第3章 数据加密标准

3.1 概述

古典密码算法不能很好地防范攻击，急切地需要一个新的加密算法来保证数据的安全。美国国家标准局(National Bureau of Standards，NBS)公开征集数据加密标准(Data Encryption Standard，DES)，经过两次甄选，最终确定标准。DES，即数据加密标准，从1976年被定为标准到2001年被高级加密标准所代替，走过了25年的历程，很好地履行了自身的职责。

DES是一种Feistel(读音为“Faistəl”)体制的分组密码，使用56比特有效密钥一次可以处理64比特数据。因为其良好的设计，可以防范多数攻击，但是随着时间的推移，DES日渐“衰老”，已经无法防范密钥的穷尽搜索。虽然DES已经被取代，但是相比较最初的10～15年设计需求，DES最终被使用了25年，至今仍然在一些安全需求不高的地方被人们所使用着。

3.1.1 DES的历史

在20世纪70年代前，密码学主要应用于军事方面，非军事密码学研究少有人参与。大多数人都知道军方采用特殊的编码设备来进行通信，但是很少有人懂得密码学。1972年，美国国家标准局(National Bureau of Standards，NBS)，即现在的美国国家标准和技术研究所(National Institute of Standards and Technology，NIST)，拟定了一个旨在保护计算机和通信数据的计划。作为该计划的一部分，他们想开发一个单独的标准密码算法。

- 1973年5月15日，NBS发布了公开征集标准密码算法的请求并确定了一系列的设计准则，但是由于提交的算法都与要求相去甚远，第一次征集失败。
- 1974年8月27日，NBS第二次征集加密算法标准，并征集到一个由IBM开发的有前途的候选算法Lucifer。其后请求美国国家安全局(NSA)帮助对算法的安全性进行评估以决定它是否适合作为联邦标准。
- 1975年3月17日，DES在“联邦公报”上发表并征集意见。公众对于NSA在开发该算法时的“看不见的手”很警惕，他们担心NSA修改了算法以安装陷门，同时还抱怨NSA将密钥长度由原来的112比特减少到56比特。有传言说，56比特长的密钥刚好能被拥有世界上最快计算资源的NSA破解，且对民间通信是安全的。

- 1976 年为针对公众的疑问，NBS 于 8 月和 9 月分别召开两次研讨会，一次研讨算法的数学问题及安装陷门的可能性，另一次研讨增加密钥长度的可能性。
- 1976 年 11 月，数据加密标准确立。

此后每隔 5 年官方都对 DES 进行评估，虽然标准确定伊始，认为 DES 服役时间为 10～15 年，然而最终 DES 服役时间远超 15 年，一方面是由于其自身的安全性，另一方面是由于没有更加可靠的密码算法。但是随着时间的推移，电子设备的性能呈指数趋势增加，暴力破解 DES 已不是难事，最终 DES 被使用更长密钥的 AES 算法所替代。

> 📖 **暴力破解与 DES 的密钥长度**
>
> 对于一切密码而言，最基本的攻击方法是暴力破解法——依次尝试所有可能的密钥。密钥长度决定了可能的密钥数量，因此也决定了这种方法的可行性。对于 DES，即使在它成为标准之前就有一些关于其密钥长度的适当性的问题，在设计时，在与包括 NSA 在内的外部顾问讨论后，密钥长度被从 128 比特减少到了 56 比特以便在单芯片上实现算法。而最终正是它的密钥长度不够，而迫使它被后续算法所替代。所以至今在一些对安全强度要求不高的芯片级应用场合，也常用到 DES。

3.1.2 DES 的描述

如图 3-1 所示，DES 是一种使用 Feistel 体制的分组密码，使用 56 比特原始密钥产生 16 组轮密钥，对 64 比特的明文分组进行 16 轮变换，最终得到密文分组。而解密时，使用加密的函数进行解密，但是需要将 16 组轮密钥逆向使用，使得密文变换为明文。

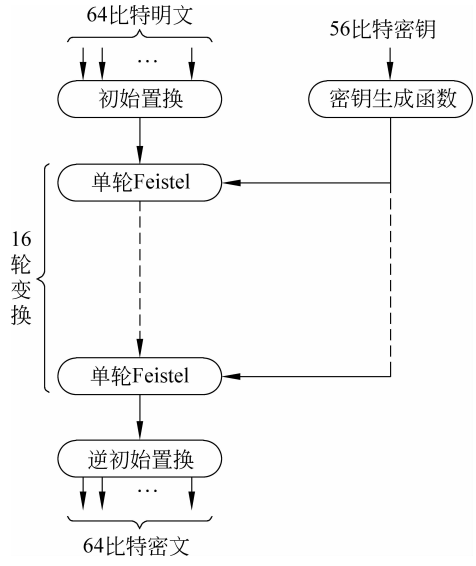

图 3-1　DES 描述图

DES 利用 Feistel 体制来实现加密的两个基本技术——混淆和扩散。混淆和扩散是

香农在其 1949 年的论文中提到的，使用该标准来判断加密效果的好坏。混淆用于掩盖明文和密文之间的关系，做到这点最容易的方法是通过代替，如凯撒密码。扩散通过将明文冗余度分散到密文中使之分散开来，做到这点最简单的方法是通过换位（也称为置换），如明文逆序。通常单独使用扩散容易被攻破，常常分组密码算法既用到混淆又用到扩散。

