第二单元 分组密码

郑州大学网络空间安全学院

王志华

第2单元　分组密码

提纲

CONTENTS

2.1 分组密码概述

2.2 分组密码分析

2.3 DES

2.4 AES

2.5 SM4

2.6 祖冲之密码

2.7 其他分组密码

2.8 分组密码工作模式

第2单元　分组密码

提纲

CONTENTS

2.1 分组密码概述

2.2 分组密码分析

2.3 DES

2.4 AES

2.5 SM4

2.6 祖冲之密码

2.7 其他分组密码

2.8 分组密码工作模式

2.1 分组密码概述

2.1.1 分组密码的基本概念

1、分组密码需求

- 分组密码是密码系统设计的一个重要组成部分：
  - 可以用分组密码构造其他密码学原语：序列密码、消息认证码、哈希函数和密码协议等；
  - 分组密码是其他密码技术的核心组成部分：数据完整性机制、消息认证技术、实体认证协议和单钥数字签名体制等。

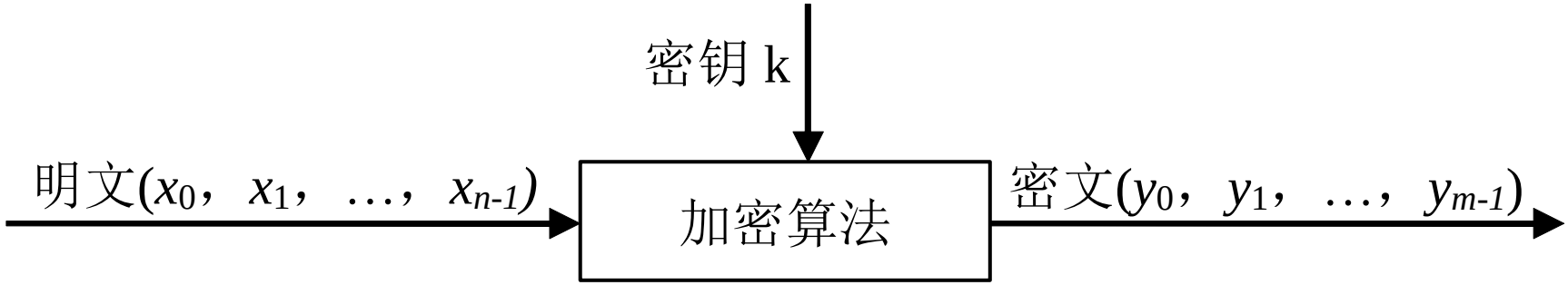
2、分组加密基本思想

- 分组加密的思想如下：

① 将二进制**明文消息序列**x_0，x_1，……，分成等长的**消息分组**(x_0，x_1，…，x_{n-1})，(x_n，x_{n+1}，…，x_{2n-1})，…，分组长度为n。

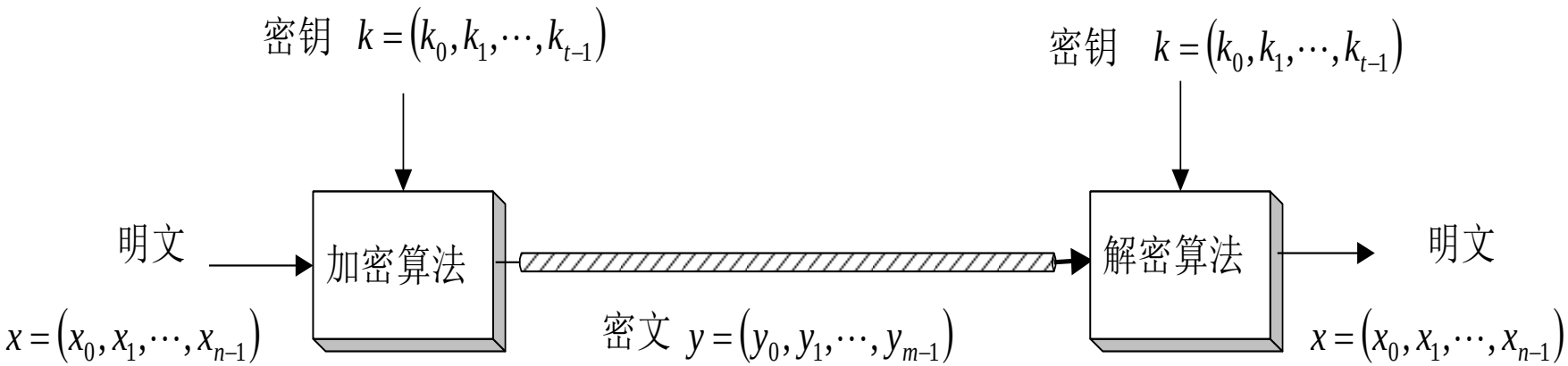
② 在**密钥k** (k_0，k_1，…，k_t)的作用下，按照**固定的加密算法**E_k**一组一组**地对**明文分组**进行加密。

③ 加密后输出等长的**密文分组**(y_0，y_1，…，y_{m-1})，(y_m，y_{m+1}，…，y_{2m-1})，……，密文分组长度为m。



3、分组密码基本定义

分组密码就是将明文消息编码表示后的数字序列 $x_0, x_1, \cdots, x_i, \cdots$ 划分成长为n的分组 $x=(x_0, x_1, \cdots, x_{n-1})$ ， 各组(长为n的矢量)分别在密钥 $k=(k_0, k_1, \cdots, k_{t-1})$ 控制下变换成等长的输出数字序列 $(y=(y_0, y_1, \cdots, y_{m-1})$(长为$m$的矢量)。其加密算法为$E$: $Vn \times Vt \rightarrow Vm$, Vn和Vm分别是n维和m 维矢量空间， Vt为t 维矢量空间。



本质：分组密码就是对输入字长为n的数字序列进行处理的代换密码。

3、分组密码形式化定义

一个分组密码是一个如下的映射：

$$F_2^{\,n} \times F_2^{\,t} \to F_2^{\,m}$$

记为 C=E(X，K)=E_K(X)，其中 X$\in F_2^{\,n}$，K$\in F_2^{\,t}$，C$\in F_2^{\,m}$，

$F_2^{\,n}$称为明文空间，$F_2^{\,m}$称为密文空间，$F_2^{\,t}$称为密钥空间。

n 为明文分组长度，t 为密钥长度，m 为密文分组长度。

4、分组密码特点

① 分组密码的**输出C**与多组长为n的明文序列（数字）有关。

② 在**相同密钥K**控制下，分组密码对长为n的输入明文分组所实施的**变换（映射）E是等同的**，使用**同一个变换规则**。

③ 在二元情况下，输入x和输出y均为二元数字序列，它们的每个分量xi，yi∈GF(2)：

- GF（2）是最简单的**有限域**，只有0，1二元及+（异或运算）、×（与运算）。

④ 密文分组长度m和明文分组长度n之间的关系：

- 若m>n，则称之为有数据扩展的分组密码；
- 若m<n，则称之为有数据压缩的分组密码；
- 当n=m且为一一映射（双射），**整体的分组加密$E_K(x)$就是x到y的置换**。

⑤ 分组密码内部实现包含一系列的**子置换和子代换操作**。

5、分组密码设计总体要求

●**设计**一个**可用的**分组密码通常需要考虑以下几个方面：

① 设计：

- 算法类型：包括加密算法、解密算法和子密钥推导算法。
- 算法参数：包括明文分组长度、密文分组长度、密钥长度。

② 评估：

- 算法性能：包括算法的（软、硬件）实现**成本**、算法的**效率**及算法的**安全性**。

③ 集成：

- 工作模式：指如何使用具体的分组密码**构建实用的密码系统**，包括电子码本、密文链接、密文反馈、输出反馈等多种工作模式。

5、分组密码设计原则

- **安全性原则**：
  - 明文分组长度n要足够大：防止生日攻击、明文穷举攻击等。
  - 密钥空间K要足够大但又不能过大：一则防止密钥穷举攻击等，二则便于密钥管理。
  - 代换加密算法E要足够复杂：**充分实现明文、密文与密钥的扩散和混淆**
    - 使得明文、密文和密钥三者之间**没有简单的关系**可循，能抗击各种已知的攻击，如差分攻击和线性攻击等；
    - 代换加密算法有高的**非线性阶数**，实现复杂的函数变换；
    - 使对手破译时除了用穷举法外，无其他捷径可循。

