

分组密码算法 FESH 设计文档

设计者：贾珂婷、董晓阳、魏淙洺、李铮、周海波、丛天硕

完成单位：清华大学、山东大学

目 录

1 算法描述.....	1
1.1 简介.....	1
1.1.1 分组密码参赛算法必须满足下列基本要求.....	1
1.1.2 术语缩略语.....	1
1.1.3 符号说明.....	2
1.2 算法整体结构.....	2
1.3 加密算法.....	4
1.4 解密算法.....	7
1.5 密钥调度.....	8
2 设计原理.....	13
2.1 设计的主要思路 and 策略.....	13
2.2 S 盒设计	13
2.3 扩散层设计.....	14
2.4 密钥调度算法.....	16
3 安全性分析.....	16
3.1 分组长度 128 的算法版本的安全性分析.....	16
3.1.1 差分类分析.....	16
3.1.2 线性分析.....	18
3.1.3 不可能差分分析.....	19
3.1.4 积分攻击.....	20

3.1.5 相关密钥分析.....	21
3.1.6 小结.....	22
3.2 分组长度 256 的算法版本安全性分析.....	23
3.2.1 差分分析.....	23
3.2.2 线性分析.....	25
3.2.3 不可能差分分析.....	26
3.2.4 积分攻击.....	26
3.2.5 相关密钥分析.....	27
3.2.6 小结.....	28
4 性能分析.....	30
4.1 软件性能分析.....	30
4.1.1 算法软件运行平台.....	30
4.1.2 CBC 模式下算法性能分析.....	30
4.1.3 ECB 模式下算法性能分析.....	30
4.2 ARM 平台下算法性能分析	32
4.3 硬件性能分析.....	32
5 优缺点声明.....	33

1 算法描述

1.1 简介

1.1.1 分组密码参赛算法必须满足下列基本要求

- 1) 分组密码算法的分组长度/密钥长度应至少支持 128/128、128/256 和 256/256 三种情况。
- 2) 分组密码算法应能抵抗差分密码分析和线性密码分析等已知攻击。
- 3) 分组密码算法应能在 32 位、64 位等软件平台和 ASIC、FPGA 等硬件平台高效实现。
- 4) 算法提交材料应满足本文档规定的形式规范的要求，提交的算法应提供能够达到其声明的安全强度。

1.1.2 术语缩略语

明文	待加密的消息
密文	明文被加密后的值
密钥	分组密码的秘密输入参数，加密和解密的密钥相同
轮函数	分组加密算法中的迭代函数
轮密钥	由密钥调度算法生成，用于加密和解密中
比特	取值为 0 或者 1 的二元数字
半字节	4 个比特组成的序列
分组	一个固定长度的比特序列，文中分组长度有两种 128 比特和 256 比特，取决于算法版本
状态	128 比特或者 256 比特的分组，加解密的中间值
字 (word)	32 或者 64 比特的序列，依赖于分组长度，128 比特分组的字为 32 比特，256 比特分组的字为 64 比特

1.1.3 符号说明

P	明文
C	密文
$x y$	表示字符串 x 和 y 的级联
$x \lll s$	字 x 循环左移 s 比特，其中 x 可以是 32 比特或者 64 比特的字
$x \vee y$	x 和 y 进行按位或运算
$x \wedge y$	x 和 y 进行按位与运算
$\sim x$	x 按比特取反
$x \oplus y$	x 和 y 进行按位异或运算
n	分组长度，如 128 比特分组时 $n=128$ ，256 比特分组时 $n=256$ ，取决于算法版本
N	算法轮数，大小依赖算法版本
m	密钥长度
RK_i	轮密钥，长度为 n 比特
$K[i \sim j]$	从左开始取密钥 K 的第 i 到第 j 比特，即 $j-i+1$ 比特长的序列

1.2 算法整体结构

FESH 算法分组长度为 n 比特，密钥长度为 m 比特，记为 FESH- n - m ，本算法包括 FESH-128-128、FESH-128-192、FESH-128-256，FESH-256-256、FESH-256-384，FESH-256-512 六个版本，每个版本的轮数 N ，见表 1-1。

表 1-1 算法版本及轮数

算法版本	分组长度	密钥长度	轮数 N
FESH-128-128	128	128	16
FESH-128-192	128	192	20
FESH-128-256	128	256	20
FESH-256-256	256	256	24
FESH-256-384	256	384	28
FESH-256-512	256	512	28

FESH 算法的加解密基于字(word)操作, 比特串到字的生成规则如下: 每 8 个连续的比特串 $b_0b_1b_2b_3b_4b_5b_6b_7$ 组成一个字节, 其中 b_0 为最高有效位, b_7 为最低有效位; 从字节到字的转换采用大端序, 即将低位字节存放在内存的高地址, 高位字节存放在内存的低地址。

128 比特分组可以表示为 4 个 32 比特字 (w_0, w_1, w_2, w_3), 每一行代表一个字, 如下图所示。

[illegible]

图 1-1 状态与字的转换

256 比特分组可以表示为 4 个 64 比特字，每一行代表一个 64 比特的字。

FESH 算法采用 SPN 结构, 包括加密算法、解密算法、密钥调度算法三部分。

- 1) 加密算法：使用秘密的密钥，将明文加密成密文。
- 2) 解密算法：使用与加密算法相同的密钥，将密文解密成明文。

3) 密钥调度算法：将密钥扩展生成加解密算法中用到的轮密钥。

1.3 加密算法

FESH 算法基于 SPN 结构，首先将明文分组 P 与白化密钥 (RK_0) 异或，然后进行 N 轮(轮函数)迭代运算,轮函数包括:S 盒替换(SubNibble),字混合(MixWord)和轮密钥加操作，见图 1-2，其中 RK_1, RK_2, \dots, RK_N 为轮密钥，由密钥调度算法生成。最后输出为密文 C 。加密算法伪代码如下：

```

 $X_0 = P \oplus RK_0$ 
for  $i=0$  to  $N-1$ 
{
    //第  $r=i+1$  轮( $r=1,2,\dots,N$ )
     $Y_i = \text{SubNibble}(X_i)$ 
     $Z_i = \text{MixWord}(Y_i)$ 
     $X_{i+1} = Z_i \oplus RK_{i+1}$ 
}
 $C = X_N$ 

```

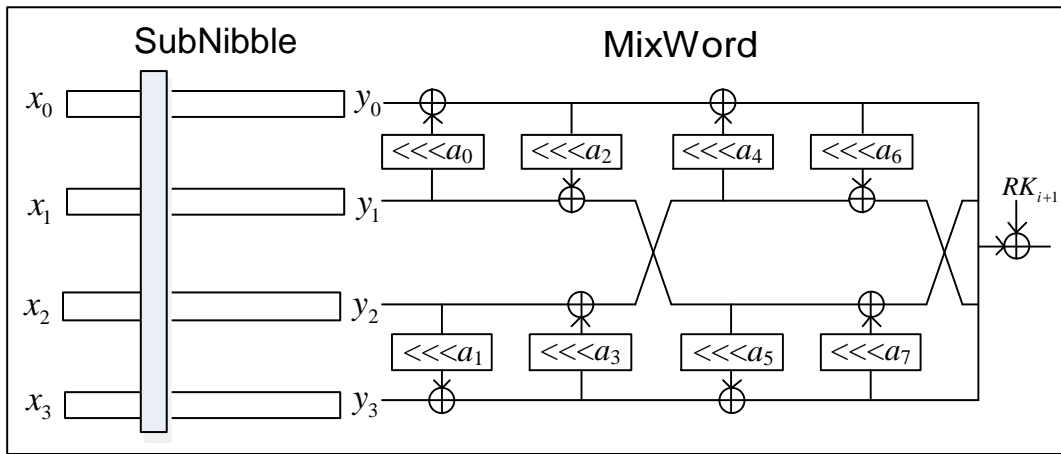


图 1-2 轮函数示意图

- S 盒替换 (SubNibble)

S 盒替换是基于半字节的非线性替换，其将 n 比特状态 X 划分为四个 $n/4$ 比特

的字(x_0, x_1, x_2, x_3), 取出每个字的第 i 比特, 合并为半字节 $s_i=(x_{0i}||x_{1i}||x_{2i}||x_{3i})$, 利用该半字节查询 S 盒, 用获得的新值替换原先的值, 获得输出状态 $Y, Y= \text{SubNibble}(X)$ 。

