

第二讲：单钥体制

- 一、分组密码概述
- 二、分组密码 DES
- 三、分组密码运行模式
- 四、分组密码 AES

一、分组密码概述

分组密码：单钥分组密码是许多系统安全的一个重要组成部分。分组密码易于构造伪随机数生成器、流密码、消息认证码 (MAC) 和杂凑函数等，还可进而成为消息认证技术、数据完整性机构、实体认证协议以及单钥数字签字体制的核心组成部分。实际应用中对于分组码可能提出多方面的要求，除了安全性外，还有运行速度、存储量 (程序的长度、数据分组长度、高速缓存大小)、实现平台 (硬、软件、芯片)、运行模式等限制条件。这些都需要与安全性要求之间进行适当的折衷选择。

分组密码

分组密码：明文序列： $x_1, x_2, \dots, x_i, \dots$

加密函数 $E: V_m \times K \rightarrow V_n$

密钥： $k = (k_0, k_1, \dots, k_{t-1})$

这种密码实质上是字长为 m 的数字序列的代换密码。

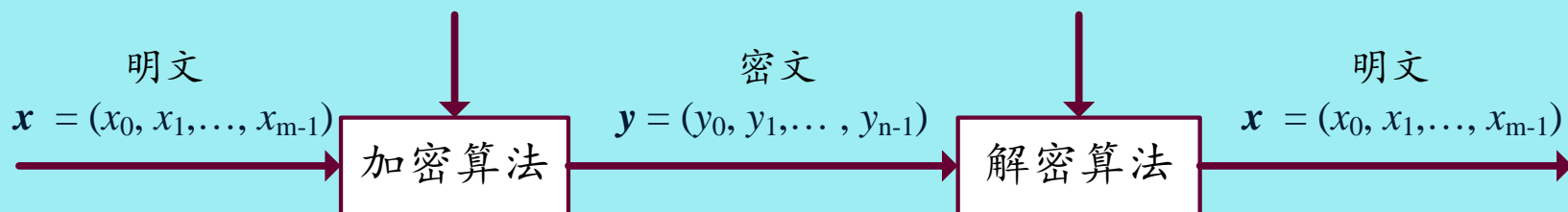


图1：分组算法

分组密码

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。

分组密码

- 加密和解密运算简单，易于软件和硬件高速实现。
- 数据扩展。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。

实现上述几点要求并不容易。首先，代换网络的复杂性随分组长度呈指数增大，常常会使设计变得复杂而难以控制和实现；实际中常常将分成几个小段，分别设计各段的代换逻辑实现电路，采用并行操作达到总的分组长度足够大，这将在下面讨论。其次，为了便于实现，实际中常常将较简单易于实现的密码系统进行组合，构成较复杂的、密钥量较大的密码系统。

二、分组密码 DES

美国数据加密标准——DES (Data Encryption Standard)

1. 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。

分组密码

- 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。

分组密码

- 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。

分组密码

- 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算法

DES 算法：分组长度为64 bit (8 bytes)，密文分组长度也是64 bit。密钥长度为64 bit，有8 bit 奇偶校验，有效密钥长度为56 bit。算法主要包括：初始置换 IP 、16轮迭代的乘积变换、逆初始置换 IP^{-1} 以及16个子密钥产生器。

- **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 $x_8, x_{16}, x_{32}, x_{64}$ 变成 IP 输出的一个字节。

分组密码

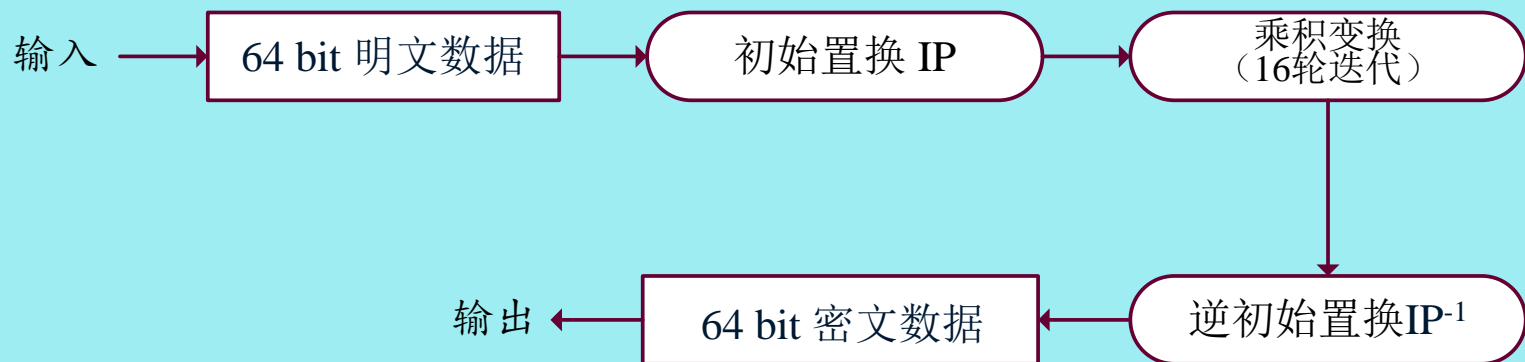


图2：标准数据加密算法-DES算法框图

分组密码

- **乘积变换**：它是DES算法的核心部分。

将经过 *IP* 置换后的数据分成 32bit 的左右两组，在迭代过程中彼此左右交换位置。每次迭代时只对右边的 32 bit 进行一系列的加密变换，在此轮迭代即将结束时，把左边的 32 bit 与右边得到的 32bit 逐位模 2 相加，作为下一轮迭代时右边的段，并将原来右边未经变换的段直接送到左边的寄存器中作为下一轮迭代时左边的段。

在每一轮迭代时，右边的段要经过选择**扩展运算 E** 、**密钥加密运算**、**选择压缩运算 S** 、**置换运算 P** 和**左右混合运算**。

分组密码

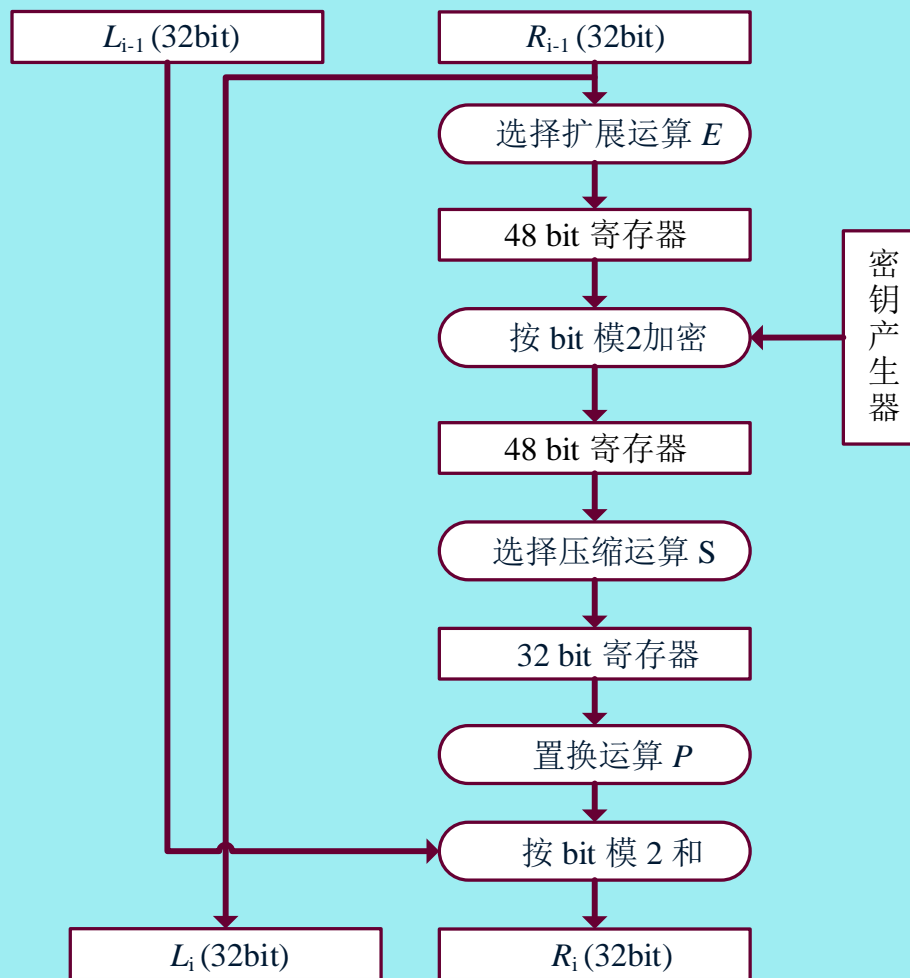


图3: DES算法 13

分组密码

- **选择扩展运算 E** ：将 32 bit 输入 R_{i-1} 扩展成 48 bit 输出。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 $1 \pmod{4}$ 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。
- **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，每组为 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。置换 P 输出的 32 bit 数据与左边 32 bit 即 R_{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。

