delta A_{18}, \Delta B_{18}, \Delta C_{18}, \Delta D_{18}) \in (L(0000**0*), L(0000000*), L(0000**0*), L(0000000*)),$$

对于过滤后的差分对继续执行后续的步骤.

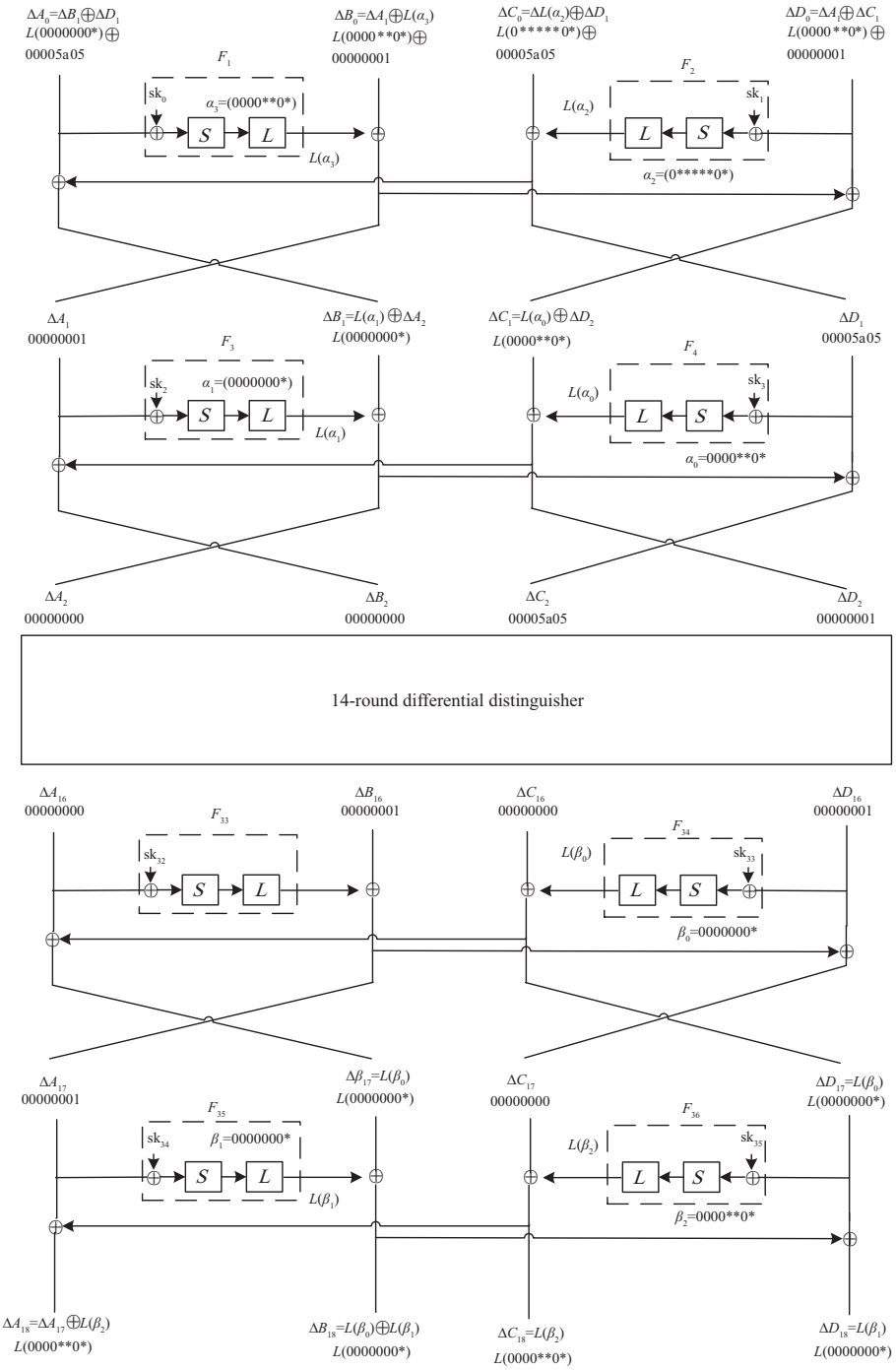

图 3 18 轮 FBC-128 差分分析

Figure 3 The differential cryptanalysis of 18-round FBC-128

密钥恢复. 密钥计算和猜测过程如下:

(1) 对于数据收集后获得的所有明密文对, 分别计算差分 ΔA_0, ΔB_0, ΔC_0 和 ΔD_0, 计算 $\alpha_3 = L^{-1}(\Delta B_0 \oplus \Delta A_1)$, 如果计算的差分值 α_3 在第 5, 6, 8 半字节活跃, 则通过 S 盒的输入输出差分查询

T_1 获得 S 盒的输入值, $\mathrm{sk}_0^{[4,5,7]}$ 通过 $A_0^{[4,5,7]}$ 的值与 S 盒的输入值异或得到. 计算 α_2, 如果计算的差分值 α_2 在第 2, 3, 4, 5, 6, 8 半字节活跃, 然后通过 S 盒的输入输出差分查询表 T_1 获得 S 盒的输入值, $\mathrm{sk}_1^{[1,2,3,4,5,7]}$ 通过 $D_0^{[1,2,3,4,5,7]}$ 的值与 S 盒的输入值异或得到. 对于尾部的差分, 通过密文对计算 ΔA_{18}, ΔB_{18}, ΔC_{18} 和 ΔD_{18}, 计算 β_2, 如果计算的差分值 β_2 在第 5, 6, 8 字节活跃, 查询 T_1 获得 S 盒的输入值, $\mathrm{sk}_{35}^{[4,5,7]}$ 通过 $B_{18}^{[4,5,7]} \oplus D_{18}^{[4,5,7]}$ 与 S 盒的输入值异或得到. 计算差分 β_1, 如果计算的差分 β_1 在第 8 字节活跃, 查询 T_1 获得 S 盒的输入值, $\mathrm{sk}_{34}^{[7]}$ 通过 $A_{18}^{[7]} \oplus C_{18}^{[7]}$ 与 S 盒的输入值异或得到. 分别将计算得到的 52 比特密钥作为索引, 保留下面的对, 平均每个索引下, 有 2^{N+7-52} 个对.

(2) 针对每一个密钥索引下的对, 猜测相关密钥, 具体猜测过程如下:

(a) 猜测 $\mathrm{sk}_0^{[0,1,2,6]}$, 加密第 1 轮, 计算差分 ΔC_1 和 ΔB_1. 然后猜测 $\mathrm{sk}_2^{[7]}$, $\mathrm{sk}_3^{[4,5,7]}$, 针对保留的每一个对, 加密第 2 轮并计算差分 ΔA_2 和 ΔD_2, 如果计算的差分 $\Delta A_2 = 00000000$, $\Delta D_2 = 00000001$, 则保留相应的对.

(b) 解密第 18 轮并计算差分 ΔD_{17}. 猜测 $\mathrm{sk}_{33}^{[7]}$, 解密第 17 轮并计算差分 ΔC_{16}, 如果计算的差分 $\Delta C_{16} = 00000000$, 则保留相应的对.

(c) 设置一个 36 比特的 $(\mathrm{sk}_0^{[0,1,2,6]}, \mathrm{sk}_2^{[7]}, \mathrm{sk}_3^{[4,5,7]}, \mathrm{sk}_{33}^{[7]})$ 计数器进行计数, 选取计数最高的密钥作为正确的轮密钥.

(3) 针对获得的 68 比特子密钥 $(\mathrm{sk}_0^{[0,1,2,4,5,6,7]}, \mathrm{sk}_1^{[1,2,3,4,5,7]}, \mathrm{sk}_2^{[7]}, \mathrm{sk}_3^{[4,5,7]})$, 穷搜其余 60 比特的子密钥, 并用一个明密文进行验证.

5.2 复杂度分析

数据复杂度. 数据收集阶段中的每一个结构体 Λ 有 13 个活跃半字节, 包括 2^{52} 个明文, 可以构造 $2^{52\times 2-1} = 2^{103}$ 个明文对. 选择 2^N 个结构体, 数据收集阶段获得 $2^{103+N} \times 2^{-4\times 24} = 2^{N+7}$ 对. 信噪比 $S_N = 4/(2^{8+16-36}) = 2^{14}$, 当 $N = 53$ 时, 正确密钥期待的计数 $\mu = 2^{103+N-52-102.25} \approx 4$.

