

SMBA 分组密码算法 分析报告

兴唐通信科技有限公司

2019 年 2 月

目 次

1. 概述.....	1
2. 设计原理.....	1
2.1. 加解密过程设计原理.....	1
2.2. 密钥扩展设计原理.....	1
2.3. S 盒设计原理.....	2
2.4. 线性变换和换位变换设计原理.....	2
3. 安全性分析.....	3
3.1. 整体结构安全性.....	3
3.2. 密码部件安全性.....	3
3.2.1. 线性部件安全性.....	3
3.2.2. 非线性部件安全性.....	5
3.3. 抗攻击安全性.....	8
3.3.1. 差分攻击.....	8
3.3.2. 线性攻击.....	11
3.3.3. 代数攻击.....	13
3.3.4. 积分攻击.....	14
3.3.5. 不可能差分攻击.....	14
3.3.6. 相关密钥差分攻击.....	17
3.3.7. 飞去来器攻击.....	19
3.3.8. 滑动攻击.....	20
3.3.9. 密钥扩展安全性.....	21
3.4. 抗侧信道攻击安全防护.....	25
3.5. 依赖性检测.....	27
3.5.1. 检测项目.....	27
3.5.2. SMBA-128 检测结果及分析.....	28
3.5.3. SMBA-256 检测结果及分析.....	29
3.6. 随机性检测.....	30
3.6.1. 显著性水平.....	30

3.6.2.	检测项目	30
3.6.3.	密文随机性检测	30
3.6.4.	检测结果	31
4.	性能分析	38
4.1.	软件实现	38
4.1.1.	8 位平台实现	38
4.1.2.	32 位平台实现	39
4.1.3.	64 位平台实现	40
4.2.	硬件实现	42
4.2.1.	ASIC 评估结果	42
4.2.2.	FPGA 评估结果	44
5.	优缺点声明	45
6.	参考文献	46
附录 A	S_0 和 S_1 深度最优实现	47
附录 B	S_0 和 S_1 的一阶门限掩码方案	54
B.1.	S_0 的一阶门限掩码方案	54
B.2.	S_1 的一阶门限掩码方案	63

1. 概述

SMBA 是分组密码算法，分组长度支持 128 和 256 比特，分组长度为 128 比特时密钥长度支持 128 和 256 比特，分组长度为 256 比特时密钥长度为 256 比特，分别简记为 SMBA-128-128、SMBA-128-256、SMBA-256-256，迭代轮数分别为 18、24、24。在下文的分析中，当不需要考虑密钥长度时，也把 SMBA-128-128、SMBA-128-256 统一简称为 SMBA-128，把 SMBA-256-256 简称为 SMBA-256。

2. 设计原理

2.1. 加解密过程设计原理

SMBA加解密过程的设计原理如下：

- (1) 整体结构采用标准Feistel结构。
- (2) 轮函数采用SP结构。
- (3) 能抵抗差分/线性密码分析、代数攻击和积分攻击等已知攻击。
- (4) 所选算法的迭代轮数确保有较大的安全冗余。
- (5) 算法各轮输出的依赖性检测结果良好。
- (6) 算法输出的密文序列通过随机性检测。
- (7) SMBA-128与SMBA-256共用主要的密码部件。
- (8) 加密过程与解密过程仅是子密钥的顺序不同。
- (9) 基本运算为查表、固定位数的循环移位，以及异或等逻辑运算。
- (10) 具备软硬件多种平台上实现的有效性和灵活性。
- (11) 在算法设计上未隐藏任何“后门”。

2.2. 密钥扩展设计原理

SMBA密钥扩展的设计原理如下：

- (1) 一次加密子密钥、解密子密钥的生成速度分别高于一个分组加密、解密运算的速度。
- (2) 加密子密钥和解密子密钥均可在加解密过程中实时计算。
- (3) 密钥扩展与加解密过程共用S盒 S_0 和 S_1 ，但不共用线性变换和换位变换。

- (4) 能够抵抗与密钥扩展有关的攻击（相关密钥差分攻击、滑动攻击等），也没有等价密钥、弱密钥和半弱密钥。
- (5) SMBA-128-128和SMBA-256-256密钥扩展的结构本质上是广义Feistel结构[7]，SMBA-128-256密钥扩展是广义Feistel结构的推广。

2.3. S 盒设计原理

SMBA 所用的 8 比特 S 盒 S_0 和 S_1 通过两种不同的方式构造生成（设计思想分别参考了[2,3]和[1]），其用到的非线性运算为查 4 比特 S 盒， $GF(2^4)$ 上的乘法，以及多路选择器。相比于随机产生然后进行检测生成的 S 盒（还包括梯度下降、遗传算法等的迭代优化）， S_0 和 S_1 有如下优势：

- (1) 更好的密码强度指标。
- (2) 更优的硬件实现性能。
- (3) 更低的侧信道攻击防护代价。

还有一个很重要的方面是，可清楚地表明 S_0 和 S_1 的设计均无后门。

2.4. 线性变换和换位变换设计原理

SMBA-128的线性变换为 L_{64} ，SMBA-256的线性变换为两个并置的 L_{64} 。 L_{64} 的设计原理为：

- (1) L_{64} 、 L_{64}' 、 L_{64}'' 的差分分支数和线性分支数都为6。
- (2) L_{64} 、 L_{64}' 、 L_{64}'' 和 L_{64}^T 、 $(L_{64}')^T$ 、 $(L_{64}'')^T$ 都将1个非零字节变换到7个非零字节。
- (3) L_{64}' 、 $(L_{64}')^T$ 、 L_{64}'' 和 $(L_{64}'')^T$ 将1个32比特非零块变换到2个32比特非零块。
- (4) 不是面向字节的运算。
- (5) 各种软件平台（8、16、32、64比特）及硬件都可有效实现。

其中， L_{64}' 和 L_{64}'' 都在“3.3.1差分攻击”中定义，上标T表示转置。

SMBA-256换位变换 P_{128} 的选取确保算法的加解密过程都形成一个整体，对提升算法抗攻击能力起着关键的作用。

3. 安全性分析

3.1. 整体结构安全性

SMBA 的整体结构采用标准 Feistel 结构，Feistel 结构已广泛应用于分组密码算法的设计中，具有很好的结构安全性，可证明：在轮函数是伪随机函数的假设下，3 轮 Feistel 结构是伪随机置换，4 轮 Feistel 结构是超伪随机置换。

另外，对于 SMBA-256，若假设函数 $L_{64} \circ S_{64}$ 为伪随机函数（相对于轮函数是伪随机函数的假设是更弱、更合理的假设），则可证明：4 轮是伪随机置换，6 轮是超伪随机置换。

3.2. 密码部件安全性

3.2.1. 线性部件安全性

SMBA 的线性变换有 L_{32} 、 L_{64} 、 L_{128} 三种。

L_{32} 是 32 比特到 32 比特线性变换，具体为：将 32 比特数 x 分为 4 个 8 比特数，即 $x = x_0 || x_1 || x_2 || x_3$ ，令

$$y_0 = x_0 \oplus x_2 \oplus x_3$$

$$y_1 = x_1 \oplus x_2 \oplus x_3$$

$$y_2 = x_0 \oplus x_1 \oplus x_2$$

$$y_3 = x_0 \oplus x_1 \oplus x_3$$

则 $L_{32}(x) = y_0 || y_1 || y_2 || y_3$ 。

L_{32} 具有以下性质：

- (1) L_{32} 是对合变换，即 $L_{32}(L_{32}(x)) = x$ ；
- (2) L_{32} 的差分分支数和线性分支数都是 4。

L_{64} 是 64 比特到 64 比特的线性变换，在 L_{32} 的基础上构造，如图 3.2.1-1 所示，将 64 比特数 x 分为 2 个 32 比特数，即 $x = x_0 || x_1$ ，令

$$z_0 = L_{32}(x_0), \quad z_1 = L_{32}(x_1)$$

$$u = z_0 \oplus z_1$$

$$v = u \lll 9$$

$$y_0 = z_0 \oplus v, \quad y_1 = z_1 \oplus v$$

则 $L_{64}(x) = y_0 || y_1$ 。

L_{64} 具有以下性质：

- (1) L_{64} 的差分分支数和线性分支数都是 6；
- (2) $y = L_{64}(x)$, $w(x) = 1$ 时, $w(y) \geq 7$;
- (3) $y = L_{64}^T(x)$, $w(x) = 1$ 时, $w(y) \geq 7$;
- (4) 若 $x_0 = 0$, $x_1 \neq 0$, 则 $y_1 = 0 \Leftrightarrow x_1 = 0 \text{xxxxxxxx}$;
- (5) 若 $x_0 \neq 0$, $x_1 = 0$, 则 $y_0 = 0 \Leftrightarrow x_0 = 0 \text{xxxxxxxx}$;

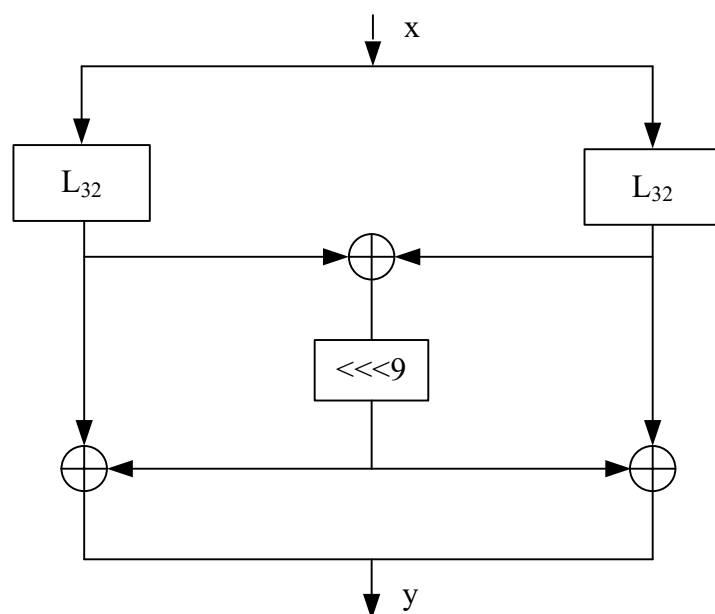


图 3.2.1-1 线性变换 L_{64}

L_{128} 是一个 128 比特到 128 比特的线性变换，由两个 L_{64} 并置组成，如图 3.2.1-2 所示，分支数等指标与 L_{64} 一致。

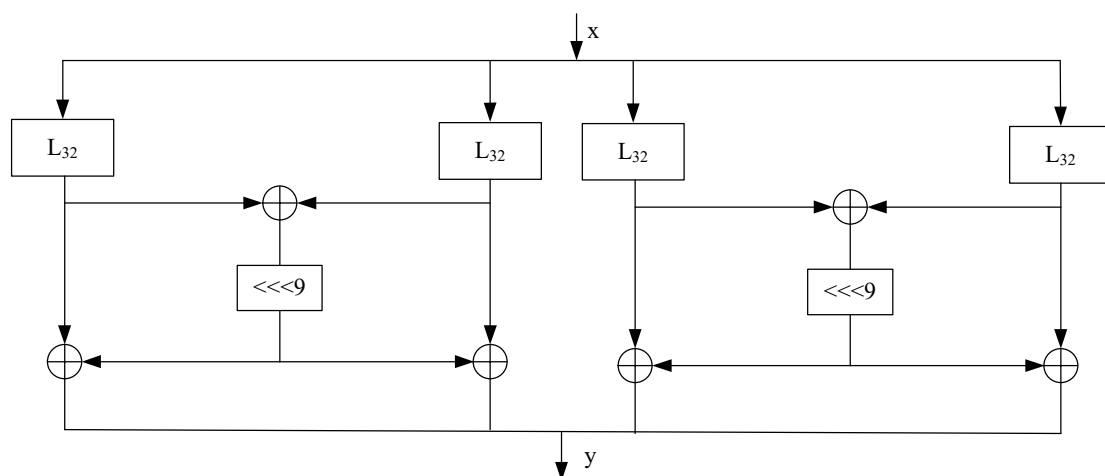


图 3.2.1-2 线性变换 L_{128}

3.2.2. 非线性部件安全性

3.2.2.1. 密码性质和指标定义

F_2^n 表示二元有限域 F_2 上的 n 维空间, $F_2^{n*} = F_2^n \setminus \{0\}$ 。设 $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_{n-1}) \in F_2^n$ 。

对于 n 比特到 1 比特的布尔函数 f , 定义 f 的 Walsh 变换

$$W_f(\omega) = \sum_{x \in F_2^n} (-1)^{f(x) \oplus \langle \omega, x \rangle}$$

其中, “ $\langle \omega, x \rangle$ ”表示 ω 与 x 的内积, 即 ω 与 x 逐比特与的异或。

f 的自相关变换

$$A_f(\omega) = \sum_{x \in F_2^n} (-1)^{f(x) \oplus f(x \oplus \omega)}$$

f_1 和 f_2 的互相关变换

$$C_{f_1, f_2}(\omega) = \sum_{x \in F_2^n} (-1)^{f_1(x) \oplus f_2(x \oplus \omega)}$$

对于 n 比特到 m 比特的多元布尔函数 $S = (S_0, S_1, \dots, S_{m-1})$ (也称为 (n, m) S 盒), 下面是一些与侧信道攻击相关指标的定义。

S 的透明阶 (Transparency Order)

$$TO(S) = m - \frac{1}{2^{2n} - 2^n} \sum_{\alpha \in F_2^{n*}} \left| \sum_{i=0}^{m-1} A_{S_i}(\alpha) \right|$$

S 的改进的透明阶 (Improved Transparency Order)

$$iTO(S) = \max_{\beta \in F_2^m} \left(m - \frac{1}{2^{2n} - 2^n} \sum_{\alpha \in F_2^{n*}} \sum_{j=0}^{m-1} \left| \sum_{i=0}^{m-1} (-1)^{\beta_i \oplus \beta_j} C_{S_i, S_j}(\alpha) \right| \right)$$

S 的 DPA 信噪比 (DPA Signal-to-Noise Ratio)

$$SNR_DPA(S) = n 2^{2n} \left(\sum_{\alpha \in F_2^n} \left(\sum_{i=0}^{m-1} W_{S_i}(\alpha) \right)^4 \right)^{-\frac{1}{2}}$$

S 相对于 Hamming 重量的混乱系数方差 (confusion coefficient variance)

$$CC(S) = \sigma^2[\kappa(k_0, k_1) : \forall k_0, k_1 \in F_2^n, k_0 < k_1]$$

其中， $\sigma^2[\cdot]$ 表示方差， $\kappa(k_0, k_1) = E_p[(L(S(k_0 \oplus p)) - L(S(k_1 \oplus p)))^2]$ ， p 遍历 S 的定义域， L 表示泄露函数（这里取 Hamming 重量）， E 是期望运算。显然有 $\kappa(k_0, k_1) = \kappa(k_0 \oplus k_1, 0)$ ，据此可简化 $CC(S)$ 的计算。

其他密码性质和指标的定义如下：

- (1) 平衡性：S 盒 s 是双射（即可逆的），则称 s 满足平衡性。
- (2) 完全性：S 盒 s 的任一输出比特与任一输入比特都有关，则称 S 满足完全性。
- (3) 代数次数：设 S 盒 $s=(f_0, f_1, \dots, f_7)$ ，则 8 个 $F_2^8 \rightarrow F_2$ 的函数 f_0, f_1, \dots, f_7 称为 s 的坐标函数， f_0, f_1, \dots, f_7 的所有非零线性组合的代数次数的最小值，称为 s 的代数次数。
- (4) 非线性度：对于 S 盒 s ，其非线性度定义为

$$NL(S) = \min \left\{ \min_{g \in A_n} wt(v \cdot S - g) \mid 0 \neq v \in F_2^8 \right\}$$

其中， A_n 为所有仿射函数的集合。

- (5) 最大差分概率：对于 S 盒 s ，对 $a, b \in F_2^8$ 定义

$$DP^s(a, b) = \frac{\#\{x \in F_2^8 : s(x) \oplus s(x \oplus a) = b\}}{2^8}$$

$$p_s = \max_{a \neq 0, b} DP^s(a, b)$$

p_s 为 s 最大差分概率。其中，“ $\#$ ”表示集合中元素的个数。

- (6) 最大线性概率：对于 S 盒 s ，对 $a, b \in GF(2)^8$ 定义

$$LP^s(a, b) = (2 \times \frac{\#\{x \in F_2^8 : \langle a, x \rangle = \langle b, s(x) \rangle\}}{2^8} - 1)^2$$

$$q_s = \max_{a, b \neq 0} LP^s(a, b)$$

q_s 为 s 的最大线性概率。

- (7) 对于 S 盒 s ，若 $GF(2)$ 中存在不全为 0 的常数 $a_i, b_i, c_{i,j}, d_{i,j}, e_{i,j}, f$ ， $0 \leq i, j \leq 7$ ，使得对所有 $x_i \in GF(2)$ ，及相应的 $y_i \in GF(2)$ ， $0 \leq i \leq 7$ ，

$$(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) = s(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7),$$

满足

$$\sum_{0 \leq i \leq 7} a_i x_i + \sum_{0 \leq i \leq 7} b_i y_i + \sum_{0 \leq i < j \leq 7} c_{i,j} x_i x_j + \sum_{0 \leq i < j \leq 7} d_{i,j} y_i y_j + \sum_{0 \leq i \leq 7, 0 \leq j \leq 7} e_{i,j} x_i y_j + f = 0$$

则称 s 存在二次关系。三次关系类似定义。

- (8) 对于 S 盒 s ，若 x 满足 $s(x)=x$ ，则 x 称为 s 的一个不动点。

3.2.2.2. S_0 和 S_1 的密码性质和指标

SMBA 算法使用了 2 个 8 比特到 8 比特的 S-盒 S_0 和 S_1 ，按密码性质和指标如下表：

表 3.2.2.2-1: S_0 和 S_1 的密码学性质

	S_0	S_1
不动点	无	无
平衡性	平衡	平衡
完全性	完全	完全
代数次数	7	7
最大差分概率	$2^{-5.415}$	2^{-6}
最大线性概率	$2^{-5.356}$	2^{-6}
非线性度	108	112
二次关系	不存在	有 39 个线性无关的二次关系
三次关系	有 441 个线性无关的三次关系	有 471 个线性无关的三次关系
坐标函数代数正	121、120、129、119、	131、116、124、135、

规型中非 0 项数	125、127、118、117	114、133、132、120
透明阶	7.770	7.857
改进的透明阶	6.852	6.940
DPA 信噪比	7.470	9.059
Hamming 重量的 混乱系数方差	0.225	0.133

其中，透明阶、改进的透明阶、DPA信噪比、Hamming重量的混乱系数方差是为了评估S盒抵抗侧信道攻击（SCA）的能力引入的指标。通常认为，透明阶、改进的透明阶和DPA信噪比越小，表明S盒抵抗差分能量攻击（DPA）的能力越强；Hamming重量的混乱系数方差越大，表明S盒抵抗侧信道攻击越强。AES算法S盒的透明阶、改进的透明阶、DPA信噪比、Hamming重量的混乱系数方差分别为7.860、6.916、9.600、0.111，根据表3.2.2.2-1知， S_0 和 S_1 比AES算法的S盒具有更强的抗侧信道攻击的能力（唯一的例外是AES算法S盒改进的透明阶小于 S_1 改进的透明阶）。

3.3. 抗攻击安全性

3.3.1. 差分攻击

差分攻击是现在分组密码分析中最有效的手段之一，安全的分组密码算法要求能够抵抗差分攻击。评估分组密码算法抵抗差分攻击的能力主要有计算密码算法的差分概率和最大差分特征概率两种方式，对于64比特以上分组长度算法，计算其差分概率的复杂度很高，在实际评估中难以实现，因此在实际密码算法分析中一般采用最大差分特征概率来评估密码算法抵抗差分攻击的强度。从算法设计角度来说，经常采用截断差分的思想评估最大差分特征概率的上界，只要能够保证算法的最大差分特征概率的上界足够小，即可认为算法能够抵抗差分攻击。

