

SMBA 分组算法

设计文档

兴唐通信科技有限公司

2019 年 10 月

目 次

| | |
|---|----|
| 1. 算法描述..... | 1 |
| 1.1. 概述..... | 1 |
| 1.2. 符号、运算符和约定..... | 1 |
| 1.2.1. 符号和运算符列表..... | 1 |
| 1.2.2. 比特、字节和子块顺序约定..... | 2 |
| 1.3. 加解密过程..... | 2 |
| 1.3.1. 加密过程..... | 2 |
| 1.3.2. 解密过程..... | 3 |
| 1.4. 轮函数及其组件..... | 3 |
| 1.4.1. 轮函数 $F[K]$ | 3 |
| 1.4.2. 白化变换 $W[K]$ | 3 |
| 1.4.3. 换位变换 P | 3 |
| 1.4.4. 代替变换 S | 4 |
| 1.4.5. 线性变换 L | 4 |
| 1.5. 密钥扩展..... | 5 |
| 1.5.1. 变换 α 、 β 、 γ | 6 |
| 1.5.2. SMBA-128-128 的子密钥生成..... | 7 |
| 1.5.3. SMBA-128-256 的子密钥生成..... | 7 |
| 1.5.4. SMBA-256-256 的子密钥生成..... | 8 |
| 1.6. S 盒 S_0 和 S_1 | 9 |
| 1.7. 样本数据..... | 10 |
| 1.7.1. SMBA-128-128 样本数据..... | 10 |
| 1.7.2. SMBA-128-256 样本数据..... | 11 |
| 1.7.3. SMBA-256-256 样本数据..... | 12 |
| 1.8. 密码算法设计框图..... | 14 |
| 1.9. S 盒 S_0 、 S_1 构造过程..... | 18 |
| 1.9.1. S_0 构造过程..... | 18 |
| 1.9.2. S_1 构造过程..... | 20 |
| 2. 设计原理..... | 23 |
| 2.1. 加解密过程设计原理..... | 23 |
| 2.2. 密钥扩展设计原理..... | 23 |
| 2.3. S 盒设计原理..... | 24 |
| 2.4. 线性变换和换位变换设计原理..... | 24 |

| | |
|--|----|
| 3. 安全性分析..... | 26 |
| 3.1. 整体结构安全性..... | 26 |
| 3.2. 密码部件安全性..... | 26 |
| 3.2.1. 线性部件安全性..... | 26 |
| 3.2.2. 非线性部件安全性..... | 28 |
| 3.3. 抗攻击安全性..... | 31 |
| 3.3.1. 差分攻击..... | 31 |
| 3.3.2. 线性攻击..... | 34 |
| 3.3.3. 代数攻击..... | 35 |
| 3.3.4. 积分攻击..... | 36 |
| 3.3.5. 不可能差分攻击..... | 36 |
| 3.3.6. 相关密钥差分攻击..... | 39 |
| 3.3.7. 飞去来器攻击..... | 41 |
| 3.3.8. 滑动攻击..... | 42 |
| 3.3.9. 密钥扩展安全性..... | 43 |
| 3.4. 抗侧信道攻击安全防护..... | 47 |
| 3.5. 依赖性检测..... | 48 |
| 3.5.1. 检测项目..... | 48 |
| 3.5.2. SMBA-128 检测结果及分析..... | 50 |
| 3.5.3. SMBA-256 检测结果及分析..... | 51 |
| 3.6. 随机性检测..... | 52 |
| 3.6.1. 显著性水平..... | 52 |
| 3.6.2. 检测项目..... | 52 |
| 3.6.3. 密文随机性检测..... | 52 |
| 3.6.4. 检测结果..... | 53 |
| 4. 性能分析..... | 59 |
| 4.1. 软件实现..... | 59 |
| 4.1.1. 8 位平台实现..... | 59 |
| 4.1.2. 32 位 ARM 环境下的算法速率优化 C 实现..... | 60 |
| 4.1.3. 64 位 Windows 环境下的算法速率优化 C 实现..... | 61 |
| 4.2. 硬件实现..... | 62 |
| 4.2.1. Verilog 硬件仿真实现..... | 62 |
| 4.2.2. FPGA 评估结果..... | 62 |
| 5. 优缺点分析..... | 65 |

| | |
|------------------------------------|----|
| 5.1. 算法特色..... | 65 |
| 5.2. 创新点..... | 65 |
| 5.3. 不足..... | 66 |
| 6. 参考文献..... | 67 |
| 附录 A S_0 和 S_1 深度优先实现 | 68 |
| 附录 B S_0 和 S_1 的一阶门限掩码方案 | 78 |
| B.1 S_0 的一阶门限掩码方案 | 78 |
| B.2 S_1 的一阶门限掩码方案 | 86 |

1. 算法描述

1.1. 概述

SMBA 是一个分组密码算法，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。分组长度（比特数，记为 BL）和密钥长度（比特数，记为 KL）对应的 SMBA 的迭代轮数（记为 N）如下表所示。

| 分组长度 \ 密钥长度 | 128 比特 | 256 比特 |
|-------------|--------|--------|
| | 128 比特 | 256 比特 |
| 128 比特 | 18 | 24 |
| 256 比特 | — | 24 |

分组长度为 128 比特的 SMBA 记为 SMBA-128，其中密钥长度为 128 或 256 比特时也分别记为 SMBA-128-128 或 SMBA-128-256。分组长度为 256 比特的 SMBA 记为 SMBA-256，也可记为 SMBA-256-256。

需要说明的是，本章所述的图都在“1.8 算法设计框图”中给出。

1.2. 符号、运算符和约定

1.2.1. 符号和运算符列表

| | |
|-------------|--------------------------------------|
| 0x | 用前缀0x表示16进制数。 |
| = | 赋值运算符，也可表示相等。 |
| \bar{h} | h的逐比特取补。 |
| \oplus | 逐比特异或运算符。 |
| \vee | 逐比特或运算符。 |
| $\&$ | 逐比特与运算符，其运算优先级高于 \oplus 和 \vee 。 |
| \parallel | 字符串连接运算符。 |
| $\lll n$ | 操作数循环左移n比特运算符。 |
| $MSB_t(x)$ | 由x的最左（即最高）t比特组成的t比特数。 |
| $LSB_t(x)$ | 由x的最右（即最低）t比特组成的t比特数。 |

| | |
|---------|--|
| M^T | 向量或矩阵 M 的转置。 |
| 0^n | n 比特0组成的比特串。 |
| \circ | 变换复合运算符，即对于变换 f_1 、 f_2 和变量 x ： $f_2 \circ f_1(x) = f_2(f_1(x))$ 。 |
| BL | 分组长度，比特数。 |
| KL | 密钥长度，比特数。 |
| N | 迭代轮数。 |

1.2.2. 比特、字节和子块顺序约定

所有数据变量用最高比特（或子块）在左边，最低比特（或子块）在右边表示，而且从左到右从0开始编号。例如，一个 n 比特数从左到右分别称为它的第0比特、第1比特、第2比特、...、第 $n-1$ 比特。当一个变量被分成几个子块时，最左边（最高）子块由原始数据的最高部分组成，等等一直到最低。

当密码算法的输入（或输出）是字节流时，我们按地址小的是高位（即左边），地址大的是低位（即右边）的顺序把每4（或2）个字节组成一个32（或16）比特字。32（或16）比特字之间的顺序是先左后右，即存储时，左边地址小，右边地址大。

约定比特串 $x_0||x_1||x_2||\dots||x_{n-1}$ 、整数 $2^{n-1}x_0+2^{n-2}x_1+2^{n-3}x_2+\dots+2x_{n-2}+x_{n-1}$ 、行向量 $(x_0,x_1,x_2,\dots,x_{n-1})$ 、列向量 $(x_0,x_1,x_2,\dots,x_{n-1})^T$ 是等价的，可互相转化。

1.3. 加解密过程

1.3.1. 加密过程

SMBA 的加密过程（参见图 1.8-1） $\text{ciphertext} = E_{\text{Key}}(\text{plaintext})$ 如下：

(1) 将 BL 比特明文 plaintext 分为各 $BL/2$ 比特的左、右两部分 x_0 和 x_1 ，即 $\text{plaintext} = x_0||x_1$ 。

(2) 对于 $i=0,1,\dots,N-1$

$$x_{i+2} = x_i \oplus F[EK_i](x_{i+1})$$

(3) x_{N+1} 和 x_N 连接成 BL 比特密文 ciphertext ，即 $\text{ciphertext} = x_{N+1}||x_N$ 。

其中， F 为轮函数， EK_i （ $0 \leq i \leq N-1$ ）为 N 个 $BL/2$ 比特的加密子密钥，由密钥 Key 用密钥扩展派生。

1.3.2. 解密过程

SMBA 的解密过程（参见图 1.8-2） $\text{plaintext} = D_{\text{Key}}(\text{ciphertext})$ 如下：

(1) 将 BL 比特密文 ciphertext 分为各 BL/2 比特的左、右两部分 x_0 和 x_1 ，即

$$\text{ciphertext} = x_0 || x_1。$$

(2) 对于 $i=0,1,\dots,N-1$

$$x_{i+2} = x_i \oplus F[DK_i](x_{i+1})$$

(3) x_{N+1} 和 x_N 连接成 BL 比特明文 plaintext ，即 $\text{plaintext} = x_{N+1} || x_N$ 。

其中，F为轮函数， DK_i ($0 \leq i \leq N-1$) 为N个BL/2比特的解密子密钥，由密钥Key用密钥扩展派生。

解密子密钥与加密子密钥满足如下关系： $DK_i = EK_{N-1-i}$ ， $0 \leq i \leq N-1$ 。

1.4. 轮函数及其组件

1.4.1. 轮函数 $F[K]$

轮函数 $y = F[K](x)$ 的输入是一个BL/2比特数 x 和一个BL/2比特子密钥 K ，输出是一个BL/2比特数 y ， $y = F[K](x) = L(S(P(W[K](x))))$ ，即 $F[K] = L \circ S \circ P \circ W[K]$ （参见图1.8-3、图1.8-4）。各变换 $W[K]$ 、 P 、 S 、 L 下面定义。

1.4.2. 白化变换 $W[K]$

对于BL/2比特数 x 和BL/2比特子密钥 K ， $W[K](x) = x \oplus K$ 。

1.4.3. 换位变换 P

换位变换 P 对于 SMBA-128 使用换位变换 P_{64} ，对于 SMBA-256 使用换位变换 P_{128} 。

1.4.3.1. 换位变换 P_{64}

P_{64} 是一个 64 比特数到 64 比特数的变换，具体为：把 64 比特数 x 分为 8 个 8 比特数 x_i ， $0 \leq i \leq 7$ ，即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7$$

令 $y_i = x_{\tau(i)}$ ，则

$$P_{64}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7$$

其中 $\tau(i)$ ， $0 \leq i \leq 7$ ，依次为：3,4,5,6, 0,1,2,7。

1.4.3.2. 换位变换 P_{128}

P_{128} 是一个 128 比特数到 128 比特数的变换，具体为：把 128 比特数 x 分为 16 个 8 比特数 x_i ， $0 \leq i \leq 15$ ，即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7 || x_8 || x_9 || x_{10} || x_{11} || x_{12} || x_{13} || x_{14} || x_{15}$$

令 $y_i = x_{pi(i)}$ ，则

$$P_{128}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7 || y_8 || y_9 || y_{10} || y_{11} || y_{12} || y_{13} || y_{14} || y_{15}$$

其中 $pi(i)$ ， $0 \leq i \leq 15$ ，依次为：3,4,5,6, 10,11,12,13, 8,9,14,15, 0,1,2,7。

1.4.4. 代替变换 S

代替变换 S 对于 SMBA-128 使用代替变换 S_{64} ，对于 SMBA-256 使用代替变换 S_{128} 。

1.4.4.1. 代替变换 S_{32}

S_{32} 是一个 32 比特数到 32 比特数的变换，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

$$y_0 = S_0(x_0), \quad y_1 = S_1(x_1), \quad y_2 = S_0(x_2), \quad y_3 = S_1(x_3)$$

则 $S_{32}(x) = y_0 || y_1 || y_2 || y_3$ ，其中， S_0 、 S_1 都是 8 比特到 8 比特的代替盒（S 盒）。

1.4.4.2. 代替变换 S_{64}

S_{64} 是一个 64 比特数到 64 比特数的变换，具体为：将 64 比特数 x 分为 2 个 32 比特数，即 $x = x_0 || x_1$ ，令

$$y_0 = S_{32}(x_0), \quad y_1 = S_{32}(x_1)$$

则 $S_{64}(x) = y_0 || y_1$ 。

1.4.4.3. 代替变换 S_{128}

S_{128} 是一个 128 比特数到 128 比特数的变换，具体为：将 128 比特数 x 分为 2 个 64 比特数，即 $x = x_0 || x_1$ ，令

$$y_0 = S_{64}(x_0), \quad y_1 = S_{64}(x_1)$$

则 $S_{128}(x) = y_0 || y_1$ 。

1.4.5. 线性变换 L

线性变换 L 对于 SMBA-128 使用线性变换 L_{64} ，对于 SMBA-256 使用线性变

换 L_{128} 。

1.4.5.1. 线性变换 L_{32}

L_{32} 是32比特数到32比特数的变换，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 $L_{32}(x) = y_0||y_1||y_2||y_3$ 。

1.4.5.2. 线性变换 L_{64}

L_{64} 是一个64比特数到64比特数的变换，具体为：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 $L_{64}(x) = y_0||y_1$ 。

1.4.5.3. 线性变换 L_{128}

L_{128} 是一个128比特数到128比特数的变换，具体为：将128比特数 x 分为2个64比特数，即 $x = x_0||x_1$ ，令

$$y_0 = L_{64}(x_0), \quad y_1 = L_{64}(x_1)$$

则 $L_{128}(x) = y_0||y_1$ 。

1.5. 密钥扩展

密钥扩展算法根据密钥 Key ，生成 N 个 $BL/2$ 比特加密子密钥 EK_i 和 N 个 $BL/2$ 比特解密子密钥 DK_i ， $0 \leq i \leq N-1$ 。

常数 C 为自然对数的底 e 的小数部分的前 128 比特，用 16 进制表示为

$$C = 0xb7e15162, 0x8aed2a6a, 0xbf715880, 0x9cf4f3c7$$

1.5.1. 变换 α 、 β 、 γ

变换 α 、 β 、 γ 对于 SMBA-128-128 和 SMBA-128-256 使用 α_{64} 、 β_{64} 、 γ_{64} ，对于 SMBA-256-256 使用 α_{128} 、 β_{128} 、 γ_{128} 。

1.5.1.1. 变换 α_{64} 、 β_{64} 、 γ_{64}

α_{64} 、 β_{64} 、 γ_{64} 都是 64 比特到 64 比特的变换，分别如下。

(1) 把 64 比特数 x 分为 8 个 8 比特数 x_i , $0 \leq i \leq 7$, 即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7$$