3.2 Feistel 体制

DES 算法使用 Feistel 体制来进行加解密，随着 DES 的公开，有大量的新开发的加密算法均使用 Feistel 体制来进行构建。Feistel 体制是一种多轮结构，每一轮操作相同，将密钥的信息注入当前轮输入的右半部分，再与左半部分异或后形成下一轮变换的输入。本节首先解释单轮的 Feistel 体制，然后再解释多轮的 Feistel 体制。

3.2.1 单轮 Feistel

单轮 Feistel 的输入和输出均为 $2n$ 比特的数据，其中分为左半部分 L 和右半部分 R，假设现在处于第 i 轮，则认为输入为 L_{i-1} 和 R_{i-1}，输入为 L_i 和 R_i，密钥为 K_i，单轮处理过程如图 3-2 所示。其中 K_i 输入到的部分成为 F 函数，不同的算法常常因为 F 函数的不同而不同。

输入 L_i 直接由 R_{i-1} 赋值得到，而 R_i 则是由 R_{i-1} 和密钥 K_i 输入到 F 函数中取得的结果再与 L_{i-1} 异或获得。L_i 和 R_i 可由式(3-1)和式(3-2)表示：

$$L_i = R_{i-1} \tag{3-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \tag{3-2}$$

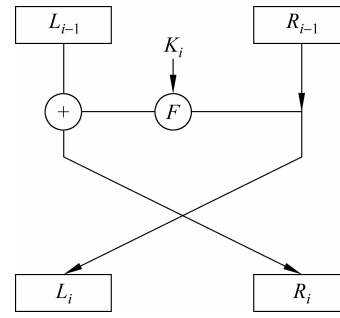

图 3-2　单轮 Feistel 体制

> **Feistel 与 Horst Feistel**
>
> 在密码学研究中，Feistel 密码结构是用于分组密码中的一种对称结构。以它的发明者 Horst Feistel 命名，而 Horst Feistel 本人是一位物理学家兼密码学家，在他为 IBM 工作的时候，为 Feistel 密码结构的研究奠定了基础。很多密码标准都采用了 Feistel 体制，其中包括 DES。Feistel 的优点在于：由于它是对称的密码结构，所以对信息的加密和解密的过程就极为相似，甚至完全一样。这就使得在实施的过程中，对编码量和线路传输的要求减少了几乎一半。

3.2.2 多轮 Feistel

多轮 Feistel 的输入为 $2w$ 长度的明文分组，而输出也为 $2w$ 长度的密文分组，同时还需要 n 个 w 长度的密钥组成的密钥序列，每一轮使用一个密钥。多轮 Feistel 相当于进行多次的 Feistel 单轮操作，输出最终结果时，将最后一轮结果的左右分组交换后输出。而

Feistel 的加密和解密结构，仅仅是密钥顺序的不同，其余结构一样，从而使得硬件上的线路可以减少一半或者软件上的编码量减少一半。加解密的 Feistel 体制示意图如图 3-3 所示。

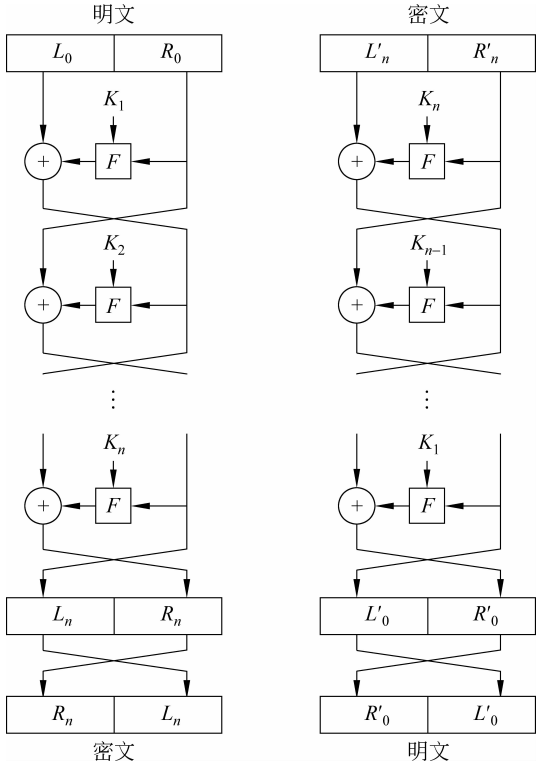

图 3-3　多轮 Feistel 体制的加解密

3.2.3　Feistel 加解密的同结构

如图 3-3 所示，Feistel 体制在加解密上可以使用同一个结构来实现，在实现过程中，只需将密钥逆序使用即可。下面按照图 3-3 来证明上面所说的正确性。

式(3-1)和式(3-2)说明了加密的数据变化，由于距离上次提出较远，故在此处重新声明：

$$L_i = R_{i-1} \tag{3-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \tag{3-2}$$

同样，对于解密有：

$$L_{i-1}' = R_i' \tag{3-3}$$

$$R_{i-1}' = L_i' \oplus F(R_i', K_i) \tag{3-4}$$

由图 3-3 可知，$L_n' = R_n$，$R_n' = L_n$，继而由式(3-1)、式(3-2)、式(3-3)和式(3-4)可得解密结构中第一轮结束后的关系为：

$$L_{n-1}' = R_n' = L_n = R_{n-1} \tag{3-5}$$

$$
\begin{aligned}
R_{n-1}' &= L_n' \oplus F(R_n', K_n) \\
&= R_n \oplus F(L_n, K_n) \\
&= (L_{n-1} \oplus F(R_{n-1}, K_n)) \oplus F(R_{n-1}, K_n) \\
&= L_{n-1}
\end{aligned} \tag{3-6}
$$

