第5章 高级加密标准

高级加密标准(Advanced Encryption Standard,AES)是目前被全世界广泛采用的一种加密标准,它在2001年取代了DES成为新的加密标准。相比DES,AES具有更长的密钥长度和分组长度,并且更加侧重于软件运行效率,已然成为对称密钥加密中最流行的算法之一。AES加密和解密过程使用两套算法,是一对互逆操作,在变换过程中,交替使用代换和置换两种操作以达到加密效果。就目前情况来看,AES具有较高的安全性。

5.1 高级加密标准的起源

随着计算能力的突飞猛进,已经超期服役的DES显得力不从心,在1999年NIST发布的一个新版本的DES标准指出,DES仅能用于遗留的系统,同时3DES将取代DES成为新的标准。3DES提高了密钥长度,并且由于是基于DES的,安全性得以保证。如果仅考虑算法安全,3DES的确成为了最近二十多年的一个加密算法标准[21,22,26]。

但是3DES的首要缺点在于软件实现该算法的速度较慢。起初DES是为20世纪70年代中期的硬件实现所设计的,难以由软件高效地实现。在3DES中轮的数量DES的三倍,故其效率更低。另一个缺点是DES和3DES的分组长度均为64比特。就效率和安全性而言,分组长度应更长[29]。

由于这些缺点,NIST在1997年公开征集新的高级加密标准AES,要求安全性能不低于3DES,同时应具有更好的运行性能。除了这些一般的要求之外,NIST特别提出了高级加密标准必须是分组长度为128比特的对称分组密码,并能支持长度为128比特、192比特和256比特的密钥[34-36]。

1998年8月12日,在首届AES候选方案会议上公布了AES的15个候选算法,任由全世界各机构和个人攻击和评论,这15个候选算法是CAST256、CRYPTON、E2、DEAL、FROG、SAFER+、RC6、MAGENTA、LOKI97、SERPENT、MARS、Rijndael、DFC、Twofish、HPC。

1999年3月,在第2届AES候选方案会议上,经过对全球各密码机构和个人对候选算法分析结果的讨论,NIST从15个候选算法中初选出了5个,分别是RC6、Rijndael、SERPENT、Twofish和MARS。

2000 年 4 月 13 日至 14 日,NIST 召开了第 3 届 AES 候选方案会议,继续对最后 5 个候选算法进行讨论。

2000 年 10 月 2 日,NIST 宣布 Rijndael 作为新的 AES。

至此,经过 3 年多的讨论,Rijndael 终于脱颖而出成为高级加密标准,但是在成为标准的过程中,NIST 也对其进行了一些修改。

Rijndael 并非 AES

严格地说,AES 和 Rijndael 加密方法并不完全一样(虽然在实际应用中二者可以互换),因为 Rijndael 加密方法可以支持更大范围的分组和密钥长度[36](AES 的分组长度固定为 128 比特,密钥长度则可以是 128 比特、192 比特或 256 比特;而 Rijndael 使用的密钥和分组长度可以是 32 比特的整数倍,以 128 比特为下限,256 比特为上限)。加密过程中使用的密钥由 Rijndael 密钥生成方案产生。

5.2 代换置换网络结构

代换置换网络(substitution-permutation network,SPN)将数学运算应用于分组密码,如 AES、Square、SAFER 等。SPN 将输入的明文进行交替的代换操作和置换操作以产生密文,这些操作均可以使用异或、移位等操作进行,使其在软件和硬件上均可很好地实现[37]。每一轮使用的子密钥产生于输入的密钥,甚至在有些基于 SPN 算法的加密算法中,用于代换操作的 S 盒也是基于子密钥产生的。图 5-1 为一个使用 2 轮 SPN 结构的简单算法,其中单轮经过密钥异或、S 盒代换和 P 盒置换三个步骤。

