西安电子科技大学
XIDIAN UNIVERSITY

现 代 密 码 学

MODERN CRYPTOGRAPHY

陈晓峰

网络与信息安全学院

Email: xfchen@xidian.edu.cn

Homepage: http://web.xidian.edu.cn/xfchen

目录

第二讲
私钥加密技术

□ 私钥密码体制

- 私钥密码体制又称为单钥密码体制或对称密码体制，主要特征是加密密钥和解密密钥相同。
- 私钥密码体制的优势：加、解密速度更快，密钥相对较短，可以低成本的软硬件实现。
- **私钥密码体制的主要问题**：安全的密钥管理 (Key Management)，即如何安全高效地产生、分发、保管密钥， 尤其是在大规模的网络环境中。
- **私钥密码体制**：
  - 明文消息按字符（如二元数字）逐位地加密，称为流密码；
  - 将明文消息分组（含多个字符），逐组地进行加密，称为分组密码。

□ 流密码概述

- Shannon 已经证明“一次一密”在理论上是不可破译的。
  - 安全、高效。
  - 随机密钥的长度至少要等于明文长度。
- **流密码**（也称**序列密码**） 是一次一密系统的弱化版本。
  - 理论比较成熟，而且实现简单、速度快、错误传播少。
  - 军事和外交等领域的主要密码体制之一。
  - 流密码中使用一个伪随机序列来代替随机的密钥流。因此，不可能真正的实现完善保密。

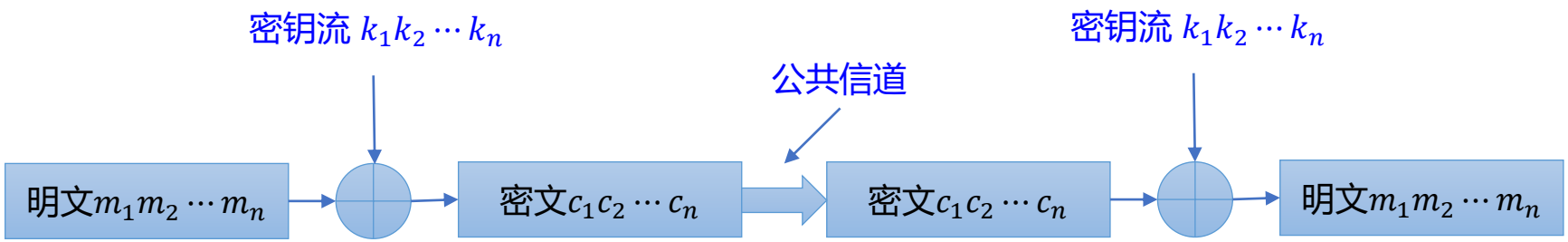
流密码概述

- 设明文为 $m = m_1 m_2 \cdots m_n$，由种子密钥通过伪随机数发生器得到的密钥流为：$k = k_1 k_2 \cdots k_n$，则**加密变换为**：

$$c = c_1 c_2 \ldots c_n$$

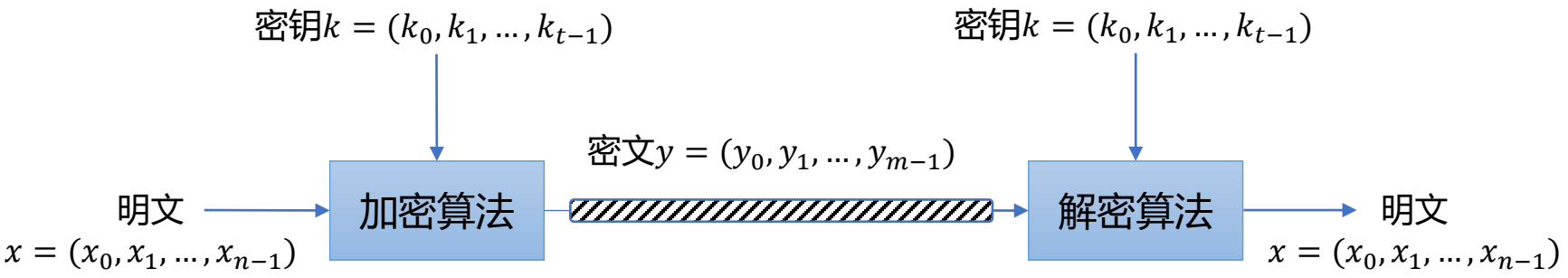
其中 $c_i = m_i \oplus k_i \ (i = 1,2,\ldots,n)$，$\oplus$表示模 2 加法（异或）。

- **解密变换为**：$m_i = c_i \oplus k_i (i = 1,2,\ldots,n)$。



口 分组密码概述

- **分组密码**是将明文消息编码表示后的数字序列 $x_1, x_2, \ldots, x_i, \ldots$ 划分成长为 n 的组 $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$，各组分别在密钥 $k = (k_0, k_1, \ldots, k_{t-1})$ 控制下变换成等长的输出数字序列 $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1})$ （长为 m 的矢量）。一般情况下， $m = n$。
- 分组密码与流密码的不同之处在于输出的每一位数字不是只与相应时刻输入的明文数字有关，而是与一组长度为 n 的明文数字有关。



□ 分组密码概述

- 分组密码的设计问题在于找到一种算法，能在密钥控制下从一个足够大且足够好的置换子集中，简单而迅速地选出一个置换，用来对当前输入的明文的数字组进行加密变换。设计的算法应满足下述要求：
  - 分组长度要足够大，防止明文穷举攻击奏效。
  - 密钥量要足够大，尽可能消除弱密钥并使所有密钥同等地好，以防止密钥穷举攻击奏效。
  - 由密钥确定置换的算法要足够复杂，充分实现明文与密钥的扩散和混淆，没有简单的关系可循，要能抗击各种已知的攻击。
  - 加密和解密运算简单，易于软件和硬件高速实现。
  - 数据扩展尽可能小。一般无数据扩展，在采用同态置换和随机化加密技术时可引入数据扩展。
  - 差错传播尽可能地小。

□ 美国制定数据加密标准简况

通信与计算机相结合是人类步入信息社会的一个阶梯，它始于六十年代末，完成于90年代初。计算机通信网的形成与发展，要求信息作业标准化，安全保密亦不例外。只有标准化，才能真正实现网的安全，才能推广使用加密手段，以便于训练、生产和降低成本。美国 NBS （National Bureau of Standards）在 1973 年 5 月15 日公布了征求建议。1974 年 8 月 27 日 NBS 再次出公告征求建议，对建议方案提出如下要求：

(1) 算法必须完全确定而无含糊之处；

(2) 算法必须有足够高的保护水准，即可以检测到威胁，恢复密钥所必须的运算时间或运算次数足够大；

(3) 保护方法必须只依赖于密钥的保密；

(4) 对任何用户或产品供应者必须是不加区分的。

2 DES算法

- IBM 公司从 60 年代末即看到通信网对于这种加密标准算法的需求，投入了相当的研究力量开发，成立了以 Tuchman 博士为领导的小组，包括A. Konkeim , E. Grossman, N. Coppersmith 和 L. Smith 等 (后二人做实现工作) 的研究新密码体制的小组，H. Fistel 进行设计，并在1971年完成的 LUCIFER 密码 (64 bit 分组，代换-置换，128 bit 密钥) 的基础上，改进成为建议的 DES 体制。NSA (National Standards Association) 组织有关专家对 IBM 的算法进行了鉴定，而成为 DES 的基础。
- 1975 年 3 月 17 日 NBS 公布了这个算法，并说明要以它作为联邦信息处理标准，征求各方意见。1977 年 1 月 15 日建议被批准为联邦标准 [FIPS PUB 46]，并设计推出 DES 芯片。DES 开始在银行、金融界广泛应用。1981 年美国 ANSI 将其作为标准，称之为 DEA [ANSI X3.92]。1983 年国际标准化组织 (ISO) 采用它作为标准，称作 DEA-1。

2 DES算法

- 1984 年 9 月美国总统签署145号国家安全决策令 (NSDD)，命令 NSA 着手发展新的加密标准，用于政府系统非机密数据和私人企事业单位。NSA宣布每隔5 年重新审议 DES 是否继续作为联邦标准，1988年 (FIPS46-1)、1993年 (FIPS46-2)，1998年不再重新批准 DES 为联邦标准。虽然 DES 不会长期地作为数据加密标准算法，但它仍是**迄今为止得到最广泛应用的一种算法，也是一种最有代表性的分组加密体制**。因此，详细地研究这一算法的基本原理、设计思想、安全性分析以及实际应用中的有关问题，对于掌握分组密码理论和当前的实际应用都很有意义。
- 1993 年 4 月，Clinton 政府公布了一项建议的加密技术标准，称作密钥托管加密技术标准 EES (Escrowed Encryption Standard)。其开发设计始于1985年，由 NSA 负责研究，1990 年完成评价工作。其算法为SKIPJACK，已由 Mykotronix 公司开发芯片产品，编程后为 MYK-78 (26美元/片)。算法属美国政府 SECRET 密级。但安全性与算法是否公开无关。

2 DES算法

- DES 发展史确定了发展公用标准算法模式，而 EES 的制定路线与 DES 的背道而驰。虽然，由美国政府精心选定的五人小组评估意见，不能解除人们对算法安全性的疑虑。人们怀疑有陷门和政府部门肆意侵犯公民权利。此举遭到广为反对，1995 年 5月 AT&T Bell Lab 的 M. Blaze 博士在 PC 机上用45分钟时间使 SKIPJACK 的 LEAF 协议失败，伪造 ID 码获得成功。虽 NSA 声称已弥补，但丧失了公众对此体制的信心。1995 年 7 月美国政府宣布放弃用 EES 来加密数据，只将它用于语音通信。
- 重新回到制定DES 标准立场。1997年1月美国 NIST 着手进行 AES (Advanced Encryption Standard) 的研究，成立了标准工作室。1997 年 4 月 15 日讨论了AES的评估标准，开始在世界范围征集 AES 的建议算法，截止时间为1998 年 6 月 15 日。1998 年 8 月 20~22 日经评审选定并公布了 15 个候选算法。

□ DES 概述

- DES (Data Encryption Standard) 是迄今为止世界上最为广泛使用和流行的一种分组密码算法，它的分组长度为 64 比特，密钥长度为 56 比特。
- DES 加密算法的流程如下，有 3 个阶段：
  1. 初始置换 IP，用于重排明文分组的 64 bit 数据。
  2. 具有相同功能的 16 轮乘积变换，每轮中都有置换和代换运算，第 16 轮变换的输出分为左右两半，并被交换次序。
  3. 逆初始置换 IP^{-1} （为 IP 的逆）从而产生 64 bit 的密文。

□ 初始置换和逆初始置换

- 初始置换IP：将 64 bit 明文的位置进行置换，得到一个乱序的 64 bit 明文组，而后分成左右两段，每段为 32 bit，以 L_0 和 R_0 表示，IP 中各列元素位置号数相差为 8，相当于将原明文各字节按列写出，各列比特经过偶采样和奇采样置换后，再对各行进行逆序。将阵中元素按行读出构成置换输出。
- 逆初始置换IP^{-1}：将 16 轮迭代后给出的 64 bit 组进行置换，得到输出的密文组。输出为阵中元素按行读得的结果。
- IP 和 IP^{-1} 在密码意义上作用不大，因为输入组x与其输出组 $y = IP(x)$(或$IP^{-1}(x)$) 是已知的一一对应关系。它们的作用在于打乱原来输入 x 的 ASCII 码字划分关系，并将原来明文的校验位 $x_8, x_{16}, \dots, x_{64}$ 变成为 IP 输出的一个字节。

