



电力轻量级分组密码算法

王涛

(全球能源互联网研究院, 北京 102209)

摘要:随着计算机技术的发展,密码算法在信息安全领域应用越来越广泛。然而,如何实现高性能、高安全性的加密算法并更加方便地应用在软、硬件环境上,一直是亟待解决的难题。未来电网环境下,研究电力行业专用的轻量级分组密码算法,可以摆脱以往只是使用通用算法的现状,使得电网信息安全更加自主可控。因此,针对电力行业核心业务安全需求,研究性能更优、安全性更高、应用更加方便的轻量级电力专用分组密码算法的实现。

关键词:密码技术;分组算法;轻量级分组密码

中图分类号:TP312

文献标识码:A

doi: 10.11959/j.issn.1000-0801.2015392

Light block cipher algorithm for electric power industry

WANG Tao

Global Energy Interconnection Research Institute, Beijing 102209, China

Abstract: With the development of computer technology, cryptographic algorithms were applied more and more widely in the field of information security. However, it has been a very important problem that how to achieve high performance, high security encryption algorithm and more convenient application in the software and hardware environment. In future grid environment, study of the electric power industry special lightweight block cipher algorithm can get rid of the actual state that only using general algorithm, also makes the information and the safety of the power system more self controlled. According to the electric power industry central business security requirements, the lightweight block cipher algorithm of special power of better performance, higher safety and more convenient application were realized.

Key words: cryptographic technique, block cipher algorithm, light block cipher

1 引言

本文设计的 Electricity(电力轻量级分组密码)算法是一个迭代型的分组密码,整体结构为 SP(substitution-permutation, 代换、置换)网络,分组长度为 64 bit,密钥长度有 64 bit, 80 bit 和 128 bit 共计 3 种可选值,对应的轮数分别为 16 轮、18 轮和 20 轮。电力轻量级分组密码是一个硬件实现与软件

实现均有极佳性能,且安全性可靠的轻量级分组密码算法。

相较 LED 密码(一种轻型分组密码算法),选用了硬件实现面积更优的 S 盒和列混淆变换。这使得在相同实验环境和实验策略下,电力轻量级分组密码更为轻量,即硬件面积需求更低。在一定硬件环境下,Electricity-80 可以在 1 000 GE 以下实现。



在很多轻量级密码的应用中, 密钥需要固化到硬件中。此外, 轻量级密码的一些应用也需要同一密码算法具有较好的软件实现性能, 而不仅仅是低的硬件面积。鉴于上述应用需求, 通过设计密钥扩展算法来保护算法, 以抵抗相关密钥攻击。这样的改变, 在保证适度安全强度的前提下, 只以较小的硬件面积增加 (即密钥扩展算法的额外开销) 来换取软件速度的提升和硬件性能的提高, 使电力专用算法兼有更好的软件和硬件实现性能。

通过分析电力专用算法针对一些重要分析方法的安全性, 得出电力专用算法具有充足的安全强度, 能够抵抗现有的已知安全性分析方法。

2 算法规范

Electricity 是迭代型的分组密码, 其为 Rijndael-like (Rijndael 算法是美国国家标准与技术协会即 NIST 所选的高级加密标准 (AES) 的候选算法) 的 SP 网络结构, 分组长度为 64 bit, 密钥长度有 3 种可选值: 64 bit、80 bit、128 bit, 3 种密钥长度对应的轮数分别为 16 轮、18 轮与 20 轮。

算法将 64 bit 明文分组载入中间状态寄存器, 表示为 $M=m_0||m_1||\dots||m_{63}$ 。以下分别介绍加密过程与解密过程所用的有关结构与数据。