多次迭代后可得：

$$L_0' = R_0 \tag{3-7}$$

$$R_0' = L_0 \tag{3-8}$$

可见，解密成功。

3.3 DES 加密

DES 算法使用 Feistel 体制作为框架进行设计，通过实现 F 函数和密钥扩展函数，并对 Feistel 进行初始置换和逆初始置换形成 DES 算法。具体的 DES 结构如图 3-4 所示，64 比特明文首先通过初始置换后，进行 16 轮的 Feistel 体制，再经过一个逆初始置换形成密文，其中 16 轮 Feistel 体制中使用的密钥由密钥生成函数通过一个 56 比特密钥生成。

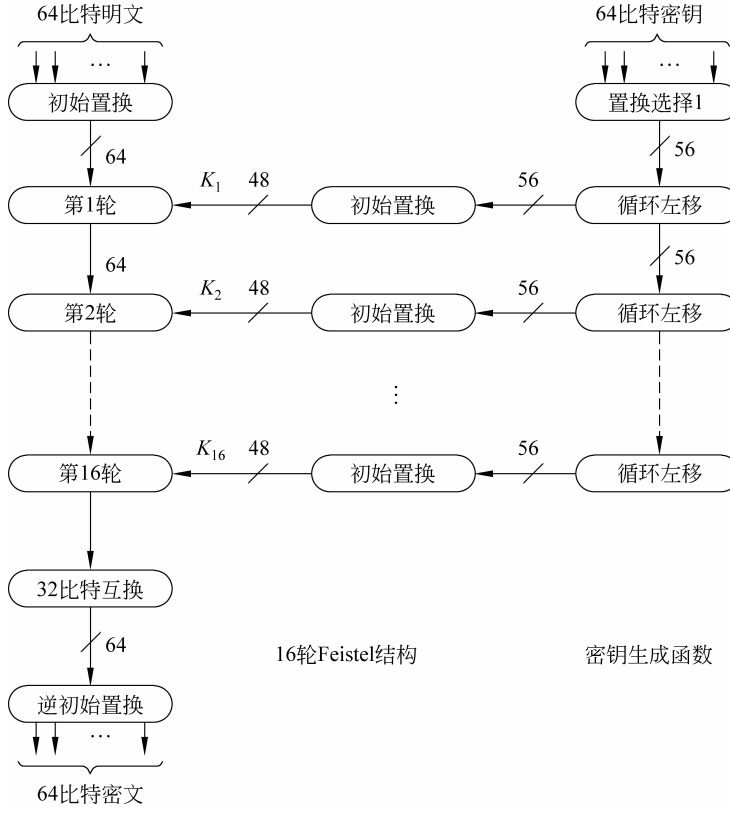

图 3-4　DES 总体结构

下面分别对初始置换、逆初始置换、密钥生成函数和 F 函数进行讲解。

3.3.1 DES 的初始置换与逆初始置换

DES 算法在执行 Feistel 体制的 16 轮变换前，需要经过一个初始置换(IP 置换)，使用的置换表为 IP 置换表，其数据如表 3-1 所示。表的输入标记为 1～64，共 64 比特。置换表中 64 个元素代表 1～64 这些数的一个置换。置换表中的每个元素表明了某个输入比特在 64 比特输出比特的位置。例如 IP 表的第 1 个元素表明，输入比特的第 1 位在输出比特的第 58 位。

表 3-1 初始置换(IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

在结束了 Feistel 体制变换后，同样，DES 还需要进行一次逆初始置换，使用的置换表为逆初始置换表(IP^{-1})，表的数据如表 3-2 所示，使用方法同初始置换表。

表 3-2 逆初始置换(IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

经过逆初始置换后，整个 DES 加密(解密)过程就结束了。而对于初始置换和逆初始置换，实际上是一次互逆的过程，即 $M=IP^{-1}(IP(M))=IP(IP^{-1}(M))$。正是由于互逆的性质，保证了加密的逆初始置换和解密的初始置换相互抵消，从而在解密的时候复原了加密时最后一轮的 Feistel 体制的输出。同理，加密的初始置换和解密的逆初始置换也相互抵消，形成明文。

3.3.2 DES 的 F 函数

DES 使用 Feistel 体制作为主结构，实现了自己 Feistel 体制中的 F 函数。DES 的 F

函数主要是由扩展置换(E 置换)、异或操作、代换选择(S 盒代换)和 P 盒置换组成,分组 R(32 比特)经过 E 置换扩展至 48 比特,接着与当前轮次密钥 K(48 比特)进行异或操作,结果经过 S 盒代替后再经过一次 P 置换取得 F 函数的输出,整个 F 函数的流程如图 3-5 所示。

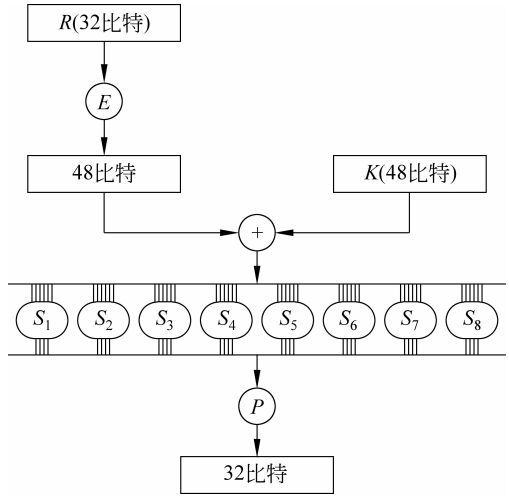

图 3-5　F 函数流程图

首先分组 R 经过的变换是 E 置换,该置换可以将一个 32 比特的数据,扩展成 48 比特的分组。E 置换用到的数据如表 3-3 所示,其使用的方法如同 IP 表一样。