分组密码

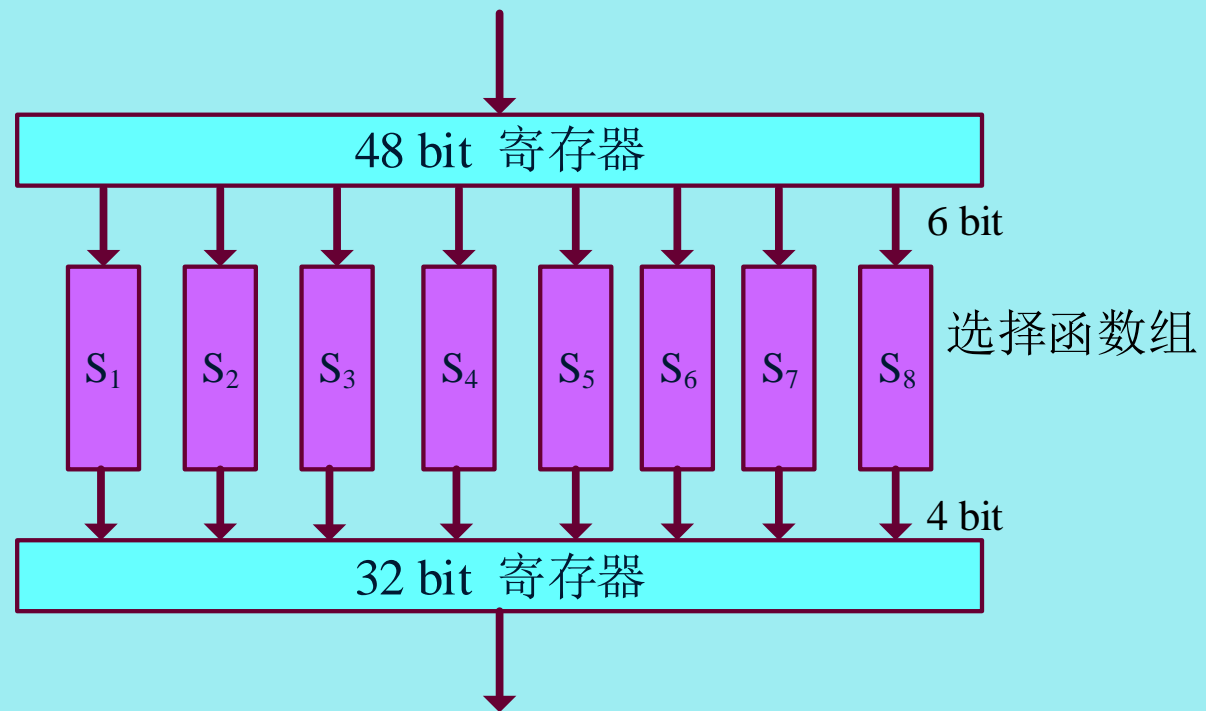


图4：选择压缩运算 S

分组密码

- **子密钥产生器**：将64 bit 初始密钥经过**置换选择 PC1**、**循环移位置换**、**置换选择 PC2** 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 **PC1**。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，移位次数在表中给出。每次移位后，将 **C** 和 **D** 寄存器原存数送给置换选择 **PC2**。置换选择 **PC2** 将 **C** 中第9、18、22、25位和 **D** 中第7、9、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 。

分组密码

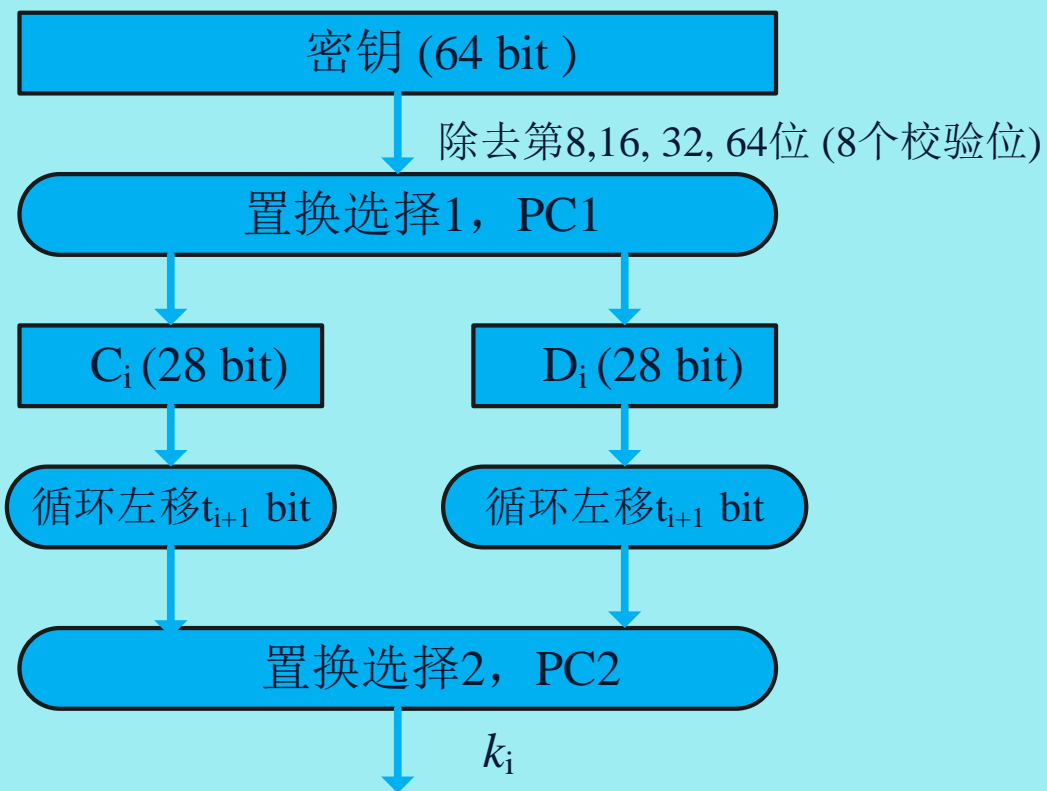


图5: 子密钥产生器框图

分组密码

表1：移位次数表

第 i 次迭代	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
循环左移次数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

至此，我们已将 DES 算法的基本构成作了介绍，加密过程可归结如下：

令 IP 表示初始置换， KS 表示密钥运算， I 为迭代次数变量， KEY 为 64 bit 密钥， f 为加密函数， \oplus 表示逐位模 2 求和。

分组密码

- **加密过程:** 式 (1) 和式 (2) 的运算进行16次后就得到密文组。

$$L_0 R_0 \leftarrow IP \text{ (} \langle 64 \text{ bit 输入码} \rangle \text{)}$$

$$L_i \leftarrow R_{i-1} \quad i = 1, \dots, 16 \quad (1)$$

$$R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, k_i) \quad i = 1, \dots, 16 \quad (2)$$

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

- **解密过程:** DES 的加密运算是可逆的, 其解密过程可类似地进行。

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

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

$$L_{i-1} \leftarrow R_i \oplus f(R_{i-1}, k_i) \quad i = 16, \dots, 1 \quad (4)$$

$$\langle 64 \text{ bit 明文} \rangle \leftarrow IP^{-1}(R_0L_0)$$

分组密码

DES 的安全性完全依赖于所用的密钥。从DES 诞生起，对它的安全性就有激烈的争论，一直延续到现在。

- **互补性:** DES 算法具有下述性质。若明文组 x 逐位取补得，密钥 k 逐位取补得，且 $y = DES_k(x)$ ，则有 $\tilde{y} = DES_{\tilde{k}}(\tilde{x})$ ，其中 \tilde{y} 是 y 的逐位取补。称这种特性为算法上的互补性。这种互补性会使 DES 在选择明文破译下所需的工作量减半。
- **弱密钥和半弱密钥:** DES 算法在每次迭代时都有一个子密钥供加密用。如果给定初始密钥 k ，各轮的子密钥都相同，即有 $k_1 = k_2 = \dots = k_{16}$ ，就称给定密钥 k 为弱密钥 (Weak key)。

分组密码

若 k 为弱密钥，则有

$$DES_k(DES_k(x)) = x$$

$$DES_k^{-1}(DES_k^{-1}(x)) = x$$

即以 k 对 x 加密两次或解密两次都可恢复出明文。其加密运算和解密运算没有区别。而对一般密钥只满足

$$DES_k^{-1}(DES_k(x)) = DES_k(DES_k^{-1}(x)) = x$$

弱密钥下使 DES 在选择明文攻击下的搜索量减半。如果随机地选择密钥，则在总数 2^{56} 个密钥中，弱密钥所占比例极小，而且稍加注意就不难避开。因此，弱密钥的存在不会危及 DES 的安全性。

分组密码

- **密文与明文、密文与密钥的相关性：** Meyer [1978] 详细研究了 DES 的输入明文与密文及密钥与密文之间的相关性：表明每个密文比特都是所有明文比特和所有密钥比特的复合函数，并且指出达到这一要求所需的迭代次数至少为5。Konheim [1981] 用 χ^2 检验证明，迭代8次后输出和输入就可认为是不相关的了。
- **S盒设计：** DES靠S 盒实现非线性变换。
- **密钥搜索机：** 对DES 安全性批评意见中，较为一致的看法是 DES 的密钥短了些。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。

分组密码

DES 的密钥量为 $2^{56} = 7.2 \times 10^{16} = 72\ 057\ 594\ 037\ 927\ 936 \approx 10^{17}$ 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 $100\mu\text{s}$ (一般微处理器能实现)，则搜索整个密钥空间需时为 7.2×10^{15} 秒，近似为 2.28×10^8 年。若以最快的 LSI 器件，DES 加密操作时间可降到 $5\mu\text{s}$ ，也要 1.1×10^4 年才能穷尽密钥。但是由于差分 and 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。