□ 初始置换和逆初始置换

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

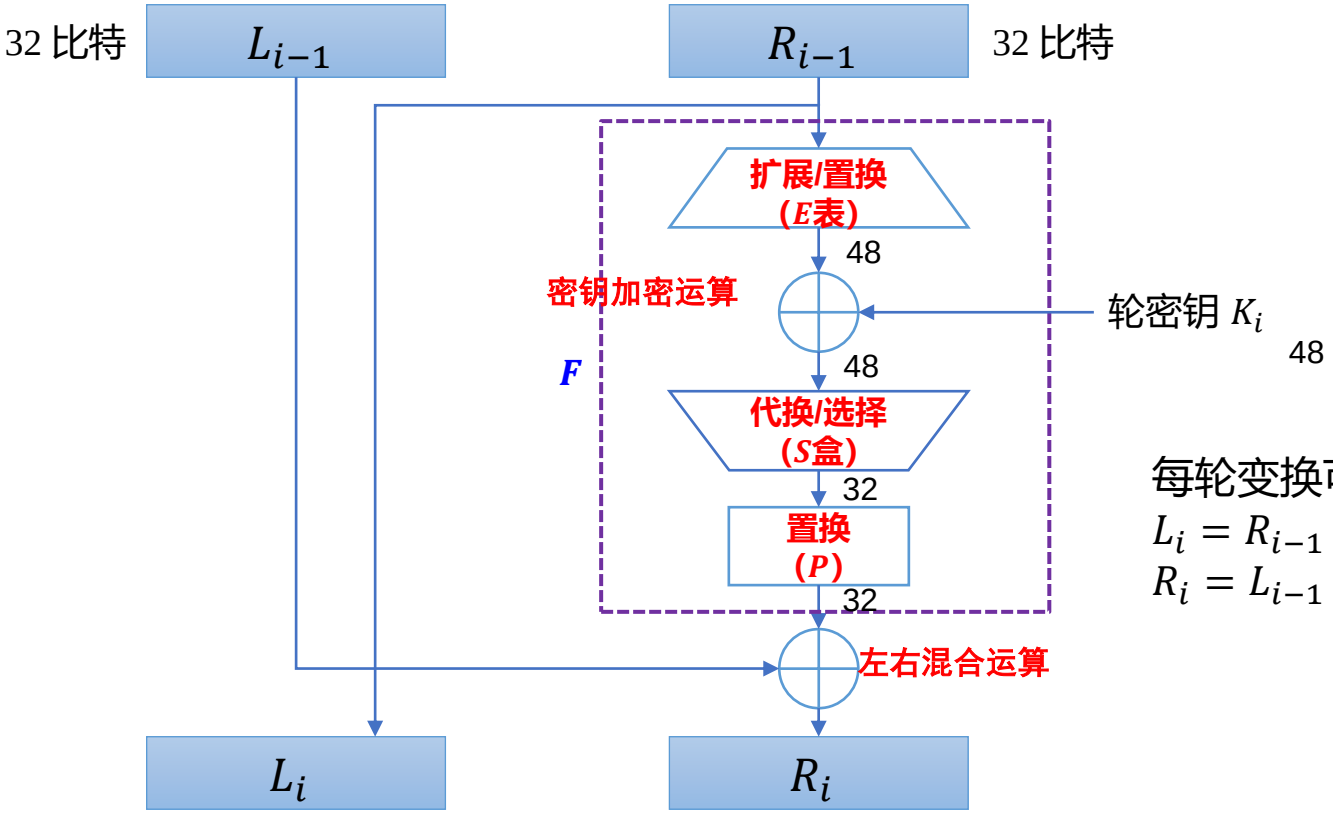
初始置换 IP

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

逆初始置换 IP^{-1}

口 16 轮乘积变换 （Feistel 结构）

- 在每轮迭代中，64 位的中间结果被分成左右两部分，且作为相互独立的 32 位数据进行处理。每轮迭代的输入是上轮的结果 L_{i-1} 和 R_{i-1} 。

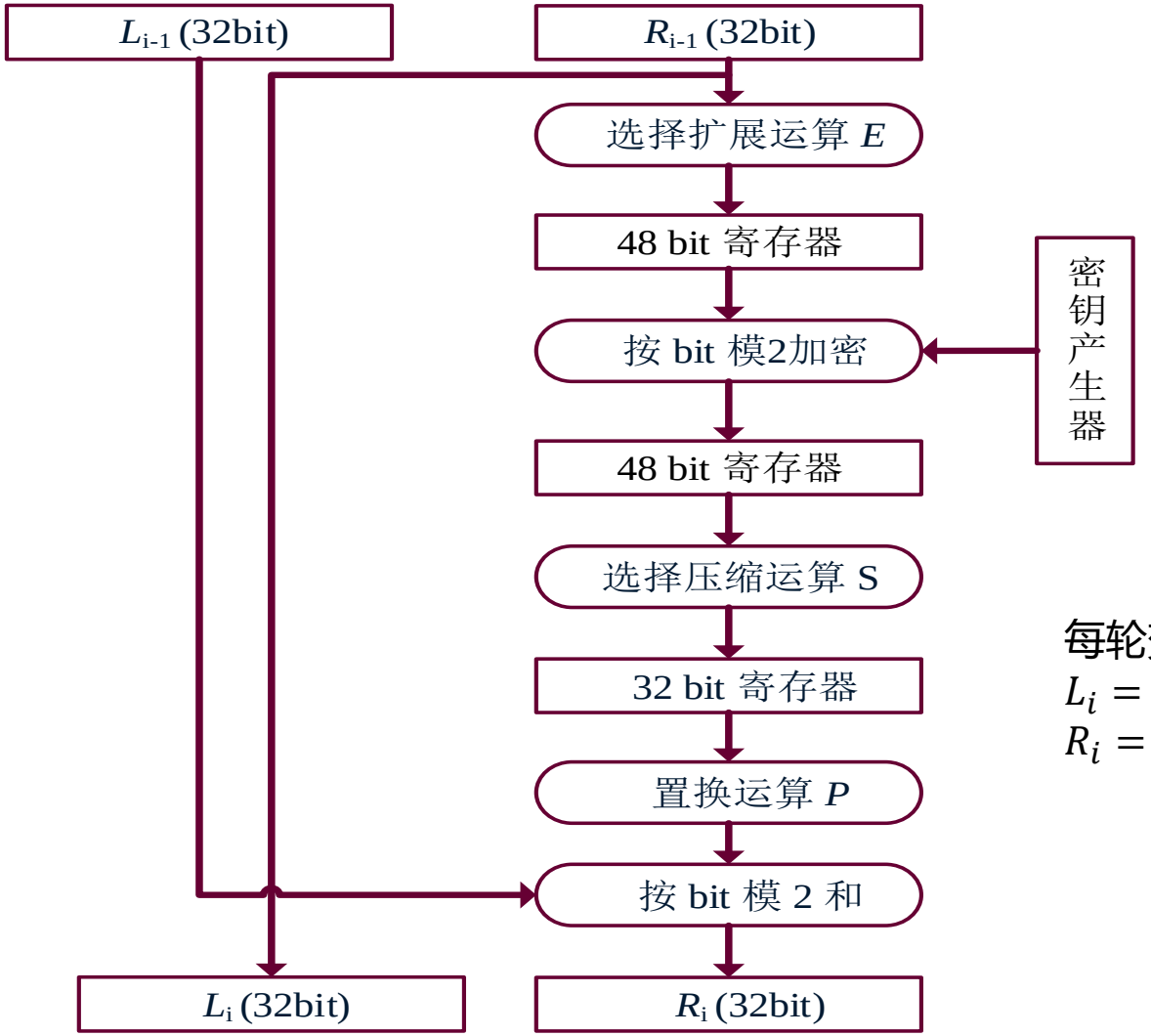


每轮变换可由以下公式表示：

$L_i = R_{i-1}$

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

□ 16 轮乘积变换 （Feistel 结构）



每轮变换可由以下公式表示：

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

□ 16 轮乘积变换 （Feistel 结构）

函数 F

- **选择扩展运算 E**: 将 32 bit 输入 R_{i-1} 扩展成 48 bit 输出。
- **密钥加密运算**: 将子密钥产生器输出的 48 bit 子密钥 k_i 与选择扩展运算 E 输出的 48 bit 数据按位模 2 相加。
- **选择压缩运算 S** : 将前面送来的 48 bit 数据自左至右分成 8 组，每组 6 bit。而后并行送入 8 个 S 盒，每个 S 盒为一非线性代换网络，有 4 bit 输出。
- **置换运算 P**: 对 S_1 至 S_8 盒输出的 32 bit 数据进行坐标置换。
- **左右混合运算**: 置换 P 输出的 32 bit 数据与左边 32 bit 即 L_{i-1}逐位模 2 相加，所得到的 32 bit 作为下一轮迭代用的右边的数字段。并将 R_{i-1} 并行送到左边的寄存器，作为下一轮迭代用的左边的数字段。

选择扩展运算 E

- 令 s 表示 E 原输入数据比特的原下标，则 E 的输出是将原下标 $s \equiv 0$ 或 $1(\mathrm{mod}4)$ 的各比特重复一次得到的，即对原第 32, 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29 各位都重复一次，实现数据扩展。将表中数据按行读出得到 48 bit 输出。

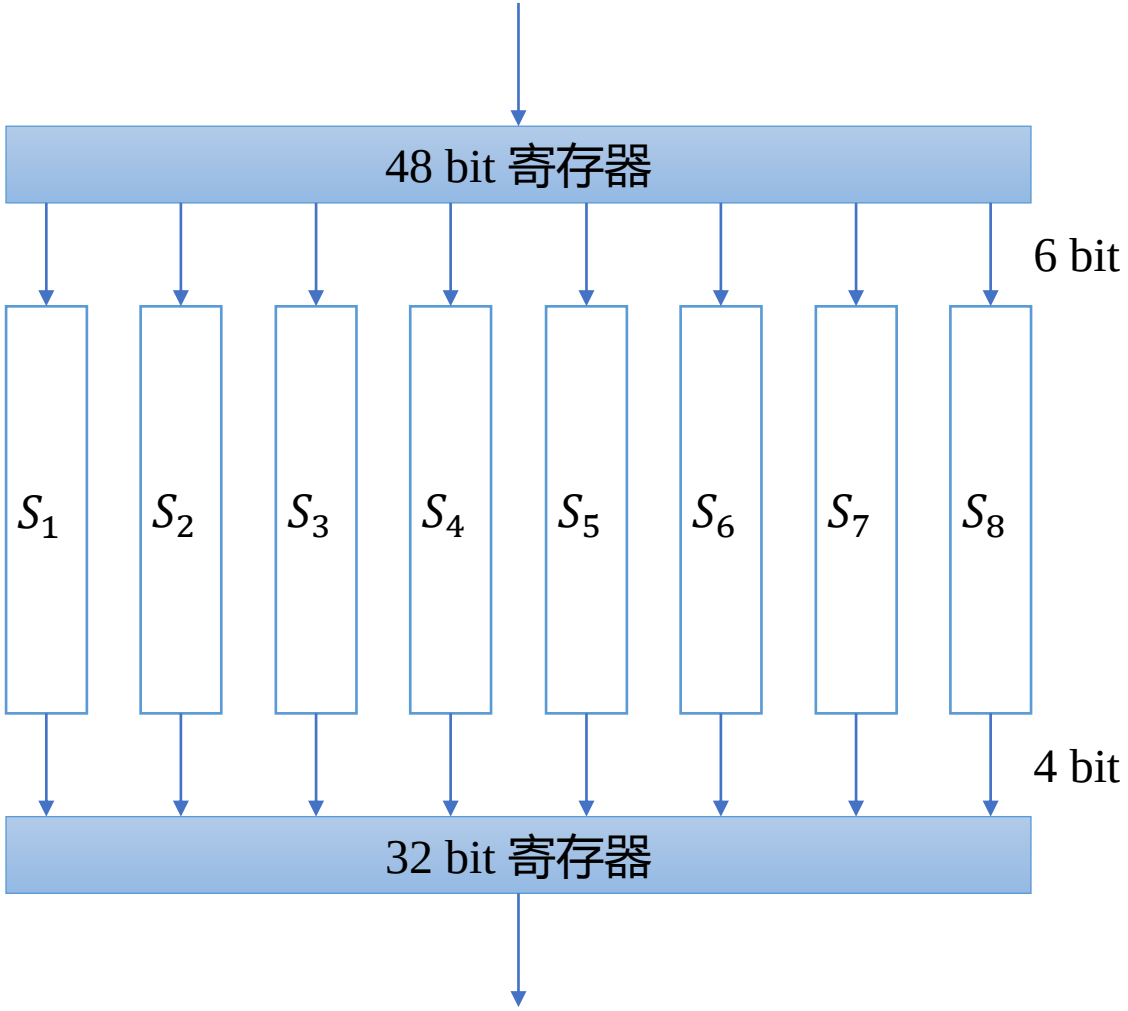
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32

扩展

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

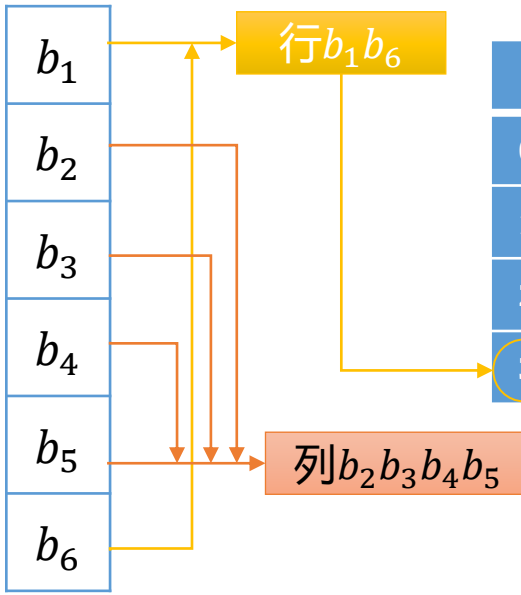
□ 选择压缩运算 S

- **选择压缩运算**由 8 个 S 盒构成，每个 S 盒都是 6 bit 输入，4 bit 输出。
- $S_i(b_1b_2b_3b_4b_5b_6)$ 输出值是对应表 S_i 中 $(b_2b_3b_4b_5)$ 列和 (b_1b_6) 行对应的值。
- S 盒对应于 Shannon 所提出的密码学思想中的混淆。