表 3-3　扩展置换(E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

分组 R 经过扩展置换后形成 48 比特的分组,该分组首先和当前轮次的密钥 K 进行异或操作,然后需要进行 S 盒的代换。DES 所用到的 S 盒的定义如表 3-4 所示,分为 8 个 S 盒。如图 3-5 所示,每个 S 盒输入为 6 比特,输出为 4 比特,故总输入为 48 比特而输出为 32 比特。而 S 盒的变换操作如下:盒 S_i 的输入为输入分组的第 $6i-5$ 到 $6i$ 比特,盒 S_i 输入的第一位和最后一位组成一个 2 比特的二进制数,用来选择 S 盒 4 行代替值中的一行,中间 4 比特用来选择 16 列中的某一列。例如,在 S_1 中,若输入为 010110,则行是 0(00),列是 11(1011),该处值是 12,所以输出是 1100。

表 3-4　DES 的 *S* 盒定义

盒 S_1：															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
盒 S_2：															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
盒 S_3：															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
盒 S_4：															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
盒 S_5：															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
盒 S_6：															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
盒 S_7：															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
盒 S_8：															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

取得 S 盒变换的结果后，还需要进行一次 P 置换才可以得到 F 函数的输出。P 置换如表 3-5 所示，该表的使用方法如 IP 置换表一样。

表 3-5　P 置换

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

F 函数与雪崩效应

明文或密钥的微小改变将对密文产生很大的影响，是任何加密算法都需要的一个好性质。特别地，明文或密钥的某一位发生变化会导致密文的很多位发生变化，这称为雪崩效应。如果相应的改变很小，可能会给分析者提供缩小搜索密钥或明文空间的渠道。实际上，雪崩效应也是香农提出的扩散和混淆的一个具体现象。DES 具有较好的雪崩效应，而这主要得益于 F 函数的设计。而 F 函数的主要部件是 S 盒，有资料指出，S 盒被 NSA 修改的一个原因在于可以提供更好的雪崩效应，且可以防范差分密码分析。

3.3.3　DES 的密钥扩展算法

在 DES 中，用户只需知道 64 比特的密钥，而在 16 轮的 Feistel 体制中使用的 16 个子密钥，是通过一个密钥扩展算法将 64 比特密钥扩展生成 16 个子密钥。该算法的输入为 64 比特密钥，首先需要经过一次置换选择 1 的变换，生成 2 个 28 比特的密钥分组。置换选择 1 中所使用的数据表如表 3-6 所示，使用方法与 IP 表使用方法一致。

表 3-6　PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

经过 PC-1 变换后，密钥被分为 2 个 28 比特的分组 C_1 和 D_1。这里解释密钥生成函数每轮的步骤。假设当前处于第 i 轮，则输入为 C_{i-1} 和 D_{i-1}，首先移位获得置换选择 2 的输入和下一轮的输入 C_i 和 D_i，接着拼接 C_i 和 D_i，进行置换选择 2，形成子密钥 K_i。算法流程如图 3-6 所示，其中密钥生成函数的迭代轮次与移位比特数的关系如表 3-7 所示，置换选择 2 使用的数据表如表 3-8 所示，使用方法同 IP 表。

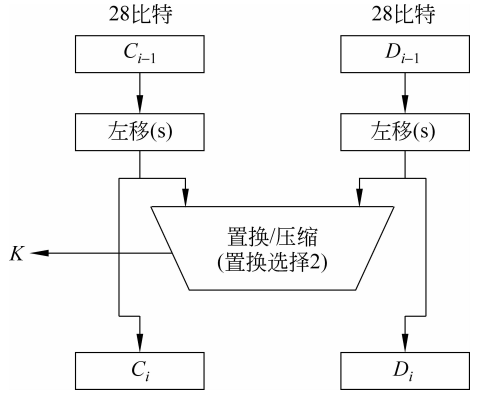

图 3-6　密钥扩展算法单轮结构

表 3-7　移位次数关系表

迭代轮次	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
移位次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

表 3-8　置换选择 2(PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	27	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

3.4　DES 解密

DES 密码属于 Feistel 密码，故其解密算法与加密算法是相同的，只是子密钥的使用次序相反，在这里就不再做讨论。

3.5　分组密码的使用

Alice 在学会了 DES 加密后，选择和朋友通信都进行加密处理。由于明文数据的长度是不定的，经常出现明文数据比 DES 一次处理的块大小长的情况，所以她每次都是使用同一个密钥——Alice123——依次对切割后的明文分组进行加密。但是某次加密过程中她发现了一个问题：对图 3-7(a)所示的图片进行加密，在加密过程中发现结果如图 3-7(b)所示，可以看出，虽然数据被加密了，但是作为图片，仍然可以被人看出其数据。Alice 在意识到这个问题后，立刻终止加密，但是她又希望数据可以加密后发送，那么此时她该怎么办呢？

(a) 未加密效果　　(b) 加密效果

图 3-7　ECB 的数字图像加密效果图

3.5.1　分组密码工作模式

分组密码在对明文加密时，可以根据加密过程中前一明文分组对下一明文分组加密过程的影响，分为多种工作模式。Alice 正是使用了最简单也是最不安全的工作模式——ECB 模式，如果 Alice 使用其余几种模式均可以避免所碰到的问题。

1. 电码本模式

电码本模式(Electronic CodeBook Model，ECB)是简单的一种工作模式，它对明文分块后，对每一块使用密钥分别加密获得密文块，再将密文块连接后获得密文，如图 3-8 所示。