2.1 加密过程

在进行后续加密操作前, 首先用轮密钥 K_0 对其进行一次轮密钥加运算, 然后进行 R (R 是对 3 种不同密钥长度的变体, R 分别为 16、18 与 20) 轮加密运算, 最后将寄存器内的值作为密文分组输出。算法的每一轮分为 4 步: nibble 代换 s 、行移位 r 、列混淆 ρ 、轮密钥加 π_{K_i} 。于是轮函数可以写作 $f_i=\pi_{K_i}\circ\rho\circ r\circ s$, 其中 i 为轮数。

nibble 代换 s : 该步将中间状态 $M=m_0||m_1||\dots||m_{63}$, 看作 16 个 4 bit nibble $n_0||n_1||\dots||n_{15}$ 。

对每个 $0\leq i\leq 15$, $n_i=(m_{i\times 4}, m_{i\times 4+1}, m_{i\times 4+2}, m_{i\times 4+3})$; 将每个 nibble n_i 的值用经由 4×4 的 4 bit S 盒 $S: F_2^4\rightarrow F_2^4$ 运算后所得的值代替。

对每个 $0\leq i\leq 15$,

$(m_{i\times 4}, m_{i\times 4+1}, m_{i\times 4+2}, m_{i\times 4+3}):=S(n_i)=S(m_{i\times 4}, m_{i\times 4+1}, m_{i\times 4+2}, m_{i\times 4+3})$, S 盒以 16 进制表示, 见表 1。

行移位 r : 该步中, 将中间状态 $n_0||n_1||\dots||n_{15}$ 看作 4×4 的 4 bit nibble 矩阵:

表 1 加密置换

x	$S[x]$
0	3
1	B
2	5
3	e
4	6
5	c
6	4
7	f
8	7
9	d
A	1
B	0
C	a
D	2
E	8
F	9

$$M=\begin{pmatrix} n_0 & n_1 & n_2 & n_3 \\ n_4 & n_5 & n_6 & n_7 \\ n_8 & n_9 & n_{10} & n_{11} \\ n_{12} & n_{13} & n_{14} & n_{15} \end{pmatrix} \quad (1)$$

对于 $i=0,1,2,3$, 将矩阵的第 i 行循环左移 i 个 nibble 的位置。

列混淆 ρ : 该步中 MDS 矩阵用于线性扩散, 该矩阵的设计原理见第 3.2 节; 具体可视为矩阵的连续 4 次运用。如上, 将中间状态看作 4×4 的 4 bit nibble 矩阵, 然后用矩阵 A 进行 4 次左乘。选取一个合适的作用于 nibble 之上的 4×4 的 4 bit 线性变换 L , 则矩阵 A 与最终的 MDS 矩阵 W 可写作:

$$W=(A)^4=\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L \\ 0 & 0 & 0 & 1 \\ L & 1 & 0 & 0 \end{pmatrix}^4=\begin{pmatrix} L & 1+L^2 & L & L^2 \\ L^3 & L+L^2 & 1+L^2 & L^3 \\ L^2 & L & L & 1+L^2 \\ L+L^3 & 1 & L & L+L^2 \end{pmatrix} \quad (2)$$

$$\text{线性变换 } L \text{ 则选取矩阵 } \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}。$$

轮密钥加 π : 该步将轮密钥 $K_i=k_i^0k_i^1\dots k_i^{63}$ 与中间状态值做异或运算, 用所得的结果更新状态寄存器。

2.2 解密过程

解密过程中, 按与加密过程相反的次序运用各轮轮密

钥,即在解密过程开始前,首先运用子密钥 K_r 进行轮密钥加,然后经过 R 轮解密轮,得出明文分组。加密过程轮函数为 $f_i = \pi_{K_i} \circ \rho \circ \sigma$, 则解密轮函数为 $f_i^{-1} = \pi_{K_i} \circ \sigma^{-1} \circ \rho^{-1}$, 此处的 i 表示第 i 次运用解密轮函数, 其中应用的密钥为子密钥 K_{R-i} 。