5、分组密码设计原则

- **简单性原则：**
  - 加密和解密运算简单，易于软件和硬件高速实现。
  - 软件实现简单：使用子块和简单的运算，子块长度自然适应软件编程。
  - 硬件实现方便：加密和解密可用同样的器件来实现。
  - 数据扩展尽可能小，一般无数据扩展有；通常取m=n。
- **关联性原则：**
  - 差错传播尽可能小，即一个密文分组的错误尽可能少的影响其他密文分组的解密。
- **灵活性原则：**
- ……

5、分组密码设计关键

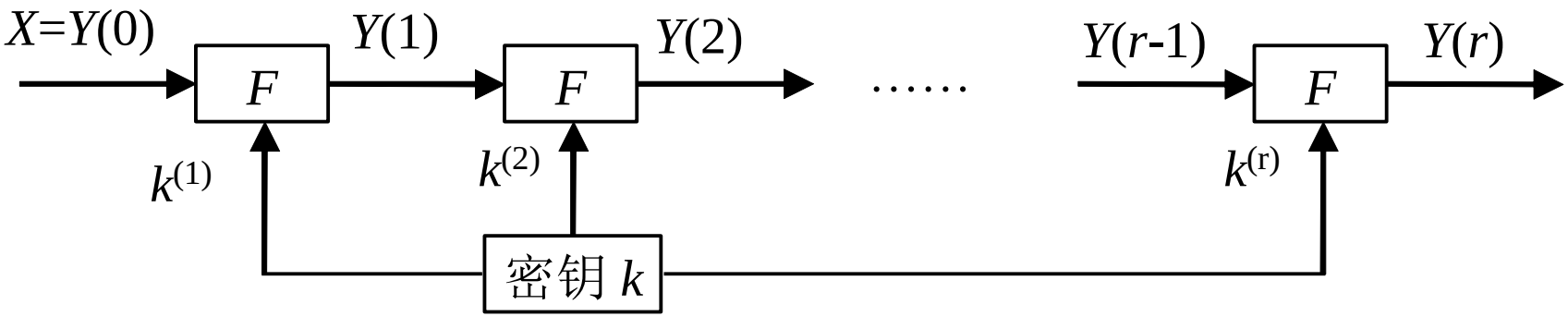
- 分组密码设计的关键:
  - 在于找到一种**加密算法**,能在**密钥控制**下从一个足够大且足够好的**置换子集**中,**简单而迅速**地**选出**一个置换,用来对当前输入的明文分组进行**加密变换**。
  - 总之，既要“**复杂**”——难于破译，又要“**简单**”——易于实现。**矛盾！！！！**
- 分组密码设计的难点:
  - **分别设计**一个**安全的**分组密码或者一个**高效的**算法并不难，但设计一个**既安全又高效还实用**的**分组密码算法**则非常困难。

6、分组密码设计思路：迭代分组密码

- **迭代分组密码（iterated cipher）的定义**：
  - 内部组成部分：
    ① 一个**轮函数F**（round function）--加密或解密
    ② 一个**密钥编排方案**（key schedule）--每轮密钥
  - 实现过程：对明文的加密将经过**r轮**，每轮的**过程类似，如下**
    ① 利用密钥编排方案得到**每轮密钥**；
    ② 针对**每一分组明文**，在**每轮密钥**作用下使用**轮函数**迭代产生**每一轮分组密文**；
    ③ 轮函数的**输入**由**前一轮的输出**和**每轮密钥**组成。

6、分组密码设计思路：迭代分组密码

■迭代分组密码的结构：



- $Y(i\text{-}1)$是第i 轮**轮函数F（置换）的输入**
- $Y(i)$是第i轮**轮函数F（置换）的输出**
- $k^{(i)}$是第i轮的**轮（子）密钥**
- k是迭代分组密码的**种子密钥**，即分组密码的**初始输入密钥**

6、分组密码设计思路：迭代分组密码

- **密钥编排方案的定义：**
  - 假设k是分组密码的**输入密钥（种子密钥）**，它是一个确定长度的随机的**二元密钥**。
  - 使用k构造N个轮密钥(也称为子密钥)，表示为 $K^{(1)}$，…$K^{(N)}$。
  - 轮密钥的列表($K^{(1)}$，…$K^{(N)}$)就是密钥编排方案。

- 因此，密钥编排方案就是用**密钥k**作为**输入**，用一个固定的、公开的**轮密钥生成算法**进行**构造，输出**所得。

6、分组密码设计思路：迭代分组密码

- **轮函数g的定义：分组密码设计的关键**
  - 两个输入：一个轮密钥(K^r)和一个当前状态(表示为w^{r-1})。
  - 初始状态w^0：明文x
  - 下一个状态w^r：$w^r = g(w^{r-1}, K^r)$，执行全部N轮后的状态就是密文y。
  - 操作过程如下：

$$
\begin{aligned}
w^0 &\leftarrow x \\
w^1 &\leftarrow g(w^0, K^1) \\
w^2 &\leftarrow g(w^1, K^2) \\
\vdots \quad & \vdots \quad \vdots \\
w^{\mathcal{N}-1} &\leftarrow g(w^{\mathcal{N}-2}, K^{\mathcal{N}-1}) \\
w^{\mathcal{N}} &\leftarrow g(w^{\mathcal{N}-1}, K^{\mathcal{N}}) \\
y &\leftarrow w^{\mathcal{N}}.
\end{aligned}
$$

6、分组密码设计思路：迭代分组密码

迭代分组密码的解密：

- 前提条件：
  - 为了能够解密，则如果轮函数g的第二个参数（轮密钥(K^r)）是固定的，那么轮函数必须具有双射(即一对一)属性
  - 即存在一个函数g^{-1}，$g^{-1}(g(w,y),y)=w$
- 解密过程如下：

$$
\begin{aligned}
w^{\mathcal{N}} &\leftarrow y \\
w^{\mathcal{N}-1} &\leftarrow g^{-1}(w^{\mathcal{N}}, K^{\mathcal{N}}) \\
\vdots \quad & \vdots \quad \vdots \\
w^1 &\leftarrow g^{-1}(w^2, K^2) \\
w^0 &\leftarrow g^{-1}(w^1, K^1) \\
x &\leftarrow w^0.
\end{aligned}
$$

6、分组密码设计思路：迭代分组密码

- **迭代分组密码的本质：Shannon思想的体现**
  - **每轮函数操作**都相当于**一次加密操作**，每轮函数都包含一系列的**子置换和子代换**操作。
  - 采用**乘积密码**，利用**多轮迭代操作**实现加密，相当于使用**很多加密函数**依次进行处理。
  - 通过将一个**弱的密码函数（轮函数）迭代若干次**，产生一个**强的密码函数**，对**明文和密钥**进行了必要的**混淆**和**扩散**。

7、扩散与混淆

- **问题和挑战：统计分析攻击**
  - 如果敌手知道明文的某些统计特性，如消息中不同字母出现的频率、或者可能出现的特定单词或短语，而且这些统计特性以某种方式在密文中反映出来，那么敌手就有可能得出**加密密钥或其中一部分**、或者包含加密密钥的**一个可能的密钥集合**。
- **抵抗统计分析的方法：扩散和混淆**
  - 基本思想：在Shannon所提出的理想密码的密码系统中，密文的所有统计特性都应该与所使用的密钥**独立（完善保密性）**。
  - 基本方法：**扩散和混淆**就是由Shannon提出的用于设计密码系统的两个基本方法，目的是抗击敌手对密码系统的**统计分析**。

7、扩散与混淆

扩散(Diffusion)：

- 背景：在加密过程中，如果当明文变化一个比特时,密文有某些比特不可能发生变化,则这个明文就与那些密文无关。因而，敌手在攻击这个明文比特时就可不利用那些密文比特。
- 思想：使得**密文中的每一位由明文中的多位产生**。即让明文中的每一位影响密文中的尽可能多的位；最好，让密文中的每一个字符出现的频率比在明文中出现的频率更接近于相等。
- 定义：将明文的统计特性散布到密文中去，使得**明文与密文之间的统计关系**尽可能复杂。
- 实现：在二元分组密码设计中，可以对相关数据**重复执行某个置换**，再对这一置换作用以**一个函数**，即可获得扩散的效果。