这里采用截断差分的思想分析SMBA算法抵抗差分攻击的能力。由线性变换 L_{64} 的性质分析知，若 L_{64} 的输入 x_0 、 x_1 存在一个为0一个非0时，只有一种输入使得输出 y_0 、 y_1 不全非0，在截断差分攻击分析中难以体现线性变换这种性质。为了能够较精确评估SMBA算法最大差分特征概率的下界，按照图3.3.1-1和3.3.1-2

所示的等价结构分析SMBA-128（线性变换记为 L_{64}' ）和SMBA-256（线性变换记为 L_{64}' 和 L_{64}'' ）的最大差分特征概率，在后面的线性攻击和相关密钥差分攻击也将按照这种等价结构进行分析。

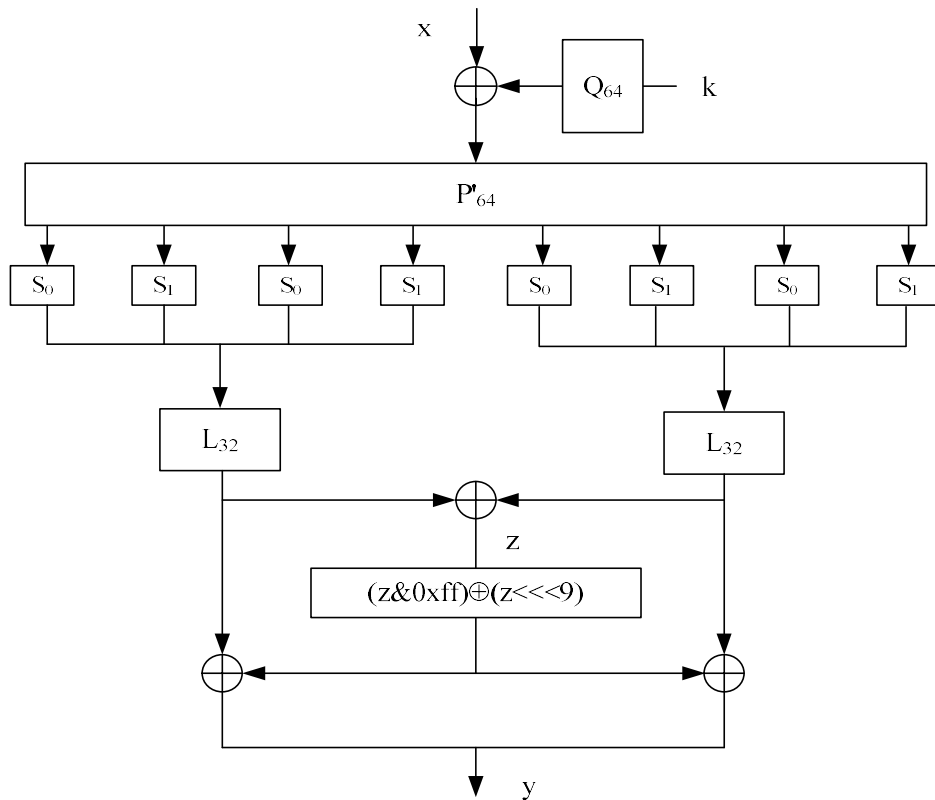


图3.3.1-1 SMBA-128轮函数等价结构

其中， P'_{64} 为：7,4,5,6,0,1,2,3， Q_{64} 为：0,1,2,7,4,5,6,3。

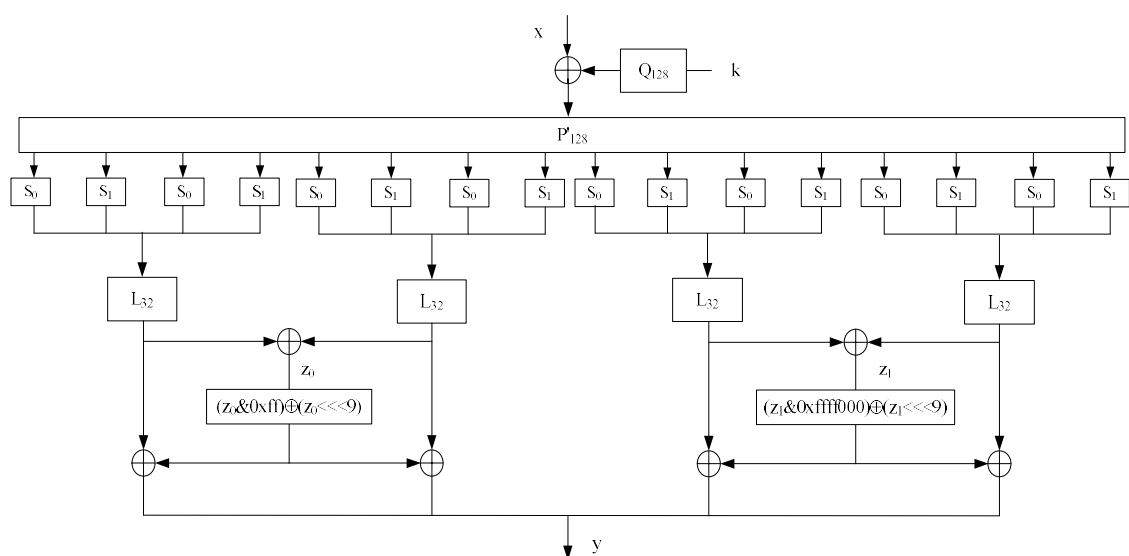


图3.3.1-2 SMBA-256轮函数等价结构

其中, P'_{128} 为: 7,4,5,6,10,11,8,9,12,13,14,15,0,1,2,3, Q_{128} 为: 0,1,2,7,4,5,6,3,12,13,10,11,8,9,14,15。

Mouha在[5]中给出了一种采用混合整数规划模型搜索差分活动的方法, 本文采用这种方式分别评估SMBA-128和SMBA-256具有的差分活动S盒数的下界, 结果分别如表3.3.1-1和表3.3.1-2所示。

表3.3.1-1 SMBA-128差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	0	1	2	6	8	10	11	14	16	18	20	22
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	24	26	28	30	32	34	36	38	40	42	44	46

可以证明, 对于所有整数 $n \geq 8$, SMBA-128任意连续 n 轮至少有 $2(n-1)$ 个差分活动S盒。

表3.3.1-2 SMBA-256差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	0	1	2	7	10	14	16	19	21	25	27	31
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	34	36	38	42	44	48	50	52	56	58	60	64

由表3.2.2.2-1知, S_0 的最大差分概率为 $2^{-5.415}$, S_1 的最大差分概率为 2^{-6} , 因此13轮SMBA-128算法的最大差分特征概率不大于 $2^{-5.415 \times 24} = 2^{-129.96}$, 18轮SMBA-256算法的最大差分特征概率不大于 $2^{-5.415 \times 48} = 2^{-259.92}$, 表明SMBA-128和SMBA-256都能够抵抗差分攻击, 且具有足够的安全冗余。

根据活动S盒数计算最大差分特征概率的方式, 没有详细区分差分特征中具体活动S盒是 S_0 或 S_1 , S_0 和 S_1 的最大差分概率不同导致这种评估结果与真实的最大差分特征概率差异较大, 不能有效体现算法抗差分攻击的强度。可以在上述计算差分活动S盒的模型的目标函数中, 以 S_0 和 S_1 的最大差分概率进行加权, 直接评估算法的最大差分特征概率, 具体结果如表3.3.1-3和3.3.1-4所示。

表3.3.1-3 SMBA-128差分特征概率上界

轮数	1	2	3	4	5	6	7	8
差分特征 概率上界	0	$2^{-5.415}$	$2^{-10.830}$	$2^{-32.490}$	$2^{-43.320}$	$2^{-54.150}$	$2^{-61.320}$	$2^{-75.810}$
轮数	9	10	11	12	13	14	15	16
差分特征 概率上界	$2^{-86.640}$	$2^{-97.470}$	$2^{-108.300}$	$2^{-119.130}$	$2^{-129.960}$	$2^{-140.790}$	$2^{-151.620}$	$2^{-162.450}$
轮数	17	18	19	20	21	22	23	24
差分特征 概率上界	$2^{-173.280}$	$2^{-184.110}$	$2^{-194.940}$	$2^{-205.770}$	$2^{-216.600}$	$2^{-227.430}$	$2^{-238.260}$	$2^{-249.090}$

表3.3.1-4 SMBA-256差分特征概率上界

轮数	1	2	3	4	5	6	7	8
差分特征 概率上界	0	$2^{-5.415}$	$2^{-10.830}$	$2^{-38.490}$	$2^{-55.320}$	$2^{-76.395}$	$2^{-87.810}$	$2^{-105.225}$
轮数	9	10	11	12	13	14	15	16
差分特征 概率上界	$2^{-116.64}$	$2^{-137.715}$	$2^{-149.130}$	$2^{-170.790}$	$2^{-184.110}$	$2^{-194.940}$	$2^{-210.450}$	$2^{-232.110}$
轮数	17	18	19	20	21	22	23	24
差分特征 概率上界	$2^{-244.110}$	$2^{-264.600}$	$2^{-276.600}$	$2^{-288.600}$	$2^{-309.090}$	$2^{-321.090}$	$2^{-333.090}$	$2^{-354.165}$

由表3.3.1-3和3.3.1-4知，SMBA-128经过13轮迭代后最大差分特征概率不大于 $2^{-129.96}$ ，小于安全界 2^{-128} ；SMBA-256经过18轮迭代后最大差分特征概率不大于 $2^{-264.600}$ ，小于安全界 2^{-256} 。

3.3.2. 线性攻击

线性攻击也是当前分组密码中最有效的分析手段之一，其分析方法和差分攻击类似，可以通过计算线性活动S盒的数量和直接计算线性特征概率的方式评估

最大线性特征概率，具体过程与差分攻击类似，这里只给出分析的结果，详见表3.3.2-1、3.3.2-2、3.3.2-3和3.3.2-4。

表3.3.2-1 SMBA-128线性活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
线性活动S盒	0	1	2	6	8	10	11	14	16	18	20	22
轮数	13	14	15	16	17	18	19	20	21	22	23	24
线性活动S盒	24	26	28	30	32	34	36	38	40	42	44	46

可以证明，对于所有整数 $n \geq 8$ ，SMBA-128任意连续 n 轮至少有 $2(n-1)$ 个线性活动S盒。

表3.3.2-2 SMBA-256线性活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
线性活动S盒	0	1	2	7	10	14	16	19	21	25	27	31
轮数	13	14	15	16	17	18	19	20	21	22	23	24
线性活动S盒	34	36	38	42	44	48	50	52	56	58	60	64

由表3.2.2.2-1知， S_0 的线性概率为 $2^{-5.356}$ ， S_1 的线性概率为 2^{-6} ，因此13轮SMBA-128算法的最大线性特征概率不大于 $2^{-5.356 \times 24} = 2^{-128.544}$ ，18轮SMBA-256算法的最大线性特征概率不大于 $2^{-5.356 \times 48} = 2^{-257.088}$ ，表明SMBA-128和SMBA-256都能够抵抗线性攻击，且具有足够的安全冗余。

表3.3.2-3 SMBA-128线性特征概率上界

轮数	1	2	3	4	5	6	7	8
线性特征 概率上界	0	$2^{-5.356}$	$2^{-10.712}$	$2^{-32.136}$	$2^{-42.848}$	$2^{-53.560}$	$2^{-60.848}$	$2^{-74.984}$
轮数	9	10	11	12	13	14	15	16
线性特征 概率上界	$2^{-85.696}$	$2^{-96.408}$	$2^{-107.120}$	$2^{-117.832}$	$2^{-128.544}$	$2^{-139.256}$	$2^{-149.968}$	$2^{-160.680}$
轮数	17	18	19	20	21	22	23	24
线性特征	$2^{-171.392}$	$2^{-182.104}$	$2^{-192.816}$	$2^{-203.528}$	$2^{-214.240}$	$2^{-224.952}$	$2^{-235.664}$	$2^{-246.376}$

概率上界								
------	--	--	--	--	--	--	--	--

表3.3.2-4 SMBA-256线性特征概率上界

轮数	1	2	3	4	5	6	7	8
线性特征 概率上界	0	$2^{-5.356}$	$2^{-10.712}$	$2^{-38.136}$	$2^{-54.848}$	$2^{-75.628}$	$2^{-86.984}$	$2^{-104.340}$
轮数	9	10	11	12	13	14	15	16
线性特征 概率上界	$2^{-115.696}$	$2^{-136.476}$	$2^{-147.832}$	$2^{-169.256}$	$2^{-182.104}$	$2^{-192.816}$	$2^{-208.680}$	$2^{-230.104}$
轮数	17	18	19	20	21	22	23	24
线性特征 概率上界	$2^{-242.104}$	$2^{-262.240}$	$2^{-274.240}$	$2^{-286.240}$	$2^{-306.376}$	$2^{-318.376}$	$2^{-330.376}$	$2^{-351.356}$

由表3.3.2-3和3.3.2-4知，SMBA-128经过13轮迭代后最大线性特征概率不大于 $2^{-128.544}$ ，小于安全界 2^{-128} ；SMBA-256经过18轮迭代后最大线性特征概率不大于 $2^{-262.240}$ ，小于安全界 2^{-256} 。

3.3.3. 代数攻击

Courtois和Pieprzyk在[4]中利用AES中S盒存在的二次关系（有39个线性无关的二次关系），把破译AES转化为求解GF(2)上的一个超定且稀疏的多元二次方程组，具体地：128比特密钥的AES的破译可转化为从一个明密文对求解GF(2)上的一个1600个变量，8000个方程的二次方程组。理论上，解一个一般的多元二次方程组是NP完全问题，没有有效的求解方法，但当方程组是超定的（即方程的数量多于未知元的数量）时，则其求解就要容易得多，如果还是稀疏的（即非零系数项很少），则求解更容易。针对由AES算法构建的二次方程组，Courtois和Pieprzyk提出了一个称为XSL的算法来求解，他们估计破译分组和密钥长度都为128比特的AES 在已知1个明密文对时用大约 2^{100} 数量级的计算，这就优于穷搜密钥的蛮力攻击，但在是分析中并没有明显的优势，许多密码分析专家也不认可这种攻击方式。

SMBA算的 S_0 不存在二次关系， S_i 存在39个二次关系，可构造的二次方程数

量低于AES，涉及的变量数不低于AES，使用XL算法求解构造的二次方程组求解复杂度应高于AES，可认为SMBA能够抵抗代数攻击。

3.3.4. 积分攻击

Todo[8]提出的基于可除性(Division Property)的积分分析是积分分析发展中的一个里程碑，不同于之前的积分分析，这种分析方法涉及到了算法非线性变换的代数次数等性质。

对于 128 比特分组的 SMBA-128 算法，令算法输入的多重集合具有可除性 $D_K^{8,16}$ ， $K=(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, 8, 8, 8, 8, 8, 8, 8, 8)$ ， $\sum_{i=0}^7 k_i = 61$ ，则经 7 轮加密后输出的多重集合的第 9 到第 16 个字节为平衡字节，而经 8 轮加密后输出的多重集合的每个字节都是不确定字节，即算法存在 7 轮积分区分器，可认为积分攻击对算法 SMBA-128 是无效的。

对于 256 比特分组的 SMBA-256 算法，令算法输入的多重集合具有可除性 $D_K^{8,32}$ ， $K=(k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8)$ ， $\sum_{i=0}^{15} k_i = 126$ ，则经 8 轮加密后输出的多重集合的第 17 到第 32 个字节为平衡字节，而经 9 轮加密后输出的多重集合的每个字节都是不确定字节，即算法存在 8 轮积分区分器，可认为积分攻击对 SMBA-256 是无效的。

3.3.5. 不可能差分攻击

不可能差分攻击利用概率为0的差分特征进行攻击，影响不可能差分攻击的主要因素是不可能差分特征的长度。评估密码算法迭代结构的不可能差分特征的方法，主要有U方法和PID，这两种评估方法都不考虑算法的设计细节，评估的结果可能与真实的不可能差分特征轮数有差异。文献[9]中介绍了一种利用方程组评估不可能差分特征轮数的方法，此方法充分考虑了算法内线性部件的设计细节，能够给出较精确的评估结果，根据此方法，我们分析了SMBA算法的不可能差分特征情况，具体结论如下。

对于SMBA-128算法，只找到了5轮的不可能差分特征，与Feistel结构天然具有的不可能差分特征一致。

对于SMBA-256算法，找到了17786条8轮不可能差分特征，没有找到更多轮

数的不可能差分特征，为了验证程序的正确性，对搜索程序给出的第一条8轮不可能差分特征进行了证明。

性质3.3.5-1: $(1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000) \not\rightarrow (0000\ 0000\ 0000\ 0000\ 0100\ 0000\ 0000\ 0000)$ 是SMBA-256算法的8轮不可能差分特征。

证明：令 $(X_{16r}\ X_{16r+1}\ X_{16r+2}\dots X_{16r+15},\ X_{16r+16}\ X_{16r+17}\dots X_{16r+31})$ 表示第 r 轮的输入截断差分， $(y_{16r}\ y_{16r+1}\ y_{16r+2}\dots y_{16r+15})$ 表示第 r 轮线性变换的输入截断差分， $r=0,1,\dots,7$ 。现在证明性质3.3.5-1成立，图3.3.5-1给出了这条不可能差分特征的演化过程。

加密方向： $(X_0\dots X_{15},\ X_{16}\dots X_{31})=(1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000)$ ， $(X_{32}\dots X_{47})=(1000\ 0000\ 0000\ 0000)$ ， $(y_{16}\dots y_{31})=(0000\ 0000\ 0000\ 1000)$ ，则 $(X_{48}\dots X_{55})=(0000\ 0000)$ ， $(X_{56}\dots X_{63})=L_{64}(0000\ 1000)$ 。根据线性变换 L_{64} 的定义计算可得， $L_{64}(0000\ 1000)$ 的输出截断差分只有 $(0111\ 1111)$ 、 $(1111\ 1101)$ 、 $(1111\ 1111)$ 和 $(1110\ 1111)$ 四种情况，可简记为 $(?11?\ 11?1)$ ，即 $(X_{56}\dots X_{63})=(?11?\ 11?1)$ ，根据S层的定义可知 $(y_{32}\dots y_{47})=(0000\ 1?11\ 1?11\ 0000)$ 。

解密方向： $(X_{128}\dots X_{143},\ X_{144}\dots X_{159})=(0000\ 0000\ 0000\ 0000\ 0100\ 0000\ 0000\ 0000)$ ，由第6轮的S层变换可知， $(y_{96}\dots y_{111})=(0000\ 0000\ 0000\ 0100)$ ，因此 $(X_{96}\dots X_{103})=(0000\ 0000)$ ， $(X_{104}\dots X_{111})=L_{64}(0000\ 0100)$ 。根据线性变换 L_{64} 的定义计算可得， $L_{64}(0000\ 0100)$ 的输出截断差分只有 $(1110\ 1111)$ 、 $(1101\ 1111)$ 、 $(1111\ 1111)$ 和 $(1111\ 1011)$ 四种情况，可简记为 $(11??\ 1?11)$ ，即 $(X_{104}\dots X_{111})=(11??\ 1?11)$ ，可知 $(y_{80}\dots y_{95})=(0000\ ??1?\ 1111\ 0000)$ 。

由第2、3、4、5、6轮的差分迭代关系可知：

$$(X_{48}\dots X_{55})\oplus L_{64}(y_{48}\dots y_{55})=L_{64}(y_{80}\dots y_{87})\oplus (X_{112}\dots X_{119})$$

$$(X_{56}\dots X_{63})\oplus L_{64}(y_{56}\dots y_{63})=L_{64}(y_{88}\dots y_{95})\oplus (X_{120}\dots X_{127})$$

即

$$L_{64}(y_{48}\dots y_{55})\oplus L_{64}(y_{80}\dots y_{87})=(0100\ 0000)$$

$$L_{64}(0000\ 1000)\oplus L_{64}(y_{56}\dots y_{63})=L_{64}(1111\ 0000)$$

可得

$$(y_{48}y_{49}y_{50}y_{51})\|(y_{52}y_{53}y_{54}y_{55}\oplus y_{84}y_{85}y_{86}y_{87})=L^{-1}_{64}(0100\ 0000)$$

$$(y_{56}\dots y_{63})=(1111\ 1000)$$

根据线性变换 L_{64} 的定义计算可得， $L^{-1}_{64}(0100\ 0000)$ 的输出截断差分只有 $(1111\ 1011)$ 、 $(1111\ 1101)$ 、 $(1111\ 1111)$ 和 $(1110\ 1111)$ 四种情况，可简记 $L^{-1}_{64}(0100$