令 $y_i = X_{\alpha_{64}}(x_i)$, 则

$$\alpha_{64}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7$$

其中 $\alpha_{64}(i)$, $0 \leq i \leq 7$, 依次为: 2,7,1,4,6,3,5,0。

(2) $\beta_{64}(x) = x \ll 16$

(3) 把 64 比特数 x 分为 8 个 8 比特数 x_i , $0 \leq i \leq 7$, 即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7$$

令 $y_i = S_{i \bmod 2}(x_i)$, 则

$$\gamma_{64}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7$$

1.5.1.2. 变换 α_{128} 、 β_{128} 、 γ_{128}

α_{128} 、 β_{128} 、 γ_{128} 都是 128 比特到 128 比特的变换，分别如下。

(1) 把 128 比特数 x 分为 16 个 8 比特数 x_i , $0 \leq i \leq 15$, 即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7 || x_8 || x_9 || x_{10} || x_{11} || x_{12} || x_{13} || x_{14} || x_{15}$$

令 $y_i = X_{\alpha_{128}}(x_i)$, 则

$$\alpha_{128}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7 || y_8 || y_9 || y_{10} || y_{11} || y_{12} || y_{13} || y_{14} || y_{15}$$

其中 $\alpha_{128}(i)$, $0 \leq i \leq 15$, 依次为: 13,4,12,0,8,10,2,6,14,3,11,15,7,9,1,5。

(2) $\beta_{128}(x) = x \ll 32$

(3) 把 128 比特数 x 分为 16 个 8 比特数 x_i , $0 \leq i \leq 15$, 即

$$x = x_0 || x_1 || x_2 || x_3 || x_4 || x_5 || x_6 || x_7 || x_8 || x_9 || x_{10} || x_{11} || x_{12} || x_{13} || x_{14} || x_{15}$$

令 $y_i = S_{i \bmod 2}(x_i)$, 则

$$\gamma_{128}(x) = y_0 || y_1 || y_2 || y_3 || y_4 || y_5 || y_6 || y_7 || y_8 || y_9 || y_{10} || y_{11} || y_{12} || y_{13} || y_{14} || y_{15}。$$

1.5.2. SMBA-128-128 的子密钥生成

加密子密钥和解密子密钥按如下方式生成：

- (1) $U_0 = \text{MSB}_{64}(\text{Key})$, $V_0 = \text{LSB}_{64}(\text{Key})$;
- (2) $D = N || \text{BlockLen} || \text{KeyLen} || 0^{40}$, 这里 $\text{BlockLen} = \text{BL}/8$ 为分组长度（字节数）， $\text{KeyLen} = \text{KL}/8$ 为密钥长度（字节数）， N 、 BlockLen 、 KeyLen 都用 8 比特表示。
- (3) $C_0 = \text{MSB}_{64}(C) \oplus D$;
- (4) 进行 N 个密钥扩展轮变换（参见图 1.8-5），即

对于 $i=0$ 到 $N-1$

{

$$X_i = U_i \oplus C_i$$

$$Y_i = \alpha_{64}(X_i)$$

$$Z_i = \gamma_{64}(Y_i)$$

$$EK_i = DK_{N-1-i} = Z_i$$

$$U_{i+1} = Z_i \oplus V_i$$

$$V_{i+1} = \beta_{64}(U_i)$$

$$C_{i+1} = C_i \lll 23$$

}

1.5.3. SMBA-128-256 的子密钥生成

加密子密钥和解密子密钥按如下方式生成：

- (1) $U_{0,0} || U_{0,1} || U_{0,2} || U_{0,3} = \text{Key}$, 其中 $U_{0,i}$ 的长度都是 64 比特, $0 \leq i \leq 3$;
- (2) $D = N || \text{BlockLen} || \text{KeyLen} || 0^{40}$, 这里 $\text{BlockLen} = \text{BL}/8$ 为分组长度（字节数）， $\text{KeyLen} = \text{KL}/8$ 为密钥长度（字节数）， N 、 BlockLen 、 KeyLen 都用 8 比特表示。
- (3) $C_0 = \text{MSB}_{64}(C) \oplus D$;
- (4) 进行 N 个密钥扩展轮变换（参见图 1.8-6），即

对于 $i=0$ 到 $N-1$

{

$$X_i = U_{i,0} \oplus C_i$$

$$Y_i = \alpha_{64}(X_i)$$

$$Z_i = \gamma_{64}(Y_i)$$

$$EK_i = DK_{N-1-i} = Z_i$$

$$U_{i+1,0} = Z_i \oplus U_{i,1}$$

$$U_{i+1,1} = U_{i,2}$$

$$U_{i+1,2} = U_{i,3}$$

$$U_{i+1,3} = \beta_{64}(U_{i,0})$$

$$C_{i+1} = C_i \lll 23$$

}

1.5.4. SMBA-256-256 的子密钥生成

加密子密钥和解密子密钥按如下方式生成：

- (1) $U_0 = \text{MSB}_{128}(\text{Key})$, $V_0 = \text{LSB}_{128}(\text{Key})$;
- (2) $D = N \parallel \text{BlockLen} \parallel \text{KeyLen} \parallel 0^{104}$, 这里 $\text{BlockLen} = \text{BL}/8$ 为分组长度（字节数）， $\text{KeyLen} = \text{KL}/8$ 为密钥长度（字节数）， N 、 BlockLen 、 KeyLen 都用 8 比特表示。
- (3) $C_0 = C \oplus D$;
- (4) 进行 N 个密钥扩展轮变换（参见图 1.8-5），即

对于 $i=0$ 到 $N-1$

{

$$X_i = U_i \oplus C_i$$

$$Y_i = \alpha_{128}(X_i)$$

$$Z_i = \gamma_{128}(Y_i)$$

$$EK_i = DK_{N-1-i} = Z_i$$

$$U_{i+1} = Z_i \oplus V_i$$

$$V_{i+1}=\beta_{128}(U_i)$$

$$C_{i+1}=C_i<<<57$$

}

1.6. S 盒 S₀ 和 S₁

S 盒 S₀ 和 S₁ 的构造过程见“1.9 S 盒 S₀、S₁ 构造过程”。用 16 进制表示的代替盒 S₀ 和 S₁ 分别如下：

S₀:

0x18,0x32,0x6b,0x69,0x9e,0xcb,0x0b,0x2f,0x03,0x94,0x9f,0x21,0xc8,0x86,0x26,0xf4,
0x46,0x67,0x0d,0x6d,0x39,0xc0,0x09,0x29,0xa1,0xda,0x82,0xf3,0x78,0x6a,0xee,0x3b,
0x84,0x87,0xd1,0xf8,0x70,0xb9,0xa3,0x3f,0xfe,0x3d,0x37,0xf7,0xfb,0x48,0xc6,0xbc,
0x65,0x4e,0x6e,0x64,0xaf,0x91,0xdb,0x1c,0x71,0x8e,0x59,0x8d,0x01,0x74,0xe3,0xa8,
0xb4,0xbf,0x4b,0x2d,0xea,0x95,0x6f,0x2b,0x5f,0xb3,0x98,0x57,0xfd,0x24,0x85,0xe9,
0x5d,0x6c,0x4d,0x4f,0x13,0xe1,0x49,0x0f,0xdc,0x36,0x55,0x3c,0xef,0x11,0x50,0x4a,
0x73,0x5b,0x8a,0xa9,0x92,0x58,0x35,0x80,0x38,0x04,0x06,0x12,0xb7,0xc7,0x68,0xd2,
0x45,0x66,0xcd,0x47,0xac,0xba,0xf2,0x16,0x4c,0xb2,0x5c,0x99,0xc2,0x8b,0x23,0x7c,
0xe5,0x9b,0xab,0x42,0xad,0x05,0x28,0xce,0xca,0x5e,0x79,0x2e,0xf5,0x72,0x43,0x77,
0x96,0x33,0x7a,0xa4,0x40,0xde,0x75,0xf9,0x89,0xd9,0xb5,0x1a,0x9a,0xb6,0xd6,0x8f,
0xb0,0xe4,0x62,0x88,0x7e,0x83,0x15,0x3a,0x2c,0x51,0x56,0xeb,0x90,0xf1,0x1b,0x00,
0x9d,0x19,0x52,0xa7,0x34,0x53,0x97,0x25,0xbe,0x30,0xaa,0xf0,0xa2,0x02,0x0c,0xe7,
0xd3,0x08,0xc1,0xf6,0xa0,0x76,0x10,0x1f,0x61,0xa5,0x14,0xec,0x22,0x31,0xd8,0xbb,
0xc5,0xe8,0x54,0x93,0x3e,0x7b,0x9c,0x27,0xdd,0x1d,0xcc,0xe2,0x0e,0xe6,0x81,0x20,
0x2a,0xfa,0xff,0xe0,0x07,0xae,0xcf,0x0a,0xed,0x1e,0xa6,0x41,0x7f,0x63,0x5a,0xfc,
0xd5,0xc9,0xd4,0xb8,0x60,0xd7,0x7d,0xd0,0xc4,0xc3,0x44,0x17,0xdf,0x8c,0xb1,0xbd.

S₁:

0x6a,0x05,0xe8,0x16,0x7f,0x37,0x68,0x54,0xbe,0x01,0xc1,0x87,0xbf,0xb3,0xd1,0x44,
0xf8,0xf9,0xd4,0xf5,0x1f,0x77,0xa5,0x3b,0x04,0x92,0x62,0xd7,0xc6,0xa7,0x0c,0xb1,
0xbd,0x5c,0xff,0xf4,0x17,0x3d,0xa3,0x32,0x9e,0x09,0x8c,0x63,0x66,0x10,0xe9,0x47,
0xa8,0xfe,0x3c,0xc0,0x7d,0x1d,0xfd,0x4d,0x4c,0xd8,0xd9,0x9c,0x34,0xd0,0x25,0xeb,
0x91,0x61,0xe0,0x1b,0x95,0x98,0xe5,0xba,0xe3,0x64,0xad,0x0f,0x73,0x19,0x39,0xbb,

0x00,0xc7,0xf2,0x3a,0x43,0x79,0x8d,0xf0,0x78,0x23,0xc4,0xa0,0x0b,0x03,0xb4,0x52,
0x69,0xf1,0xf6,0xc9,0x0d,0xb6,0x8a,0xa9,0x27,0xf3,0x86,0x88,0xea,0xc2,0x29,0x14,
0x8f,0x22,0x9b,0xc7,0x26,0xed,0xa2,0xab,0xd5,0xaa,0xc5,0x6d,0x60,0x42,0x65,0xa6,
0x99,0xb2,0xac,0x81,0x3f,0x5d,0xa1,0x72,0x74,0x30,0xc3,0xef,0x9a,0x1e,0x7b,0x2a,
0x67,0x48,0x24,0x31,0x35,0x15,0xf7,0x2f,0x89,0x20,0x40,0xdd,0x50,0x2e,0x08,0x11,
0xde,0x80,0x5b,0x6c,0x5f,0x75,0x4a,0x7e,0x5e,0x70,0xce,0x6b,0xcf,0x84,0xda,0xec,
0x76,0xb7,0x83,0xee,0x55,0x57,0xcd,0x51,0x8b,0x02,0xb9,0xbc,0xd3,0xb5,0xae,0x90,
0x46,0xe2,0x71,0xc8,0x6e,0x85,0x8e,0x36,0x38,0x2b,0x18,0x0a,0xb0,0xfa,0x3e,0x21,
0x59,0x7c,0x7a,0xb8,0x6f,0x96,0xcc,0x1c,0xe1,0x2c,0x06,0xfc,0x97,0x9d,0xe4,0x07,
0xca,0x0e,0xcb,0xd6,0x4b,0x13,0x4f,0x49,0xdf,0x82,0x4e,0xdc,0x5a,0x1a,0xdb,0x53,
0x56,0x33,0xaf,0xfb,0x9f,0xd2,0x2d,0x41,0x45,0x93,0x58,0x28,0xe6,0xa4,0x12,0x94.

1.7. 样本数据

1.7.1. SMBA-128-128 样本数据

第 1 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

明文: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

密文: 0x21,0x9a,0x3b,0x8f,0x61,0x47,0x98,0x64,0x8f,0xc4,0xd6,0xd1,0x9d,0x72,0x86,0x6d,

第 2 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

明文: 0x21,0x9a,0x3b,0x8f,0x61,0x47,0x98,0x64,0x8f,0xc4,0xd6,0xd1,0x9d,0x72,0x86,0x6d,

密文: 0x1e,0xd5,0x77,0x29,0x53,0x7a,0xaf,0x45,0x10,0x67,0x92,0x64,0x1a,0x4f,0x21,0x7e,

第 3 组样本数据:

密钥: 0x21,0x9a,0x3b,0x8f,0x61,0x47,0x98,0x64,0x8f,0xc4,0xd6,0xd1,0x9d,0x72,0x86,0x6d,

明文: 0x1e,0xd5,0x77,0x29,0x53,0x7a,0xaf,0x45,0x10,0x67,0x92,0x64,0x1a,0x4f,0x21,0x7e,

密文: 0x66,0xc5,0xa2,0x73,0x42,0x0c,0x3c,0x23,0x84,0x8a,0xf4,0x2e,0x8f,0xb3,0x9e,0xed,

第 4 组样本数据:

密钥: 0x1e,0xd5,0x77,0x29,0x53,0x7a,0xaf,0x45,0x10,0x67,0x92,0x64,0x1a,0x4f,0x21,0x7e,

明文: 0x66,0xc5,0xa2,0x73,0x42,0x0c,0x3c,0x23,0x84,0x8a,0xf4,0x2e,0x8f,0xb3,0x9e,0xed,

密文: 0x32,0x10,0x80,0xcd,0x29,0x39,0x4e,0x91,0xb0,0x0e,0xfc,0xd5,0x97,0xa4,0xa2,0xa3,

第 5 组样本数据:

密钥: 0x66,0xc5,0xa2,0x73,0x42,0x0c,0x3c,0x23,0x84,0x8a,0xf4,0x2e,0x8f,0xb3,0x9e,0xed,

明文: 0x32,0x10,0x80,0xcd,0x29,0x39,0x4e,0x91,0xb0,0x0e,0xfc,0xd5,0x97,0xa4,0xa2,0xa3,

密文: 0x98,0xc9,0xe0,0x38,0x42,0x8f,0x45,0xe5,0xf3,0xf5,0xc2,0x35,0x9d,0x72,0x7b,0x71,

1.7.2. SMBA-128-256 样本数据

第 1 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

明文: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

密文: 0x40,0x76,0x91,0x2d,0xfc,0x2b,0x9b,0xe2,0x20,0xb9,0x76,0x27,0xe1,0x72,0xf3,0x33,

第 2 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

明文: 0x40,0x76,0x91,0x2d,0xfc,0x2b,0x9b,0xe2,0x20,0xb9,0x76,0x27,0xe1,0x72,0xf3,0x33,

密文: 0x93,0x7b,0x88,0xd4,0x88,0x71,0x18,0x28,0x90,0x4d,0x14,0x6b,0x8b,0x60,0x27,0x37,