7、扩散与混淆

- 扩散示例：对英文消息$M = m_1 m_2 m_3 \cdots$ 的加密施加以下操作：

$$y_n = chr\left(\sum_{i=1}^{k} ord(m_{n+i}) (\bmod 26)\right)$$

  - 其中，$ord(m_i)$是求字母m_i对应的序号，$chr(i)$是求序号i对应的字母，则密文字母y_n就是由明文中k个连续的字母相加而得。
  - 此时，明文的统计特性将被散布到密文。因而，每一字母在密文中出现的频率比在明文中出现的频率更接近于相等。
  - 而且，双字母和多字母出现的频率也更接近于相等。

7、扩散与混淆

- 混淆(Confusion)：
  - 目的：即使敌手能得到密文的一些统计关系，但是由于密钥和密文之间统计关系复杂化，敌手无法得到密钥。
  - 定义：使密文和密钥之间的统计关系变得尽可能复杂，以使敌手无法得到密钥。
  - 实现：因为密钥主要体现在代换算法上，所以使用复杂的代换算法（线性和非线性）可得预期的混淆效果，而简单的线性代换函数得到的混淆效果不够理想。

7、扩散与混淆

- 混淆和扩散在原则上就是将**密文、明文、密钥三者之间**的**统计关系和代数（线性和非线性）关系**变得尽可能复杂：
  ① 使得敌手即使获得了密文和明文,也无法求出密钥的任何信息—**混淆**;
  ② 即使获得了密文和明文的统计规律,也无法求出明文的新的信息—**扩散**。
- 混淆和扩散的结果，使得:
  ① **明文**不能由已知的明文,密文及少许密钥比特代数地或统计地表示出来;
  ② **密钥**不能由已知的明文,密文及少许密钥比特代数地或统计地表示出来。

7、扩散与混淆

- 扩散和混淆的思想满足了分组密码设计的**安全性原则**，因而成为**设计现代分组密码**的基础。
- **混淆和扩散，代换和置换！比如：**

① **先利用多个复杂的代换函数实现混淆（密钥和密文）**

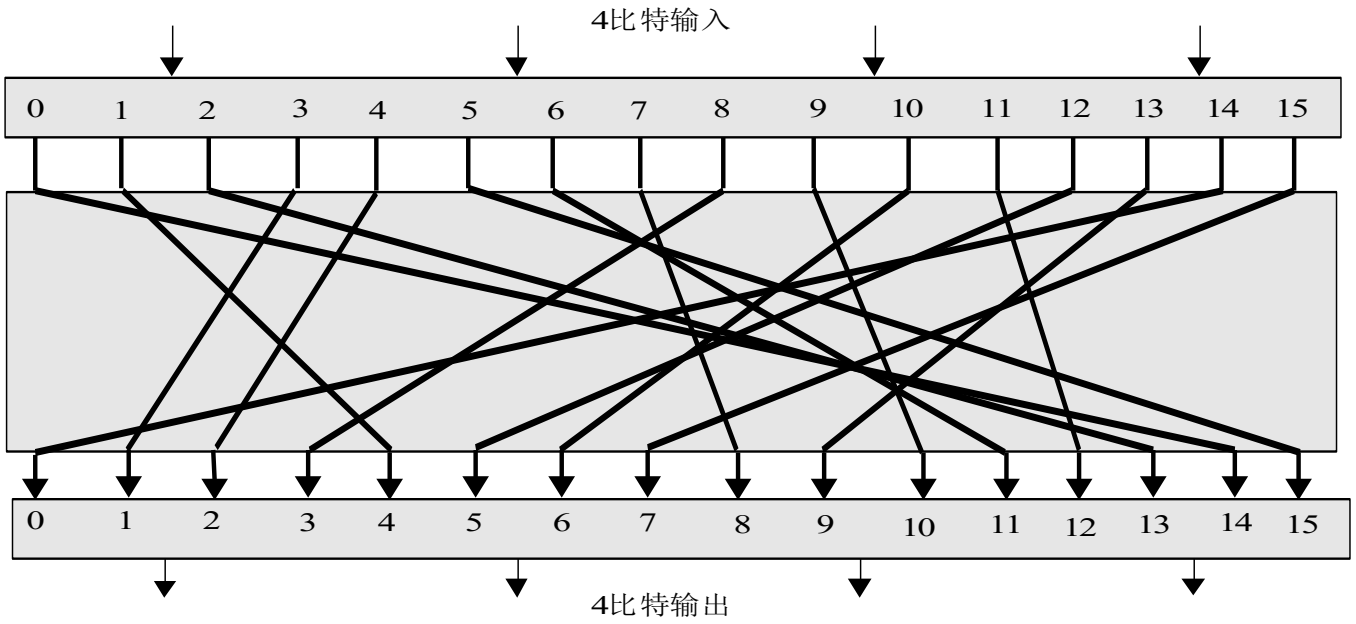
② **再利用一次置换操作实现扩散（明文和密文）**

8、分组密码实现的基本操作：代换和置换

- 如果明文和密文的分组长度都为 n比特，则明文的每一个分组都有 2^n个可能的取值。
- 为使加密运算可逆（使解密运算可行），明文的每一个分组都应产生**惟一**的一个密文分组，这样的变换是**可逆的**，称明文分组到密文分组的**可逆变换**为**代换**。
  - 因此，不同可逆变换的个数有 $2^n!$个。

代换结构的概念：

- 分组长度为4的代换算法的一般结构：代换结构
  - 4比特输入产生16个可能输入状态中的一个，由代换结构将这一状态映射为16个可能输出状态中的一个，每一输出状态由4个密文比特表示。



代换表的概念：

- **代换表**是分组密码最常见的形式，能用于**定义**明文和密文之间的任何**可逆映射**。
  - 加密映射和解密映射可由**代换表**来定义。
  - **代换表的个数（密钥空间）**是多少？ $2^n!$

明文		密文		明文		密文
0000		1110		1000		0011
0001		0100		1001		1010
0010		1101		1010		0110
0011		0001		1011		1100
0100		0010		1100		0101
0101		1111		1101		1001
0110		1011		1110		0000
0111		1000		1111		0111

密文		明文		密文		明文
0000		1110		1000		0111
0001		0011		1001		1101
0010		0100		1010		1001
0011		1000		1011		0110
0100		0001		1100		1011
0101		1100		1101		0010
0110		1010		1110		0000
0111		1111		1111		0101

