SM4 分组密码算法综述

吕述望[1] 苏波展[2] 王 鹏[3] 毛颖颖[4] 霍利利[4]
[1](中国科学院数据与通信保护研究教育中心 北京 100093)
[2](密码科学技术国家重点实验室 北京 100878)
[3](中国科学院信息工程研究所 北京 100093)
[4](国家密码管理局商用密码检测中心 北京 100036)
(subozhan@163.com)

Overview on SM4 Algorithm

Lü Shuwang[1], Su Bozhan[2], Wang Peng[3], Mao Yingying[4], and Huo Lili[4]
[1](*Data Assurance & Communication Security Center, Chinese Academy of Science*, 100093)
[2](*State Key Laboratory of Cryptology, Beijing* 100878)
[3](*Innstitute of Information Engineering, Chinese Academy of Science*, 100093)
[4](*Commercial Cryptography Testing Center*, 100036)

Abstract SM4 Algorithm, short for SM4 Block Cipher Algorithm, was published in 2006 to promote the application of WAPI. It became a cryptography industrial standard (GM/T 0002—2012) in March 2012 and a national standard (GB/T 32907—2016) in August 2016. This paper describes SM4's calculating process, structural features and cryptographic properties. Furthermore, we introduce some latest researches on SM4's security and compare SM4's security with several international block cipher standards such as AES, HIGHT and MISTY1.

Key words SM4 block cipher algorithm; differential cryptanalysis; linear cryptanalysis; S box

摘 要 SM4 分组密码算法简称为 SM4 算法,为配合 WAPI 无线局域网标准的推广应用,SM4 算法于 2006 年公开发布,2012 年 3 月发布成为国家密码行业标准(标准号为 GM/T 0002—2012),2016 年 8 月发布成为国家标准(标准号为 GB/T 32907—2016).介绍了 SM4 分组密码算法的算法流程、结构特点及其密码特性,以及 SM4 算法的安全性分析研究现状,并与国际标准分组算法的安全性进行了对比.

关键词 SM4 分组密码算法;差分密码分析;线性密码分析;S 盒

中图法分类号 TP309

分组密码算法是现代密码学中的一个重要研究方向,是保障信息机密性和完整性的重要技术手段.分组密码的设计理念源于 Shannon 在 1949 年发表的经典论文 Communication Theory of Secret System[1],对于分组密码算法的公开研究始于 20 世纪 70 年代末 DES 算法[2]的公布,而分组密码理论及应用的飞速发展则得益于 20 世纪 90 年代末美国的 AES 计划[3]和本世纪初欧洲的 NESSIE 计划[4].

1977 年,美国公布了数据加密标准 DES.尽管目前 DES 算法已逐步退出历史舞台,但 DES 的出现对分组密码理论的发展起到了极大的促进作

收稿日期:2016-10-25

用.通过对DES算法安全性的研究,分组密码的分析理论日渐成熟,而日益发展的分组密码分析理论也带来了分组密码设计理论新的发展.

1997年,美国国家标准技术研究院(National Institute of Standard Technology, NIST)发起了对称密码算法推选活动,获选算法将用于联邦政府敏感信息的加密保护.2000年,NIST推选Rijndael算法作为高级加密标准AES.欧洲则于2000年启动了NESSIE(New European Schemes for Signatures, Integrity, and Encryption)计划,并于2003年公布Camellia算法、MISTY算法、SHACAL-2算法以及AES算法共同作为欧洲新世纪的分组密码算法标准.

在AES计划和NESSIE计划中,分组密码算法的设计与分析理论得到了深入的研究与发展,同时各国也竞相设计能同时满足安全性及实现效能需求的分组密码算法.为配合我国WAPI无线局域网标准的推广应用,SM4分组密码算法(原名SMS4[5])于2006年公开发布.随着我国密码算法标准化工作的开展,SM4算法于2012年3月发布成为国家密码行业标准[6](标准号为GM/T 0002—2012),于2016年8月发布成为国家标准[7](标准号为GB/T 32907—2016).2016年10月,ISO/IEC SC27会议专家组一致同意将SM4算法纳入ISO标准的学习期,我国SM4分组密码算法正式进入了ISO标准化历程.

1 SM4算法描述

SM4分组密码算法是一个迭代分组密码算法,由加解密算法和密钥扩展算法组成.SM4分组密码算法采用非平衡Feistel结构,分组长度为128 b,密钥长度为128 b.加密算法与密钥扩展算法均采用32轮非线性迭代结构.加密运算和解密运算的算法结构相同,解密运算的轮密钥的使用顺序与加密运算相反.

1.1 密钥及密钥参量

SM4分组密码算法的加密密钥长度为128 b,表示为$MK=(MK_0, MK_1, MK_2, MK_3)$,其中$MK_i(i=0,1,2,3)$为32 b.

轮密钥表示为$(rk_0, rk_1, \cdots, rk_{31})$,其中$rk_i$ $(i=0,1,\cdots,31)$为32 b.轮密钥由加密密钥生成.

$FK=(FK_0, FK_1, FK_2, FK_3)$为系统参数,$CK=(CK_0, CK_1, \cdots, CK_{31})$为固定参数,用于密钥扩展算法,其中$FK_i(i=0,1,\cdots,3)$,$CK_i(i=0,1,\cdots,31)$均为32 b.

1.2 加密算法

SM4加密算法由32次迭代运算和1次反序变换R组成.

设明文输入为$(X_0, X_1, X_2, X_3)\in(Z_2^{32})^4$,密文输出为$(Y_0, Y_1, Y_2, Y_3)\in(Z_2^{32})^4$,轮密钥为$rk_i\in Z_2^{32}$, $i=0,1,\cdots,31$.加密算法的运算过程如下.

1) 首先执行32次迭代运算:

$$X_{i+4}=F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i)=X_i\oplus T(X_{i+1}\oplus X_{i+2}\oplus X_{i+3}\oplus rk_i),\ i=0,1,\cdots,31$$

2) 对最后一轮数据进行反序变换并得到密文输出:

$$(Y_0, Y_1, Y_2, Y_3)=R(X_{32}, X_{33}, X_{34}, X_{35})=(X_{35}, X_{34}, X_{33}, X_{32}).$$

其中,$T: Z_2^{32}\rightarrow Z_2^{32}$一个可逆变换,由非线性变换$\tau$和线性变换$L$复合而成,即$T(\cdot)=L(\tau(\cdot))$.