第 3 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x40,0x76,0x91,0x2d,0xfc,0x2b,0x9b,0xe2,0x20,0xb9,0x76,0x27,0xe1,0x72,0xf3,0x33,
明文: 0x93,0x7b,0x88,0xd4,0x88,0x71,0x18,0x28,0x90,0x4d,0x14,0x6b,0x8b,0x60,0x27,0x37,
密文: 0x47,0xc7,0x8d,0x6e,0xad,0xd6,0xb6,0x19,0x2b,0x7c,0x01,0xf0,0xe6,0x72,0x05,0x40,

第 4 组样本数据:

密钥: 0x40,0x76,0x91,0x2d,0xfc,0x2b,0x9b,0xe2,0x20,0xb9,0x76,0x27,0xe1,0x72,0xf3,0x33,
0x93,0x7b,0x88,0xd4,0x88,0x71,0x18,0x28,0x90,0x4d,0x14,0x6b,0x8b,0x60,0x27,0x37,
明文: 0x47,0xc7,0x8d,0x6e,0xad,0xd6,0xb6,0x19,0x2b,0x7c,0x01,0xf0,0xe6,0x72,0x05,0x40,
密文: 0xef,0xf3,0xb8,0x2f,0x5a,0xf5,0x4d,0x60,0xb0,0xb5,0xca,0x67,0xae,0x8a,0xb6,0x14,

第 5 组样本数据:

密钥: 0x93,0x7b,0x88,0xd4,0x88,0x71,0x18,0x28,0x90,0x4d,0x14,0x6b,0x8b,0x60,0x27,0x37,
0x47,0xc7,0x8d,0x6e,0xad,0xd6,0xb6,0x19,0x2b,0x7c,0x01,0xf0,0xe6,0x72,0x05,0x40,
明文: 0xef,0xf3,0xb8,0x2f,0x5a,0xf5,0x4d,0x60,0xb0,0xb5,0xca,0x67,0xae,0x8a,0xb6,0x14,
密文: 0x39,0xfd,0x4f,0x48,0x34,0x3b,0xda,0xd7,0x3e,0xc4,0x01,0xf1,0xde,0x55,0x2a,0x01,

1.7.3. SMBA-256-256 样本数据

第 1 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
明文: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
密文: 0x1b,0x8b,0x33,0x77,0x1d,0x34,0xa3,0x64,0xd2,0x89,0x20,0x99,0xb3,0xd9,0x0b,0x1f,
0x43,0x62,0x50,0xea,0x73,0x6c,0x45,0x30,0xa3,0xd3,0xf8,0xa4,0xab,0x1c,0xb7,0x59,

第 2 组样本数据:

密钥: 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
明文: 0x1b,0x8b,0x33,0x77,0x1d,0x34,0xa3,0x64,0xd2,0x89,0x20,0x99,0xb3,0xd9,0x0b,0x1f,
0x43,0x62,0x50,0xea,0x73,0x6c,0x45,0x30,0xa3,0xd3,0xf8,0xa4,0xab,0x1c,0xb7,0x59,
密文: 0xbb,0x46,0x09,0x4e,0x8d,0xd7,0x12,0xb3,0x65,0xf0,0x29,0x36,0xd0,0x27,0x77,0xea,
0xa0,0xf8,0xa0,0xab,0xe9,0x26,0x98,0x82,0x5b,0xa2,0x41,0xa7,0xe5,0x98,0x09,0x04,

第 3 组样本数据:

密钥: 0x1b,0x8b,0x33,0x77,0x1d,0x34,0xa3,0x64,0xd2,0x89,0x20,0x99,0xb3,0xd9,0x0b,0x1f,
0x43,0x62,0x50,0xea,0x73,0x6c,0x45,0x30,0xa3,0xd3,0xf8,0xa4,0xab,0x1c,0xb7,0x59,
明文: 0xbb,0x46,0x09,0x4e,0x8d,0xd7,0x12,0xb3,0x65,0xf0,0x29,0x36,0xd0,0x27,0x77,0xea,
0xa0,0xf8,0xa0,0xab,0xe9,0x26,0x98,0x82,0x5b,0xa2,0x41,0xa7,0xe5,0x98,0x09,0x04,
密文: 0xac,0xca,0x96,0x73,0xff,0xa5,0xea,0x6a,0xbf,0x4c,0x32,0xc7,0x37,0x04,0x49,0xa0,
0x39,0x02,0x48,0xac,0xc1,0x21,0x61,0x00,0x9a,0x57,0xca,0x24,0xb9,0x49,0xa2,0xe3,

第 4 组样本数据:

密钥: 0xbb,0x46,0x09,0x4e,0x8d,0xd7,0x12,0xb3,0x65,0xf0,0x29,0x36,0xd0,0x27,0x77,0xea,
0xa0,0xf8,0xa0,0xab,0xe9,0x26,0x98,0x82,0x5b,0xa2,0x41,0xa7,0xe5,0x98,0x09,0x04,
明文: 0xac,0xca,0x96,0x73,0xff,0xa5,0xea,0x6a,0xbf,0x4c,0x32,0xc7,0x37,0x04,0x49,0xa0,
0x39,0x02,0x48,0xac,0xc1,0x21,0x61,0x00,0x9a,0x57,0xca,0x24,0xb9,0x49,0xa2,0xe3,
密文: 0x9c,0x36,0x26,0x31,0x05,0x93,0x38,0x27,0xa3,0x07,0x9a,0x69,0x29,0xee,0x13,0x2f,
0x49,0xbb,0xf3,0x5e,0x47,0x52,0x35,0xb3,0x3b,0x94,0x31,0x8d,0x7f,0xcd,0x3d,0xc6,

第 5 组样本数据:

密钥: 0xac,0xca,0x96,0x73,0xff,0xa5,0xea,0x6a,0xbf,0x4c,0x32,0xc7,0x37,0x04,0x49,0xa0,
0x39,0x02,0x48,0xac,0xc1,0x21,0x61,0x00,0x9a,0x57,0xca,0x24,0xb9,0x49,0xa2,0xe3,
明文: 0x9c,0x36,0x26,0x31,0x05,0x93,0x38,0x27,0xa3,0x07,0x9a,0x69,0x29,0xee,0x13,0x2f,
0x49,0xbb,0xf3,0x5e,0x47,0x52,0x35,0xb3,0x3b,0x94,0x31,0x8d,0x7f,0xcd,0x3d,0xc6,
密文: 0x4c,0x1c,0x1f,0x69,0x38,0x2e,0x71,0x37,0x5e,0xe0,0xb1,0x03,0x37,0xae,0x4d,0x09,