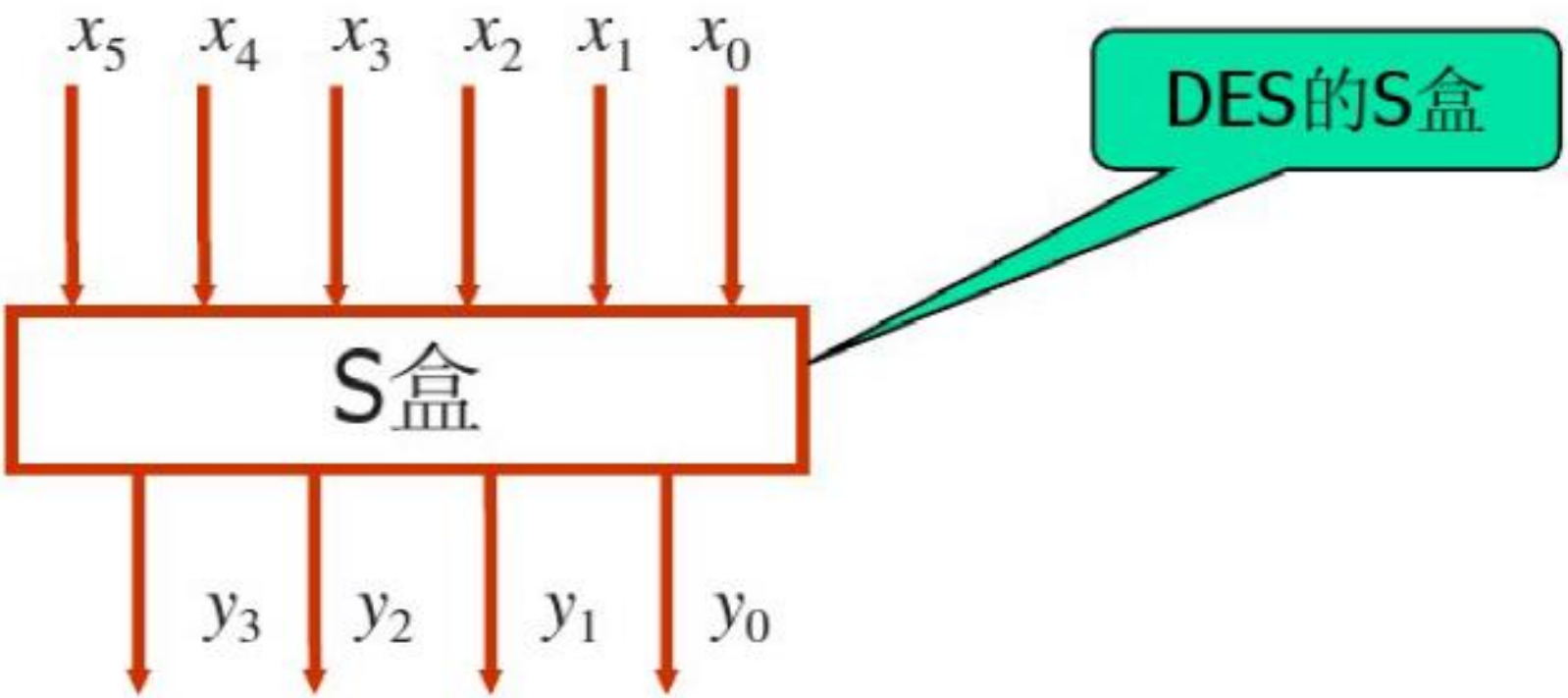
代换结构中分组长度的设计需求：

- 理论上：
  - 如果分组长度太小，如n=4，则分组密码基本等价于古典的**代换密码**，容易通过对明文的统计分析而攻破。
  - 如果分组长度足够大，则从明文到密文可有足够多的**可逆变换**，那么明文的统计特性将被隐藏而不能被有效攻击 。

代换结构中分组长度的设计需求：

- 实际上：
  - 分组长度太大也不可取，因为对n比特的代换结构，**密钥的大小**是 $n*2^n$（n是密钥的长度，2^n是代换表的大小）。比如，对64比特的分组，**密钥大小**为 $64*2^{64}=2^{70}=10^{21}$，因此，难以处理。
  - 实际设计中，**将n比特长度分成多个较小的段。**比如 $n=r*n^0$，将设计**n个输入变量**的代换**转化为**设计**r个**较小的**子代换**，而每个子代换只有**n^0个输入变量**。一般，n^0都不太大。
  - 一般称**每个子代换**为**代换盒**，简称**S盒**。用**多个S盒**实现**迭代的轮函数F**。

DES代换盒S盒的结构



DES代换盒S1盒的输入和输出关系：

x_5 x_0

1 0

x_5 x_4 x_3 x_2 x_1 x_0

1 0 1 1 0 0

$(y_3, y_2, y_1, y_0)=(0,0,1,0)$

行号 \ 列号	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	2	14	10	0	6	13

单个S盒的设计准则：

- 迄今为止,有关方面未曾完全公开有关DES的S盒的设计准则，但 Branstad曾披露过下述准则:

① S盒的输出都不是其输入的线性或仿射函数—**非线性**。

② 改变S盒的一个输入比特,其输出至少有两比特产生变化，即近一半产生变化。

③ 当S盒的任一输入比特保持不变,其它5比特输入变化时(共有2^5=32种情况），输出数字中的0和1的总数接近相等。

- 以上三点使得DES的S盒能够实现较好的**混淆**。

多个S盒的组合构建：

- 如何将几个**S盒组合**构成一个**n值较大的分组**
  - 将几个S盒的**输入端并行**,并通过**位置置换(P盒)**将各个S盒**输出比特次序打乱**,再作为下一级**各个S盒的输入**,进而实现 Shannon所谓的“**扩散**”功能。
    - **S盒**提供非线性变换,将来自上一级不同的S盒的输出进行“混淆”。
    - P盒通过置换，实现“扩散”，使分组长度为n的各个比特均匀地分散到整个输出矢量中,从而保证了**输出密文统计上的均匀性**。
    - 以上过程起到Shannon所提出的“**乘积密码**”的作用。

8、分组密码实现的基本操作： S-变换与P-变换

- 分组密码可以看成是一系列**S-变换与P-变换的组合**。
  - 代换（Substitution，简称**S-变换**）：用一个向量代替另一个向量（两个向量的维数可能不一样），又称为**S-盒**。
  - 置换（Permutation，简称**P-变换**）：一个有限集到其自身的一个一一映射。
- 典型的分组密码结构：
  - **代换-置换网络**和**Feistel 网络**
  - 他们都是分组密码的一种**分组设计结构**，而不是一种具体的密码算法

2.1.2 代换-置换网络（SPN）

1、SPN的基本思想

- SPN （Substitution-Permutation Network）包含N轮，在**每一轮中**(除了最后一轮稍有不同):

  ① 首先通过一个简单的异或操作来**混合轮密钥**

  ② 然后使用πS进行m次代换

  ③ 最后使用πP进行1次置换

SPN的分组：

给定一个 lm 比特的二元串 $x=(x_1,\cdots,x_{lm})$，可以将其看做 m 个 l 比特子串 $x_{(1)}\cdots,x_m$ 的并联。因此，

$$x = x_{(1)} \| \cdots \| x_{(m)}$$

且对于 $1\leqslant i\leqslant m$，我们有

$$x_{(i)} = (x_{(i-1)l+1},\cdots,x_{il})$$

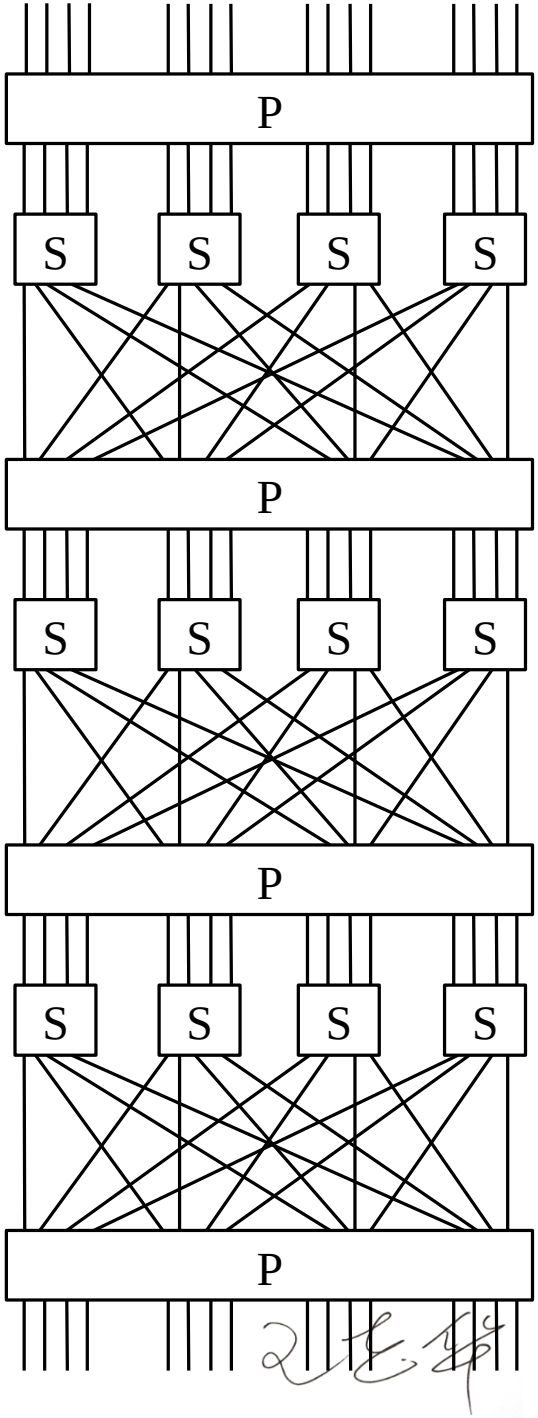
SPN的描述：

- 假设l和m是正整数，明文和密文都是单位长度为l*m的二元向量(即l*m是该分组密码的分组长度)。则SPN由两个变换组成，分别表示为πS和πP，他们都是置换（每个变换的输入长度和输出长度都相同）：
  - πS（S盒）是长度为l的**2^l比特串的一个置换**，它用一个l比特向量来代替另一个l比特向量；
  - πP（P盒）是整数**1,…l*m（位置）的一个置换**，它通过改变l*m比特的顺序来实现置换。

$$\pi_S : \{0,1\}^\ell \to \{0,1\}^\ell$$

$$\pi_P : \{1,\ldots,\ell m\} \to \{1,\ldots,\ell m\}$$

SPN的模型：



2、SPN的密码体制

密码体制 3.1　代换-置换网络

设 l, m 和 Nr 都是正整数，$\pi_S:\{0,1\}^l \to \{0,1\}^l \to$ 与 $\pi_P:\{1,\cdots,lm\} \to \{1,\cdots,lm\}$ 均为置换。