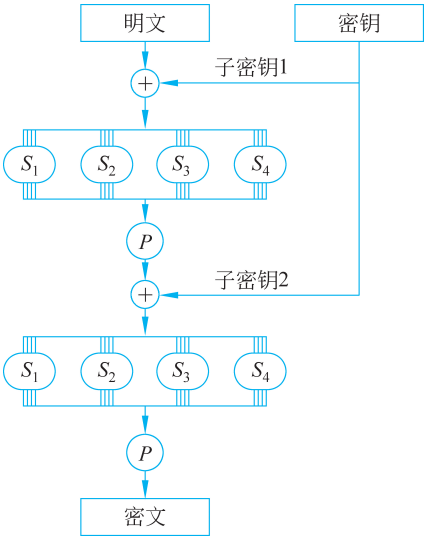


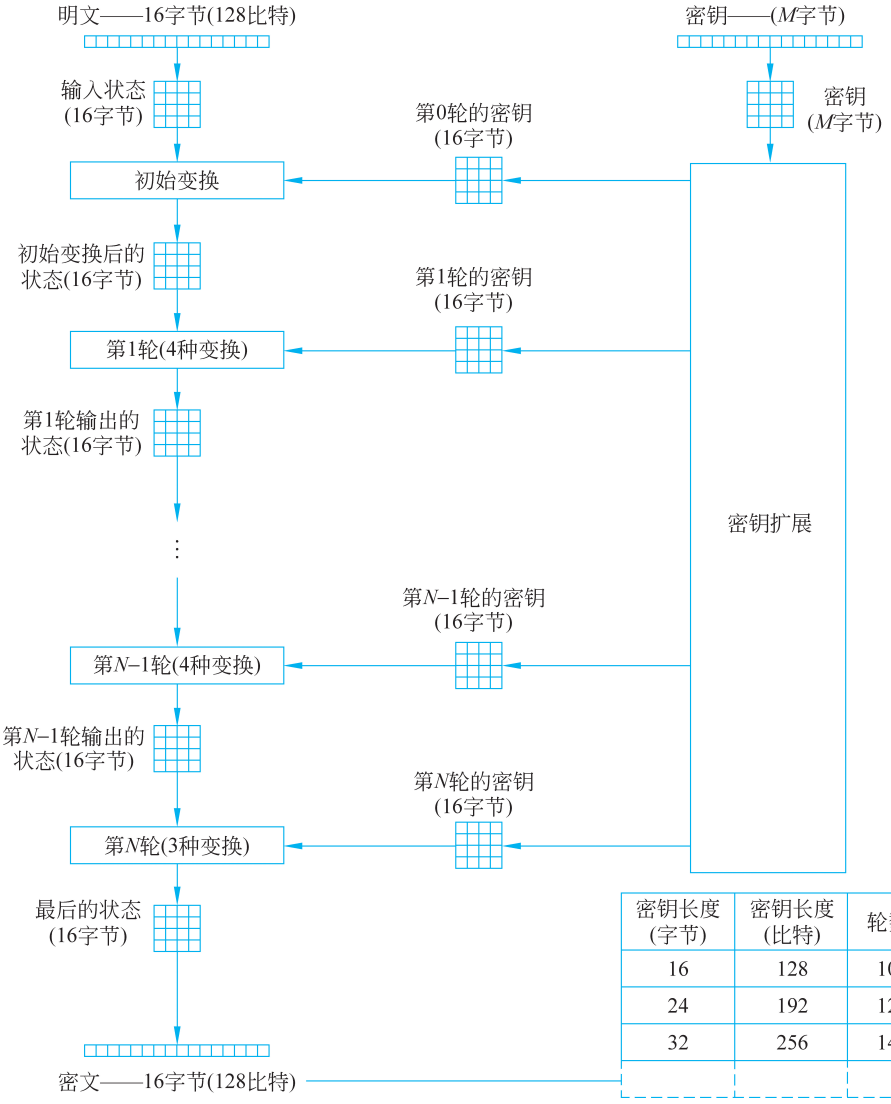
图 5-1 简单 SPN 网络

5.3 高级加密标准的结构

AES 使用 SPN 作为基础进行设计，保留了 SPN 的框架，但是在每轮 SPN 中添加了列混合的线性变换过程，并且指定了分组长度为 128 比特，但是密钥可以是 128 比特、192 比特或 256 比特[38]。