0xf9,0xbb,0xc4,0x15,0x0e,0x05,0x78,0x50,0x4e,0x83,0x79,0xee,0xcd,0x5a,0x11,0x6b,

1.8. 密码算法设计框图

1.8.1. 加密过程图

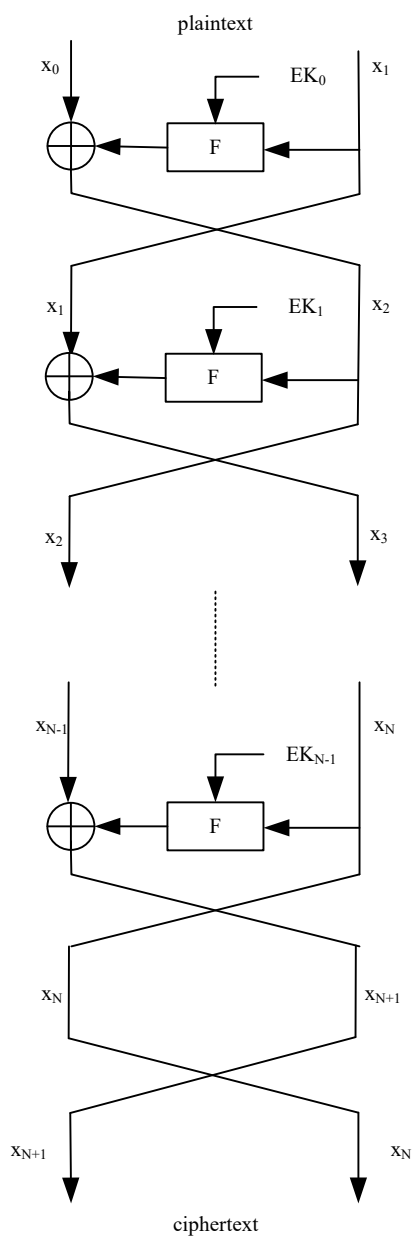


图 1.8-1 加密过程图

1.8.2. 解密过程图

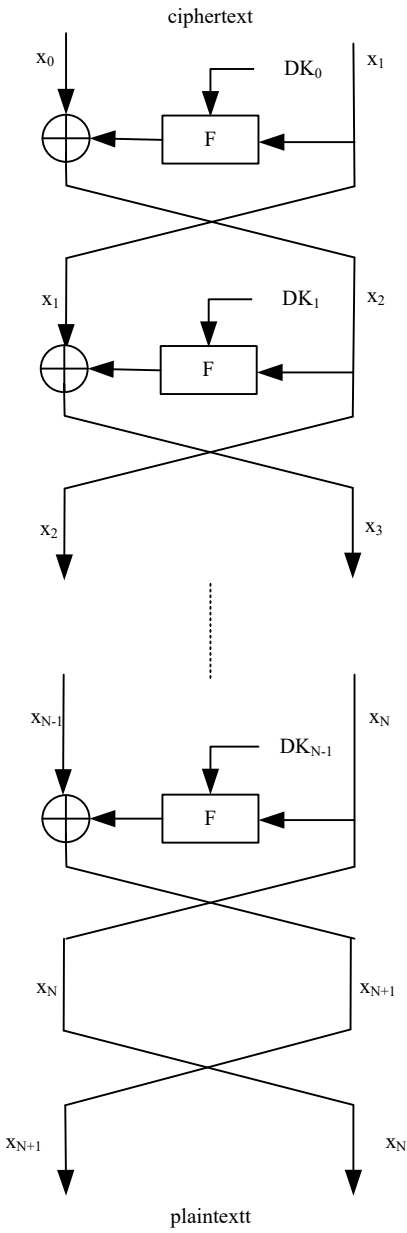


图 1.8-2 解密过程图

1.8.3. SMBA-128 轮函数图

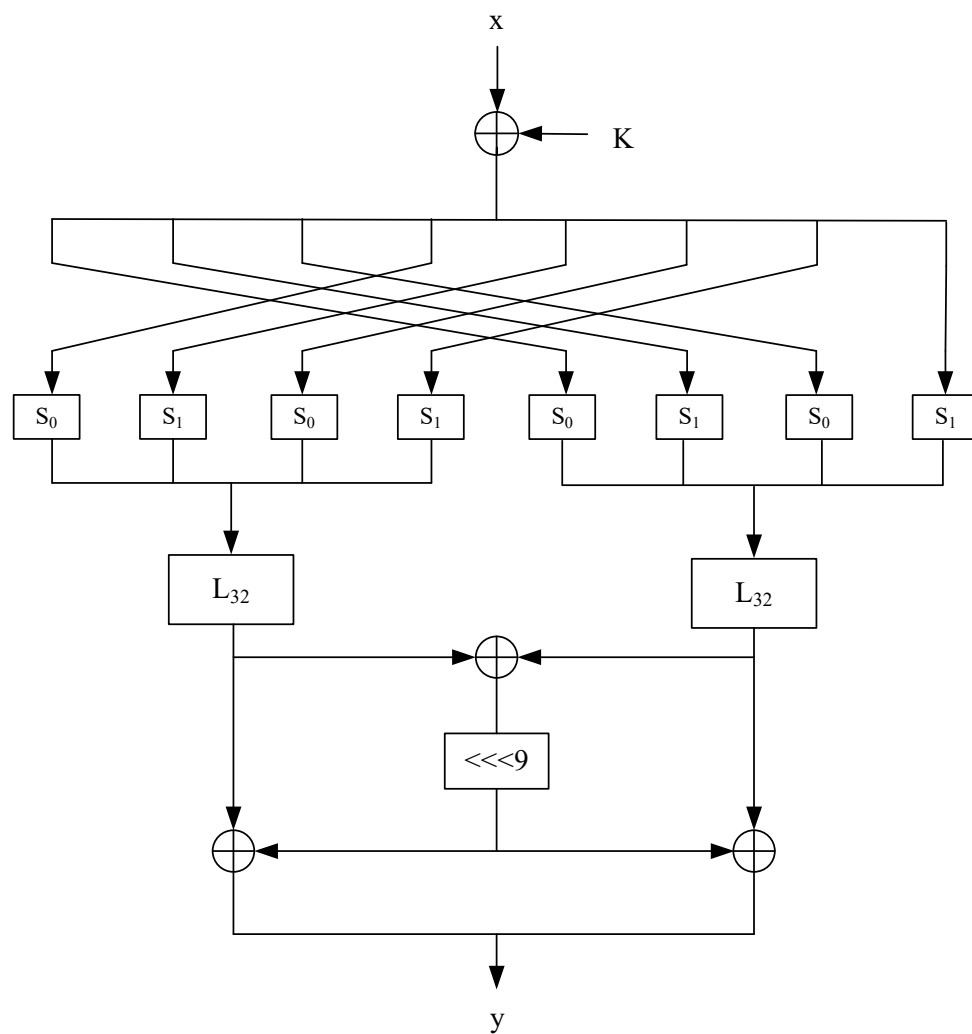


图 1.8-3 SMBA-128 轮函数图

1.8.4. SMBA-256 轮函数图

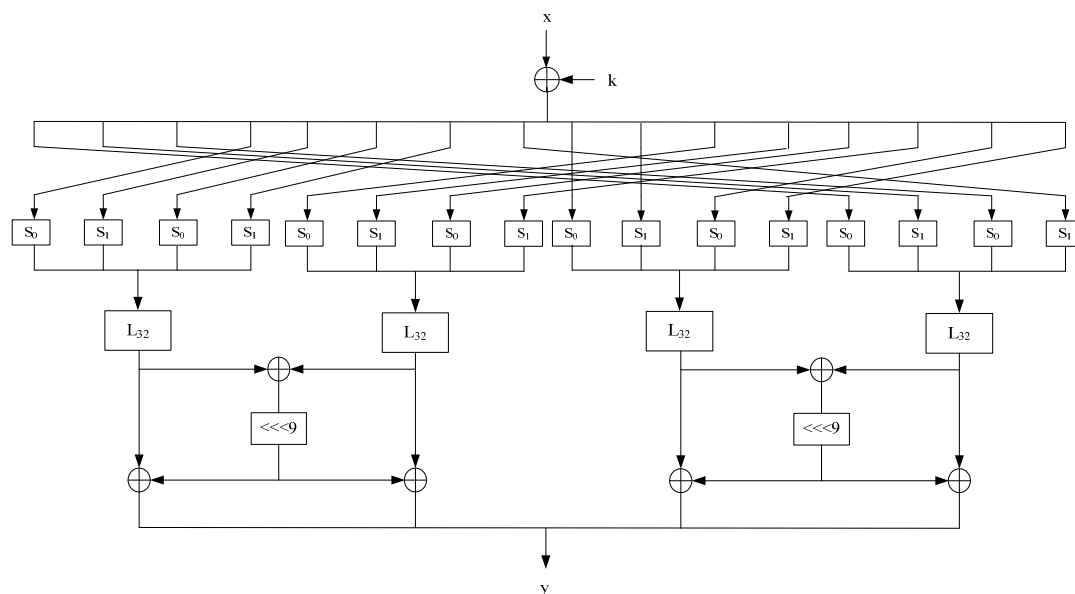


图 1.8-4 SMBA-256 轮函数图

1.8.5. SMBA-128-128 和 SMBA-256-256 密钥扩展轮变换图

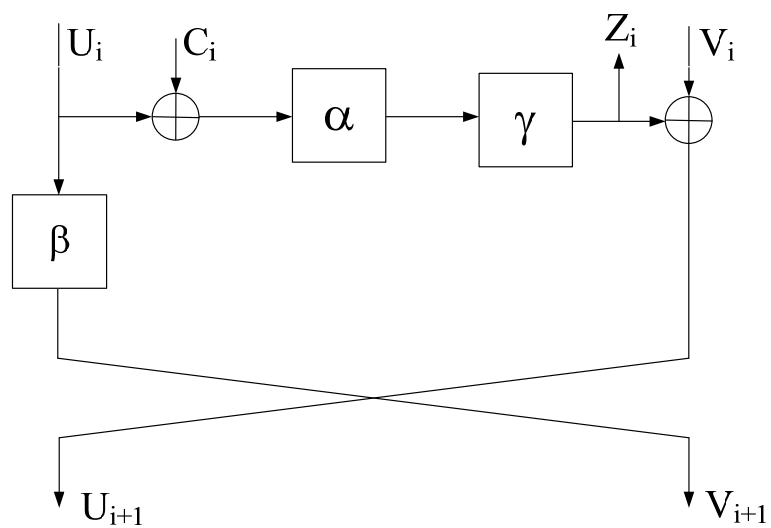


图 1.8-5 SMBA-128-128 和 SMBA-256-256 密钥扩展轮变换图

1.8.6. SMBA-128-256 密钥扩展轮变换图

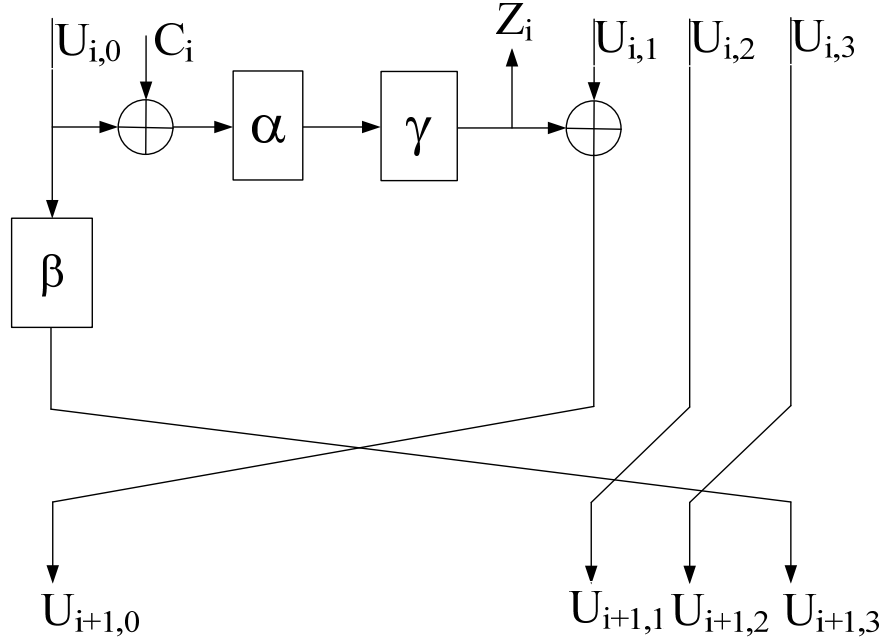


图 1.8-6 SMBA-128-256 密钥扩展轮变换图

1.9. S 盒 S_0 、 S_1 构造过程

1.9.1. S_0 构造过程

1.9.1.1. 有限域多项式基上的计算

设 Z 是 $GF(2)$ 中的不可约多项式 z^2+z+1 的根，则 $(Z,1)$ 是 $GF(2^2)$ 在 $GF(2)$ 上的一组多项式基。

设 Y 是 $GF(2^2)$ 中的不可约多项式 y^2+y+Z (因为 $\text{tr}_{4/2}(Z)=Z+Z^2=1$) 的根，则 $(Y,1)$ 是 $GF(2^4)$ 在 $GF(2^2)$ 上的一组多项式基， $(ZY,Y,Z,1)$ 是 $GF(2^4)$ 在 $GF(2)$ 上的一组基。

有 $z^2+z+1=(z+Z)(z+Z^2)$ ， $y^2+y+Z=(y+Y)(y+Y^4)$ 。

$GF(2^2)$ 在基 $(Z,1)$ 上的乘法：

$$\begin{aligned} (b_0Z+b_1)(c_0Z+c_1) &= (b_0c_0+b_0c_1+b_1c_0)Z+(b_0c_0+b_1c_1) \\ &= ((b_0+b_1)(c_0+c_1)+b_1c_1)Z+(b_0c_0+b_1c_1) \end{aligned}$$

$GF(2^2)$ 在基 $(Z,1)$ 上的乘法：

$$\begin{aligned} Z(b_0Z+b_1)(c_0Z+c_1) &= ((b_0c_1+b_1c_0)+b_1c_1)Z+(b_0c_0+(b_0c_1+b_1c_0)) \\ &= ((b_0+b_1)(c_0+c_1)+b_0c_0)Z+((b_0+b_1)(c_0+c_1)+b_1c_1) \end{aligned}$$

GF(2⁴)在基(Y,1)上的乘法:

$$(b_0Y+b_1)(c_0Y+c_1)=((b_0+b_1)(c_0+c_1)+b_1c_1)Y+(Zb_0c_0+b_1c_1)$$

1.9.1.2. S₀ 构造图示

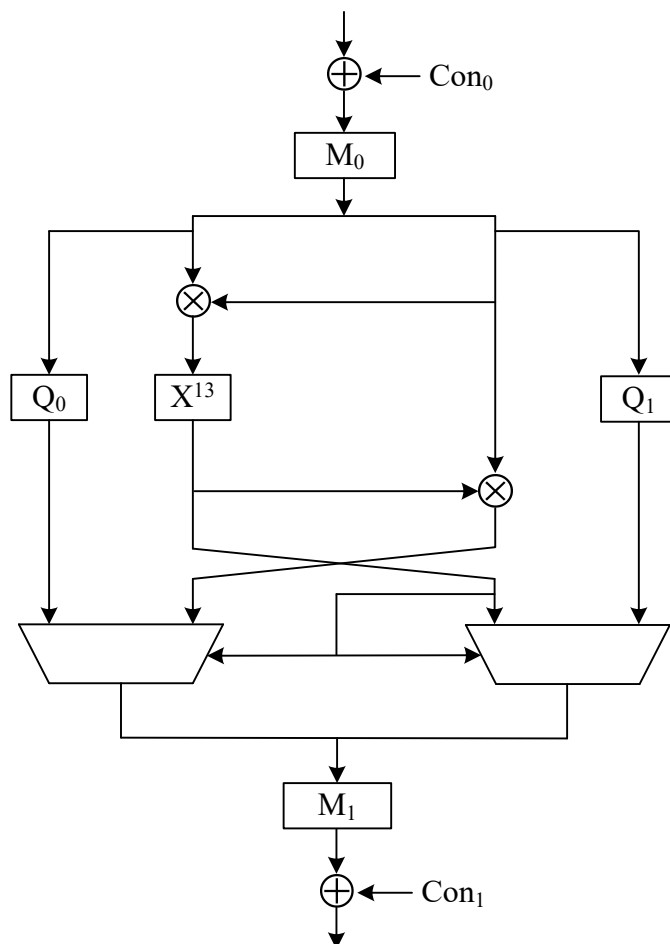


图 1.9-1 S₀ 结构图

1.9.1.3. S₀ 构造过程描述

设 S₀ 的 8 比特输入为 x，则如下计算 S₀ 的 8 比特输出 S₀[x]:

- (1) $a = M_0 * (x \oplus \text{Con}_0)$ ，将 a 分为 2 个 4 比特 a_0 、 a_1 ，即 $a = a_0 || a_1$;
- (2) $b = a_0 \otimes a_1$;
- (3) $c = b^{13}$;
- (4) $d = c \otimes a_1$;
- (5) $e_0 = Q_0[a_0]$, $e_1 = Q_1[a_1]$;
- (6) 若 c 为 0，则 $f = e_0 || e_1$ ；否则 $f = d || c$;

$$(7) \quad S_0[x] = M_1 * (f) \oplus \text{Con}_1$$

其中，符号 \otimes 表示 $GF(2^4)$ 在基 $(ZY, Y, Z, 1)$ 上的乘法， $*$ 表示矩阵与列向量相乘。 b^{13} 表示 $GF(2^4)$ 在基 $(ZY, Y, Z, 1)$ 上 b 的 13 次幂，该变换用 16 进制表示为：

$$0x0, 0x1, 0x2, 0x3, 0x9, 0xb, 0xc, 0xf, 0xe, 0x5, 0xd, 0x4, 0x7, 0x8, 0xa, 0x6.$$

两个 4 比特 S 盒 Q_0 、 Q_1 分别为：

$$0x0, 0x8, 0x4, 0x2, 0xa, 0xd, 0x9, 0x1, 0xb, 0x6, 0xf, 0xe, 0x5, 0xc, 0x3, 0x7.$$

$$0x0, 0x2, 0x8, 0x1, 0x6, 0x9, 0x7, 0x5, 0x4, 0xa, 0xc, 0xd, 0x3, 0xf, 0xe, 0xb.$$

两个矩阵 M_0 和 M_1 都是置换矩阵， M_0 的第 i 行第 $\tau_0(i)$ 列为 1，其余为 0； M_1 的第 i 行第 $\tau_1(i)$ 列为 1，其余为 0； $0 \leq i \leq 7$ 。 τ_0 和 τ_1 分别为：

$$0, 4, 2, 6, 5, 3, 1, 7.$$

$$1, 4, 7, 0, 2, 5, 6, 3.$$

两个 8 比特常数分别为： $\text{Con}_0 = 0x52$ ， $\text{Con}_1 = 0x4d$ 。

上述步骤 (7) 的“若 c 为 0，则 $f = e_0 \| e_1$ ；否则 $f = d \| c$ ；”也可如下实现：

设 $c = c_0 \| c_1 \| c_2 \| c_3$ ，令 $g = c_0 \vee c_1 \vee c_2 \vee c_3$ ， $h = g \| g \| g \| g \| g \| g \| g \| g$ ， $f = ((e_0 \| e_1) \oplus (d \| c)) \& h \oplus (e_0 \| e_1)$ 或者 $f = (e_0 \| e_1) \& \bar{h} \vee (d \| c) \& h$ ；

1.9.2. S_1 构造过程

1.9.2.1. 有限域正规基上的计算

当 $q = 2^n$ 时， (α, α^q) 是 $GF(q^2)$ 在 $GF(q)$ 上的一组正规基，当且仅当 $\text{tr}_{q^2/q}(\alpha) = \alpha + \alpha^q \neq 0$ 。

设 Z 是 $GF(2)$ 中的不可约多项式 $z^2 + z + 1$ 的根，则 (Z^2, Z) 是 $GF(2^2)$ 在 $GF(2)$ 上的一组正规基。

设 Y 是 $GF(2^2)$ 中的不可约多项式 $y^2 + y + Z$ （因为 $\text{tr}_{4/2}(Z) = 1$ ）的根，则 (Y^4, Y) 是 $GF(2^4)$ 在 $GF(2^2)$ 上的一组正规基， $(Z^2 Y^4, ZY^4, Z^2 Y, ZY)$ 是 $GF(2^4)$ 在 $GF(2)$ 上的一组基。

设 X 是 $GF(2^4)$ 中的不可约多项式 $x^2 + x + Z^2 Y^4$ （因为 $\text{tr}_{16/4}(Z^2 Y^4) = 1$ ）的根，则 (X^{16}, X) 是 $GF(2^8)$ 在 $GF(2^4)$ 上的一组正规基。

有 $z^2 + z + 1 = (z + Z)(z + Z^2)$ ， $y^2 + y + Z = (y + Y)(y + Y^4)$ ， $x^2 + x + Z^2 Y^4 = (x + X)(x + X^{16})$ 。约定 $0^{-1} = 0$ 。

GF(2²)在基(Z²,Z)上的乘法:

$$(b_0Z^2+b_1Z)(c_0Z^2+c_1Z)=((b_0+b_1)(c_0+c_1)+b_0c_0)Z^2+((b_0+b_1)(c_0+c_1)+b_1c_1)Z$$

GF(2²)在基(Z²,Z)上的乘法:

$$Z(b_0Z^2+b_1Z)(c_0Z^2+c_1Z)=(b_0c_0+b_1c_1)Z^2+((b_0+b_1)(c_0+c_1)+b_0c_0)Z$$

GF(2⁴)在基(Y⁴,Y)上的乘法:

$$(b_0Y^4+b_1Y)(c_0Y^4+c_1Y)=(Z(b_0+b_1)(c_0+c_1)+b_0c_0)Y^4+(Z(b_0+b_1)(c_0+c_1)+b_1c_1)Y$$

GF(2⁴)在基(Y⁴,Y)上的乘法:

$$\begin{aligned} Z^2Y^4(b_0Z^2Y^4+b_1ZY^4+b_2Z^2Y+b_3ZY)^2= \\ b_0Z^2Y^4+(b_0+b_1)ZY^4+(b_1+b_3)Z^2Y+(b_0+b_2)ZY \end{aligned}$$

GF(2⁸)在基(X¹⁶,X)上的求逆:

$$(b_0X^{16}+b_1X)^{-1}=(b_0b_1+Z^2Y^4(b_0+b_1)^2)^{-1}(b_1X^{16}+b_0X)$$

1.9.2.2. S₁ 构造图示

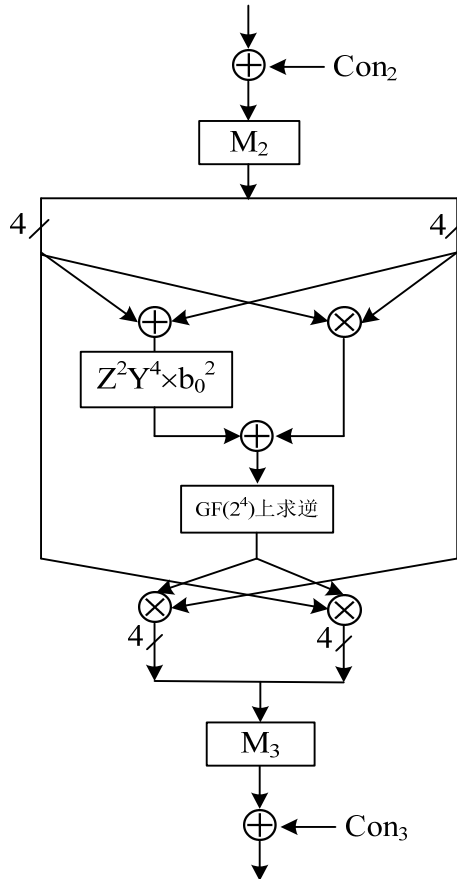


图 1.9-2 S₁ 结构图

1.9.2.3. S₁ 构造过程描述

设 S₁ 的 8 比特输入为 x，则如下计算 S₁ 的 8 比特输出 S₁[x]：

- (1) $a = M_2 * (x \oplus \text{Con}_2)$ ，将 a 分为 2 个 4 比特 a₀、a₁，即 $a = a_0 || a_1$ ；
- (2) $b_0 = a_0 \oplus a_1$ ， $b_1 = a_0 \otimes a_1$ ；
- (3) $c = Z^2 Y^4 \otimes b_0^2$ ；
- (4) $d = c \oplus b_1$ ；
- (5) $e = d^{-1}$ ；
- (6) $f_0 = e \otimes a_1$ ， $f_1 = e \otimes a_0$ ；
- (7) $S_1[x] = M_3 * (f_0 || f_1) \oplus \text{Con}_3$

其中，符号 \otimes 表示 GF(2⁴) 在基 (Z²Y⁴, ZY⁴, Z²Y, ZY) 上的乘法。d⁻¹ 表示 GF(2⁴) 在基 (Z²Y⁴, ZY⁴, Z²Y, ZY) 上 d 的逆，该变换用 16 进制表示为：

0x0,0x4,0xc,0x8,0x1,0xa,0xe,0xd,0x3,0xb,0x5,0x9,0x2,0x7,0x6,0xf.