□ 选择压缩运算 S

- 以将 110011 输入到 S_1 中为例：



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

S_1盒

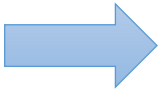
$$\frac{b_1b_2b_3b_4b_5b_6}{110011} \Rightarrow \frac{\text{行：}\ b_1b_6 = (11)_2 = 3}{\text{列：}\ b_2b_3b_4b_5 = (1001)_2 = 9} \Rightarrow \frac{S_1\text{盒3行9列}}{\text{值：}\ 11 = (1011)_2}$$

□ 置换运算 P

- **置换运算 P** 对 8 个 S 盒的输出（32 bit）进行坐标置换，对应于 Shannon 所提出的密码学思想中的扩散。

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

置换



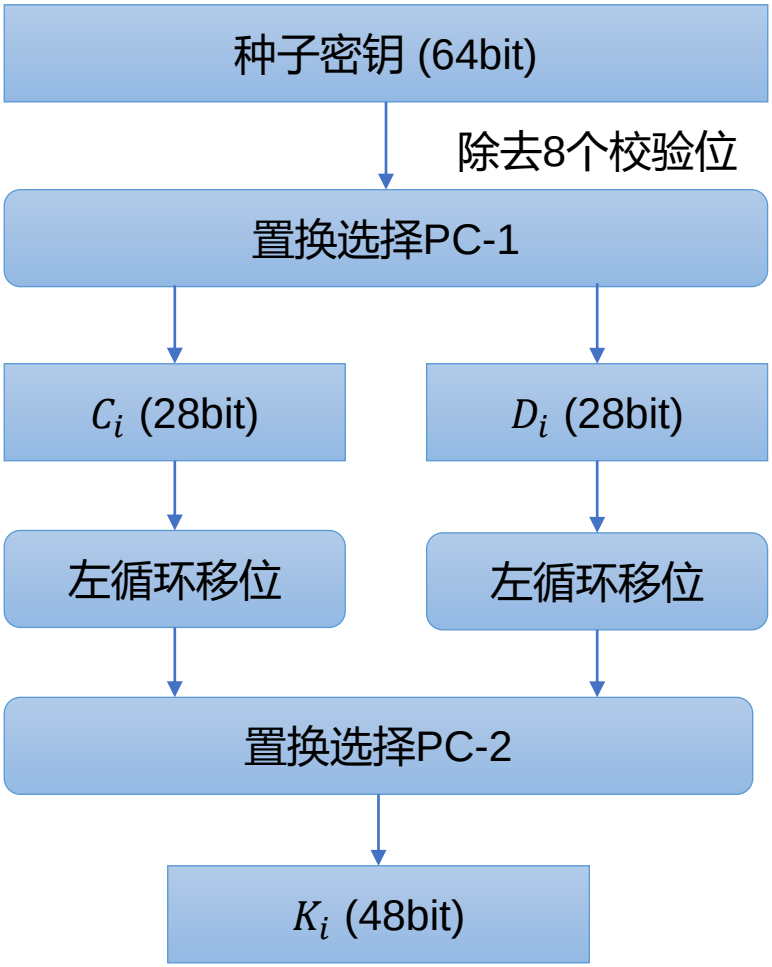
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

子密钥生成

子密钥产生器：将 64 bit 初始密钥经过置换选择 PC-1、循环移位置换、置换选择 PC-2 给出每次迭代加密用的子密钥K_i。

- 在64 bit 初始密钥中有8位为校验位，其位置号为 8、16、24、32、40、48、56 和 64。其余 56 位为有效位，用于子密钥计算。将这56位送入置换选择 PC-1。经过坐标置换后分成两组，每组为 28 bit，分别送入 C 寄存器和 D 寄存器中。
- 在各次迭代中，C 和 D 寄存器分别将存数进行左循环移位置换，所移位数与轮数有关。
- 每次移位后，将 C 和 D 寄存器原存数送给置换选择 PC-2。置换选择 PC-2 将 C 中第 9、18、22、25 位和 D 中第 7、10、15、26 位删去，并将其余数字置换位置后送出 48 bit 数字作为第 i 次迭代时所用的子密钥 K_i。

子密钥生成


种子密钥 (64bit)
除去8个校验位
置换选择PC-1
C_i (28bit)
D_i (28bit)
左循环移位
左循环移位
置换选择PC-2
K_i (48bit)

子密钥生成

置换选择 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

置换选择 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	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	52

左循环移位位数

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

口 加解密过程

DES 算法加解密过程可总结如下:

- 加密: 式 (1) 和式 (2) 的运算进行 16 次后就得到密文组。

$$L_0 R_0 \leftarrow IP \quad (\langle 64 \text{ bit 明文} \rangle)$$

$$L_i \leftarrow R_{i-1} \qquad i = 1, \ldots, 16 \qquad (1)$$

$$R_i \leftarrow L_{i-1} \oplus F(R_{i-1}, k_i) \qquad i = 1, \ldots, 16 \qquad (2)$$

$$\langle 64 \text{ bit密文} \rangle \leftarrow IP^{-1}(R_{16} L_{16})$$

- 解密: DES 的加密运算是可逆的，其解密过程可类似地进行。

$$R_{16} L_{16} \leftarrow IP \quad (\langle 64 \text{ bit密文} \rangle)$$

$$R_{i-1} \leftarrow L_i \qquad i = 16, \ldots, 1 \qquad (3)$$

$$L_{i-1} \leftarrow R_i \oplus F(R_{i-1}, k_i) \qquad i = 16, \ldots, 1 \qquad (4)$$

$$\langle 64 \text{ bit 明文} \rangle \leftarrow IP^{-1}(L_0 R_0)$$

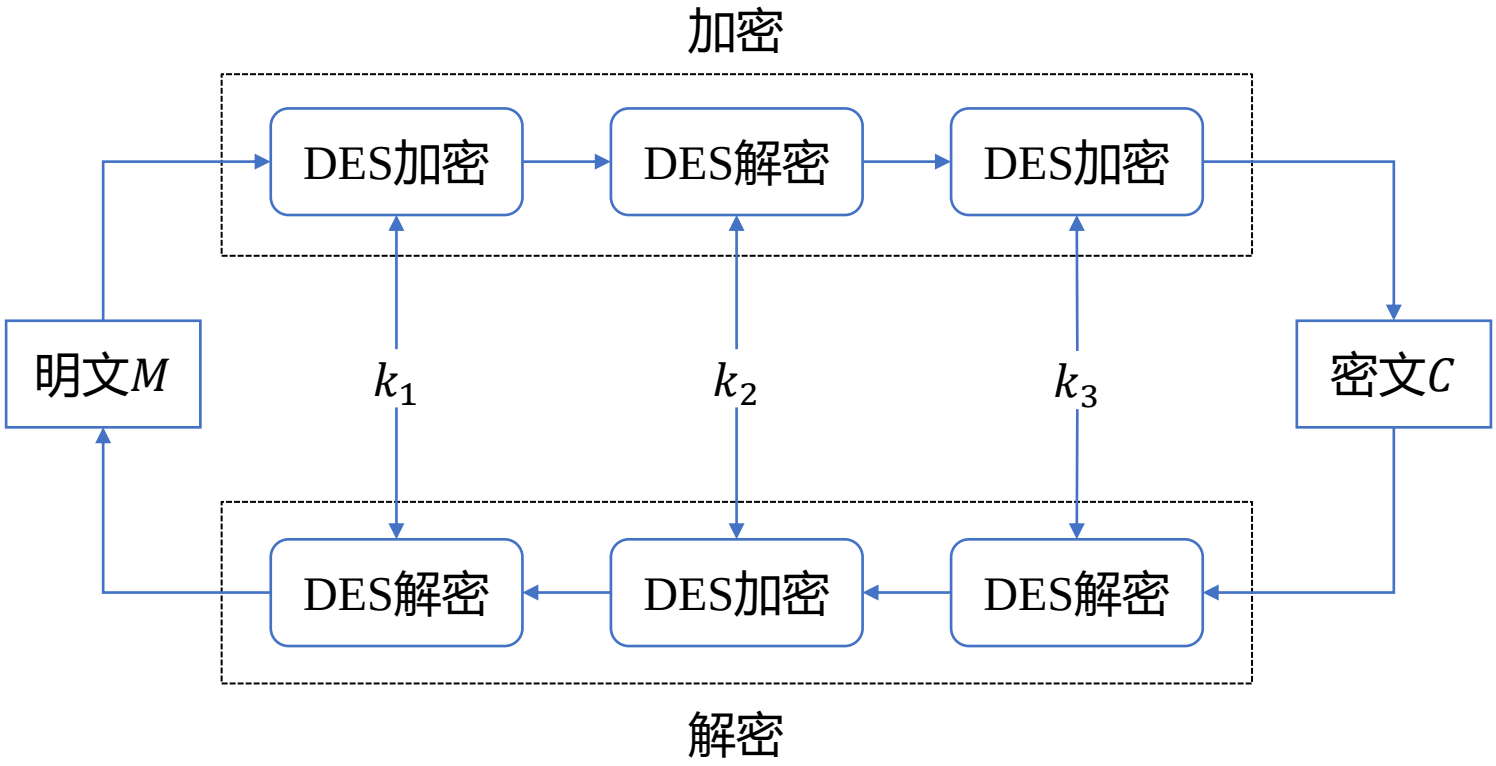
□ DES的安全性

- **密钥搜索攻击**：对 DES 安全性批评意见中，较为一致的看法是 DES 的密钥短。IBM 最初向 NBS 提交的建议方案采用 112 bit 密钥，但公布的 DES 标准采用64 bit 密钥。有人认为 NSA 故意限制 DES 的密钥长度。

- DES 的密钥量为2^{56} = 72 057 594 037 927 936 = 7.2×10^{16} 个。若要对 DES 进行密钥搜索破译，分析者在得到一组明文-密文对条件下，可对明文用不同的密钥加密，直到得到的密文与已知的明文-密文对中的相符，就可确定所用的密钥了。密钥搜索所需的时间取决于密钥空间的大小和执行一次加密所需的时间。若假定 DES 加密操作需时为 100μs (一般微处理器能实现)，则搜索整个密钥空间需时为 7.2×10^{12} 秒，近似为 2.28×10^{5} 年。若以最快的 LSI 器件，DES 加密操作时间可降到 5 μs，也要 1.1×10^{4} 年才能穷尽密钥。但是由于差分和线性攻击法的出现以及计算技术的发展，按 Wiener 介绍，在 1993 年破译 DES 的费用为100万美元，需时 3 个半小时。如果将密钥加大到 80 bits，采用这类搜索机找出一个密钥所需的时间约为 **6700 年**。

三重DES

为增强 DES 算法的安全性，人们提出许多DES的改进方案。3DES 是 DES的一个重要的改进算法，已在因特网的许多应用（如 PGP 和 S/MIME）中被采用。



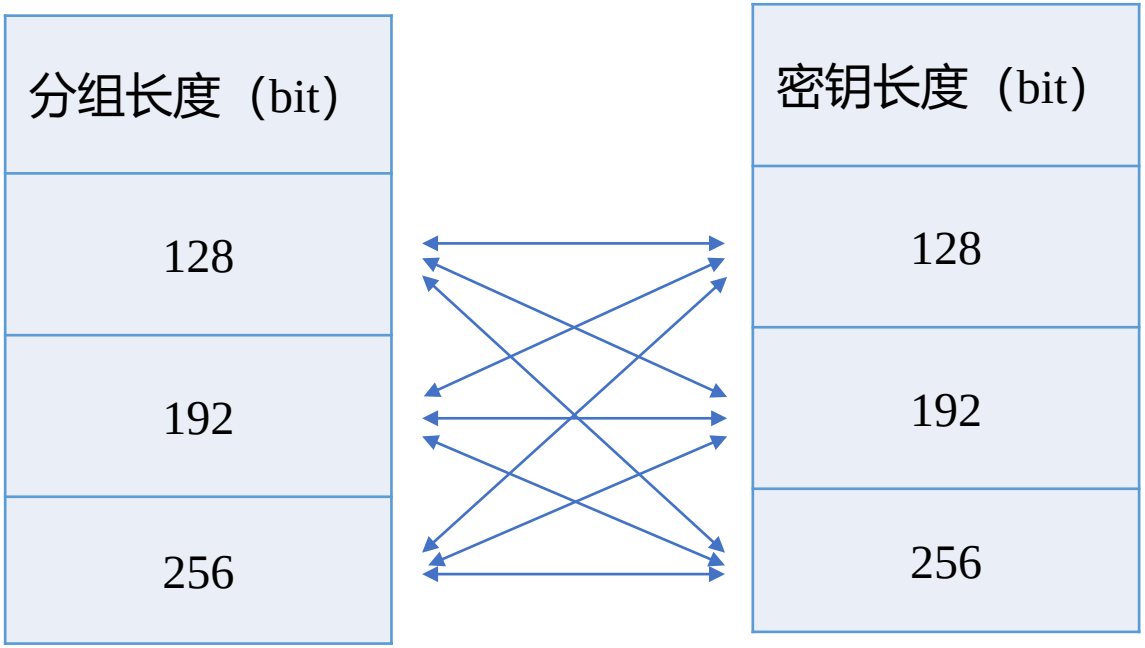
3DES 加解密示意图

□ AES概述

- 1997 年 4 月 5 日，NIST 发起征集 AES (Advanced Encryption Standard) 算法， 要求具有 128 比特的分组长度，并支持 128、192 和 256 比特的密钥长度，且要求 AES 能在全世界范围内免费使用。
- 1998 年 8 月 20 日，NIST 召开了第一次 AES 候选会议，并公布了15个AES候选算法。后来，NIST 又筛选出 5 个 AES 候选算法 (MARS, RC6, Rijindael, SERPENT, Twofish)。
- 2000 年 10 月，NIST 宣布由比利时的密码专家 Joan Daemen 博士和 Vincent Rijmen 博士开发的 Rijndael 算法做为 AES 的最终算法 。
- AES 的设计原则：
  - 能抵抗所有已知的攻击；
  - 在各种平台上易于实现，速度快；
  - 设计简单。

