21世纪高等学校计算机规划教材

现代密码学

Modern Cryptography

作　者：何大可 彭代渊 唐小虎 何明星 梅其祥

出版社：人民邮电出版社

现代密码学

Modern Cryptography

第3章 分组密码

孙玉花

中国石油大学 理学院

Sunyuhua_1@163.com

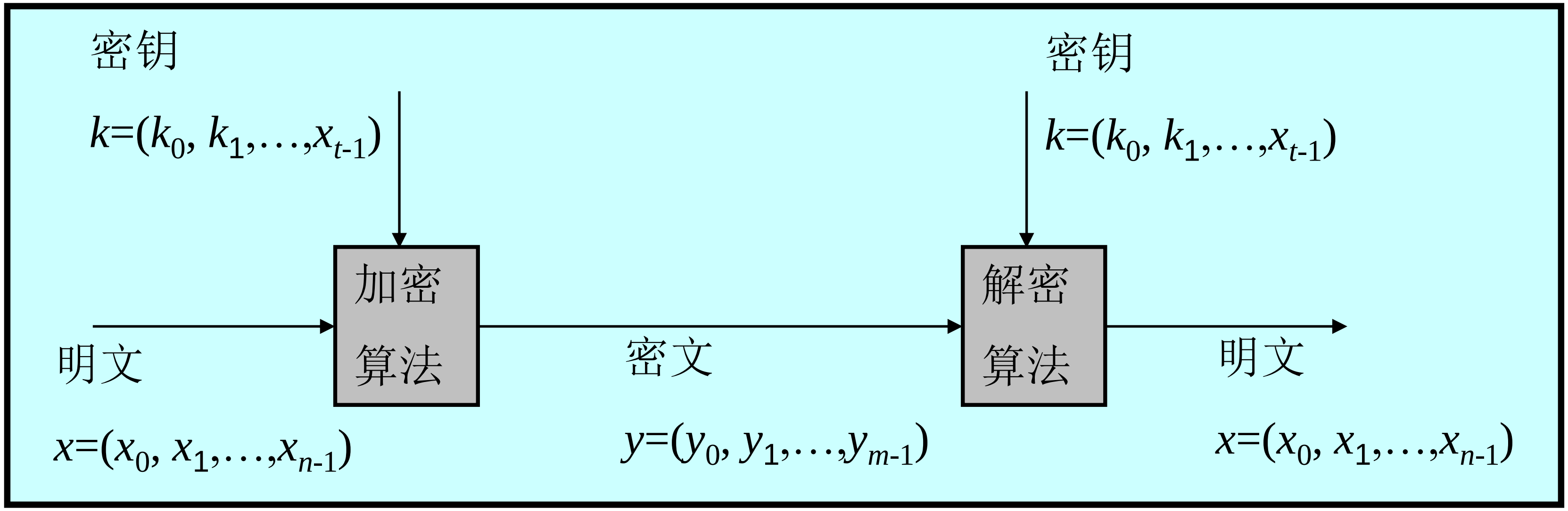
2019年9月

第 3章　分组密码

- 3.1 分组密码概述
- 3.2 数据加密标准（DES）
- 3.3 国际数据加密算法（IDEA）
- 3.4 高级数据加密标准（AES）
- 3.5 分组密码工作模式

3.1 分组密码概述

- **分组密码（block cipher）框图**



- 分组长度：n
- 数据扩展：$n<m$
- 数据压缩：$n>m$
- 一般要求：$n=m$; $x_i, y_i \in\{0,1\}$.