列混淆逆 ρ^{-1} : 该步中同样可视为矩阵 A 的逆矩阵 A^{-1} 的连续 4 次(左乘)运用, A^{-1} 利用 L 的逆变换 L^{-1} 定义如下:

$$W^{-1} = (A^{-1})^4 = \begin{pmatrix} L^{-1} & 0 & 0 & L^{-1} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & L & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^4 = \begin{pmatrix} L^{-1} + L^{-4} & 1 + L^{-2} & L + L^{-1} + L^{-3} & L^{-4} \\ L^{-3} & L^{-1} & 1 + L^{-2} & L^{-3} \\ L^{-2} + 1 + L^2 & L^3 & L^{-1} + L^4 & 1 + L^2 \\ L + L^{-1} & L^2 & L^3 & L^{-1} \end{pmatrix} \quad (3)$$

$$L^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (4)$$

行移位逆 r^{-1} : 将中间状态 $n_0 || n_1 || \dots || n_{15}$ 看作 4×4 的 4 bit nibble 矩阵, 对于 $i=0,1,2,3$, 将矩阵的第 i 行循环右移 i 个 nibble 的位置。

nibble 代换逆 s^{-1} : 所用的 4×4 的 4 bit 逆 S 盒 $S^{-1}: F_2^4 \rightarrow F_2^4$ 以 16 进制表示, 见表 2。

轮密钥加 π : 将轮密钥 $K_i = k_i^0 k_i^1 \dots k_i^{63}$ 与中间状态值做异

表 2 解密置换

x	$S[x]$
0	b
1	A
2	D
3	0
4	6
5	2
6	4
7	8
8	e
9	f
A	C
B	1
C	5
D	9
E	3
F	7

或运算, 用所得的结果更新状态寄存器。解密过程的轮密钥加运算与加密过程的轮密钥加运算操作是相同的, 只是每次运算使用的轮密钥不同。或者说, 从整体上加密与解密过程的轮密钥加运算仅仅是使用轮密钥的次序不相同。

2.3 密钥编排

Electricity 的密钥长度可选为 64 bit、80 bit 或 128 bit, 处理过程统一描述如下。

将 k 比特主密钥看作 ω 数目的 4 bit nibble, 写作 $V = v_0 || v_1 || \dots || v_k = n_0 || n_1 || \dots || n_{\omega}$ 。首先将其载入 k bit 密钥寄存器 $R = n_0 || n_1 || \dots || n_{\omega} = r_0 || r_1 || \dots || r_k = v_0 || v_1 || \dots || v_k$, 将寄存器中前 64 bit 取出作为轮密钥 $K_0: K_0 = k_0^0 k_0^1 \dots k_0^{63} = r_0 || r_1 || \dots || r_{63} (=v_0 || v_1 || \dots || v_{63})$, 然后对于 $i=1, \dots, r$ 来说, 对密钥寄存器进行如下更新与取轮密钥操作。

加轮常量: 将 5 bit LFSR 初始化为全 1 值 $(l_0, l_1, l_2, l_3, l_4) = (1, 1, 1, 1, 1)$, 首先依据反馈多项式 $x^5 + x^3 + 1$ 对其进行更新, 然后将其值异或到密钥寄存器的前 5 bit 中, $(r_0', r_1', r_2', r_3', r_4') = (r_0 \oplus l_0, r_1 \oplus l_1, r_2 \oplus l_2, r_3 \oplus l_3, r_4 \oplus l_4)$ 。以后每轮都首先依据 $x^5 + x^3 + 1$ 更新 LFSR 状态, 再进行异或操作。

nibble 代换: 将 k bit 密钥寄存器看作 $4 \times \lambda$ 的 4 bit nibble 矩阵。

$$R = \begin{pmatrix} n_0 & n_1 & n_2 & \dots & n_{\lambda-1} \\ n_{\lambda} & n_{\lambda+1} & n_{\lambda+2} & \dots & n_{2\lambda-1} \\ n_{2\lambda} & n_{2\lambda+1} & n_{2\lambda+2} & \dots & n_{3\lambda-1} \\ n_{3\lambda} & n_{3\lambda+1} & n_{3\lambda+2} & \dots & n_{\omega-1} \end{pmatrix} \quad (5)$$