分组密码

RSA 数据安全公司提供10000 美元奖金。现已被 DESCHALL 小组经过近四个月的努力，通过Internet 搜索了 3×10^{16} 个密钥，找出了 DES 的密钥，恢复出明文。1998年5月美国EFF (Electronic Frontier Foundation) 宣布，他们以一台价值 20 万美元的计算机改装成的专用解密机，用56小时破译了56 bits 密钥 DES 。

Crypto Bytes, Vol.3, No.2, 1991, DES 搜索机成本

\$1	万	2.5 天
\$10	万	6小时
\$100	万	35分钟 (采用57000个ASIC, 50000万个密钥/秒,片)
\$1000	万	3.5分钟

分组密码

- **DES 实现：**自DES 正式成为美国标准以来，已有许多公司设计并推出了实现DES 算法的产品。有的设计专用 LSI 器件或芯片，有的用现成的微处理器实现。有的只限于实现DES 算法，有的则可运行各种工作模式。
- **DES 变型：**为提高安全性和适应不同情况的需求而设计了多种应用DES 的方式。提出了多种DES 的修正形式。如独立子密钥方式、DESX、CRYPT (3)、S 盒可变的DES、RDES、snDESi 、xDESi、GDES 等。

三、分组密码运行模式

分组密码每次加密的明文数据量是固定的分组长度，而实用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。因此需要做一些变通，灵活地运用分组密码。另一方面，即使有了安全的分组密码算法，也需要采用适当的工作模式来隐蔽明文的统计特性、数据的格式等，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。所采用的工作模式应当力求简单、有效和易于实现。本节将以 DES 为例介绍分组密码的实用工作模式。美国NSB在[FIPS PUB 74和81]中规定了DES 的四种基本工作模式：电子码本 (ECB)，密码反馈链接 (CBC)，密码反馈 (CFB)，输出反馈 (OFB)。ANSI, ISO 和 ISO/IEC 也规定了类似的工作模式 [ANSI X3.106; ISO 8732; ISO/IEC 10116]。这四种模式也可用于其它分组密码。

分组密码

1. 电码本 ECB (Electronic Code Book) 模式

它直接利用 DES 算法分别对各64 bits 数据组加密。在给定密钥 k 时，各明文组 x^i 分别对应于不同的密文组 $y_i = DES_k(x^i)$ ；在给定密钥下， x 有 2^{64} 种可能取值， y 也有 2^{64} 种可能取值，各 (x, y) 对彼此独立，构成一个巨大的单表代换密码，因而称为电码本模式。

缺点：在给定的密钥下同一明文组总产生同样的密文组，这会暴露明文数据的格式和统计特征。明文数据都有固定的格式，需要以协议的形式定义，有大量重复和较长的零串，重要的数据常常在同一位置上出现。其最薄弱的环节是消息起始部分，其中包括格式化报头，内含通信地址、作业号、发报时间等信息。所有密码体制中都需要认真对待这类格式的死框框。在ECB 模式，所有这些特征都将被反映到密文中，使密码分析者可以对其进行统计分析、重传和代换攻击。

分组密码

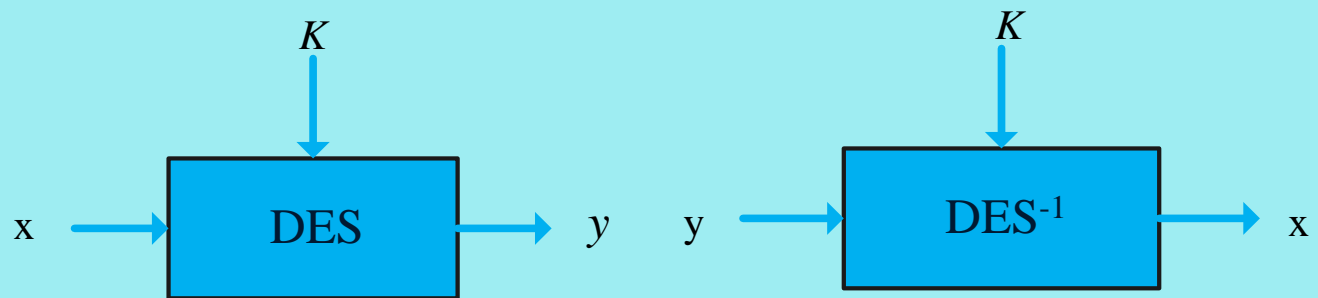


图6: ECB 模式

分组密码

2. 密码分组链接 CBC (Cipher Block Chaining) 模式

在 CBC 模式下, 每个明文组 x_i 加密之前, 先与反馈至输入端的前一组密文 y_{i-1} 按位模 2 求和后, 再送至 DES 加密。所有运算均按 64 bit 并行实施。在 CBC 模式下有 $y = DES_k(x_i \oplus y_{i-1})$ 。可知, 各密文组 y_i 不仅与当前明文组 x_i 有关, 而且通过反馈作用还与以前的明文组 x_1, x_2, \dots, x_{i-1} , 有关。密文经由存储器实现前馈, 使解密输出

$$\begin{aligned}x_i &= DES^{-1}_k(y_i) \oplus y_{i-1} \\&= DES^{-1}_k(DES_k(x_i \oplus y_{i-1})) \oplus y_{i-1} \\&= x_i \oplus y_{i-1} \oplus y_{i-1}\end{aligned}$$

分组密码

初始矢量IV (Initial Vector): 第一组明文 x_i 加密时尚无反馈密文，为此需要预先置入一个。收发双方必须选用同一IV。此时有

$$y_i = DES_k (x_i + IV)$$

$$x_i = DES_k^{-1} (y_i) + IV$$

通信中一般将 IV 作为一个秘密参数，可以采用 ECB 模式用同一密钥且加密后送给收方。实际上，IV 的完整性要比其保密性更为重要。

在CBC模式下，最好是每发一个消息，都改变IV，比如将其值加一。

分组密码

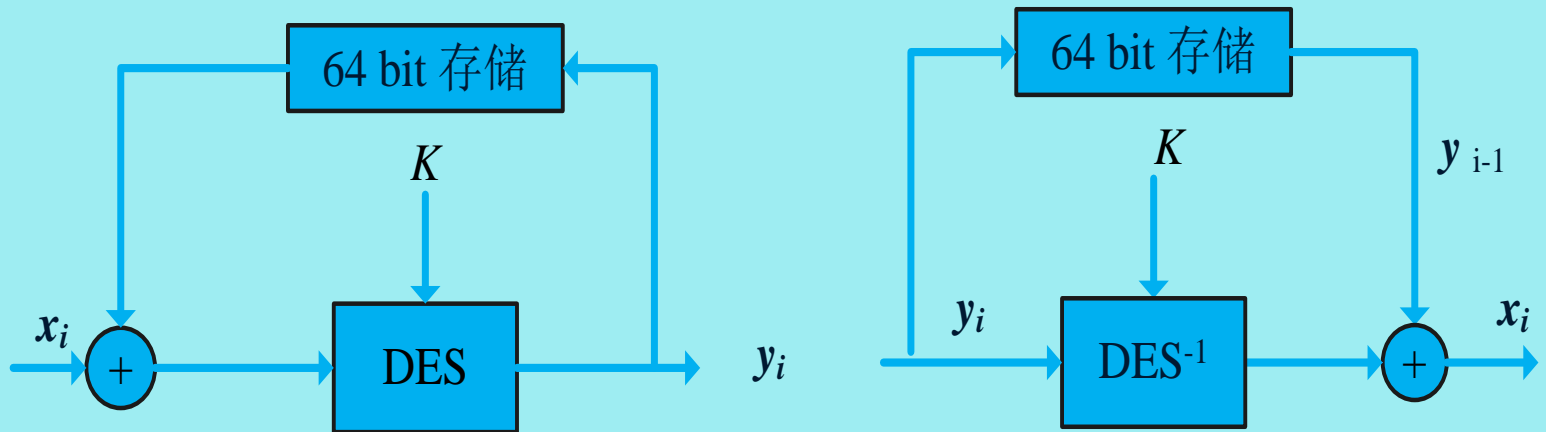


图7: CBC 模式

分组密码

填充 (Padding): 给定加密消息的长度是随机的, 按64 bit 分组时, 最后一组消息长度可能不足64 bit。可以填充一些数字, 如0, 凑够64 bit。当然, 用随机选取的数字填充更安全些。接收者如何知道哪些数字是填充的无用数字呢? 这需要加上指示信息, 通常用最后八位 (为1字节) 作为**填充指示符**, 简记作 *PI*。它所表示的十进制数字就是填充占有的字节数。数据尾部、填充字符和填充指示符一起作为一组进行加密。这种方法会有些数据扩展。当不希望有扩展时, 可采用其它措施。若消息分组最后一段只有 k bit, 可将前一组加密结果 y_{n-1} 中最左边 k bit 作为密钥与其逐位模2相加后作为密文送出。这种做法使最后 k bit 的安全性要降低, 但一般消息的最后部分多为检验位, 因此问题不大。