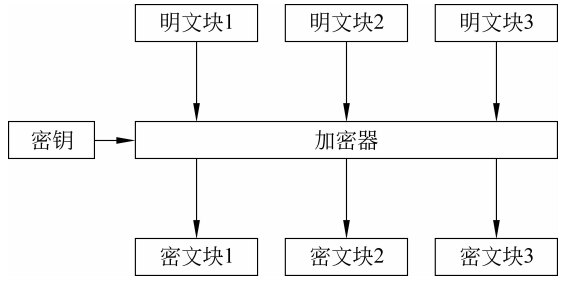

图 3-8　ECB 模式示意图

在这种模式下，前后明文互不影响，故在密钥一定的情况下，内容相同的明文块会被加密成一样的密文块，故当攻击者获得足够多的密文块和明文块的对应关系时，不需要解密即可阅读，类似于中文和英文的翻译。显而易见，这种方法简单明了，可以并行操作，一个密文块传输错误不会影响后续密文解密，但是缺点是无法避免重放攻击。

2. 密码分组链接模式

密码分组链接(Cipher-Block Chaining model，CBC)是由 IBM 在 1976 年发明的。在 CBC 模式中，每个明文块先与前一个密文块相异或，再进行加密。而对于第一个密文块，由于没有前一个密文块，故和一个初始向量(Initialization Vector，IV)相异或，再进行加密，初始向量通常由双方协商指定，如图 3-9 所示。

在这种模式下，前一密文块对后一明文块产生影响，从而使得同样的明文块会产生不同的密文，提高了安全性。但是缺点也显而易见，正是由于前一密文块对后一明文块产生影响，使其不利于并行计算，如果传输过程中存在 1 比特错误，将导致后续所有密文无法解读。

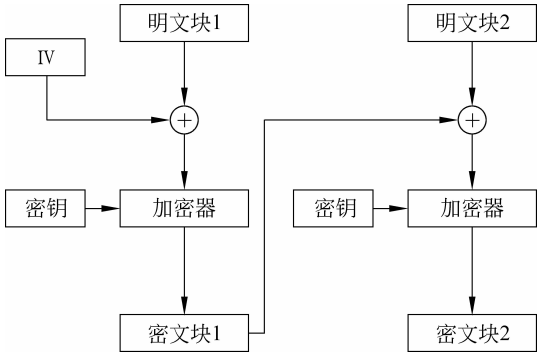

图 3-9　CBC 模式示意图

3. 密文反馈模式

在密文反馈(Cipher FeedBack model,CFB)模式下,明文不进入加密算法中处理,而是与加密算法的输出异或得到密文,加密算法的输入为上一次的密文以及加密用的密钥,对于第一次加密,同样引入了一个初始向量 IV,具体的流程如图 3-10 所示。

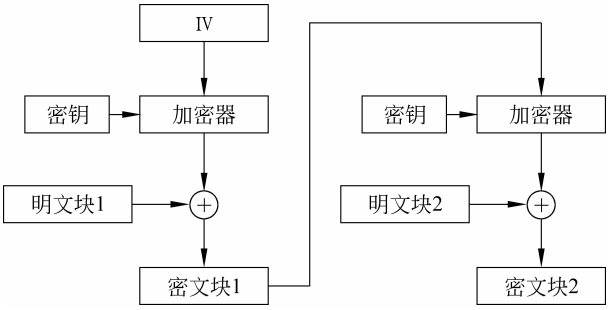

图 3-10　CFB 模式示意图

CFB 模式和 CBC 模式类似,其优缺点也类似,但是该种方法还可以使 IV 大小和明文块大小不同,从而将分组密码转变为流密码。

> **分组密码和流密码**
>
> 分组密码和流密码均属于对称密码。分组密码将明文分成多个等长的块,使用确定的算法和对称密钥对每组分别加密解密。在实践中,多以 64 位、128 位和 256 位作为分组长度。而流密码是一种一次只对一位进行操作的密码,通常加解密双方持有共同的一个随机种子,然后生成随机数,使用随机数中的某一位对数据位进行加密。正是由于流密码一次处理一位,使得其在网络协议中可以做到加密一位发送一位,使得其效率较高。

4. 输出反馈模式

如图 3-11 所示,输出反馈模式(Output FeedBack model,OFB)与 CFB 模式相似,与

CFB 模式只有一点不同：CFB 使用上一密文块作为加密算法输入，而 OFB 是使用上一加密算法的输出作为下一轮加密算法的输入。

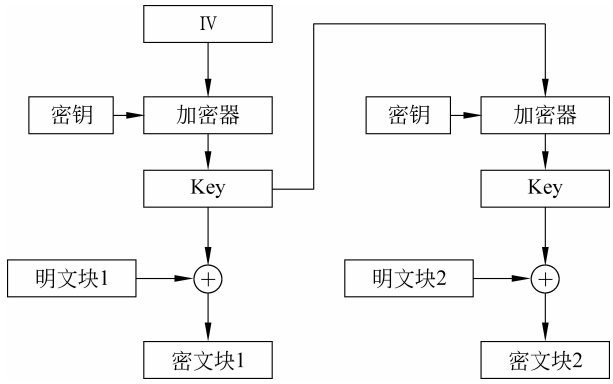

图 3-11　OFB 模式示意图

OFB 模式虽然与 CFB 模式极度相似，但是由于与明文块异或操作的 Key 未掺入明文的信息，而是使用同一密钥对 IV 进行加密获得，使得 OFB 模式不会出现一个分组错误影响后续所有分组，同时它也克服了图 3-8 无法抵抗重放攻击的缺点。