5.3.1 总体结构

图 5-2 展示了 AES 加密过程的总体结构。明文分组的长度为 128 比特，密钥可以为 3 种长度。根据密钥的长度，AES 算法被称为 AES-128、AES-192 和 AES-256，并且密钥



密钥长度(字节)	密钥长度(比特)	轮数
16	128	10
24	192	12
32	256	14

图 5-2　AES 加密过程

长度的改变也将导致 SPN 轮次的改变,具体数值见图 5-2 中的表格。后续小节将以 128 比特密钥 AES 作为示例进行讲解。

加密和解密算法的输入是一个 128 比特的分组,通常将这个分组描述为一个 4×4 的字节方阵,存储于状态数组,并在加密或解密的各个阶段被修改。而在整个 AES 算法中,变换状态分组的子算法有 7 个,分别为字节代换、逆字节代换、行移位、逆行移位、列混合、逆列混合和轮密钥加变换[39,40]。

128 究竟比 56 强多少?

假设可以制造一台可以在 1 秒内穷举破解 DES 密码的机器,那么使用这台机器破解一个 128 比特 AES 密码需要大约 149 万亿年的时间(更进一步比较,宇宙一般被认为存在了不到 300 亿年)。

5.3.2 详细结构

在 AES-128 的加密过程中,明文首先被复制入 state 数组中,进行一次初始变换(实际上是一次轮密钥加变换),紧接着进行 9 轮变换和最后第 10 轮的变换。在 9 轮变换中,每一轮均按照字节代换、行移位、列混合和轮密钥加这样的顺序进行。但是在最后一轮(即第 10 轮)变换中则只进行 3 种变换,即字节代换、行移位和轮密钥加。解密过程中密文同样被复制入一个状态数组并进行一次初始变换(使用加密第 10 轮密钥的轮密钥加变换),接着进行与加密过程对应的 9 轮变换和第 10 轮变换。在 9 轮变换中,每一轮均按照逆向行移位、逆向字节代换、轮密钥加和逆向列混合这样的顺序进行。在最后一轮,即第 10 轮变换中,只进行逆向行移位、逆向字节代换和轮密钥加。从图 5-3 可以看出,加密和解密是一组互逆过程,解密中的操作均是加密中对应操作的逆向过程,使得密文恢复成明文。

5.3.3 轮密钥加变换

轮密钥加变换是一个状态数组和子密钥异或的操作。如图 5-4 所示,参与运算的是状态数组和子密钥,变换结果由二者按比特异或运算得出。

轮密钥加变换非常简单,却能根据密钥去影响状态数组中的每一位。密钥扩展的复杂性和 AES 其他阶段(如 SPN 的重复操作)的复杂性共同确保了 AES 的安全性。

5.3.4 字节代换

字节代换和逆字节代换实际上就是一个 S 盒代换的过程。AES 中定义了两个 S 盒,即 S 盒和逆 S 盒,其中加密的字节代换使用 S 盒,解密的逆字节代换使用逆 S 盒。如表 5-1 和表 5-2 所示,S 盒和逆 S 盒均是由 16×16 字节组成的矩阵,包含了 8 比特所能表示的 256 个数的置换。状态数组中的每字节按照如下的方式代换为一个新的字节:把该字节的高 4 比特作为行值,低 4 比特作为列值,以这些行列值从 S 盒或逆 S 盒中索引到位置的元素作为输出。例如,在加密过程中,十六进制数$(EA)_{16}$所对应的行为 14,列为 10(从 0 开始),S 盒中在此位置的是$(87)_{16}$,则$(EA)_{16}$将被代换为$(87)_{16}$。