根据文献 [21] 计算成功率

$$P_s = \Phi\left(\frac{\sqrt{\mu S_N} - \Phi^{-1}\left(1 - 2^{-h}\right)}{\sqrt{S_N + 1}}\right) \approx 0.98,$$

其中 $h = 1$, 因此数据复杂度为 $2^{53+52} = 2^{105}$ 选择明文量.

时间复杂度. 预计算阶段的时间复杂度相对于其他阶段可以忽略不计. 采用快速排序法 [22, 23] 进行数据收集阶段的过滤, 对于输出差分格式为 $(L(0000**0*), L(0000000*), L(0000**0*), L(0000000*))$ 的结构体进行筛选的复杂度为 $(\#\Lambda)\log_2(\#\Lambda) = 2^{52} \times 52 \approx 2^{57.7}$.

一共有 2^{53} 个明文结构, 总共有 $2^{53} \times 2^{57.7}$ 次筛选, 根据设计文档 [1] , 一轮实现需要 33 个基本操作, 因此, 数据收集阶段的时间复杂度相当于 $2^{53} \times 2^{57.7} \div (33 \times 18) \approx 2^{101.5}$ 次 18 轮加密.

密钥恢复阶段:

步骤 1. 对于过滤后 $2^{N+7} = 2^{60}$ 个对, 计算 $\mathrm{sk}_0^{[4,5,7]}$, $\mathrm{sk}_1^{[1,2,3,4,5,7]}$, $\mathrm{sk}_{34}^{[7]}$, $\mathrm{sk}_{35}^{[4,5,7]}$ 的时间复杂度约为 2^{60}, 对于每一个密钥值索引, 预计保留 $2^{60} \times 2^{-52} = 2^8$ 对.

步骤 2(a). 猜测 $\mathrm{sk}_0^{[0,1,2,6]}$ 并计算差分, 然后猜测 $\mathrm{sk}_2^{[7]}$, $\mathrm{sk}_3^{[4,5,7]}$ 并进行筛选, 预计保留 2^8 个明文对, 每个明文对对应 2^{16} 个 32 比特密钥. 时间复杂度约为 $2^{8+16+16+52} = 2^{92}$.

步骤 2(b). 猜测 $\mathrm{sk}_{33}^{[7]}$ 计算差分并过滤, 预计保留 2^8 个密文对, 每个密文对对应 1 个 4 比特密钥. 时间复杂度约为 $2^{8+4+52} = 2^{64}$.

步骤 2 的时间复杂度约为 $2^{64} + 2^{92} \approx 2^{92}$.

步骤 3. 穷搜其余 60 比特的子密钥, 计算复杂度约为 $2^{52+8+16-36}\times 2^{60}=2^{100}$.

18 轮密钥恢复攻击的时间复杂度约为 $2^{101.5}+2^{60}+2^{92}+2^{100}\approx 2^{101.5}$ 次 18 轮加密.

存储复杂度. 预计算表 T_1 需要 2^4 个内存单元. 在数据收集阶段, 快速排序方法的存储复杂度为 2^{52}, 密钥恢复阶段需要存储 2^{60} 个对, 存储复杂度约为 2^{52} (分组存储, 每个区块有 2^{52} 个内存单元, 并且可以重复利用). 此外密钥恢复阶段步骤 2(c) 还需要 2^{36} 比特的计数器, 因此密钥恢复攻击的存储复杂度约为 2^{52}.

6 20 轮 FBC128-256 的差分分析

6.1 密钥恢复攻击

基于上文中 FBC-128 的 14 轮差分路线, 可以给出 20 轮 FBC128-256 的密钥恢复攻击的结果 (图 4), 在区分器的头部和尾部各添加 3 轮, FBC128-256 的 20 轮差分攻击过程具体如下. 为了降低时间复杂度, 预先计算以下表:

表 T_1: S 代表 FBC 算法的 S 盒, $\Delta_{\rm in}$, $\Delta_{\rm out}$ 是 S 盒的输入和输出差分, 当 $\Delta_{\rm in}$ 和 $\Delta_{\rm out}$ 非零时, 平均有 1 个 x 值满足等式 $S(\Delta_{\rm in}\oplus x)\oplus S(\Delta_{\rm in})=\Delta_{\rm out}$. 建立表 T_1 存储 $(x,S(x))$ 的值, 并由 $\Delta_{\rm in}$, $\Delta_{\rm out}$ 索引.