两个矩阵 M₂ 和 M₃ 都是置换矩阵，M₂ 的第 i 行第 $\tau_2(i)$ 列为 1，其余为 0；M₃ 的第 i 行第 $\tau_3(i)$ 列为 1，其余为 0；0 ≤ i ≤ 7。τ₂ 和 τ₃ 分别为：

4,6,5,1,0,7,2,3.

5,1,3,6,2,4,0,7.

两个 8 比特常数分别为：Con₂=0xa4，Con₃=0x5f。

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：

整体结构采用标准Feistel结构，其优点在于：

- (1) 加解密过程除子密钥的顺序外完全相同，降低软硬件的实现代价。
- (2) 与SPN结构相比，其使用的硬件资源更小。更易做资源与效率的折中。

轮函数使用SP结构，其主要考虑如下：

- (1) 安全性和有效性的综合性能更高。
- (2) 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，易于硬件实现以及软件的查大表快速实现。

通过整体结构以及轮函数的设计，SMBA算法具备软硬件多种平台上实现的有效性和灵活性。SMBA-128与SMBA-256共用主要的密码部件，一起实现所付出的额外代价低。同时，算法可以准确评估抵抗差分/线性密码分析、代数攻击和积分攻击等已知攻击所需的轮数，安全界清晰。所选迭代轮数确保算法有较大的安全冗余。在算法设计上不隐藏任何“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。这样可以尽量降低在短消息条件下密钥编排过程对算法性能的影响。
- (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，且与加解密过程相比硬件实现电路深度更小，可以为硬件实现提供更多的灵活性、更好的有效性。

综合以上的考虑，我们设计了SMBA的密钥扩展算法。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。此外密钥扩展与加解密过程共用S盒 S_0 和 S_1 ，在特定场景下可通过复用的方式降低硬件资源。

SMBA的密钥扩展算法保证了算法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），安全界清晰。此外也保证算法不存在等价密钥、弱密钥和半弱密钥等弱点。

2.3. S 盒设计原理

SMBA 使用 8 比特 S 盒，主要的考虑是 8 比特 S 盒逻辑更加复杂，相比较 4 比特 S 盒，其抵抗潜在的攻击能力更强。在非轻量级环境下，一般这样的 S 盒的实现代价是可以接受的。

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。在4比特S盒的选取上，我们尽量采用硬件资源小的S盒，同时通过选取特定的基，使得 $GF(2^4)$ 上的乘法运算可以节省一定的资源，可以追求更小的面积。我们也有追求更小深度的实现方式，更适用于追求更高速度的环境。
- (3) 更低的侧信道攻击防护代价。在4比特S盒的选取上，我们除了实现资源外，更考虑了其中与门的个数。通过采用具有更少与门的S盒，并配以搜索得到的前后仿射变换，我们获得的S盒在抵抗侧信道攻击的指标更优。
- (4) 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

由于64比特的MDS线性变换软硬件实现代价较大，关键路径深，资源占用多，我们希望在密码学性质和实现代价之间取得折中。SMBA线性变换的主要设计思想是将少量简单的线性变换组合起来，获得具有更高分支数、更好密码学性质的大规模线性变换。我们采用了两个简单的 L_{32} 变换，以及一个L-M结构的简单线性变换，成功构造了一个新的变换 L_{64} ，其具有如下的特性：

- (1) L_{64} 、 L_{64}' 和 L_{64}'' 的差分分支数和线性分支数都为6。其中， L_{64}' 和 L_{64}'' 都在“3.3.1差分攻击”中定义。
- (2) L_{64} 、 L_{64}^T 、 L_{64}' 、 $(L_{64}')^T$ 、 L_{64}'' 和 $(L_{64}'')^T$ 都将1个非零字节变换到7个非零字节，上标T表示转置。

(3) L_{64}' 、 $(L_{64}')^T$ 、 L_{64}'' 和 $(L_{64}'')^T$ 都将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、32、64比特）及硬件都可有效实现。

对SMBA-256，换位变换 P_{128} 不仅确保算法的加解密过程都形成一个整体，并且使得算法抗差分、线性等攻击能力显著提升。

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 $y_1 = 0 \Leftrightarrow x_1 = 0\text{ffffff}$;
- (5) 若 $x_0 \neq 0$, $x_1 = 0$, 则 $y_0 = 0 \Leftrightarrow x_0 = 0\text{ffffff}$;

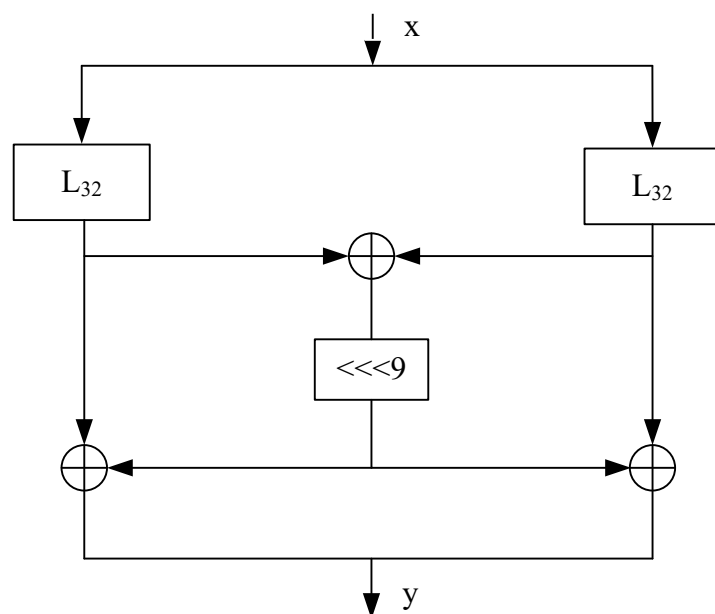


图 3.2.1-1 线性变换 L_{64}

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。

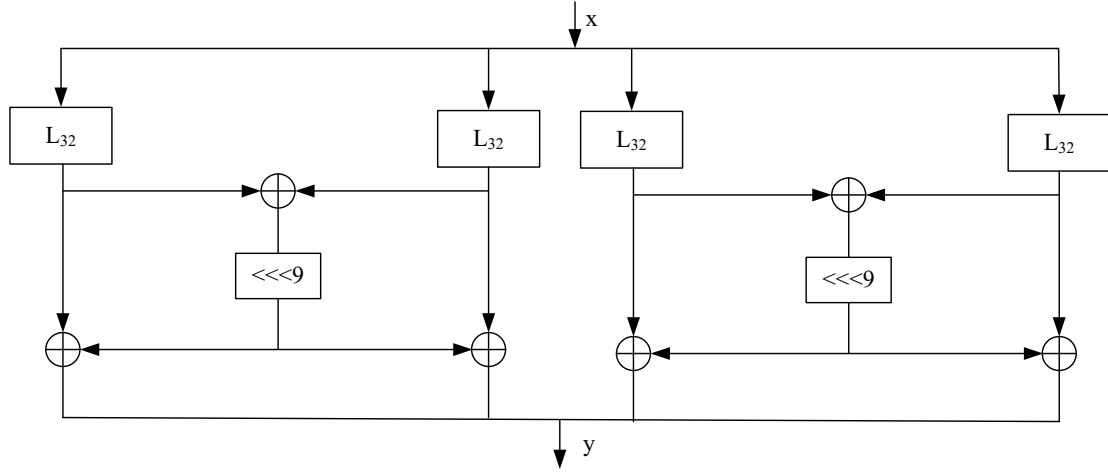


图 3.2.1-2 线性变换 L_{128}

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)

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

其中, $\beta=(\beta_0, \beta_1, \dots, \beta_{m-1})$ 。

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

$$SNR_DPA(S) = n2^{2n} (\sum_{\alpha \in F_2^n} (\sum_{i=0}^{m-1} W_{S_i}(\alpha))^4)^{-\frac{1}{2}}$$

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 |
| 坐标函数代数正规型中非 0 项数 | 121、120、129、119、 125、127、118、117 | 131、116、124、135、 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盒抵抗侧信道攻击的能力越强。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。 L_{64}' 和 L_{64}'' 都满足当输入 x_0 、 x_1 存在一个为0一个非0时，输出 y_0 、 y_1 均非0。

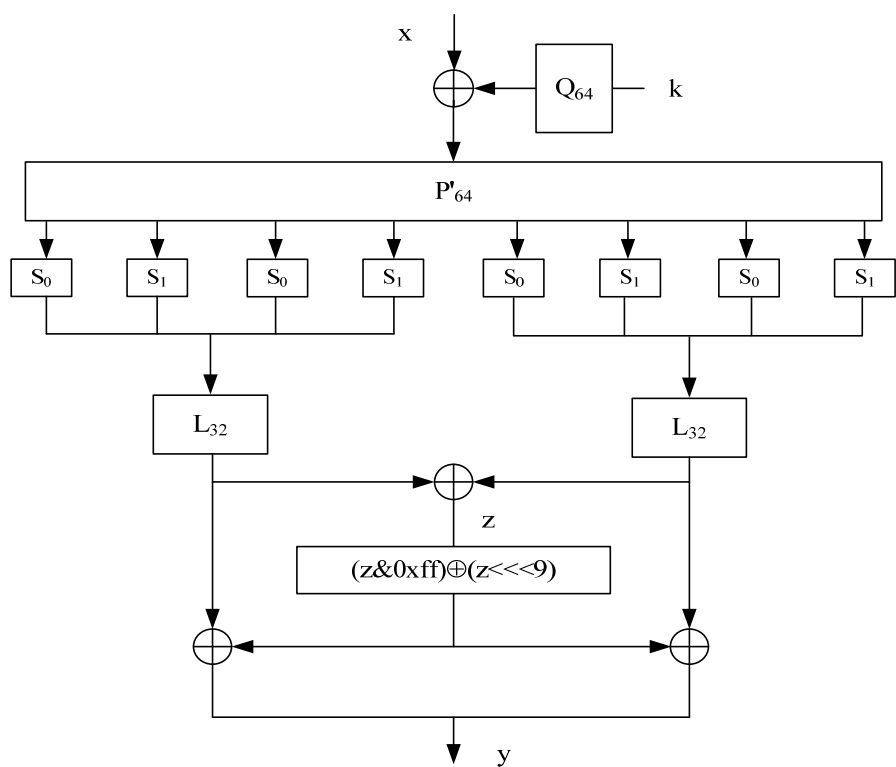


图3.3.1-1 SMBA-128轮函数等价结构

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

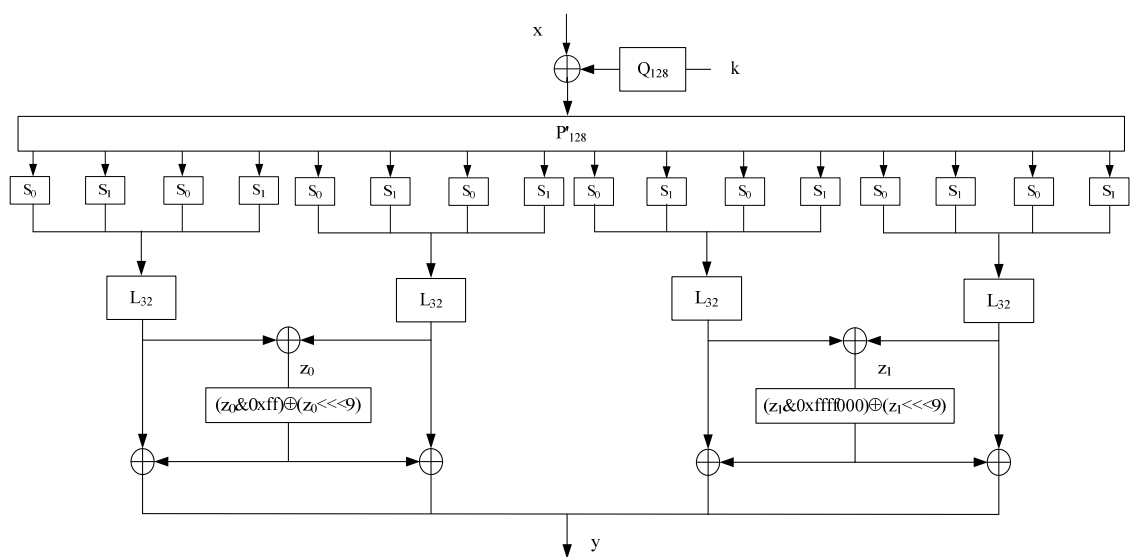


图3.3.1-2 SMBA-256轮函数等价结构

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 |
| 线性特征概率上界 | $2^{-171.392}$ | $2^{-182.104}$ | $2^{-192.816}$ | $2^{-203.528}$ | $2^{-214.240}$ | $2^{-224.952}$ | $2^{-235.664}$ | $2^{-246.376}$ |

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

差分轮数的方法，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，与Feistel结构天然具有的不可能差分一致。

对于SMBA-256算法，找到了17786条8轮不可能差分，没有找到更多轮数的不可能差分，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。

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 轮线性变换的输出截断差分， $r=0,1,\dots,7$ 。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 $(y_{32}\dots y_{47})=(0000\ 1?11\ ?1?1\ 0000)$ 。