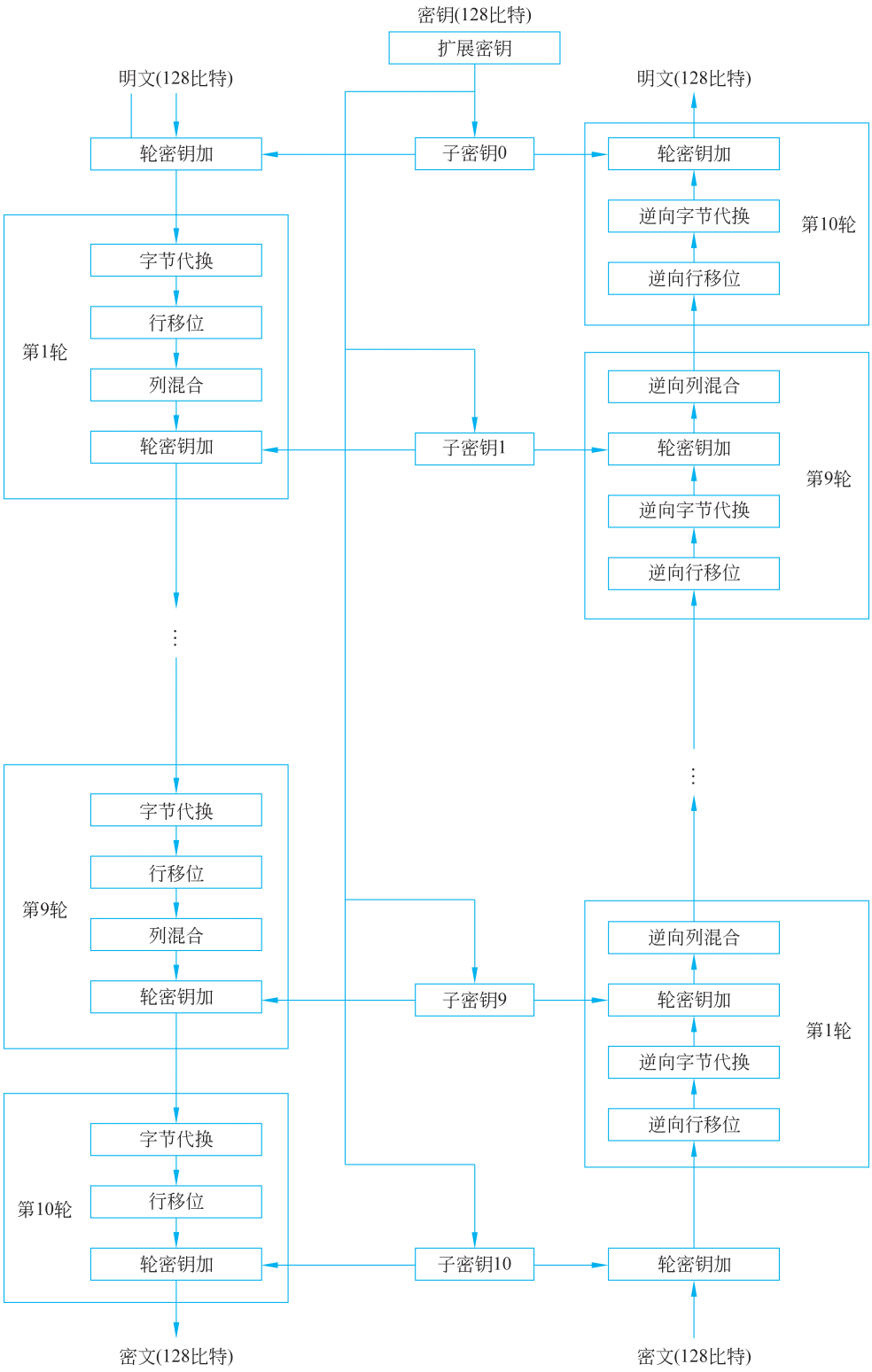


图 5-3　AES-128 加解密详细流程

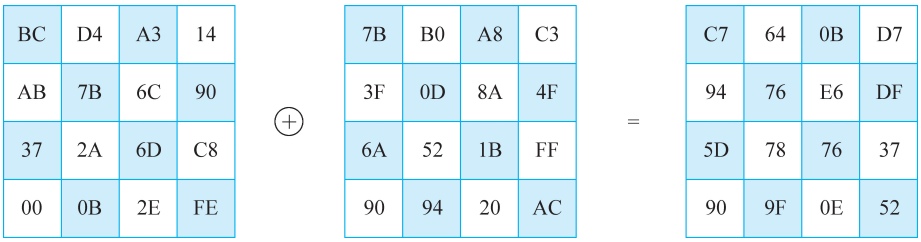


图 5-4　轮密钥加示意图

而在解密过程中,$(87)_{16}$对应的 8 行 7 列值为$(EA)_{16}$。实际上,同样的数据经过 S 盒变换后再经过逆 S 盒变换,即可得到原始的数据,所以字节代换和逆字节代换是一个互逆的过程。

表 5-1　S 盒

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

表 5-2　逆 S 盒

5C	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B

续表

3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

5.3.5　行移位

行移位和逆行移位如同其名称一样，是一个简单的移位过程。不过根据状态数组行标不同，每一行移动的位数也不同。如图 5-5 所示，按照大多数程序语言来说，在加密过程中，第 0 行循环左移 0 个字节，第 1 行循环左移 1 字节，第 2 行循环左移 2 字节，第 3 行循环左移 3 字节。而在解密过程中，就是这个操作的逆过程，即循环左移改变成循环右移。

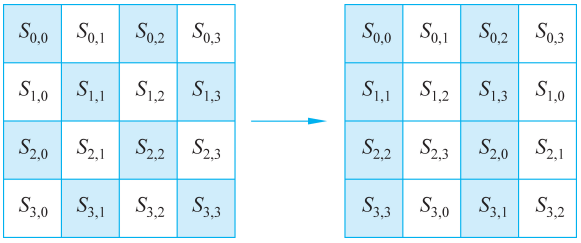


图 5-5　行移位示意图

在不同密钥长度的 AES 中，行移位的长度略不相同。在 AES-128 和 AES-192 中，行移位的偏移量是 0、1、2 和 3，而在 AES-256 中，行移位的偏移量是 0、1、3 和 4。