数据收集. 进行选择密文攻击, 收集满足输入和输出差分的明密文对. 定义 1 个密文结构, 满足

$$\begin{aligned}\Lambda=\{&L\left(c_0c_1c_2c_3c_4c_5c_6a_0\right)\oplus\left(c_7a_1a_2a_3a_4a_5c_8a_6\right),L(c_9c_{10}c_{11}c_{12}a_7a_8c_{13}a_9),L(c_{14}c_{15}c_{16}c_{17}c_{18}c_{19}c_{20}a_{10}),\\&(c_{21}c_{22}c_{23}c_{24}a_{11}a_{12}c_{25}a_{13})\}\in\left(\mathbb{F}_2^4\right)^{64},\end{aligned}$$

其中 $c_0,\ldots,c_{25}$ 为固定常数, 遍历 $a_0,\ldots,a_{13}$ 所有取值, 共 2^{56} 种取值. 选取 2^N 个结构体, 询问其 20 轮解密的明文, 用明文差分为 0 的半字节作为索引, 存储相应的明文值, 对每个索引下的明密文进行两两组对并异或固定的差分值, 平均有 $2^{N+111-9\times4}=2^{N+75}$ 对. 对应的输出差分值满足

$$(\Delta A_0,\Delta B_0,\Delta C_0,\Delta D_0)\in(L(0000**0*),L(0*****0*),(*******),L(0*****0*)),$$

对于过滤后的差分对继续执行后续的步骤.

密钥恢复. 密钥计算和猜测过程如下:

(1) 计算差分 β_3, 如果计算的差分在第 5, 6, 8 半字节活跃, 查询 T_1 获得 S 盒的输入值, $\mathrm{sk}_{38}^{[4,5,7]}$ 通过 $A_{20}^{[4,5,7]}\oplus C_{20}^{[4,5,7]}$ 与 S 盒的输入值异或得到. 计算差分 β_4, 如果计算的差分值在第 2, 3, 4, 5, 6, 8 半字节活跃, 查询 T_1 获得 S 盒的输入值, $\mathrm{sk}_{39}^{[1,2,3,4,5,7]}$ 通过 $B_{20}^{[1,2,3,4,5,7]}\oplus D_{20}^{[1,2,3,4,5,7]}$ 与 S 盒的输入值异或得到. 计算差分 α_5, 如果计算的差分在第 2, 3, 4, 5, 6, 8 半字节活跃, 然后通过 S 盒的输入输出差分查询表 T_1 获得 S 盒的输入值, $\mathrm{sk}_0^{[1,2,3,4,5,7]}$ 通过 $A_0^{[1,2,3,4,5,7]}$ 的值与 S 盒的输入值异或得到. 计算差分 α_4, 如果计算的差分值在第 1, 2, 3, 4, 5, 6, 7, 8 半字节活跃, 则通过 S 盒的输入输出差分查询 T_1 获得 S 盒的输入值, sk_1 通过 D_0 的值与 S 盒的输入值异或得到. 分别将计算得到的 92 比特密钥作为索引, 保留下面的对, 平均每个索引下, 有 $2^{N+75-92}$ 个对.

(2) 针对每一个密钥索引下的对, 猜测相关密钥, 具体猜测过程如下:

(a) 计算差分 ΔA_{20}, ΔB_{20}, ΔC_{20} 和 ΔD_{20}. 解密第 20 轮并计算 ΔA_{19}, ΔD_{19}, ΔC_{19}. 猜测 $\mathrm{sk}_{36}^{[7]}$, $\mathrm{sk}_{37}^{[4,5,7]}$, 解密第 19 轮并计算差分 ΔB_{18}, ΔC_{18} 和 ΔD_{18}, 如果计算的差分 $\Delta C_{18}=00000000$, $\Delta B_{18}=L(0000000*)$ 则保留相应的对. 猜测 $\mathrm{sk}_{35}^{[7]}$, 解密第 18 轮并计算差分 ΔC_{17}, 如果计算的差分 $\Delta C_{17}=00000000$, 则保留相应的对.