设 $\mathcal{P}=\mathcal{C}=\{0,1\}^{lm}$，$\mathcal{K} \subseteq (\{0,1\}^{lm})^{Nr+1}$ 是由初始密钥 K 用密钥编排算法生成的所有可能密钥编排方案之集。对一个密钥编排方案 $(K^1,\cdots,K^{Nr+1})$，我们使用算法 3.1 来加密明文 x。

在算法 3.1 中，u^r 是第 r 轮对 S 盒的输入，v^r 是第 r 轮对 S 盒的输出。w^r 由 v^r 应用置换 π_P 得到，然后 u^{r+1} 由轮密钥 K^{r+1} 异或 w^r 得到(这叫做轮密钥混合)，最后一轮没有用置换 π_P。

Algorithm 4.1: $\mathrm{SPN}(x, \pi_S, \pi_P, (K^1, \ldots, K^{\mathcal{N}+1}))$

$$
\begin{aligned}
& w^0 \leftarrow x \\
& \textbf{for } r \leftarrow 1 \textbf{ to } \mathcal{N}-1 \\
& \quad \textbf{do} \begin{cases} u^r \leftarrow w^{r-1} \oplus K^r \\ \textbf{for } i \leftarrow 1 \textbf{ to } m \\ \quad \textbf{do } v^r_{<i>} \leftarrow \pi_S(u^r_{<i>}) \\ w^r \leftarrow (v^r_{\pi_P(1)}, \ldots, v^r_{\pi_P(\ell m)}) \end{cases} \\
& u^{\mathcal{N}} \leftarrow w^{\mathcal{N}-1} \oplus K^{\mathcal{N}} \\
& \textbf{for } i \leftarrow 1 \textbf{ to } m \\
& \quad \textbf{do } v^{\mathcal{N}}_{<i>} \leftarrow \pi_S(u^{\mathcal{N}}_{<i>}) \\
& y \leftarrow v^{\mathcal{N}} \oplus K^{\mathcal{N}+1} \\
& \textbf{output } (y)
\end{aligned}
$$

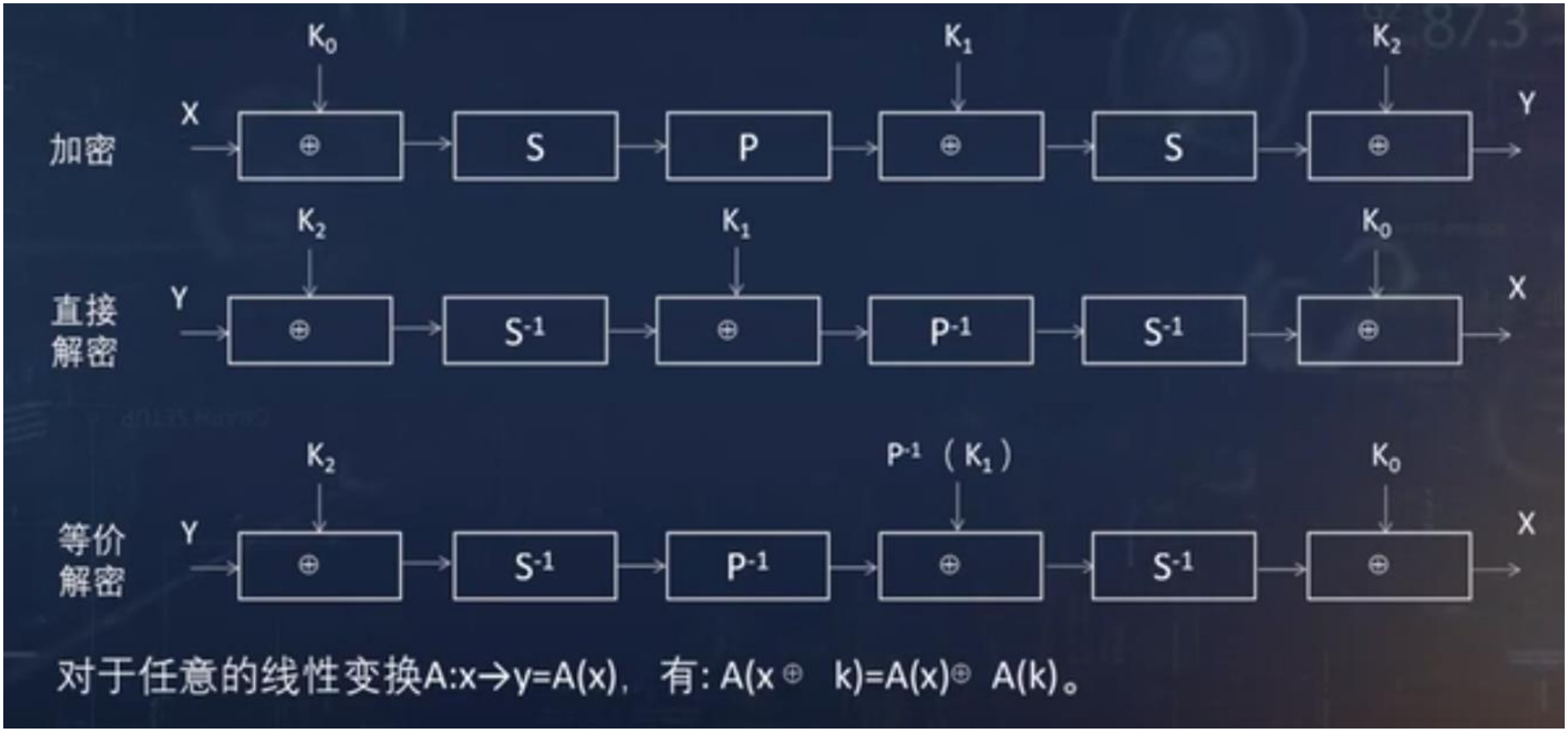
因此，如果对密钥编排方案作适当修改并用**S盒的逆**来取代S盒，那么该加密算法也能用来**解密**。

为什么在开始和结束要白化呢？

可逆的需要！！！

SPN中执行的第一个和最后一个操作都是**异或轮密钥（子密钥）**。这被称为**白化（whitening）**，它被认为是一种有效的方法，可以防止攻击者在不知道密钥的情况下开始进行加密或解密操作

3、SPN的加解密结构



对于任意的线性变换A:x→y=A(x)，有：A(x ⊕ k)=A(x)⊕ A(k)。

4、SPN的示例

例 3.1 设 $l=m=N_r=4$，π_S 如下定义，这里输入 z 和输出 $\pi_S(z)$ 都以十六进制表示（$0\leftrightarrow(0,0,0,0)$，$1\leftrightarrow(0,0,0,1)$，…，$9\leftrightarrow(1,0,0,1)$，$A\leftrightarrow(1,0,1,0)$，…，$F\leftrightarrow(1,1,1,1)$）：