0000)=(111? 1??1), 因此 $y_{48}=y_{49}=y_{50}=1$, 由S层的逆变换可得 $(x_{64}...x_{71})=(1001\ 11?0)$ 。由第1、2、3轮的迭代关系知: $(x_{32}...x_{39})\oplus L_{64}(y_{32}...y_{39})=(x_{64}...x_{67})$, 即 $L_{64}(0000\ 1?11)=(1000\ 0000)\oplus(1001\ 11?0)=(?001\ 11?0)$ 。记 $L_{64}(0000\ 1?11)$ 的输出截断差分为 $(z_0...z_7)$, 若限定 $z_1=z_2=z_7=0$, 通过计算可知, 此时 $L_{64}(0000\ 1?11)$ 的输出截断差分只有一种情况(1000 1110), 与 $x_{67}=1$ 矛盾。定理成立。

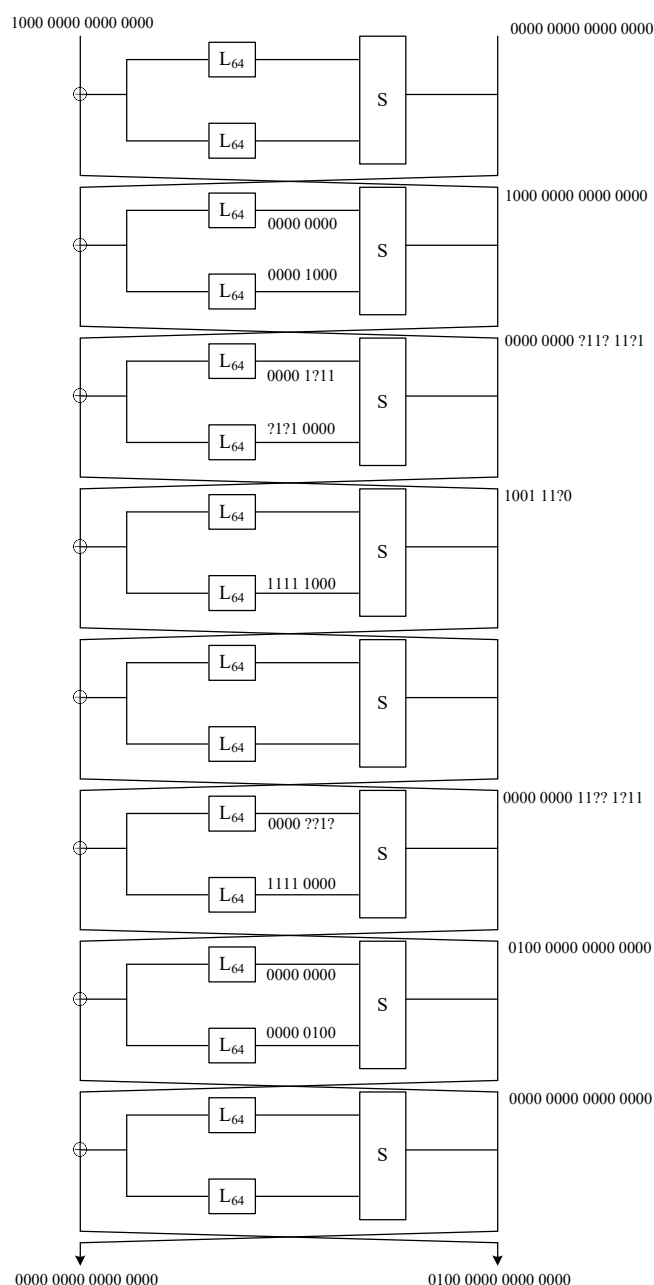


图3.3.5-1 SMBA-256的8轮不可能差分特征

根据上述分析, 可认为SMBA算法能够抵抗不可能差分攻击。

3.3.6. 相关密钥差分攻击

相关密钥差分攻击是差分攻击的延伸，是一种综合考虑密钥差分和明文差分的攻击方式，赋予攻击者的能力一般强于差分攻击。评估算法抵抗相关密钥差分攻击的强度和差分攻击类似，都是计算最大差分特征概率，不同之处在于是否在密钥处引入差分。按照评估差分攻击强度的方式，我们搜索了在密钥处引入差分后，加密算法和密钥扩展整体的差分活动S盒数，具体结果见表3.3.6-1、3.3.6-2和3.3.6-3。

表3.3.6-1 SMBA-128-128相关密钥差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9
差分活动S盒	0	1	3	6	10	12	16	20	24
轮数	10	11	12	13	14	15	16	17	18
差分活动S盒	28	32	36	38	42	45	49	52	55

表3.3.6-2 SMBA-128-256相关密钥差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	0	1	1	2	3	7	10	12	13	15	19	22
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	24	27	30	32	36	38	42	44	46	50	52	58

表3.3.6-3 SMBA-256-256相关密钥差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	0	1	3	6	10	12	18	22	27	35	44	54
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	64	--	--	--	--	--	--	--	--	--	--	--

注：此表只给出了前13轮SMBA-256-256的相关密钥差分活动S盒的搜索结果，其中第13轮的搜索花费了约7个小时，第14轮运行8个小时之后没有得到结果。

由表3.2.2.2-1知， S_0 的最大差分概率为 $2^{-5.415}$ ， S_1 的最大差分概率为 2^{-6} ，因此，9轮SMBA-128-128算法的最大相关密钥差分特征概率不大于 $2^{-5.415 \times 24} = 2^{-129.86}$ ，22

轮SMBA-128-256算法的最大相关密钥差分特征概率不大于 $2^{-5.415 \times 50} = 2^{-270.75}$ ，12轮SMBA-256-256算法的最大相关密钥差分特征概率不大于 $2^{-5.415 \times 54} = 2^{-292.41}$ ，表明SMBA-128-128、SMBA-128-256、SMBA-256-256都能够抵抗相关密钥差分攻击，且具有足够的安全冗余。

进一步，在详细考虑差分特征中 S_0 和 S_1 的最大差分特征概率情况后，给出了SMBA-128-128、SMBA-128-256、SMBA-256-256各轮最大相关密钥差分特征概率更详细的结果，分别如表3.3.6-4、3.3.6-5和3.3.6-6所示。

表3.3.6-4 SMBA-128-128相关密钥差分特征概率上界

轮数	1	2	3	4	5	6
差分特征概率上界	0	$2^{-5.415}$	$2^{-5.415}$	$2^{-10.830}$	$2^{-16.245}$	$2^{-39.075}$
轮数	7	8	9	10	11	12
差分特征概率上界	$2^{-88.980}$	$2^{-111.225}$	$2^{-133.470}$	$2^{-156.885}$	$2^{-181.470}$	$2^{-203.130}$
轮数	13	14	15	16	17	18
差分特征概率上界	$2^{-214.545}$	$2^{-237.960}$	$2^{-257.130}$	$2^{-279.375}$	$2^{-294.450}$	$2^{-311.280}$

表3.3.6-5 SMBA-128-256相关密钥差分特征概率上界

轮数	1	2	3	4	5	6	7	8
差分特征概率上界	0	$2^{-5.415}$	$2^{-5.415}$	$2^{-10.830}$	$2^{-16.245}$	$2^{-39.075}$	$2^{-54.735}$	$2^{-66.150}$
轮数	9	10	11	12	13	14	15	16
差分特征概率上界	$2^{-72.735}$	$2^{-84.150}$	$2^{-106.980}$	$2^{-122.640}$	$2^{-134.055}$	$2^{-148.545}$	$2^{-165.375}$	$2^{-176.790}$
轮数	17	18	19	20	21	22	23	24
差分特征概率上界	$2^{-198.450}$	$2^{-213.960}$	$2^{-233.865}$	$2^{-248.205}$	$2^{-259.620}$	$2^{-281.865}$	$2^{-294.450}$	$2^{-324.600}$

表3.3.6-6 SMBA-256-256相关密钥差分特征概率上界

轮数	1	2	3	4	5	6	7	8
差分特征 概率上界	0	$2^{-5.415}$	$2^{-16.245}$	$2^{-32.490}$	$2^{-55.320}$	$2^{-69.075}$	$2^{-103.320}$	$2^{-126.735}$
轮数	9	10	11	12	13	14	15	16
差分特征 概率上界	$2^{-156.735}$	$2^{-201.810}$	$2^{-251.865}$	$2^{-304.110}$	$2^{-364.260}$	$2^{-140.790}$	$2^{-151.620}$	$2^{-162.450}$

表3.3.6-4给出9轮SMBA-128-128的相关密钥差分特征概率不大于 $2^{-133.470}$ ，小于表3.3.6-1给出的 $2^{-129.86}$ ；表3.3.6-5给出21轮SMBA-128-256的相关密钥差分特征概率不大于 $2^{-259.620}$ ，小于安全界 2^{-256} ，比表3.3.6-2给出的结果提高了1轮，表3.3.6-2中21轮最大SMBA-128-256相关密钥差分特征概率为 $2^{-249.090}$ ；表3.3.6-6给出12轮最大SMBA-256-256相关密钥差分特征概率为 $2^{-304.11}$ ，小于表3.3.6-3给出的结果 $2^{-292.41}$ 。

3.3.7. 飞去来器攻击

飞去来器攻击是由David Wagner于1999年提出的，是一种差分类型分析方法，其主要思想是利用两条短的高概率差分路径来代替一条长的低概率差分路径，使得对一些加密算法的分析可以进行到更多轮。

设分组密码算法 E 有 R 轮，分组长度为 n 比特，记 P^r 为任意连续 r 轮的最大差分特征概率，若对任意的 r，都有 $(P^r \times P^{R-r})^2 \leq 2^{-n}$ ，则认为密码算法 E 能够抗飞去来器攻击。对于 r 轮 128 比特和 256 比特分组的 SMBA 算法，任意分成两段至少具有的差分活动 S 盒数如表 3.3.7-1 和 3.3.7-2 所示。

表3.3.7-1 r轮SMBA-128任意分成两段具有的差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	--	0	1	2	3	4	8	10	12	13	16	18
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	20	22	24	26	28	30	32	34	36	38	40	42

表3.3.7-2 r轮SMBA-256任意分成两段具有的差分活动S盒数下界

轮数	1	2	3	4	5	6	7	8	9	10	11	12
差分活动S盒	--	0	1	2	3	4	9	12	16	18	21	23
轮数	13	14	15	16	17	18	19	20	21	22	23	24
差分活动S盒	27	29	33	36	38	40	44	46	50	52	54	58

由表3.3.7-1知, 9轮SMBA-128算法任意分成两段至少具有12个差分活动S盒, 飞去来器攻击成功的概率不大于 $2^{-5.415 \times 12 \times 2} = 2^{-129.96} < 2^{-128}$; 由表3.3.7-2知, 13轮SMBA-256算法任意分成两段至少具有27个差分活动S盒, 飞去来器攻击成功的概率不大于 $2^{-5.415 \times 27 \times 2} = 2^{-292.41} < 2^{-256}$ 。因此, 可认为SMBA算法能抵抗飞去来器攻击。

3.3.8. 滑动攻击

滑动攻击和高级滑动攻击对于有相同的轮函数且子密钥周期很短的算法可进行有效攻击, 若密钥扩展算法所生成的子密钥不具有周期性, 则可认为算法不具有此弱点, 能够抵抗滑动攻击。

性质3.3.8-1: 对SMBA-128-128, 不同密钥Key和Key'不会有任意连续3轮相同的子密钥。

证明: 设Key和Key'所生成的子密钥序列分别为 $(ek_0, ek_1, \dots, ek_{17})$ 和 $(ek'_0, ek'_1, \dots, ek'_{17})$ 。不妨设其中 $ek_i = ek'_j$, $ek_{i+1} = ek'_{j+1}$, $ek_{i+2} = ek'_{j+2}$, 由密钥扩展可知存在如下关系:

$$\beta_{64}((\gamma_{64} \circ \alpha_{64})^{-1}(ek_i) \oplus C_i) \oplus ek_{i+1} = (\gamma_{64} \circ \alpha_{64})^{-1}(ek_{i+2}) \oplus C_{i+2}$$

$$\beta_{64}((\gamma_{64} \circ \alpha_{64})^{-1}(ek'_j) \oplus C_j) \oplus ek'_{j+1} = (\gamma_{64} \circ \alpha_{64})^{-1}(ek'_{j+2}) \oplus C_{j+2}$$

将前述关系带入, 化简可得 $\beta_{64}(C_i \oplus C_j) = C_{i+2} \oplus C_{j+2}$ 。若 $i=j$, 则通过简单的推导可得 $Key = Key'$, 矛盾。若 $i \neq j$, 则我们通过检验各轮常数之间的关系可知 $\beta_{64}(C_i \oplus C_j) = C_{i+2} \oplus C_{j+2}$ 不成立。因此不会有任意连续3轮相同的子密钥。

性质3.3.8-2: 对SMBA-128-128, 同一个密钥Key不会有不同位置的连续3轮相同的子密钥。

证明: 同性质3.3.8-1。

性质3.3.8-3: 对SMBA-128-256, 不同密钥Key和Key'不会有任意连续5轮相同的

子密钥。

证明：设Key和Key'所生成的子密钥序列分别为(ek₀, ek₁, ..., ek₂₃)和(ek'₀, ek'₁, ..., ek'₂₃)。不妨设其中ek_i=ek'_j, ek_{i+1}=ek'_{j+1}, ek_{i+2}=ek'_{j+2}, ek_{i+3}=ek'_{j+3}, ek_{i+4}=ek'_{j+4}, 由密钥扩展可知存在如下关系：

$$\beta_{128}((\gamma_{128} \circ \alpha_{128})^{-1}(ek_i) \oplus C_i) \oplus ek_{i+3} = (\gamma_{128} \circ \alpha_{128})^{-1}(ek_{i+4}) \oplus C_{i+4}$$

$$\beta_{128}((\gamma_{128} \circ \alpha_{128})^{-1}(ek'_j) \oplus C_j) \oplus ek'_{j+3} = (\gamma_{128} \circ \alpha_{128})^{-1}(ek'_{j+4}) \oplus C_{j+4}$$

将前述关系带入，化简可得 $\beta_{128}(C_i \oplus C_j) = C_{i+4} \oplus C_{j+4}$ 。若i=j，则通过简单的推导可得Key=Key'，矛盾。若i≠j，则我们通过检验各轮常数之间的关系可知 $\beta_{128}(C_i \oplus C_j) = C_{i+4} \oplus C_{j+4}$ 不成立。因此不会有任意连续5轮相同的子密钥。

性质3.3.8-4：对SMBA-128-256，同一个密钥Key不会有不同位置的连续5轮相同的子密钥。

证明：同性质3.3.8-3。

性质3.3.8-5：对SMBA-256-256，不同密钥Key和Key'不会有任意连续3轮相同的子密钥。

证明：同性质3.3.8-1。

性质3.3.8-6：对SMBA-256-256，同一个密钥Key不会有不同位置的连续3轮相同的子密钥。

证明：同性质3.3.8-1。

根据性质3.3.8-1至性质3.3.8-6，可知对于同一个密钥，SMBA-128-128、SMBA-128-256、SMBA-256-256的密钥扩展算法生成的子密钥都不具有短周期；对于不同的两个密钥，生成的子密钥不具有较长连续相等的块，因此可认为SMBA算法能够抵抗滑动攻击。

3.3.9. 密钥扩展安全性

定义3.3.9-1：若存在不同的密钥Key和Key'，对于任意的明文P， $E_{Key}(P) = E_{Key'}(P)$ ，则称Key与Key'为等价密钥。

定义3.3.9-2：若存在密钥Key，对于任意的明文P， $E_{Key}(P) = D_{Key}(P)$ ，则称Key为弱密钥。

性质3.3.9-3：SMBA密钥扩展具有保墒性，即不同密钥生成的子密钥不同。

证明：由于 α 、 β 、 γ 都是单射，可以容易证明SMBA-128-128、SMBA-128-256、SMBA-256-256的密钥扩展都是单射，不同的密钥Key生成的子密钥不同。

性质3.3.9-4：SMBA密钥扩展生成的加解密子密钥不同，即不存在密钥Key使得： $EK=DK$ 。

证明：分三种情况分析。

1) 对SMBA-128-128，记 $EK=(ek_0, ek_1, \dots, ek_{17})$ ， $DK=(dk_0, dk_1, \dots, dk_{17})$ ，满足 $ek_i=dk_{17-i}$ 。利用前三轮密钥生成过程可得

$$\beta_{64}((\gamma_{64} \circ \alpha_{64})^{-1}(ek_0) \oplus C_0) \oplus ek_1 = (\gamma_{64} \circ \alpha_{64})^{-1}(ek_2) \oplus C_2$$

利用后三轮密钥生成过程可得

$$\beta_{64}((\gamma_{64} \circ \alpha_{64})^{-1}(dk_2) \oplus C_{15}) \oplus dk_1 = (\gamma_{64} \circ \alpha_{64})^{-1}(dk_0) \oplus C_{17}$$

若 $EK=DK$ ，则有 $ek_0=dk_0$ ， $ek_1=dk_1$ ， $ek_2=dk_2$ ，通过化简整理可得(简记 $f=(\gamma_{64} \circ \alpha_{64})^{-1}$):

$$\beta_{64}(f(ek_0) \oplus f(ek_2)) \oplus (f(ek_0) \oplus f(ek_2)) = C_2 \oplus \beta_{64}C_0 \oplus C_{17} \oplus \beta_{64}C_{15}$$

记 $x = f(ek_0) \oplus f(ek_2)$ ，则上式可转化为一个线性方程组 $Ax=C$ ，其中

$$C = C_2 \oplus \beta_{64}C_0 \oplus C_{17} \oplus \beta_{64}C_{15}$$

$$A = \beta_{64} \oplus I = (a_{ij})_{64 \times 64}, \quad a_{ij} = 1, \text{ 若 } i=j \text{ 或 } (j-i) \bmod 64 = 16, \text{ 否则 } a_{ij} = 0$$

为了说明该方程是否有解，我们只需要比较A和其增广矩阵 $A'=[A|C]$ 的秩是否相等即可。经检验 $\text{rank}(A)=48$ ， $\text{rank}(A')=49$ ，因此该线性方程组无解，也即是说明 $ek_0=dk_0$ ， $ek_1=dk_1$ ， $ek_2=dk_2$ 不能同时满足，因此 $EK \neq DK$ 。

2) 对SMBA-256-256，记 $EK=(ek_0, ek_1, \dots, ek_{23})$ ， $DK=(dk_0, dk_1, \dots, dk_{23})$ ，同上可得(简记 $f=(\gamma_{128} \circ \alpha_{128})^{-1}$)

$$\beta_{128}(f(ek_0) \oplus f(ek_2)) \oplus (f(ek_0) \oplus f(ek_2)) = C_2 \oplus \beta_{128}C_0 \oplus C_{23} \oplus \beta_{128}C_{21}$$

同理可得 $\text{rank}(A)=96$ ， $\text{rank}(A')=97$ ，因此 $EK \neq DK$ 。

3) 对SMBA-128-256，记 $EK=(ek_0, ek_1, \dots, ek_{23})$ ， $DK=(dk_0, dk_1, \dots, dk_{23})$ ，这种情况比较复杂，以下详述。

假设 $EK=DK$ ，由 $ek_i=dk_{23-i}, 0 \leq i \leq 23$ 得：

$$ek_i = ek_{23-i}, 0 \leq i \leq 23$$

那么有 $ek_{15}=ek_8, ek_{14}=ek_9, ek_{13}=ek_{10}, ek_{12}=ek_{11}$ ，下面我们由此推出矛盾。

根据密钥扩展算法 8-15 轮，可得等式：

$$\beta[(\gamma \circ \alpha)^{-1}(ek_9) \oplus C_9] \oplus ek_{11} = (\gamma \circ \alpha)^{-1}(ek_{10}) \oplus C_{13}, \quad (1)$$

和如下关系

$$\beta[(\gamma \circ \alpha)^{-1}(ek_{10}) \oplus C_{10}] \oplus ek_{10} = (\gamma \circ \alpha)^{-1}(ek_9) \oplus C_{14}$$