5.3.6　列混合

列混合是对每列独立地进行操作，每列中的每字节被映射为一个新的值，该值由该列中的 4 字节通过函数变换得到。该变换可以由下面所示的基于状态的矩阵乘法表示。

$$\begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

在逆向列混合中，和列混合一样，仅仅是与状态运算的矩阵不同，使用的矩阵如下所示。

$$\begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

行移位是算法对行进行变换,而列混合是算法对列进行变换,经过行移位和列混合的状态数组已经完全变样,从而使算法的安全性更好。

在列混合和逆向列混合中,乘积矩阵中的每个元素均是一行和一列中的对应元素的乘积之和。在这个矩阵运算中的乘法和加法都是被定义在有限域上的运算,且是基于 $GF(2^8)$的。关于 $GF(2^8)$上的乘法和加法可详见 5.5 节,如果读者对此没有兴趣而只想知道计算机中如何操作,那么可以跳过 5.5 节前面部分,直接阅读 5.5.6 节。

AES 的软件优化

使用 32 或更多比特寻址的系统可以事先对所有可能的输入创建对应表,利用查表实现字节代替、行移位和列混合步骤以达到加速的效果。这么做需要产生 4 个表,每个表都有 256 个格子,一个格子记载 32 比特的输出;约占 4KB(4096 字节)存储器空间,即每个表占 1KB 的存储器空间。如此一来,在每个加密循环中只需要查 16 次表,作 12 次 32 比特的异或运算,以及轮密钥加步骤中 4 次 32 比特异或运算。