5. 计数器模式

计数器模式(Counter mode，CTR)的提出是为了可以随机地解密密文分组，因为有时可能不需要解密整个明文，只需对明文中的某一部分进行解密。该技术常用于数据库技术中，如同现实中的刮刮乐，只需刮出第一个字为“谢”即知道自己没有中奖，而无须继续刮出其余的字。CTR 模式的示意图如图 3-12 所示，每一个分组具有自己的计数，通过计数可以获得用于输入到加密算法中的“明文”，经加密后，输出与明文块异或获得密文块。

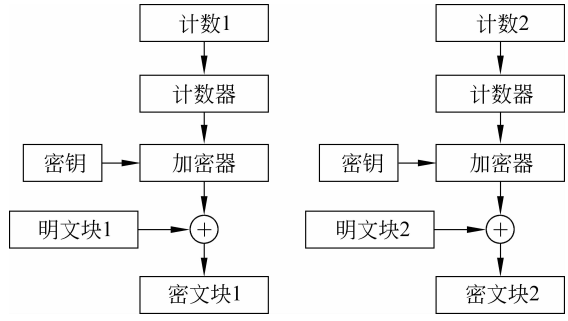

图 3-12　CTR 模式示意图

> **计数器的选择**
>
> 计数器模式既保证了数据不存在密码本，又保证了可以随机存储，故在数据库等需要随机存储的场合下大量运用。如果计数器选择不当，将会带来很大的隐患。最简

单的计数器即为直接将计数作为计数器的输出，这种方法只能保证密文不存在电子密码本问题。所以为了保证数据的安全性，需要选择更为复杂的计数器，这样的计数器应当是一个输入到输出的非线性映射函数，且通常输入为一个无穷大的集合，用于保证可以使用足够长的时间。

3.5.2 分组密码填充模式

Alice 在查阅资料获知以上工作模式后，意识到是自身的问题带来了数据的不安全性，所以在数据加密中，使用了 CBC 模式。但是很快她就发现了另外一个问题：数据的最后一个分组比特数不够，只含有 32 比特数据，其十六进制表示为 DDDDDDDD，而这不足以作为加密算法的输入。她想要在后面加上 0 作为填充进行加密，但是她又担心 Bob 不知道删去多余的 0，那么她该如何让 Bob 知道应该删去作为填充数据的多余数据呢？

其实上述问题主要是由于分组密码的特性导致——一次以一个特定长度块作为分组进行加密。针对这样的情况，有两种方式可以解决：一种是使用工作模式，如 CFB、OFB 或 CRT，进行流密码的转换；而另一种方法是，加密方和解密方约定一种填充模式，加密方按照规则进行填充，解密方即可知道在什么情况下需要删除多少数据。

1. 零字节填充

零字节填充(Zero Padding)是在需要填充字节的位置填充为 0。对于这种填充模式，Alice 会将最后的四个字节填充为 0，即 DDDDDDDD00000000。而 Bob 在收到密文解密得到该字符串后，会将末尾的所有 0 字节全部删去，来理解整段话。

2. PKCS7

对于使用 PKCS7 填充模式，Alice 会在最后的四个字节上填充上 04040404。如果是最后填充 1 个字节，则为 01，填充 2 个字节则为 0202，以此类推。Bob 在收到密文并解密后，查看最后一个字符并检查是否符合填充规则，对于是经过填充得到的字符串删去后理解明文。

3. ANSI X.923

对于使用 ANSI X.923 填充模式，Alice 会在最后的四个字节填充上 00000004。如果是填充最后一个字节，则为 01，填充 2 个字节则为 0002，以此类推。Bob 在收到密文并解密后，查看最后一个字符并检查是否符合填充规则，对于是经过填充得到的字符串删去后理解明文。

4. ISO 10126

对于使用 ISO 10126 填充模式，Alice 会在最后一个字节上填充 04，而另外三个字节填充随机数。即对于这种填充模式，最后一个填充字节表明填充比特数，其余为随机数。Bob 在收到密文并解密后，查看最后一个字符并检查是否符合填充规则，对于经过填充得到的字符串删去后理解明文。

经过学习了工作模式和填充模式，只要和 Bob 商定好了工作模式和填充模式，Alice

就可以随心所欲地传递自己想要的数据，而不用担心数据无法准确无误地传递给 Bob。

3.6 破解 DES

DES 在 20 世纪最后 20 年一直作为密码学标准的加密体制，在历史上它多少带有一些神秘色彩：更多的问题集中在人们认为 NSA 修改算法的时候设置了陷门。人们花费了大量的时间和精力寻找 NSA 可能安放在算法中的陷门和漏洞，虽然 DES 的真相目前不为人所知(除非政府机密文件公之于世)，但它已展示出了较强的抵抗攻击的能力。随着时间推移，DES 已经开始暴露它的“老态”，这主要源于计算机硬件的发展。

到目前为止，针对 DES 最实用的攻击方法仍然是暴力破解法——通过测试所有密钥进行破解。虽然针对 DES 一些可能导致加密强度降低的密码学特征具有一些攻击，但是需要已知明文或选择明文数量，这是不现实的，因此并无实用价值。