消去 ek_9 ，可以得到等式：

$$\beta^2[(\gamma \circ \alpha)^{-1}(ek_{10})] \oplus (\gamma \circ \alpha)^{-1}(ek_{10}) \oplus \beta(ek_{10}) \oplus ek_{11} = \beta(C_9) \oplus \beta^2(C_{10}) \oplus C_{13} \oplus \beta(C_{14}) \quad (2)$$

又由如下关系

$$(\gamma \circ \alpha)^{-1}(ek_{11}) \oplus C_{12} \oplus ek_{11} = \beta[(\gamma \circ \alpha)^{-1}(ek_8) \oplus C_8]$$

$$\beta[(\gamma \circ \alpha)^{-1}(ek_{11}) \oplus C_{11}] \oplus ek_9 = (\gamma \circ \alpha)^{-1}(ek_8) \oplus C_{15}$$

消去 ek_8 得到等式：

$$\beta(ek_9) \oplus \beta^2[(\gamma \circ \alpha)^{-1}(ek_{11})] \oplus (\gamma \circ \alpha)^{-1}(ek_{11}) \oplus ek_{11} = \beta(C_8) \oplus \beta^2(C_{11}) \oplus C_{12} \oplus \beta(C_{15}) \quad (3)$$

根据函数 α, β, γ 的定义，展开等式(1)，得到等式组(1-1)和(1-2)：

$$\begin{aligned} S_0^{-1}(ek_{9,0}) \oplus S_1^{-1}(ek_{10,7}) \oplus ek_{11,0} &= C_{9,2} \oplus C_{13,0} \\ S_1^{-1}(ek_{9,5}) \oplus S_0^{-1}(ek_{10,2}) \oplus ek_{11,1} &= C_{9,3} \oplus C_{13,1} \\ S_0^{-1}(ek_{9,4}) \oplus S_1^{-1}(ek_{10,3}) \oplus ek_{11,4} &= C_{9,6} \oplus C_{13,4} \\ S_1^{-1}(ek_{9,1}) \oplus S_0^{-1}(ek_{10,6}) \oplus ek_{11,5} &= C_{9,7} \oplus C_{13,5} \end{aligned} \quad (1-1)$$

$$\begin{aligned} S_1^{-1}(ek_{9,3}) \oplus S_0^{-1}(ek_{10,0}) \oplus ek_{11,2} &= C_{9,4} \oplus C_{13,2} \\ S_0^{-1}(ek_{9,6}) \oplus S_1^{-1}(ek_{10,5}) \oplus ek_{11,3} &= C_{9,5} \oplus C_{13,3} \\ S_1^{-1}(ek_{9,7}) \oplus S_0^{-1}(ek_{10,4}) \oplus ek_{11,6} &= C_{9,0} \oplus C_{13,6} \\ S_0^{-1}(ek_{9,2}) \oplus S_1^{-1}(ek_{10,1}) \oplus ek_{11,7} &= C_{9,1} \oplus C_{13,7} \end{aligned} \quad (1-2)$$

展开等式(2)，得到等式组(2-1)和(2-2)：

$$\begin{aligned}
S_1^{-1}(ek_{10,3}) \oplus S_1^{-1}(ek_{10,7}) \oplus ek_{10,2} \oplus ek_{11,0} &= C_{9,2} \oplus C_{10,4} \oplus C_{13,0} \oplus C_{14,2} \\
S_0^{-1}(ek_{10,6}) \oplus S_0^{-1}(ek_{10,2}) \oplus ek_{10,3} \oplus ek_{11,1} &= C_{9,3} \oplus C_{10,5} \oplus C_{13,1} \oplus C_{14,3} \\
S_1^{-1}(ek_{10,7}) \oplus S_1^{-1}(ek_{10,3}) \oplus ek_{10,6} \oplus ek_{11,4} &= C_{9,6} \oplus C_{10,0} \oplus C_{13,4} \oplus C_{14,6} \\
S_0^{-1}(ek_{10,2}) \oplus S_0^{-1}(ek_{10,6}) \oplus ek_{10,7} \oplus ek_{11,5} &= C_{9,7} \oplus C_{10,1} \oplus C_{13,5} \oplus C_{14,7}
\end{aligned} \tag{2-1}$$

$$\begin{aligned}
S_0^{-1}(ek_{10,4}) \oplus S_0^{-1}(ek_{10,0}) \oplus ek_{10,4} \oplus ek_{11,2} &= C_{9,4} \oplus C_{10,6} \oplus C_{13,2} \oplus C_{14,4} \\
S_1^{-1}(ek_{10,1}) \oplus S_1^{-1}(ek_{10,5}) \oplus ek_{10,5} \oplus ek_{11,3} &= C_{9,5} \oplus C_{10,7} \oplus C_{13,3} \oplus C_{14,5} \\
S_0^{-1}(ek_{10,0}) \oplus S_0^{-1}(ek_{10,4}) \oplus ek_{10,0} \oplus ek_{11,6} &= C_{9,0} \oplus C_{10,2} \oplus C_{13,6} \oplus C_{14,0} \\
S_1^{-1}(ek_{10,5}) \oplus S_1^{-1}(ek_{10,1}) \oplus ek_{10,1} \oplus ek_{11,7} &= C_{9,1} \oplus C_{10,3} \oplus C_{13,7} \oplus C_{14,1}
\end{aligned} \tag{2-2}$$

展开等式(3)，得到等式组(3-1)和(3-2)：

$$\begin{aligned}
S_1^{-1}(ek_{11,3}) \oplus S_1^{-1}(ek_{11,7}) \oplus ek_{9,2} \oplus ek_{11,0} &= C_{8,2} \oplus C_{11,4} \oplus C_{12,0} \oplus C_{15,2} \\
S_0^{-1}(ek_{11,6}) \oplus S_0^{-1}(ek_{11,2}) \oplus ek_{9,3} \oplus ek_{11,1} &= C_{8,3} \oplus C_{11,5} \oplus C_{12,1} \oplus C_{15,3} \\
S_1^{-1}(ek_{11,7}) \oplus S_1^{-1}(ek_{11,3}) \oplus ek_{9,6} \oplus ek_{11,4} &= C_{8,6} \oplus C_{11,0} \oplus C_{12,4} \oplus C_{15,6} \\
S_0^{-1}(ek_{11,2}) \oplus S_0^{-1}(ek_{11,6}) \oplus ek_{9,7} \oplus ek_{11,5} &= C_{8,7} \oplus C_{11,1} \oplus C_{12,5} \oplus C_{15,7}
\end{aligned} \tag{3-1}$$

$$\begin{aligned}
S_0^{-1}(ek_{11,0}) \oplus S_0^{-1}(ek_{11,4}) \oplus ek_{9,0} \oplus ek_{11,6} &= C_{8,0} \oplus C_{11,2} \oplus C_{12,6} \oplus C_{15,0} \\
S_1^{-1}(ek_{11,5}) \oplus S_1^{-1}(ek_{11,1}) \oplus ek_{9,1} \oplus ek_{11,7} &= C_{8,1} \oplus C_{11,3} \oplus C_{12,7} \oplus C_{15,1} \\
S_0^{-1}(ek_{11,4}) \oplus S_0^{-1}(ek_{11,0}) \oplus ek_{9,4} \oplus ek_{11,2} &= C_{8,4} \oplus C_{11,6} \oplus C_{12,2} \oplus C_{15,4} \\
S_1^{-1}(ek_{11,1}) \oplus S_1^{-1}(ek_{11,5}) \oplus ek_{9,5} \oplus ek_{11,3} &= C_{8,5} \oplus C_{11,7} \oplus C_{12,3} \oplus C_{15,5}
\end{aligned} \tag{3-2}$$

记：

$$\begin{aligned}
ek_{10}^{[2,3,6,7]} &= ek_{10,2} || ek_{10,3} || ek_{10,6} || ek_{10,7}, \quad ek_{10}^{[0,1,4,5]} = ek_{10,0} || ek_{10,1} || ek_{10,4} || ek_{10,5}, \\
ek_{11}^{[2,3,6,7]} &= ek_{11,2} || ek_{11,3} || ek_{11,6} || ek_{11,7}, \quad ek_{11}^{[0,1,4,5]} = ek_{11,0} || ek_{11,1} || ek_{11,4} || ek_{11,5}, \\
ek_{12}^{[2,3,6,7]} &= ek_{12,2} || ek_{12,3} || ek_{12,6} || ek_{12,7}, \quad ek_{12}^{[0,1,4,5]} = ek_{12,0} || ek_{12,1} || ek_{12,4} || ek_{12,5},
\end{aligned}$$

由(1-1)得： $ek_9^{[0,1,4,5]}$ ， $ek_{10}^{[2,3,6,7]}$ ， $ek_{11}^{[0,1,4,5]}$ 已知其中两个可以得到第三个，

由(1-2)得： $ek_9^{[2,3,6,7]}$ ， $ek_{10}^{[0,1,4,5]}$ ， $ek_{11}^{[2,3,6,7]}$ 已知其中两个可以得到第三个；

由(2-1)得： 已知 $ek_{10}^{[2,3,6,7]}$ 可以得到 $ek_{11}^{[0,1,4,5]}$ ，

由(2-2)得： 已知 $ek_{10}^{[0,1,4,5]}$ 可以得到 $ek_{11}^{[2,3,6,7]}$ ；

由(3-1)得： 已知 $ek_{11}^{[2,3,6,7]}$ 、 $ek_{11}^{[0,1,4,5]}$ 可以得到 $ek_9^{[2,3,6,7]}$ ，

已知 $ek_{11}^{[2,3,6,7]}$ 、 $ek_9^{[2,3,6,7]}$ 可以得到 $ek_{11}^{[0,1,4,5]}$ ，

由(3-2)得：已知 $ek_{11}^{[0,1,4,5]}$ 、 $ek_{11}^{[2,3,6,7]}$ 可以得到 $ek_9^{[0,1,4,5]}$ ，

已知 $ek_{11}^{[0,1,4,5]}$ 、 $ek_9^{[0,1,4,5]}$ 可以得到 $ek_{11}^{[2,3,6,7]}$ ，

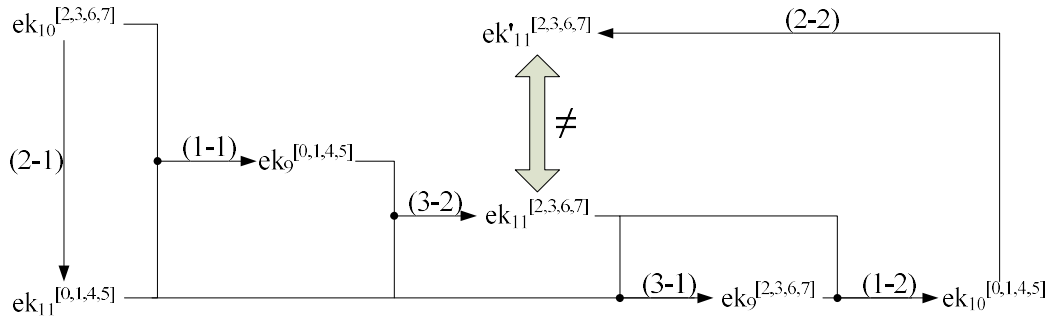


图 3.3.9-1 变量关系推导示意图

如上图所示，其中 \rightarrow 表示“推导出”，我们通过遍历 $ek_{10}^{[2,3,6,7]}$ 的所有 2^{32} 个可能取值，总会得到 $ek_{11}^{[2,3,6,7]} \neq ek'_{11}^{[2,3,6,7]}$ 。因此同时满足等式(1)(2)(3)的子密钥无解，与假设矛盾。

综上所述， $EK \neq DK$ 。

因此，由性质3.3.9-3和性质3.3.9-4，可认为SMBA算法不存在等价密钥和弱密钥。

3.4. 抗侧信道攻击安全防护

SMBA 算法的 2 个 S 盒都采用代数构造的方式设计，具有较好的透明阶、改进的透明阶、Hamming 重量的混乱系数方差等指标，使得在相同安全防护条件下，抵抗侧信道攻击的能力更强。另外，S 盒掩码防护实现难度（芯片面积占用、随机数使用量等）和 S 盒逻辑表达式的乘法复杂度密切相关，逻辑复杂度越低，安全防护实现时使用的随机数量越少、芯片占用面积越低，因此，在 S_0 和 S_1 构造过程中本算法尽量使用乘法复杂度低的部件、运算，使得 S_0 和 S_1 的乘法复杂度低。

S_0 的构造过程使用了 2 个 4 比特 S 盒 Q_0 和 Q_1 ，以及有限域 $GF(2^4)$ 上 X^{13} 运算， S_1 采用有限域 $GF(2^8)$ 上的逆运算进行构造。在进行安全防护实现时， Q_0 和 Q_1 采用逻辑表达式进行实现，有限域上的运算可按代数结构进行实现，根据[6]介绍的方法，下面分别给出 Q_0 和 Q_1 一种乘法复杂度优的实现方案，其中 $x_0||x_1||x_2||x_3$ 为 4 比特输入， $y_0||y_1||y_2||y_3$ 为 4 比特输出， $t_0, t_1, \dots, q_0, q_1, \dots$

为中间值。

Q₀ 的乘法复杂度优的实现方案：

$$\begin{aligned}q_0 &= 1 \oplus x_0 \oplus x_3 \\q_1 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \\t_0 &= q_0 \& q_1 \\q_2 &= 1 \oplus x_2 \oplus x_3 \oplus t_0 \\q_3 &= x_0 \oplus x_2 \oplus x_3 \\t_1 &= q_2 \& q_3 \\q_4 &= x_0 \oplus x_1 \oplus x_2 \oplus t_0 \oplus t_1 \\q_5 &= 1 \oplus x_2 \oplus t_1 \\t_2 &= q_4 \& q_5 \\q_6 &= x_2 \\q_7 &= 1 \oplus x_1 \oplus x_2 \oplus t_0 \oplus t_1 \\t_3 &= q_6 \& q_7 \\q_8 &= x_0 \oplus t_0 \oplus t_2 \oplus t_3 \\q_9 &= 1 \oplus x_0 \oplus x_1 \oplus t_2 \oplus t_3 \\t_4 &= q_8 \& q_9 \\y_0 &= x_2 \oplus x_3 \oplus t_0 \oplus t_1 \oplus t_2 \oplus t_3 \oplus t_4 \\y_1 &= x_0 \oplus t_1 \oplus t_3 \oplus t_4 \\y_2 &= x_2 \oplus t_0 \oplus t_1 \oplus t_2 \oplus t_4 \\y_3 &= x_0 \oplus x_1 \oplus t_0 \oplus t_2 \oplus t_3 \oplus t_4\end{aligned}$$

Q₁ 的乘法复杂度优的实现方案：

$$\begin{aligned}q_0 &= 1 \oplus x_0 \oplus x_3 \\q_1 &= 1 \oplus x_0 \oplus x_1 \oplus x_2 \oplus x_3 \\t_0 &= q_0 \& q_1 \\q_2 &= x_0 \oplus x_1 \oplus x_3 \\q_3 &= 1 \oplus x_0 \oplus x_1 \oplus t_0 \\t_1 &= q_2 \& q_3 \\q_4 &= x_2 \oplus x_3 \\q_5 &= 1 \oplus x_1 \oplus t_0 \\t_2 &= q_4 \& q_5 \\q_6 &= x_0 \oplus x_1 \oplus t_1 \oplus t_2 \\q_7 &= 1 \oplus x_0 \oplus x_2 \oplus t_0 \oplus t_1 \\t_3 &= q_6 \& q_7 \\q_8 &= x_1 \oplus x_2 \oplus t_0 \oplus t_1 \\q_9 &= x_2 \oplus t_0 \oplus t_1 \oplus t_3 \\t_4 &= q_8 \& q_9 \\y_0 &= x_2 \oplus x_3 \oplus t_0 \oplus t_1 \oplus t_2 \oplus t_3 \oplus t_4 \\y_1 &= x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus t_2 \\y_2 &= x_1 \oplus x_3 \oplus t_0 \oplus t_2 \oplus t_3 \oplus t_4 \\y_3 &= x_1 \oplus x_2 \oplus x_3 \oplus t_2 \oplus t_3\end{aligned}$$

在附录 B 中给出 S_0 和 S_1 的一阶门限掩码方案。

3.5. 依赖性检测

3.5.1. 检测项目

对于一个向量 $x=(x_1, \dots, x_n) \in GF(2)^n$, 向量 $x^{(i)} \in GF(2)^n$ 记 x 的第 i 比特取反（其余比特不变）得到的向量（对于 $i=1, \dots, n$ ）。 x 的 Hamming 重量 $w(x)$ 定义为 x 的非零分量的数目。

n 个输入比特到 m 个输出比特的函数 $f: GF(2)^n \rightarrow GF(2)^m$ 称为是完全的, 如果每一输出比特依赖于每一输入比特, 即对于所有 $i=1, 2, \dots, n, j=1, 2, \dots, m$, 存在 $x \in GF(2)^n$ 满足 $(f(x^{(i)}))_j \neq (f(x))_j$

函数 $f: GF(2)^n \rightarrow GF(2)^m$ 称为具有雪崩效果, 如果每当唯一输入比特取反, 输出比特改变的平均值为 $1/2$, 即对于所有 $i=1, 2, \dots, n$

$$\frac{1}{2^n} \sum_{x \in GF(2)^n} w(f(x^{(i)}) \oplus f(x)) = \frac{m}{2}$$

函数 $f: GF(2)^n \rightarrow GF(2)^m$ 称为满足严格雪崩准则, 如果每当唯一输入比特取反, 每一输出比特以 $1/2$ 的概率改变, 即对于所有 $i=1, 2, \dots, n, j=1, 2, \dots, m$

$$\Pr[(f(x^{(i)}))_j \neq (f(x))_j] = \frac{1}{2}$$

函数 $f: GF(2)^n \rightarrow GF(2)^m$ 的依赖矩阵是一个 $n \times m$ 矩阵 A , 它的第 (i, j) 个元素 a_{ij} 定义为

$$a_{ij} = \#\{x \in GF(2)^n | (f(x^{(i)}))_j \neq (f(x))_j\}, \quad i=1, 2, \dots, n, \quad j=1, 2, \dots, m$$

函数 $f: GF(2)^n \rightarrow GF(2)^m$ 的距离矩阵是一个 $n \times (m+1)$ 矩阵 B , 它的第 (i, j) 个元素 b_{ij} 定义为

$$b_{ij} = \#\{x \in GF(2)^n | w(f(x^{(i)}) \oplus f(x)) = j\}, \quad i=1, 2, \dots, n, \quad j=0, 1, 2, \dots, m$$

对于 $n=m \geq 128$, 精确计算依赖矩阵和距离矩阵是不可能的, 只能考虑适当数目的随机选取的输入。依赖矩阵和距离矩阵的定义修改为

$$a_{ij} = \#\{x \in X | (f(x^{(i)}))_j \neq (f(x))_j\}, \quad i=1, 2, \dots, n, \quad j=1, 2, \dots, m$$

$$b_{ij} = \#\{x \in X | w(f(x^{(i)}) \oplus f(x)) = j\}, \quad i=1, 2, \dots, n, \quad j=0, 1, 2, \dots, m$$

这里 X 是 $GF(2)^n$ 的适当随机选取的子集。

假设函数 $f: GF(2)^n \rightarrow GF(2)^m$ 对于 $GF(2)^n$ 的适当随机选取的子集 X 计算好了

依赖矩阵和距离矩阵。当唯一输入比特取反，输出比特改变的平均数目为

$$d_v = \frac{\sum_{i=1}^n \sum_{j=1}^m j b_{ij}}{n \times \#X}$$

f 的完全性的度定义为

$$d_c = 1 - \frac{\#\{(i, j) | a_{ij} = 0\}}{nm}$$

f 的雪崩效果的度定义为

$$d_a = \frac{\sum_{i=1}^n \left| \frac{1}{\#X} \sum_{j=1}^m 2j b_{ij} - m \right|}{nm}$$

f 的严格雪崩准则的度定义为

$$d_{sa} = 1 - \frac{\sum_{i=1}^n \sum_{j=1}^m \left| \frac{2a_{ij}}{\#X} - 1 \right|}{nm}$$