3.1 分组密码概述

- **加密算法**

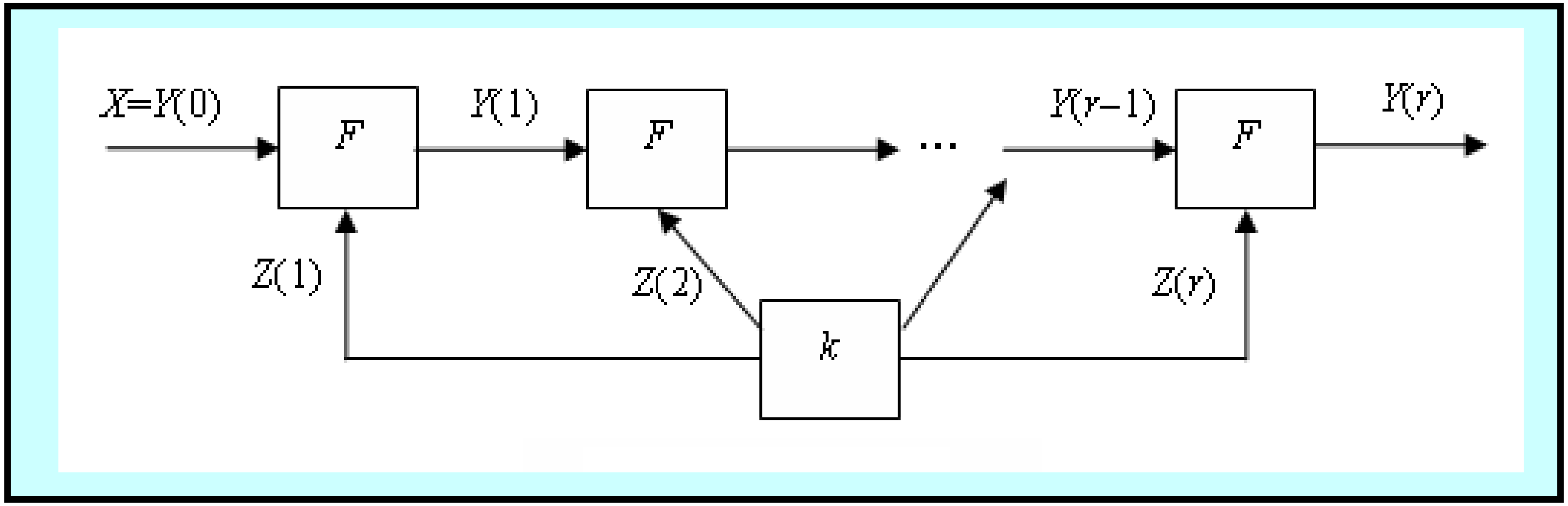
$$E_k : F_2^n \to F_2^n$$

$$x = (x_0, x_1, ..., x_{n-1}) \to y = E_k(x) = (y_0, y_1, ..., y_{n-1})$$

- **分组密码算法基本要求**
  - 分组长度n足够大
  - 密钥长度t足够大
  - 加密算法足够复杂
  - 差错传播尽可能小

3.1 分组密码概述

- **迭代密码**



- 明文：$X=Y(0)$
- 密文：$Y=Y(r)$
- 迭代 函数:F
- 迭代 次数:r
- 种子密钥：k
- 迭代的子密钥：$Z(i)$

Feistel 密码

64=2w

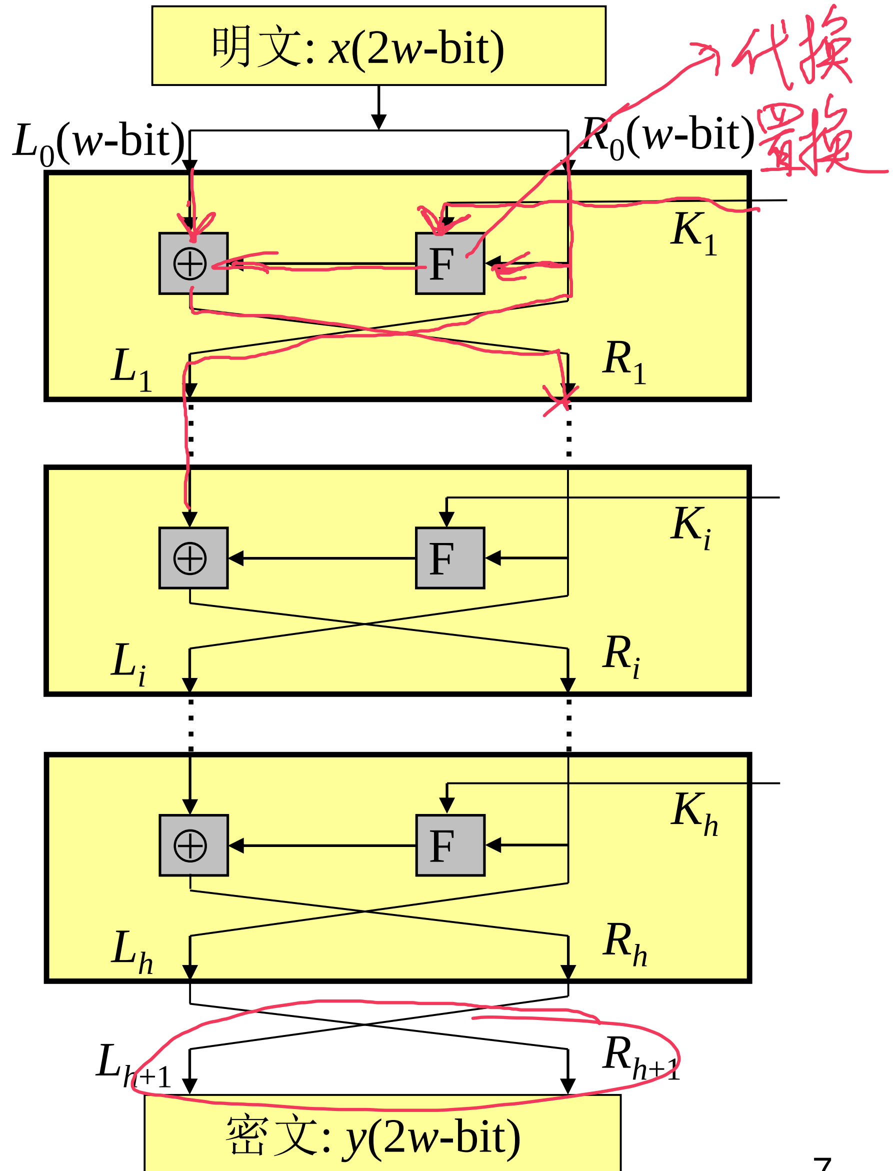
- **Feistel加密结构**
  - 子密钥产生算法

    $K \Rightarrow K_1, K_2, \ldots, K_h.$
  - 明文:$x = L_0 \| R_0$
  - 第i轮迭代

    $L_i = R_{i-1}$

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

  F: 轮函数
  - 密文:$y = L_{h+1} \| R_{h+1}$
  - 代换-置换网络 (substitution-permutation network)