非线性变换τ由4个并行的S盒构成.设输入为$A=(a_0, a_1, a_2, a_3)\in(Z_2^8)^4$,非线性变换$\tau$的输出为$B=(b_0, b_1, b_2, b_3)\in(Z_2^8)^4$,即:

$$(b_0, b_1, b_2, b_3)=\tau(A)=(Sbox(a_0), Sbox(a_1), Sbox(a_2), Sbox(a_3)),$$

其中,S盒数据如表1所示:

表1 Sbox数据

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	D6	90	E9	FE	CC	E1	3D	B7	16	B6	14	C2	28	FB	2C	05
1	2B	67	9A	76	2A	BE	04	C3	AA	44	13	26	49	86	06	99
2	9C	42	50	F4	91	EF	98	7A	33	54	0B	43	ED	CF	AC	62
3	E4	B3	1C	A9	C9	08	E8	95	80	DF	94	FA	75	8F	3F	A6
4	47	07	A7	FC	F3	73	17	BA	83	59	3C	19	E6	85	4F	A8
5	68	6B	81	B2	71	64	DA	8B	F8	EB	0F	4B	70	56	9D	35
6	1E	24	0E	5E	63	58	D1	A2	25	22	7C	3B	01	21	78	87
7	D4	00	46	57	9F	D3	27	52	4C	36	02	E7	A0	C4	C8	9E
8	EA	BF	8A	D2	40	C7	38	B5	A3	F7	F2	CE	F9	61	15	A1
9	E0	AE	5D	A4	9B	34	1A	55	AD	93	32	30	F5	8C	B1	E3
A	1D	F6	E2	2E	82	66	CA	60	C0	29	23	AB	0D	53	4E	6F
B	D5	DB	37	45	DE	FD	8E	2F	03	FF	6A	72	6D	6C	5B	51
C	8D	1B	AF	92	BB	DD	BC	7F	11	D9	5C	41	1F	10	5A	D8
D	0A	C1	31	88	A5	CD	7B	BD	2D	74	D0	12	B8	E5	B4	B0
E	89	69	97	4A	0C	96	77	7E	65	B9	F1	09	C5	6E	C6	84
F	18	F0	7D	EC	3A	DC	4D	20	79	EE	5F	3E	D7	CB	39	48

设S盒的输入为EF，则经S盒运算的输出结果为表中第E行、第F列的值，即 $Sbox(EF)=0x84$.

L 是线性变换，非线性变换 τ 的输出是线性变换 L 的输入. 设输入为 $B\in Z_2^{32}$，输出为 $C\in Z_2^{32}$，则：

$$C=L(B)=B\oplus(B<<<2)\oplus(B<<<10)\oplus(B<<<18)\oplus(B<<<24).$$

其加密算法的运算如图1、图2所示：

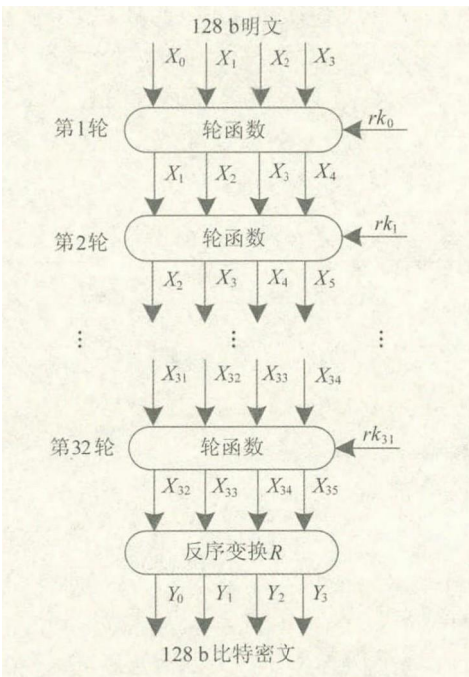

图1 SM4算法加密流程示意图

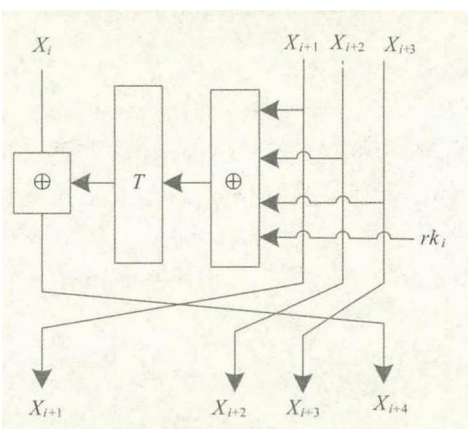

图2 SM4算法轮函数示意图

1.3 解密算法

本算法的解密变换与加密变换结构相同，不同的仅是轮密钥的使用顺序. 解密时使用轮密钥序 $(rk_{31},rk_{30},\cdots,rk_0)$.

1.4 密钥扩展算法

本算法轮密钥由加密密钥通过密钥扩展算法生成. 设加密密钥为 MK，

$$MK=(MK_0,MK_1,MK_2,MK_3)\in(Z_2^{32})^4.$$

轮密钥生成方法为

$$rk_i=K_{i+4}=K_i\oplus T'(K_{i+1}\oplus K_{i+2}\oplus K_{i+3}\oplus CK_i),(i=0,1,\cdots,31),$$

其中：

$$K_0=MK_0\oplus FK_0;$$
$$K_1=MK_1\oplus FK_1;$$
$$K_2=MK_2\oplus FK_2;$$
$$K_3=MK_3\oplus FK_3.$$

1) T' 是将1.2节中合成置换 T 的线性变换 L 替换为 L'：

$$L'(B)=B\oplus(B<<<13)\oplus(B<<<23).$$

2) 系统参数 FK 的取值为

$$FK_0=(A3B1BAC6);$$
$$FK_1=(56AA3350);$$
$$FK_2=(677D9197);$$
$$FK_3=(B27022DC).$$

3) 固定参数 CK 取值方法为：

设 $ck_{i,j}$ 为 CK_i 的第 j 字节 $(i=0,1,\cdots,31;j=0,1,2,3)$，即 $CK_i=(ck_{i,0},ck_{i,1},ck_{i,2},ck_{i,3})\in(Z_2^8)^4$，则 $ck_{i,j}=(4i+j)\times7(\mathrm{mod}\ 256)$.

固定参数 $CK_i(i=0,1,\cdots,31)$ 的具体值为：00070E15，1C232A31，383F464D，545B6269，70777E85，8C939AA1，A8AFB6BD，C4CBD2D9，E0E7EEF5，FC030A11，181F262D，343B4249，50575E65，6C737A81，888F969D，A4ABB2B9，C0C7CED5，DCE3EAF1，F8FF060D，141B2229，30373E45，4C535A61，686F767D，848B9299，A0A7AEB5，BCC3CAD1，D8DFE6ED，F4FB0209，10171E25，2C333A41，484F565D，646B7279.

2 SM4算法设计原理

2.1 基础置换

混乱原则和扩散原则是分组密码设计的2个基本原则. 一个设计精良的分组密码体制应该以一类密码学特征良好的基础置换为主体来构造，其单轮运算应当基于一类密码学特征良好的基础

置换. 基础置换的密码学性质决定明密文变换的效率.

SM4 算法是基于正形置换[8]构造的,SM4 算法的单轮变换构成正形置换,其密码特性可以由正形置换的性质推出.

设 SM4 分组密码算法的单轮置换为 P,对于任意给定的明文 X,如果密钥 $K'\neq K$,则 $P(X,K')\neq P(X,K)$.

该结论表明,如果以 X 为行变量,以 K 为列变量,则方阵 $P(X,K)$构成拉丁方. 在密码学性质上包含了 2 个结论:

1) SM4 分组密码算法在不同密钥作用下的轮变换必然不同;

2) SM4 分组密码算法的单轮变换在不同的密钥作用下,输入明文相同而输出必然不同.