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

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

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

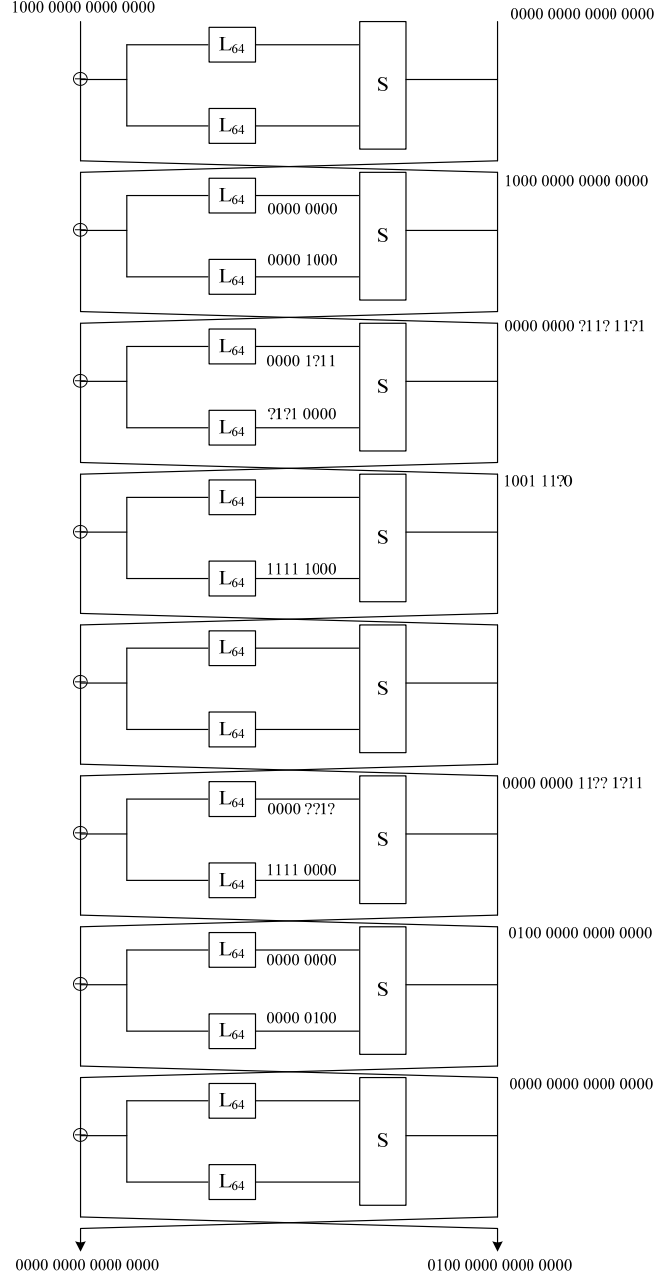


图3.3.5-1 SMBA-256的8轮不可能差分

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 $(x_{64} \dots x_{71}) = (1001\ 11? \ 0)$ 。由第1、2、3轮的迭代关系知： $(x_{32} \dots x_{39}) \oplus L_{64}(y_{32} \dots y_{39}) = (x_{64} \dots x_{71})$ ，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

($z_0 \dots z_7$), 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。

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 $2^{-5.415 \times 54} = 2^{-292.41}$ ，这些结果表明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 | $2^{-156.735}$ | $2^{-201.810}$ | $2^{-251.865}$ | $2^{-304.110}$ | $2^{-364.260}$ | | | |

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

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 $(ek_0, ek_1, \dots, ek_{23})$ 和 $(ek'_0, ek'_1, \dots, ek'_{23})$ 。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 $i \neq j$ ，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)：

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

$$S_0^{-1}(ek_{9,4}) \oplus S_1^{-1}(ek_{10,3}) \oplus ek_{11,4} = C_{9,6} \oplus C_{13,4}$$

$$S_1^{-1}(ek_{9,1}) \oplus S_0^{-1}(ek_{10,6}) \oplus ek_{11,5} = C_{9,7} \oplus C_{13,5}$$

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

$$S_1^{-1}(ek_{9,7}) \oplus S_0^{-1}(ek_{10,4}) \oplus ek_{11,6} = C_{9,0} \oplus C_{13,6}$$

$$S_0^{-1}(ek_{9,2}) \oplus S_1^{-1}(ek_{10,1}) \oplus ek_{11,7} = C_{9,1} \oplus C_{13,7}$$

展开等式(2)，得到等式组(2-1)和(2-2)：

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

$$S_1^{-1}(ek_{10,7}) \oplus S_1^{-1}(ek_{10,3}) \oplus ek_{10,6} \oplus ek_{11,4} = C_{9,6} \oplus C_{10,0} \oplus C_{13,4} \oplus C_{14,6}$$

$$S_0^{-1}(ek_{10,2}) \oplus S_0^{-1}(ek_{10,6}) \oplus ek_{10,7} \oplus ek_{11,5} = C_{9,7} \oplus C_{10,1} \oplus C_{13,5} \oplus C_{14,7}$$

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

$$S_0^{-1}(ek_{10,0}) \oplus S_0^{-1}(ek_{10,4}) \oplus ek_{10,0} \oplus ek_{11,6} = C_{9,0} \oplus C_{10,2} \oplus C_{13,6} \oplus C_{14,0}$$

$$S_1^{-1}(ek_{10,5}) \oplus S_1^{-1}(ek_{10,1}) \oplus ek_{10,1} \oplus ek_{11,7} = C_{9,1} \oplus C_{10,3} \oplus C_{13,7} \oplus C_{14,1}$$

展开等式(3)，得到等式组(3-1)和(3-2)：

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

$$S_1^{-1}(ek_{11,7}) \oplus S_1^{-1}(ek_{11,3}) \oplus ek_{9,6} \oplus ek_{11,4} = C_{8,6} \oplus C_{11,0} \oplus C_{12,4} \oplus C_{15,6}$$

$$S_0^{-1}(ek_{11,2}) \oplus S_0^{-1}(ek_{11,6}) \oplus ek_{9,7} \oplus ek_{11,5} = C_{8,7} \oplus C_{11,1} \oplus C_{12,5} \oplus C_{15,7}$$

$$S_0^{-1}(ek_{11,0}) \oplus S_0^{-1}(ek_{11,4}) \oplus ek_{9,0} \oplus ek_{11,6} = C_{8,0} \oplus C_{11,2} \oplus C_{12,6} \oplus C_{15,0}$$

$$S_1^{-1}(ek_{11,5}) \oplus S_1^{-1}(ek_{11,1}) \oplus ek_{9,1} \oplus ek_{11,7} = C_{8,1} \oplus C_{11,3} \oplus C_{12,7} \oplus C_{15,1} \tag{3-2}$$

$$S_0^{-1}(ek_{11,4}) \oplus S_0^{-1}(ek_{11,0}) \oplus ek_{9,4} \oplus ek_{11,2} = C_{8,4} \oplus C_{11,6} \oplus C_{12,2} \oplus C_{15,4}$$

$$S_1^{-1}(ek_{11,1}) \oplus S_1^{-1}(ek_{11,5}) \oplus ek_{9,5} \oplus ek_{11,3} = C_{8,5} \oplus C_{11,7} \oplus C_{12,3} \oplus C_{15,5}$$

记：

$$ek_{10}^{[2,3,6,7]} = ek_{10,2} || ek_{10,3} || ek_{10,6} || ek_{10,7}, \quad ek_{10}^{[0,1,4,5]} = ek_{10,0} || ek_{10,1} || ek_{10,4} || ek_{10,5},$$

$$ek_{11}^{[2,3,6,7]} = ek_{11,2} || ek_{11,3} || ek_{11,6} || ek_{11,7}, \quad ek_{11}^{[0,1,4,5]} = ek_{11,0} || ek_{11,1} || ek_{11,4} || ek_{11,5},$$

$$ek_{12}^{[2,3,6,7]} = ek_{12,2} || ek_{12,3} || ek_{12,6} || ek_{12,7}, \quad ek_{12}^{[0,1,4,5]} = ek_{12,0} || ek_{12,1} || ek_{12,4} || ek_{12,5},$$

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 $ek_{11}^{[2,3,6,7]}$ ，

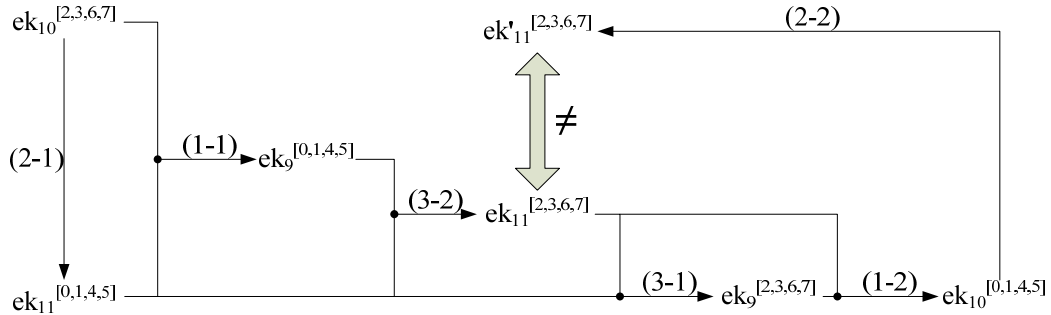


图 3.3.9-1 变量关系推导示意图

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。

Q_0 的乘法复杂度优的实现方案：

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

Q_1 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

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

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 $S \times \left(1 - \alpha - 3\sqrt{\alpha(1-\alpha)/S} \right) \approx 96.01$ 。

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L₃₂都基于8比特设计，特别是L₃₂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，L₃₂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，L₆₄可以按如下方式进行实现， $Y=L_{32}(X)$, $X=(x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7)$, $Y=(y_0,y_1,y_2,y_3,y_4,y_5,y_6,y_7)$:

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

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

$$t_0 = z_0 \oplus z_4$$

$$t_1 = z_1 \oplus z_5$$

$$t_2 = z_2 \oplus z_6$$

$$t_3 = z_3 \oplus z_7$$

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

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

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

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

$$y_0 = z_0 \oplus s_0$$

$$y_1 = z_1 \oplus s_1$$

$$y_2 = z_2 \oplus s_2$$

$$y_3 = z_3 \oplus s_3$$

$$y_4 = z_4 \oplus s_0$$

$$y_5 = z_5 \oplus s_1$$

$$y_6 = z_6 \oplus s_2$$

$$y_7 = z_7 \oplus s_3$$

4.1.2. 32 位 ARM 环境下的算法速率优化 C 实现

4.1.2.1. 平台描述

32比特ARM环境采用基于STM32 F103的开发板(主频72Mhz): stm32f1zet6 核心板6、512K片内Flash、64K片内RAM。

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. 性能自测试结果

采用CBC模式,用256字节数据作为输入进行性能测试,每次测试变换密钥,取 10^5 次CBC模式运算测试结果的平均值作为速率测试的结果。具体结果见表4.1.2.3-1。

表4.1.2.3-1 32位ARM平台软件测试性能

| 算法软件实现类别 | | 加密速率 | 解密速率 |
|--------------------------|--------------|----------|----------|
| 32比特ARM环境下的 算法速率优化C实现 | SMBA-128-128 | 6.84Mbps | 6.71Mbps |
| | SMBA-128-256 | 3.48Mbps | 3.48Mbps |
| | SMBA-256-256 | 3.91Mbps | 3.90Mbps |

4.1.3. 64 位 Windows 环境下的算法速率优化 C 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) i7-9750H CPU、主频2.6GHz、Windows10、内存32G、Microsoft Visual Studio 2017编译器。

4.1.3.2. 加密实现方法

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比特的表 L_0 、 L_1 、 L_2 、 L_3 、 L_4 、 L_5 、 L_6 、 L_7 ：

$$L_0[x] = L_{64}(s_0(x), 0, 0, 0, 0, 0, 0, 0)$$

$$L_1[x] = L_{64}(0, s_1(x), 0, 0, 0, 0, 0, 0)$$

$$L_2[x] = L_{64}(0, 0, s_0(x), 0, 0, 0, 0, 0)$$

$$L_3[x] = L_{64}(0, 0, 0, s_1(x), 0, 0, 0, 0)$$

$$L_4[x] = L_{64}(0, 0, 0, 0, s_0(x), 0, 0, 0)$$

$$L_5[x] = L_{64}(0, 0, 0, 0, 0, s_1(x), 0, 0)$$

$$L_6[x] = L_{64}(0, 0, 0, 0, 0, 0, s_0(x), 0)$$

$$L_7[x] = L_{64}(0, 0, 0, 0, 0, 0, 0, s_1(x))$$

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. 解密实现方法

由于 CBC 模式解密的特性，可以使用 AVX 指令对其进行优化实现。定义 256 比特的类型 `__m256i`，包含 8 个 32 比特的字，每个字代表一个分组中的 32 比特值，将算法基于 32 位系统的大表进行实现，使用并行查表指令 `_mm256_i32gather_epi32` 一次查 8 个大表，因此，一次可以解密 8 个密文分组，若分组个数不是 8 的倍数，则将剩下的分组按照普通方式进行解密。

4.1.3.4. 性能测试结果

采用CBC模式，用256字节数据作为输入进行性能测试，每次测试变换密钥，取 10^5 次CBC模式运算测试结果的平均值作为速率测试的结果。具体结果见表4.1.3.4-1。

表4.1.3.4-1 64比特Windows平台软件测试性能

| 算法软件实现类别 | | 加密速率 | 解密速率 |
|----------------------------------|--------------|----------|----------|
| 64比特Windows 环境下的算法速 率优化C实现 | SMBA-128-128 | 1405Mbps | 2751Mbps |
| | SMBA-128-256 | 1110Mbps | 2035Mbps |
| | SMBA-256-256 | 1655Mbps | 2831Mbps |

4.2. 硬件实现

4.2.1. Verilog 硬件仿真实现

算法的 ASIC 实现环境，采用的设计库为 HJTC0.11um 工艺库，采用的综合工具为 Synopsys Design Vision(F-2011.09-SP1 Version)。仿真环境为 Modelsim SE 10.1a。测试结果如下：

表 4.2.1-1 Verilog 硬件仿真测试性能

| | 自测试结果 | | |
|-------|----------------------|----------------------|-----------------------|
| | SMBA-128-128 | SMBA-128-256 | SMBA-256-256 |
| 运算周期 | 627 | 825 | 825 |
| 时钟约束 | 11ns | 11ns | 11ns |
| 加密速率 | 566.37Mbps | 430.44Mbps | 860.88Mbps |
| 解密速率 | 566.37Mbps | 430.44Mbps | 860.88Mbps |
| 面积 | 69065um ² | 87851um ² | 164683um ² |
| 加密吞面比 | 0.00820 | 0.00490 | 0.00523 |
| 解密吞面比 | 0.00820 | 0.00490 | 0.00523 |

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. 优缺点分析

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。

5.1. 算法特色

- (1) 算法设计简洁，三个版本共用非线性变换 S_0 和 S_1 ，以及线性变换 L_{64} ，扩展自然。
- (2) 算法适用于多种软件平台，如 8/32/64 位平台，且算法的软件性能高，实用性强。
- (3) 算法安全性分析充分，安全界清晰，具有较高的安全冗余。
- (4) 算法具有良好的抵抗侧信道攻击的能力。在 3.4 节中，我们给出了 SMBA 算法针对侧信道攻击的防护策略。由于我们在 S_0 和 S_1 的设计中充分考虑到了侧信道防护的问题， S_0 和 S_1 在透明阶、DPA 信噪比、Hamming 重量混乱系数方差等评价侧信道防护能力的指标上都优于 AES 的 S 盒。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。由于针对线性变换的侧信道防护策略是平凡的，所以我们认为 SMBA 具有良好的抗侧信道攻击能力。