AES概述

- AES 是一个迭代型分组密码，其分组长度和密钥长度都可变，各自可以独立的指定为 128 比特、192 比特、256 比特。



AES的分组长度和密钥长度的组合

口 状态、种子密钥和轮数

- **状态 (State)**：密码运算中间结果称为状态，状态用以字节为基本构成元素的矩阵阵列来表示，该阵列有 4 行，列数记为 N_b：

$$N_b = \text{分组长度 (bits)} \div 32$$

N_b 取 4, 6, 8 对应分组长度 128, 192, 256 比特。算法的输入以字节为单位按 $a_{00}a_{10}a_{20}a_{30}a_{01}a_{11}a_{21}a_{31}$ ... 的顺序放置到状态阵列中。

- **种子密钥：**种子密钥类似地用一个 4 行的矩阵阵列来表示，列数记为N_k：

$$N_k = \text{密钥长度 (bits)} \div 32$$

N_k可以取的值为 4, 6, 8；对应的密钥长度为 128, 192, 256 比特。种子密钥以字节为单位按 $k_{00}k_{10}k_{20}k_{30}k_{01}k_{11}k_{21}k_{31}$ ... 的顺序放置到种子密钥阵列中。

- **轮数**：算法迭代的轮数记为 N_r； N_r 与 N_b 和 N_k 有关。

□ 状态、种子密钥和轮数

- 以 $N_b = 6$ 和 $N_k = 4$ 为例，状态和种子密钥的矩阵阵列表示如下：

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}

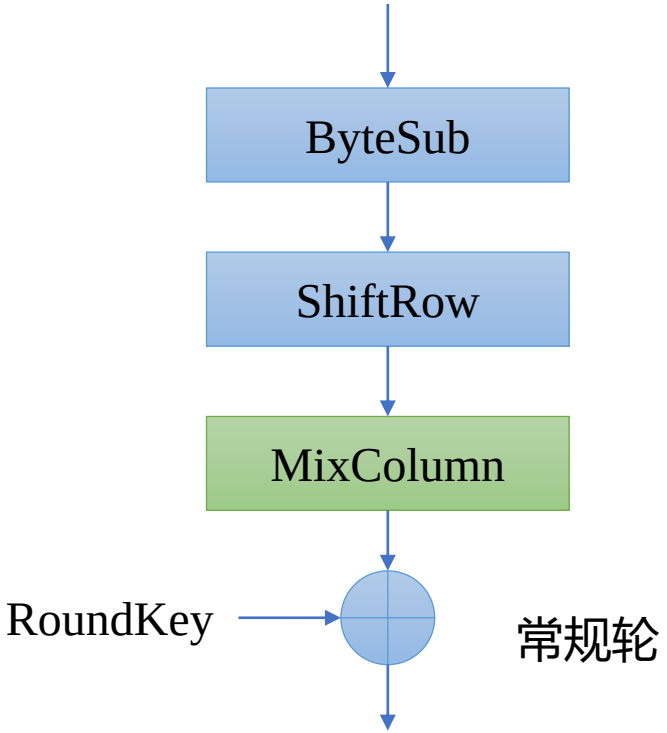
k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

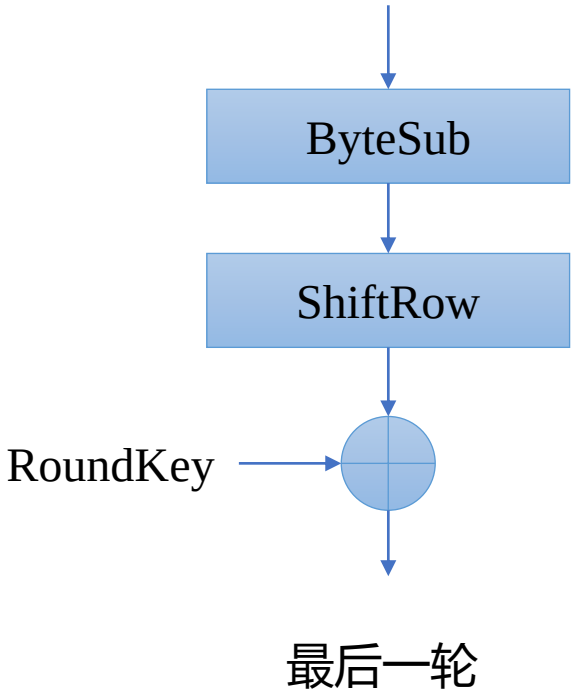
- 下表给出了 N_r 与 N_b 和 N_k 的关系：

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

轮函数

- AES 的轮函数由 4 个不同的计算部件组成，分别是字节代换 (ByteSub)、行移位 (ShiftRow)、列混合 (MixColumn)、密钥加 (AddRoundKey)。
- 最后一轮的轮函数与前面各轮不同，将 MixColumn 这一步去掉。





口 字节代换

- 字节代换(ByteSub)是一个非线性变换，独立地在每个状态字节上进行运算。它包括两个变换。
  - 在有限域 $GF(2^8)$ 上求乘法逆，00 映射到它自身。
  - 在 $GF(2)$ 上进行下面的仿射变换：

$$\begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ y_0 \end{bmatrix}^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

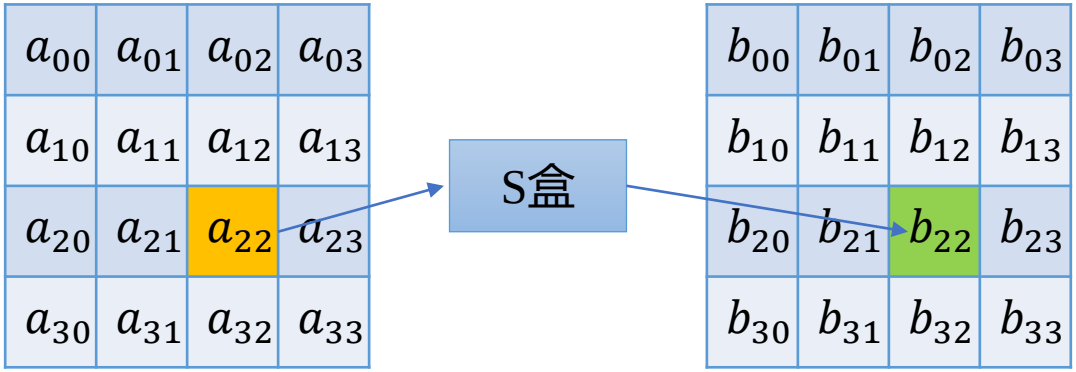
□ 字节代换

- 可预先将 $GF(2^8)$ 上的每个元素做字节代换形成代换表，也称 S 盒。

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

口 字节代换

- 进行字节代换时只需要查表操作：如果状态中的一个字节为 xy（x, y均为4 bit），则在 S 盒的第 x 行，第 y 列对应的字节为输出。



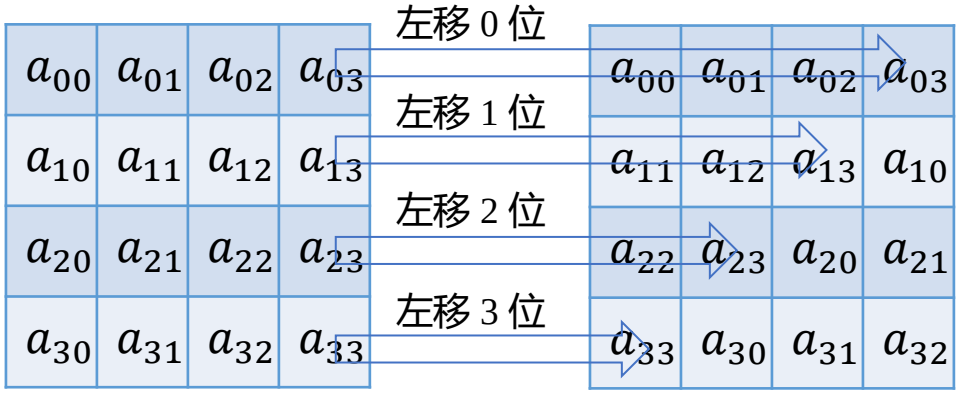
输入			
12	2A	21	0B
45	BD	04	C1
23	0A	00	1C
89	11	2A	FC

输出			
C9	E5	FD	2B
6E	7A	F2	78
26	67	63	9C
A7	82	E5	B0

口 行移位

- 行移位 (ShiftRow) 是将状态阵列的各行进行循环移位，不同状态行的位移量不同。第 0 行不移动，第 1 行循环左移 C_1 个字节，第 2 行循环左移 C_2 个字节，第 3 行循环左移 C_3 个字节
- 偏移量 C_1 、C_2 、C_3 与分组长度 N_b 有关，如下表（左）所示。当 $N_b = 4$ 时，行移位示意图如下图（右）所示。

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4



口 列混合

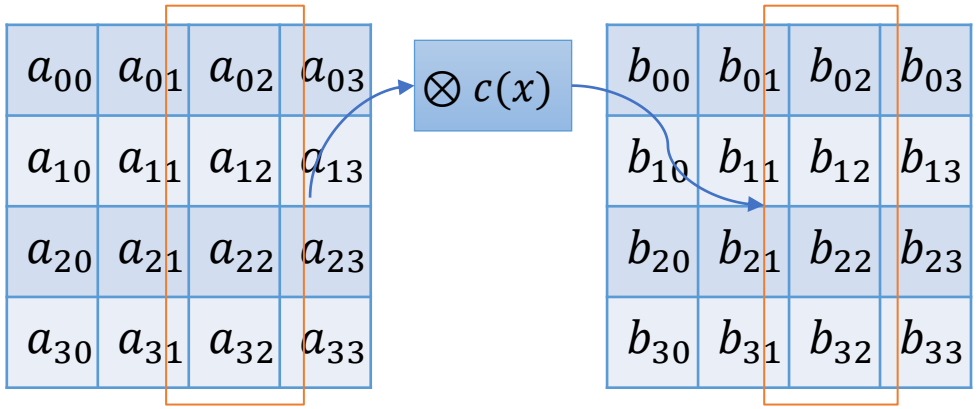
- 列混合（MixColumn）对一个状态逐列进行变换，即将每列视为有限域$GF(2^8)$上的多项式，再与一个固定的多项式$c(x)$进行模x^4+1乘法：

$$c(x) = 03x^3 + 01x^2 + 01x + 02$$

- 列混合运算也可以写为矩阵乘法。设$b(x) = c(x) * a(x)$，则

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

- 列混合运算示意图：



口 密钥加

- 密钥加(AddRoundKey)是将轮密钥简单地与状态进行逐比特异或，即：

$$b_{ij} = a_{ij} \oplus k_{ij}。$$

- 密钥加运算示意图：

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

$\oplus$

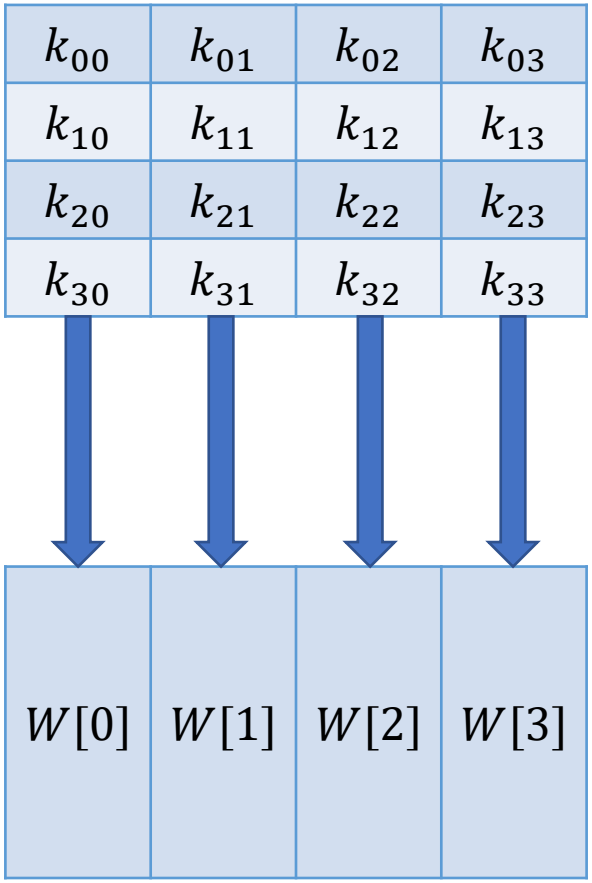
k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