其中, $\lambda = \frac{\omega}{4}$, 对参数 $k=64, 80, 128$ 而言, 分别有 $\omega=16, 20, 32$ 与 $\lambda=4, 5, 8$ 。对上述矩阵第 0 行的 λ 个 nibble $n_0, n_1, \dots, n_{\lambda-1}$ 运用 4 bit S 盒, 该 S 盒与加密过程中所用相同。

行移位: 与加密过程相似, 该步将密钥寄存器看作 $4 \times \lambda$ 的 4 bit nibble 矩阵, 当主密钥长度为 64 bit 和 128 bit 时, 对于 $i=0, 1, 2, 3$, 将矩阵的第 i 行循环左移 i 个 nibble 的位置; 当主密钥长度为 80 bit 时, 对于 $i=0, 1, 2, 3$, 将矩阵的第 i 行循环左移 $i+3$ 个 nibble 的位置。

列混淆: 与加密过程相似, 该步用矩阵 A 对轮密钥 nibble 矩阵进行 4 次左乘, 所用矩阵 A 与加密过程中所用 A 相同。

取 i 轮轮密钥: 取密钥寄存器的前 64 bit 作为第 i 轮的轮密钥, 即 $K_i = k_i^0 k_i^1 \dots k_i^{63} = r_0 || r_1 || \dots || r_{63}$ 。

2.4 数据填充算法

对长度不足 64 bit 的数据分组, 要先对其进行 10×1



式填充,即先填充一个比特 1,再填充足够多的比特 0,最后填充一个比特 1,将其补足为 64 倍数比特(64 或 128),详细算法如下。

算法 1 数据填充算法

输入 长度少于 64 bit 的数据块 m 。

输出 长度为 64 bit 或者 128 bit 的填充数据块 p 。

if 长度 of m 是 63

$p \leftarrow m || 1 || 0^{63} || 1$

else

$l \leftarrow \text{length of } m$

$p \leftarrow m || 1 || 0^{64-l-2} || 1$

end if

3 设计准则

Rijndael 自其诞生并被选为 AES 标准以来,受到了广泛的分析,实践证明其承受住了绝大部分的考验。在其基础上发展出了 Rijndael-like 结构,其定义如下。

定义 1 Rijndael-like 结构:指满足下述条件的 SP 网络型分组密码:

- 其线性变换形式为 $(\theta_1, \theta_2, \theta_3, \theta_4) \circ \pi$;
- (π 的条件) y_i 的各 nibble 来自不同的 x_i , 其中 $x=(x_1, x_2, x_3, x_4)$ 是 π 的输入, $y=(y_1, y_2, y_3, y_4)$ 是 π 的输出,即对于 $\forall i=0,1,2,3, y_i$ 的 4 个 nibble 分别来自 x_0, x_1, x_2 与 x_3 ;
- ($\theta=(\theta_1, \theta_2, \theta_3, \theta_4)$ 的条件) 将各 θ_i 看作线性变换,则要求: $\beta_d^{\theta_1} = \beta_d^{\theta_2} = \beta_d^{\theta_3} = \beta_d^{\theta_4}, \beta_l^{\theta_1} = \beta_l^{\theta_2} = \beta_l^{\theta_3} = \beta_l^{\theta_4}, \beta_d^{\theta_i}$ 与 $\beta_l^{\theta_i}$ 分别是线性变换 θ_i 在差分与线性分析中的分枝数目(branch number)。

对于 Rijndael-like 结构,其各种安全参数已有较完备的分析,得出了大量安全界。基于此,本方案延用 Rijndael-like 结构作为基础。

3.1 S 盒

轻量级密码学方案中,为了较好地兼顾安全性与性能,常选用 4 bit×4 bit 的 S 盒,选择依据如下。

- 最大差分概率为 2^{-2} :

$$\max_{a,b \neq 0} \frac{\#\{x | x \in F_2^4, S(x \oplus a) \oplus S(x) = b\}}{16} = 2^{-2}.$$