S 盒替换表见表 1-2, 例如, S 盒输入为 1, 对应的 S 盒的输出为 13。

表 1-2 S 替换表 S0

输入	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
输出	3	13	15	10	0	7	12	1	4	2	9	5	11	14	6	8

为了便于软件实现, S0 可以采用 bit-slice 实现, 4 比特输入为 ($x_{0i}||x_{1i}||x_{2i}||x_{3i}$), 实现如下:

```

r0 = x1i
r1 = x2i
r2 = x0i
r3 = x3i
r4 = r0 ∨ r2
r1 = r1 ⊕ r4
r4 = r3 ∨ r1
r2 = r2 ⊕ r4
r1 = r1 ⊕ r3
r4 = r0 ∧ r2
r3 = r3 ⊕ r4
r2 = r2 ⊕ r0
r0 = r0 ⊕ r1
r4 = ~r2
r4 = r4 ∨ r3
r1 = r1 ⊕ r4
r4 = ~r0
r4 = r4 ∨ r1
r3 = r3 ⊕ r4
y0i = r2
y1i = r0
y2i = r1
y3i = r3

```

最后获得 S0 的 4 比特输出为 ($y_{0i}||y_{1i}||y_{2i}||y_{3i}$)。

- 字混合 (MixWord)

字混合运算是基于字的线性运算, 对于 S 盒替换后的状态 Y 划分为四个字 (y_0, y_1, y_2, y_3), 然后执行下面的运算, 获得 $Z=(z_0, z_1, z_2, z_3)= \text{MixWord}(Y)$, 见图 1-3。

$$\begin{aligned}
y_0 &= y_0 \oplus (y_1 \lll a_0) \\
y_3 &= y_3 \oplus (y_2 \lll a_1) \\
y_1 &= y_1 \oplus (y_0 \lll a_2) \\
y_2 &= y_2 \oplus (y_3 \lll a_3) \\
t &= y_1 \\
y_1 &= y_2 \\
y_2 &= t \\
y_0 &= y_0 \oplus (y_1 \lll a_4) \\
y_3 &= y_3 \oplus (y_2 \lll a_5) \\
y_1 &= y_1 \oplus (y_0 \lll a_6) \\
y_2 &= y_2 \oplus (y_3 \lll a_7) \\
z_0 &= y_0 \\
z_1 &= y_2 \\
z_2 &= y_1 \\
z_3 &= y_3
\end{aligned}$$

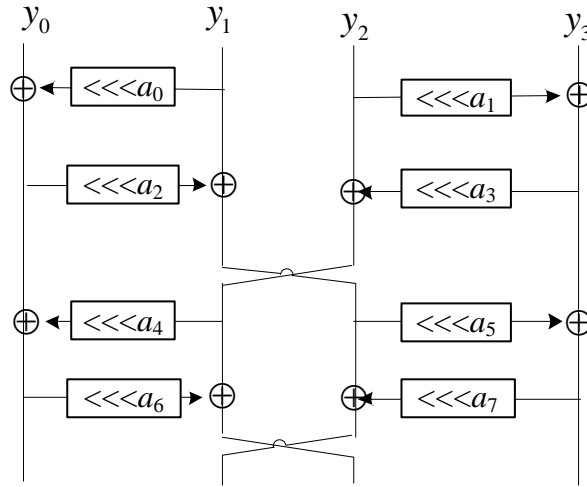


图 1-3 字混合示意图

对于分组长度 128 比特的分组，字长为 32 比特，对于分组长度为 256 比特的分组，字长为 64 比特，两种分组长度的字混合运算的移位常数 a_0, \dots, a_7 见表 1-3。

表 1-3 字混合参数

版本	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
FESH-128-128/192/256	29	13	4	21	15	19	25	6
FESH-256-256/384/512	58	33	8	1	17	44	5	9

• 轮密钥加

加密时，首先将明文分组逐比特异或白化密钥 RK_0 。在第 r 轮 ($r=1,2,\dots,N$) 的加密过程中，将字混合运算后状态 Z 逐比特异或轮密钥 RK_r ，获得输出状态。

1.4 解密算法

解密算法为加密算法的逆运算。进行 N 轮（轮函数）迭代运算，迭代运算包括：解密轮密钥加操作，字混合逆运算 ($MixWord^{-1}$)，S 盒替换逆运算 ($SubNibble^{-1}$)。最后输出与白化密钥 RK_0 异或获得明文，解密算法伪代码如下：

```

 $X_N = C$ 
for  $i = N-1$  to 0
{
    //第  $r = N-i$  轮 ( $r=1,2,\dots,N$ )
     $Z_i = X_{i+1} \oplus RK_{i+1}$ 
     $Y_i = MixWord^{-1}(Z_i)$ 
     $X_i = SubNibble^{-1}(Y_i)$ 
}
 $P = X_0 \oplus RK_0$ 

```

• 解密轮密钥加

解密时，初始状态为密文，在第 r 轮 ($r=1,2,\dots,N$) 的解密过程中，将状态逐比特异或轮密钥 RK_{N-r+1} ，将最后一个状态异或白化密钥 RK_0 获得输出明文。

• 字混合逆运算 ($MixWord^{-1}$)

将上一步获得的状态 Z 划分为四个字 (z_0, z_1, z_2, z_3)，然后执行下面的运算，获得 $Y=(y_0, y_1, y_2, y_3) = MixWord^{-1}(Z)$ 。分组长度 128 比特和分组长度 256 比特的字混合逆

运算的移位常数 a_0, \dots, a_7 与字混合运算所采用的移位常数相同，见表 1-3。

$$\begin{aligned}
 y_0 &= z_0 \\
 y_1 &= z_2 \\
 y_2 &= z_1 \\
 y_3 &= z_3 \\
 y_2 &= y_2 \oplus (y_3 \lll a_7) \\
 y_1 &= y_1 \oplus (y_0 \lll a_6) \\
 y_3 &= y_3 \oplus (y_2 \lll a_5) \\
 y_0 &= y_0 \oplus (y_1 \lll a_4) \\
 t &= y_1 \\
 y_1 &= y_2 \\
 y_2 &= t \\
 y_2 &= y_2 \oplus (y_3 \lll a_3) \\
 y_1 &= y_1 \oplus (y_0 \lll a_2) \\
 y_3 &= y_3 \oplus (y_2 \lll a_1) \\
 y_0 &= y_0 \oplus (y_1 \lll a_0)
 \end{aligned}$$

• S 盒替换逆运算 (SubNibble^{-1})

S 盒替换逆运算是基于半字节的非线性替换，其将 n 比特状态 Y 划分为四个字 (y_0, y_1, y_2, y_3) ，取出每个字的第 i 比特，合并为半字节 $s_i = (y_{0i} \parallel y_{1i} \parallel y_{2i} \parallel y_{3i})$ ，利用该半字节查询 S 盒逆向替换表，用获得的新值替换原先的值，获得输出状态 X ， $X = \text{SubNibble}^{-1}(Y)$ 。

S 盒逆运算见表 1-4，例如，输入为 7，对应的输出为 5。

表 1-4 S 盒逆向替换表

输入	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
输出	4	7	9	0	8	11	14	5	15	10	3	12	6	1	13	2

1.5 密钥调度

密钥调度算法采用 F 函数生成轮密钥 RK_i 。F 函数采用简化的轮函数生成轮密钥。

- 对于算法 FESH-128-128 和 FESH-256-256，密钥长度与分组长度相等，其密钥调度算法利用函数 F 和密钥 K 生成轮密钥 RK_i ($i=0, 1, \dots, N$)。

$$RK_0=K$$

$$RK_i=F(RK_{i-1}, Cst_{i-1}), i=1, \dots, N$$

- 对于算法 FESH-128-256 和 FESH-256-512，密钥长度为分组长度的两倍，密钥调度算法首先将 $2n$ 长度的主密钥 K 分割为两个 n 比特的轮密钥 RK_0 和 RK_1 ，然后采用 F 函数生成其他轮密钥，见图 1-4。