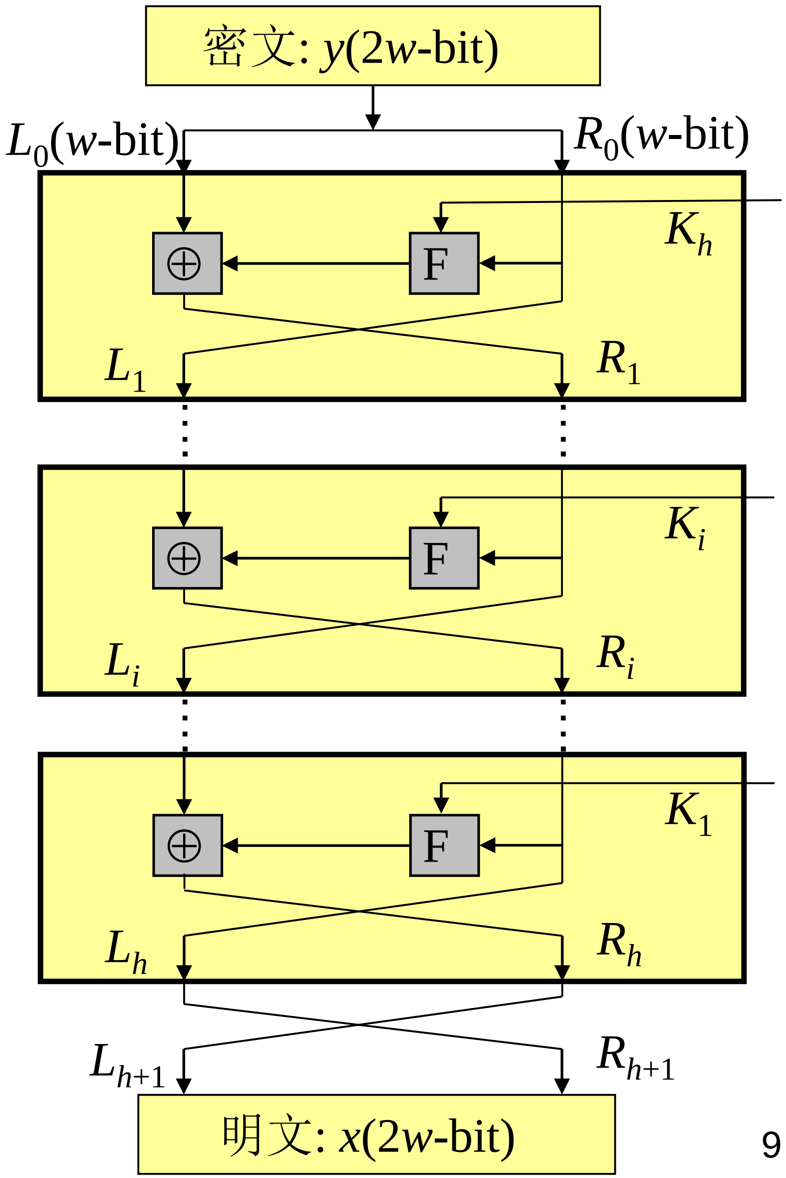


Feistel 密码

- **Feistel代换-置换网络(substitution-permutation network)**
- **1971年，IBM的Feistel H. 领导的项目组首次提出，并用于劳埃德保险公司的现金分配系统**
- **Feistel代换-置换网络主要参数**
  - 分组大小: $(2w=64)$
  - 密钥大小: $(|K|=128)$
  - 轮数:h
  - 子密钥产生算法: $K \Rightarrow K_1, K_2, \ldots, K_h$.