2.2 非线性变换

S 盒本质上可以看作映射:

$S(X)=(f_1(X),f_2(X),\cdots,f_m(X)):F_2^n\to F_2^m$,

其中,$S(x):F_2^n\to F_2^m$ 可表示为一个 n 元输入 m 元输出的多输出布尔函数,也可简称 S 是一个 $n\times m$ 的 S 盒(n 进 m 出的 S 盒),通常采用 n 比特输入到 m 比特输出的替代表来表示或实现,对于 SM4 分组密码算法中的 S 盒,$n=m=8$.

S 盒是很多分组密码算法中的唯一非线性模块,用于提供混淆作用,可提高算法的非线性性,隐藏其代数结构.

S 盒的密码性质直接影响了整个分组密码算法的安全强度. 分组密码算法的设计必须充分考量 S 盒的密码强度,通常可用非线性度、差分均匀性等指标来衡量 S 盒的安全强度.

1) S 盒代数表达式

为防止插入攻击,通常要求密码变换的代数式具有足够高的次数和复杂度. 用拉格朗日插值多项式可求得 SM4 算法 S 盒的代数表达式. 这是一个 254 次、255 项的多项式,具有最高的复杂程度[9].

$$f(x):\sum_{i=0}^{255}y_i\prod_{j\neq i,j=0}^{255}\frac{x-x_j}{x_i-x_j}.$$

2) 代数次数及项数分布

文献[10]提到任何 n 元布尔函数 $f(X):F_2^n\to F_2$ 都可以唯一地表示成如下的代数正规形式:

$$f(X)=a_0+\sum_{\substack{1\leqslant i_1<\cdots<i_k\leqslant n\\1\leqslant k\leqslant n}}a_{i_1i_2\cdots i_k}x_{i_1}x_{i_2}\cdots x_{i_k},$$

$$X=(x_1,x_2,\cdots,x_n),a_0,a_{i_1i_2\cdots i_k}\in F_2.$$

文献[10]给出了 n 元布尔函数代数项数及次数的定义:代数正规形式中的最高项的次数称为 $f(X)$的次数;它的代数正规形式中的 i 次项的个数称为 $f(X)$的 i 次项数;所有 $i(0\leqslant i\leqslant n)$次项数之和称为 $f(X)$的项数.

$S(X)$可以表示为 m 个分量函数 $S(X)=(f_1(X),f_2(X),\cdots,f_m(X)):F_2^n\to F_2^m$,若将 $S(X)$看成一个随机置换,它的每个分量函数的代数次数最佳为 $n-1$,每个分量函数的 i 次项数应接近于 $C_n^i/2$. 若代数次数太低,例如,每个分量函数的次数都是 2,则算法易受高阶差分密码分析的攻击. 若项数太少,有可能提高插值攻击的成功率.

SM4 算法 S 盒的代数次数及项数分布如表 2 所示:

表 2 SM4 算法 S 盒的代数次数及项数分布

分量函数	代数次数								
	8	7	6	5	4	3	2	1	0
$Y0$	0	3	15	31	28	29	14	3	1
$Y1$	0	3	12	34	40	33	12	4	1
$Y2$	0	5	17	24	40	24	11	3	0
$Y3$	0	2	11	31	34	27	15	5	1
$Y4$	0	5	15	28	33	24	13	5	0
$Y5$	0	5	11	25	41	25	16	4	1
$Y6$	0	4	15	29	27	32	18	4	1
$Y7$	0	4	14	32	35	30	16	3	0
期望值	1/2	4	14	28	35	28	14	4	1/2

3) 差分均匀性

文献[10]给出了差分均匀性的定义. 差分密码分析是一种选择明文攻击,其基本思想是通过分析特定明文差对相应密文差的影响来获得可能性最大的密钥. 差分均匀性是针对差分密码分析而引入的,用来度量一个密码函数抗击差分密码分析的能力. 令:

$$\delta_S=\frac{1}{2^n}\max_{\substack{\alpha\in F_2^n\\\alpha\neq 0}}\max_{\beta\in F_2^m}|\{X\in F_2^n:S(X\oplus\alpha)-S(X)=\beta\}|,$$

称 δ_S 为 S 盒的差分均匀性.

根据差分均匀性的定义,可以得到 $2^{-m}\leqslant\delta_S\leqslant 2^{m-n}$,如有 $\delta_S=2^{m-n}$则称 S 是从 F_2^n 到 F_2^m 的完全非线性函数. 为了抵抗差分密码攻击,差分均匀度

应该越低越好.

SM4 算法 S 盒的最大差分概率仅为 2^{-6}，具有较好的抗差分分析特性.

4) 非线性度

文献[10]给出了 S 盒的非线性度定义：令 $S(X)=(f_1(X),f_2(X),\cdots,f_m(X)):F_2^n\to F_2^m$ 是一多输出函数，称 $N_S=\min\limits_{l\in L_n} d_H(u\cdot S(X),l(X))$ $0\neq u\in F_2^m$ 为 $S(X)$ 的非线性度. 其中 L_n 表示全体 n 元仿射函数集合，$d_H(f,l)$ 表示 f 与 l 之间的汉明距离. 从定义可以看出，S 盒的非线性度就是输出位的任意线性组合和所有关于输入的仿射函数的最小汉明距离. 可以证明，非线性度的上界为 $2^{n-1}-2^{\frac{n}{2}-1}$. 达到上界的布尔函数称为 Bent 函数.

布尔函数的非线性度是用来衡量抵抗“线性攻击”能力的一个非线性准则，非线性度越大，则布尔函数 $f(x)$ 抵抗“线性攻击”的能力越强；反之，非线性度越小，则布尔函数抵抗“线性攻击”的能力越弱.

SM4 算法 S 盒非线性度为 112.

5) 最大线性优势

文献[10]给出了 S 盒的线性逼近的定义：假设一个 n 进 m 出的 S 盒，其任意线性逼近都可以表示为：$a\cdot X=b\cdot Y$，其中 $a\in F_2^n, b\in F_2^m$. $a\cdot X=b\cdot Y$ 成立的概率 p 满足 $\left|p-\frac{1}{2}\right|\leqslant\frac{1}{2}-\frac{N_S}{2^n}$，$\left|p-\frac{1}{2}\right|$ 称为线性逼近等式的优势，$\lambda_S=\frac{1}{2}-\frac{N_S}{2^n}$ 为 S 盒的最佳优势.

SM4 算法的最佳优势为 2^{-4}.

6) 平衡性

文献[11]提到 $S(x)=(f_1(x),f_2(x),\cdots,f_m(x)):F_2^n\to F_2^m$ 是平衡的，若对任意的 $\beta\in F_2^m$，恰好有 2^{n-m} 个 $x\in F_2^n$，使得 $S(x)=\beta$. 满足平衡性质的 S 盒也被称为是正交的.

SM4 算法 S 盒满足平衡性.

7) 完全性及雪崩效应

文献[10]给出了 S 盒完全性的定义：$S(X)=(f_1(X),f_2(X),\cdots,f_m(X)):F_2^n\to F_2^m$ 是完全的，是指输出的任一比特和输入的每一比特有关. 体现在代数表达式中，是指每个分量函数的代数表达式包含所有未知变量 $x_1,x_2,\cdots,x_n$. 也就是说对任意 $(s,t)\in\{(i,j)\mid 1\leqslant i\leqslant n, 1\leqslant j\leqslant m\}$，存在 X，使得 $S(X)$ 和 $S(X\oplus e_s)$ 的第 t 比特不同.