$$RK_0=K[0\sim n-1]$$

$$RK_1=K[n\sim 2n-1]$$

$$RK_{i+1}=F(RK_i, Cst_{i-1}) \oplus RK_{i-1}, i=1, \dots, N-1$$

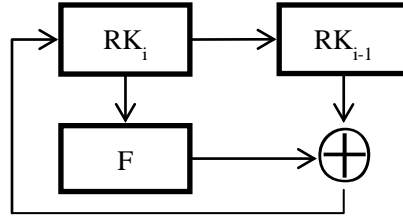


图 1-4 密钥生成示意图

- 其他密钥长度时， $n < |K| < 2n$ ，首先将密钥填充为 $2n$ 长度。填充方案为直接在左边补 0 填充成 $2n$ 比特。如果密钥长度是 $1.5n$ 比特时，我们在密钥前面填充 $0.5n$ 个 0 使之成为 $2n$ 比特。然后采用 $2n$ 比特长度的密钥调度方法，区别是 F 函数采用的常数是不同的，本算法给出了 FESH-128-192 和 FESH-256-384 的两种情况的密钥调度如下：

$$RK_0=0\dots\dots 0\|K[0\sim 0.5n-1]$$

$$RK_1=K[0.5n\sim 1.5n-1]$$

$$RK_{i+1}=F(RK_i, Cst_{i-1}) \oplus RK_{i-1}, i=1, \dots, N-1$$

F 函数是简化的轮函数，包括常量加(ARC)，查 S 盒(SubNibble_K)和密钥字混

合(MixWord_K)运算，见图 1-5。轮函数 $Z_K=F(RK_i, Cst_{i-1})$ 的伪代码如下：

$$X_K = ARC(RK_i, Cst_{i-1})= RK_i \oplus (Cst_{i-1},0,0,0)$$

$$Y_K = SubNibble_K (X_K)$$

$$Z_K= MixWord_K (Y_K).$$

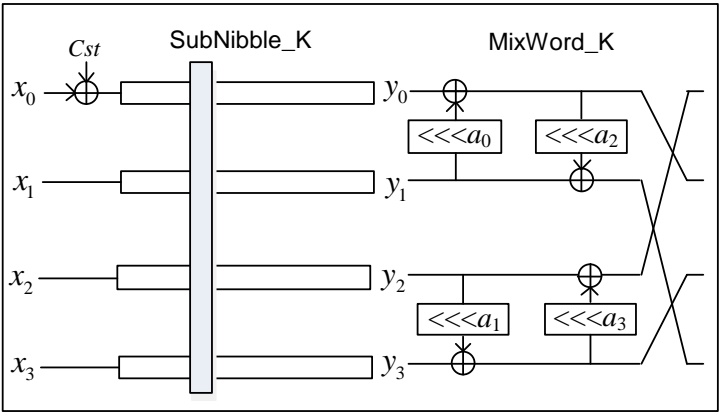


图 1-5 F 函数示意图

常量加 ARC 通过异或常数，破坏密钥调度算法的对称特性，每一轮的常数都不同。初始值 Cst_0 是一个常数字，分组长度 128 时 Cst_0 是 32 位，分组长度 256 时 Cst_0 是 64 位。不同的算法版本采用不同的常数初始值，见表 1-5，其余常数由初始常数通过循环移位生成，用于密钥调度过程中，第 i 个常数 $Cst_i= Cst_0<<<i$ 。常数从圆周率 π 的小数部分选取。

表 1-5 初始常量表

版本	常量 Cst_0
FESH-128-128	0x243F6A88
FESH-128-192	0x85A308D3
FESH-128-256	0x13198A2E
FESH-256-256	0x03707344A4093822
FESH-256-384	0x299F31D0082EFA98
FESH-256-512	0xEC4E6C89452821E6

SubNibble_K 采用 4 比特的 S 盒，也是基于半字节的非线性替换，其将 n 比特状态 X_K 划分为四个字(x_0, x_1, x_2, x_3)，取出每个字的第 i 比特，合并为半字节 $s_i = (x_{0i} || x_{1i} || x_{2i} || x_{3i})$ ，利用该半字节查询 S 盒，用获得的新值替换原先的值，获得输出状态 Y_K ， $Y_K = \text{SubNibble_K}(X_K)$ 。采用 4 比特的 S 盒见表 1-6。

表 1-6 密钥调度采用的 S 盒替换表 S1

输入	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
输出	7	12	0	10	14	13	5	6	2	8	15	4	11	9	3	1

为了便于软件实现，S1 可以采用 bit-slice 实现，输入 $(x_{0i} || x_{1i} || x_{2i} || x_{3i})$ ，bit-slice 实现如下：

```

r3 = x0i
r0 = x1i
r1 = x2i
r2 = x3i
r1 = ~r1
r4 = r1 ∨ r3
r2 = r2 ⊕ r4
r4 = r0 ∨ r1
r3 = r3 ⊕ r4
r4 = ~r0
r4 = r4 ∧ r2
r1 = r1 ⊕ r4
r4 = r2 ∧ r3
r0 = r0 ⊕ r4

```

最后获得 S1 的输出为 $(r1 || r3 || r2 || r0)$ 。

MixWord_Key 基于字的线性运算，密钥调度中字混合运算采用轮函数数字混合运算的简化版，见图 1-6，输入 $Y_K = (y_0, y_1, y_2, y_3)$ ，输出 $Z_K = (z_0, z_1, z_2, z_3) = \text{MixWord_K}(Y_K)$ ，运算如下：

$$\begin{aligned}
 y_0 &= y_0 \oplus (y_1 \lll a_0) \\
 y_3 &= y_3 \oplus (y_2 \lll a_1) \\
 y_1 &= y_1 \oplus (y_0 \lll a_2) \\
 y_2 &= y_2 \oplus (y_3 \lll a_3) \\
 z_0 &= y_2 \\
 z_1 &= y_0 \\
 z_2 &= y_3 \\
 z_3 &= y_1
 \end{aligned}$$

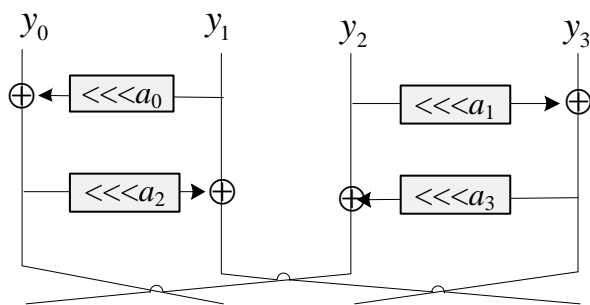


图 1-6 密钥调度中的字混合示意图

分组长度为 128 和 256 的版本密钥调度中字混合运算（MixWord_K）采用的参数 a_0, a_1, a_2, a_3 见表 1-7。

表 1-7 密钥调度中字混合参数

版本	a_0	a_1	a_2	a_3
FESH-128-128/192/256	24	30	7	18
FESH-256-256/384/512	1	18	50	24

2 设计原理

2.1 设计的主要思路 and 策略

我们致力于设计一个安全性强，软硬件运行效率高，结构简单，易于实现，灵活性强的分组密码算法，不仅适用于计算机通信网络，还可以用于物联网、移动通信网络等，提供安全高效的密码保护。采用替换-置换网络（SPN）结构作为算法整体结构，该结构是目前广泛采用的分组密码结构，国际上重要的密码算法标准 AES、PRESENT 等都采用该结构。SPN 结构通过多次迭代轮函数实现，轮函数包括混淆层和扩散层，混淆层由非线性运算构成如 S 盒，扩散层由可逆的线性运算构成。

算法支持 128 比特的分组和 256 比特的分组，采用统一的轮函数结构，设计方案简洁，便于软硬件实现，灵活性高，设计方案兼顾了安全性和软硬件实现性能。在保证算法安全性的条件下，算法在 8 位、32 位和 64 位等不同的软件处理平台和 FPGA、ASIC 硬件平台上都具有较好的性能。

密钥扩展算法采用简化的轮函数，具有更高运行效率的 S 盒以及简化的字混合运算，使密钥扩散与轮函数扩散尽可能不相交，能够有效抵抗相关密钥攻击和弱密钥攻击等。

2.2 S 盒设计

4 比特的 S 盒具有很好的密码学性质，并且占用面积小，延时低，增加侧信道防护代价相对低，被用于很多密码算法的设计中。本算法采用了 4 比特的 S 盒，S 盒的构造不仅考虑了常规的安全特征，如差分概率、线性偏差优势、代数次数、固定点等，还特别考虑了低汉明重量的差分分布情况，以及低汉明重量的线性掩码分布。轮函数选择安全强度优的 S 盒，也充分考虑 S 盒的实现问题，对于 4 比特的 S 盒，采用查表方式进行软件实现比较耗时，我们采用 bit-slice 技术，对 S 盒进行并行实现，提高软件性能。bit-slice 技术目前已经被用于 Serpent、RECTANGLE 等算法。