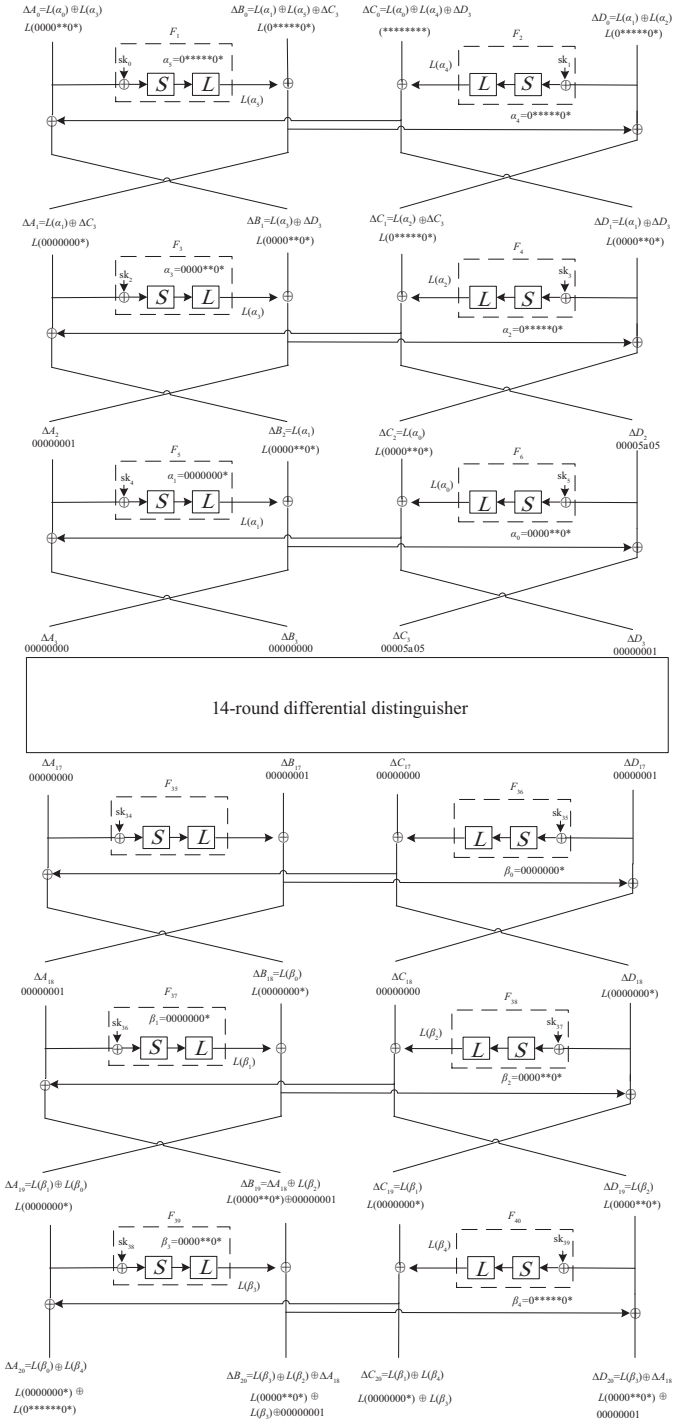

图 4　20 轮 FBC128-256 差分分析

Figure 4　The differential cryptanalysis of 20-round FBC128-256

(b) 猜测 $\mathrm{sk}_0^{[0,6]}$ 加密第 1 轮并计算差分 ΔB_1, ΔC_1. 猜测 $\mathrm{sk}_2^{[0,1,2,4,5,6,7]}$, $\mathrm{sk}_3^{[1,2,3,4,5,7]}$, 加密第 2 轮并计算差分 ΔA_2, ΔB_2, ΔC_2, ΔD_2 如果计算的差分 $\Delta A_2 = 00000001$, $\Delta D_2 = 00005a05$ 则保留相

应的对, 猜测 $\mathrm{sk}_4^{[7]}$, $\mathrm{sk}_5^{[4,5,7]}$, 加密第 3 轮并计算差分 ΔA_3, ΔD_3, 如果计算的差分 $\Delta A_3 = 00000000$, $\Delta D_3 = 00000001$ 则保留相应的对.