雪崩效应[10]是指改变输入的 1 b，大约有一半输出比特改变.

SM4 算法的 S 盒满足完全性及雪崩效应.

2.3 线性变换

线性变换用于提供扩散作用. 分组密码算法通常采用若干 $m\times m$ 的 S 盒并置构成混淆层，一个 S 盒输出的 m 比特仅与其输入的 m 比特有关，与其他 S 盒的输入无关，此时引入线性变换可以将这些 S 盒的输出打乱、混合，使得输出的 m 比特数据尽可能地与其他 S 盒的输入相关. 好的线性变换设计使得 S 盒的输出得到扩散，使得密码算法能够抵抗差分分析和线性分析. 衡量一个线性变换的扩散性的重要指标是分支数. 文献[10]给出了分支数的定义：

$$B(\theta)=\min_{x\neq 0}(w_b(x)+w_b(\theta(x))),$$

称 $B(\theta)$ 为变换 θ 的分支数，其中 $w_b(x)$ 表示非零 $x_i(1\leqslant i\leqslant m)$ 的个数，称为 x_i 的包重量(bundle weight).

分支数的概念可用于量化分组密码算法对差分密码分析及线性密码分析的抵抗能力，针对差分密码分析及线性密码分析，可类似地定义 θ 的差分分支数[10]：

$$B_d(\theta)=\min_{x,x\neq x^*}(w_b(x\oplus x^*)+w_b(\theta(x)\oplus\theta(x^*)))$$

及线性分支数[10]：

$$B_l(\theta)=\min_{\alpha,\beta,c(x\cdot\alpha^t,\theta(x)\cdot\beta)\neq 0}(w_b(\alpha)+w_b(\beta)),$$

其中，$c(x\cdot\alpha^t,\theta(x)\cdot\beta)=2\times Pr(x\cdot\alpha^t=\theta(x)\cdot\beta)-1$，$x\cdot\alpha^t$ 是矩阵乘. 对于线性变换，分支数的概念反映了其扩散性的好坏，分支数越大，扩散效果越好. 线性变换的差分(线性)分支数越大，差分(线性)密码分析所需的选择(已知)明文数越多.

SM4 分组密码算法线性变换的差分分支数及线性分支数均为 5.

2.4 密钥扩展算法

密钥扩展算法的设计充分考虑了加密算法对密钥扩展算法的安全需求及其实现的便利性，尽可能使算法达到更高的性能.

子密钥是由加密密钥派生的，理论上子密钥

总是统计相关的，文献[10]也提到，在实用密码算法的设计中，子密钥统计独立是不可能做到的，设计者只是尽可能使得子密钥趋近于统计独立. 密钥扩展算法的目的就是使子密钥间的统计相关性不易被破解利用，或者说使子密钥看上去更像是统计独立的. 在密钥扩展算法的设计上 SM4 分组密码算法满足以下准则：

1）子密钥间不存在明显的统计相关性；

2）没有弱密钥；

3）密钥扩展的速度不低于加密算法的速度，且资源占用少；

4）由加密密钥可以直接生成任何一个子密钥.

3 SM4 算法安全性分析

3.1 SM4 算法安全性分析现状

SM4 分组密码算法自从 2006 年 1 月发布以来，国内外众多的科研人员对其安全性进行了评估，评估方法几乎涵盖了目前已知的所有分组密码分析方法，如差分密码分析、线性密码分析、不可能差分分析等等. 公开的评估结果表明，SM4 分组密码算法能够抵抗目前已知的所有攻击，拥有足够的安全冗余度.

1）差分密码分析

张蕾等人[12]首先给出了 SM4 的 21 轮差分分析，数据复杂度为 2^{118} 个选择明文，时间复杂度为 $2^{126.8}$ 次算法加密. 接着，张文涛等人[13]给出了 SM4 算法 18 轮差分特征，能够攻击到 22 轮，数据复杂度为 2^{117} 个选择明文，时间复杂度为 $2^{112.3}$ 次算法加密. 张美玲等人[14]也利用一些 18 轮的差分特征分析了 22 轮 SM4 算法，数据复杂度为 2^{117} 个选择明文，时间复杂度为 2^{123} 次算法加密，存储复杂度为 2^{112}. 2011 年，苏波展等人[15]找到了 19 轮的有效差分特征，将 SM4 的差分分析推进了 1 轮，达到 23 轮，攻击需要的数据复杂度为 2^{118} 个选择明文，时间复杂度为 $2^{126.7}$ 次加密，存储复杂度为 $2^{120.7}$.

2）线性密码分析

2008 年，Etrog 等人[16]给出了 SM4 的 22 轮线性分析结果，分析方法的数据复杂度为 $2^{118.4}$ 个已知明文，时间复杂度为 2^{117} 次算法加密. 2011 年，董晓丽[17]给出了 20 轮 SM4 的线性分析结果，需要的数据复杂度为 $2^{110.4}$ 个已知明文，时间复杂度为 $2^{106.8}$ 次算法加密，存储量为 2^{90}. 2014 年，Liu 等人[18]给出了 23 轮 SM4 算法的线性分析结果，时间复杂度为 2^{122} 次算法加密，数据复杂度为 $2^{126.54}$ 个已知明文，存储复杂度为 2^{116}.

3）多维线性密码分析

2010 年，Liu 等人[19]基于一条 5 轮的循环线性迹，构造了多条 18 轮的线性迹，可以攻击到 22 轮 SM4，需要的数据复杂度为 2^{112} 个已知明文，时间复杂度为 $2^{124.21}$ 次算法加密，存储量为 $2^{118.83}$. 同年，Cho 等人[20]给出了 23 轮 SM4 的线性分析，需要的数据复杂度为 $2^{126.7}$ 个已知明文，时间复杂度为 2^{127} 次算法加密，存储量为 $2^{120.7}$. 2014 年，Liu 等人[18]给出了 23 轮 SM4 算法的多维线性分析结果，时间复杂度为 $2^{122.7}$ 次算法加密，数据复杂度为 $2^{122.6}$ 个已知明文，存储复杂度为 $2^{120.6}$.

4）不可能差分密码分析

Lu 等人[21]首先给出了 SM4 的 16 轮不可能差分分析，需要的数据量为 2^{105} 个选择明文，时间复杂度为 2^{107} 次加密. Toz 等人[22]对文献[21]中的结果进行了修正，攻击轮数不变，但是修正后的数据复杂度为 $2^{117.06}$ 个选择明文，时间复杂度达到了 $2^{132.06}$ 内存查询. 注意到，该复杂度的值已经超越了 2^{128} 这个限值，如果 16 轮 SM4 的加密能够在 16 次内存查询中完成，那么这个攻击结果就失效了. 2010 年，Wang[23]将 SM4 的不可能差分分析做到了 17 轮，需要的数据复杂度为 2^{117} 个选择明文，时间复杂度为 2^{132} 次内存查询.