  - 轮函数设计: F

Feistel 密码

- **Feistel解密结构**
  - **与加密结构相同**
  - **子密钥使用次序相反:**
    $K_h, K_{h-1},\ldots,K_2,K_1$
  - **输入：密文*y***
  - **输出：明文*x***



第 3章 分组密码

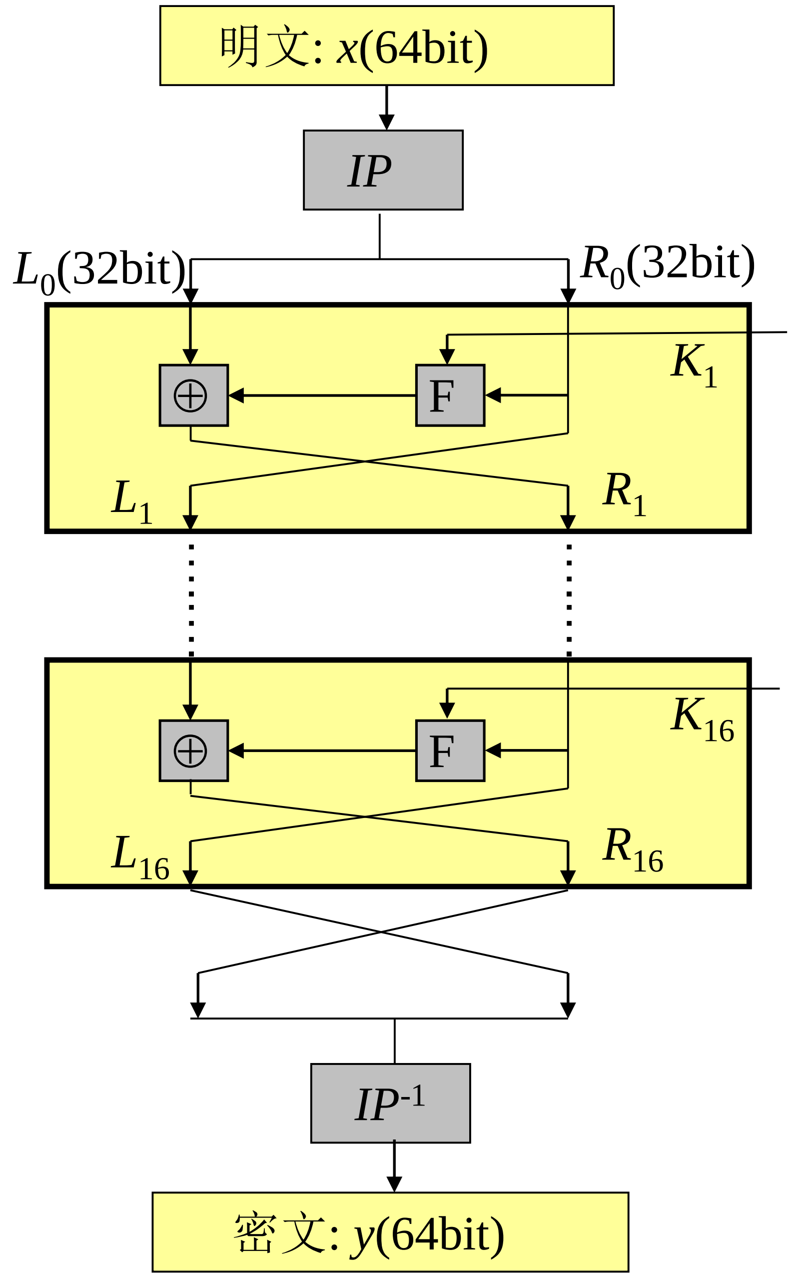
- 3.1 分组密码概述
- 3.2 数据加密标准（DES）
- 3.3 国际数据加密算法（IDEA）
- 3.4 高级数据加密标准（AES）
- 3.5 分组密码工作模式