5.2. 创新点

SMBA 分组密码算法的创新点主要体现在如下 3 个方面：

- (1) 创新设计了密码性质优、软硬件实现性能高的线性变换 L_{64} ：与同等规模的 MDS 变换相比， L_{64} 的实现代价非常小，其只由两个简单的 L_{32} 变换和一个 L-M 结构线性变换构成，软硬件实现简单，但同时其分支数可达到 6，且能够将 1 个非零字节变换为 7 个非零字节，具有良好的密码学性质。利用 L_{64} 的密码性质，通过理论证明和计算机搜索（两者结果相同）均可证明，SMBA-128 具有很高的抵抗差分攻击和线性攻击的能力。
- (2) SMBA-256 整体设计保证安全：换位变换 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

法的扩散性和抗攻击能力。

- (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减少。

5.3. 不足

由于设计中采用了8比特S盒，且分组长度/密钥长度大，因此SMBA算法不适用于低功耗和硬件资源受限的轻量级应用环境。

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 深度优先实现

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} 的深度优先硬件实现方案，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。

$Q_0(6,4)$

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

$Q_1(9,3)$

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

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

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

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

$$\begin{aligned}y_0 &= t_{15} \oplus x_1 \oplus 1 \\y_3 &= t_{13} \oplus x_1 \oplus 1 \\y_2 &= t_{14} \oplus t_{12} \\y_1 &= t_3 \oplus t_{14}\end{aligned}$$

$$\begin{aligned}t_0 &= (x_3 \& x_1) \oplus 1 \\t_1 &= (x_2 \& x_0) \oplus 1 \\t_2 &= (x_0 \& x_3) \oplus 1 \\t_3 &= (x_3 \& x_1) \oplus 1 \\t_4 &= (x_1 \& x_2) \oplus 1 \\t_5 &= (x_2 \& x_0) \oplus 1 \\t_6 &= x_1 \vee t_5 \\t_7 &= t_3 \oplus t_2 \oplus 1 \\t_8 &= t_4 \vee x_0 \\t_9 &= t_3 \oplus t_4 \oplus 1 \\t_{10} &= (t_2 \& x_2) \oplus 1 \\t_{11} &= x_0 \oplus t_0 \\t_{12} &= t_{11} \oplus t_9 \oplus 1 \\t_{13} &= t_{11} \oplus t_8 \oplus 1 \\t_{14} &= t_{10} \oplus t_1 \oplus 1 \\t_{15} &= (t_{11} \& x_2) \oplus 1 \\t_{16} &= x_0 \oplus t_6 \\t_{17} &= t_{10} \oplus t_7 \oplus 1 \\y_0 &= (t_7 \& t_{14}) \oplus 1 \\y_3 &= t_{13} \oplus x_1 \oplus 1 \\y_1 &= x_3 \oplus t_{17} \oplus 1 \\y_2 &= (t_9 \& t_{16}) \oplus 1 \\t_{23} &= t_{13} \oplus x_3 \oplus 1\end{aligned}$$

在此基础上，可以分别计算 S_0 和 S_1 的深度优先实现方案，其中 S_0 的深度为 18， S_1 的深度为 16。另外， S_0 有深度为 14 的实现，但需要更多的门数。

S_0 :

$$\begin{aligned}t_0 &= x_1 \oplus 1 \\t_1 &= x_3 \oplus 1 \\t_2 &= x_6 \oplus 1 \\t_3 &= x_0 \oplus x_2 \\t_4 &= x_4 \oplus t_2\end{aligned}$$

$$\begin{aligned}
t5 &= x5 \oplus t0 \\
t6 &= t1 \oplus x7 \\
t7 &= x2 \oplus t2 \\
t8 &= t0 \oplus x7 \\
t9 &= x2 \& t0 \\
t10 &= t2 \& x7 \\
t11 &= t7 \& t8 \\
t12 &= t11 \oplus t10 \\
t13 &= t9 \oplus t10 \\
t14 &= x0 \oplus x4 \\
t15 &= x5 \oplus t1 \\
t16 &= x0 \& x5 \\
t17 &= x4 \& t1 \\
t18 &= t14 \& t15 \\
t19 &= t18 \oplus t16 \\
t20 &= t18 \oplus t17 \\
t21 &= t3 \oplus t4 \\
t22 &= t5 \oplus t6 \\
t23 &= t3 \& t5 \\
t24 &= t4 \& t6 \\
t25 &= t21 \& t22 \\
t26 &= t25 \oplus t24 \\
t27 &= t23 \oplus t24 \\
t28 &= t26 \oplus t12 \\
t29 &= t27 \oplus t13 \\
t30 &= t19 \oplus t12 \\
t31 &= t20 \oplus t13 \\
t32 &= (t28 \& t30) \oplus 1 \\
t33 &= (t31 \& t29) \oplus 1 \\
t34 &= (t31 \& t28) \oplus 1 \\
t35 &= t34 \& t33 \\
t36 &= (t29 \& t30) \oplus 1 \\
t37 &= t36 \vee t35 \\
t38 &= (t36 \oplus t35) \oplus 1 \\
t39 &= (t33 \oplus t34) \oplus 1 \\
t40 &= (t32 \oplus t37) \oplus 1 \\
t41 &= (t28 \& t38) \oplus 1 \\
t42 &= t31 \oplus t40
\end{aligned}$$

$$\begin{aligned}
t_{43} &= (t_{30} \& t_{34}) \oplus 1 \\
t_{44} &= t_{28} \oplus t_{38} \\
t_{45} &= (t_{41} \oplus t_{29}) \oplus 1 \\
t_{46} &= t_{44} \oplus t_{45} \\
t_{47} &= t_{42} \oplus t_{46} \\
t_{48} &= t_{43} \oplus t_{44} \\
t_{49} &= t_{39} \oplus t_{44} \\
t_{50} &= t_{45} \oplus t_{48} \\
t_{51} &= t_{49} \oplus t_{47} \\
t_{52} &= t_{48} \oplus t_{47} \\
t_{53} &= t_{48} \& t_0 \\
t_{54} &= t_{47} \& x_7 \\
t_{55} &= t_{52} \& t_8 \\
t_{56} &= t_{55} \oplus t_{54} \\
t_{57} &= t_{53} \oplus t_{54} \\
t_{58} &= t_{45} \oplus t_{49} \\
t_{59} &= t_{45} \& x_5 \\
t_{60} &= t_{49} \& t_1 \\
t_{61} &= t_{58} \& t_{15} \\
t_{62} &= t_{61} \oplus t_{59} \\
t_{63} &= t_{61} \oplus t_{60} \\
t_{64} &= t_{50} \oplus t_{51} \\
t_{65} &= t_{50} \& t_5 \\
t_{66} &= t_{51} \& t_6 \\
t_{67} &= t_{64} \& t_{22} \\
t_{68} &= t_{67} \oplus t_{66} \\
t_{69} &= t_{65} \oplus t_{66} \\
t_{70} &= t_{68} \oplus t_{56} \\
t_{71} &= t_{69} \oplus t_{57} \\
t_{72} &= t_{62} \oplus t_{56} \\
t_{73} &= t_{63} \oplus t_{57} \\
t_{74} &= (x_2 \& x_4) \oplus 1 \\
t_{75} &= (x_0 \& x_4) \oplus 1 \\
t_{76} &= t_2 \vee t_{75} \\
t_{77} &= (t_2 \& x_4) \oplus 1 \\
t_{78} &= (t_2 \& x_2) \oplus 1 \\
t_{79} &= t_{74} \& t_{76} \\
t_{80} &= (t_{75} \oplus t_{77}) \oplus 1
\end{aligned}$$

$$\begin{aligned}
t_{81} &= (t_2 \& x_0) \oplus 1 \\
t_{82} &= t_{78} \& t_{74} \\
t_{83} &= (t_{81} \oplus t_{82}) \oplus 1 \\
t_{84} &= t_{83} \& t_{80} \\
t_{85} &= (t_{79} \oplus t_{81}) \oplus 1 \\
t_{86} &= t_{85} \oplus x_0 \\
t_{87} &= x_4 \oplus t_{84} \\
t_{88} &= t_{82} \vee t_{85} \\
t_{89} &= t_{87} \& t_{86} \\
t_{90} &= t_{89} \oplus t_{88} \\
t_{91} &= (x_2 \oplus t_{84}) \oplus 1 \\
t_{92} &= (t_{86} \& t_{80}) \oplus 1 \\
t_{93} &= (t_{89} \oplus t_2) \oplus 1 \\
t_{94} &= (x_7 \& t_0) \oplus 1 \\
t_{95} &= (x_7 \& x_5) \oplus 1 \\
t_{96} &= (t_0 \& t_1) \oplus 1 \\
t_{97} &= t_{94} \oplus x_5 \\
t_{98} &= x_5 \oplus t_{96} \\
t_{99} &= t_{97} \vee t_{98} \\
t_{100} &= (x_5 \& t_1) \oplus 1 \\
t_{101} &= (t_{94} \oplus t_{96}) \oplus 1 \\
t_{102} &= (t_{95} \oplus t_{99}) \oplus 1 \\
t_{103} &= (t_{100} \oplus t_{101}) \oplus 1 \\
t_{104} &= (t_{103} \oplus t_{95}) \oplus 1 \\
t_{105} &= (t_{103} \& t_0) \oplus 1 \\
t_{106} &= (t_{104} \oplus t_{105}) \oplus 1 \\
t_{107} &= (t_{103} \& x_7) \oplus 1 \\
t_{108} &= t_1 \& t_{107} \\
t_{109} &= t_{103} \oplus t_1 \\
t_{110} &= (t_{108} \oplus t_{102}) \oplus 1 \\
t_{111} &= (t_{107} \oplus t_1) \oplus 1 \\
t_{112} &= (t_{109} \oplus t_{108}) \oplus 1 \\
t_{113} &= (t_{112} \oplus t_{106}) \oplus 1 \\
t_{114} &= t_{45} \vee t_{49} \\
t_{115} &= t_{48} \vee t_{47} \\
t_{116} &= t_{114} \vee t_{115} \\
t_{117} &= t_{93} \oplus t_{70} \\
t_{118} &= t_{117} \& t_{116}
\end{aligned}$$

$$\begin{aligned}
y_3 &= t_{118} \oplus t_{93} \\
t_{120} &= t_{91} \oplus t_{71} \\
t_{121} &= t_{120} \& t_{116} \\
y_0 &= t_{121} \oplus t_{91} \\
t_{123} &= t_{90} \oplus t_{72} \\
t_{124} &= t_{123} \& t_{116} \\
y_4 &= (t_{124} \oplus t_{90}) \oplus 1 \\
t_{126} &= t_{92} \oplus t_{73} \\
t_{127} &= t_{126} \& t_{116} \\
y_7 &= (t_{127} \oplus t_{92}) \oplus 1 \\
t_{129} &= t_{113} \oplus t_{45} \\
t_{130} &= t_{129} \& t_{116} \\
y_1 &= (t_{130} \oplus t_{113}) \oplus 1 \\
t_{132} &= t_{110} \oplus t_{49} \\
t_{133} &= t_{132} \& t_{116} \\
y_5 &= (t_{133} \oplus t_{110}) \oplus 1 \\
t_{135} &= t_{111} \oplus t_{48} \\
t_{136} &= t_{135} \& t_{116} \\
y_6 &= t_{136} \oplus t_{111} \\
t_{138} &= t_{112} \oplus t_{47} \\
t_{139} &= t_{138} \& t_{116} \\
y_2 &= t_{139} \oplus t_{112}
\end{aligned}$$

S₁:

$$\begin{aligned}
t_0 &= x_0 \oplus 1; \\
t_1 &= x_2 \oplus 1; \\
t_2 &= x_5 \oplus 1; \\
t_3 &= x_4 \oplus t_2; \\
t_4 &= x_6 \oplus x_1; \\
t_5 &= t_0 \oplus t_1; \\
t_6 &= x_7 \oplus x_3; \\
t_7 &= x_4 \oplus x_6; \\
t_8 &= t_0 \oplus x_7; \\
t_9 &= x_4 \& t_0; \\
t_{10} &= x_6 \& x_7; \\
t_{11} &= t_7 \& t_8; \\
t_{12} &= t_{11} \oplus t_9; \\
t_{13} &= t_{11} \oplus t_{10}; \\
t_{14} &= t_2 \oplus x_1;
\end{aligned}$$

$t_{15} = t_1 \oplus x_3;$
 $t_{16} = t_2 \& t_1;$
 $t_{17} = x_1 \& x_3;$
 $t_{18} = t_{14} \& t_{15};$
 $t_{19} = t_{18} \oplus t_{16};$
 $t_{20} = t_{18} \oplus t_{17};$
 $t_{21} = t_3 \oplus t_4;$
 $t_{22} = t_5 \oplus t_6;$
 $t_{23} = t_3 \& t_5;$
 $t_{24} = t_4 \& t_6;$
 $t_{25} = t_{21} \& t_{22};$
 $t_{26} = t_{23} \oplus t_{24};$
 $t_{27} = t_{25} \oplus t_{23};$
 $t_{28} = t_{26} \oplus t_{12};$
 $t_{29} = t_{27} \oplus t_{13};$
 $t_{30} = t_{26} \oplus t_{19};$
 $t_{31} = t_{27} \oplus t_{20};$
 $t_{32} = x_4 \oplus t_0;$
 $t_{33} = x_6 \oplus x_7;$
 $t_{34} = t_2 \oplus t_1;$
 $t_{35} = x_1 \oplus x_3;$
 $t_{36} = t_{32} \oplus t_{33};$
 $t_{37} = t_{33} \oplus t_{35};$
 $t_{38} = t_{32} \oplus t_{34};$
 $t_{39} = t_{28} \oplus t_{32};$
 $t_{40} = t_{29} \oplus t_{36};$
 $t_{41} = t_{30} \oplus t_{37};$
 $t_{42} = t_{31} \oplus t_{38};$
 $t_{43} = (t_{39} \& t_{41}) \oplus 1;$
 $t_{44} = (t_{40} \& t_{42}) \oplus 1;$
 $t_{45} = (t_{44} \oplus t_{43}) \oplus 1;$
 $t_{46} = (t_{42} \& t_{39}) \oplus 1;$
 $t_{47} = (t_{46} \oplus t_{45}) \oplus 1;$
 $t_{48} = (t_{41} \& t_{40}) \oplus 1;$
 $t_{49} = t_{46} \vee t_{44};$
 $t_{50} = (t_{44} \& t_{41}) \oplus 1;$
 $t_{51} = (t_{48} \oplus t_{49}) \oplus 1;$
 $t_{52} = t_{51} \oplus t_{39};$