5）零相关线性分析

文献[24]给出了 14 轮 SM4 算法的多维零相关线性分析结果，需要的数据复杂度为 $2^{123.5}$ 个已知明文，时间复杂度为 $2^{120.7}$ 次算法加密，存储复杂度为 2^{73} 个分组长度.

6）积分密码分析

Liu 等人[25]给出了 SM4 的 13 轮积分分析，需要的数据复杂度为 2^{16} 个选择明文，时间复杂度为 2^{114} 次加密. 钟名富等人[26]构造了 12 轮积分区分器，能够攻击 14 轮 SM4 算法，需要的数据复杂度为 2^{32} 个选择明文，时间复杂度为 $2^{96.5}$ 次算法加密.

7）代数攻击

Ji 等人[27]和 Erickson 等人[28]分别使用了 XL

方法、Groebner 基方法以及 SAT 方法等不同的代数方法对 SM4 抵抗代数攻击的能力进行了评估，结果表明 SM4 在代数攻击下是安全的. 特别地，在 XL 方法下，SM4 抗代数攻击的能力比 AES 还要好.

8) 矩阵攻击

2007 年，Lu 等人[21]给出了 SM4 算法 14 轮的矩阵攻击，需要的数据复杂度为 $2^{121.82}$ 个选择明文，时间复杂度为 $2^{116.66}$ 次算法加密. 2008 年，Toz 等人在文献[22]中进一步降低了文献[21]中给出的矩阵攻击的数据和时间复杂度，最终的数据复杂度为 $2^{106.89}$ 个选择明文，时间复杂度为 $2^{107.89}$ 次算法加密. 同年，张蕾等人在文献[12]中给出了 SM4 算法 16 轮的矩阵攻击，需要的数据复杂度为 2^{125} 个选择明文，时间复杂度为 2^{116} 次算法加密. 2012 年，薛萍在其硕士论文[29]中给出了 SM4 算法 16 轮的矩形区分器，可以攻击到 18 轮，需要的数据量为 2^{127} 个选择明文，时间复杂度为 $2^{110.77}$ 次加密，存储量为 2^{130}. 此外，魏航等人[30]结合差分分析和代数攻击，对 20 轮 SM4 算法进行分析，发现其效果比直接用差分分析攻击 20 轮 SM4 算法的效果略好.

9) 抗差分及线性密码分析的可证明安全性

SM4 结构与之前的 Feistel 结构、SP 结构等相比，是一种新型的结构. 文献[31]证明了 SM4 型的非平衡 Feistel 结构是伪随机的. 针对 SM4 非平衡 Feistel 结构，有学者从结构上分析了其抗差分和抗线性分析的能力. 在分支数为 5 的 SP 型轮函数下，文献[32]证明了 27 轮 SM4 算法至少保证 22 个活跃的 S 盒. 因此，SM4 算法是抗差分安全的. 文献[33]则分析了 SM4 型算法抗线性分析的能力.

10) 抗相关密钥差分密码分析的可证明安全性

相关密钥差分分析与加密算法、密钥编排都有关. 在进行相关密钥攻击时，攻击者同时在密钥和消息 2 个地方引入差分. 孙思维等人[34]提出的自动化差分路线搜索的方法是一种基于 MILP (mixed-integer linear pragrarnrning)的方法，可以评估分组密码在(相关密钥)差分攻击下的安全界. 针对 SM4，孙思维等人用整数规划方法(MILP)给出了该算法抵抗相关密钥差分攻击的安全性分析结果. 表 3 是相关密钥下的 SM4 各轮的最少活跃 S 盒个数分布情况：

表 3　SM4 相关密钥自动化差分活跃 S 盒个数分布情况

轮数	3	4	5	6	7	8	9	10	11	12	13
S盒数	0	1	2	2	4	6	8	9	10	≤14	—

由于 SM4 的 S 盒差分概率是 2^{-6}，当最小活跃 S 盒的个数达到 22 时差分路径的概率为 2^{-132}，攻击难度高于穷举攻击. 因此，可以用最小活跃 S 盒个数是否达到 22 作为衡量是否抵抗差分攻击的一个标准. 在相关密钥攻击下，当密钥的差分非零时，32 轮的 SM4 算法的最小活跃 S 盒个数是 27 个. 因此 32 轮的 SM4 分组密码算法能够抵抗相关密钥差分攻击.

3.2　国际标准分组算法的安全性分析现状

关于 ISO/IEC 18033-3:2010 标准中的分组密码算法，近年来出现了许多重要的分析进展. 对 MISTY1，HIGHT，AES 3 个分组密码算法，已经找到了全轮的攻击，即找到了比穷搜密钥更有效的分析方法. 对 MISTY1，CAST-128，HIGHT，AES，Camellia，SEED 等算法目前攻击轮数最多的攻击如表 4 所示：

表 4　若干国际分组密码算法标准的攻击研究进展

算法名称	攻击轮数	攻击方法	数据复杂度	时间复杂度	参考文献
MISTY1	full	Integral	$2^{63.994}$ 2^{64}	$2^{107.9}$ $2^{69.5}$	[35] [36]
CAST-128	8(16)	Meet-in-the-middle	8	2^{118}	[37]
HIGHT	full	Related-key Rectangle	$2^{57.84}$	$2^{125.833}$	[38]
AES-128	7(10) full	Square Biclique	$2^{128}\sim2^{119}$ 2^{88}	2^{120} $2^{126.18}$	[39] [40]
AES-192	full	Biclique Related-key Boomerang	2^{80} 2^{123}	$2^{189.74}$ 2^{176}	[40] [41]
AES-256	full	Biclique Related-key Boomerang	2^{40} $2^{99.5}$	$2^{254.42}$ $2^{99.5}$	[40] [41]
Camellia-128 Camellia-192 Camellia-256	11(18) 12(24) 14(24)	Impossible Differential	$2^{92.4}$ $2^{119.7}$ 2^{118}	$2^{118.43}$ $2^{161.06}$ 2^{220}	[42]
SEED	9(16)	Differential	2^{125}	$2^{126.36}$	[43]

3.3 SM4 算法安全性结论

SM4 分组密码算法最好分析结果如表 5 所示：

表 5 SM4 分组密码算法目前的最好分析结果

攻击方法	攻击轮数	时间复杂度	数据复杂度	存储复杂度	参考文献
差分攻击	23	$2^{126.7}$	2^{118}	$2^{120.7}$	[15]
线性攻击	23	2^{122}	$2^{126.54}$	$2^{120.7}$	[18]
多维线性攻击	23	$2^{122.7}$	$2^{122.6}$	$2^{120.6}$	[18]
不可能差分攻击	17	2^{132}	2^{117}	—	[23]
零相关线性攻击	14	$2^{120.7}$	$2^{123.5}$	2^{73}	[24]
积分攻击	14	$2^{96.5}$	2^{32}	—	[26]
矩形攻击	18	$2^{96.5}$	2^{32}	—	[29]

在差分密码分析方面，对 SM4 算法最好的分析结果是苏波展等人[15]给出的攻击结果. 该文献的思路是手动推导活跃 S 盒个数的差分模式，通过低轮数的拼接推导得出 19 轮的区分器，利用 19 轮的区分器进行扩展攻击到 23 轮.