- 1975 年以前，许多学术机构抱怨 DES 密钥的长度不够。事实上，NBS 公开 DES 几个月后，名为“NBS 数据加密标准详尽密码分析”的论文指出，使用两千万美元(1977 年的标准)制造的设备可以使用大约一天的时间破解 DES。
- 1993 年，工作在 Bell-Northern 研究所的研究员 Michael Wiener 提出并设计了一种设备，它比以往任何一种设备都能更高效地攻破 DES。
- 1996 年，出现了攻击对称密码，如 DES 的三种方法的详尽说明。第一种方法是在一个有大量情报的机器上做分布式计算，它的优点是相对便宜，费用也易于分配。第二种方法是设计自定义的体系结构，如 Michael Winener 的思想，来破译 DES，一般来说，它的效率较高，但费用也很昂贵，可以认为是最尖端的技术。最后一种方法较为折中，它采用可编程逻辑阵列，在后来的时间里，这种方法得到了一定的关注。
- 1997 年 RSA 数据安全公司组织了一次具有挑战性的活动，寻找密钥并破解 DES 加密信息，谁破解了这个信息，谁就可以获得一万美元的奖金。此时，用分布式计算方法来破译 DES 越来越受人喜爱，特别是随着 Internet 的普及。在挑战赛开始后 5 个月，RockeVerser 就利用分布式计算提交了正确的 DES 密钥。这一成果标志着分布式计算已经可以成功地攻破 DES。RockeVerser 是将已经自行编制的程序发布在了 Internet 成千上万的计算机上，共同运行来破译 DES 密码，而这些机器是由网络上的人自愿贡献的。
- 接下来的一年里，RSA 数据安全委员会组织了第二届 DES 挑战赛，这次的密钥仍然是由分布式计算技术发现的，且用时更短，只经历了 39 天。第二届比赛的获胜者搜索了比第一届更多的密钥空间，且执行速度更快，这表明，一年来技术上的进步对密码分析产生了戏剧性的影响。
- 1998 年，电子前沿基金会制造了一台 DES 解密高手机器，花费 25 万美元，使用 56 小时即可攻破 DES，它显示了迅速破解 DES 的可能性。
- 1999 年 1 月，distributed.net 与电子前哨基金会合作，在 22 小时 15 分钟内即公

开破解了一个 DES 密钥。

由此可见，DES 已经不再安全，但是，为何还有很多的厂商在使用 DES 呢？这其中的原因主要是厂商更换设备需要成本，而更换设备后所带来的收益不足以支付更换设备成本，或者是厂商所保密的数据并不值得使用更加有效的加密手段，这就涉及一个安全代价的话题——根据所保密数据的价值，采取合适的保密措施。例如个人计算机上的数据可能并不是很重要，仅仅是不为访问者看到明文数据，那又何必使用更高成本的加密手段加密呢？

在 DES 初显衰败的时候，为何 NSA 没有更换新设计的算法，而是保持了 DES 的位置？在 1987 年 DES 需要承受第二个五年审查。针对 DES 的讨论看出，NSA 反对给 DES 换发新证，在当时，NBS 提出 DES 开始显现出弱点，并建议全部废除 DES，采用一套 NSA 新设计的算法代替，算法内部的工作过程只有 NSA 知道，这样可以很好地从工程技术的方面保护它。这个建议最终未被采纳，部分原因在于几个主要的美国公司在置换算法期间利益得不到保护。最后，重新提出把 DES 作为标准，但在讨论的过程中，人们已经认识到 DES 存在弱点。

IDEA 对称加密算法

IDEA（国际数据加密标准，International Data Encryption Algorithm）是由上海交通大学来学嘉教授（见右图）与瑞士学者 James Massey 提出的。IDEA 使用长度为 128b 的密钥，数据块大小为 64b。它基于“相异代数群上的混合运算”设计思想，算法用硬件和软件实现都很容易，且比 DES 在实现上快得多。自问世以来，IDEA 已经经历了大量的详细审查，对密码分析具有很强的抵抗能力。目前 IDEA 在工程中已有大量应用实例，PGP（Pretty Good Privacy）就使用 IDEA 作为其分组加密算法；安全套接字层（Secure Socket Layer，SSL）也将 IDEA 包含在其加密算法库 SSLRef 中；IDEA 算法专利的所有者 Ascom 公司也推出了一系列基于 IDEA 算法的安全产品，包括基于 IDEA 的 Exchange 安全插件、IDEA 加密芯片、IDEA 加密软件包等。

3.7 三重 DES

随着时间的推移，DES 在穷举攻击之下相对比较脆弱，因此很多人在想办法用某种算法代替它。一种方案是设计一个全新的算法，例如后来出现的 AES。另外一种方案是通过使用同一个算法、多个密钥进行多次加密，例如三重 DES，或称为 3DES。后一种方案通常是在第一种方案无法实施时的替代方案，并且它能够保护已有软件和硬件的投资。正是由于还没有全新的算法足以替代 DES，NIST 才选择使用第二种方案，推荐人们使用一个过渡性的加密算法——三重 DES（3DES）。

NIST 建议的 3DES 的加密流程如图 3-13 所示，明文在经过密钥 1 的两次加密中间加入一次密钥 2 的解密。如此做的一个主要目的是：当密钥 2 与密钥 1 相同的时候，相当于 3DES 只进行了一次密钥 1 的加密，从而使得 3DES 也可以当 DES 使用。

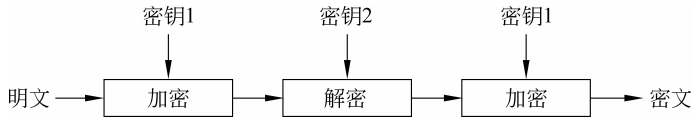

图 3-13　3DES 流程图

当然，也存在另外的 3DES，比如使用 3 个不同密钥的 3 次 DES 加密，只需是三重加密即可（解密也可以算是一种加密）。

思　考　题

1. 哪些参数与设计选择决定了实际的 Feistel 密码算法？
2. 混淆和扩散的差别是什么？
3. 雪崩效应的实际意义是什么？
4. 分组密码和流密码的差别是什么？
5. 为什么推荐的 3DES 的中间部分采用解密而不是加密？
6. Feistel 体制中，如果最后一轮之后附加的分组交换，全部放在第一轮之前是否可以？
7. 各个工作模式的特点、优点和缺点分别是什么？

参 考 文 献

[1] Smid M E, Branstad D K. Data encryption standard: past and future[J]. Proceedings of the IEEE, 1988, 76(5): 550-559.