$t_{53} = (t_{47} \& t_{40}) \oplus 1;$
 $t_{54} = (t_{42} \& t_{47}) \oplus 1;$
 $t_{55} = t_{53} \oplus t_{52};$
 $t_{56} = (t_{41} \oplus t_{54}) \oplus 1;$
 $t_{57} = t_{45} \oplus t_{52};$
 $t_{58} = t_{50} \oplus t_{47};$
 $t_{59} = t_{58} \oplus t_{57};$
 $t_{60} = t_{56} \oplus t_{55};$
 $t_{61} = t_{58} \oplus t_{56};$
 $t_{62} = t_{58} \& t_0;$
 $t_{63} = t_{56} \& x_7;$
 $t_{64} = t_{61} \& t_8;$
 $t_{65} = t_{64} \oplus t_{62};$
 $t_{66} = t_{64} \oplus t_{63};$
 $t_{67} = t_{57} \oplus t_{55};$
 $t_{68} = t_{57} \& t_1;$
 $t_{69} = t_{55} \& x_3;$
 $t_{70} = t_{67} \& t_{15};$
 $t_{71} = t_{70} \oplus t_{68};$
 $t_{72} = t_{70} \oplus t_{69};$
 $t_{73} = t_{59} \oplus t_{60};$
 $t_{74} = t_{59} \& t_5;$
 $t_{75} = t_{60} \& t_6;$
 $t_{76} = t_{73} \& t_{22};$
 $t_{77} = t_{74} \oplus t_{75};$
 $t_{78} = t_{76} \oplus t_{74};$
 $y_6 = (t_{77} \oplus t_{65}) \oplus 1;$
 $y_1 = (t_{78} \oplus t_{66}) \oplus 1;$
 $y_4 = (t_{77} \oplus t_{71}) \oplus 1;$
 $y_2 = t_{78} \oplus t_{72};$
 $t_{83} = t_{58} \& x_4;$
 $t_{84} = t_{56} \& x_6;$
 $t_{85} = t_{61} \& t_7;$
 $t_{86} = t_{85} \oplus t_{83};$
 $t_{87} = t_{85} \oplus t_{84};$
 $t_{88} = t_{57} \& t_2;$
 $t_{89} = t_{55} \& x_1;$
 $t_{90} = t_{67} \& t_{14};$
 $t_{91} = t_{90} \oplus t_{88};$

```
t92 = t90 ⊕ t89;  
t93 = t59 & t3;  
t94 = t60 & t4;  
t95 = t73 & t21;  
t96 = t93 ⊕ t94;  
t97 = t95 ⊕ t93;  
y5 = (t96 ⊕ t86) ⊕ 1;  
y0 = t97 ⊕ t87;  
y3 = (t96 ⊕ t91) ⊕ 1;  
y7 = (t97 ⊕ t92) ⊕ 1;
```

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) $S_0^1 = M_1(f^1) \oplus \text{Con}_1, S_0^2 = M_1(f^2)$

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。

$(c^1, c^2) = \text{b13_TI}(b^1, b^2)$ 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) $e^1 = \text{INV_2}(d^1)$, $e^2 = \text{INV_2}(d^2)$;
- (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) $c^1 = \text{SQU}(h^1)$, $c^2 = \text{SQU}(h^2)$

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 $\text{GF}(2^2)$ 在基 $(Z, 1)$ 上的求逆运算， SQU 表示 $\text{GF}(2^2)$ 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) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (59) $t_3^1 = s^1 \oplus s^2;$
- (60) $t_3^2 = s^3 \oplus s^4;$
- (61) $q_8^1 = x_0^1 \oplus t_0^1 \oplus t_2^1 \oplus t_3^1;$
- (62) $q_8^2 = x_0^2 \oplus t_0^2 \oplus t_2^2 \oplus t_3^2;$
- (63) $q_9^1 = 1 \oplus x_0^1 \oplus x_1^1 \oplus t_2^1 \oplus t_3^1;$
- (64) $q_9^2 = x_0^2 \oplus x_1^2 \oplus t_2^2 \oplus t_3^2;$
- (65) $s^1 = q_8^1 \& q_9^1;$
- (66) $s^2 = q_8^2 \& q_9^1;$
- (67) $s^3 = q_8^1 \& q_9^2;$
- (68) $s^4 = q_8^2 \& q_9^2;$
- (69) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (70) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (71) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (72) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (73) $t_4^1 = s^1 \oplus s^2;$
- (74) $t_4^2 = s^3 \oplus s^4;$

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

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

$$(77) \quad y_1^1 = x_0^1 \oplus t_1^1 \oplus t_3^1 \oplus t_4^1;$$

$$(78) \quad y_1^2 = x_0^2 \oplus t_1^2 \oplus t_3^2 \oplus t_4^2;$$

$$(79) \quad y_2^1 = x_2^1 \oplus t_0^1 \oplus t_1^1 \oplus t_2^1 \oplus t_4^1;$$

$$(80) \quad y_2^2 = x_2^2 \oplus t_0^2 \oplus t_1^2 \oplus t_2^2 \oplus t_4^2;$$

$$(81) \quad y_3^1 = x_0^1 \oplus x_1^1 \oplus t_0^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1;$$

$$(82) \quad y_3^2 = x_0^2 \oplus x_1^2 \oplus t_0^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2;$$

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

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

$(e^1, e^2) = Q1_TI(a^1, a^2)$ 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：

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

$$(2) \quad x^2 = a^2;$$

$$(3) \quad \text{将 } x^1 \text{ 分为 4 个 1 比特 } x^1 = x_0^1 \parallel x_1^1 \parallel x_2^1 \parallel x_3^1;$$

$$(4) \quad \text{将 } x^2 \text{ 分为 4 个 1 比特 } x^2 = x_0^2 \parallel x_1^2 \parallel x_2^2 \parallel x_3^2;$$

$$(5) \quad q_0^1 = 1 \oplus x_0^1 \oplus x_3^1;$$

$$(6) \quad q_0^2 = x_0^2 \oplus x_3^2;$$

$$(7) \quad q_1^1 = 1 \oplus x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1;$$

$$(8) \quad q_1^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus x_3^2;$$

$$(9) \quad s^1 = q_0^1 \& q_1^1;$$

$$(10) \quad s^2 = q_0^2 \& q_1^1;$$

$$(11) \quad s^3 = q_0^1 \& q_1^2;$$

$$(12) \quad s^4 = q_0^2 \& q_1^2;$$

$$(13) \quad s^1 = s^1 \oplus R_1 \oplus R_2;$$

$$(14) \quad s^2 = s^2 \oplus R_2 \oplus R_3;$$

$$(15) \quad s^3 = s^3 \oplus R_3 \oplus R_4;$$

$$(16) \quad s^4 = s^4 \oplus R_4 \oplus R_1;$$

$$(17) \quad t_0^1 = s^1 \oplus s^2;$$

$$(18) \quad t_0^2 = s^3 \oplus s^4;$$

- (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) $q_8^2 = x_1^2 \oplus x_2^2 \oplus t_0^2 \oplus t_1^2;$
- (63) $q_9^1 = x_2^1 \oplus t_0^1 \oplus t_1^1 \oplus t_3^1;$
- (64) $q_9^2 = x_2^2 \oplus t_0^2 \oplus t_1^2 \oplus t_3^2;$
- (65) $s^1 = q_8^1 \& q_9^1;$
- (66) $s^2 = q_8^2 \& q_9^1;$
- (67) $s^3 = q_8^1 \& q_9^2;$
- (68) $s^4 = q_8^2 \& q_9^2;$
- (69) $s^1 = s^1 \oplus R_1 \oplus R_2;$
- (70) $s^2 = s^2 \oplus R_2 \oplus R_3;$
- (71) $s^3 = s^3 \oplus R_3 \oplus R_4;$
- (72) $s^4 = s^4 \oplus R_4 \oplus R_1;$
- (73) $t_4^1 = s^1 \oplus s^2;$
- (74) $t_4^2 = s^3 \oplus s^4;$
- (75) $y_0^1 = x_2^1 \oplus x_3^1 \oplus t_0^1 \oplus t_1^1 \oplus t_2^1 \oplus t_3^1 \oplus t_4^1;$
- (76) $y_0^2 = x_2^2 \oplus x_3^2 \oplus t_0^2 \oplus t_1^2 \oplus t_2^2 \oplus t_3^2 \oplus t_4^2;$
- (77) $y_1^1 = x_0^1 \oplus x_1^1 \oplus x_2^1 \oplus x_3^1 \oplus t_2^1;$
- (78) $y_1^2 = x_0^2 \oplus x_1^2 \oplus x_2^2 \oplus x_3^2 \oplus t_2^2;$

- (19) $y1=y1\oplus R_1\oplus R_2, y2=y2\oplus R_2\oplus R_3, y3=y3\oplus R_3\oplus R_4, y4=y4\oplus R_4\oplus R_5,$
 $y5=y5\oplus R_5\oplus R_6, y6=y6\oplus R_6\oplus R_7, y7=y7\oplus R_7\oplus R_8, y8=y8\oplus R_8\oplus R_9,$
 $y9=y9\oplus R_9\oplus R_{10}, y10=y10\oplus R_{10}\oplus R_{11}, y11=y11\oplus R_{11}\oplus R_{12},$
 $y12=y12\oplus R_{12}\oplus R_{13}, y13=y13\oplus R_{13}\oplus R_{14}, y14=y14\oplus R_{14}\oplus R_{15},$
 $y15=y15\oplus R_{15}\oplus R_{16}, y16=y16\oplus R_{16}\oplus R_1$
- (20) $g^1 = y1\oplus y2\oplus y3\oplus y4\oplus y5\oplus y6\oplus y7\oplus y8,$
 $g^2 = y9\oplus y10\oplus y11\oplus y12\oplus y13\oplus y14\oplus y15\oplus y16$
- (21) $h^1 = g^1\|g^1\|g^1\|g^1\|g^1\|g^1\|g^1\|g^1, h^2 = g^2\|g^2\|g^2\|g^2\|g^2\|g^2\|g^2\|g^2$
- (22) $e^1 = e_0^1\|e_1^1, e^2 = e_0^2\|e_1^2, f^1 = d^1\|c^1, f^2 = d^2\|c^2$
- (23) $t^1 = e^1\oplus f^1, t^2 = e^2\oplus f^2$
- (24) $b^1 = t^1 \& h^1, b^2 = t^1 \& h^2, b^3 = t^2 \& h^1, b^4 = t^2 \& h^2;$
- (25) $b^1 = b^1\oplus R_1\oplus R_2, b^2 = b^2\oplus R_2\oplus R_3, b^3 = b^3\oplus R_3\oplus R_4, b^4 = b^4\oplus R_4\oplus R_1;$
- (26) $b^1 = b^1\oplus b^2, b^2 = b^3\oplus b^4;$
- (27) $f^1 = b^1\oplus e^1, f^2 = b^2\oplus e^2$

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) $a^1 = M_2(x^1 \oplus \text{Con}_2)$ ，将 a^1 分为 2 个 4 比特 a_0^1, a_1^1 ，即 $a^1 = a_0^1\|a_1^1$ ；
 $a^2 = M_2(x^2)$ ，将 a^2 分为 2 个 4 比特 a_0^2, a_1^2 ，即 $a^2 = a_0^2\|a_1^2$ ；
- (2) $b_0^1 = a_0^1\oplus a_1^1, b_0^2 = a_0^2\oplus a_1^2$ ；
- (3) $c^1 = (Z^2 Y^4) \otimes (b_0^1 \otimes b_0^1); c^2 = (Z^2 Y^4) \otimes (b_0^2 \otimes b_0^2)$ ；
- (4) $b_1^1 = a_0^1 \otimes a_1^1 \oplus c^1, b_1^2 = a_0^1 \otimes a_1^2, b_1^3 = a_0^2 \otimes a_1^1, b_1^4 = a_0^2 \otimes a_1^2 \oplus c^2$ ；
- (5) $c_1^1 = b_1^1 \oplus R_1 \oplus R_2, c_1^2 = b_1^2 \oplus R_2 \oplus R_3, c_1^3 = b_1^3 \oplus R_3 \oplus R_4, c_1^4 = b_1^4 \oplus R_4 \oplus R_1$ ；
- (6) $d^1 = c_1^1 \oplus c_1^2, d^2 = c_1^3 \oplus c_1^4$ ；
- (7) $(e^1, e^2) = \text{INV_4_TI}(d^1, d^2)$ ；
- (8) $f_0^1 = e^1 \otimes a_1^1, f_0^2 = e^1 \otimes a_1^2, f_0^3 = e^2 \otimes a_1^1, f_0^4 = e^2 \otimes a_1^2$ ；

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

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

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

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

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

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：

$$(1) \quad \text{将 } d^1 \text{ 分为 2 个 2 比特 } d_0^1, d_1^1, \text{ 即 } d^1 = d_0^1 \parallel d_1^1;$$

$$(2) \quad \text{将 } d^2 \text{ 分为 2 个 2 比特 } d_0^2, d_1^2, \text{ 即 } d^2 = d_0^2 \parallel d_1^2;$$

$$(3) \quad b_0^1 = d_0^1 \oplus d_1^1, \quad b_0^2 = d_0^2 \oplus d_1^2;$$

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

$$(5) \quad b_1^1 = d_0^1 \otimes d_1^1 \oplus c^1, \quad b_1^2 = d_0^1 \otimes d_1^2, \quad b_1^3 = d_0^2 \otimes d_1^1, \quad b_1^4 = d_0^2 \otimes d_1^2 \oplus c^2;$$

$$(6) \quad c_1^1 = b_1^1 \oplus R_1 \oplus R_2, \quad c_1^2 = b_1^2 \oplus R_2 \oplus R_3, \quad c_1^3 = b_1^3 \oplus R_3 \oplus R_4, \quad c_1^4 = b_1^4 \oplus R_4 \oplus R_1;$$

$$(7) \quad d^1 = c_1^1 \oplus c_1^2, \quad d^2 = c_1^3 \oplus c_1^4;$$

$$(8) \quad e^1 = \text{INV_2}(d^1), \quad e^2 = \text{INV_2}(d^2);$$

$$(9) \quad f_0^1 = e^1 \otimes d_1^1, \quad f_0^2 = e^1 \otimes d_1^2, \quad f_0^3 = e^2 \otimes d_1^1, \quad f_0^4 = e^2 \otimes d_1^2;$$

$$(10) \quad f_1^1 = e^1 \otimes d_0^1, \quad f_1^2 = e^1 \otimes d_0^2, \quad f_1^3 = e^2 \otimes d_0^1, \quad f_1^4 = e^2 \otimes d_0^2;$$

$$(11) \quad f^1 = f_0^1 \parallel f_1^1, \quad f^2 = f_0^2 \parallel f_1^2, \quad f^3 = f_0^3 \parallel f_1^3, \quad f^4 = f_0^4 \parallel f_1^4;$$

$$(12) \quad g^1 = f^1 \oplus R_1 \oplus R_2, \quad g^2 = f^2 \oplus R_2 \oplus R_3, \quad g^3 = f^3 \oplus R_3 \oplus R_4, \quad g^4 = f^4 \oplus R_4 \oplus R_1;$$

$$(13) \quad e^1 = g^1 \oplus g^2, \quad e^2 = g^3 \oplus g^4$$

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。