Saarinen 等对所有 4-4 的活性 S 盒进行分析, 证明“最优”的 S 盒共 16 个等价类, 我们对其给出的最大 $BN=3$ 的类进一步分析, 寻找输入输出 1 比特的线性掩码数量最小的 S 盒, 最小值为 4, 最后发现只有两个等价类中存在这种情况, 基于此我们从这两个等价类构造 S 盒, 从中寻找 bit-slice 指令运行效率高的 S 盒。张文涛等在 RECTANGLE 算法的设计中也考虑了低汉明重量差分和线性的情况, 不过我们用的 S 盒属性不同, 不在同一个类。

S 盒 S_0 的指标如下:

- 1) 最大差分概率为 2^{-2} , 输入输出是 1 比特的差分个数为 0;
- 2) 线性逼近概率 2^{-2} , 输入输出掩码是 1 比特的线性偏差个数是 4;
- 3) S 盒代数次数为 3, S 盒逆的代数次数为 3;
- 4) S 盒是置换, 没有固定点;
- 5) S 盒便于采用 bit-slice 实现, 指令数为 15, 具有较高的软件实现效率。

2.3 扩散层设计

在算法扩散层构造方面, 我们希望扩散层便于求逆, 逆向运算与正向运算性能相当, 同时也考虑扩散层与 S 盒的运算资源及实现效率, 产生强的扩散, 要求每个比特采用的异或个数不能超过 2 个。考虑到可逆性, 我们利用 2 分支、4 分支的 Feistel 结构和 MISTY 结构构造扩散层, 通过计算机搜索比较其扩散性能及实现效率, 最后采用 8 个字循环移位和 8 个字异或构成的深度为 4 的 Feistel 结构作为扩散层。该扩散层分支数为 5, 参数选取使分支数达到最优, 在与 S 盒搭配上更具优势。线性层打破 S 盒的结构, 增加了算法的扩散强度。

扩散层基于字混合运算采用变种的 4 分支 Feistel 结构实现, 打破 S 盒的结构, 各分支扩散较均衡, 使得算法可以抵抗碰撞攻击、中间相遇攻击等基于半字节结构的攻击。S 盒与字混合运算的结合, 能够实现算法的快速雪崩效应使得算法能够抵抗差分攻击、线性攻击、不可能差分攻击等密码学攻击方法, 特别是对于比特级的

MILP 自动化搜索方法，增加了差分路线、线性路线等密码特征搜索的复杂度。

分组长度 128 的字混合运算性质如下：

- 1) 算法由四分支 Feistel 结构构成，包括 8 个循环移位，8 个异或，异或深度为 4。
- 2) 扩散层分支数为 5，输入 2 个活性盒时，输出至少为 3 个，或相反；
- 3) 1 个活性 S 盒，最少可以生成 5 个活性盒，最多可以生成 18 个活性盒；逆向最少可以生成 5 个活性盒，最多可以生成 18 个活性盒；
- 4) 2 个活性 S 盒，最少可以生成 3 个活性盒，最多可以生成 29 个活性盒；逆向最少可以生成 3 个活性盒，最多可以生成 28 个活性盒。
- 5) 1 个活性比特两轮可以扩散到所有的 S 盒，2.5 轮扩散到所有的比特。

分组长度 256 的字混合运算性质如下：

- 1) 算法由四分支 Feistel 结构构成，包括 8 个循环移位，8 个异或，异或深度为 4。
- 2) 扩散层分支数为 5；
- 3) 1 个活性 S 盒，最少可以生成 5 个活性盒，最多可以生成 23 个活性盒；逆向最少可以生成 5 个活性盒，最多可以生成 23 个活性盒；
- 4) 2 个活性 S 盒，最少可以生成 3 个活性盒，最多可以生成 41 个活性盒；逆向最少可以生成 3 个活性盒，最多可以生成 64 个活性盒。
- 5) 1 个活性比特三轮可以扩散到所有的比特。

轮函数设计中还综合考虑了扩散层与 S 盒的结合，特别是针对线性偏差汉明重量 1 比特的情况进行考虑，128 比特分组长度保证 3 轮线性逼近路线活性盒最少为 14，3 轮差分路线活性盒最少为 16。

2.4 密钥调度算法

密钥调度算法具有高效的软硬件实现效率，比轮函数简单。密钥调度算法是置换，不存在等价密钥，与算法搭配抗相关密钥、弱密钥攻击等。

密钥调度算法采用简化的轮函数 F ，是可逆的，采用更高运行效率的 S 盒 $S1$ ， $S1$ 最大差分概率为 2^{-2} ，线性逼近概率 2^{-2} ， $S1$ 及其逆的最高代数次数均为 3，无固定点， $S1$ 软件实现仅需要 9 条指令。密钥扩展算法采用简化的字混合运算 $SubNibble_K$ ，密钥扩散与轮函数扩散尽可能不相交，能够有效抵抗相关密钥攻击、弱密钥攻击等。

针对 128/256 版本，算法加解密密钥调度采用相同的逻辑结构，加解密同时实现时，可以节省面积。

3 安全性分析

本章对我们设计的分组密码进行了自评估，主要考虑了差分分析、回轮攻击、线性分析、不可能差分分析、积分攻击和相关密钥分析等。

3.1 分组长度 128 的算法版本的安全性分析

3.1.1 差分类分析

差分密码分析是迭代密码最有效的攻击方法之一，该方法由 Biham 和 Shamir 在 1991 年提出，是一种选择明文攻击。其基本思想是通过分析明文对的异或差分对密文对的异或差分的影响来恢复某些密钥比特。

回轮 (Boomerang) 攻击，又称飞去来器攻击，是一种差分类的分析方法，最初由 Wagner 在 1999 年提出并用于分组密码的分析。它是一种自适应的选择明密文的攻击。在 FSE 2000 上，Kelsey 等进一步将该方法发展成选择明文的攻击，称为加强的回轮攻击 (Amplified Boomerang Attack)。2001 年，Biham 也独立的提出了回

轮攻击的变种——矩形（Rectangle）攻击。对于差分类攻击方法，关键是寻找具有概率优势的较长的差分特征，并构造攻击区分器。

FESH-128 的线性层 MixWord 的差分最小分支数为 5，分支为 2 个活跃盒到 3 个，3 个活跃盒到 2 个。

三轮差分路线最小活跃盒数为 16，分别为 $5 \rightarrow 1 \rightarrow 10$ 、 $11 \rightarrow 2 \rightarrow 3$ 和 $8 \rightarrow 1 \rightarrow 7$ 。那么 $3 \times 4 = 12$ 轮即达到 $16 \times 4 = 64$ 个差分活跃盒，实际上这样的路线并不存在。

四轮差分路线最小活性 S 盒个数的估计分为以下三个步骤：1) 首先，对于第 i 轮中差分活跃 S 盒个数为 1 或 2 的情形，进行遍历。例如，限制第 2 轮中只含有 1 个差分活跃 S 盒时，4 轮差分路线的活跃 S 盒个数最小为 29，模式形如 5-1-12-11。该类情况活跃 S 盒数大于等于 25。

2) 对相邻两轮活跃 S 盒数小于等于 8 的所有情形，首先计算出所有可能的两轮差分模式，然后将这些模式分别代入到 4 轮线性规划模型中，分别证明了这些模式无法产生小于等于 24 活跃 S 盒的差分路线。

3) 两轮前向和两轮后向路线变量。首先设置第 1 轮的活性 S 盒为 3、4、5 或 6，对第一轮输出活性 S 盒个数进行约束。由于第 2) 排除了连续两轮活跃 S 盒个数小于等于 8 的情况，因此本步中后两轮活跃 S 盒个数大于 8。因此，当第一轮输入为 3、4 时，输出活跃 S 盒只保留小于等于 12 的情况。当输入 5、6 个活跃 S 盒时，只保留输出小于等于 10 的情况保留。类似的，从第四轮向前推，前两轮需大于 8 个活跃 S 盒，同样第四轮遍历活性 S 盒为 3、4、5 或 6 的情形，以及采用类似排除不可能情形。总之，对总活跃 S 盒个数小于等于 24 的进行对接试验，未发现成功对接的路线。

经过上述搜索和评估，计算得出 4 轮差分路线中至少有 25 个活跃 S 盒。根据四轮和三轮最小活跃 S 盒个数的估计，我们证明 $(4+4+3=) 11$ 轮 FESH-128 差分最小活跃盒个数至少为 $(25+25+16=) 66$ 。