[2] Schneier B. Applied cryptography: protocols, algorithms, and source code in C[M]. New York: John wiley& sons, 2007.

[3] Stinson D R. Cryptography: theory and practice[M]. FL: CRC press, 2005.

[4] William S, Stallings W. Cryptography and Network Security, 4/E[M]. New Delhi: Pearson Education India, 2006.

[5] Biham E, Shamir A. Differential cryptanalysis of the data encryption standard[M]. New York: Springer-Verlag, 1993.

[6] Matsui M. The first experimental cryptanalysis of the Data Encryption Standard[C]//Advances in Cryptology—Crypto'94. Springer Berlin Heidelberg, 1994: 1-11.

[7] Standard D E. FIPS pub 46[J]. Appendix A, Federal Information Processing Standards Publication, 1977.

[8] 冯登国. 国内外密码学研究现状及发展趋势[J]. 通信学报，2002，23(5)：18-26.

[9] 刘晓星，胡畅霞，刘明生，等. 安全加密算法 DES 的分析与改进[J]. 微计算机信息，2006，(12)：

32-33.
[10] 吴文玲,冯登国. 分组密码工作模式的研究现状[J]. 计算机学报,2006,29(1): 22-25.
[11] 沈昌祥,张焕国,冯登国,等. 信息安全综述[J]. 中国科学 E 辑: 信息科学,2007,37(2): 129-150
[12] 吴文玲,冯登国,张文涛. 分组密码的设计与分析[M]. 第 2 版. 北京: 清华大学出版社,2009.
[13] Feistel cipher[OL]. 维基百科. http://en. wikipedia. org/wiki/Feistel_cipher. 2014
[14] Data Encryption Standard[OL]. 维基百科. http://en. wikipedia. org/wiki/Data_Encryption_Standard. 2014
[15] Block cipher mode of operation[OL]. 维基百科. http://en. wikipedia. org/wiki/Block_cipher_mode_of_operation. 2014
[16] Padding (cryptography)[OL]. 维基百科.
[17] http://en. wikipedia. org/wiki/Padding_(cryptography). 2014

第4章 高级加密标准

高级加密标准(Advanced Encryption Standard,AES)是目前被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,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时间,高级加密标准已然成为对称密钥加密中最流行的算法之一。其加密和解密过程使用两套算法,是一个互逆操作,在变换过程中,交替使用代替和置换两种操作来达到加密效果。与以往加密算法不同的是,将数学运算代入加密算法,进一步提高了算法的安全性,就目前情况来看,AES具有较高的安全性。

4.1 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标准中指出,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能成为未来数十年加密算法标准的合适选择。

但是3DES的根本缺点在于用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,难以用软件有效地实现该算法。在3DES中轮的数量三倍于DES中轮的数量,故其速度慢得多。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。

由于这些缺点,3DES不能成为长期使用的加密算法标准。故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,同时应具有更好的执行性能。除了这些通常的要求之外,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。

- 1998年8月12日,在首届AES候选方案会议上公布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(发音为"Rain dale")、DFC、Twofish、HPC。
- 1999年3月,在第2届AES候选方案会议(second AES candidate conference)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,从15个候选算法中选出了5个,分别是RC6、Rijndael、SERPENT、Twofish和MARS。

- 2000 年 4 月 13 日至 14 日，召开了第 3 届 AES 候选方案会议（third AES candidate conference），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，但是在成为标准的同时，也进行了一些修改。

> **Rijndael 并非 AES**
>
> 严格地说，AES 和 Rijndael 加密法并不完全一样（虽然在实际应用中二者可以互换），因为 Rijndael 加密法可以支持更大范围的分组和密钥长度：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。

4.2 代替置换网络结构

代替置换网络（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替操作和置换操作产生密文，这些操作均可以使用硬件上的异或、移位等操作进行，使其在软件和硬件上均可很好地使用。每一轮使用的子密钥产生于输入的密钥，且在有些基于 SPN 算法的加密算法中，用于进行代替操作的 S 盒也是基于子密钥产生的。图 4-1 为一个使用 2 轮 SPN 结构的算法，其中单轮经过密钥异或、S 盒代替和 P 盒置换三个步骤。

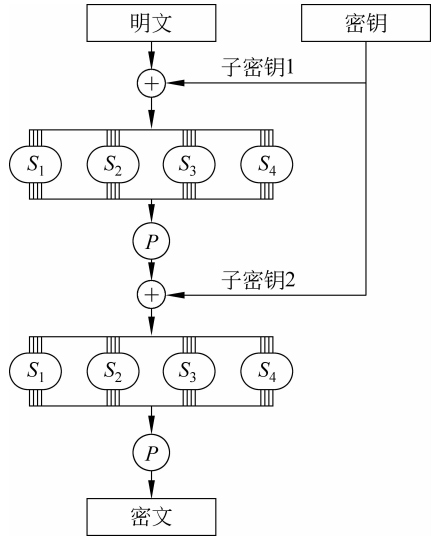

图 4-1　简单 SPN 网络