- 最大线性偏差为 2^{-2} :

$$\max_{u,v \neq 0} \left| \frac{\#\{x | x \in F_2^4, \langle u, s(x) \rangle = \langle v, s(x) \rangle\}}{16} \right| = 2^{-2}.$$

- 没有不动点。

- 代数次数达到最大值 3。

- 硬件实现面积小。

最终选定的 S 盒,硬件实现只需 8 次操作,共计约

15.01 GE:

- $t_1 = \sim(a \& b)$;
- $e = c \oplus t_1$;
- $t_3 = b | c$;
- $g = d \oplus t_3$;
- $t_s = \sim(a \& g)$;
- $f = b \oplus t_s$;
- $t_7 = \sim(e | g)$;
- $h = a \oplus t_7$ 。

3.2 线性变换层 r 与 p

AES 的设计过程中,提出了线性变换/扩散层(diffusion layer)分枝数目的概念,含义是差分/线性掩码经线性变换后,输入输出值中不为 0 的 S 盒的最小数目。分枝数目越大,扩散层的扩散效果越好。对 S 盒数目为 s 的加密方案而言,其分枝数目为 $s+1$ 。将达到了此数值的线性变换称为完美扩散层(perfect diffusion layer)。

已经证明,当 $[m, s, d]$ 码的生成矩阵所有的子方阵均为非奇异矩阵时,该码是 MDS 码,这一性质可用于检验所构造的线性变换是否完美(达到分枝数目)。实际构造这样的矩阵时,一方面希望构造过程简明,另一方面希望构造的矩阵有较为简单的硬件实现,一种构造方式是利用多个基于 bundle(bundle-based)的 LFSR(linear feedback shifting register,一种流密码技术),经过 d 次更新,实现线性变换。在这种构造方式下,最终的线性变换矩阵 D 就是这些 LFSR 的状态转移矩阵 A 的 d 次幂: $D = A^d$ 。本方案选取的就是这样一个矩阵:

$$D = (A)^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L \\ 0 & 0 & 0 & 1 \\ L & 1 & 0 & 0 \end{pmatrix}^4 = \begin{pmatrix} L & 1+L^2 & L & L^2 \\ L^3 & L+L^2 & 1+L^2 & L^3 \\ L^2 & L & L & 1+L^2 \\ L+L^3 & 1 & L & L+L^2 \end{pmatrix} \quad (6)$$

矩阵 D 是作用于 4 个 4 bit bundle 上的矩阵(16 bit),其中的矩阵 L 则是作用于 4 bit bundle 上的矩阵。若 L 选为有且仅有 4 个 1 的可逆矩阵,则 D 成为比特置换,而实际倾向于选取硬件实现面积最小且满足最终构造的矩阵 D 是 MDS 矩阵的 L 。

上述矩阵 D 是 MDS 矩阵的条件是其所有子方阵都非奇异,这样可以计算得出 7 个条件,即下述 7 个矩阵均非奇

异: $L, L+1, L^2+L+1, L^3+L+1, L^3+L^2+1, L^4+L^3+1, L^4+L^3+L^2+L+1$ 。

$$L \text{ 的一种选择是取矩阵 } \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}。$$

这个 L 硬件实现时仅需要若干布线与一个异或门,其开销很低。总的来看,此时矩阵 D 的硬件实现需 $2n+2$ 数目的异或门电路,在同类变换中是最低的,且达到 4-bundle 线性变换的分枝数,在性能与安全性两方面都十分出色。

3.3 密钥编排设计

本方案的密钥编排设计遵循如下理念:

(1)在编排过程中使用非线性变换以增加安全强度,降低相关密钥攻击的可能性;

(2)在(1)的前提下,使用尽量少的 S 盒,以减少硬件实现开销,方案中每次只对状态寄存器中 1/4 的 4 bit 单元运用 S 盒;

(3)编排过程尽量使用加密过程中使用的部件,以便实现时共用部件,降低实现开销;