给出了概率为 2^{-116} 的 5 轮差分路线，见表 3-1，共有 48 个活跃 S 盒，

12→20→5→1→10。3 轮路线概率为（11→2→3、5→1→10）概率均小于等于 2^{-32} 。

利用 3 轮和 2 轮的差分路线构造 Boomerang 区分器，可构造 3+1+2=6 轮区分器，前后扩展 1 轮，最多给出 8 轮 FESH-128-128 的分析。利用相同的路线，可以攻击 9 轮 FESH-128-256。

表 3-1 FESH-128 五轮差分路线

	X[0]	X[1]	X[2]	X[3]	Pr= $2^{\{-116\}}$
1 轮 S_in	0x00040000	0x04209451	0x02248408	0x02000080	
1 轮 S_out	0x2041041	0xc9	0x6209459	0x60000d1	$2^{\{-25\}}$
1 轮 L_out	0x209a9219	0x94c98204	0x36c33019	0x3ed2221d	
2 轮 S_out	0x8a9b1001	0x98908008	0x813205	0x26488014	$2^{\{-58\}}$
2 轮 L_out	0x98000	0x88001	0x12001	0x2000	
3 轮 S_out	0x18000	0x80000	0x1	0x2000	$2^{\{-11\}}$
3 轮 L_out	0x0	0x0	0x1	0x0	
4 轮 S_out	0x1	0x1	0x0	0x0	$2^{\{-2\}}$
4 轮 L_out	0x20000001	0x26000013	0x2400000	0x980000	
5 轮 S_out	0x00000000	0x20d80001	0x06400012	0x04980012	$2^{\{-20\}}$
5 轮 L_out	0x6db60e80	0x413ae682	0x1ddbff47	0x04814b9a	

在 5 轮差分路线的尾部扩展一轮攻击 6 轮，最后一轮，产生 29 个差分活跃盒。数据和时间复杂度约为 2^{118} 。下扩 2 轮，攻击 7 轮，复杂度约为 2^{246} 。

如果在 5 轮差分路线头部添加一轮，攻击 6 轮，(0x00040000, 0x04209451, 0x02248408, 0x02000080) 经过 L^{-1} 得到差分 (0xfa8415bc, 0xe420a575, 0x32909d8c, 0xb33da7d7)，共 29 个活性盒，攻击复杂度与上面攻击相当。

3.1.2 线性分析

线性密码分析是 Matsui 在 1993 年针对 DES 算法提出的一种有效的密码分析方法，是已知明文攻击。攻击者对获取的一些明密文对，利用明文、密文和密钥之间的线性逼近来恢复密钥，关键是寻找多轮有效线性逼近。

FESH-128 的线性层 MixWord 的线性最小分支数为 5，3 个活跃 S 盒可以到 2 个活跃 S 盒，2 个活跃 S 盒也可以到 3 个。三轮线性路线最小活跃 S 盒个数为 14，

包括 $5 \rightarrow 1 \rightarrow 8$ 和 $8 \rightarrow 1 \rightarrow 5$ 。对四轮线性路线进行与四轮差分路线类似的分析，计算得出四轮线性路线中至少有 25 个活跃 S 盒。根据四轮和三轮最小活跃 S 盒个数的估计，我们证明 $(4+4+3=)11$ 轮 FESH-128 线性最小活跃盒个数至少为 $(25+25+14=)$ 64。

给出五轮线性路线偏差为 2^{-62} ，共 46 个活性 S 盒，每轮活跃盒个数分别是：
 $12 \rightarrow 20 \rightarrow 5 \rightarrow 1 \rightarrow 8$ ，见表 3-2。

表 3-2 FESH-128 五轮线性路线

	X[0]	X[1]	X[2]	X[3]	Pr= $2^{\{-108+46\}}$
0 轮 L 输入	0x701e3558	0x8022bb44	0x1a2464ae	0x8c111254	
1 轮 S_in	0x00110400	0x04110001	0x80000400	0x88206210	0
1 轮 S_out	0x106200	0x8c204401	0x4014411	0x8206610	$2^{\{-24\}}$
1 轮 L_out	0x1840040d	0x9255d400	0x9a0134a6	0x241a0e4	
2 轮 S_out	0x90401409	0x9201a0c5	0xa40802a	0x544048	$2^{\{-52\}}$
2 轮 L_out	0x1	0x10000001	0x28008	0x8	
3 轮 S_out	0x1	0x8	0x28000	0x10000000	$2^{\{-13\}}$
3 轮 L_out	0x1	0x0	0x0	0x0	
4 轮 S_out	0x0	0x0	0x0	0x1	$2^{\{-3\}}$
4 轮 L_out	0x4000000	0x2800	0x80000	0x400000a1	
5 轮 S_out	0x4080000	0x4002800	0x400828a1	0x400800a1	$2^{\{-16\}}$
5 轮 L_out	0x075e3e20	0xc5110229	0x40022479	0xf65b4e81	

将该 5 轮线性路线尾部添加一轮，最后一轮共有 25 个线性活跃盒。将该 5 轮线性路线头部添加一轮，第一轮活跃盒数 28， $(0x00110400, 0x04110001, 0x80000400, 0x88206210)$ 经过 L^{-1} 得到 $(0x701e3558, 0x8022bb44, 0x1a2464ae, 0x8c111254)$ 。可攻击 6 轮 FESH-128-128 或 7 轮 FESH-128-256。

3.1.3 不可能差分分析

不可能差分分析是差分分析的一个变种，是由 Knudsen 与 Biham 等分别独立提出。Knudsen 在研究 DEAL 算法的安全性时发现，如果 Feistel 结构密码的轮函数是双射，则算法自然存在 5 轮的不可能差分，从而对 6 轮的密码算法存在攻击；在 1999 年欧密会上，Biham 等在研究 Skipjack 算法安全性时提出不可能差分的概念，并在

FSE 1999 上系统地讲述了如何采用“中间不可能相遇”的方法寻找不可能差分。

不可能差分分析与经典差分分析方法的思想相反。传统的差分分析方法利用高概率差分来恢复密钥，而不可能差分是用概率为 0 的差分（不可能差分）恢复密钥。其基本思想是正确的密钥下的明密文不可能导致这样的差分，错误的密钥下以随机概率满足这样的不可能差分，则我们可以排除那些导致概率为 0 的差分对应的密钥。

我们采用 Sasaki 等、Cui 等提出的基于 MILP 的不可能差分路线搜索方法，找到了 FESH-128 的 4.5 轮不可能差分路线，即头尾各一个 S 盒为活跃盒，是不可能差分路线，如：

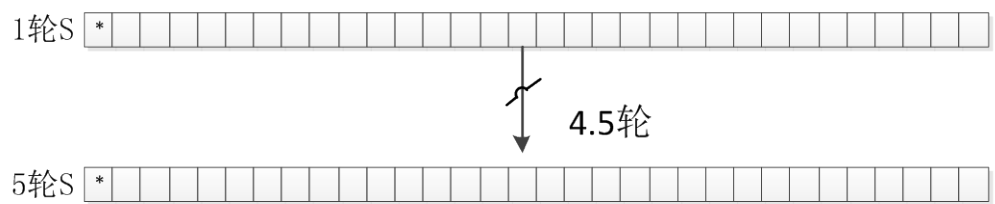


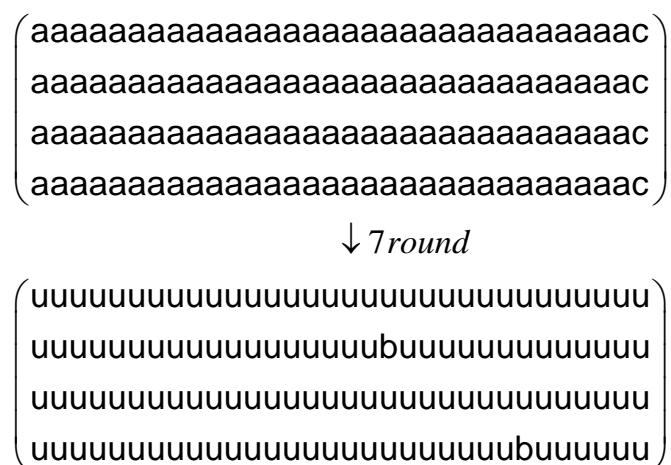
图 3-1 FESH-128 的 4.5 轮不可能差分路线

向上向下各扩展一轮，各产生 18 个活跃盒，尾部的线性层对分析影响不大，可以转换为等价密钥。基于该路线，可以分析 7 轮 FESH-128-128 和 8 轮 FESH-128-256。