分组密码

3. k -比特密码反馈 CFB (Cipher Feedback) 模式

若待加密消息必须按字符 (如电传电报) 或按比特处理时, 可采用 CFB 模式。 x_i 和 y_i 都为 k -bit 段, $L = \lceil 64/k \rceil$, 即 $kL \geq 64$ bit。 x_i 是取自寄存器右边的 64 bit。 k 可取 1 到 64, 一般为 8 的倍数, 常用 $k = 8$ 。CFB 实际上是将 DES 作为一个密钥流产生器, 在 k bit 密文反馈下, 每次输出 k -bits 密钥, 对输入的明文 k -bits 进行并行加密。这就是一种自同步流密码 (SSSC—Self-Synchronizing Stream Cipher)。当 $k = 1$ 时就退化为前面讨论的流密码了。CFB 与 CBC 的区别是反馈的密文不再是 64 bit, 而是长度为 k , 且不是直接与明文相加, 而是反馈至密钥产生器。

分组密码

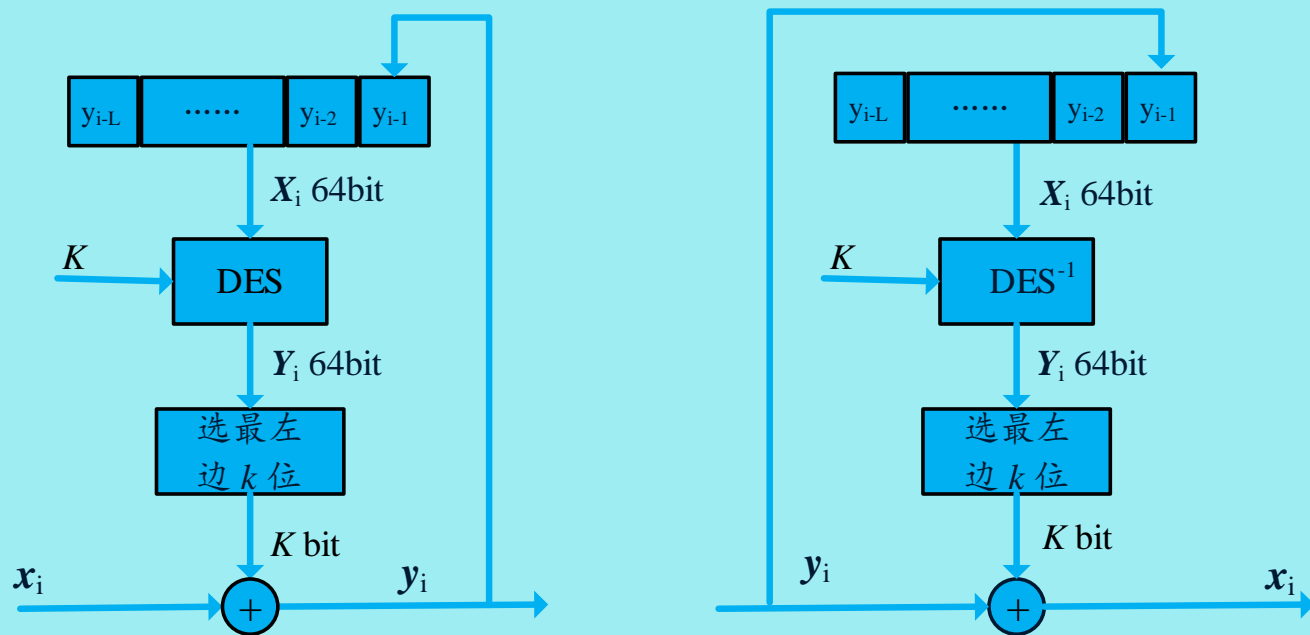


图8: CFB模式

分组密码

CFB 优点：它特别适于用户数据格式的需要。在密码体制设计中，应尽量避免更改现有系统的数据格式和一些规定，这是一个重要的设计原则。CFB 和CBC 一样，由于反馈的作用而能隐蔽明文数据图样，也能检测出对手对于密文的篡改。

CFB 缺点：类似于CBC 之处，也有不同之处。首先，它对信道错误较敏感，且会造成错误传播。所幸的是，这种模式多用于数据网中较低层次，其数据速率都不太高。最后，CFB 也需要一个初始矢量，并要和密钥同时进行更换，但由于其初始矢量在起作用过程中要经过 DES 加密，故可以明文形式传给收方。

分组密码

4. 输出反馈 OFB (Output Feedback) 模式

这种模式将DES 作为一个密钥流产生器，其输出的 k -bit 密钥直接反馈至DES 的输入端，同时这 k -bit 密钥和输入的 k -bit 明文段进行对应位模2相加。这一模式的引入是为了克服CBC 和CFB 的错误传播所带来的问题。由于语言或图像编码信号的冗余度较大，可容忍传输和存储过程中产生的少量错误，但CBC 或CFB 中错误传播的效应，可能使偶然出现的孤立错误扩大化而造成难以容忍的噪声。

这种密钥反馈流加密方式虽然克服了错误传播，但同时也引入了流密码的缺点。对于密文被篡改难以进行检测，但由于OFB 多在同步信道中运行，对手难以知道消息的起止点而使这类主动攻击不易奏效。

OFB 模式不具有自同步能力，要求系统要保持严格的同步，否则难以解密。重新同步时需要新的 IV，可以用明文形式传送。在实用中还要防止密钥流重复使用，以保证系统的安全。

分组密码

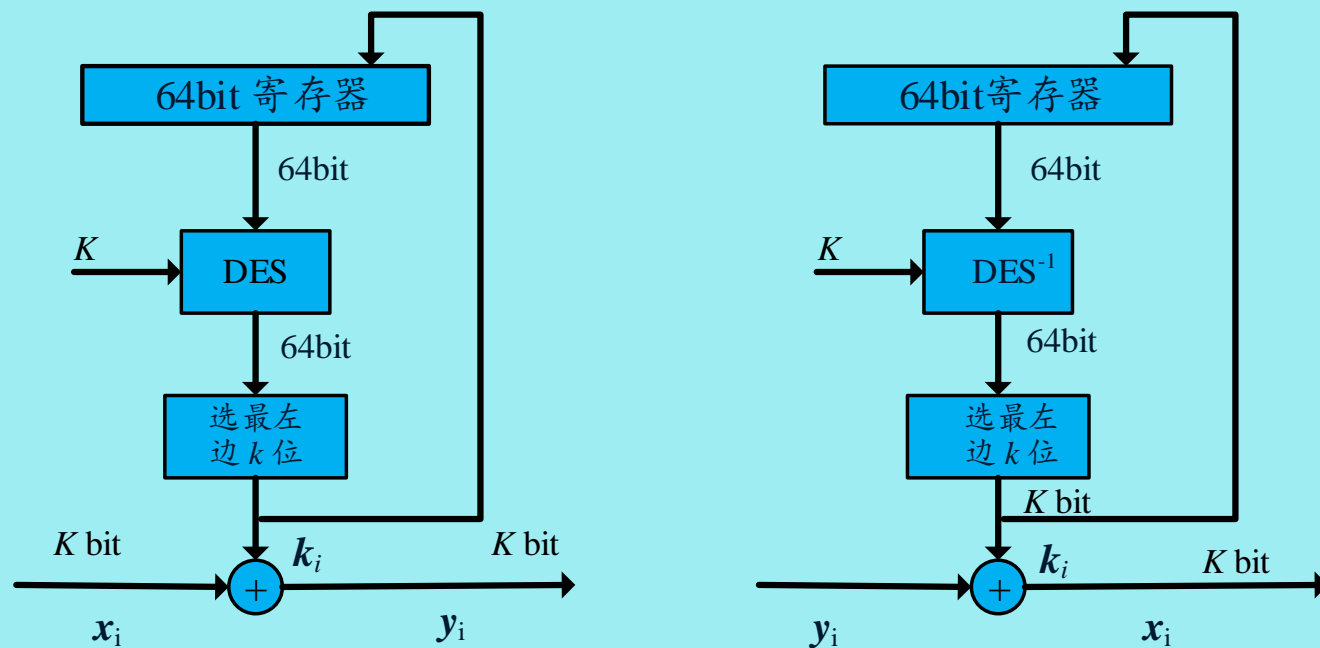


图9: OFB 模式

分组密码

比较和选用： 上述四种基本模式各有其特点和用途。

- ECB 模式，简单、高速，但最弱、易受重发攻击，一般不推荐。
- CBC, CFB, OFB 的选择取决于实用特殊考虑。
- CBC 适用于文件加密，但较ECB 慢，且需要另加移存器和组的异或运算。但安全性加强。当有少量错误时，也不会造成同步错误。软件加密最好选用此种方式。
- OFB 和 CFB 较CBC 慢许多。每次迭代只有少数 bit (如一字节) 完成加密。若可以容忍少量错误扩展，则可换来恢复同步能力，此时用CFB。若不容许少量错误扩展，则选用 OFB。

在字符为单元的流密码中多选CFB 模式，如终端和主机间通信。而 OFB 用于高速同步系统，不容忍差错传播。

分组密码——AES

- 1997年1月，美国NIST 向全世界密码学界发出征集21世纪高级加密标准 (AES——Advanced Encryption Standard) 算法的公告，并成立了AES 标准工作研究室，1997年4月15日的例会制定了对AES 的评估标准。
- **AES 的标准提纲:** (1) AES 是公开的；(2) AES 为单钥体制分组密码；(3) AES 的密钥长度可变，可按需要增大；(4) AES 适于用软件和硬件实现；(5) AES 可以自由地使用，或按符合美国国家标准 (ANST) 策略的条件使用；(6) 满足以上要求的AES 算法，需按下述条件判断优劣：a.安全性，b.计算效率，c.内存要求，d.使用简便性，e.灵活性。