(4)引入与轮数相关的量以抵御 slide attack (滑动攻击)。

综合上述几点,最终选用了第 2.3 节、第 3.3 节所述的编排方案。

4 安全性分析

本节从差分概率/线性闭包偏差上界、密钥编排安全性等方面分析本方案的安全性。

4.1 差分概率/线性闭包偏差上界

对 Rijndael-like 结构的差分概率/线性闭包偏差上界已有较完善的研究,这样的设计不仅若干轮的单条轨迹概率/偏差很低,甚至连多条迹聚合而成的概率/偏差也不会很高。

具体地,对于 2 轮的 SPN 结构,差分方面有如下定理。

定理 1 若 L 的差分分枝数为 β_d , 则 2 轮 SPN 结构的差分概率上界为:

$$\max \left\{ \begin{array}{l} \max_{1 \leq i \leq n} \max_{1 \leq u \leq 2^m-1} \sum_{j=1}^{2^m-1} \{DP^{S_i}(u, j)\}^{\beta_d}, \\ \max_{1 \leq i \leq n} \max_{1 \leq u \leq 2^m-1} \sum_{j=1}^{2^m-1} \{DP^{S_i}(j, u)\}^{\beta_d} \end{array} \right\} \quad (7)$$

其中, $DP^S(u, j)$ 是第 i 个 S 盒将输入差分 u 映射到输出

差分 j 的概率。将本方案的 S 盒差分分布数据代入式(7), 计算得出 2 轮差分对概率不超过 2^{-8} ; 再依照与定理 1 中类似的分析过程, 可得本方案 4 轮差分对的概率不超过 2^{-32} 。

线性方面, 定义相关系数为两倍的线性偏差, 则有如下定理。

定理 2 若 L 的线性分枝数为 β_l , 则 2 轮 SPN 结构的线性相关系数平方上界为:

$$\max \left\{ \begin{array}{l} \max_{1 \leq i \leq n} \max_{1 \leq u \leq 2^m-1} \sum_{j=1}^{2^m-1} \{LP^{S_i}(u, j)\}^{\beta_l}, \\ \max_{1 \leq i \leq n} \max_{1 \leq u \leq 2^m-1} \sum_{j=1}^{2^m-1} \{LP^{S_i}(j, u)\}^{\beta_l} \end{array} \right\} \quad (8)$$

此处 $LP^S(u, j)$ 是掩码对 (u, j) 在第 i 个 S 盒上的相关系数平方值。将本方案的 S 盒线性参数代入式(8), 计算得出 2 轮线性掩码的相关系数平方值不超过 2^{-8} ; 再依与定理 2 中类似的分析过程, 可得出本方案 4 轮线性掩码的相关系数平方值不超过 2^{-32} 。此处的计算结果在值上与差分方面是相等的。

4.2 高阶差分攻击、插值攻击与代数攻击

高阶差分攻击利用某些加密方案代数次数相对较低的特点, 将高阶差分值作为区分依据, 当加密方案输出结果的某个比特可以写作输入比特的 n 次多项式时, 方案输出值的阶差分值在该比特位上是常数。插值攻击和代数攻击也利用这种相对较低的代数次数: 插值攻击通过插值法求解输出比特或中间比特的多项式表达式, 实施区分; 代数攻击则在输入比特、密钥比特与输入比特之间建立二次代数方程组, 设法通过求解方程组完成密钥的恢复。

鉴于本方案采用的 S 盒代数次数最高为 3, 仅在部分输出上为 2, 即使在最不利的情形下, 经 7 轮以上后, 各输出比特表达为输入比特的代数形式后, 次数也已达到 $2^7=128$, 这意味着至少要 2^{128} 数目的明文才能得到固定值的高阶(128 阶)差分; 鉴于方案的分组长度仅为 64 bit, 获取这么多的明文是不可能的。考虑到方案的实际轮数超过 16 轮, 各经过改进的高阶差分变体成功的可能性也是极低的。