3.2 数据加密标准(DES)

- **数据加密标准(DES: data encryption standard)概况**
  - 1972美国国家标准局(NBS)开始实施计算机数据保护标准的开发计划
  - 1973.5.13 NBS发布文告征集在传输和存储数据中保护计算机数据的密码算法
  - 1975.3.17首次公布DES算法描述，进行公开讨论
  - 1977.1.15正式批准为无密级应用的DES (美国联邦信息处理标准：FIPS-46)，1977.7.15正式生效
  - 以后每5年NBS做出评估，并重新确定是否继续作为加密标准
  - 1994年1月，NBS做了最后一次评估，决定1998年12月以后不再作为加密标准

- **DES算法描述**
  - 分组大小: $2w$=64
  - 密钥大小: $|K|$=56

    子密钥: $|K_i|$=48
  - 轮数: h=16
  - 对明文作置换IP后开始第1次迭代
  - 第16次迭代后，交换左、右32bit数据，再作逆置换IP^{-1}，即得密文



3.2 数据加密标准(DES)

- **初始置换*IP***

  将64位明文打乱重新排列.

  设$x=x_1x_2\ldots x_{64}$，**则**$IP(x)=x_{58}x_{50}x_{42}\ldots x_{23}x_{15}x_7$

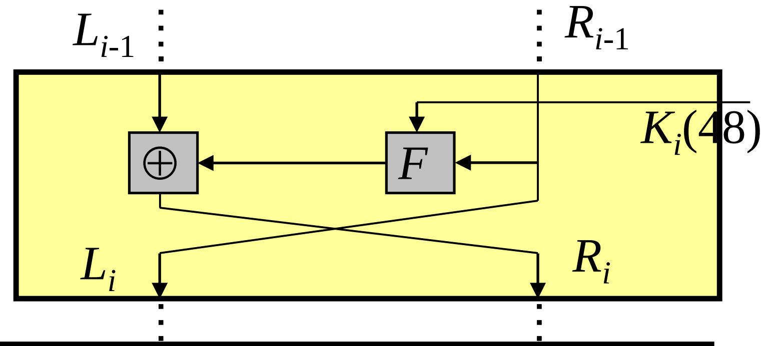
IP								IP^{-1}							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	8	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	6	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	13	52	20	60	28
59	51	43	35	27	19	11	3	35	3	43	11	51	19	59	27
61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