若目标的平台存储器空间不足 4KB,那么也可以利用循环交换的方式一次查一个 256 格 32 比特的表。

5.3.7 密钥扩展算法

AES-128 中的密钥扩展算法的输入值是 16 字节,输出值是一个由 176 字节组成的移位线性数组。这足以为 AES-128 提供初始变换和其他 10 轮中的每一轮提供 16 字节的轮密钥。

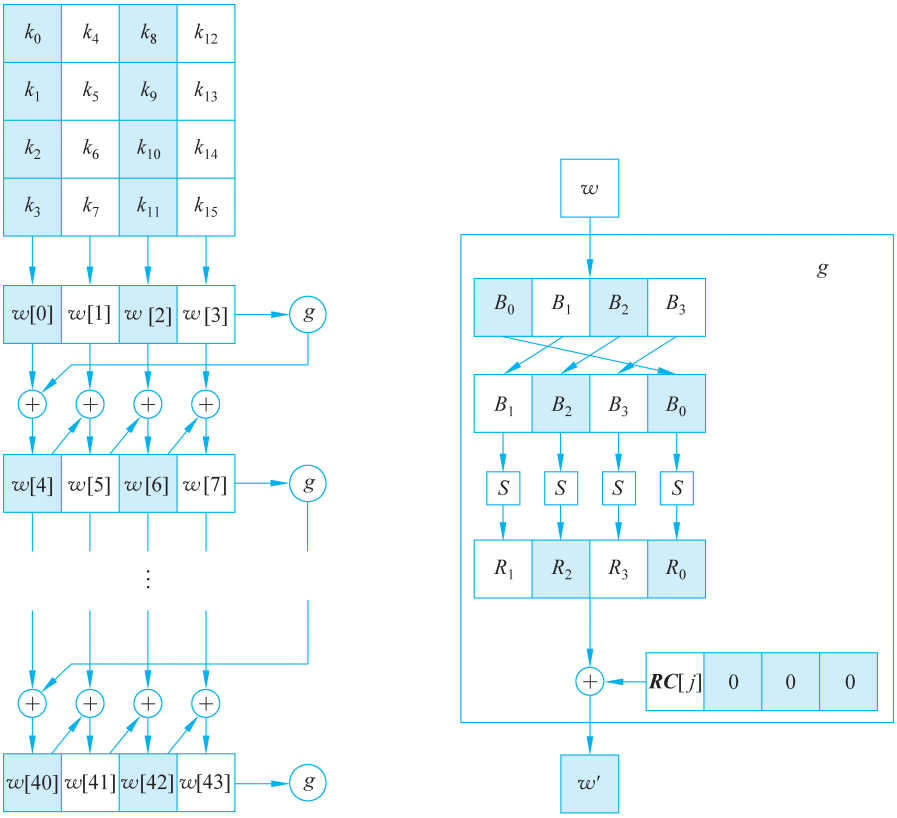
输入密钥被直接复制到扩展密钥数组的 4 个字(字长为 32 字节)。然后每次用 4 个字填充扩展密钥数组余下的部分。在扩展密钥数组中,每个新增的字 $w[i]$的值依赖 $w[i-1]$和 $w[i-4]$。在 4 个情形中有 3 个使用了异或运算。对 w 数组中索引为 4 的倍数的元素,采用了更复杂的函数计算。图 5-6(a)展示了密钥扩展算法的整体流程,图 5-6(b)展示了 w 数组中索引为 4 的倍数的元素所采用的函数 g。

函数 g 的输入为一个 4 字节的字,首先进行一次字移位,再对每个字按照表 5-1 进行一次 S 盒的代换,最后与轮常量 **Rcon**$[j]$相异或。轮密钥 **Rcon**$[j]$由 1 个 ***RC***$[j]$和 3 个 0 字节组成,其中 ***RC***$[j]$的值如下。

$$\boldsymbol{RC}[j]=\begin{cases} 1, & j=1 \\ 2*\boldsymbol{RC}[j-1], & j>1 \text{ 且 } \boldsymbol{RC}[j-1]<80_{16} \\ \boldsymbol{RC}[j]=2*\boldsymbol{RC}[j-1]+1\mathrm{B}_{16}, & j>1 \text{ 且 } \boldsymbol{RC}[j-1]>=80_{16} \end{cases}$$