$=$

b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	b_{33}

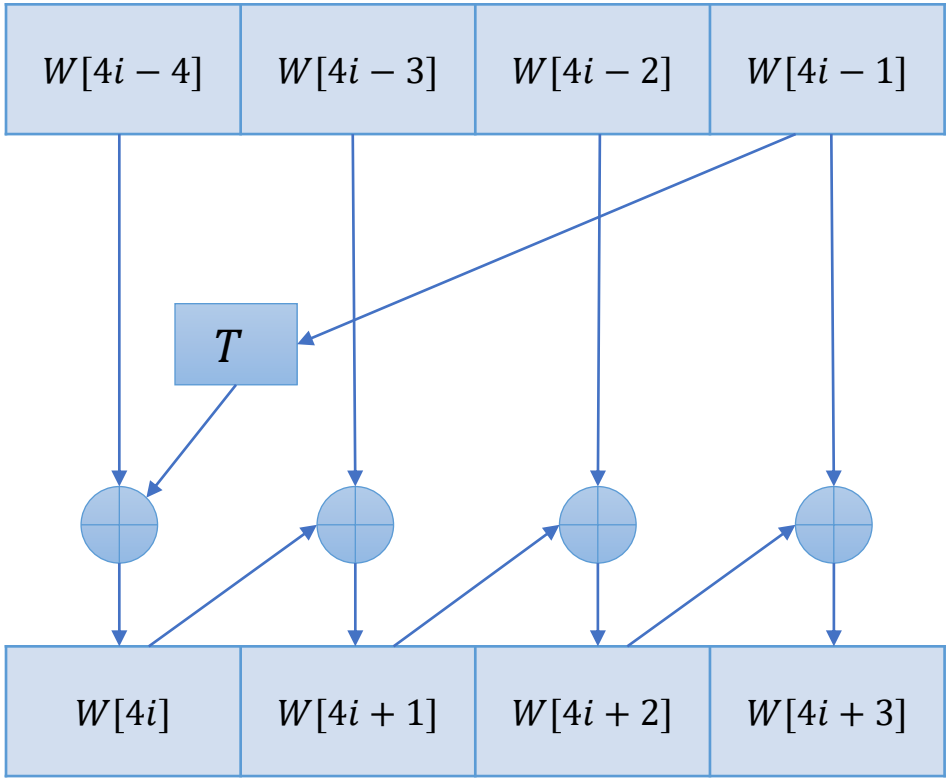
□ 密钥编排

- 密钥编排是指从种子密钥得到轮密钥的过程，包括两个部分：密钥扩展和轮密钥选取。
- 密钥比特总数 = 分组长度 × (轮数 + 1)
  - 例如：分组长度 = 128 和轮数 = 10 时，轮密钥长度为 128×(10 + 1) = 1408
- 首先将种子密钥输入到一个 4*4 矩阵中，每列组成 1 个字。



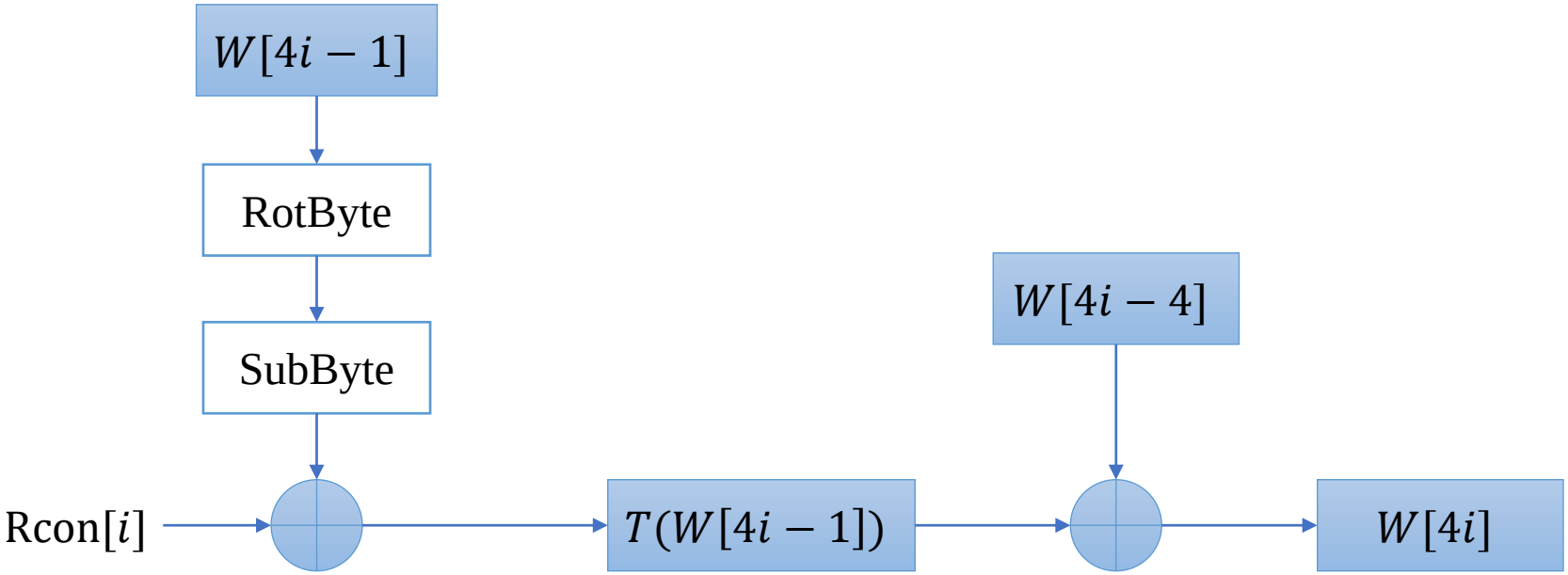
密钥编排

- **密钥扩展** 对 W 数组扩展 40 个新列，构成 44 列的密钥数组：
  - 如果 4 不整除 i，则第 i 列为：
    $W[i] = W[i-4] \oplus W[i-1]$
  - 如果 4 整除 i，则第 i 列为：
    $W[i] = W[i-4] \oplus T[W[i-1]]$



密钥编排

- T 函数由三部分组成：1 字节的循环移位 RotByte 、用 S 盒进行变换 SubByte 、异或轮常数 Rcon[i]。示意图如下：

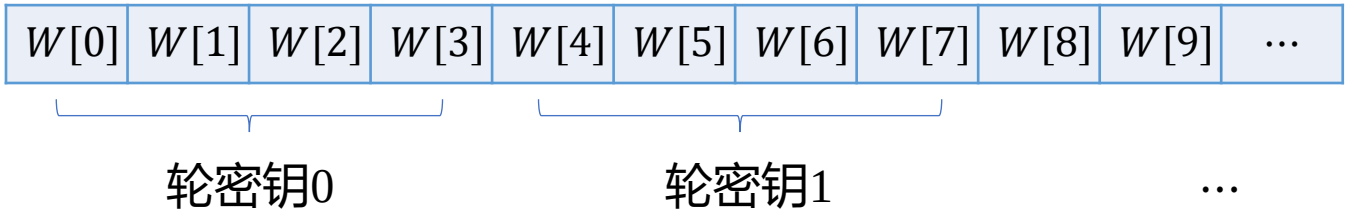


密钥编排

- 函数 RotByte 返回 4 字节字，当输入字为 (a,b,c,d) 时，输出字为 (b,c,d,a)，即左移 1 字节。
- 函数 SubByte 也返回 4 字节字，其中每一个字节都是用 AES 的 S 盒作用到输入字对应的字节得到。
- 轮常数 Rcon[i] 的右边 3 个字节总是 0， 记为 Rcon[i]=(RC[i],0,0,0)，轮数 = 10 时的 RC[i] 表如下：

i	1	2	3	4	5	6	7	8	9	10
RC[i]（16进制）	01	02	04	08	10	20	40	80	1B	36

- **轮密钥选取**（以 $N_k = 4$ 为例）：

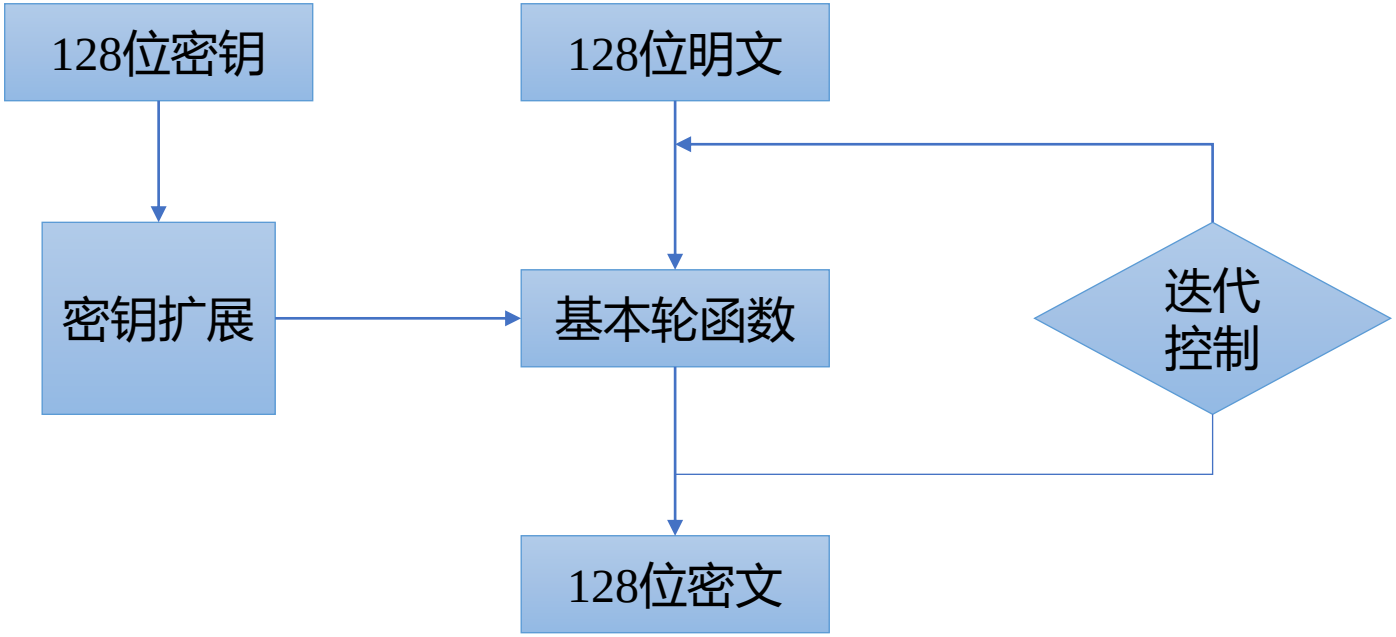


□ SM4 概述

- 国家商用密码管理办公室制定了一系列国家密码标准，包括 SM1、SM2、SM3、SM4、SM7、SM9、祖冲之密码算法。其中 SM1、SM4、SM7、祖冲之密码是对称算法；SM2、SM9 是非对称算法；SM3 是哈希算法。
- SM4 是分组密码算法，用于 WAPI (WLAN Authentication & Privacy Infrastructure) 。分组长度 128 比特，密钥长度 128 比特。加密算法与密钥扩展算法都采用 32 轮非线性迭代结构，以字节（8 位）和字节（32 位）为单位进行数据处理。
- SM4 密码算法的基本运算有模 2 加和循环移位。
  - 模 2 加：记为⊕，为 32 位逐比特异或运算。
  - 循环移位：$<<< i$，把 32 位字循环左移 i 位。