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

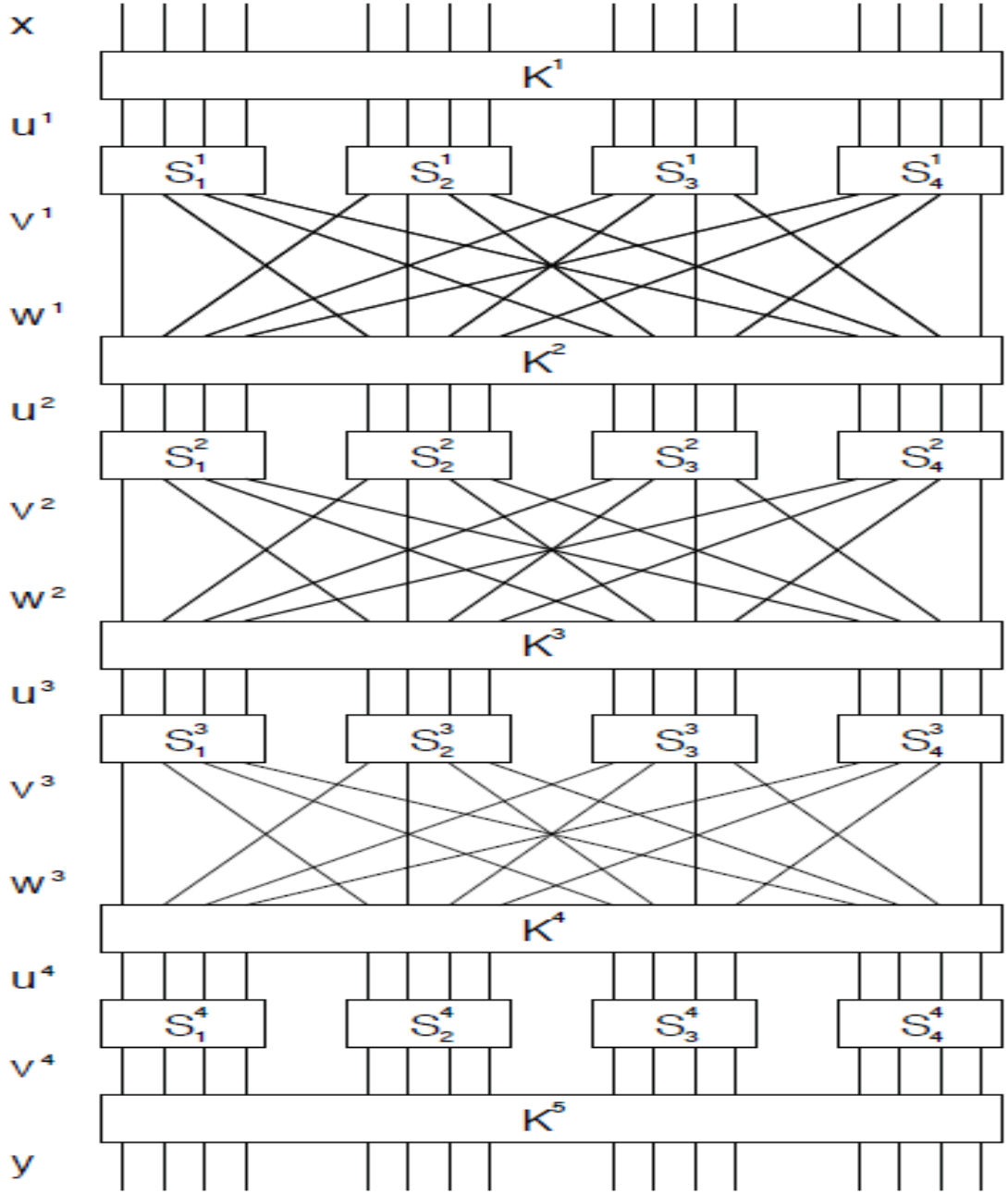
π_P 如下定义：

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

下图是这个特定SPN的图形表示。S盒为 $S_i^r\ (1\le i\le 4, 1\le r\le 4)$

所有的**16个S盒**都基于**同样的代换函数π_S**进行**混淆**。

这样设置会有攻击风险吗？



置换P

FIGURE 4.1: A substitution-permutation network

4、SPN的示例

- 密钥编排算法的设计：
  - 从一个32比特的**种子密钥K**开始，$K = (k_1, \ldots, k_{32}) \in \{0,1\}^{32}$
  - 对轮数1≤r≤5，定义**轮密钥（子密钥）K^r**是由K中从k_{4r-3}开始的16个连续的比特组成

- 假设密钥K如下：$K = 0011\ 1010\ 1001\ 0100\ 1101\ 0110\ 0011\ 1111$

则轮密钥如下：

$$
\begin{aligned}
K^1 &= 0011\ 1010\ 1001\ 0100 \\
K^2 &= 1010\ 1001\ 0100\ 1101 \\
K^3 &= 1001\ 0100\ 1101\ 0110 \\
K^4 &= 0100\ 1101\ 0110\ 0011 \\
K^5 &= 1101\ 0110\ 0011\ 1111.
\end{aligned}
$$

- 设明文为：$x = 0010\ 0110\ 1011\ 0111.$

4、SPN的示例

对明文x加密过程如右图

$$
\begin{aligned}
w^0 &= 0010\ 0110\ 1011\ 0111 \\
K^1 &= 0011\ 1010\ 1001\ 0100 \\
u^1 &= 0001\ 1100\ 0010\ 0011 \\
v^1 &= 0100\ 0101\ 1101\ 0001 \\
w^1 &= 0010\ 1110\ 0000\ 0111 \\
K^2 &= 1010\ 1001\ 0100\ 1101 \\
u^2 &= 1000\ 0111\ 0100\ 1010 \\
v^2 &= 0011\ 1000\ 0010\ 0110 \\
w^2 &= 0100\ 0001\ 1011\ 1000 \\
K^3 &= 1001\ 0100\ 1101\ 0110 \\
u^3 &= 1101\ 0101\ 0110\ 1110 \\
v^3 &= 1001\ 1111\ 1011\ 0000 \\
w^3 &= 1110\ 0100\ 0110\ 1110 \\
K^4 &= 0100\ 1101\ 0110\ 0011 \\
u^4 &= 1010\ 1001\ 0000\ 1101 \\
v^4 &= 0110\ 1010\ 1110\ 1001 \\
K^5 &= 1101\ 0110\ 0011\ 1111 \\
y &= 1011\ 1100\ 1101\ 0110
\end{aligned}
$$

5、SPN的特点

- 以上SPN的特点：
  - **简单、有效：**软件、硬件实现方便。软件实现S盒采用查表，硬件实现相对较小的S盒非常迅速。
  - **不甚安全：**32位的密钥长度对穷尽密钥搜索来讲太短。但是，可以设计更大的SPN，使其能够安全地抵御所有已知的攻击。
- 一个实用且安全的SPN：
  - 更长的密钥长度
  - 更长的分组长度
  - 更大的S盒
  - 更多的轮数
- Rijndael算法实现的AES：
  - 最小密钥长度为128比特
  - 最小分组长度为128比特
  - S-盒实现8比特到8比特映射
  - 最少轮数是10轮

6、SPN的变体

- SPN改进思想：
  - **使用多个不同的s盒**。例如在以上示例中，可以在每轮中使用四个不同的s盒，而不是使用相同的s盒。
  - 除了S盒实现非线性变换，**在每一轮中增加一个可逆的线性变换（类似于希尔密码体制中的线性变换）**，可以作为置换操作的替代（**前面分析过置换密码体制是特殊的希尔密码体制**），也可以作为置换操作的补充。

7、SPN的应用

- SPN改进思想在具体分组密码算法中的应用：
  - DES：在每一轮使用**8个不同的S盒**。
  - AES：包含**可逆的线性变换**。

2.1.3 Feistel网络

1、Feistel网络的基本思想

- Feistel网络的构建思想：

① 利用代换和置换的基本方法

② 利用乘积密码

➢ 顺序地执行两个或多个基本密码系统，使得最后**组合的密码强度高于每个基本密码系统的密码强度**。

Feistel的描述：

- Feistel网络加密结构
  - 分为n轮，第一轮：
    - 输入是分组长度为 $2w$ 比特的明文和一个密钥K
    - 处理前将比特数据分成左右两半 L_0 和R_0
  - 在进行完 n 轮迭代后：
    - 将左右两半再合并到一起以产生$2w$ 比特长度的分组密文