四、分组密码AES

- 1998 年 4 月 15 日全面征集 AES 算法的工作结束。1998 年 8 月 20 日举行了首届 AES 讨论会，对涉及 14 个国家的密码学家所提出的候选 AES 算法进行了评估和测试，初选并公布了 15 个被选方案，供大家公开讨论。15 个候选算法有：CAST-256，RC-6，CRYPTON-128，DEAL-128，FROG，简易布丁密码，LOKI-97，MAGENTA，MARS，Vaudenay，RIJNDAEL，SAFER+，SERPENT，E-2，TWOFISH。这些算法设计思想新颖，技术水平先进，算法的强度都超过 3-DES，实现速度快于 3-DES。
- 1999 年 8 月 9 日 NIST 宣布第二轮筛选出的 5 个候选算法为：**MARS** (C. Burwick 等，IBM)，**RC-6** (R. Rivest 等，RSA Lab.)，**RIJNDEAL** (Rijndael、Daemen 等，比)，**SERPENT** (R. Anderson 等，英、以、挪威)，**TWOFISH** (B. Schiener)。
- 2000 年 10 月 2 日 NIST 宣布 Rijndael 为 AES 的算法。

AES 算法

Rijndael 是一个分组密码算法，其分组长度和密钥长度相互独立，都可以改变。

设计原则：

- 能密码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

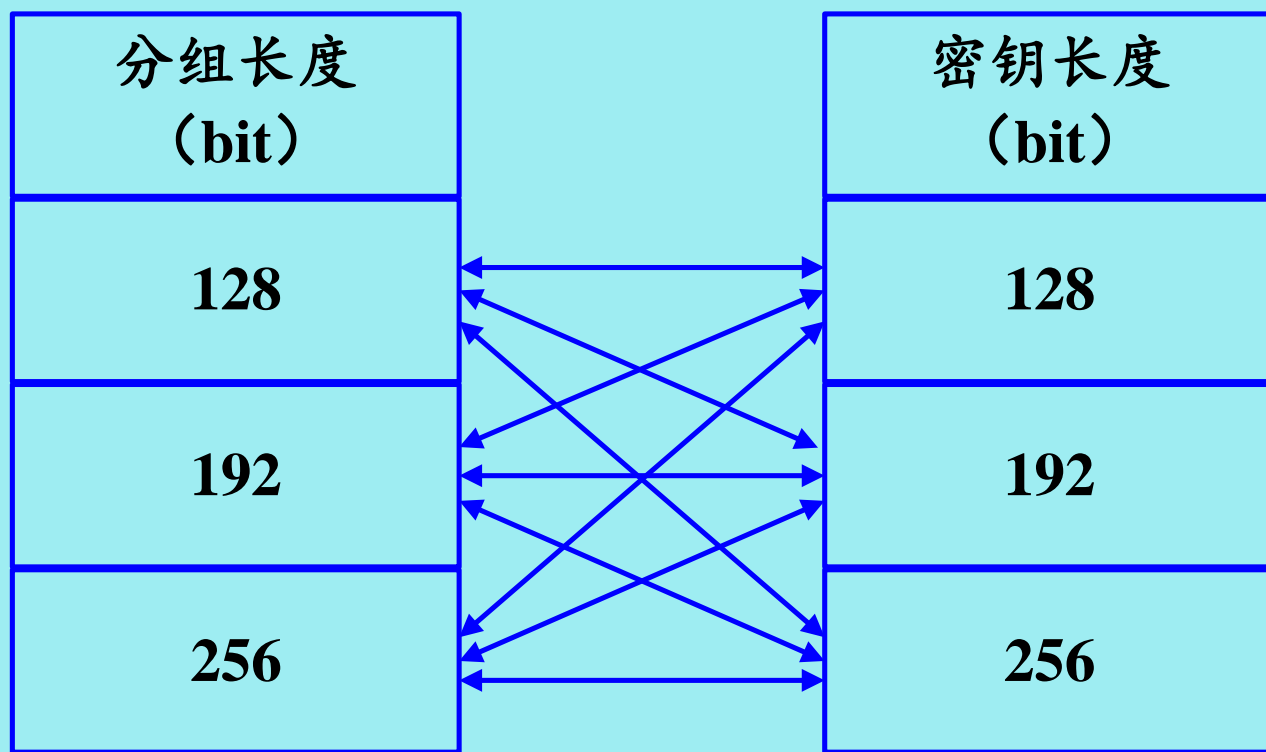


图 10：分组长度和密钥长度的组合

明文分组和密钥的组织排列方式

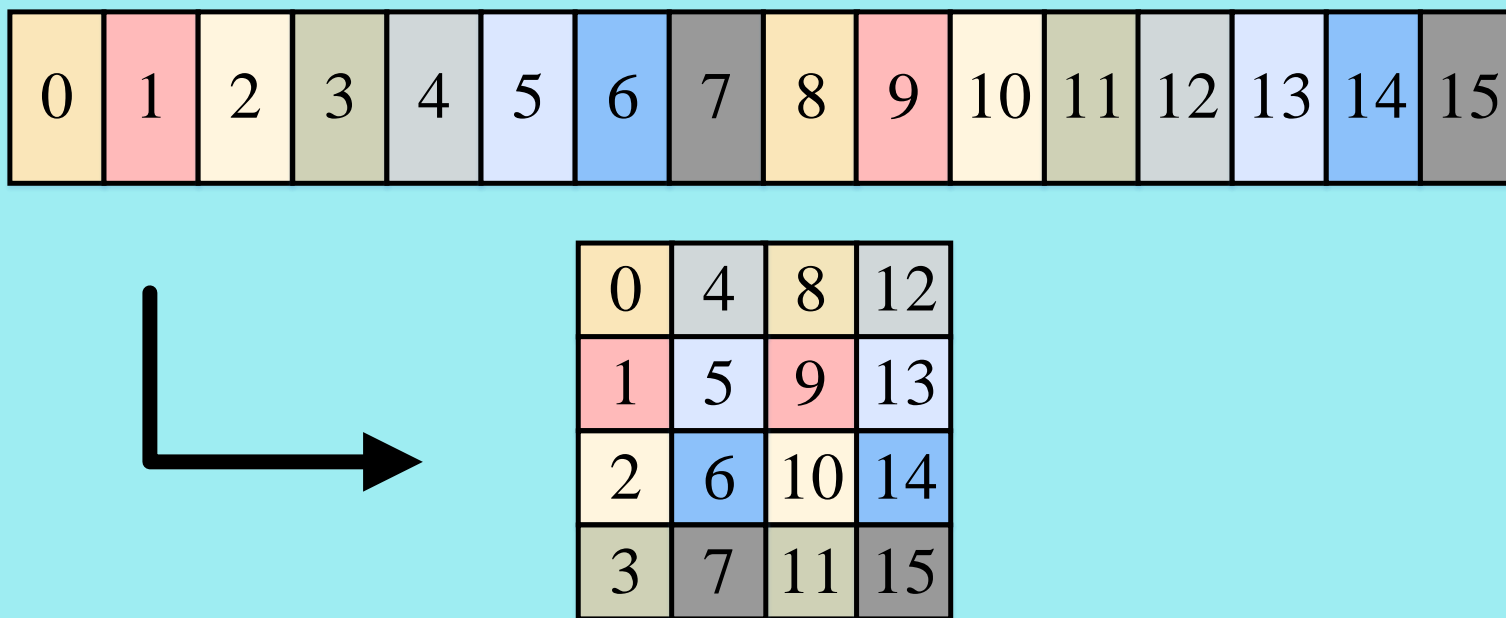



图11：当明文(密钥)分组为128 bits 时，组成的阵列



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

0	4	8	12	16	20
1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

图12：明文分组（或密钥）为128bits、192bits、256bits 组成的阵列

一些相关术语的定义和表示

- **状态 (state):** 密码运算中间结果称为状态
- **state的表示:** 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 bits

- **密钥的表示:** Cipher Key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 bits

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$N_b = 6$
 Block Length = 192 bits