3.1.4 积分攻击

积分攻击是一种选择明文的攻击，最先用于 Square 分组密码分析，也叫 Square 攻击，其基本思想是通过寻找一些中间状态的“零和”来恢复密钥。由于在攻击中需要求和，被称之为积分攻击。积分攻击被广泛运用于分组密码，尤其是基于 SPN 结构的分组密码分析中。

我们利用 Xiang 等提出的基于 MILP 自动化搜索积分路线的方法，找到了 7 轮积分路线，输入 124 个活跃比特，即前 31 个 S 盒的比特为遍历，输出两个平衡比特，如图 3-2，其中 a 表示遍历，c 表示常量，u 表示未知，b 表示平衡。



利用该 7 轮积分路线, 向下扩展 2 轮可以攻击 9 轮 FESH-128-128, 恢复两比特密钥。复杂度约为 2^{124} , 也可以用于攻击 10 轮 FESH-128-256。

1992 年和 1993 年，Knudsen 和 Biham 各自独立地提出了相关密钥攻击。相关密钥攻击反映了密钥扩展算法对分组密码安全性的影响，在这种攻击下，攻击者不知道密钥，但是他可以选择与当前密钥相关的其他密钥，可以选择密钥之间的关系，利用分组密码密钥扩展算法的弱点，通过选择合适的、与当前密钥相关的密钥，寻找原有密钥与新的相关密钥下分别对应的加密算法之间的关联，就有可能恢复出密钥。

对于 FESH-128-256, 由于密钥量是 FESH-128-128 两倍, 考虑密钥关系, 可以将 FESH-128-128 的三轮相关密钥差分路线扩展到 4 轮。利用 4 轮加 1 轮加 4 轮, 构造 9 轮回轮区分器, 密钥恢复时, 最多上下各增加 2 轮, 可以攻击至多 13 轮

FESH-128-256。

表 3-3 4 轮 FESH-128-128 相关密钥差分路线

状态差分					密钥差分			
0x130000	0x26000300	0x3022	0x98000400					
0x0	0x0	0x0	0x0		0x130000	0x26000300	0x3022	0x98000400
S								
0x0	0x0	0x0	0x0		0x3522	0x132200	0x98000400	0x26002700
L								
0x400	0x2600	0x2600	0x2200		0x400	0x2600	0x2600	0x2200
S								
0x600	0x2000	0x0	0x0		0x0	0x2600	0x0	0x0
L								
0x200	0x0	0x4	0x0		0x0	0x26	0x0	0x3500
S								
0x200	0x4	0x204	0x0		0x0	0x0	0x3500	0x526
L								
0x850a0200	0x820204c	0x10a1e10	0x208001		0x21983500	0x0	0x866	0x0
S								
0x850a0200	0x9c11	0x8003c5c	0x8c20a04d					

3.1.6 小结

基于字的线性变换 MixWord 打破了 S 盒结构，但是在半字节分配上具有不均匀性，我们重点考虑差分类和线性类分析，给出了差分分析、回轮攻击、线性分析、不可能差分分析、积分攻击和相关密钥分析，FESH-128-128/256 的分析结果分别见表 3-4 和 3-5。

表 3-4 FESH-128-128 的分析结果汇总

攻击类型	区分器轮数	分析轮数
差分分析	5 轮	6 轮

回轮攻击	6 轮	7 轮
线性分析	5 轮	6 轮
不可能差分	4.5 轮	7 轮
积分攻击	7 轮	9 轮
相关密钥回轮攻击	7 轮	9 轮

表 3-5 FESH-128-256 的分析结果汇总

攻击类型	区分器轮数	分析轮数
差分分析	5 轮	7 轮
回轮攻击	6 轮	8 轮
线性分析	5 轮	7 轮
不可能差分	4.5 轮	8 轮
积分攻击	7 轮	10 轮
相关密钥回轮攻击	9 轮	13 轮

3.2 分组长度 256 的算法版本安全性分析

3.2.1 差分分析

FESH-256 的线性层 MixWord 的差分最小分支数为 5，分支为 2 个活跃盒到 3 个，3 个活跃盒到 2 个。类似于 FESH-128，我们计算出 4 轮 FESH-256 差分最小活跃盒个数为 32，因此 16 轮 FESH-256 差分最小活跃盒个数至少为 128 个。

给出了概率为 2^{-174} 的 5 轮差分路线，见表 3-6，共有 79 个活跃 S 盒， $28 \rightarrow 5 \rightarrow 1 \rightarrow 12 \rightarrow 33$ 。基于 5 轮差分路线，向前扩头部会产生 59 个活跃 S 盒；向尾部扩，会产生 60 个活跃盒，可以分析 6 轮的 FESH-256-256 和 7 轮的 FESH-256-512。

3 轮路线概率为 $(5 \rightarrow 1 \rightarrow 12)$ 概率为小于等于 2^{-36} 。利用两条 3 轮的差分路线构

造 Boomerang 区分器，可构造 $3+1+3=7$ 轮区分器，前后各扩展 1 轮，最多给出 9 轮 FESH-256-256 的分析；对于 FESH-256-512，采用相同的路线，在 9 轮的分析结果上至多增加 2 轮，给出 11 轮的分析结果。

表 3-6 FESH-256 五轮差分路线

	X[0]	X[1]	X[2]	X[3]	$Pr=2^{\{-17, -4\}}$
0 轮 L_in	0x1c3c126c3b7abe76	0x0ea5e80cf0e41e12	0xc207490342268921	0x26cd62c3864ed6b2	
1 轮 S_in	0x0400088008282001	0x0000200449ca4041	0x0706b80243a20068	0x0384100200000028	
1 轮 S_out	0x0402888402202000	0x008280000a080001	0x0000200441c24040	0x0384908200406028	$2^{\{-57\}}$
1 轮 L_out	0x00000000020a0000	0x0000000202080001	0x0000000000020001	0x0000000200000000	
2 轮 S_out	0x00000000000a0000	0x0000000002000000	0x0000000000000001	0x0000000200000000	$2^{\{-12\}}$
2 轮 L_out	0x0000000000000000	0x0000000000000000	0x0000000000000001	0x0000000000000000	
3 轮 S_out	0x0000000000000001	0x0000000000000001	0x0000000000000000	0x0000000000000000	$2^{\{-2\}}$
3 轮 L_out	0x0400000000000001	0x20a0000000000105	0x8000000000000000	0x0010500000000000	
4 轮 S_out	0xa4205000000000124	0xa0105000000000125	0x04a010000000000000	0x009040000000000000	$2^{\{-30\}}$
4 轮 L_out	0x12a0344080000a21	0x0261103080412132	0x5186181012810422	0x1281300018402000	
5 轮 S_out	0x0000044000000a01	0x412608008a800420	0x4346083002810512	0x10c1346018402b11	$2^{\{-73\}}$
5 轮 L_out	0x6876e252e5cadb91	0x77ee567238a8e6c8	0x661c3ee5864201dd	0xb4992a1f384c1179	

3.2.2 线性分析

FESH-256 的线性层 MixWord 的线性最小分支数为 5，3 个活跃 S 盒可以到 2 个活跃 S 盒，2 个活跃 S 盒也可以到 3 个。类似于 FESH-128，我们计算出 4 轮 FESH-256 线性最小活跃盒个数为 32，因此 16 轮 FESH-256 线性最小活跃盒个数至少为 128 个。