(c) 设置一个 96 比特的 $(\mathrm{sk}_{36}^{[7]}, \mathrm{sk}_{37}^{[4,5,7]}, \mathrm{sk}_{35}^{[7]}, \mathrm{sk}_0^{[0,6]}, \mathrm{sk}_2^{[0,1,2,4,5,6,7]}, \mathrm{sk}_3^{[1,2,3,4,5,7]}, \mathrm{sk}_4^{[7]}, \mathrm{sk}_5^{[4,5,7]})$ 计数器进行计数, 选取计数最高的密钥作为正确的轮密钥.

(3) 针对获得的 132 比特子密钥 $(\mathrm{sk}_0, \mathrm{sk}_1, \mathrm{sk}_2^{[0,1,2,4,5,6,7]}, \mathrm{sk}_3^{[1,2,3,4,5,7]}, \mathrm{sk}_4^{[7]}, \mathrm{sk}_5^{[4,5,7]})$, 穷搜其余 124 比特的子密钥, 并用一个明密文进行验证.

6.2 复杂度分析

数据复杂度. 数据收集阶段中的每一个结构体 Λ 有 14 个活跃半字节, 包括 2^{56} 个密文, 可以构造 $2^{56\times 2-1} = 2^{111}$ 个明文对. 选择 2^N 个结构体, 数据收集阶段获得 $2^{111+N} \times 2^{-4\times 9} = 2^{75+N}$ 对. 信噪比 $S_N = 4/(2^{32+4+24-96}) = 2^{38}$, 当 $N = 49$ 时, 正确密钥期待的计数 $\mu = 2^{111+N-56-102.25} \approx 4$.

根据文献 [21] 计算成功率

$$P_s = \Phi\left(\frac{\sqrt{\mu S_N} - \Phi^{-1}\left(1 - 2^{-h}\right)}{\sqrt{S_N + 1}}\right) \approx 0.98,$$

其中 $h = 1$, 因此数据复杂度为 $2^{49+56} = 2^{105}$ 选择明文量.

时间复杂度. 预计算阶段的时间复杂度相对于其他阶段可以忽略不计. 采用快速排序法 [22,23] 进行数据收集阶段的过滤, 对于输入差分格式为 $((L(0000{*}{*}0{*}), L(0{*}{*}{*}{*}{*}0{*}), L({*}{*}{*}{*}{*}{*}{*}), L(0{*}{*}{*}{*}{*}0{*}))$ 的结构体进行筛选的复杂度为 $(\#\Lambda)\log_2(\#\Lambda) = 2^{56} \times 56 \approx 2^{61.8}$.

一共有 2^{49} 个密文结构, 总共有 $2^{49} \times 2^{61.8}$ 次筛选, 根据设计文档 [1], 一轮实现需要 33 个基本操作, 因此, 时间复杂度相当于 $2^{49} \times 2^{61.8} \div (33 \times 20) \approx 2^{101.4}$ 次 20 轮加密.

步骤 1. 对于过滤后 $2^{N+75} = 2^{124}$ 个对, 计算 $\mathrm{sk}_{38}^{[4,5,7]}$, $\mathrm{sk}_{39}^{[1,2,3,4,5,7]}$, $\mathrm{sk}_0^{[1,2,3,4,5,7]}$, sk_1 的时间复杂度大概为 2^{124}. 对于每一个密钥值索引, 预计保留 $2^{124-92} = 2^{32}$.

步骤 2(a). 猜测 $\mathrm{sk}_{37}^{[4,5,7]}$, $\mathrm{sk}_{36}^{[7]}$ 计算差分并过滤, 预计保留 2^{32} 个对, 每个对下有 2^4 个 16 比特密钥, 时间复杂度大概为 $2^{32+16+92} = 2^{140}$. 猜测 $\mathrm{sk}_{35}^{[7]}$, 解密一轮计算差分并过滤, 预计保留 2^{32} 个对, 每个对下有 2^4 个 20 比特密钥, 时间复杂度约为 $2^{36+4+92} = 2^{132}$.

步骤 2(b). 猜测 $\mathrm{sk}_0^{[0,6]}$ 加密一轮, 预计保留 2^{32} 个对, 每个对下有 2^8 个 8 比特密钥, 继续猜测 $\mathrm{sk}_2^{[0,1,2,4,5,6,7]}$, $\mathrm{sk}_3^{[1,2,3,4,5,7]}$ 并过滤, 预计保留 2^{32} 个对, 每个对下有 2^{24} 个 60 比特密钥, 时间复杂度约为 $2^{32+8+52+92}$=2^{184}. 猜测 $\mathrm{sk}_4^{[7]}$, $\mathrm{sk}_5^{[4,5,7]}$ 并过滤, 预计保留 2^{32} 个对, 每个对下有 2^{24} 个 76 比特密钥, 时间复杂度约为 $2^{32+24+16+92}$=2^{164}.