$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$

$N_k = 4$
 Key Length = 128 bits

图13: 当 $N_b=6$ 时的状态和 $N_k=4$ 时的密钥布局

Rijndael 密码构成

Rijndael 密码由以下三个部分组成：

- 一个初始圈密钥相加；
- $Rnd - 1$ 圈；
- 一个结尾圈；

Plaintext

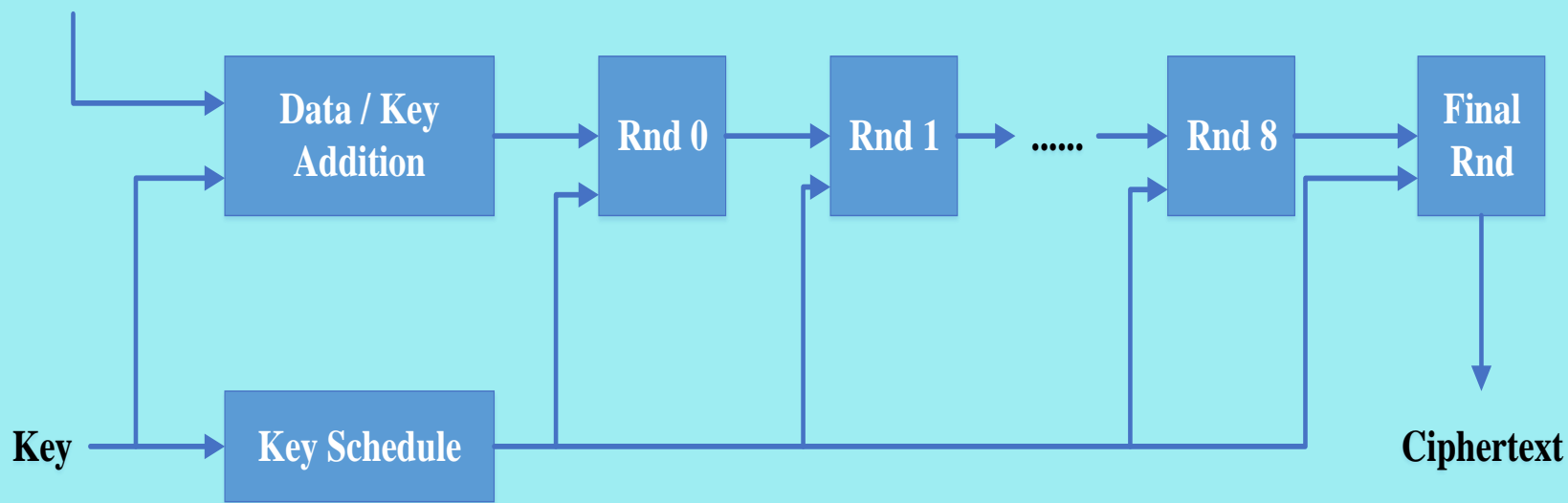


图14：分组长度和密钥长度都为 128 bits 时的加密算法框图

表2：圈数 (Round) 的不同取值

圈数(Round)	Block Length = 128	Block Length = 192	Block Length = 256
Key Length = 128	10	12	14
Key Length = 192	12	12	14
Key Length = 256	14	14	14

Rijndael Round 的构成

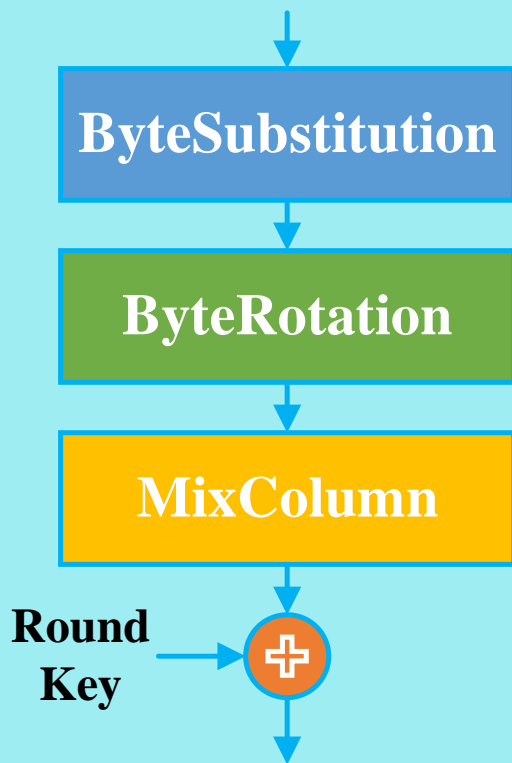


图15：一般的圈代换

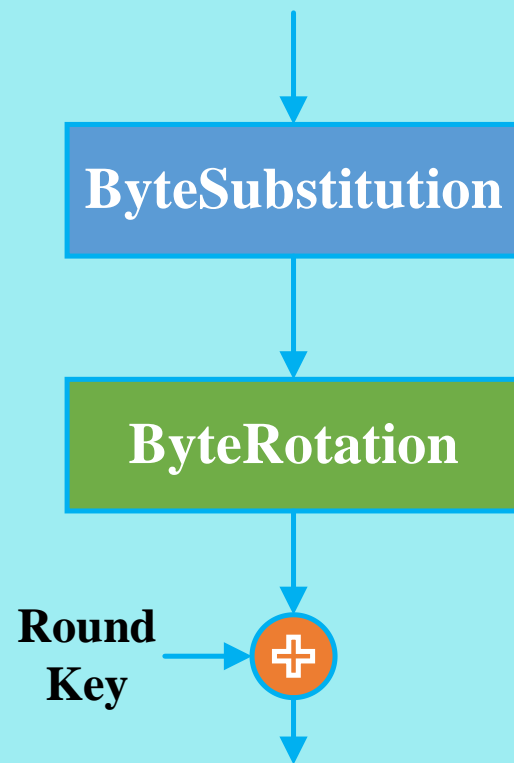


图16：最后一轮圈代换

伪代码表示 Rijndael 圈变换

一般的圈代换

```
Round (State, Roundkey)
{
    ByteSubstitution;
    ByteRotation;
    MixColumn;
    AddRoundkey;
}
```


结尾圈代换

```
FinalRound (State, Roundkey)
{
    ByteSubstitution;
    ByteRotation;
    AddRoundkey;
}
```


ByteSubstitution (字节替代)

ByteSubstitution 是一个非线性的字节替代，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：



$$\begin{bmatrix} y_7 \\ y_6 \\ y_5 \\ y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$

$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$B_{0,3}$
$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$
$B_{2,0}$	$B_{2,1}$	$B_{2,2}$	$B_{2,3}$
$B_{3,0}$	$B_{3,1}$	$B_{3,2}$	$B_{3,3}$

取逆仿射变换

图17: ByteSubstitution 用一个256字节的表实现

ByteRotation (字节移位)

在 ByteRotation 变换中，状态数组的后3行循环移位不同的偏移量。第1行循环移位C1字节，第2行循环移位C2字节，第3行循环移位C3字节。

偏移量C1、C2、C3与分组长度 N_b 有关，如下表所示：

表3：不同情况下 N_b 的取值

N_b	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

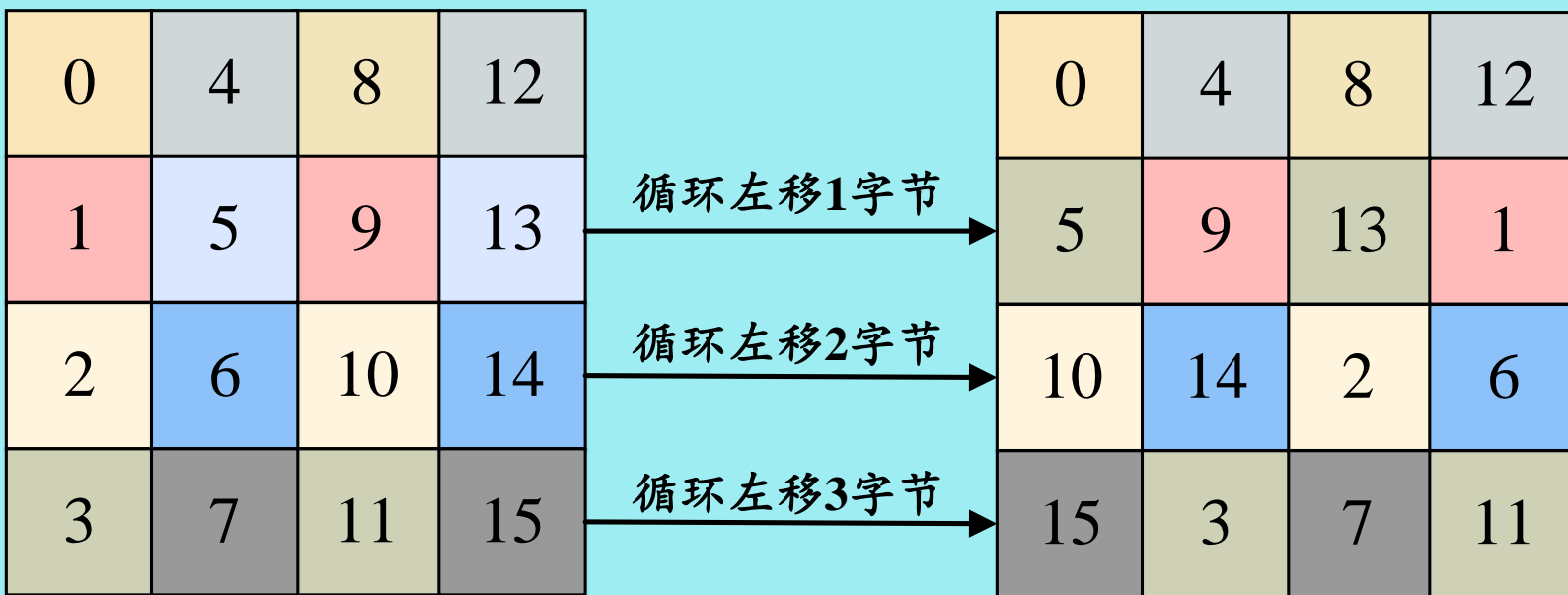


图18: ByteRotation 示意图

MixColumn (列混合)

将状态的列看作是有限域 $GF(2^8)$ 上的多项式 $a(x)$ ，与多项式 $c(x) = 03 * x^3 + 01 * x^2 + 01 * x + 02$ (模 $x^4 + 1$)。

令 $b(x) = c(x) * a(x)$ ，写成矩阵形式为：

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

这一运算作用在每一列上

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$

$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$B_{0,3}$
$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$
$B_{2,0}$	$B_{2,1}$	$B_{2,2}$	$B_{2,3}$
$B_{3,0}$	$B_{3,1}$	$B_{3,2}$	$B_{3,3}$

$*C(x)$

图19: MixColumn 示意图

AddRoundKey (圈密钥加)