Feistel的描述：

- Feistel网络加密结构
  - 第 i 轮迭代：
    - 输入为前一轮输出的函数，分为左右两个部分：

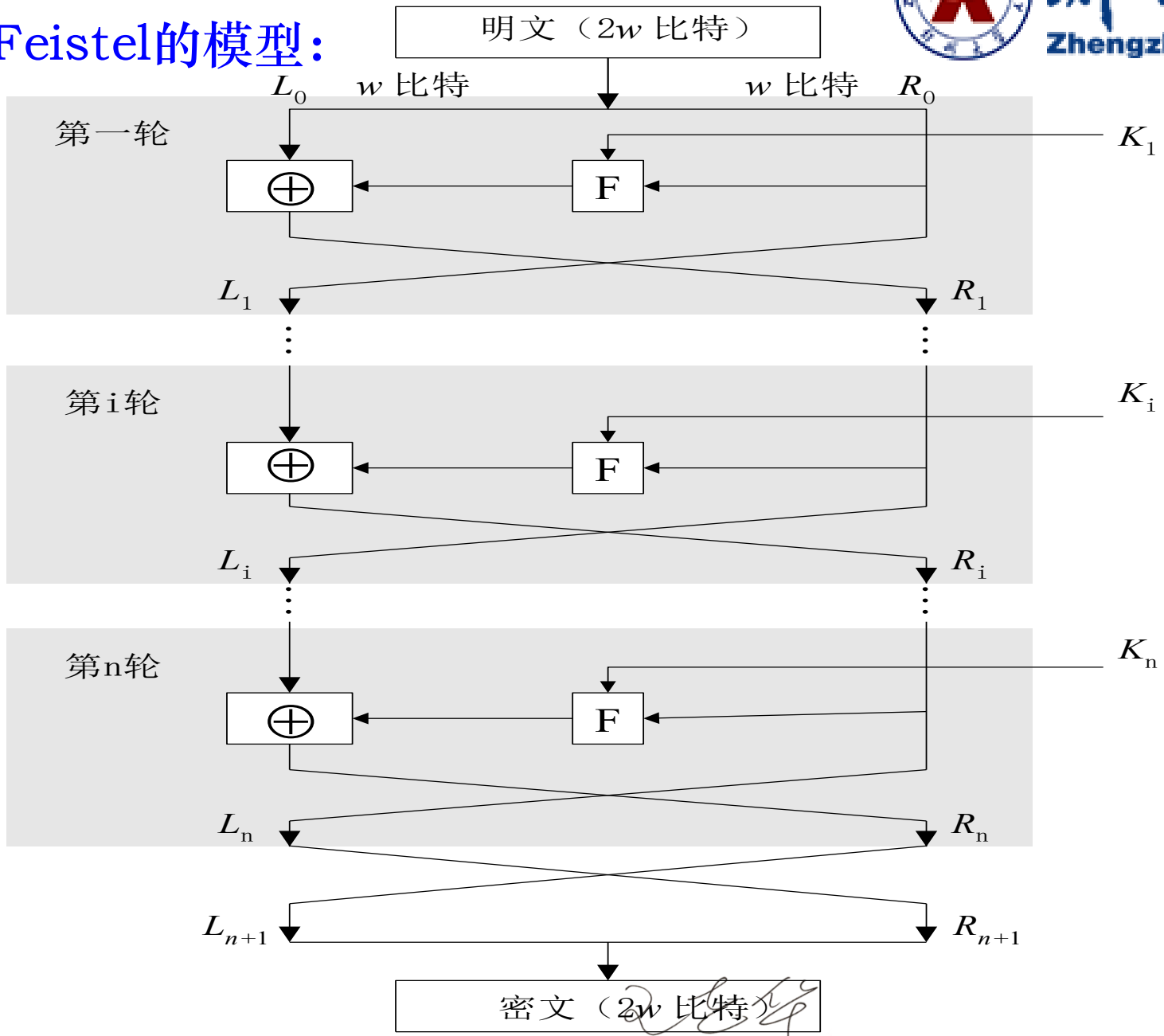
$$L_i = R_{i-1}$$

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

✓ 其中 K_i 是第 i 轮用的子密钥，由分组密码输入密钥 K 获得；F 为轮函数

➢ 一般，各轮子密钥彼此不同而且与 K 也不同

Feistel的模型：


明文（2w 比特）
L_0
w 比特
w 比特
R_0
第一轮
K_1
F
L_1
R_1
第i轮
K_i
F
L_i
R_i
第n轮
K_n
F
L_n
R_n
L_{n+1}
R_{n+1}
密文（2w 比特）

Feistel的说明：

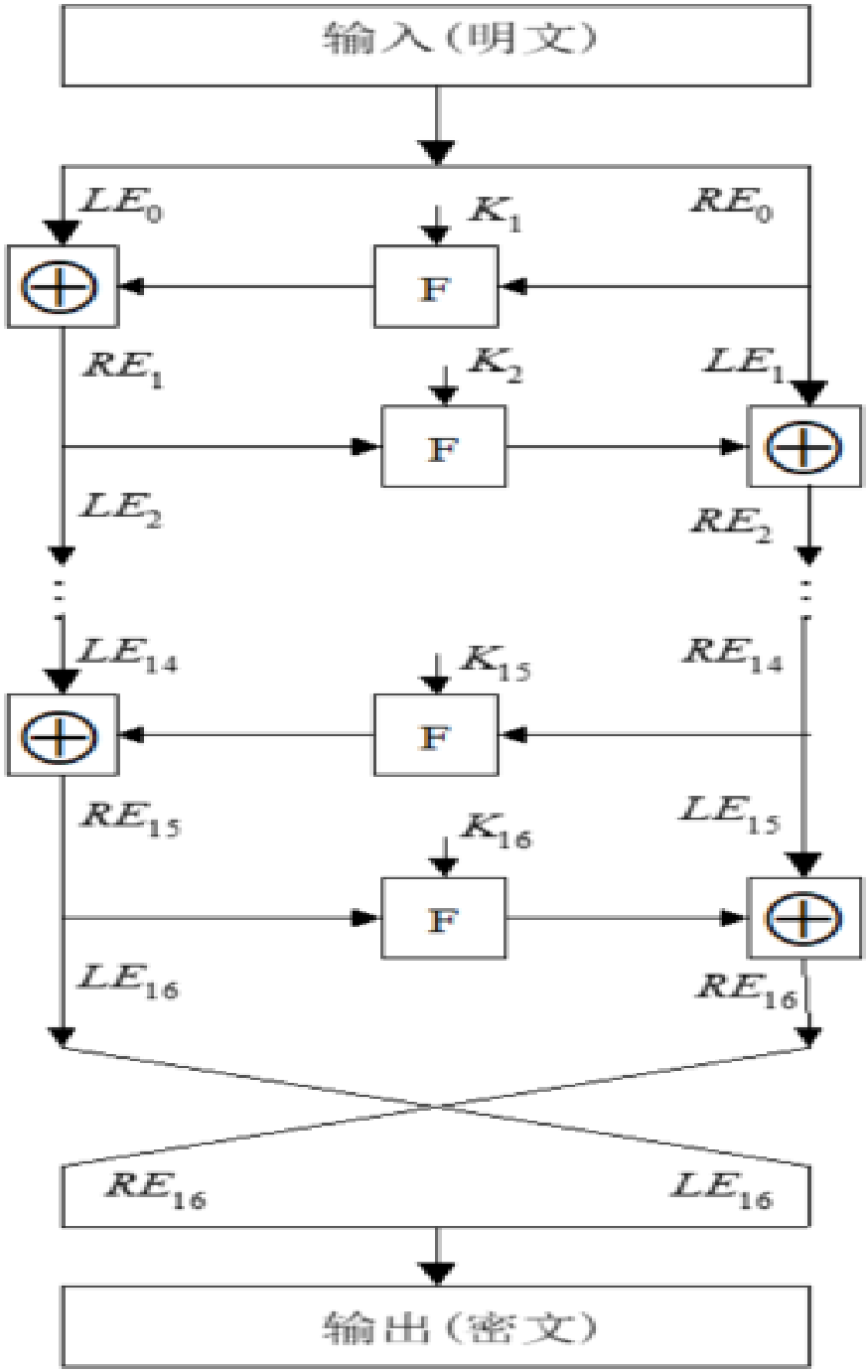
- Feistel网络是SPN的特有形式：
  - 每轮模型结构都相同；
  - 每轮轮函数F都相同，但以不同的子密钥K_i作为输入；
  - 每轮中右半部分数据被作用于轮函数F后，再与左半数据进行异或运算，这一过程本质上就是代换（类似于SPN中的S盒）；
  - 代换过程完成后，再交换左、右两半部分数据，这一过程本质上就是置换（类似于SPN中的P盒）。
- Feistel网络的特有结构保证了**加密和解密可以采用同一个算法实现**。