步骤 2 的时间复杂度大约为 $2^{140} + 2^{132} + 2^{184} + 2^{164} \approx 2^{184}$.

步骤 3. 穷搜其余 124 比特的子密钥, 计算复杂度约为 $2^{92+32+4+24-96} \times 2^{124}$=$2^{180}$.

20 轮密钥恢复攻击的时间复杂度为 $2^{101.4} + 2^{124} + 2^{184} + 2^{180} \approx 2^{184}$ 次 20 轮加密.

存储复杂度. 预计算表 T_1 需要 2^4 个内存单元. 在数据收集阶段, 快速排序方法的存储复杂度为 2^{56}, 密钥恢复阶段需要存储 2^{124} 个对, 存储复杂度为 2^{56} (分组存储, 每个区块可以重复利用). 此外密钥恢复阶段步骤 2(c) 还需要 2^{96} 比特的计数器, 因此密钥恢复攻击的存储复杂度约为 2^{96}.

7 结论

本文致力于评估 FBC 算法在差分分析方面的安全性. 基于 SAT 自动搜索方法, 本文提出了约束首尾两轮活跃 S 盒数的模型, 搜索到了 FBC-128 新的 14 轮差分区分器, 概率为 $2^{-102.25}$. 基于

新的差分区分器, 本文给出了 18 轮 FBC128-128 和 20 轮 FBC128-256 的差分分析. 其中, 对 18 轮 FBC128-128 分析所需的数据复杂度为 2^{105}, 时间复杂度为 $2^{101.5}$ 次 18 轮加密, 存储复杂度为 2^{52}, 对于 20 轮 FBC128-256 分析所需的数据复杂度为 2^{105}, 时间复杂度为 2^{184} 次 20 轮加密, 存储复杂度为 2^{96}. 本文提出的差分分析覆盖轮数与全轮的 FBC 算法相比仍有一定距离, 没有威胁到 FBC 算法的安全性. Sun 等 [10] 提出的 SAT 自动搜索方法在布尔基数约束的编码上还有提升空间, 提出更好的编码方法, 进一步改进差分分析结果将是下一步研究的工作.