对于随机的函数 f，应满足

$$d_v \approx \frac{m}{2}, \quad d_c = 1, \quad d_a \approx 1, \quad d_{sa} \approx 1$$

3.5.2. SMBA-128 检测结果及分析

我们对于 10 万个随机选取的 128 比特明文，在一个随机选取的 256 比特密钥加密下，检测轮变换每一轮结束后函数的 d_v 、 d_c 、 d_a 和 d_{sa} ，检测 24 轮，得到如下依赖性检测结果。

轮数	d_v	d_c	d_a	d_{sa}
0:	1.000000	0.007813	0.015625	0.000000
1:	14.958210	0.226563	0.233722	0.206627
2:	44.416349	0.691406	0.694005	0.662633
3:	61.957608	0.968750	0.967975	0.954742
4:	63.998987	1.000000	0.999778	0.997473
5:	63.998483	1.000000	0.999765	0.997480
6:	63.998183	1.000000	0.999783	0.997485
7:	63.996191	1.000000	0.999794	0.997478
8:	63.996714	1.000000	0.999781	0.997480
9:	64.000550	1.000000	0.999804	0.997488
10:	64.002730	1.000000	0.999788	0.997491

11:	64.002199	1.000000	0.999775	0.997487
12:	63.999169	1.000000	0.999769	0.997476
13:	63.998606	1.000000	0.999770	0.997476
14:	64.000318	1.000000	0.999777	0.997477
15:	64.000959	1.000000	0.999772	0.997470
16:	63.999883	1.000000	0.999782	0.997470
17:	63.999712	1.000000	0.999758	0.997476
18:	63.999936	1.000000	0.999758	0.997479
19:	63.999226	1.000000	0.999757	0.997461
20:	63.999347	1.000000	0.999766	0.997469
21:	64.000135	1.000000	0.999756	0.997488
22:	64.002188	1.000000	0.999784	0.997468
23:	64.002782	1.000000	0.999781	0.997481
24:	64.000222	1.000000	0.999783	0.997496

从检测结果可知，SMBA-128 的轮变换经过 4 轮的输出函数与随机函数没有区别。

3.5.3. SMBA-256 检测结果及分析

我们对于 10 万个随机选取的 256 比特明文，在一个随机选取的 256 比特密钥加密下，检测轮变换每一轮结束后函数的 d_v 、 d_c 、 d_a 和 d_{sa} ，检测 24 轮，得到如下依赖性检测结果。

轮数	d_v	d_c	d_a	d_{sa}
0:	1.000000	0.003906	0.007812	0.000000
1:	14.956403	0.113281	0.116847	0.103316
2:	60.410450	0.470703	0.471957	0.455970
3:	109.952037	0.859375	0.858935	0.851379
4:	127.999523	1.000000	0.999850	0.997465
5:	128.000576	1.000000	0.999835	0.997478
6:	127.999715	1.000000	0.999834	0.997481
7:	128.002065	1.000000	0.999839	0.997492
8:	128.002201	1.000000	0.999842	0.997482
9:	127.999058	1.000000	0.999836	0.997471
10:	127.998406	1.000000	0.999852	0.997477

11:	127.999889	1.000000	0.999856	0.997483
12:	127.999023	1.000000	0.999845	0.997484
13:	127.998353	1.000000	0.999845	0.997479
14:	128.000351	1.000000	0.999853	0.997481
15:	128.001645	1.000000	0.999842	0.997484
16:	128.002709	1.000000	0.999848	0.997483
17:	128.003751	1.000000	0.999847	0.997478
18:	128.001965	1.000000	0.999840	0.997478
19:	128.001265	1.000000	0.999844	0.997480
20:	128.001160	1.000000	0.999842	0.997479
21:	128.002230	1.000000	0.999857	0.997478
22:	128.000171	1.000000	0.999840	0.997470
23:	127.999292	1.000000	0.999836	0.997467
24:	128.000895	1.000000	0.999844	0.997474

从检测结果可知，SMBA-256 的轮变换经过 4 轮的输出函数与随机函数没有区别。

3.6. 随机性检测

3.6.1. 显著性水平

随机性检测的显著性水平设为 $\alpha=0.01$ 。

3.6.2. 检测项目

采用商密随机数检查标准 GB/T 32915-2016，具体包含单比特频数检测、块内频数检测、扑克检测、重叠子序列检测、游程总数检测、游程分布检测、块内最大游程检测、二元推导检测、自相关检测、矩阵秩检测、累加和检测、近似熵检测、线性复杂度检测、通用统计检测、离散傅里叶检测。

3.6.3. 密文随机性检测

- (1) 检测分组密码算法在每种数据生成模式下产生 $S=100$ 组长度为 128K 字节的序列通过每种随机性检测项目的比率。
- (2) 数据生成模式包括高密度明文模式、低密度明文模式、高密度明文模式、低密度明文模式、随机明文随机密钥模式。

(3) 每项检测通过的组数应不少于 $S \times \left(1 - \alpha - 3\sqrt{\frac{\alpha(1-\alpha)}{S}} \right) \approx 96.01$ 。

3.6.4. 检测结果

3.6.4.1. SMBA-128-128 检测结果

表3.6.4.1-1 高密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	97	累加和	99
重叠子序列	100	离散傅里叶	98
单比特频数	99	扑克	98
二元推导	98	通用统计	99
近似熵	100	线性复杂度	100
矩阵秩	98	游程总数	98
块内频数	99	游程分布	97
块内最大游程	99	自相关	98

表3.6.4.1-2 低密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	99
重叠子序列	100	离散傅里叶	100
单比特频数	98	扑克	99
二元推导	99	通用统计	100
近似熵	99	线性复杂度	98
矩阵秩	99	游程总数	99

块内频数	99	游程分布	98
块内最大游程	99	自相关	99

表3.6.4.1-3 高密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	100
重叠子序列	100	离散傅里叶	100
单比特频数	100	扑克	100
二元推导	100	通用统计	100
近似熵	100	线性复杂度	100
矩阵秩	100	游程总数	99
块内频数	99	游程分布	100
块内最大游程	99	自相关	99

表3.6.4.1-4 低密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	98
重叠子序列	100	离散傅里叶	99
单比特频数	99	扑克	99
二元推导	100	通用统计	99
近似熵	99	线性复杂度	99
矩阵秩	99	游程总数	99

块内频数	98	游程分布	99
块内最大游程	99	自相关	98

表3.6.4.1-5 随机明文随机密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	98
重叠子序列	100	离散傅里叶	97
单比特频数	99	扑克	100
二元推导	99	通用统计	97
近似熵	100	线性复杂度	100
矩阵秩	99	游程总数	100
块内频数	99	游程分布	98
块内最大游程	99	自相关	100

3.6.4.2. SMBA-128-256 检测结果

表3.6.4.2-1 高密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	99
重叠子序列	100	离散傅里叶	100
单比特频数	100	扑克	98
二元推导	98	通用统计	99
近似熵	100	线性复杂度	99

矩阵秩	99	游程总数	98
块内频数	99	游程分布	100
块内最大游程	100	自相关	99

表3.6.4.2-2 低密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	100
重叠子序列	100	离散傅里叶	98
单比特频数	100	扑克	99
二元推导	99	通用统计	97
近似熵	99	线性复杂度	97
矩阵秩	100	游程总数	99
块内频数	100	游程分布	97
块内最大游程	98	自相关	99

表3.6.4.2-3 高密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	99
重叠子序列	100	离散傅里叶	100
单比特频数	99	扑克	97
二元推导	99	通用统计	100
近似熵	99	线性复杂度	97

矩阵秩	100	游程总数	100
块内频数	100	游程分布	99
块内最大游程	100	自相关	100

表3.6.4.2-4 低密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	99
重叠子序列	100	离散傅里叶	98
单比特频数	100	扑克	100
二元推导	100	通用统计	98
近似熵	99	线性复杂度	100
矩阵秩	100	游程总数	99
块内频数	99	游程分布	100
块内最大游程	100	自相关	99

表3.6.4.2-5 随机明文随机密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	99
重叠子序列	100	离散傅里叶	99
单比特频数	98	扑克	98
二元推导	98	通用统计	100
近似熵	100	线性复杂度	100

矩阵秩	99	游程总数	98
块内频数	100	游程分布	100
块内最大游程	98	自相关	98

3.6.4.3. SMBA-256-256 检测结果

表3.6.4.3-1 高密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	98	累加和	99
重叠子序列	100	离散傅里叶	99
单比特频数	99	扑克	100
二元推导	99	通用统计	100
近似熵	100	线性复杂度	100
矩阵秩	99	游程总数	99
块内频数	100	游程分布	100
块内最大游程	97	自相关	99

表3.6.4.3-2 低密度明文模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	99	累加和	99
重叠子序列	100	离散傅里叶	99
单比特频数	98	扑克	99
二元推导	100	通用统计	99

近似熵	98	线性复杂度	100
矩阵秩	100	游程总数	99
块内频数	100	游程分布	97
块内最大游程	98	自相关	100

表3.6.4.3-3 高密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	98	累加和	97
重叠子序列	100	离散傅里叶	98
单比特频数	98	扑克	98
二元推导	99	通用统计	99
近似熵	100	线性复杂度	99
矩阵秩	100	游程总数	99
块内频数	100	游程分布	99
块内最大游程	100	自相关	99

表3.6.4.3-4 低密度密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	98
重叠子序列	100	离散傅里叶	99
单比特频数	98	扑克	98
二元推导	99	通用统计	99

近似熵	99	线性复杂度	97
矩阵秩	98	游程总数	99
块内频数	99	游程分布	100
块内最大游程	99	自相关	99

表3.6.4.3-5 随机明文随机密钥模式

检测项	通过检测组数	检测项	通过检测组数
重叠子序列	100	累加和	100
重叠子序列	100	离散傅里叶	97
单比特频数	100	扑克	99
二元推导	97	通用统计	98
近似熵	99	线性复杂度	100
矩阵秩	99	游程总数	98
块内频数	100	游程分布	99
块内最大游程	100	自相关	98

4. 性能分析

4.1. 软件实现

4.1.1. 8 位平台实现

SMBA算法的基础部件S盒和线性变换 L_{32} 都基于8比特设计，特别是 L_{32} 的简洁设计，使得SMBA算法可以在8比特平台上友好实现。

在8比特平台上， L_{32} 可以按如下方式用6个字节异或运算进行实现， $Y=L_{32}(X)$ ， $X=(x_0,x_1,x_2,x_3)$ ， $Y=(y_0,y_1,y_2,y_3)$ ：

$$u = x_2 \oplus x_3$$

$$y_0 = u \oplus x_0$$

$$y_1 = u \oplus x_1$$

$$u = x_0 \oplus x_1$$

$$y_2 = u \oplus x_2$$

$$y_3 = u \oplus x_3$$

在8比特平台上， L_{64} 可以按如下方式进行实现， $Y=L_{32}(X)$ ， $X=(x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7)$ ， $Y=(y_0,y_1,y_2,y_3,y_4,y_5,y_6,y_7)$ ：

$$(z_0,z_1,z_2,z_3)=L_{32}(x_0,x_1,x_2,x_3)$$

$$(z_4,z_5,z_6,z_7)=L_{32}(x_4,x_5,x_6,x_7)$$

$$t_0 = z_0 \oplus z_4$$

$$t_1 = z_1 \oplus z_5$$

$$t_2 = z_2 \oplus z_6$$

$$t_3 = z_3 \oplus z_7$$

$$s_0 = (t_1 \ll 1) \mid (t_2 \gg 7)$$

$$s_1 = (t_2 \ll 1) \mid (t_3 \gg 7)$$

$$s_2 = (t_3 \ll 1) \mid (t_0 \gg 7)$$

$$s_3 = (t_0 \ll 1) \mid (t_1 \gg 7)$$

$$y_0 = z_0 \oplus s_0$$

$$y_1 = z_1 \oplus s_1$$

$$y_2 = z_2 \oplus s_2$$

$$y_3 = z_3 \oplus s_3$$

$$y_4 = z_4 \oplus s_0$$

$$y_5 = z_5 \oplus s_1$$

$$y_6 = z_6 \oplus s_2$$

$$y_7 = z_7 \oplus s_3$$

4.1.2. 32 位平台实现

4.1.2.1. 平台描述

SMBA的32位软件实现平台采用Intel Core(TM) i5-4460 CPU、主频3.20GHz、

Windows7、标准C语言、Microsoft Visual Studio 2010编译器。

4.1.2.2. 实现方法

在32位平台上，把线性变换 L_{32} 和S盒代替变换结合，构造4个8到32比特的大表进行快速实现，记 $Y=L_{32}^{\circ}S(X)$ ， $X=(x_0, x_1, x_2, x_3)$ ，则

$$Y = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \times \begin{pmatrix} S_0(x_0) \\ S_1(x_1) \\ S_0(x_2) \\ S_1(x_3) \end{pmatrix} = \begin{bmatrix} S_0(x_0) \\ 0 \\ S_0(x_0) \\ S_0(x_0) \end{bmatrix} \oplus \begin{bmatrix} 0 \\ S_1(x_1) \\ S_1(x_1) \\ S_1(x_1) \end{bmatrix} \oplus \begin{bmatrix} S_0(x_2) \\ S_0(x_2) \\ S_0(x_2) \\ 0 \end{bmatrix} \oplus \begin{bmatrix} S_1(x_3) \\ S_1(x_3) \\ 0 \\ S_1(x_3) \end{bmatrix}$$

可按如下方式构造4个8到32比特的大表 L_0 、 L_1 、 L_2 、 L_3 进行快速实现：

$$L_0[x] = S_0(x) \parallel 0 \parallel S_0(x) \parallel S_0(x)$$

$$L_1[x] = 0 \parallel S_1(x) \parallel S_1(x) \parallel S_1(x)$$

$$L_2[x] = S_0(x) \parallel S_0(x) \parallel S_0(x) \parallel 0$$

$$L_3[x] = S_1(x) \parallel S_1(x) \parallel 0 \parallel S_1(x)$$

则， $Y=L_{32}^{\circ}S(X)=L_0[x_0] \oplus L_1[x_1] \oplus L_2[x_2] \oplus L_3[x_3]$ 。

其余变换按定义实现。

4.1.2.3. 性能测试结果

采用ECB模式，分别测试算法在256字节和1M字节两种数据长度模式下的加密速度，具体结果见表4.1.2.3-1。

表4.1.2.3-1 32位平台软件测试性能

	OBJ文件大小	256 字节速度	1M字节速度
SMBA-128-128	13692B	1044.45 Mbps	1107.57Mbps
SMBA-128-256	13846B	781.25 Mbps	843.44 Mbps
SMBA-256-256	17114B	895.93 Mbps	1070.66 Mbps

4.1.3. 64 位平台实现

4.1.3.1. 平台描述

SMBA的64位软件实现平台采用Intel Core(TM) i7-6700 CPU、主频3.4GHz、Windows7、标准C语言、Microsoft Visual Studio 2010编译器。

4.1.3.2. 实现方法

在64位平台上，把线性变换 L_{64} 和S盒代替变换结合，构造8个8到64比特的表进行快速实现，记 $Y=L_{64} \circ S(X)$ ， $X=(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ ，则

$$\begin{aligned} Y &= L_{64} \circ S(X) = L_{64}(s_0(x_0), s_1(x_1), s_0(x_2), s_1(x_3), s_0(x_4), s_1(x_5), s_0(x_6), s_1(x_7)) \\ &= L_{64}(s_0(x_0), 0, \dots, 0) \\ &\oplus L_{64}(0, s_1(x_1), \dots, 0) \\ &\dots \\ &\oplus L_{64}(0, \dots, 0, s_1(x_7)) \end{aligned}$$

可按如下方式构造8个8到64比特的大表 L_0 、 L_1 、 L_2 、 L_3 、 L_4 、 L_5 、 L_6 、 L_7 ：

$$\begin{aligned} L_0[x] &= L_{64}(s_0(x), 0, 0, 0, 0, 0, 0, 0) \\ L_1[x] &= L_{64}(0, s_1(x), 0, 0, 0, 0, 0, 0) \\ L_2[x] &= L_{64}(0, 0, s_0(x), 0, 0, 0, 0, 0) \\ L_3[x] &= L_{64}(0, 0, 0, s_1(x), 0, 0, 0, 0) \\ L_4[x] &= L_{64}(0, 0, 0, 0, s_0(x), 0, 0, 0) \\ L_5[x] &= L_{64}(0, 0, 0, 0, 0, s_1(x), 0, 0) \\ L_6[x] &= L_{64}(0, 0, 0, 0, 0, 0, s_0(x), 0) \\ L_7[x] &= L_{64}(0, 0, 0, 0, 0, 0, 0, s_1(x)) \end{aligned}$$

则， $Y=L_{64} \circ S(X)=L_0[x_0] \oplus L_1[x_1] \oplus L_2[x_2] \oplus L_3[x_3] \oplus L_4[x_4] \oplus L_5[x_5] \oplus L_6[x_6] \oplus L_7[x_7]$ 。

其余变换按定义实现。

4.1.3.3. 性能测试结果

采用ECB模式，分别测试算法在256字节和1M字节两种数据长度模式下的加密速度，具体结果见表4.1.3.3-1。

表4.1.3.3-1 64平台软件测试性能

	OBJ文件大小	256 字节速度	1M字节速度
SMBA-128-128	15055 B	1461.26 Mbps	1591.41 Mbps
SMBA-128-256	15208 B	1065.65 Mbps	1183.08 Mbps
SMBA-256-256	18449 B	1496.42 Mbps	1767.96 Mbps

4.2. 硬件实现

4.2.1. ASIC 评估结果

算法的 ASIC 实现环境，采用的设计库为中芯国际 SMIC65nm 工艺标准库，采用的综合工具为 Design Compiler C-2009.06-SP5。仿真环境为 Modelsim SE 10.2c。

(1) SMBA-128-128 算法的评估结果

表 4.2.1-1 SMBA-128-128 算法 ASIC 实现的资源占用情况

Resources	Size
Combinational area	23456.56 μm^2
Noncombinational area	13897.88 μm^2
Total Cell Area	37353.44 μm^2
Number of standard cell	25940 Gate

表 4.2.1-2 SMBA-128-128 算法 ASIC 实现的性能情况

性能指标	性能值
最大时钟频率	1000MHz
最小时钟周期	1.00ns
加密/解密时钟数	18
密钥扩展时钟数	19
吞吐量	7.11Gbps
加密/解密延时	18 ns
效费比	2740.94Mbps/万门

(2) SMBA-128-256 算法的评估结果

表 4.2.1-3 SMBA-128-256 算法 ASIC 实现的资源占用情况

Resources	Size
Combinational area	25371.96 μm^2
Noncombinational area	18629.16 μm^2
Total Cell Area	44000.12 μm^2

Number of standard cell	30556 Gate
-------------------------	------------

表 4.2.1-4 SMBA-128-256 算法 ASIC 实现的性能情况

性能指标	性能值
最大时钟频率	1000MHz
最小时钟周期	1.00ns
加密/解密时钟数	24
密钥扩展时钟数	25
吞吐量	5.33Gbps
加密/解密延时	24ns
效费比	1744.34Mbps/万门

(3) SMBA-256-256 算法的评估结果

表 4.2.1-5 SMBA-256-256 算法 ASIC 实现的资源占用情况

Resources	Size
Combinational area	49742.12 μm^2
Noncombinational area	34813.52 μm^2
Total Cell Area	84556.64 μm^2
Number of standard cell	58720 Gate

表 4.2.1-6 SMBA-256-256 算法 ASIC 实现的性能情况

性能指标	性能值
最大时钟频率	1000MHz
最小时钟周期	1.00ns
加密/解密时钟数	24
密钥扩展时钟数	25
吞吐量	10.67Gbps
加密/解密延时	24ns
效费比	1817.10Mbps/万门

4.2.2. FPGA 评估结果

算法的 FPGA 实现环境，采用 Xilinx 的 4vfx100ff1152-12，采用的综合工具为 ISE14.7。仿真环境为 Modelsim SE 10.2c。

(1) SMBA-128-128 算法的评估结果

表 4.2.2-1 SMBA-128-128 算法 FPGA 实现的资源占用情况

Number of Slices	1517
Number of Slice Flip Flops	1488
Number of 4 input LUTs	2813