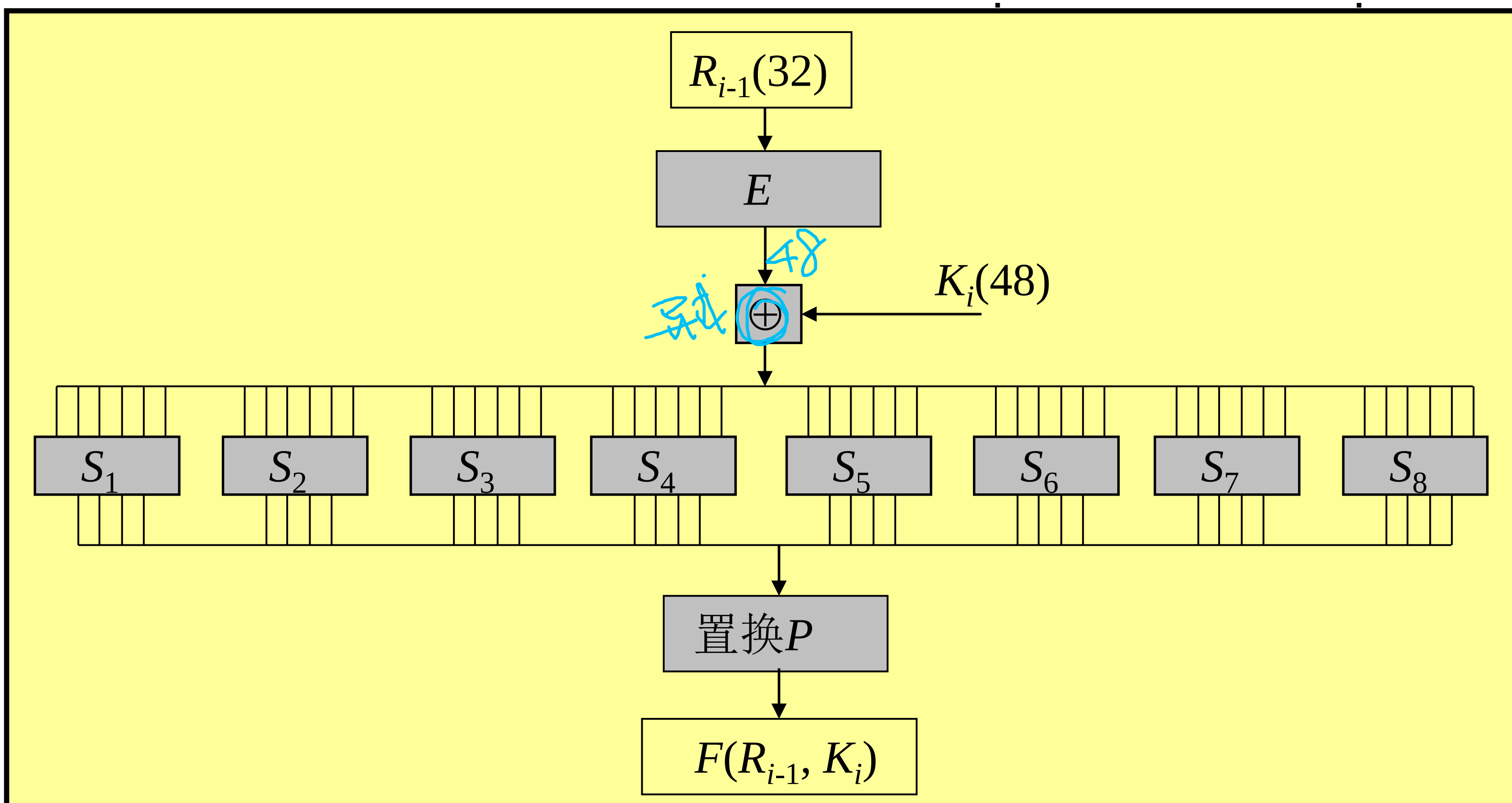
轮函数F的设计

- **轮函数F的结构**

  $F(R_{i-1}, K_i)$:

  $\{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$





扩展变换*E*

- **扩展变换*E*：将32位变为48位，扩展了16位**

扩展变换*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

S-盒

48 : 6×8
=
6位

- S_k: $\{0,1\}^6 \rightarrow \{0,1\}^4$ (k=1,2,…,8)
- 输入: $b_1\, b_2\, b_3\, b_4\, b_5\, b_6$, 用10进制表示:

  $(i)_{10}=b_1 b_6$ $(0 \le i \le 3)$, $(j)_{10}=b_2 b_3 b_4 b_5$ $(0 \le j \le 15)$
- 输出: S_k-盒的表中第i行j列位置元素(4位二进制)
- 例: 对于S_1，输入b=101011, 有i=11=3, j=0101=5,

  输出: $S_1(b)= S_1(3,5)=9=1001$. =4

S_k		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	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-盒

- **S-盒设计准则(1992年，IBM与NSA公布)**
  - S-盒的每一行都是整数0-15的一个置换;
  - 每个S-盒的输出都不是输入的线性或仿射函数;
  - 任意改变输入的一位,输出至少有2位发生变化;
  - 保持输入的1位不变,其余5位变化,则输出中的0和1的个数接近相等;
  - 对任何输入x, 有$S_k(x)$和$S_k(x \oplus 001100)$至少有2位不同;
  - 对任何输入x, $e,f \in \{0,1\}$, 有 $S_k(x) \neq S_k(x \oplus 11ef00)$.