□ SM4的算法结构

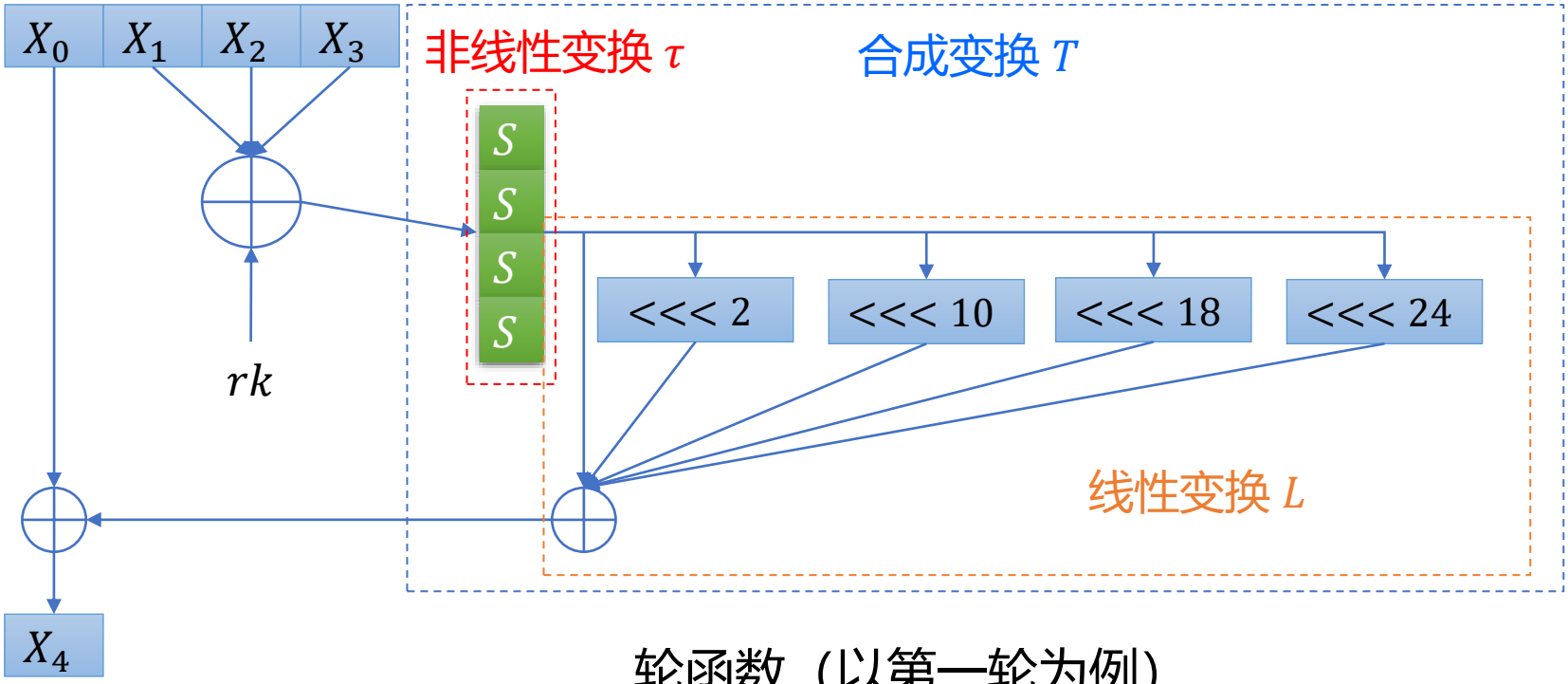
- SM4 的算法结构如下图所示：



轮函数

- 设轮函数的输入为 4 个 32 位字 (X_0, X_1, X_2, X_3)，共128 位，轮密钥为一个 32 位的字 rk。输出也是一个 32 位的字，由下式给出：

$$F(X_0, X_1, X_2, X_3, rk) = X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk)$$



轮函数（以第一轮为例）

非线性变换

- 非线性变换部件 τ 由 4 个 S 盒并置构成，其密码学作用是混淆。每个 S 盒的输入和输出都是 8 位的字节。设输入字节为 a，输出字节为 b，则 S 盒的运算规则可表示为

$$b = S(a)$$

- 例：设输入为“EF”，则行号为 E，列号为 F，于是 S 盒的输出值为表中第 E 行和第 F 列交叉点的值：

$$S(EF) = 84$$

4 中国商用密码算法SM4

□ 非线性变换

	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

线性变换

- 线性变换部件 L 是以字为处理单位的线性变换，其输入输出都是 32 位的字，它的密码学作用是扩散。

- 设 L 的输入为字 B，输出为字 C，则

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

合成变换 T 由非线性变换 τ 和线性变换 L 复合而成，数据处理单位是字。设输入为字 X，则先对 X 进行非线性 τ 变换，再进行线性 L 变换。记为

$$T(X) = L(\tau(X))$$

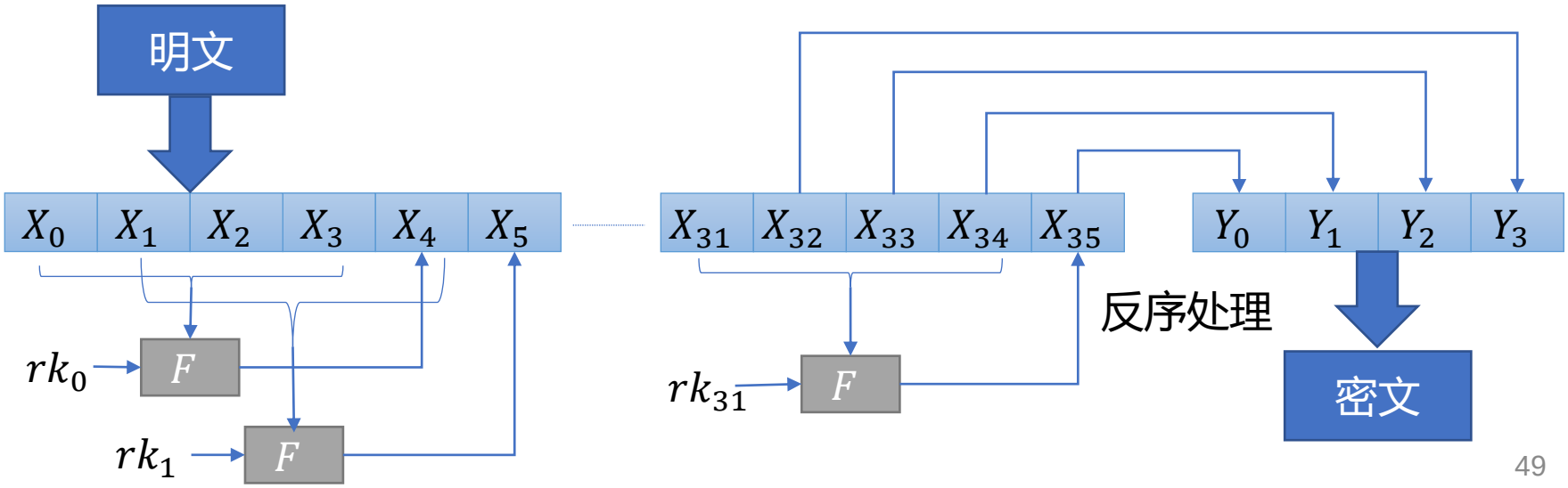
加解密算法

- 设输入明文为 (X_0, X_1, X_2, X_3)，输入的轮密钥为 rk_i $(i = 0,1,\ldots,31)$。则输出的密文为 (Y_0, Y_1, Y_2, Y_3)。加密算法描述如下：

$$\text{For } i = 0,1,\ldots,31 \text{ Do } X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i)$$

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

- 为了与解密算法需要的顺序一致，在加密算法后还需要一个反序处理。解密算法与加密算法相同，只是轮密钥的使用顺序相反。



□ 密钥扩展算法

- 设输入的加密种子密钥为 $MK = (MK_0, MK_1, MK_2, MK_3)$，输出轮密钥为 $rk_i (i = 0,1, \dots, 30,31)$，密钥扩展算法描述如下：

  - $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$

  - For $i = 0,1, \dots, 31$ Do

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

其中的变换 T' 与加密算法轮函数中的 T 基本相同，只将其中的线性变化 L 修改为以下的 L'：

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

□ 密钥扩展算法：常数 FK 和固定参数 CK

- 在密钥扩展算法中使用如下的常数：

$$FK_0 = (\text{A3B1BAC6}), FK_1 = (\text{56AA3350}),$$
$$FK_2 = (\text{677D9197}), FK_3 = (\text{B27022DC})$$

- 共使用 32 个固定参数 CK_i，每个 CK_i 是一个字，其产生规则如下：

设 $ck_{i,j}$ 为 CK_i 的第 j 个字节 $(i = 0,1,\dots,31; j = 0,1,2,3)$，即 $CK_i = (ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3})$，则 $ck_{i,j} = (4i + j) \times 7 (\text{mod } 256)$。

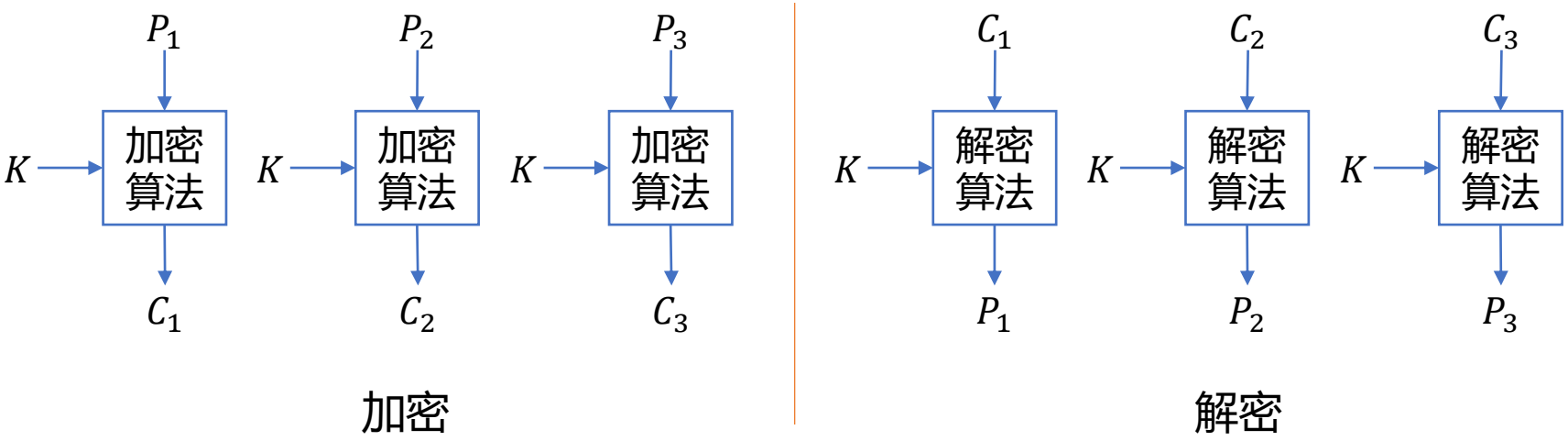
00070E15	1C232A31	383F464D	545B6269
70777E85	8C939AA1	A8AFB6BD	C4CBD2D9
E0E7EEF5	FC030A11	181F262D	343B4249
50575E65	6C737A81	888F969D	A4ABB2B9
C0C7CED5	DCE3EAF1	F8FF060D	141B2229
30373E45	4C535A61	686F767D	848B9299
A0A7AEB5	BCC3CAD1	D8DFE6ED	F4FB0209
10171E25	2C333A41	484F565D	646B7279

分组密码的工作模式

- 分组密码在加密时明文分组的长度是固定的，而实际中待加密消息的数据量是不定的，数据格式可能是多种多样的。为了能在各种应用场合使用分组密码算法，需要考虑分组密码的工作模式。另一方面，一些工作模式也可以隐蔽明文的统计特性、数据格式等，以提高整体的安全性，降低删除、重放、插入和伪造成功的机会。
- 美国在 FIPS 中定义了五种工作模式：
  - 电码本模式（ECB）
  - 密码分组链接模式（CBC）
  - 输出反馈模式（OFB）
  - 密码反馈模式（CFB）
  - 计数器模式（CTR）

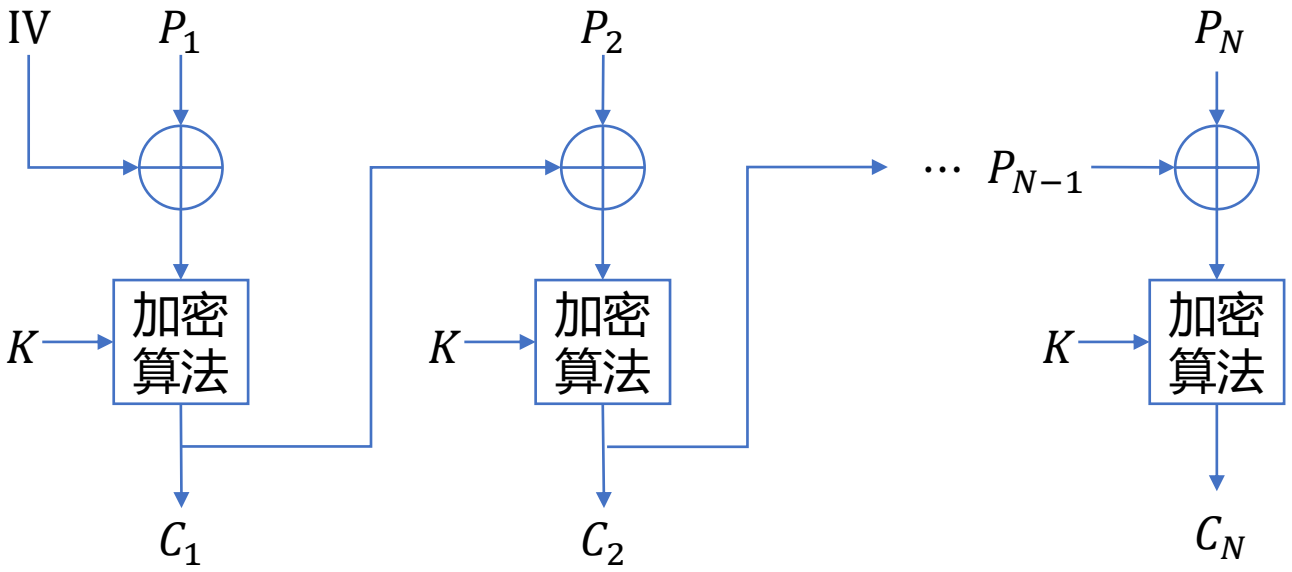
□ 电码本模式 (ECB)