同样,该过程中的乘法也是定义在 $GF(2^8)$上的。在 AES-128 中的 ***RC*** 值可以参考

表 5-3(十六进制表示)。



(a) 密钥扩展算法　　(b) g函数

图 5-6　AES 密钥扩展算法

表 5-3　AES-128 密钥扩展中的 *RC* 值

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	01	02	04	08	10	20	40	80	1B	36

5.4 AES 设计上的考虑

AES 作为新一代的标准替代了 DES 地位，必然在设计时考虑到一些已有的攻击[41]，同时还需要考虑一些可能存在的攻击方法。与 DES 相比，AES 在设计上有什么特点呢？

1. 扩散速度更快

AES 算法是一种 SPN 体制，与 DES 的 Feistel 体制的每次循环时有一半数据未被更改不同，AES 将数据中的所有比特同等对待，使输入比特的扩散影响更快，通常两轮循环就足以得到完全的扩散，即所有的 128 比特输出都完全依赖 128 比特输入。而这一扩散效果有一部分依赖行移位和列混合。

2. S 盒的构造方法

与 DES 的 S 盒构造方法存在神秘色彩不同,AES 的 S 盒构造使用一种简单而清晰的方法,这样就可以避免任何建立在算法上的陷门,从而使其得到广泛应用。

3. 抗攻击能力

AES 的 S 盒是基于有限域建立的,用它对抗差分分析和线性分析效果显著。AES-128 的轮数为 10 轮,这是因为较低的轮次会导致蛮力攻击成功。密钥扩展函数中的循环常量用于消除在循环过程中生成每个循环差别的对称性,并且使用 S 盒以防止攻击者在知道部分密钥的情况下发起攻击。而 128 比特的密钥相较于 DES 的 56 比特,防穷尽搜索的能力更强,且最大可支持 256 比特密钥,提供更高的防护级别[42]。但是随着计算能力的提高和攻击方式的优化,精心保护的 AES 也会存在安全隐患。例如,针对重用掩码 AES 算法的侧信道碰撞攻击在噪声为 0.0029 时,成功率能够达到 95%[43]。上海交通大学郁昱教授曾在 BlackHat 现场演示侧信道攻击破解 SIM 卡 AES-128 加密,用其方法能使用一张克隆 SIM 卡成功伪装成卡主,更改支付宝的密码,而且有可能将账户中的资金全部盗走。

对于 AES 设计上的改进,主要集中在提高硬件的吞吐量[44,45]和算法安全性[46]两方面。

5.5 有限域

在古典密码学中,常以模运算作为密码学加密运算技术。而随着计算机能力的提升,单纯的模运算并不安全。AES 引入了数论中的有限域,提高了加密算法抵抗攻击的能力。本节将简单讨论有限域的知识,主要侧重其在计算机上的实现,详细的理论不在本书的讨论范围。在讨论有限域前,需要先介绍一个有趣的问题——多项式算术,这里只讨论单自变量的多项式。

5.5.1 什么是有限域

有限域是一个有限集合 S 再添加两种特定运算的代数系统,与现实生活中的十进制运算不同的是,其上自定义的加法运算和乘法运算只要满足以下条件,即可以认为 S 是一个有限域。

① 加法的封闭性:如果 a 和 b 属于 S,则 $a+b$ 也属于 S。

② 加法结合律:对 S 中任意元素 a,b,c 有 $a+(b+c)=(a+b)+c$。

③ 加法单位元:S 中存在一个元素 a,使得对于 S 中任意元素 b,有 $a+b=b+a=b$,通常记 a 为 0。

④ 加法逆元:对于 S 中的任意元素 a,S 中一定存在一个元素 $-a$,使得 $a+(-a)=(-a)+a=0$。