置换P

- P：整数1-32的置换

置换P							
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

- 总结:

  轮函数$F(R_{i-1}, K_i)$运算过程

  $R_{i-1} \rightarrow e=E(R_{i-1}) \rightarrow c=e\oplus K_i \rightarrow s=S(c) \rightarrow P(s)$

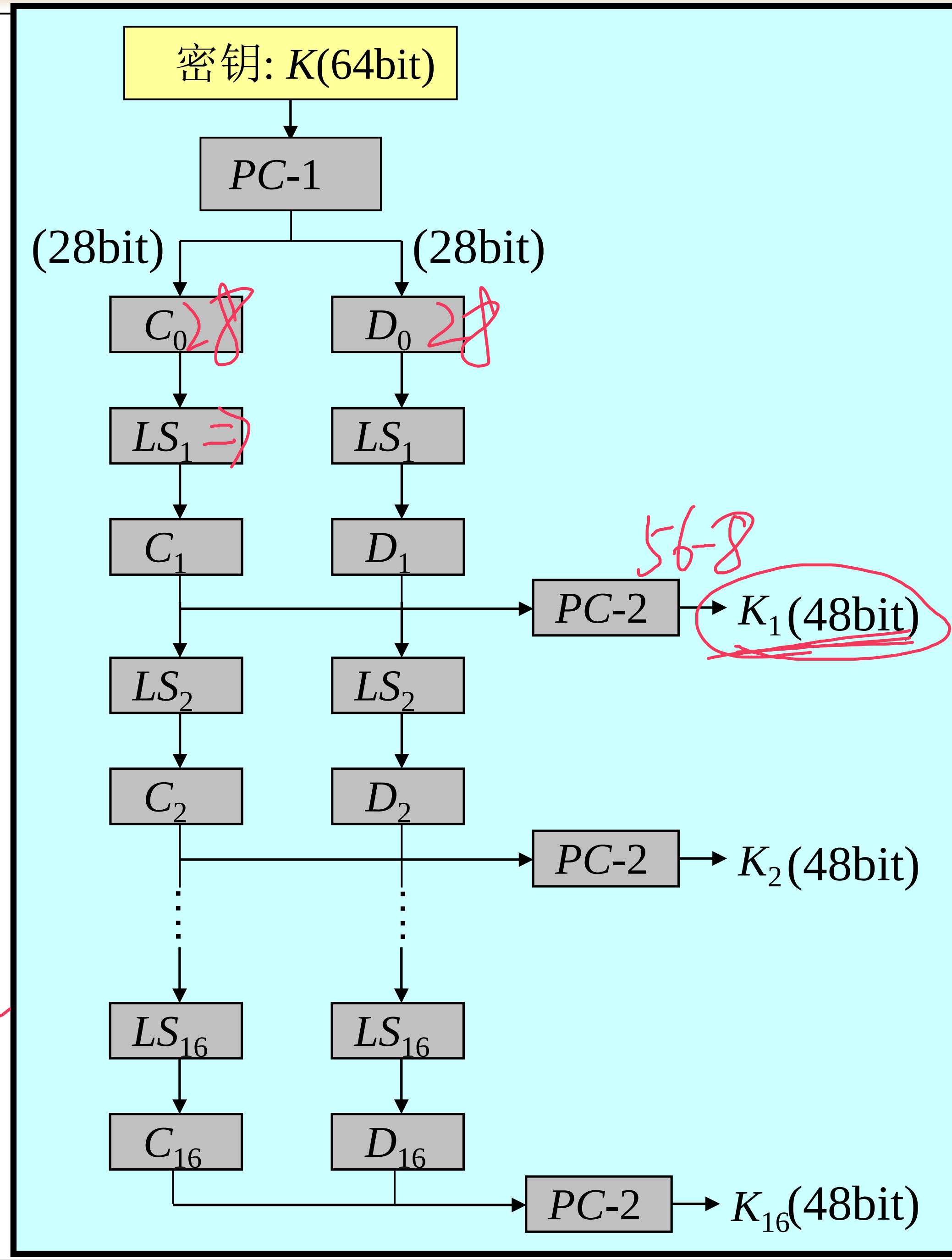
  $=F(R_{i-1}, K_i)=P(S(E(R_{i-1})\oplus K_i))$

DES子密钥生成算法

- **DES子密钥生成算法**

  由64位密钥K产生16个48位的子密钥:

  $K_1, K_2, \ldots, K_{16}$.



DES子密钥生成算法

- **置换选择1: *PC*-1**

  64位密钥K的第8, 16, 24,…,64位共8位是奇偶校验位, 其余56位作为密钥用. 选择K的第57,49,…位共28位作为密钥段C_0;选择K的第63,55,…位共28位作为密钥段D_0.