- ECB 模式直接用分组密码算法来进行消息的加密和解密。
  - 一个明文分组被加密成一个密文分组，相同的明文分组加密成相同的密文分组。明文分成 64 bit（根据算法确定）的分组进行加密，必要时填充。
  - Padding（填充）：加密消息按 64 bit 分组时，最后一组消息长度可能不足 64 bit 时可以填充一些字符。



□ 密码分组链接模式 (CBC)

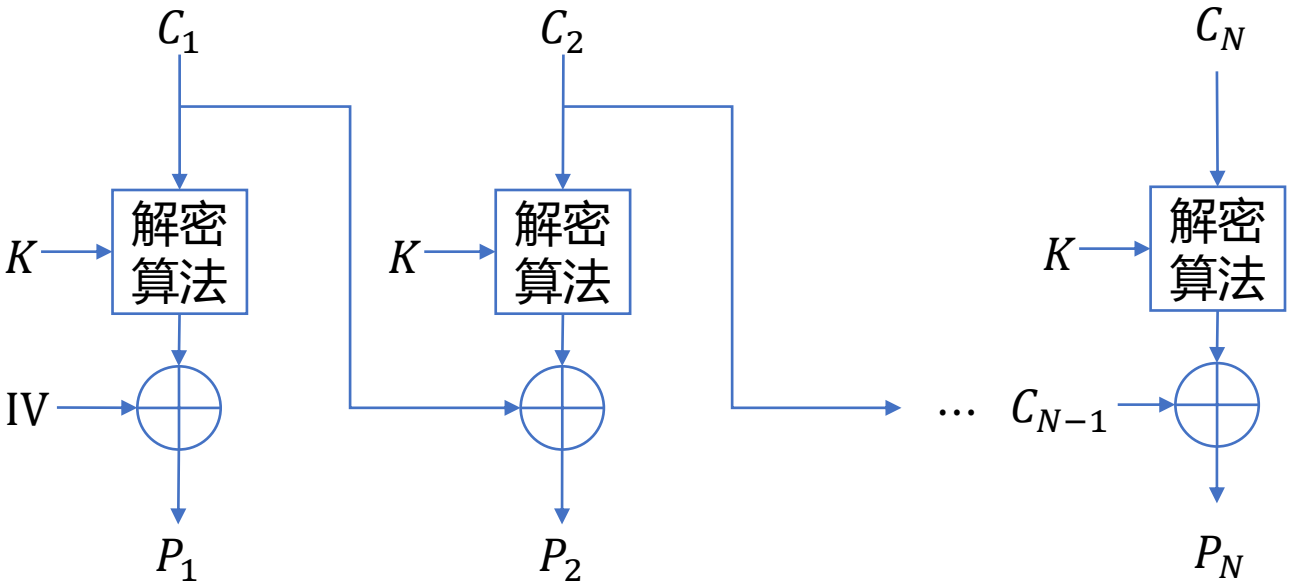
- CBC 模式中，加密算法的输入是当前明文分组和前一密文分组的异或。加密算法的输入不会显示出与这次明文分组之间的固定关系，所以重复的明文分组不会在密文中暴露出这种重复关系。
- 最开始时，需要有一个初始向量 IV 与第一个明文分组异或。



$$C_1 = E_k(P_1 \oplus \text{IV}) \qquad C_i = E_k(P_i \oplus C_{i-1}), i = 2,3,4,\ldots,N$$

□ 密码分组链接模式 (CBC)

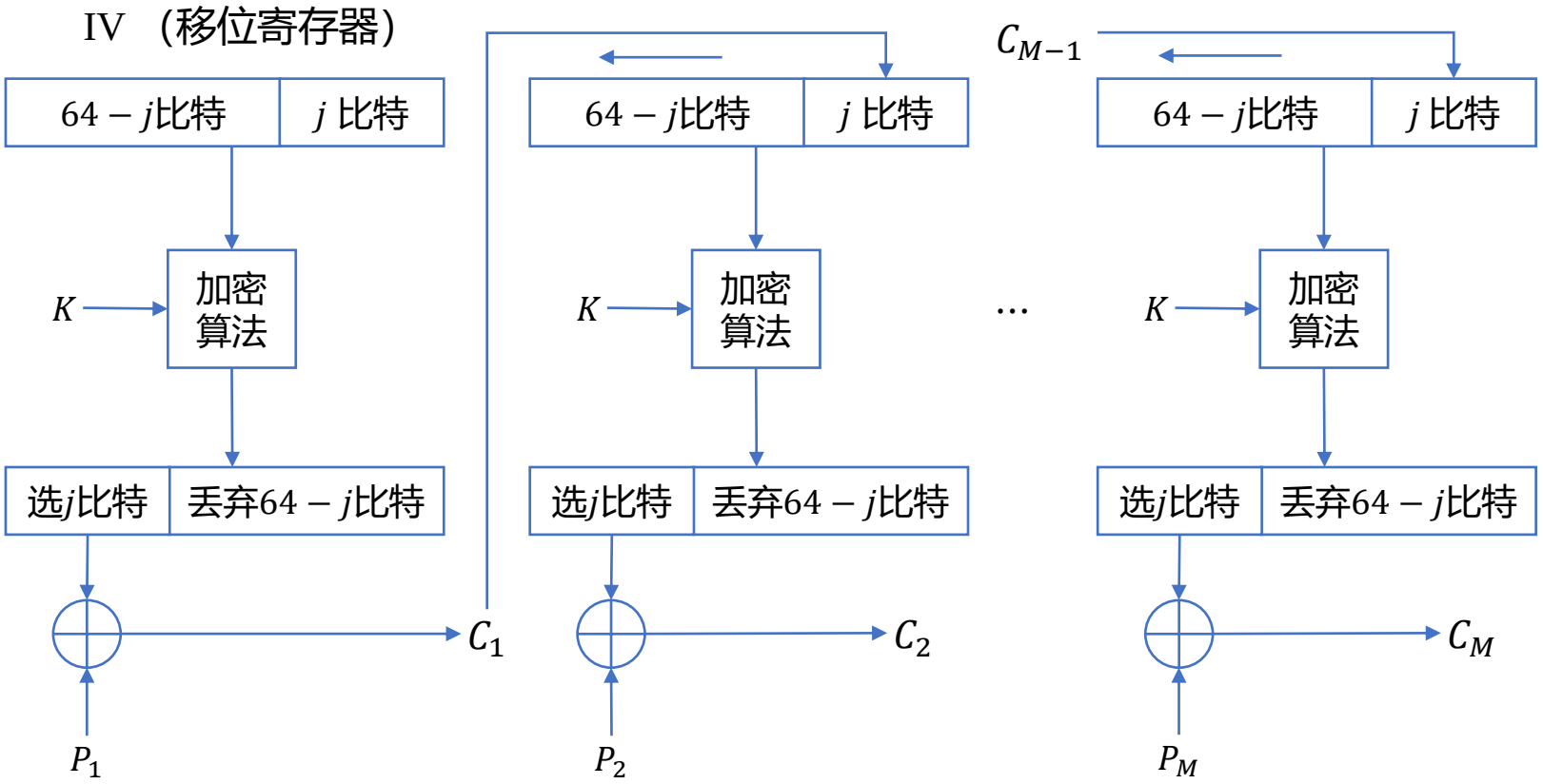
- CBC 模式解密时，IV 和解密算法对第一个密文分组的输出进行异或以恢复第一个明文分组。



$$P_1 = D_k(C_1) \oplus \text{IV} \qquad P_i = D_k(C_i) \oplus C_{i-1}, i = 2,3,4,\ldots,N$$

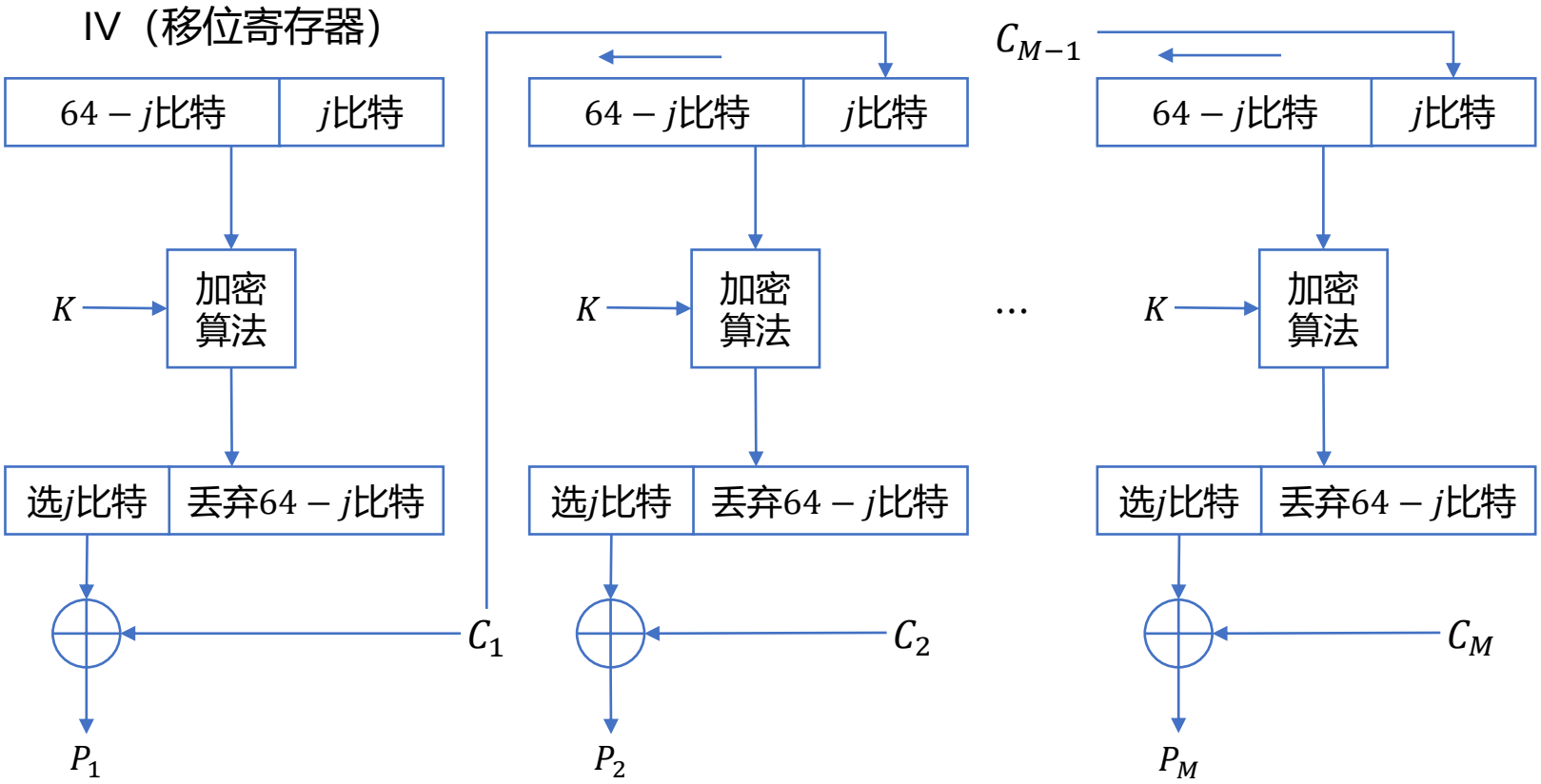
□ 密码反馈模式 (CFB)

- CFB 模式将分组密码转换为流密码。输入是 64 比特的移位寄存器，其初值为初始向量 IV。输出最左边 j 比特与明文第一个单元 P_1 进行异或，产生密文的第 1 个单元 C_1。然后将移位寄存器左移 j 位，并将 C_1 送入移位寄存器的最右边 j 位，直至明文所有单元被加密。