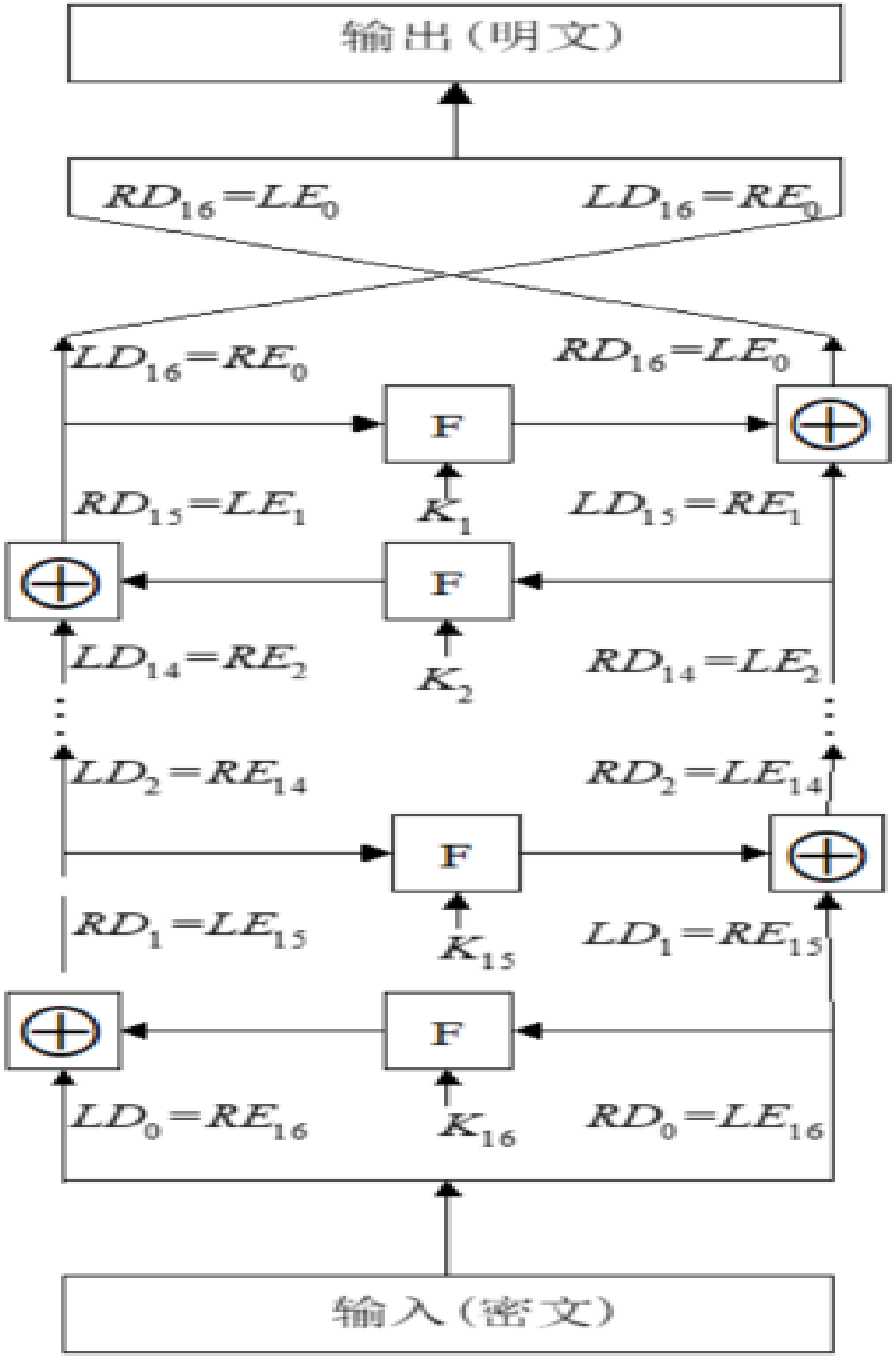
Feistel的实际应用：

- Feistel网络的**实际应用**需要考虑以下**参数的设置**：
  - 分组大小：分组长度越大则安全性越高，但加密速度就越慢。
  - 密钥大小：密钥长度越大则安全性越高，但加密速度就越慢。
  - 轮数：轮数越多则安全性越高，但计算量增大。
  - 子密钥产生算法：该算法的复杂性越大，则密码分析的困难性就越大。
  - 轮函数：轮函数的复杂性越大，密码分析的困难性也越大。
- Feistel网络的**实际应用还**需要考虑以下**实现的特点**：
  - 快速的软件实现：在很多情况下,算法是被镶嵌在应用程序中,因而无法用硬件实现。此时算法的执行速度是考虑的关键。
  - 算法容易分析：如果算法能被无疑义地解释清楚,就可容易地分析算法抵抗攻击的能力,有助于设计高强度的算法。

2、Feistel网络的解密

- Feistel网络解密结构：本质上和加密过程一样
  - 解密算法使用密文作为输入。
  - 轮函数使用子密钥K_i的次序与加密过程相反，即第一轮使用 K_n，第二轮使用 K_{n-1}，一直下去，最后一轮使用K_1 。此特性保证了解密和加密可采用同一算法。


输入（明文）
LE_0
K_1
RE_0
F
RE_1
K_2
LE_1
F
LE_2
RE_2
LE_{14}
K_{15}
RE_{14}
F
RE_{15}
K_{16}
LE_{15}
F
LE_{16}
RE_{16}
RE_{16}
LE_{16}
输出（密文）


输出（明文）
$RD_{16}=LE_0$
$LD_{16}=RE_0$
$LD_{16}=RE_0$
$RD_{16}=LE_0$
F
$RD_{15}=LE_1$
K_1
$LD_{15}=RE_1$
F
$LD_{14}=RE_2$
K_2
$RD_{14}=LE_2$
$LD_2=RE_{14}$
$RD_2=LE_{14}$
F
$RD_1=LE_{15}$
K_{15}
$LD_1=RE_{15}$
F
$LD_0=RE_{16}$
K_{16}
$RD_0=LE_{16}$
输入（密文）

3、 Feistel网络的示例

- Feistel示例：
  - 左边表示16轮Feistel网络的加密过程，右边表示解密过程。
    - 加密过程由上而下，解密过程由下而上。
  - 加密过程每轮的左右两半用 LE_i 和 RE_i 表示，解密过程每轮的左右两半用 LD_i 和 RD_i 表示。
  - 右边标出了解密过程中每一轮的中间值与左边加密过程每一轮中间值的**对应关系**：
    - 加密过程第i轮的输出是$LE_i||RE_i$（||表示链接），则解密过程第16-i轮相应的输入是$RD_i||LD_i$。

3、 Feistel网络的示例

- Feistel示例：
  - 加密过程的最后一轮执行完毕后，两半输出再交换，因此密文是$RE_{16}||LE_{16}$。
  - 解密过程取以上密文作为同一算法的输入，即第一轮输入是$RE_{16}||LE_{16}$，等于加密过程第16轮两半输出交换的结果。
  - 在加密过程中：$LE_{16} = RE_{15}$

    $RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$
  - 在解密过程中：

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$RD_1 = LD_0 \oplus F(RD_0, K_{16}) = RE_{16} \oplus F(RE_{15}, K_{16})$$

$$= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$$

$$= LE_{15}$$

3、 Feistel网络的示例

所以解密过程第1轮的输出为$RE_{15} \| LE_{15}$，等于加密过程第16轮输入左右两半交换后的结果。容易证明这种对应关系在16轮中每轮都成立。一般地，加密过程的第i轮有

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F\left(RE_{i-1},\ K_i\right)$$

因此解密

$$RE_{i-1} = LE_i$$

$$LE_{i-1} = RE_i \oplus F\left(RE_{i-1},\ K_i\right) = RE_i \oplus F\left(LE_i,\ K_i\right)$$

3、 Feistel网络的示例

- Feistel示例：
  - 以上两个式子描述了加密和解密过程中第i轮的输入与第i轮的输出之间的函数关系。此关系表示为右图中LD_i和RD_i的取值关系。
  - 解密过程最后一轮的输出是$RE_0||LE_0$，左右两半再经过一次交换后即可得到最初的明文。

4、 Feistel的特点

- 加密过程和解密过程充分利用了**异或操作的特性**。
- Feistel网络中**对所有函数F都适用**。
- Feistel网络中**只对特定函数F来说是安全的**。
- Feistel网络中**轮函数F不需要可逆**。
- Feistel网络的优点在于所有安全性问题可以**转化为轮函数F的安全性问题**。

5、 SPN和Feistel网络的比较

- SPN结构（典型代表AES）
  - 加解密结构相似
  - 扩散速度快
- Feistel结构（典型代表DES）
  - 解密算法同加密算法，只是轮密钥逆序使用。
  - 结构保证可逆，S盒不需要可逆。

思考：

- 分组密码的设计思想和设计原则是什么？
- 什么是混淆和扩散？
- SPN网络的分组密码如何设计，有什么特点？
- Feistel的分组密码如何设计，有什么特点？
- SPN网络和Feistel的分组密码有何异同点？

要求：

- 以例子4.1为例，代码实现SPN加密算法。
- 自己构建轮函数F，代码实现Feistel加密算法。