在线性密码分析方面，对 SM4 算法最好的分析结果是 Liu 等人[18]给出的攻击，他们利用一条 19 轮的线性逼近构造了一个 23 轮的线性攻击，数据复杂度要优于同样轮数的多维线性攻击. 该攻击方案的时间复杂度为 2^{122} 次算法加密，数据复杂度为 $2^{126.54}$ 个已知明文，存储复杂度为 2^{116}.

在多(维)线性分析方面，对 SM4 算法最好的分析结果是 Liu 等人[18]给出的 23 轮 SM4 算法的多维线性分析结果，时间复杂度为 $2^{122.7}$ 次算法加密，数据复杂度为 $2^{122.6}$ 个已知明文，存储复杂度为 $2^{120.6}$.

在不可能差分分析方面，对 SM4 算法最好的分析结果是 Wang[23]将 SM4 的不可能差分分析做到了 17 轮，需要的数据复杂度为 2^{117} 个选择明文，时间复杂度为 2^{132} 次内存查询.

在零相关线性分析方面，对 SM4 算法最好的分析结果是马猛等人[24]给出的 14 轮 SM4 算法的多维零相关线性分析结果，需要的数据复杂度为 $2^{123.5}$ 个已知明文，时间复杂度为 $2^{120.7}$ 次算法加密，存储复杂度为 2^{73} 个分组长度.

在积分分析方面，对 SM4 算法最好的分析结果是钟名富等人[26]构造的 12 轮积分区分器，能够攻击 14 轮 SM4 算法，需要的数据复杂度为 2^{32} 个选择明文，时间复杂度为 $2^{96.5}$ 次算法加密.

在矩阵分析方面，对 SM4 算法最好的分析结果是薛萍在其硕士论文[29]中给出的 SM4 算法 16 轮的矩形区分器，可以攻击到 18 轮，需要的数据量为 2^{127} 个选择明文，时间复杂度为 $2^{110.77}$ 次加密，存储量为 2^{130}.

在差分密码分析和代数攻击结合分析方面，对 SM4 算法最好的分析结果是魏航等人[30]对 20 轮 SM4 算法进行的分析.

从公开的研究结果可以看出，目前还没有一种分析方法能够在理论上攻破 24 轮的 SM4 算法. 因此，从传统的分析方法来看，SM4 算法具有较强的安全冗余度. 尤其是对比 MISTY1，AES 等已有全轮攻击方案的分组密码算法，SM4 算法具备一定的安全性优势.

4 小　结

本文介绍了 SM4 分组密码算法的算法流程、设计原理、算法特点和安全性分析进展. 在算法设计方面，SM4 算法体制基于正形置换设计构造，具有明确的理论特性，算法结构简洁清晰，易于理解、易于实现. 在安全性方面，目前尚未有研究表明有任何攻击方法可以威胁 SM4 算法的安全性.

参 考 文 献

[1] Shannon C E. Communication theory of secret system [J]. Bell System Technical Joural, 1949, 28(4): 656-715

[2] National Institute of Standard Technology. FIPS 46-3 Data Encryption Standard [S]. Gaithersburg: Federal Information Processing Standard, 1977

[3] NIST. AES 计划主页[EB/OL]. [2016-11-04]. http://csrc.nist.gov/encryption/aes/

[4] NIST. NESSIE 计划主页[EB/OL]. [2016-11-04]. http://www.cryptonessie.org

[5] 国家密码管理局. 国家密码管理局公告(第 7 号) [EB/OL]. [2016-11-04]. http://www.oscca.gov.cn/News/200709/News_1105.htm

[6] 吕述望，李大为，张超，等. GM/T 0002—2012 SM4 分组密码算法[S]. 北京：中国标准出版社，2012

[7] 中国标准化委员会. GB/T 32907—2016 信息安全技术 SM4 分组密码算法[S]. 北京：中国质检出版社，2016

[8] 吕述望. 完全映射及其密码学应用[M]. 北京：中国科学技术大学出版社，2008

[9] 刘佳，韦宝典，戴宪华. SMS4 算法 S 盒的密码学性质[J]. 计算机工程，2008，34(5)：158-160

[10] 吴文玲，冯登国，张文涛. 分组密码的设计与分析[M]. 2 版. 北京：清华大学出版社，2009

[11] 陈华. 密码算法的安全性检测及关键组件的设计[D]. 北京：中国科学院软件研究所，2004

[12] Zhang Lei, Zhang Wentao, Wu Wenling. Cryptanalysis of reduced-round SM4 block cipher [C] //Proc of ACISP2008. Berlin: Springer, 2008: 216-229

[13] Zhang Wentao, Wu Wenling, Feng Dengguo, et al. Some new observations on the SM4 block cipher in the Chinese WAPI standard [C] //Proc of Information Security Practice and Experience(ISPEC). Berlin: Springer, 2009: 324-335

[14] 张美玲，刘景美，王新梅. 22-轮 SM4 的差分分析[J]. 中山大学学报：自然科学版，2010，49(2)：43-47

[15] Su Bozhan, Wu Wenling, Zhang Wentao. Security of the SM4 block cipher against differential cryptanalysis [J]. Journal of Computer Science and Technology, 2001, 26 (1): 130-138

[16] Etrog J, Robshaw Matt J B. The cryptanalysis of reduced-round SM4 [C] //Proc of SAC2009. Berlin: Springer, 2009: 51-65

[17] 董晓丽. 分组密码 AES 和 SM4 的安全性分析[D]. 西安：西安电子科技大学，2011

[18] Liu Mingjie, Chen Jiazhe. Improved linear attacks on the chinese block cipher standard [J]. Journal of Computer Science and Technology, 2014, 29(6): 1123-1133

[19] Liu Zhiqiang, Gu Dawu, Zhang Jing. Multiple linear cryptanalysis of reduced-round SM4 block cipher [J]. Chinese Journal of Electronics, 2010, 19(3): 389-393

[20] Cho J Y, Nyberg K. Improved linear cryptanalysis of SM4 block cipher [C] //Proc of the 9th Int Workshop on Symmetric Key Encryption Workshop(SKEW). Vienna, Austria: SKEW, 2010

[21] Lu Jiqiang. Attacking reduced-round versions of the SM4 block cipher in the Chinese WAPI standard [C] //Proc of ICICS2007. Berlin: Springer, 2007: 306-318

[22] Toz D, Dunkelman O. Analysis of two attacks on reduced-round versions of the SM4 [C] //Proc of ICICS 2008. Berlin: Springer, 2008: 141-156

[23] Wang Gaoli. Improved impossible differential cryptanalysis on SM4 [C] //Proc of Int Conf on Communications and Intelligence Information Security. Piscataway, NJ: IEEE, 2010: 105-108

[24] 马猛，赵亚群，刘庆聪，等. SM4 算法的多维零相关线性分析[J]. 密码学报，2015，2(5)：458-466

[25] Liu Fen, Ji Wen, Hu Lei, et al. Analysis of the SM4 block cipher [C] //Proc of ACISP 2007. Berlin: Springer, 2007: 158-170

[26] 钟名富，胡予濮，陈杰. 分组密码算法 SM4 的 14 轮 Square 攻击[J]. 西安电子科技大学学报：自然科学版，2008，35(1)：105-109