参考文献

1 Feng X T, Zeng X Y, Zhang F, et al. On the lightweight block cipher FBC. J Cryptologic Res, 2019, 6: 768–785
2 Mouha N, Wang Q J, Gu D W, et al. Differential and linear cryptanalysis using mixed-integer linear programming. In: Proceedings of the 7th China International Conference on Information Security and Cryptography, 2012. 57–76
3 Zhang Y, Liu G, Li C, et al. Impossible differential cryptanalysis of FBC-128. J Inf Secur Appl, 2022, 69: 103279
4 Ren B Q, Chen J G, Zhou S H, et al. Cryptanalysis of raindrop and FBC. In: Proceedings of the 13th International Conference on Network and System Security, 2019. 536–551
5 Sörensson N, Eén N. A SAT solver with conflict-clause minimization. In: Proceedings of the Theory and Applications of Satisfiability Testing, 2005. 1–2
6 Biere A. CaDiCaL at the SAT Race 2019. SAT Race 2019, 2019. https://cca.informatik.uni-freiburg.de/papers/Biere-SAT-Race-2019-solvers.pdf
7 Mouha N, Preneel B. Towards finding optimal differential characteristics for ARX: application to Salsa20. 2013. https://eprint.iacr.org/2013/328.pdf
8 Kölbl S, Leander G, Tiessen T. Observations on the SIMON block cipher family. In: Proceedings of the 35th Annual Cryptology Conference, 2015. 161–185
9 Liu Y W, Wang Q J, Rijmen V. Automatic search of linear trails in ARX with applications to SPECK and Chaskey. In: Proceedings of the 14th International Conference on Applied Cryptography and Network Security, 2016. 485–499
10 Sun L, Wang W, Wang M Q. Accelerating the search of differential and linear characteristics with the SAT method. IACR Trans Symmetric Cry, 2021. https://eprint.iacr.org/2021/213.pdf
11 冯秀涛, 曾祥勇, 张凡, 等. 一种分组密码 FBC 的实现方法及装置. 中国专利号: CN 110247754 A, 2019-09-17
12 Cook S A. The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971. 151–158
13 Sun L, Wang W, Wang M Q. More accurate differential properties of LED64 and Midori64. IACR Trans Symmetric Cry, 2018, 2018: 93–123
14 Brayton R K, Hachtel G D, McMullen C, et al. Logic Minimization Algorithms for VLSI Synthesis. Berlin: Springer, 1984
15 Sinz C. Towards an optimal CNF encoding of Boolean cardinality constraints. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2005. 827–831
16 Matsui M. On correlation between the order of S-boxes and the strength of DES. In: Proceedings of Workshop on the Theory and Application of of Cryptographic Techniques, 1995. 366–375
17 Biryukov A, Velichkov V, Le Corre Y. Automatic search for the best trails in ARX: application to block cipher Speck. In: Proceedings of the 23rd International Conference on Fast Software Encryption, 2016. 289–310
18 Huang M J, Wang L M. Automatic tool for searching for differential characteristics in ARX ciphers and applications. In: Proceedings of the 20th International Conference on Cryptology in India, 2019. 115–138
19 Liu Z, Li Y, Jiao L, et al. A new method for searching optimal differential and linear trails in ARX ciphers. IEEE Trans Inform Theory, 2020, 67: 1054–1068
20 Chen J, Teh J, Liu Z, et al. Towards accurate statistical analysis of security margins: new searching strategies for differential attacks. IEEE Trans Comput, 2017, 66: 1763–1777
21 Selçuk A A. On probability of success in linear and differential cryptanalysis. J Cryptol, 2008, 21: 131–147
22 Jiang Z L, Jin C H. Multiple impossible differentials cryptanalysis on 7-round ARIA-192. Secur Commun Netw, 2018, 2018: 1–11

23 Zhang Q G. Plaintext pair sieve methods in impossible differential attack. Comput Eng, 2010, 36: 127–129

Differential analysis of block cipher FBC

Duan LIU[1,5], Yibo LUO[2,7], Keting JIA[3,4], Guoyan ZHANG[1,5,6*], Guangnan ZOU[8], Qidi YOU[7] & Ying CHEN[9]

1. *School of Cyberspace Security, Shandong University, Qingdao* 266237*, China*;
2. *School of Cyberspace Security, University of Science and Technology of China, Hefei* 230022*, China*;
3. *Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084*, China*;
4. *Zhongguancun Laboratory, Beijing* 100095*, China*;
5. *Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Qingdao* 266237*, China*;
6. *Shandong Institute of Blockchain, Jinan* 250101*, China*;
7. *State Key Laboratory of Space-Ground Integrated Information Technology, Beijing Institute of Satellite Information Engineering, Beijing* 100086*, China*;
8. *Department of Computer Sciences and Technology, Tsinghua University, Beijing* 100084*, China*;
9. *Department of Cryptography Science and Technology, Beijing Electronic Science & Technology Institute, Beijing* 100070*, China*

* Corresponding author. E-mail: guoyanzhang@sdu.edu.cn

Abstract FBC is a lightweight block cipher algorithm with a simple structure and is flexible for implementation with hardware and software. It was one of the 10 algorithms that was promoted to the second round of the National Cryptographic Algorithm Design Competition held by the Chinese Cryptographic Association (CACR) in 2018. The block cipher FBC family includes three versions and supports 128- and 256-bit blocks and 128- and 256-bit keys. In this paper, we focus on the 128-bit versions. We develop a new 14-round differential path for FBC128-128 based on the SAT-based automatic search model, with a probability of $2^{-102.25}$. Based on this differential path, we perform differential cryptanalysis of 18-round FBC128-128 and 20-round FBC128-256 keys. The differential cryptanalysis of 18-round FBC128-128 costs the time complexity of $2^{101.5}$ and memory complexity of 2^{52}. For the differential analysis of 20-round FBC128-256, the time and memory complexities are 2^{184} and 2^{96}, respectively.

Keywords block cipher, differential cryptanalysis, FBC, Boolean satisfiability problem