鉴于类似的考量, 也无法得到实施插值攻击所需的数据规模。在代数攻击方面, 则会因为次数过高、方程组规模过大而无法进行。

4.3 密钥编排安全性

涉及密钥编排的攻击包括 slide attack、相关密钥攻击

与中间相遇攻击等。为了抵御 slide attack, 设计中引入了与轮数相关的常量, 因此这种攻击成功的可能性不大; 由于轮函数采用了 Rijndael-like 结构, 引入的差分会迅速扩散, 因此相信相关密钥攻击也不太可能; 而对于中间相遇攻击, 由于 Rijndael-like 结构中各比特的值会随 nibble 迅速扩散开, 因此不太可能找到对较多轮数有效的中立密钥比特, 也就无法高效地实施中间相遇攻击。

5 硬件实现评估

本方案采用的结构与各部件都已有较完善的讨论。根据已有成熟结论, 给出本方案的串行化硬件实现结构, 其 64 bit 变体的实现如图 1 所示 (datapath 为 4 bit, 密钥编排部分与加密部分很相似)。

考虑上述实现的硬件面积。加密部分的状态框内含 16 个 4 bit 存储单元, 其中实线框为双输入的单元, 平均每比特存储约需 6 GE, 虚线框为单输入的单元, 平均每比特可以仅使用约 4.67 GE; 密钥编排模块的 64 bit 存储与之类似; S 盒约需 15.01 GE; 列混淆模块总计约需 26.7 GE; 用于生成常量的 5 bit LFSR 约需 26.02 GE; 控制逻辑需一个 idle、一个控制初始的数据载入、一个控制轮密钥加与代换、3 个控制行移位、2 个控制列混淆, 总计 8 个状态; 此外, 需 3 bit 的 LFSR 作为转移条件, 总计约需 80 GE。再考虑数据处理过程中用到的异或与片选组件, 估计总计约需 916.28 GE; 其中控制逻辑约占 8.73%。这样的实现每轮约需 39 个时钟周期, 因此 16 轮加密总计需 624 个时钟周期。

其中 L 与 S 盒都是 4 bit 输入、4 bit 输出的元件, 模块 L 的细节如图 2 所示。

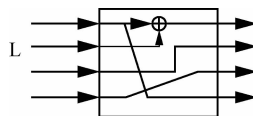


图 2 模块 L 的实现示意

在上述实现中, 加密与密钥编排部分分别用了两个列混淆模块与 S 盒, 还有一种高度串行的实现方法是令加密与密钥编排部分共用同一个列混淆与 S 盒, 但实践发现, 这种实现看似节约了两个加密模块, 却要额外使用多个片选电路, 且增加控制逻辑的规模与复杂度, 加倍加密所需的时钟周期数目, 实际能减少的实现面积却极为有限, 因此不予讨论。

80 bit 与 128 bit 的变体在硬件实现架构上与 64 bit 变体是相似的, 不同点在于密钥编排部分需要更多的状态寄存器。此外, 由于每轮密钥编排中全部完成列混淆运算所需的时钟周期数目不相同 (64 bit 分组需 $4 \times 4 + 4 = 20$ 个周期, 80 bit 密钥状态需 $5 \times 4 + 5 = 25$ 个周期, 128 bit 密钥状态需 $8 \times 4 + 8 = 40$ 个周期)。对于 128 bit 密钥的变体, 由于密钥编排多出的周期数目 40 为 4 的倍数, 尚可在其期间内令数据状态寄存器反复循环移位来消耗时间; 对于 80 bit 密钥变体, 多出的 5 个时钟周期则难处理, 为此将其在行移位阶段设计成移动位, 以便使用多出的 3 个周期将其补齐。这样最终推算结果见表 3。

6 软件实现评估

利用 SSE 指令实现的本方案实例在便携计算机上进行了软件实现性能测试实验。该便携机处理器为 Intel i5-2400, 主频为 2.5 GHz, 内存为 4 GB。Electricity 软件实现性能数据见表 4。