[27] Ji Wen, Hu Lei, Ou Haiwen. Algebraic attack to SM4 and the comparison with AES [C] //Proc of the 5th Int Conf on Information Assurance and Security. Piscataway, NJ: IEEE, 2009: 662-665

[28] Erickson J, Ding J, Christensen C. Algebraic cryptanalysis of SM4: Groebner basis attack and SAT attack compared [C] //Proc of ICISC2009. Berlin: Springer, 2010: 73-86

[29] 薛萍. 对分组密码算法 SM4 的矩形攻击[D]. 济南：山东大学，2012

[30] 魏航，崔会丽，吕晓庆. SM4 分组密码算法的差分-代数分析[J]. 成都大学学报：自然科学版，2012，31(2)：158-160

[31] 张立廷，吴文玲. 使用压缩函数的非平衡 Feistel 结构的伪随机性和超伪随机性[J]. 计算机学报，2009，32(7)：1320-1330)

[32] Zhang Meiling, Liu Yuanhua, Liu Jingmei. Practically secure against differential cryptanalysis for block cipher SM4 [J]. American Journal of Engineering and Technology Research, 2011, 11(12): 1923-1928

[33] Zhang Bin, Jin Chenhui. Practical security against linear cryptanalysis for SM4-like ciphers with SP round function [J]. Science China: Information Sciences, 2012, 55(9): 2161-2170

[34] Sun Siwei, Hu Lei, Wang Peng, et al. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block cipher [C] //Proc of ASIACRYPT(2014). Berlin: Springer, 2014: 158-178

[35] Todo Y. Integral cryptanalysis on full MISTY1 [C] //Proc of Advances in Cryptology—CRYPTO 2015. Berlin: Springer, 2015: 413-432

[36] Bar-On A, Keller N. A 2^{70} attack on the full MISTY1 [C] //Proc of Advances in Cryptology—CRYPTO 2016. Berlin: Springer, 2016: 435-456

[37] Isobe T, Shibutani K. Generic key recovery attack on feistel scheme [C] //Proc of Part I of the 19th Int Conf on Advances in Cryptology (ASIACRYPT2013). Berlin: Springer, 2013: 464-485

[38] Bonwook K, Hong D, Kwon D. Related-key attack on the full HIGHT [C] //Proc of the 13th Int Conf on Information Security and Cryptology (ICISC'10). Berlin: Springer, 2011: 49-67

[39] Ferguson N, Kelsey J, Lucks S, et al. Improved cryptanalysis of Rijndael [C] //Proc of the 7th Int Workshop on Fast Software Encryption Fse. Berlin: Springer, 2001: 213-230

[40] Andrey B, Khovratovich D, Rechberger C. Biclique cryptanalysis of the full AES [C] //Proc of ASIACRYPT 2011. Berlin: Springer, 2011: 344-371

[41] Biryukov A, Khovratovich D. Related-key cryptanalysis of the full AES-192 and AES-256 [C] //Proc of Advances in Cryptology—ASIACRYPT 2009. Berlin: Springer, 2009: 1-18

[42] Christina B, Naya-Plasencia M, Suder V. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon [C] // Proc of ASIACRYPT 2014. Berlin: Springer, 2014: 179-199

[43] Lu Jiqiang, Yap W, Henricksen M, et al. Differential attack on nine rounds of the SEED block cipher [J]. Information Processing Letters, 2014, 114(3): 116-123

吕述望
教授，博士生导师，主要研究方向为密码学、知识安全、信息安全.
swlu@ustc.edu.cn

苏波展
博士，研究员，主要研究方向为密码学.
subozhan@163.com

王 鹏
博士，副研究员，主要研究方向为对称密码算法的设计与分析.
wp@is.ac.cn

毛颖颖
硕士，工程师，主要研究方向为密码算法检测及密码产品检测.
maoyy2000@163.com

霍利利
硕士，工程师，主要研究方向为密码学.
lily.home.hao@163.com

附录 A.