表 4.2.2-2 SMBA-128-128 算法 FPGA 实现的性能情况

性能指标	性能值
最大时钟频率	153.312MHz
最小时钟周期	6.523ns
加密/解密时钟数	18
密钥扩展时钟数	19
吞吐量	1.09Gbps
加密/解密延时	117.414ns

(2) SMBA-128-256 算法的评估结果

表 4.2.2-3 SMBA-128-256 算法 FPGA 实现的资源占用情况

Number of Slices	1809
Number of Slice Flip Flops	2000
Number of 4 input LUTs	3321

表 4.2.2-4 SMBA-128-256 算法 FPGA 实现的性能情况

性能指标	性能值
最大时钟频率	153.312MHZ
最小时钟周期	6.523ns
加密/解密时钟数	24
密钥扩展时钟数	25

吞吐量	0.82Gbps
加密/解密延时	156.552ns

(3) SMBA-256-256 算法的评估结果

表 4.2.2-5 SMBA-256-256 算法 FPGA 实现的资源占用情况

Number of Slices	3296
Number of Slice Flip Flops	3728
Number of 4 input LUTs	6041

表 4.2.2-6 SMBA-256-256 算法 FPGA 实现的性能情况

性能指标	性能值
最大时钟频率	143.542MHz
最小时钟周期	6.967ns
加密/解密时钟数	24
密钥扩展时钟数	25
吞吐量	1.53Gbps
加密/解密延时	167.208ns

5. 优缺点声明

SMBA分组密码算法的创新点主要体现在如下4个方面：

- (1) 提出线性变换新的密码性质，并创新设计了密码指标优、软硬件实现性能高的线性变换 L_{64} 。利用新的密码指标，通过理论证明和计算机搜索（两者结果相同）均可证明，SMBA-128的差分/线性密码攻击活动S盒的个数得以显著提高。
- (2) 整体设计保证安全：设计了换位变换 P_{128} 。 P_{128} 结合两个并置的线性变换 L_{64} ，确保SMBA-256的加解密过程都形成一个整体，大幅度提升密码算法的扩散性，增强了算法抗差分/线性等密码攻击的能力。
- (3) 面向实现优化设计：设计了密码指标高的S盒 S_0 和 S_1 ，尤其是 S_0 的设计。与已有主要密码指标相同的S盒相比，硬件实现电路的门数和深度均有明显减少。
- (4) SMBA-256有适于软硬件实现和密码分析的两种等价描述，分别应用变

换 L_{64} 、 P_{128} 和 L_{64}' 、 P_{128}' 。这使得SMBA-256的有效性和安全性的综合性能得以优化。

SMBA分组密码算法经充分的抗差分/线性密码分析、代数攻击、积分攻击、相关密钥差分等多种已知密码攻击的分析，通过了依赖性和随机性检测，在多种软硬件平台上进行了实现和分析，并且给出了S盒的抗侧信道攻击防护实现。由于共用主要的密码部件，因此不同分组长度/密钥长度的SMBA一起实现所付出的额外代价低。总之，我们确信SMBA是安全性高、实用性强、创新设计的分组密码算法，算法的设计还具备简洁性、灵活性和兼容性，完全能够满足本次算法设计竞赛的要求。

由于时间有限，对SMBA算法抗有些密码攻击（如零相关攻击、双系攻击等）的分析尚未开展，算法的软硬件实现性能仍有较大的提升空间。

6. 参考文献

1. Abbasi, I., and Afzal, M., A Compact S-Box Design for SMS4 Block Cipher. Cryptology ePrint Archive, Report 2011/522, 2011.
2. Antonio, R., On some methods for constructing almost optimal S-Boxes and their resilience against side-channel attacks. Cryptology ePrint Archive, Report 2018/618, 2018.
3. Biryukov, A., Perrin L., and Udovenko A., Reverse engineering the S-Box of streebog, kuznyechik and STRIBOBr1. Advances in Cryptology EUROCRYPT 2016, Part I, volume 9665 of LNCS, pages 372-402. Springer, Heidelberg(2016).
4. Courtois, N., and Pieprzyk, J., Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. ASIACRYPT 2002, LNCS 2501, pp.267-287 Springer, Heidelberg(2002).
5. Mouha, N., Wang, Q., Gu, D., and Preneel, B., Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming. Information Security and Cryptology, LNCS, vol.7537, pp.57-76. Springer, Heidelberg(2012).
6. Stoffelen. K. Optimizing S-box Implementations for Several Criteria using SAT Solvers. Cryptology ePrint Archive, Report 2016/198, 2016.
7. Suzaki, T., Minematsu, K.: Improving the Generalized Feistel. FSE 2010. LNCS, vol. 6147, pp. 19-39. Springer, Heidelberg (2010).
8. Todo, Y., Structural evaluation by generalized integral property. In Advances in Cryptology EUROCRYPT 2015, Part I, LNCS 9056, pp. 287–314, 2015.

9. Wu, S., Wang, M.: Automatic Search of Truncated Impossible Differentials for Word-Oriented Block Ciphers. Indocrypt 2012, LNCS 7668, pp. 283-302, Springer, Heidelberg (2012).

附录 A S_0 和 S_1 深度最优实现

SMBA算法采用2个8比特S盒 S_0 和 S_1 ， S_0 和 S_1 都是由代数构造的方式得到， S_0 的构造过程使用了2个4比特S盒 Q_0 和 Q_1 ，以及有限域 $GF(2^4)$ 上 X^{13} 运算， S_1 采用有限域 $GF(2^8)$ 上的逆运算进行构造，可转化为 $GF(2^4)$ 上的逆运算。在硬件实现时，为了降低关键路径长度，提升算法吞吐率，根据[6]，分别给出了 Q_0 、 Q_1 、 X^{13} 和 X^{-1} 的深度最优硬件实现方案，具体方案如下，其中 $x_0||x_1||x_2||x_3$ 为4比特输入， $y_0||y_1||y_2||y_3$ 为4比特输出， t_0 、 t_1 、.....为中间值。

$Q_0:(6,4)$

$$\begin{aligned}
 t_0 &= (x_2 \& x_1) \oplus 1 \\
 t_1 &= (x_1 \& x_3) \oplus 1 \\
 t_2 &= (x_0 \& x_3) \oplus 1 \\
 t_3 &= (x_3 \& x_2) \oplus 1 \\
 t_4 &= (x_1 \& x_0) \oplus 1 \\
 t_5 &= t_1 \oplus t_3 \oplus 1 \\
 t_6 &= t_4 | t_3 \\
 t_7 &= t_4 | t_4 \\
 t_8 &= t_3 \oplus x_2 \\
 t_9 &= (x_0 \& t_1) \oplus 1 \\
 t_{10} &= x_3 \oplus t_5 \\
 t_{11} &= t_0 \oplus t_6 \oplus 1 \\
 t_{12} &= t_2 \oplus t_9 \oplus 1 \\
 t_{13} &= t_2 \oplus t_7 \oplus 1 \\
 t_{14} &= t_9 \& t_3 \\
 t_{15} &= t_{14} \oplus x_1 \oplus 1 \\
 t_{16} &= t_{12} \oplus t_1 \oplus 1 \\
 t_{17} &= t_{11} \oplus t_1 \\
 t_{18} &= t_8 \& t_{13} \\
 t_{19} &= t_9 \oplus t_{10} \oplus 1 \\
 y_1 &= t_{18} \oplus t_{17} \oplus 1 \\
 t_{21} &= (t_{18} | x_2) \oplus 1 \\
 y_2 &= t_{15} \oplus t_{17} \\
 y_0 &= t_{19} \oplus x_1 \oplus 1 \\
 y_3 &= (t_0 \& t_{16}) \oplus 1
 \end{aligned}$$

$Q_1:(9,3)$

$$\begin{aligned}
 t_0 &= (x_3 \& x_2) \oplus 1 \\
 t_1 &= (x_3 \& x_0) \oplus 1 \\
 t_2 &= (x_2 \& x_1) \oplus 1
 \end{aligned}$$

$$\begin{aligned}
t_3 &= t_0 \oplus x_0 \\
t_4 &= x_0 \oplus t_2 \\
t_5 &= t_3 \mid t_4 \\
t_6 &= (x_0 \& x_1) \oplus 1 \\
t_7 &= t_0 \oplus t_2 \oplus 1 \\
t_8 &= t_1 \oplus t_5 \oplus 1 \\
t_9 &= t_6 \oplus t_7 \oplus 1 \\
t_{10} &= t_9 \oplus t_1 \oplus 1 \\
t_{11} &= (t_9 \& x_2) \oplus 1 \\
t_{12} &= t_{10} \oplus t_{11} \oplus 1 \\
t_{13} &= (t_9 \& x_3) \oplus 1 \\
t_{14} &= x_1 \& t_{13} \\
y_1 &= t_{14} \oplus t_8 \oplus 1 \\
t_{16} &= t_9 \oplus x_1 \\
y_2 &= t_{13} \oplus x_1 \oplus 1 \\
y_3 &= t_{16} \oplus t_{14} \oplus 1 \\
y_0 &= y_3 \oplus t_{12} \oplus 1
\end{aligned}$$

$X^{13}:(5,4)$

$$\begin{aligned}
t_0 &= (x_3 \& x_0) \oplus 1 \\
t_1 &= (x_0 \& x_2) \oplus 1 \\
t_2 &= (x_1 \& x_2) \oplus 1 \\
t_3 &= (x_3 \& x_1) \oplus 1 \\
t_4 &= (x_3 \& t_2) \oplus 1 \\
t_5 &= t_3 \mid x_0 \\
t_6 &= t_1 \oplus x_3 \\
t_7 &= x_1 \mid t_1 \\
t_8 &= t_4 \oplus t_7 \oplus 1 \\
t_9 &= t_7 \oplus t_6 \oplus 1 \\
t_{10} &= t_0 \mid t_6 \\
t_{11} &= t_5 \oplus t_2 \oplus 1 \\
t_{12} &= t_{10} \oplus x_2 \\
t_{13} &= t_{11} \oplus t_8 \oplus 1 \\
t_{14} &= t_{11} \oplus x_0 \\
t_{15} &= (x_0 \& t_9) \oplus 1 \\
y_0 &= t_{15} \oplus x_1 \oplus 1 \\
y_3 &= t_{13} \oplus x_1 \oplus 1 \\
y_2 &= t_{14} \oplus t_{12} \\
y_1 &= t_3 \oplus t_{14}
\end{aligned}$$

$X^{-1}:(4,6)$

$$\begin{aligned}
t_0 &= (x_3 \& x_1) \oplus 1 \\
t_1 &= (x_2 \& x_0) \oplus 1 \\
t_2 &= (x_0 \& x_3) \oplus 1 \\
t_3 &= (x_3 \& x_1) \oplus 1 \\
t_4 &= (x_1 \& x_2) \oplus 1
\end{aligned}$$

$$\begin{aligned}
t5 &= (x2 \& x0) \oplus 1 \\
t6 &= x1 \mid t5 \\
t7 &= t3 \oplus t2 \oplus 1 \\
t8 &= t4 \mid x0 \\
t9 &= t3 \oplus t4 \oplus 1 \\
t10 &= (t2 \& x2) \oplus 1 \\
t11 &= x0 \oplus t0 \\
t12 &= t11 \oplus t9 \oplus 1 \\
t13 &= t11 \oplus t8 \oplus 1 \\
t14 &= t10 \oplus t1 \oplus 1 \\
t15 &= (t11 \& x2) \oplus 1 \\
t16 &= x0 \oplus t6 \\
t17 &= t10 \oplus t7 \oplus 1 \\
y0 &= (t7 \& t14) \oplus 1 \\
y3 &= t13 \oplus x1 \oplus 1 \\
y1 &= x3 \oplus t17 \oplus 1 \\
y2 &= (t9 \& t16) \oplus 1 \\
t23 &= t13 \oplus x3 \oplus 1
\end{aligned}$$

在此基础上，可以分别计算 S_0 和 S_1 的深度最优实现方案，其中 S_0 的深度为 18， S_1 的深度为 16。

S_0 :

$$\begin{aligned}
t0 &= x1 \oplus 1 \\
t1 &= x3 \oplus 1 \\
t2 &= x6 \oplus 1 \\
t3 &= x0 \oplus x2 \\
t4 &= x4 \oplus t2 \\
t5 &= x5 \oplus t0 \\
t6 &= t1 \oplus x7 \\
t7 &= x2 \oplus t2 \\
t8 &= t0 \oplus x7 \\
t9 &= x2 \& t0 \\
t10 &= t2 \& x7 \\
t11 &= t7 \& t8 \\
t12 &= t11 \oplus t10 \\
t13 &= t9 \oplus t10 \\
t14 &= x0 \oplus x4 \\
t15 &= x5 \oplus t1 \\
t16 &= x0 \& x5 \\
t17 &= x4 \& t1 \\
t18 &= t14 \& t15 \\
t19 &= t18 \oplus t16 \\
t20 &= t18 \oplus t17 \\
t21 &= t3 \oplus t4 \\
t22 &= t5 \oplus t6
\end{aligned}$$

$t_{23} = t_3 \& t_5$
 $t_{24} = t_4 \& t_6$
 $t_{25} = t_{21} \& t_{22}$
 $t_{26} = t_{25} \oplus t_{24}$
 $t_{27} = t_{23} \oplus t_{24}$
 $t_{28} = t_{26} \oplus t_{12}$
 $t_{29} = t_{27} \oplus t_{13}$
 $t_{30} = t_{19} \oplus t_{12}$
 $t_{31} = t_{20} \oplus t_{13}$
 $t_{32} = (t_{28} \& t_{30}) \oplus 1$
 $t_{33} = (t_{31} \& t_{29}) \oplus 1$
 $t_{34} = (t_{31} \& t_{28}) \oplus 1$
 $t_{35} = t_{34} \& t_{33}$
 $t_{36} = (t_{29} \& t_{30}) \oplus 1$
 $t_{37} = t_{36} | t_{35}$
 $t_{38} = (t_{36} \oplus t_{35}) \oplus 1$
 $t_{39} = (t_{33} \oplus t_{34}) \oplus 1$
 $t_{40} = (t_{32} \oplus t_{37}) \oplus 1$
 $t_{41} = (t_{28} \& t_{38}) \oplus 1$
 $t_{42} = t_{31} \oplus t_{40}$
 $t_{43} = (t_{30} \& t_{34}) \oplus 1$
 $t_{44} = t_{28} \oplus t_{38}$
 $t_{45} = (t_{41} \oplus t_{29}) \oplus 1$
 $t_{46} = t_{44} \oplus t_{45}$
 $t_{47} = t_{42} \oplus t_{46}$
 $t_{48} = t_{43} \oplus t_{44}$
 $t_{49} = t_{39} \oplus t_{44}$
 $t_{50} = t_{45} \oplus t_{48}$
 $t_{51} = t_{49} \oplus t_{47}$
 $t_{52} = t_{48} \oplus t_{47}$
 $t_{53} = t_{48} \& t_0$
 $t_{54} = t_{47} \& x_7$
 $t_{55} = t_{52} \& t_8$
 $t_{56} = t_{55} \oplus t_{54}$
 $t_{57} = t_{53} \oplus t_{54}$
 $t_{58} = t_{45} \oplus t_{49}$
 $t_{59} = t_{45} \& x_5$
 $t_{60} = t_{49} \& t_1$
 $t_{61} = t_{58} \& t_{15}$
 $t_{62} = t_{61} \oplus t_{59}$
 $t_{63} = t_{61} \oplus t_{60}$
 $t_{64} = t_{50} \oplus t_{51}$
 $t_{65} = t_{50} \& t_5$
 $t_{66} = t_{51} \& t_6$
 $t_{67} = t_{64} \& t_{22}$
 $t_{68} = t_{67} \oplus t_{66}$
 $t_{69} = t_{65} \oplus t_{66}$
 $t_{70} = t_{68} \oplus t_{56}$
 $t_{71} = t_{69} \oplus t_{57}$

$t_{72} = t_{62} \oplus t_{56}$
 $t_{73} = t_{63} \oplus t_{57}$
 $t_{74} = (x_2 \& x_4) \oplus 1$
 $t_{75} = (x_0 \& x_4) \oplus 1$
 $t_{76} = t_2 \mid t_{75}$
 $t_{77} = (t_2 \& x_4) \oplus 1$
 $t_{78} = (t_2 \& x_2) \oplus 1$
 $t_{79} = t_{74} \& t_{76}$
 $t_{80} = (t_{75} \oplus t_{77}) \oplus 1$
 $t_{81} = (t_2 \& x_0) \oplus 1$
 $t_{82} = t_{78} \& t_{74}$
 $t_{83} = (t_{81} \oplus t_{82}) \oplus 1$
 $t_{84} = t_{83} \& t_{80}$
 $t_{85} = (t_{79} \oplus t_{81}) \oplus 1$
 $t_{86} = t_{85} \oplus x_0$
 $t_{87} = x_4 \oplus t_{84}$
 $t_{88} = t_{82} \mid t_{85}$
 $t_{89} = t_{87} \& t_{86}$
 $t_{90} = t_{89} \oplus t_{88}$
 $t_{91} = (x_2 \oplus t_{84}) \oplus 1$
 $t_{92} = (t_{86} \& t_{80}) \oplus 1$
 $t_{93} = (t_{89} \oplus t_2) \oplus 1$
 $t_{94} = (x_7 \& t_0) \oplus 1$
 $t_{95} = (x_7 \& x_5) \oplus 1$
 $t_{96} = (t_0 \& t_1) \oplus 1$
 $t_{97} = t_{94} \oplus x_5$
 $t_{98} = x_5 \oplus t_{96}$
 $t_{99} = t_{97} \mid t_{98}$
 $t_{100} = (x_5 \& t_1) \oplus 1$
 $t_{101} = (t_{94} \oplus t_{96}) \oplus 1$
 $t_{102} = (t_{95} \oplus t_{99}) \oplus 1$
 $t_{103} = (t_{100} \oplus t_{101}) \oplus 1$
 $t_{104} = (t_{103} \oplus t_{95}) \oplus 1$
 $t_{105} = (t_{103} \& t_0) \oplus 1$
 $t_{106} = (t_{104} \oplus t_{105}) \oplus 1$
 $t_{107} = (t_{103} \& x_7) \oplus 1$
 $t_{108} = t_1 \& t_{107}$
 $t_{109} = t_{103} \oplus t_1$
 $t_{110} = (t_{108} \oplus t_{102}) \oplus 1$
 $t_{111} = (t_{107} \oplus t_1) \oplus 1$
 $t_{112} = (t_{109} \oplus t_{108}) \oplus 1$
 $t_{113} = (t_{112} \oplus t_{106}) \oplus 1$
 $t_{114} = t_{45} \mid t_{49}$
 $t_{115} = t_{48} \mid t_{47}$
 $t_{116} = t_{114} \mid t_{115}$
 $t_{117} = t_{93} \oplus t_{70}$
 $t_{118} = t_{117} \& t_{116}$
 $y_3 = t_{118} \oplus t_{93}$
 $t_{120} = t_{91} \oplus t_{71}$

$$\begin{aligned}
t121 &= t120 \& t116 \\
y0 &= t121 \oplus t91 \\
t123 &= t90 \oplus t72 \\
t124 &= t123 \& t116 \\
y4 &= (t124 \oplus t90) \oplus 1 \\
t126 &= t92 \oplus t73 \\
t127 &= t126 \& t116 \\
y7 &= (t127 \oplus t92) \oplus 1 \\
t129 &= t113 \oplus t45 \\
t130 &= t129 \& t116 \\
y1 &= (t130 \oplus t113) \oplus 1 \\
t132 &= t110 \oplus t49 \\
t133 &= t132 \& t116 \\
y5 &= (t133 \oplus t110) \oplus 1 \\
t135 &= t111 \oplus t48 \\
t136 &= t135 \& t116 \\
y6 &= t136 \oplus t111 \\
t138 &= t112 \oplus t47 \\
t139 &= t138 \& t116 \\
y2 &= t139 \oplus t112
\end{aligned}$$