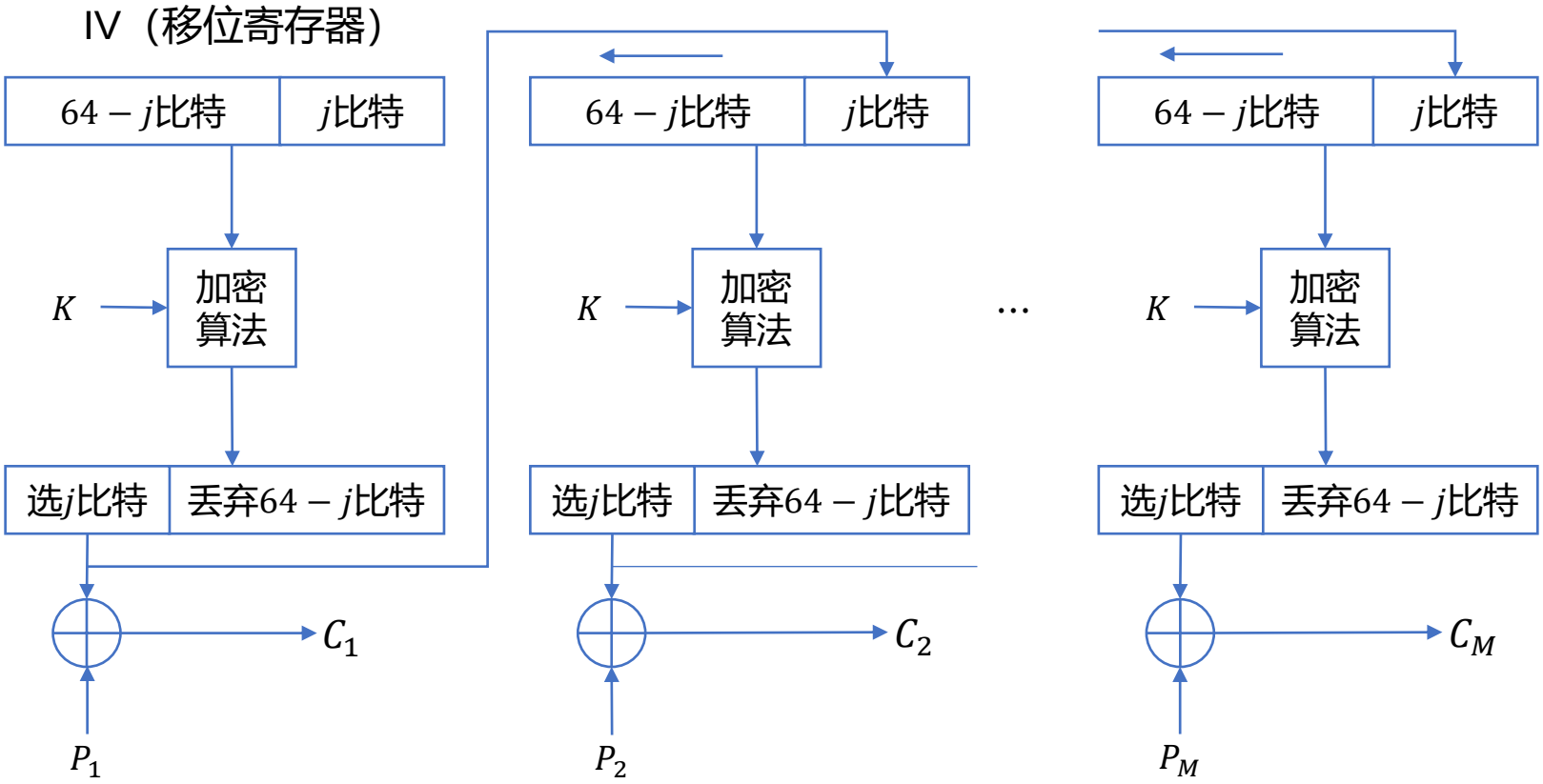
□ 密码反馈模式 (CFB)

- CFB 模式的解密仍使用分组密码的加密算法而不是解密算法。



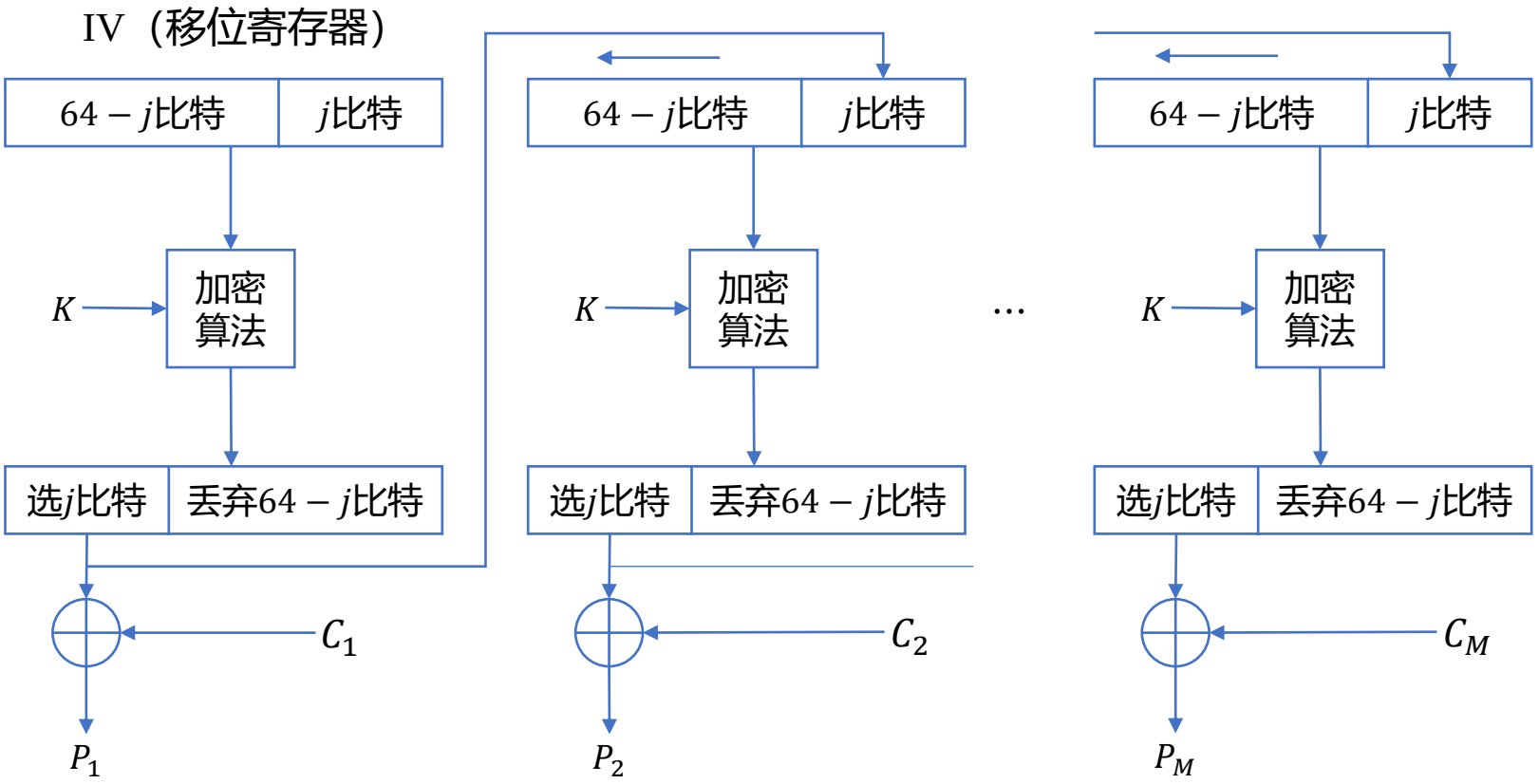
口 输出反馈模式 (OFB)

- OFB 模式的结构与 CFB 的结构类似，不同之处在于：OFB 模式将分组密码的加密算法的输出反馈到移位寄存器。



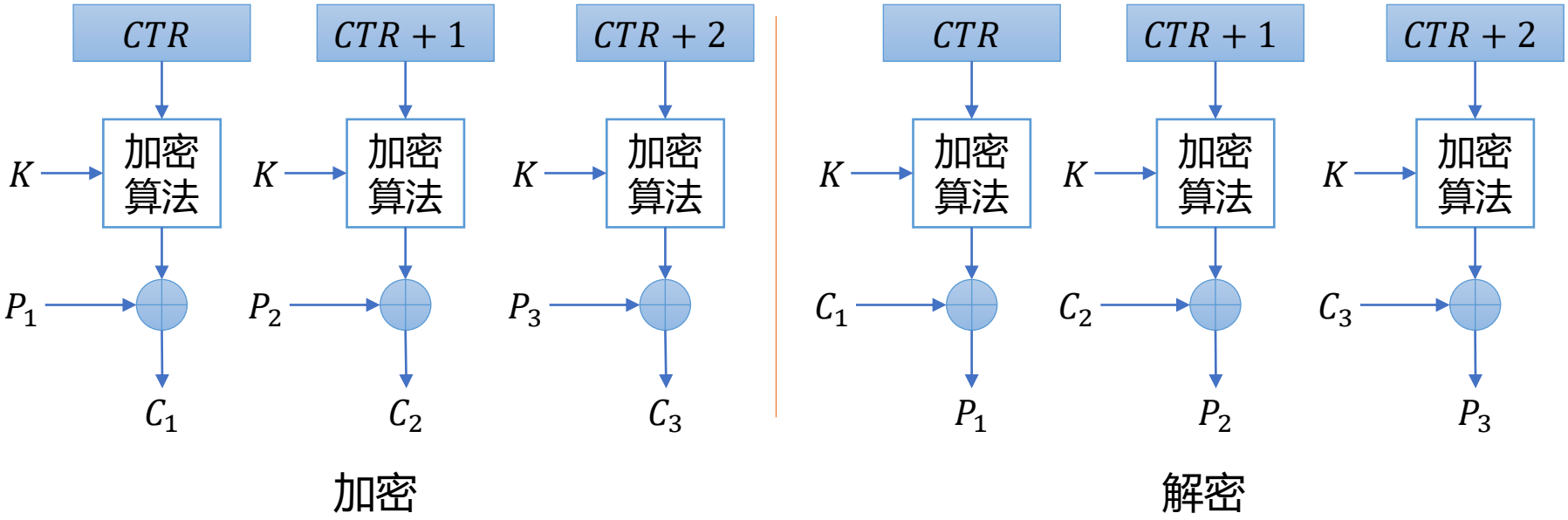
口 输出反馈模式 (OFB)

- 类似于 CFB 模式，OFB 模式的解密使用的是分组密码的加密算法。



口 计数器模式 (CTR)

- CTR 模式使用与明文分组相同的计数器长度，但加密不同的分组所用的计数器值必须不同。计数器从某一初值开始，依次递增 1。计数器值经分组密码的加密算法变换后的结果与明文分组异或，从而得到密文；解密时使用相同的计数器值序列，经加密算法变换后的计数器值与密文分组异或，从而恢复明文。



口 工作模式的比较和选用

- ECB 模式：简单、高速， 但最弱、易受重放攻击，一般不推荐。
- CBC 适用于文件加密，但较 ECB 慢。安全性加强。
- OFB 和 CFB 较 CBC 慢许多。每次迭代只有少数比特完成加密。若可以容忍少量错误扩展，此时用 CFB。在字符为单元的流密码中多选 CFB 模式。
- OFB 用于高速同步系统，没有错误传播。
- CTR 适用于对实时性和速度要求比较高的场合。

5 分组密码的工作模式

□ 工作模式的比较和选用

模式	描述	特点	用途
ECB	每个明文组独立地以同一密钥加密	明文相同，密文相同；操作简单，容易受到重放和代换攻击；无差错传播	传送短数据（如一个加密密钥)
CBC	加密算法的输入是当前明文组和前一密文组	初始向量 IV 保密和完整性，密文差错传播	传输数据分组；认证
CTR	加密算法的输入为当前计数器的值，该输出值与当前明文异或以产生密文	只需加密，没有解密；简单，可预处理，并行处理	实时性和速度要求比较高的加密场合
OFB	与CFB相似，不同之处是本次加密算法的输入为前一次加密算法的输出	分组密码算法作为一个密钥流产生器，对于密文被篡改难以进行检测	有扰信道上(如卫星通讯)传送数据流
CFB	每次只处理输入的 j 比特，将上一次密文用作加密算法的输入以产生伪随机输出，该输出与当前明文异或以产生密文	有差错传播	传送数据流，认证

小结

- 私钥密码体制的特征是加密密钥和解密密钥相同，特点是加、解密速度快，密钥相对较短。私钥密码包括流密码和分组密码。
- 分组密码的加密算法通常由多轮相同或相近的轮函数组成，每一轮所用的密钥由密钥扩展算法产生；分组密码的解密算法是加密算法的逆操作。
- 分组密码的设计理念体现了 Shannon 密码学的思想：扩散与混淆。其中置换或移位的作用是扩散，非线性变换组件 S 盒的作用是混淆。
- 为了能在各种应用场合高效安全的使用分组密码算法，需要考虑分组密码的工作模式，包括电码本模式（ECB）、密码分组链接模式（CBC）、输出反馈模式（OFB）、密码反馈模式（CFB）和计数器模式（CTR）。


西安电子科技大学
XIDIAN UNIVERSITY
团结勤奋
求实创新
谢 谢！
THANK YOU