SM4 分组密码算法源代码示例：

```
//
#define SM4_Rotl32(buf, n) (((buf)<<
  n)|((buf)>>(32-n)))

unsigned int SM4_CK[32]={0x00070E15,
0x1C232A31, 0x383F464D, 0x545B6269,
0x70777E85, 0x8C939AA1, 0xA8AFB6BD,
0xC4CBD2D9, 0xE0E7EEF5, 0xFC030A11,
0x181F262D, 0x343B4249, 0x50575E65,
0x6C737A81, 0x888F969D, 0xA4ABB2B9,
0xC0C7CED5, 0xDCE3EAF1, 0xF8FF060D,
0x141B2229, 0x30373E45, 0x4C535A61,
0x686F767D, 0x848B9299, 0xA0A7AEB5,
0xBCC3CAD1, 0xD8DFE6ED, 0xF4FB0209,
0x10171E25, 0x2C333A41, 0x484F565D,
0x646B7279};

unsigned char SM4_Sbox[256]={0xD6,
0x90, 0xE9, 0xFE, 0xCC, 0xE1, 0x3D, 0xB7,
0x16, 0xB6, 0x14, 0xC2, 0x28, 0xFB, 0x2C,
0x05, 0x2B, 0x67, 0x9A, 0x76, 0x2A, 0xBE,
0x04, 0xC3, 0xAA, 0x44, 0x13, 0x26, 0x49,
0x86, 0x06, 0x99, 0x9C, 0x42, 0x50, 0xF4,
0x91, 0xEF, 0x98, 0x7A, 0x33, 0x54, 0x0B,
0x43, 0xED, 0xCF, 0xAC, 0x62, 0xE4, 0xB3,
0x1C, 0xA9, 0xC9, 0x08, 0xE8, 0x95, 0x80,
0xDF, 0x94, 0xFA, 0x75, 0x8F, 0x3F, 0xA6,
0x47, 0x07, 0xA7, 0xFC, 0xF3, 0x73, 0x17,
0xBA, 0x83, 0x59, 0x3C, 0x19, 0xE6, 0x85,
0x4F, 0xA8, 0x68, 0x6B, 0x81, 0xB2, 0x71,
0x64, 0xDA, 0x8B, 0xF8, 0xEB, 0x0F, 0x4B,
0x70, 0x56, 0x9D, 0x35, 0x1E, 0x24, 0x0E,
0x5E, 0x63, 0x58, 0xD1, 0xA2, 0x25, 0x22,
0x7C, 0x3B, 0x01, 0x21, 0x78, 0x87, 0xD4,
0x00, 0x46, 0x57, 0x9F, 0xD3, 0x27, 0x52,
0x4C, 0x36, 0x02, 0xE7, 0xA0, 0xC4, 0xC8,
0x9E, 0xEA, 0xBF, 0x8A, 0xD2, 0x40, 0xC7,
0x38, 0xB5, 0xA3, 0xF7, 0xF2, 0xCE, 0xF9,
0x61, 0x15, 0xA1, 0xE0, 0xAE, 0x5D, 0xA4,
0x9B, 0x34, 0x1A, 0x55, 0xAD, 0x93, 0x32,
0x30, 0xF5, 0x8C, 0xB1, 0xE3, 0x1D, 0xF6,
0xE2, 0x2E, 0x82, 0x66, 0xCA, 0x60, 0xC0,
0x29, 0x23, 0xAB, 0x0D, 0x53, 0x4E, 0x6F,
0xD5, 0xDB, 0x37, 0x45, 0xDE, 0xFD, 0x8E,
0x2F, 0x03, 0xFF, 0x6A, 0x72, 0x6D, 0x6C,
0x5B, 0x51, 0x8D, 0x1B, 0xAF, 0x92, 0xBB,
```

```
0xDD, 0xBC, 0x7F, 0x11, 0xD9, 0x5C, 0x41,
0x1F, 0x10, 0x5A, 0xD8, 0x0A, 0xC1, 0x31,
0x88, 0xA5, 0xCD, 0x7B, 0xBD, 0x2D, 0x74,
0xD0, 0x12, 0xB8, 0xE5, 0xB4, 0xB0, 0x89,
0x69, 0x97, 0x4A, 0x0C, 0x96, 0x77, 0x7E,
0x65, 0xB9, 0xF1, 0x09, 0xC5, 0x6E, 0xC6,
0x84, 0x18, 0xF0, 0x7D, 0xEC, 0x3A, 0xDC,
0x4D, 0x20, 0x79, 0xEE, 0x5F, 0x3E, 0xD7,
0xCB, 0x39, 0x48};

    unsigned int SM4_FK[4]={0xA3B1BAC6,
0x56AA3350, 0x677D9197, 0xB27022DC};

    /* * * * * * * * * * * * * * * * * * * *
    Function:
      void SM4_KeySchedule(unsigned char
MK[], unsigned int rk[]);
    Description:
      Generate round keys
    Calls:
    Called By:
      SM4_Encrypt;
      SM4_Decrypt;
    Input:
      MK[]: Master key
    Output:
      rk[]: round keys
    Return:null
    Others:
    * * * * * * * * * * * * * * * * * * * */
    void SM4_KeySchedule(unsigned char MK
[],unsigned int rk[])
    {
      unsigned int tmp,buf,K[36];
      int i;
      for(i=0;i<4;i++)
      {
      K[i]=SM4_FK[i]^((MK[4*i]<<24)
            |(MK[4*i+1]<<16)
            |(MK[4*i+2]<<8)
            |(MK[4*i+3]) );
      }
      for (i=0;i<32;i++)
      {
        tmp=K[i+1]^K[i+2]^K[i+3]^
        SM4_CK[i];

      //nonlinear operation
      buf=(SM4_Sbox[(tmp>>24) &
      0xFF])<<24
      |(SM4_Sbox[(tmp>>16) & 0xFF])
      <<16
      |(SM4_Sbox[(tmp>>8) & 0xFF])<<8
      |(SM4_Sbox[tmp & 0xFF]);

      //linear operation
      K[i+4]=K[i]^((buf)^(SM4_Rotl32
      ((buf),13))^(SM4_Rotl32((buf),23)));

      rk[i]=K[i+4];
      }
    }

    /* * * * * * * * * * * * * * * * * * * *
    Function:
      void SM4_Encrypt(unsigned char MK[],
unsigned char PlainText [], unsigned char
CipherText[]);
    Description:
      Encryption function
    Calls:
      SM4_KeySchedule
    Called By:
    Input:
      MK[]: Master key
      PlainText[]: input text
    Output:
      CipherText[]: output text
    Return:null
    Others:
    * * * * * * * * * * * * * * * * * * * */
    void SM4_Encrypt(unsigned char MK[],
unsigned char PlainText [], unsigned char
CipherText[])
```

```
{
  unsigned int rk[32],X[36],tmp,buf;
  int i,j;
  SM4_KeySchedule(MK,rk);

  for(j=0;j<4;j++)
  {
    X[j]=(PlainText[j*4]<<24)
    |(PlainText[j*4+1]<<16)
    |(PlainText[j*4+2]<<8)|
    (PlainText[j*4+3]);
  }

    for(i=0;i<32;i++)
  {
      tmp=X[i+1]^X[i+2]^X[i+3]^
      rk[i];

      //nonlinear operation
       buf=( SM4_Sbox[(tmp>>24) &
       0xFF])<<24
       |(SM4_Sbox[(tmp>>16) &
       0xFF])<<16
       |(SM4_Sbox[(tmp>>8) & 0xFF])
       <<8
       |(SM4_Sbox[tmp & 0xFF]);

      //linear operation
       X[i+4]=X[i]^(buf^SM4_Rotl32
       ((buf),2)
^SM4_Rotl32((buf),10)
^SM4_Rotl32((buf),18)
^SM4_Rotl32((buf),24));
  }

    for(j=0;j<4;j++)
  {
      CipherText[4*j]=(X[35-j]>>
      24)& 0xFF;
      CipherText[4*j+1]=(X[35-j]>
      >16)& 0xFF;
      CipherText[4*j+2]=(X[35-j]>
      >8)& 0xFF;
      CipherText[4*j+3]=(X[35-j])&
      0xFF;
  }
}

/*********************
Function:
  void SM4_Decrypt(unsigned char MK[],
unsigned char CipherText [], unsigned char
PlainText[]);
Description:
  Decryption function
Calls:
  SM4_KeySchedule
Called By:
Input:
  MK[]: Master key
  CipherText[]: input text
Output:
  PlainText[]: output text
Return:null
Others:
*********************/
void SM4_Decrypt(unsigned char MK[],
unsigned char CipherText [], unsigned char
PlainText[])
{
  unsigned int rk[32],X[36],tmp,buf;
  int i,j;

  SM4_KeySchedule(MK,rk);

  for(j=0;j<4;j++)
  {
    X[j]=(CipherText[j*4]<<24)|
    (CipherText[j*4+1]<<16)|
    (CipherText[j*4+2]<<8)|
    (CipherText[j*4+3]);
  }

  for(i=0;i<32;i++)
```

```
    {
      tmp=X[i+1]^X[i+2]^X[i+3]^
      rk[31-i];

      //nonlinear operation
buf=(SM4_Sbox[(tmp>>24) & 0xFF])
<<24
|(SM4_Sbox[(tmp>>16) & 0xFF])<<16
|(SM4_Sbox[(tmp>>8) & 0xFF])<< 8
|(SM4_Sbox[tmp & 0xFF]);
//linear operation
X[i+4]=X[i]^(buf^SM4_Rotl32((buf),2)^
SM4_Rotl32((buf),10)^SM4_Rotl32((buf),
18)^ SM4_Rotl32((buf),24));
}

for(j=0;j<4;j++)
{
  PlainText[4 * j]=(X[35-j]>>24)&
  0xFF;
  PlainText[4 * j+1]=(X[35-j]>>16)
  & 0xFF;
  PlainText[4 * j+2]=(X[35-j]>>8)&
  0xFF;
  PlainText[4 * j+3]=(X[35-j]) &
  0xFF;
}
}
```

附录 B.

SM4 分组密码算法对一组明文进行加密的运算示例.

输入明文：01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10；

输入密钥：01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10；

输出密文：68 1E DF 34 D2 06 96 5E 86 B3 E9 4F 53 6E 42 46.