将圈密钥与状态按比特异或。圈密钥是通过 Key Schedule 过程从密码密钥中得到的，圈密钥长度等于分组长度。

$$A_{3,3} + K_{3,3} = B_{3,3} \pmod{2}$$

$A_{0,0}$	$A_{0,1}$	$A_{0,2}$	$A_{0,3}$
$A_{1,0}$	$A_{1,1}$	$A_{1,2}$	$A_{1,3}$
$A_{2,0}$	$A_{2,1}$	$A_{2,2}$	$A_{2,3}$
$A_{3,0}$	$A_{3,1}$	$A_{3,2}$	$A_{3,3}$



$K_{0,0}$	$K_{0,1}$	$K_{0,2}$	$K_{0,3}$
$K_{1,0}$	$K_{1,1}$	$K_{1,2}$	$K_{1,3}$
$K_{2,0}$	$K_{2,1}$	$K_{2,2}$	$K_{2,3}$
$K_{3,0}$	$K_{3,1}$	$K_{3,2}$	$K_{3,3}$



$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$B_{0,3}$
$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$
$B_{2,0}$	$B_{2,1}$	$B_{2,2}$	$B_{2,3}$
$B_{3,0}$	$B_{3,1}$	$B_{3,2}$	$B_{3,3}$

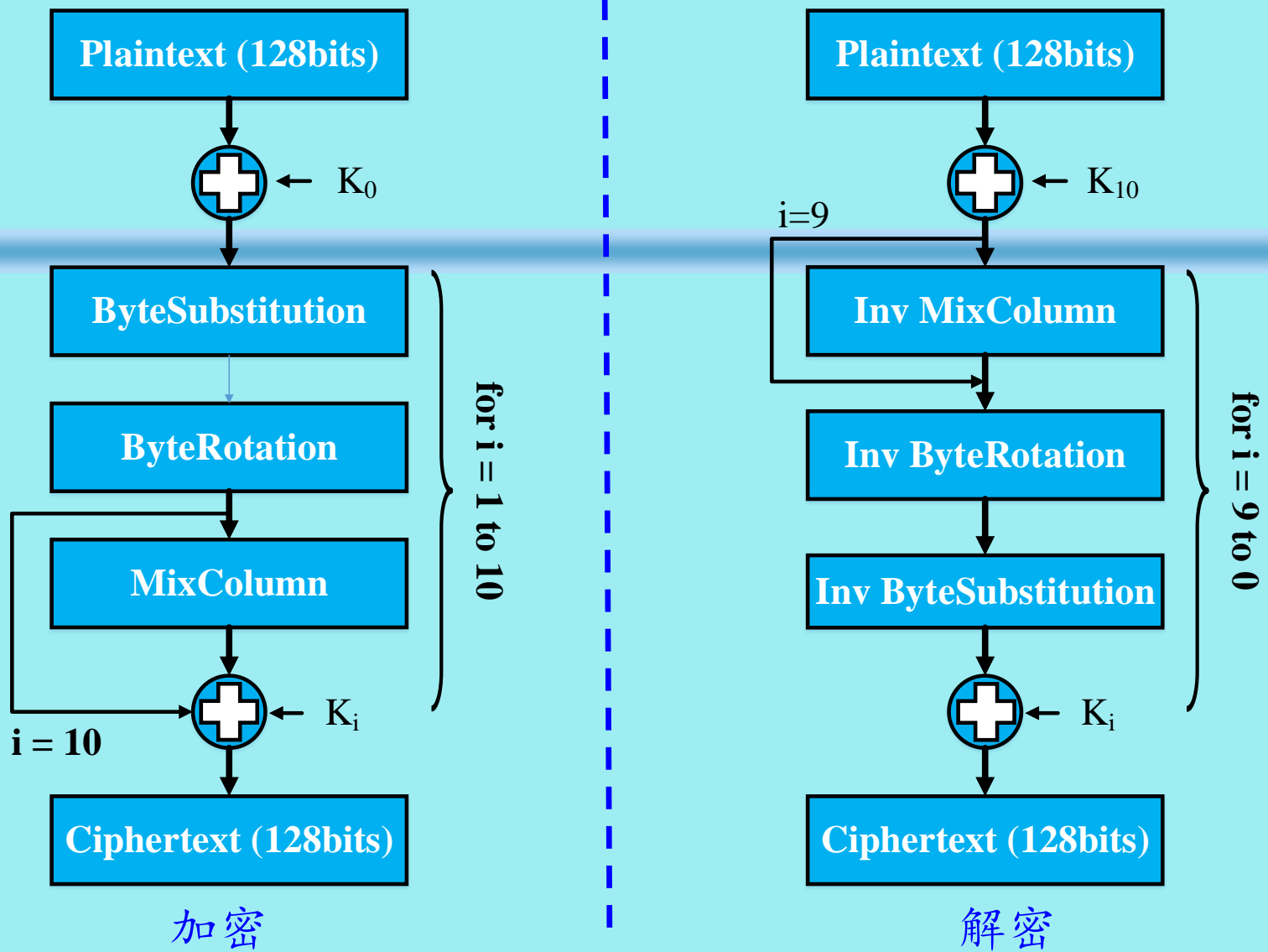


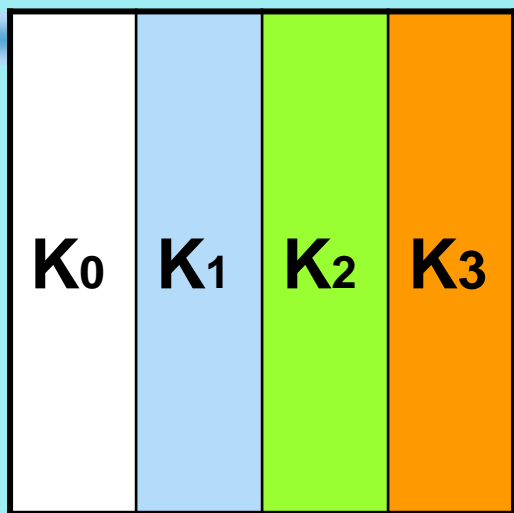
图20: Rijndael加密及解密的标准结构
Block , Key Length = 128 bits