S₁:

$$\begin{aligned}
t0 &= x0 \oplus 1; \\
t1 &= x2 \oplus 1; \\
t2 &= x5 \oplus 1; \\
t3 &= x4 \oplus t2; \\
t4 &= x6 \oplus x1; \\
t5 &= t0 \oplus t1; \\
t6 &= x7 \oplus x3; \\
t7 &= x4 \oplus x6; \\
t8 &= t0 \oplus x7; \\
t9 &= x4 \& t0; \\
t10 &= x6 \& x7; \\
t11 &= t7 \& t8; \\
t12 &= t11 \oplus t9; \\
t13 &= t11 \oplus t10; \\
t14 &= t2 \oplus x1; \\
t15 &= t1 \oplus x3; \\
t16 &= t2 \& t1; \\
t17 &= x1 \& x3; \\
t18 &= t14 \& t15; \\
t19 &= t18 \oplus t16; \\
t20 &= t18 \oplus t17; \\
t21 &= t3 \oplus t4; \\
t22 &= t5 \oplus t6; \\
t23 &= t3 \& t5; \\
t24 &= t4 \& t6; \\
t25 &= t21 \& t22; \\
t26 &= t23 \oplus t24; \\
t27 &= t25 \oplus t23; \\
t28 &= t26 \oplus t12;
\end{aligned}$$

$t_{29} = t_{27} \oplus t_{13};$
 $t_{30} = t_{26} \oplus t_{19};$
 $t_{31} = t_{27} \oplus t_{20};$
 $t_{32} = x_4 \oplus t_0;$
 $t_{33} = x_6 \oplus x_7;$
 $t_{34} = t_2 \oplus t_1;$
 $t_{35} = x_1 \oplus x_3;$
 $t_{36} = t_{32} \oplus t_{33};$
 $t_{37} = t_{33} \oplus t_{35};$
 $t_{38} = t_{32} \oplus t_{34};$
 $t_{39} = t_{28} \oplus t_{32};$
 $t_{40} = t_{29} \oplus t_{36};$
 $t_{41} = t_{30} \oplus t_{37};$
 $t_{42} = t_{31} \oplus t_{38};$
 $t_{43} = (t_{39} \& t_{41}) \oplus 1;$
 $t_{44} = (t_{40} \& t_{42}) \oplus 1;$
 $t_{45} = (t_{44} \oplus t_{43}) \oplus 1;$
 $t_{46} = (t_{42} \& t_{39}) \oplus 1;$
 $t_{47} = (t_{46} \oplus t_{45}) \oplus 1;$
 $t_{48} = (t_{41} \& t_{40}) \oplus 1;$
 $t_{49} = t_{46} \mid t_{44};$
 $t_{50} = (t_{44} \& t_{41}) \oplus 1;$
 $t_{51} = (t_{48} \oplus t_{49}) \oplus 1;$
 $t_{52} = t_{51} \oplus t_{39};$
 $t_{53} = (t_{47} \& t_{40}) \oplus 1;$
 $t_{54} = (t_{42} \& t_{47}) \oplus 1;$
 $t_{55} = t_{53} \oplus t_{52};$
 $t_{56} = (t_{41} \oplus t_{54}) \oplus 1;$
 $t_{57} = t_{45} \oplus t_{52};$
 $t_{58} = t_{50} \oplus t_{47};$
 $t_{59} = t_{58} \oplus t_{57};$
 $t_{60} = t_{56} \oplus t_{55};$
 $t_{61} = t_{58} \oplus t_{56};$
 $t_{62} = t_{58} \& t_0;$
 $t_{63} = t_{56} \& x_7;$
 $t_{64} = t_{61} \& t_8;$
 $t_{65} = t_{64} \oplus t_{62};$
 $t_{66} = t_{64} \oplus t_{63};$
 $t_{67} = t_{57} \oplus t_{55};$
 $t_{68} = t_{57} \& t_1;$
 $t_{69} = t_{55} \& x_3;$
 $t_{70} = t_{67} \& t_{15};$
 $t_{71} = t_{70} \oplus t_{68};$
 $t_{72} = t_{70} \oplus t_{69};$
 $t_{73} = t_{59} \oplus t_{60};$
 $t_{74} = t_{59} \& t_5;$
 $t_{75} = t_{60} \& t_6;$
 $t_{76} = t_{73} \& t_{22};$
 $t_{77} = t_{74} \oplus t_{75};$

$$\begin{aligned}
t78 &= t76 \oplus t74; \\
y6 &= (t77 \oplus t65) \oplus 1; \\
y1 &= (t78 \oplus t66) \oplus 1; \\
y4 &= (t77 \oplus t71) \oplus 1; \\
y2 &= t78 \oplus t72; \\
t83 &= t58 \& x4; \\
t84 &= t56 \& x6; \\
t85 &= t61 \& t7; \\
t86 &= t85 \oplus t83; \\
t87 &= t85 \oplus t84; \\
t88 &= t57 \& t2; \\
t89 &= t55 \& x1; \\
t90 &= t67 \& t14; \\
t91 &= t90 \oplus t88; \\
t92 &= t90 \oplus t89; \\
t93 &= t59 \& t3; \\
t94 &= t60 \& t4; \\
t95 &= t73 \& t21; \\
t96 &= t93 \oplus t94; \\
t97 &= t95 \oplus t93; \\
y5 &= (t96 \oplus t86) \oplus 1; \\
y0 &= t97 \oplus t87; \\
y3 &= (t96 \oplus t91) \oplus 1; \\
y7 &= (t97 \oplus t92) \oplus 1;
\end{aligned}$$

附录 B S_0 和 S_1 的一阶门限掩码方案

SMBA 算法的轮函数包含 4 个变换，其中 $W[K]$ 、 P 、 L 为线性变换， S 为唯一的非线性变换。线性变换的门限掩码方案是平凡的，我们只给出非线性变换 S 的门限掩码方案。非线性变换 S 包含了 2 个不同的代替盒 S_0 和 S_1 ，下面我们给出 S_0 和 S_1 的一阶门限掩码方案。

B.1 S_0 的一阶门限掩码方案

设 S_0 的门限实现的 8 比特输入为 x^1, x^2 ，则如下计算 S_0 的 8 比特输出 S_0^1 和 S_0^2 ：

- (1) $a^1 = M_0(x^1 \oplus \text{Con}_0)$ ，将 a^1 分为 2 个 4 比特 a_0^1 、 a_1^1 ，即 $a^1 = a_0^1 \parallel a_1^1$ ；
 $a^2 = M_0(x^2)$ ，将 a^2 分为 2 个 4 比特 a_0^2 、 a_1^2 ，即 $a^2 = a_0^2 \parallel a_1^2$ ；
- (2) $b^1 = a_0^1 \otimes a_1^1$ ， $b^2 = a_0^1 \otimes a_1^2$ ， $b^3 = a_0^2 \otimes a_1^1$ ， $b^4 = a_0^2 \otimes a_1^2$ ；
- (3) $b^1 = b^1 \oplus R_1 \oplus R_2$ ， $b^2 = b^2 \oplus R_2 \oplus R_3$ ， $b^3 = b^3 \oplus R_3 \oplus R_4$ ， $b^4 = b^4 \oplus R_4 \oplus R_1$ ；
- (4) $b^1 = b^1 \oplus b^2$ ， $b^2 = b^3 \oplus b^4$ ；
- (5) $(c^1, c^2) = \text{b13_TI}(b^1, b^2)$ ；

- (6) $d^1 = c^1 \otimes a_1^1, d^2 = c^1 \otimes a_1^2, d^3 = c^2 \otimes a_1^1, d^4 = c^2 \otimes a_1^2;$
- (7) $d^1 = d^1 \oplus R_1 \oplus R_2, d^2 = d^2 \oplus R_2 \oplus R_3, d^3 = d^3 \oplus R_3 \oplus R_4, d^4 = d^4 \oplus R_4 \oplus R_1;$
- (8) $d^1 = d^1 \oplus d^2, d^2 = d^3 \oplus d^4;$
- (9) $(e_0^1, e_0^2) = Q_0_TI(a_0^1, a_0^2), (e_1^1, e_1^2) = Q_1_TI(a_1^1, a_1^2);$
- (10) $(f^1, f^2) = E0E1_DC_SEL_TI(e_0^1, e_0^2, e_1^1, e_1^2, d^1, d^2, c^1, c^2);$
- (11) $S_0^1 = M_1(f^1) \oplus Con_1, S_0^2 = M_1(f^2)$

其中，在步骤（3）中的 R_1, R_2, R_3 和 R_4 分别为 4 比特随机数，在硬件实现中，步骤（3）的输出应用寄存器进行寄存。在步骤（7）中的 R_1, R_2, R_3 和 R_4 分别为 4 比特随机数，在硬件实现中，步骤（7）的输出应用寄存器进行寄存。

$(c^1, c^2) = b13_TI(b^1, b^2)$ 的计算过程如下：

- (1) 将 b^1 分为 2 个 2 比特 b_0^1, b_1^1 ，即 $b^1 = b_0^1 \parallel b_1^1$;
- (2) 将 b^2 分为 2 个 2 比特 b_0^2, b_1^2 ，即 $b^2 = b_0^2 \parallel b_1^2$
- (3) $c^1 = b_0^1 \oplus b_1^1, c^2 = b_0^2 \oplus b_1^2;$
- (4) $d^1 = (Z) \otimes (b_0^1 \otimes b_0^1), d^2 = (Z) \otimes (b_0^2 \otimes b_0^2);$
- (5) $b_1^1 = c^1 \otimes b_1^1 \oplus d^1, b_1^2 = c^1 \otimes b_1^2, b_1^3 = c^2 \otimes b_1^1, b_1^4 = c^2 \otimes b_1^2 \oplus d^2;$
- (6) $c_1^1 = b_1^1 \oplus R_1 \oplus R_2, c_1^2 = b_1^2 \oplus R_2 \oplus R_3, c_1^3 = b_1^3 \oplus R_3 \oplus R_4, c_1^4 = b_1^4 \oplus R_4 \oplus R_1;$
- (7) $d^1 = c_1^1 \oplus c_1^2, d^2 = c_1^3 \oplus c_1^4;$
- (8) $e^1 = INV_2(d^1), e^2 = INV_2(d^2);$
- (9) $f_0^1 = e^1 \otimes b_0^1, f_0^2 = e^1 \otimes b_0^2, f_0^3 = e^2 \otimes b_0^1, f_0^4 = e^2 \otimes b_0^2;$
- (10) $f_1^1 = e^1 \otimes c^1, f_1^2 = e^1 \otimes c^2, f_1^3 = e^2 \otimes c^1, f_1^4 = e^2 \otimes c^2;$
- (11) $f^1 = f_0^1 \parallel f_1^1, f^2 = f_0^2 \parallel f_1^2, f^3 = f_0^3 \parallel f_1^3, f^4 = f_0^4 \parallel f_1^4;$
- (12) $g^1 = f^1 \oplus R_1 \oplus R_2, g^2 = f^2 \oplus R_2 \oplus R_3, g^3 = f^3 \oplus R_3 \oplus R_4, g^4 = f^4 \oplus R_4 \oplus R_1;$
- (13) $h^1 = g^1 \oplus g^2, h^2 = g^3 \oplus g^4;$
- (14) $c^1 = SQU(h^1), c^2 = SQU(h^2)$

其中，在步骤（6）中的 R_1, R_2, R_3 和 R_4 分别为 2 比特随机数，在硬件实现中，步骤（6）的输出应用寄存器进行寄存。在步骤（12）中的 R_1, R_2, R_3 和 R_4 分别为 4 比特随机数，在硬件实现中，步骤（12）的输出应用寄存器进行寄存。
INV_2 表示 GF(2²) 在基(Z,1)上的求逆运算，SQU 表示 GF(2²) 在基(Z,1)上的平方。

$(e^1, e^2) = Q_0_TI(a^1, a^2)$ 的计算过程中使用的随机数均为 1 比特，在硬件实现

中，使用随机数对状态进行更新后应用寄存器进行寄存。具体计算过程如下：

- (1) $x^1 = a^1$;
- (2) $x^2 = a^2$;
- (3) 将 x^1 分为 4 个 1 比特 $x^1 = x_0^1 \parallel x_1^1 \parallel x_2^1 \parallel x_3^1$;
- (4) 将 x^2 分为 4 个 1 比特 $x^2 = x_0^2 \parallel x_1^2 \parallel x_2^2 \parallel x_3^2$;
- (5) $q_0^1 = 1 \oplus x_0^1 \oplus x_3^1$;
- (6) $q_0^2 = x_0^2 \oplus x_3^2$;
- (7) $q_1^1 = x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1$;
- (8) $q_1^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus x_3^2$;
- (9) $s^1 = q_0^1 \& q_1^1$;
- (10) $s^2 = q_0^2 \& q_1^1$;
- (11) $s^3 = q_0^1 \& q_1^2$;
- (12) $s^4 = q_0^2 \& q_1^2$;
- (13) $s^1 = s^1 \oplus R_1 \oplus R_2$;
- (14) $s^2 = s^2 \oplus R_2 \oplus R_3$;
- (15) $s^3 = s^3 \oplus R_3 \oplus R_4$;
- (16) $s^4 = s^4 \oplus R_4 \oplus R_1$;
- (17) $t_0^1 = s^1 \oplus s^2$;
- (18) $t_0^2 = s^3 \oplus s^4$;
- (19) $q_2^1 = 1 \oplus x_2^1 \oplus x_3^1 \oplus t_0^1$;
- (20) $q_2^2 = x_2^2 \oplus x_3^2 \oplus t_0^2$;
- (21) $q_3^1 = x_0^1 \oplus x_2^1 \oplus x_3^1$;
- (22) $q_3^2 = x_0^2 \oplus x_2^2 \oplus x_3^2$;
- (23) $s^1 = q_2^1 \& q_3^1$;
- (24) $s^2 = q_2^2 \& q_3^1$;
- (25) $s^3 = q_2^1 \& q_3^2$;
- (26) $s^4 = q_2^2 \& q_3^2$;
- (27) $s^1 = s^1 \oplus R_1 \oplus R_2$;
- (28) $s^2 = s^2 \oplus R_2 \oplus R_3$;

- (29) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (30) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (31) $t_1^1 = s^1 \oplus s^2;$
- (32) $t_1^2 = s^3 \oplus s^4;$
- (33) $q_4^1 = x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus t_0^1 \oplus t_1^1;$
- (34) $q_4^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus t_0^2 \oplus t_1^2;$
- (35) $q_5^1 = 1 \oplus x_2^1 \oplus t_1^1;$
- (36) $q_5^2 = x_2^2 \oplus t_1^2;$
- (37) $s^1 = q_4^1 \& q_5^1;$
- (38) $s^2 = q_4^2 \& q_5^1;$
- (39) $s^3 = q_4^1 \& q_5^2;$
- (40) $s^4 = q_4^2 \& q_5^2;$
- (41) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (42) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (43) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (44) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (45) $t_2^1 = s^1 \oplus s^2;$
- (46) $t_2^2 = s^3 \oplus s^4;$
- (47) $q_6^1 = x_2^1;$
- (48) $q_6^2 = x_2^2;$
- (49) $q_7^1 = 1 \oplus x_1^1 \oplus x_2^1 \oplus t_0^1 \oplus t_1^1;$
- (50) $q_7^2 = x_1^2 \oplus x_2^2 \oplus t_0^2 \oplus t_1^2;$
- (51) $s^1 = q_6^1 \& q_7^1;$
- (52) $s^2 = q_6^2 \& q_7^1;$
- (53) $s^3 = q_6^1 \& q_7^2;$
- (54) $s^4 = q_6^2 \& q_7^2;$
- (55) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (56) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (57) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (58) $s^4 = s^4 \oplus R_4 \oplus R_1;$

$$\begin{aligned}
(59) \quad & t_3^1 = s^1 \oplus s^2; \\
(60) \quad & t_3^2 = s^3 \oplus s^4; \\
(61) \quad & q_8^1 = x_0^1 \oplus t_0^1 \oplus t_2^1 \oplus t_3^1; \\
(62) \quad & q_8^2 = x_0^2 \oplus t_0^2 \oplus t_2^2 \oplus t_3^2; \\
(63) \quad & q_9^1 = 1 \oplus x_0^1 \oplus x_1^1 \oplus t_2^1 \oplus t_3^1; \\
(64) \quad & q_9^2 = x_0^2 \oplus x_1^2 \oplus t_2^2 \oplus t_3^2; \\
(65) \quad & s^1 = q_8^1 \& q_9^1; \\
(66) \quad & s^2 = q_8^2 \& q_9^1; \\
(67) \quad & s^3 = q_8^1 \& q_9^2; \\
(68) \quad & s^4 = q_8^2 \& q_9^2; \\
(69) \quad & s^1 = s^1 \oplus R_1 \oplus R_2; \\
(70) \quad & s^2 = s^2 \oplus R_2 \oplus R_3; \\
(71) \quad & s^3 = s^3 \oplus R_3 \oplus R_4; \\
(72) \quad & s^4 = s^4 \oplus R_4 \oplus R_1; \\
(73) \quad & t_4^1 = s^1 \oplus s^2; \\
(74) \quad & t_4^2 = s^3 \oplus s^4; \\
(75) \quad & y_0^1 = x_2^1 \oplus x_3^1 \oplus t_0^1 \oplus t_1^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1; \\
(76) \quad & y_0^2 = x_2^2 \oplus x_3^2 \oplus t_0^2 \oplus t_1^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2; \\
(77) \quad & y_1^1 = x_0^1 \oplus t_1^1 \oplus t_3^1 \oplus t_4^1; \\
(78) \quad & y_1^2 = x_0^2 \oplus t_1^2 \oplus t_3^2 \oplus t_4^2; \\
(79) \quad & y_2^1 = x_2^1 \oplus t_0^1 \oplus t_1^1 \oplus t_2^1 \oplus t_4^1; \\
(80) \quad & y_2^2 = x_2^2 \oplus t_0^2 \oplus t_1^2 \oplus t_2^2 \oplus t_4^2; \\
(81) \quad & y_3^1 = x_0^1 \oplus x_1^1 \oplus t_0^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1; \\
(82) \quad & y_3^2 = x_0^2 \oplus x_1^2 \oplus t_0^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2; \\
(83) \quad & e^1 = y_0^1 \parallel y_1^1 \parallel y_2^1 \parallel y_3^1; \\
(84) \quad & e^2 = y_0^2 \parallel y_1^2 \parallel y_2^2 \parallel y_3^2;
\end{aligned}$$

$(e^1, e^2) = Q_{1_TI}(a^1, a^2)$ 的计算过程中使用的随机数 R 均为 1 比特，在硬件实现中，使用随机数对状态进行更新后应用寄存器进行寄存，具体计算过程如下：

$$(1) \quad x^1 = a^1;$$

- (2) $x^2 = a^2$;
- (3) 将 x^1 分为 4 个 1 比特 $x^1 = x_0^1 \parallel x_1^1 \parallel x_2^1 \parallel x_3^1$;
- (4) 将 x^2 分为 4 个 1 比特 $x^2 = x_0^2 \parallel x_1^2 \parallel x_2^2 \parallel x_3^2$;
- (5) $q_0^1 = 1 \oplus x_0^1 \oplus x_3^1$;
- (6) $q_0^2 = x_0^2 \oplus x_3^2$;
- (7) $q_1^1 = 1 \oplus x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1$;
- (8) $q_1^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus x_3^2$;
- (9) $s^1 = q_0^1 \& q_1^1$;
- (10) $s^2 = q_0^2 \& q_1^1$;
- (11) $s^3 = q_0^1 \& q_1^2$;
- (12) $s^4 = q_0^2 \& q_1^2$;
- (13) $s^1 = s^1 \oplus R_1 \oplus R_2$;
- (14) $s^2 = s^2 \oplus R_2 \oplus R_3$;
- (15) $s^3 = s^3 \oplus R_3 \oplus R_4$;
- (16) $s^4 = s^4 \oplus R_4 \oplus R_1$;
- (17) $t_0^1 = s^1 \oplus s^2$;
- (18) $t_0^2 = s^3 \oplus s^4$;
- (19) $q_2^1 = x_0^1 \oplus x_1^1 \oplus x_3^1$;
- (20) $q_2^2 = x_0^2 \oplus x_1^2 \oplus x_3^2$;
- (21) $q_3^1 = 1 \oplus x_0^1 \oplus x_1^1 \oplus t_0^1$;
- (22) $q_3^2 = x_0^2 \oplus x_1^2 \oplus t_0^2$;
- (23) $s^1 = q_2^1 \& q_3^1$;
- (24) $s^2 = q_2^2 \& q_3^1$;
- (25) $s^3 = q_2^1 \& q_3^2$;
- (26) $s^4 = q_2^2 \& q_3^2$;
- (27) $s^1 = s^1 \oplus R_1 \oplus R_2$;
- (28) $s^2 = s^2 \oplus R_2 \oplus R_3$;
- (29) $s^3 = s^3 \oplus R_3 \oplus R_4$;
- (30) $s^4 = s^4 \oplus R_4 \oplus R_1$;
- (31) $t_1^1 = s^1 \oplus s^2$;