给出六轮线性路线偏差为 2^{-123} ，共 99 个活性 S 盒，每轮活跃 S 盒个数分别是：
32→8→1→5→25→28，见表 3-7。

表 3-7 FESH-256 六轮线性路线

	X[0]	X[1]	X[2]	X[3]	Pr= $2^{\{-221+98\}}$
0 轮 L_in	0x7e9676ddbc8afce3	0xb18d7af22873c8d7	0x997f5a804f2a1cb4	0xfc041cb5b2ecc3c2	
1 轮 S_in	0x0442621121913002	0x04426a11a1913000	0x580080200220022e	0xdc02a03022a8022d	0
1 轮 S_out	0x9800022002002008	0x4000880080280226	0x184048218311100b	0x8402201020880003	$2^{\{-64\}}$
1 轮 L_out	0x8000020000000000	0x0000000000000000	0x0800040040000103	0x0000040000000002	
2 轮 S_out	0x0800000000000000	0x0000000000000002	0x0000040040000101	0x8000020000000000	$2^{\{-23\}}$
2 轮 L_out	0x0000000000000000	0x0000000000000000	0x0000000000000001	0x0000000000000000	
3 轮 S_out	0x0000000000000000	0x0000000000000000	0x0000000000000001	0x0000000000000000	$2^{\{-3\}}$
3 轮 L_out	0x0800000000000000	0x0000000000000008	0x0000000000000001	0x8000000000000000	
4 轮 S_out	0x0000000000000000	0x0000000000000000	0x8000000000000000	0x8800000000000008	$2^{\{-13\}}$
4 轮 L_out	0x0c00201002200020	0x1002208060048020	0x8004020044000401	0x580a001062380440	
5 轮 S_out	0xd008208062148020	0x180a2000221c0460	0x1c0e021026340001	0x4c0c209000388060	$2^{\{-62\}}$

利用该 8 轮积分路线，向下扩展 2 轮可以攻击 10 轮 FESH-256-256，恢复两比特密钥。复杂度约为 2^{252} ；对于 FESH-256-512，路线一样，可多分析一轮。

3.2.5 相关密钥分析

找到了 FESH-256-256 的三轮相关密钥差分路线，见表 3-8， $0 \rightarrow 1 \rightarrow 5$ ，概率为 2^{-12} 。可以用该路线构造 3 轮加 1 轮加 3 轮的回轮区分器，并攻击 9 轮 FESH-256-256。给出了 4 轮 FESH-256-256 相关密钥差分路线（见表 3-9）： $0 \rightarrow 1 \rightarrow 12 \rightarrow 37$ ，概率为 2^{-116} 。

对于 FESH-256-512，由于密钥量是 FESH-256-256 两倍，考虑密钥关系，可以将 FESH-256-256 的三轮相关密钥差分路线，扩展到 4 轮。利用 4 轮加 1 轮加 4 轮，构造 9 轮回轮区分器，最多上下各添加 2 轮，可以攻击 13 轮 FESH-256-512。

表 3-8 3 轮 FESH-256-256 相关密钥差分路线

状态差分				密钥差分			
0x400000 00000000 0	0x40000 0400010 0	0x800000 00000020 0	0x100 00010 0				
0x0	0x0	0x0	0x0	0x400000 00000000 0	0x400000 4000100	0x800000 00000020 0	0x10000 0100
S							
0x0	0x0	0x0	0x0	0x800000 00000030 0	0x400000 00000010 0	0x100000 100	0x40000 0400010 0
L							
0x100	0x100	0x100	0x100	0x100	0x100	0x100	0x100
S							
0x0	0x0	0x0	0x100	0x0	0x0	0x0	0x100
L							
0x400000 0	0x20000	0x800002 00	0x100	0x100000 000	0x0	0x100	0x0
S							
0x400000 0	0x80000 300	0x800202 00	0x402 0100				

表 3-9 4 轮 FESH-256-256 相关密钥差分路线

状态差分					密钥差分			
0x800000	0x0	0x1000000	0x0					
0x0	0x0	0x0	0x0		0x800000	0x0	0x1000000	0x0
S								
0x0	0x0	0x0	0x0		0x1000000	0x800000	0x0	0x0
L								
0x0	0x0	0x0	0x800000		0x0	0x0	0x0	0x800000
S								
0x800000	0x800000	0x0	0x0		0x0	0x0	0x800000	0x0
L								
0x820000	0x8290500	0x10400000	0x828		0x800002	0x0	0x20000000000	0x0
S								
0x92d25008	0x92500828	0x5000	0x2020820		0x200000000000	0x800002	0x0	0x0
L								
0x20000808101b102a	0x104000941f51488c	0x410207664504	0x2082000020b08b4		0x0	0x20001000004	0x0	0x10000008800402
S								
0x2000080800001002	0x208610210340518	0x104041960d654500	0x3248289c0a0a18b6					

3.2.6 小结

针对 FESH-256-256/512，我们重点考虑差分类和线性类分析，给出了差分分析、回轮攻击、线性分析、不可能差分分析、积分攻击等分析结果，FESH-256-256/512 的评估结果分别见表 3-10 和 3-11。

表 3-10 FESH-256-256 的分析结果汇总

攻击类型	区分器轮数	分析轮数
差分分析	5 轮	6 轮
回轮攻击	7 轮	9 轮
线性分析	6 轮	8 轮
不可能差分	4.5 轮	7 轮
积分攻击	8 轮	10 轮
相关密钥攻击	7 轮	9 轮

表 3-11 FESH-256-512 的分析结果汇总

攻击类型	区分器轮数	分析轮数
差分分析	5 轮	7 轮
回轮攻击	7 轮	11 轮
线性分析	6 轮	9 轮
不可能差分	4.5 轮	8 轮
积分攻击	8 轮	11 轮
相关密钥攻击	9 轮	13 轮

4 性能分析

4.1 软件性能分析

4.1.1 算法软件运行平台

Intel 酷睿 i7 处理器，16GB 内存，Windows 7 64 位操作系统，Microsoft Visual Studio 2017 Professional Edition。

本算法便于软件实现，S 盒可以采用 bit-slice 技术实现，实现的过程中只需要保留 n 比特的状态， n 或者 $2n$ 比特的密钥，以及生成 $N*n$ 比特的轮密钥。

4.1.2 CBC 模式下算法性能分析

分组密码算法 FESH-128-128、FESH-128-192、FESH-128-256 性能测试采用 256B 数据作为输入数据，采用 CBC 模式加密运行 100000 次，求平均值。算法性能见表 4-1。

表 4-1 CBC 模式下 FESH 算法性能（单位：Mbps）

算法	加密 256B	解密 256B
FESH-128-128	1215	1001
FESH-128-256	962	845
FESH-256-256	1228	1190

4.1.3 ECB 模式下算法性能分析

算法 FESH-128-128、FESH-128-256、FESH-256-256 的性能测试包括 256B、2MB 长度数据流下算法加解密速度（单位 Mbps），其中 256B 运行 1000000 次，2MB 运行 10 次，求平均值。

本算法采用标准 C 实现，算法性能如表 4-2。由运行结果可知，加密 2MB 长度

数据，FESH-128-128 算法加密速度为 1.4Gbps，约为 AES 的 93%。FESH-128-256 算法加密速度为 1.1Gbps，比 AES-256 稍快，性能比较见表 4-2。

表 4-2 FESH 算法性能与 AES、SM4 比较（单位：Mbps）

算法	加密 256B	加密 2MB	解密 256B	解密 2MB
AES-128	1368	1509	1406	1553
AES-192	1178	1259	1198	1311
AES-256	1007	1081	1015	1134
SM4	902	963	933	987
FESH-128-128	1318	1415	1195	1221
FESH-128-256	1066	1142	953	981
FESH-256-256	1302	1391	1293	1290

注：AES 和 SM4 算法都采用的是优化的查大表的实现方式

本算法便于并行实现，国际上分组密码算法的工作模式中，ECB、CTR 和磁盘加密的 XTS 模式中都支持分组密码算法并行处理，我们的算法在这些模式下具有高效率。以 ECB 模式为代表，我们给出了算法并行处理的性能分析。我们采用 SSE2 指令和 AVX2 指令实现了算法的并行处理。算法性能见表 4-2。由运行结果可知，FESH-128-128 算法加密速度可达 2.6Gbps，FESH-256-256 算法加密速度可达 3.7Gbps。

表 4-3 FESH 算法并行实现性能（单位：Mbps）

算法	加密 256B	加密 2MB	解密 256B	解密 2MB
FESH-128-128	2494	2622	2392	2539
FESH-128-256	2104	2253	2007	2162
FESH-256-256	3041	3703	2991	3603

4.2 ARM 平台下算法性能分析

选用的 ARM32 平台为基于 STM32 F103 的开发板(主频 72Mhz):stm32f1zet6 核心板 6、512K 片内 Flash、64K 片内 RAM。分组密码算法 FESH-128-128、FESH-128-192、FESH-128-256 性能测试包括 256B、采用 CBC 模式加密 256B 运行 100000 次,求平均值。在 ARM32 平台下实现性能见表 4-4。

表 4-4 FESH 算法 ARM 平台下的运行效率(单位: Mbps)