AES 算法的密钥调度

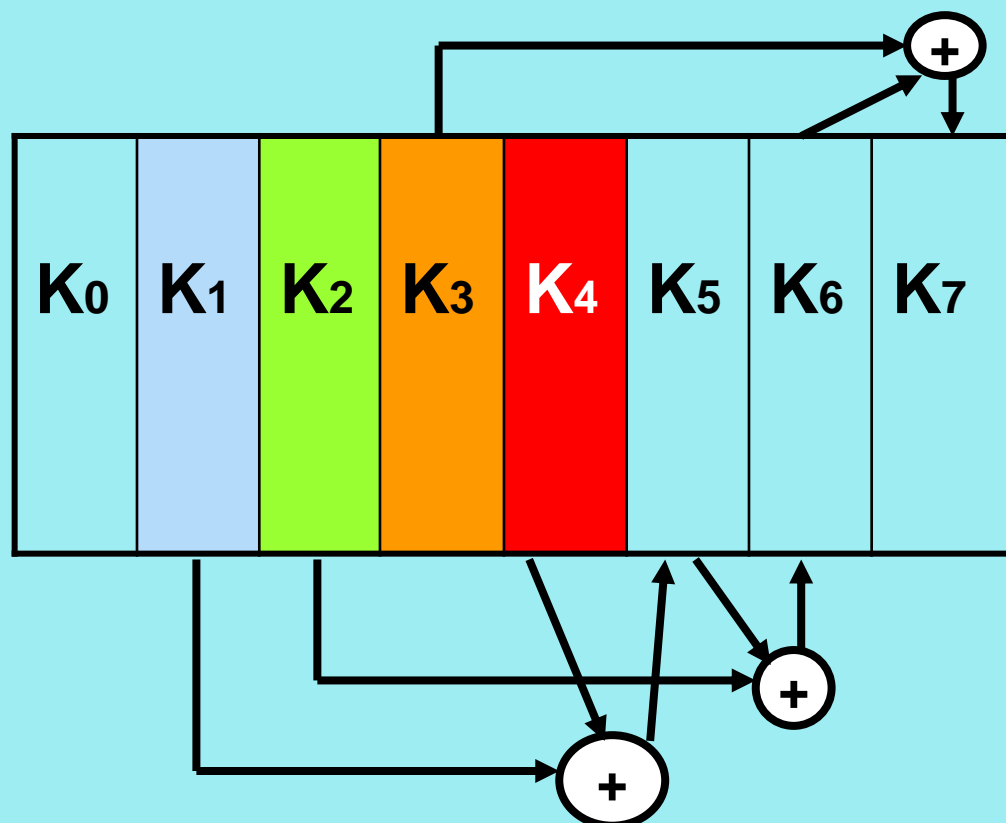
密钥调度包括两个部分：密钥扩展和圈密钥选取。

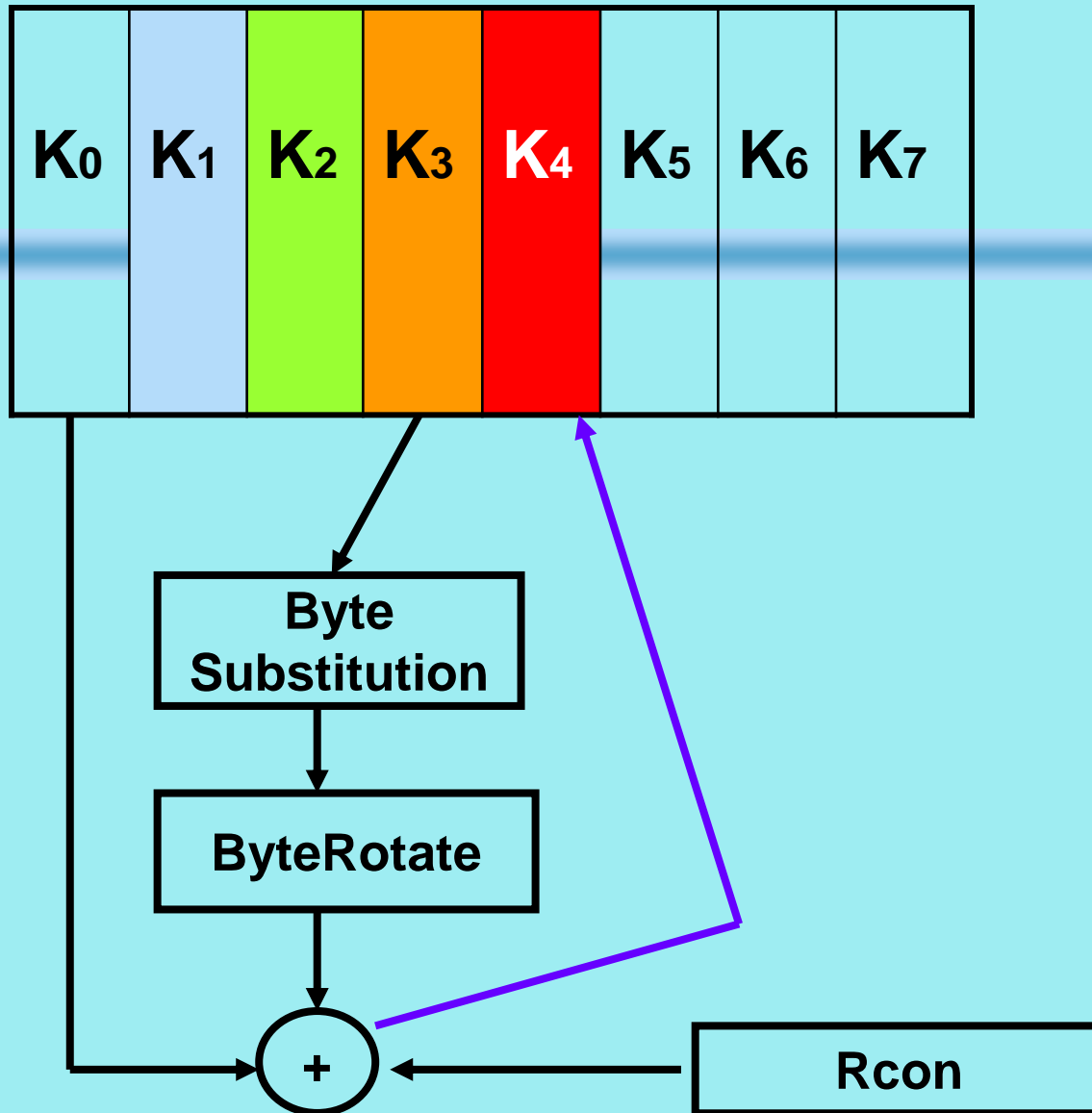
- 密钥bit的总数 = 分组长度 \times (圈数Round+1) 例如当分组长度为128 bits 和圈数 Round 为10时，圈密钥长度为 $128 \times (10+1) = 1408$ bits。
- 将密码密钥扩展成一个扩展密钥。
- 从扩展密钥中取出圈密钥：第一个圈密钥由扩展密钥的第一个 N_b 个4字节字，第二个圈密钥由接下来的 N_b 个4字节字组成，以此类推

密钥扩展

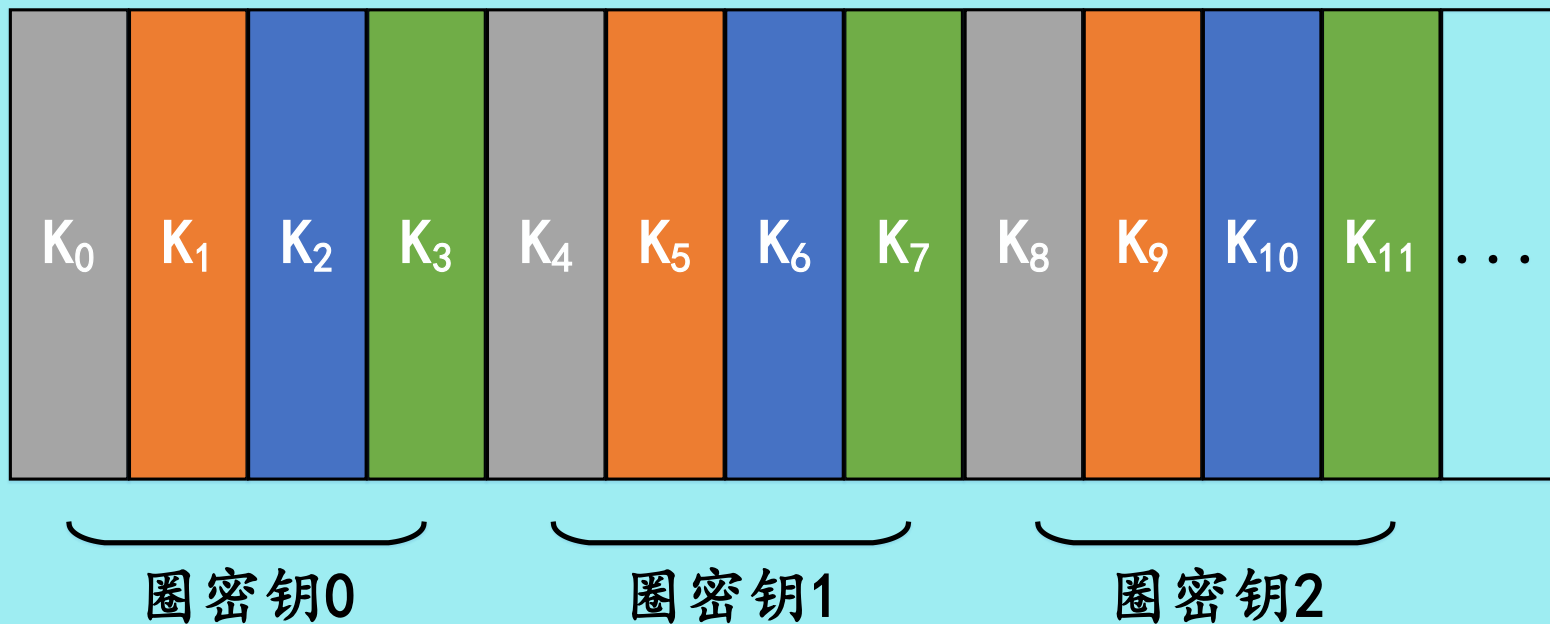


K _{0,0}	K _{0,1}	K _{0,2}	K _{0,3}
K _{1,0}	K _{1,1}	K _{1,2}	K _{1,3}
K _{2,0}	K _{2,1}	K _{2,2}	K _{2,3}
K _{3,0}	K _{3,1}	K _{3,2}	K _{3,3}





圈密钥选取



AES 算法的设计原理

- $GF(2^8)$ 中乘法使用的多项式是 8 次不可约多项式列表中的第一个多项式。
- ByteSubstitution (称为S盒) 在设计时考虑到抵抗差分密码分析、线性密码分析的要求, 应满足以下条件:
 - 可逆性;
 - 输入比特的线性组合与输出比特的组合之间的最大非平凡相关性的极小化;
 - 异或差分表中最大非平凡值的极小化;
 - $GF(2^8)$ 中代数表示的复杂性
 - 描述的简单性。

AES 算法的设计原理

满足前3条准则的S盒的构造方法已被给出，AES 的作者从众多候选构造中选择将 x 映射到它的逆的S盒。该映射过于简单，为了抵抗插入攻击，加入仿射变换：

$$b(x) = (x_7 + x_6 + x_2 + x) + a(x)(x_7 + x_6 + x_5 + x_4 + 1) \bmod x_8 + 1$$

模数多项式 $x^8 + 1$ 选择为可能是最简单的模数多项式。可以找到其它的S盒满足以上准则。

AES 算法的设计原理

- MixColumn 变换符合以下准则

- 可逆性；
- GF(2) 中的线性性；
- 适当的扩散性能；
- 8 位处理器上实现速度快；
- 对称性；
- 描述的简单性。

选择模数多项式 $x^4 + 1$ 可满足准则 2、5、6。准则 1、3、4 要求系数的值要小，故选 00、01、02、03。

AES 算法的设计原理

- ByteRotation 符合以下准则：
 - 4 个位移量互不相同且 $C_0=0$;
 - 能抵抗差分截断攻击;
 - 能抗Square攻击;
 - 简单

从满足准则2和准则3出发，AES的作者选取了最简单的组合。

与其他算法的比较

- 与 DES 相比：
 - 无 DES 中的弱密钥和半弱密钥；
 - 紧凑的设计使得没有足够的空间来隐藏陷门。
- 与 IDEA 相比：无 IDEA 中的弱密钥。
- 具有扩展性：密钥长度可以扩展到为 32bits 倍数的任意密钥长度，分组长度可以扩展到为 64bits 倍数的任意分组长度。圈数和循环移位偏移量作为参数，要重新定义。



thank you