- (32) $t_1^2 = s^3 \oplus s^4;$
- (33) $q_4^1 = x_2^1 \oplus x_3^1;$
- (34) $q_4^2 = x_2^2 \oplus x_3^2;$
- (35) $q_5^1 = 1 \oplus x_1^1 \oplus t_0^1;$
- (36) $q_5^2 = x_1^2 \oplus t_0^2;$
- (37) $s^1 = q_4^1 \& q_5^1;$
- (38) $s^2 = q_4^2 \& q_5^1;$
- (39) $s^3 = q_4^1 \& q_5^2;$
- (40) $s^4 = q_4^2 \& q_5^2;$
- (41) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (42) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (43) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (44) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (45) $t_2^1 = s^1 \oplus s^2;$
- (46) $t_2^2 = s^3 \oplus s^4;$
- (47) $q_6^1 = x_0^1 \oplus x_1^1 \oplus t_1^1 \oplus t_2^1;$
- (48) $q_6^2 = x_0^2 \oplus x_1^2 \oplus t_1^2 \oplus t_2^2;$
- (49) $q_7^1 = 1 \oplus x_0^1 \oplus x_2^1 \oplus t_0^1 \oplus t_1^1;$
- (50) $q_7^2 = x_0^2 \oplus x_2^2 \oplus t_0^2 \oplus t_1^2;$
- (51) $s^1 = q_6^1 \& q_7^1;$
- (52) $s^2 = q_6^2 \& q_7^1;$
- (53) $s^3 = q_6^1 \& q_7^2;$
- (54) $s^4 = q_6^2 \& q_7^2;$
- (55) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (56) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (57) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (58) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (59) $t_3^1 = s^1 \oplus s^2;$
- (60) $t_3^2 = s^3 \oplus s^4;$
- (61) $q_8^1 = x_1^1 \oplus x_2^1 \oplus t_0^1 \oplus t_1^1;$

$$(62) \quad q_8^2 = x_1^2 \oplus x_2^2 \oplus t_0^2 \oplus t_1^2;$$

$$(63) \quad q_9^1 = x_2^1 \oplus t_0^1 \oplus t_1^1 \oplus t_3^1;$$

$$(64) \quad q_9^2 = x_2^2 \oplus t_0^2 \oplus t_1^2 \oplus t_3^2;$$

$$(65) \quad s^1 = q_8^1 \& q_9^1;$$

$$(66) \quad s^2 = q_8^2 \& q_9^1;$$

$$(67) \quad s^3 = q_8^1 \& q_9^2;$$

$$(68) \quad s^4 = q_8^2 \& q_9^2;$$

$$(69) \quad s^1 = s^1 \oplus R_1 \oplus R_2;$$

$$(70) \quad s^2 = s^2 \oplus R_2 \oplus R_3;$$

$$(71) \quad s^3 = s^3 \oplus R_3 \oplus R_4;$$

$$(72) \quad s^4 = s^4 \oplus R_4 \oplus R_1;$$

$$(73) \quad t_4^1 = s^1 \oplus s^2;$$

$$(74) \quad t_4^2 = s^3 \oplus s^4;$$

$$(75) \quad y_0^1 = x_2^1 \oplus x_3^1 \oplus t_0^1 \oplus t_1^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1;$$

$$(76) \quad y_0^2 = x_2^2 \oplus x_3^2 \oplus t_0^2 \oplus t_1^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2;$$

$$(77) \quad y_1^1 = x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1 \oplus t_2^1;$$

$$(78) \quad y_1^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus x_3^2 \oplus t_2^2;$$

$$(79) \quad y_2^1 = x_1^1 \oplus x_3^1 \oplus t_0^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1;$$

$$(80) \quad y_2^2 = x_1^2 \oplus x_3^2 \oplus t_0^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2;$$

$$(81) \quad y_3^1 = x_1^1 \oplus x_2^1 \oplus x_3^1 \oplus t_2^1 \oplus t_3^1;$$

$$(82) \quad y_3^2 = x_1^2 \oplus x_2^2 \oplus x_3^2 \oplus t_2^2 \oplus t_3^2;$$

$$(83) \quad e^1 = y_0^1 \parallel y_1^1 \parallel y_2^1 \parallel y_3^1;$$

$$(84) \quad e^2 = y_0^2 \parallel y_1^2 \parallel y_2^2 \parallel y_3^2;$$

$(f^1, f^2) = E0E1_DC_SEL_TI(e_0^1, e_0^2, e_1^1, e_1^2, d^1, d^2, c^1, c^2)$ 的计算过程如下:

$$(1) \quad \text{将 } c^1 \text{ 分为 4 个 1 比特 } c^1 = c_0^1 \parallel c_1^1 \parallel c_2^1 \parallel c_3^1;$$

$$(2) \quad \text{将 } c^2 \text{ 分为 4 个 1 比特 } c^2 = c_0^2 \parallel c_1^2 \parallel c_2^2 \parallel c_3^2;$$

$$(3) \quad y_1 = c_3^1 \oplus c_2^1 \oplus c_1^1 \oplus c_0^1 \oplus c_0^1 \& c_2^1 \oplus c_0^1 \& c_1^1 \oplus c_2^1 \& c_3^1 \oplus c_1^1 \& c_3^1 \oplus c_0^1 \& c_1^1 \& c_2^1 \& c_3^1 \oplus c_0^1 \& c_1^1 \& c_2^1 \oplus c_0^1 \& c_1^1 \& c_3^1 \oplus c_0^1 \& c_2^1 \& c_3^1 \oplus c_1^1 \& c_2^1 \& c_3^1$$

- (4) $y_2 = c_0^2 \oplus c_0^2 \& c_2^1 \oplus c_0^2 \& c_1^1 \oplus c_0^2 \& c_1^1 \& c_2^1 \& c_3^1 \oplus c_0^2 \& c_1^1 \& c_2^1 \oplus c_0^2 \& c_1^1 \& c_3^1 \oplus c_0^2 \& c_2^1 \& c_3^1$
- (5) $y_3 = c_1^2 \oplus c_0^1 \& c_3^1 \oplus c_0^1 \& c_1^2 \oplus c_1^2 \& c_3^1 \oplus c_0^1 \& c_1^2 \& c_2^1 \& c_3^1 \oplus c_0^1 \& c_1^2 \& c_2^1 \oplus c_0^1 \& c_1^2 \& c_3^1 \oplus c_1^2 \& c_2^1 \& c_3^1$
- (6) $y_4 = c_0^2 \& c_3^1 \oplus c_0^2 \& c_1^2 \oplus c_0^2 \& c_1^2 \& c_2^1 \& c_3^1 \oplus c_0^2 \& c_1^2 \& c_2^1 \oplus c_0^2 \& c_1^2 \& c_3^1$
- (7) $y_5 = c_2^2 \oplus c_0^1 \& c_2^2 \oplus c_2^2 \& c_3^1 \oplus c_0^1 \& c_1^1 \& c_2^2 \& c_3^1 \oplus c_0^1 \& c_1^1 \& c_2^2 \oplus c_0^1 \& c_2^2 \& c_3^1 \oplus c_1^1 \& c_2^2 \& c_3^1$
- (8) $y_6 = c_0^2 \& c_2^2 \oplus c_0^2 \& c_1^1 \& c_2^2 \& c_3^1 \oplus c_0^2 \& c_1^1 \& c_2^2 \oplus c_0^2 \& c_2^2 \& c_3^1$
- (9) $y_7 = c_1^2 \& c_2^2 \oplus c_0^1 \& c_1^2 \& c_2^2 \& c_3^1 \oplus c_0^1 \& c_1^2 \& c_2^2 \oplus c_1^2 \& c_2^2 \& c_3^1$
- (10) $y_8 = c_0^2 \& c_1^2 \& c_2^2 \& c_3^1 \oplus c_0^2 \& c_1^2 \& c_2^2$
- (11) $y_9 = c_3^2 \oplus c_1^1 \& c_2^1 \oplus c_2^1 \& c_3^2 \oplus c_0^1 \& c_3^2 \oplus c_0^1 \& c_1^1 \& c_2^1 \& c_3^2 \oplus c_0^1 \& c_1^1 \& c_2^2 \oplus c_0^1 \& c_2^1 \& c_3^2 \oplus c_1^1 \& c_2^1 \& c_3^2$
- (12) $y_{10} = c_0^2 \& c_3^2 \oplus c_0^2 \& c_1^1 \& c_2^1 \& c_3^2 \oplus c_0^2 \& c_1^1 \& c_3^2 \oplus c_0^2 \& c_2^1 \& c_3^2$
- (13) $y_{11} = c_1^2 \& c_2^1 \oplus c_0^1 \& c_1^2 \& c_2^1 \& c_3^2 \oplus c_0^1 \& c_1^2 \& c_3^2 \oplus c_1^2 \& c_2^1 \& c_3^2$
- (14) $y_{12} = c_0^2 \& c_1^2 \& c_2^1 \& c_3^2 \oplus c_0^2 \& c_1^2 \& c_3^2$
- (15) $y_{13} = c_1^1 \& c_2^2 \oplus c_2^2 \& c_3^2 \oplus c_0^1 \& c_1^1 \& c_2^2 \& c_3^2 \oplus c_0^1 \& c_2^2 \& c_3^2 \oplus c_1^1 \& c_2^2 \& c_3^2$
- (16) $y_{14} = c_1^1 \& c_3^2 \oplus c_0^2 \& c_1^1 \& c_2^2 \& c_3^2 \oplus c_0^2 \& c_2^2 \& c_3^2$
- (17) $y_{15} = c_0^1 \& c_1^2 \& c_2^2 \& c_3^2 \oplus c_1^2 \& c_2^2 \& c_3^2$
- (18) $y_{16} = c_0^2 \& c_1^2 \& c_2^2 \& c_3^2 \oplus c_1^2 \& c_3^2$
- (19) $y_1 = y_1 \oplus R_1 \oplus R_2, y_2 = y_2 \oplus R_2 \oplus R_3, y_3 = y_3 \oplus R_3 \oplus R_4, y_4 = y_4 \oplus R_4 \oplus R_5$
 $y_5 = y_5 \oplus R_5 \oplus R_6, y_6 = y_6 \oplus R_6 \oplus R_7, y_7 = y_7 \oplus R_7 \oplus R_8, y_8 = y_8 \oplus R_8 \oplus R_9$
 $y_9 = y_9 \oplus R_9 \oplus R_{10}, y_{10} = y_{10} \oplus R_{10} \oplus R_{11}, y_{11} = y_{11} \oplus R_{11} \oplus R_{12},$
 $y_{12} = y_{12} \oplus R_{12} \oplus R_{13}$
 $y_{13} = y_{13} \oplus R_{13} \oplus R_{14}, y_{14} = y_{14} \oplus R_{14} \oplus R_{15}, y_{15} = y_{15} \oplus R_{15} \oplus R_{16},$
 $y_{16} = y_{16} \oplus R_{16} \oplus R_1$
- (20) $g^1 = y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5 \oplus y_6 \oplus y_7 \oplus y_8,$
 $g^2 = y_9 \oplus y_{10} \oplus y_{11} \oplus y_{12} \oplus y_{13} \oplus y_{14} \oplus y_{15} \oplus y_{16}$
- (21) $h^1 = g^1 \parallel g^1 \parallel g^1 \parallel g^1 \parallel g^1 \parallel g^1 \parallel g^1, h^2 = g^2 \parallel g^2 \parallel g^2 \parallel g^2 \parallel g^2 \parallel g^2 \parallel g^2$
- (22) $e^1 = e_0^1 \parallel e_1^1, e^2 = e_0^2 \parallel e_1^2, f^1 = d^1 \parallel c^1, f^2 = d^2 \parallel c^2$

$$(23) \quad t^1 = e^1 \oplus f^1, \quad t^2 = e^2 \oplus f^2$$

$$(24) \quad b^1 = t^1 \& h^1, \quad b^2 = t^1 \& h^2, \quad b^3 = t^2 \& h^1, \quad b^4 = t^2 \& h^2;$$

$$(25) \quad b^1 = b^1 \oplus R_1 \oplus R_2, \quad b^2 = b^2 \oplus R_2 \oplus R_3, \quad b^3 = b^3 \oplus R_3 \oplus R_4, \quad b^4 = b^4 \oplus R_4 \oplus R_1;$$

$$(26) \quad b^1 = b^1 \oplus b^2, \quad b^2 = b^3 \oplus b^4;$$

$$(27) \quad f^1 = b^1 \oplus e^1, \quad f^2 = b^2 \oplus e^2$$

其中，在步骤（19）中的 $R_1, R_2, R_3, \dots, R_{16}$ 分别为 1 比特随机数，在硬件实现中，步骤（19）的输出应用寄存器进行寄存。在步骤（25）中的 R_1, R_2, R_3 和 R_4 分别为 8 比特随机数，在硬件实现中，步骤（25）的输出应用寄存器进行寄存。

B.2 S_1 的一阶门限掩码方案

设 S_1 的一阶门限实现的 8 比特输入为 x^1, x^2 ，则如下计算 S_1 的 8 比特输出 S_1^1 和 S_1^2 ：

$$(1) \quad a^1 = M_2(x^1 \oplus \text{Con}_2), \quad \text{将 } a^1 \text{ 分为 2 个 4 比特 } a_0^1, a_1^1, \text{ 即 } a^1 = a_0^1 \parallel a_1^1;$$

$$a^2 = M_2(x^2), \quad \text{将 } a^2 \text{ 分为 2 个 4 比特 } a_0^2, a_1^2, \text{ 即 } a^2 = a_0^2 \parallel a_1^2;$$

$$(2) \quad b_0^1 = a_0^1 \oplus a_1^1, \quad b_0^2 = a_0^2 \oplus a_1^2;$$

$$(3) \quad c^1 = (Z^2 Y^4) \otimes (b_0^1 \otimes b_0^1); \quad c^2 = (Z^2 Y^4) \otimes (b_0^2 \otimes b_0^2);$$

$$(4) \quad b_1^1 = a_0^1 \otimes a_1^1 \oplus c^1, \quad b_1^2 = a_0^1 \otimes a_1^2, \quad b_1^3 = a_0^2 \otimes a_1^1, \quad b_1^4 = a_0^2 \otimes a_1^2 \oplus c^2;$$

$$(5) \quad c_1^1 = b_1^1 \oplus R_1 \oplus R_2, \quad c_1^2 = b_1^2 \oplus R_2 \oplus R_3, \quad c_1^3 = b_1^3 \oplus R_3 \oplus R_4, \quad c_1^4 = b_1^4 \oplus R_4 \oplus R_1;$$

$$(6) \quad d^1 = c_1^1 \oplus c_1^2, \quad d^2 = c_1^3 \oplus c_1^4;$$

$$(7) \quad (e^1, e^2) = \text{INV_4_TI}(d^1, d^2);$$

$$(8) \quad f_0^1 = e^1 \otimes a_1^1, \quad f_0^2 = e^1 \otimes a_1^2, \quad f_0^3 = e^2 \otimes a_1^1, \quad f_0^4 = e^2 \otimes a_1^2;$$

$$(9) \quad f_1^1 = e^1 \otimes a_0^1, \quad f_1^2 = e^1 \otimes a_0^2, \quad f_1^3 = e^2 \otimes a_0^1, \quad f_1^4 = e^2 \otimes a_0^2;$$

$$(10) \quad f^1 = f_0^1 \parallel f_1^1, \quad f^2 = f_0^2 \parallel f_1^2, \quad f^3 = f_0^3 \parallel f_1^3, \quad f^4 = f_0^4 \parallel f_1^4;$$

$$(11) \quad g^1 = f^1 \oplus R_1 \oplus R_2, \quad g^2 = f^2 \oplus R_2 \oplus R_3, \quad g^3 = f^3 \oplus R_3 \oplus R_4, \quad g^4 = f^4 \oplus R_4 \oplus R_1;$$

$$(12) \quad h^1 = g^1 \oplus g^2, \quad h^2 = g^3 \oplus g^4;$$

$$(13) \quad S_1^1 = M_3(h^1) \oplus \text{Con}_3, \quad S_1^2 = M_3(h^2)$$

其中，在步骤（5）中的 R_1, R_2, R_3 和 R_4 分别为 4 比特随机数，在硬件实现中，步骤（5）的输出应用寄存器进行寄存。在步骤（11）中的 R_1, R_2, R_3 和 R_4 分别为 8 比特随机数，在硬件实现中，步骤（11）的输出应用寄存器进行寄存。

$(e^1, e^2) = \text{INV_4_TI}(d^1, d^2)$ 的计算过程如下:

- (1) 将 d^1 分为 2 个 2 比特 d_0^1, d_1^1 , 即 $d^1 = d_0^1 \parallel d_1^1$;
- (2) 将 d^2 分为 2 个 2 比特 d_0^2, d_1^2 , 即 $d^2 = d_0^2 \parallel d_1^2$;
- (3) $b_0^1 = d_0^1 \oplus d_1^1, b_0^2 = d_0^2 \oplus d_1^2$;
- (4) $c^1 = (Z) \otimes (b_0^1 \otimes b_0^1); c^2 = (Z) \otimes (b_0^2 \otimes b_0^2)$;
- (5) $b_1^1 = d_0^1 \otimes d_1^1 \oplus c^1, b_1^2 = d_0^1 \otimes d_1^2, b_1^3 = d_0^2 \otimes d_1^1, b_1^4 = d_0^2 \otimes d_1^2 \oplus c^2$;
- (6) $c_1^1 = b_1^1 \oplus R_1 \oplus R_2, c_1^2 = b_1^2 \oplus R_2 \oplus R_3, c_1^3 = b_1^3 \oplus R_3 \oplus R_4, c_1^4 = b_1^4 \oplus R_4 \oplus R_1$;
- (7) $d^1 = c_1^1 \oplus c_1^2, d^2 = c_1^3 \oplus c_1^4$;
- (8) $e^1 = \text{INV_2}(d^1), e^2 = \text{INV_2}(d^2)$;
- (9) $f_0^1 = e^1 \otimes d_1^1, f_0^2 = e^1 \otimes d_1^2, f_0^3 = e^2 \otimes d_1^1, f_0^4 = e^2 \otimes d_1^2$;
- (10) $f_1^1 = e^1 \otimes d_0^1, f_1^2 = e^1 \otimes d_0^2, f_1^3 = e^2 \otimes d_0^1, f_1^4 = e^2 \otimes d_0^2$;
- (11) $f^1 = f_0^1 \parallel f_1^1, f^2 = f_0^2 \parallel f_1^2, f^3 = f_0^3 \parallel f_1^3, f^4 = f_0^4 \parallel f_1^4$;
- (12) $g^1 = f^1 \oplus R_1 \oplus R_2, g^2 = f^2 \oplus R_2 \oplus R_3, g^3 = f^3 \oplus R_3 \oplus R_4, g^4 = f^4 \oplus R_4 \oplus R_1$;
- (13) $e^1 = g^1 \oplus g^2, e^2 = g^3 \oplus g^4$

其中, 在步骤(6)中的 R_1, R_2, R_3 和 R_4 分别为2比特随机数, 在硬件实现中, 步骤(6)的输出应用寄存器进行寄存。在步骤(12)中的 R_1, R_2, R_3 和 R_4 分别为4比特随机数, 在硬件实现中, 步骤(12)的输出应用寄存器进行寄存。 INV_2 表示 $\text{GF}(2^2)$ 在基 (Z^2, Z) 上的求逆。