算法	加密 256B	解密 256B
FESH-128-128	8.871	8.334
FESH-128-256	7.009	6.972
FESH-256-256	4.052	4.273

4.3 硬件性能分析

我们采用规定的接口对算法进行硬件实现。函数接口为:

```
crypt_ecb(input clk, input start, input rst, input mode, input [4:0] blocknum, input [127:0] textin, input [127:0] key, output [127:0] textout, output enable, output done);
```

算法运行周期为 N 轮,在实现中增加了 2 个周期控制信号实现 32 个分组的加解密,因此算法运行周期按照 N+2 计算。在 ModelSim SE 下仿真,加密 32 个分组数据,再解密 32 个分组数据,结果全部正确。

我们利用 Synopsys Design Compiler B 在两种 ASIC 平台下,对算法的三个版本进行了综合。采用中芯国际 130nm 工艺库,算法三个版本加解密效率见表 4-5, FESH-128/128 加密效率为 4.18Gbps, FESH-256/256 加密效率为 5.96Gbps。采用算法规定的库文件实现效率见表 4-6。

表 4-5 FESH 算法硬件实现效率（工艺元件库是中芯国际 130nm_ulvt）

	自测试结果		
	128/128	128/256	256/256
运算周期	1170	1430	1690
时钟约束	1.7ns	1.65ns	1.65ns
加密速率	4183Mbps	3526Mbps	5967Mbps
解密速率	4056Mbps	3419Mbps	5786Mbps
面积	54085.95um ²	59261um ²	58899um ²
加密吞面比	0.077339985	0.05950238	0.10131523
解密吞面比	0.074996349	0.05769928	0.09824507

其中运算周期包括加密周期与解密周期，解密需额外一个分组生成解密密钥，计算如下：128/128：18*32+18*(32+1) = 1170，128/256：22*32+22*(32+1) = 1430，256/256：26*32+26*(32+1) = 1690。

5 优缺点声明

1) 算法理论创新性

本算法采用分组密码设计广泛采用的替换-置换网络（SPN）结构构建，构造方案简洁便于实现，同时算法兼顾了安全性、软硬件实现性能。算法对 128 比特的分组和 256 比特的分组，采用统一的结构，都可以转化为基于字的基本逻辑运算，这种实现方式在现代 Intel CPU 架构下可以很方便的结合扩展指令 SSE 和 AVX 实现分组密码的并行处理，支持 ECB、CTR 和 XTS 等模式下分组密码的高效实现，同时

算法采用基于字的实现代替查表运算，在抗侧信道攻击具有优势。本算法便于软硬件实现，支持多种密钥长度包括 128 比特、192 比特、256 比特、384 比特和 512 比特，包含了国际上主流的分组长度和密钥长度，可以应用于各种安全协议中，具有兼容性。

算法的轮函数非线性层采用 4 比特的 S 盒实现混淆，S 盒安全强度高，不仅考虑差分概率达最优，线性偏差概率达最优，代数次数为 3，还避免了输入输出汉明重量为 1 比特的差分情况，同时输入输出汉明重量为 1 比特的线性掩码数量仅为 4。S 盒在满足安全性的同时，具有并行实现性，15 条指令可以实现。

算法扩散层，通过计算机搜索比较各种扩散层的扩散程度以及运算效率，我们采用 4 分支的 Feistel 结构，包括 8 个字循环移位和 8 个字异或运算，深度为 4。该扩散层分支数为 5，参数选取使分支数达到最优，在与 S 盒搭配上更具优势。线性层打破 S 盒的结构，增加了算法的扩散强度，支持 128 和 256 分组长度，结构统一，并且在两个版本上都具有良好的扩散性：3 轮实现全扩散。

S 盒和字混合运算是轮函数的两个重要构成部分，参数选取的时候充分考虑了 S 盒扩散情况，使 S 盒快速传播，产生强雪崩效用，抵抗差分、线性和积分等国际上的分组密码攻击方法，128 分组长度的 3 轮差分最小活性盒个数为 16，3 轮线性最小活性盒个数为 14。

本算法密钥调度算法基于简化的轮函数实现，密钥调度算法扩散层设计考虑轮函数的扩散情况，抗弱密钥攻击、相关密钥攻击。针对 FESH-128-256 版本，加解密采用相同的逻辑，硬件实现节省面积。

2) 算法的特点

本算法的轮函数与 Serpent 的轮函数结构相似，采用 4 比特的 S 盒和字混合运算，不过我们采用的 S 盒安全强度更高，特别是低汉明重量的差分分布情况，以及低汉明重量的线性掩码分布都要优于 Serpent 的 S 盒，轮函数中的 S 盒不存在输入输出汉明重量都是 1 的差分，而汉明重量为 1 的输入和输出的线性掩码仅有 4 个。

而且 S 盒给出的 bit-slice 实现指令只有 15 条，也是少于 Serpent 的 S 盒的实现指令数。我们采用的字混合运算基于 4 分支 Feistel 结构变种，各分支扩散比较均衡，而且最小分支数为 5，大于 Serpent 的最小分支数。因此我们的轮函数比 Serpent 具有更强的抗差分和线性攻击的能力，目前我们发现分组长度 128 的 5 轮的差分路线和 5 轮的线性路线概率都比较接近于临界值。

我们对算法进行了自评估，重点针对差分类和线性类分析，给出了差分分析、回轮攻击、线性分析、不可能差分分析、积分攻击和相关密钥分析。在分析的过程中，我们结合了最新的一些分析方法以及基于 MILP 的自动化搜索等。我们的分析结果显示：在单密钥下 FESH-128-128 最多可以分析到 9 轮，FESH-128-256 可以分析到 10 轮，相关密钥下 FESH-128-128 可以分析到 9 轮，FESH-128-256 可以分析到 13 轮，这两个版本的轮数分别为 16 和 24，具有较大的安全冗余；在单密钥下 FESH-256-256 最多可以分析到 10 轮，相关密钥下 FESH-256-256 可以分析到 9 轮，该版本的轮数 24，也具有很大的安全冗余。

本算法整体实现可以采用“与”、“或”、“非”、“异或”和“循环移位”这些基本操作构成，因此算法实现具有很强的灵活性，适用于各种平台上实现，一般的 PC 机、嵌入式平台 ARM，单片机 51 等，以及基于硬件的 FPGA 平台和 ASIC 平台等。

在 PC 机上，本算法软件实现与国际标准算法 AES 的查大表的优化实现效率相当，效率优于我国标准 SM4 的查大表优化实现。特别是算法轮函数都是基于字实现的，便于并行处理，使用扩展指令集，FESH-128-128 加密的速度可达 2.6Gbps，FESH-256-256 加密的速度高达 3.7Gbps。在分组密码的工作模式中，ECB、CTR 和磁盘加密的 XTS 模式中都支持分组密码算法并行处理，我们的算法在这些应用下具有高效率。我们的算法在中芯国际 130nm 工艺库上实现，在最大吞面比的约束下，FESH-128/128 加密效率为 4.18Gbps，FESH-256/256 加密效率为 5.96Gbps。本算法在实现的过程中考虑到算法面积要小，延时小，因此采用 4 比特的 S 盒，并对扩散层异或的个数和异或深度进行综合考虑。FESH-128/128 算法实现 1.7ns 延时，加密需要 16 轮总共的延时为 27.2ns。本算法在低端 FPGA 上实现也具有较好的效率。

本算法可以在一些低端设备，如微控制器上有效实现，这些微控制器随机存取存储器（RAM）和只读存储器（ROM）的数量可能非常有限。算法运算只有简单的逻辑运算，密钥调度可以采用 on-the-fly 模式进行实现，不需要额外占用内存。FESH-128 的轮函数中的移位常数，选取的时候，尽可能选择 8 的倍数或者 4 的倍数，或者在此基础上加 1 或者减 1，这样在 8 位机上便于实现，需要的指令数较少。我们的算法逻辑运算在 8 位机上也将会有比较好的性能。

本算法基于 SPN 结构构建，128 比特分组划分为 4 个 32 比特字，256 比特分组划分为 4 个 64 比特字，两个版本采用相同的 S 盒实现混淆层，线性层都是基于字的 4 分支 Feistel 结构实现，方案统一，简洁优美。S 盒可以基于 bit-slice 实现，因此算法便于进行侧信道防护，根据 Begul 等关于 4*4S 盒的分析，我们的 S 盒需要 3share。此外，采用指令实现不需要查 S 盒也有利于抵抗 cache 攻击等。