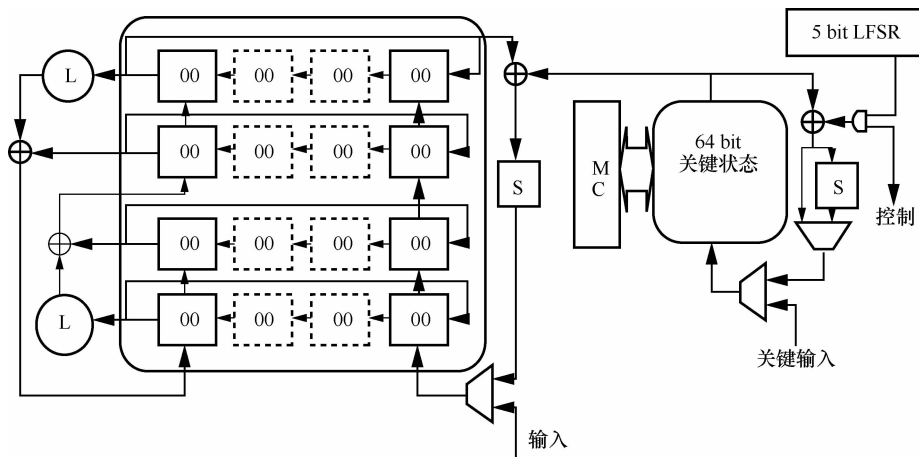


图 1 硬件实现示意

表 3 Electricity 硬件实现性能

对比项	硬件面积/GE	每分组数据加密 所需时钟周期	FOM: $\frac{bits \times 10^9}{clk \cdot GE^2}$
Electricity-64	916.28	624	122
Electricity-80	991	846	77
Electricity-128	1 215.16	1 180	37

表 4 Electricity 软件实现性能数据

对比项	密钥编排时 耗/CPU 周期	加密速度/(CPU 周期 · byte ⁻¹)	解密速度/(CPU 周期 · byte ⁻¹)
Electricity-64	275	29.6	67.8
Electricity-80	284	33.3	75.7
Electricity-128	575	36.9	83.8

7 结束语

得益于 S 盒与 MDS 矩阵研究的后期进展，本方案获得了相对已有方案而言极有竞争力的硬件实现面积。选取了合适的轮数后，本方案的硬件实现加密延时得到控制，这方面优于 LED 很多，加之各部件面积的优化，整体硬件实现面积也优于 LED。

参考文献：

[1] QIU S, BAI G Q, CHEN H Y. One-dimensional confusion coefficient for block cipher [J]. Journal of Cryptologic Research, 2014, 1(2): 124-133.

[2] ZHOU Y B, LI J T, LIU J Y. The evolution of security

requirements for cryptographic modules: the status quo, dilemma and future trends [J]. Journal of Chengdu University of Information Technology. 2011, 26(2): 109-122.

[3] ZHANG S Y, CHEN G L, FAN L, et al. Algorithm of G-AES[J]. Journal of Cryptologic Research, 2014, 1(2): 187-199.

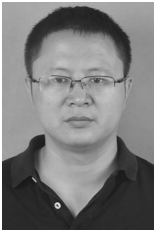
[4] LIU Y Z, ZHANG L, LUO P, et al. Research of trustworthy software system in the network [C]// The 5th International Symposium on Parallel Architectures, Algorithms and Programming, December 17-20, 2012, Taipei, China. New Jersey: IEEE Press, 2012: 287-294.

[5] HONG D, SUNG J, HONG S, et al. HIGHT: a new block cipher suitable for low-resource device [M]. Berlin: Springer, 2006: 46-59.

[6] Leander G, PAAR C, POSCHMANN A, et al. New lightweight DES variants [J]. Fast Software Encryption 2007-FSE 2007, 4593: 196-210.

[7] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: an ultra-lightweight block cipher [J]. Lecture Notes in Computer Science, 2007(4727): 450-466.

[作者简介]



王涛(1977-),男,全球能源互联网研究院工程师,主要研究方向为信息系统数据安全、密码算法实现及应用。