***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

DES子密钥生成算法

- **循环左移LS_i**

  将28位的密钥段作为C_i, D_i循环左移1或2位,左移位数由下表确定.

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

DES子密钥生成算法

- **置换选择2: *PC*-2**

  从56位密钥段$C_i||D_i$中选择48位作为子密钥K_i.

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	32

DES解密算法

- 与DES加密结构相同
- 子密钥使用次序相反: K_{16} $K_{15},\ldots,K_2,K_1$
- 输入：密文y
- 输出：明文x

DES算法的实现

- 硬件实现
  - 1984年，DES芯片每秒加密25.6万次
  - 1987年，DES芯片每秒加密51.2万次
  - 目前最快的DES芯片每秒加密1G比特（DEC:美国数字设备公司开发）
- 软件实现
  - 在IBM3090大型机上，DES软件实现每秒加密3.2万次
  - 在80486处理器，速度为66MHz，总线宽32位的微机上，DES软件实现每秒加密4.3万次

DES的安全性

- **互补性**
  - $y = \mathrm{DES}_K(x) \Rightarrow \overline{y} = \mathrm{DES}_{\overline{K}}(\overline{x})$
  - **在选择明文攻击时，只需实验2^{56}个密钥的一半2^{28}**
  - **不要使用互补密钥**
- **弱密钥K：$\mathrm{DES}_K(\mathrm{DES}_K(x))=x$.**

  DES至少有4个弱密钥，很可能不存在其它弱密钥.

  01 01 01 01 01 01 01 01;　　1F 1F 1F 1F 0F 0F 0F 0F;

  E0 E0 E0 E0 F1 F1 F1 F1;　FE FE FE FE FE FE FE FE
- **半弱密钥K：存在密钥K', 满足$\mathrm{DES}_K(\mathrm{DES}_{K'}(x))=x$.**

  DES至少有12个半弱密钥，很可能不存在其它半弱密钥.

  弱密钥与半弱密钥，能使二重DES加密复原!

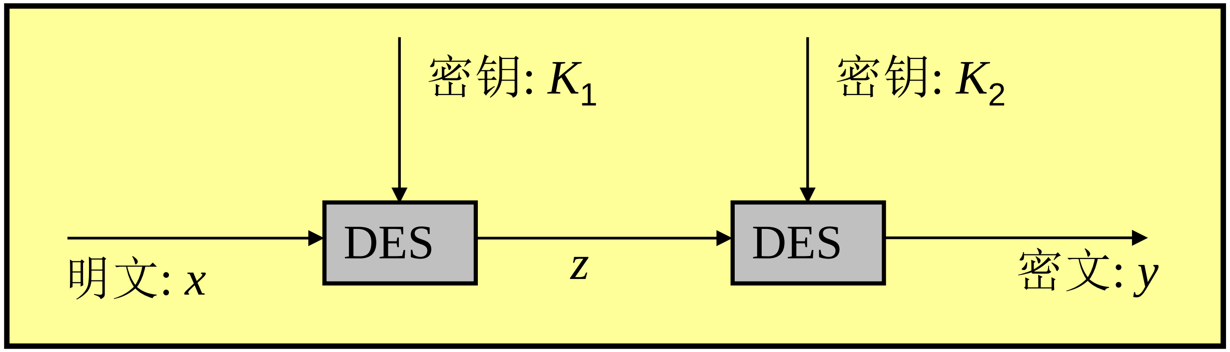
DES的安全性

- S-盒的设计
  - S-盒是DES的心脏
  - S-盒的设计原理尚未公开
  - 密码学家怀疑NSA设计S-盒时隐藏了“陷门”
- 密钥搜索机
  - 密钥量小：$2^{56} \approx 10^{17}$
  - 1997.1.28，美国RSA数据安全公司悬赏10000美元破译DES. 美国克罗拉多州的程序员Verser从1997.3.13起,用了96天,在Internet上数万名志愿者的协同下,于1997. 6. 17成功找到了DES的密钥,从而获得10000美元的奖金.
  - 1998年5月，美国电子边境基金学会(EFF)使用一台价值20万美元的计算机改装成专用密码机，用了56小时破译了56bit密钥的DES.

二重DES

- **二重DES的结构**

  $y=\mathrm{DES}_{K_2}(z)=\mathrm{DES}_{K_2}(\mathrm{DES}_{K_1}(x))$



- **二重DES的安全性**

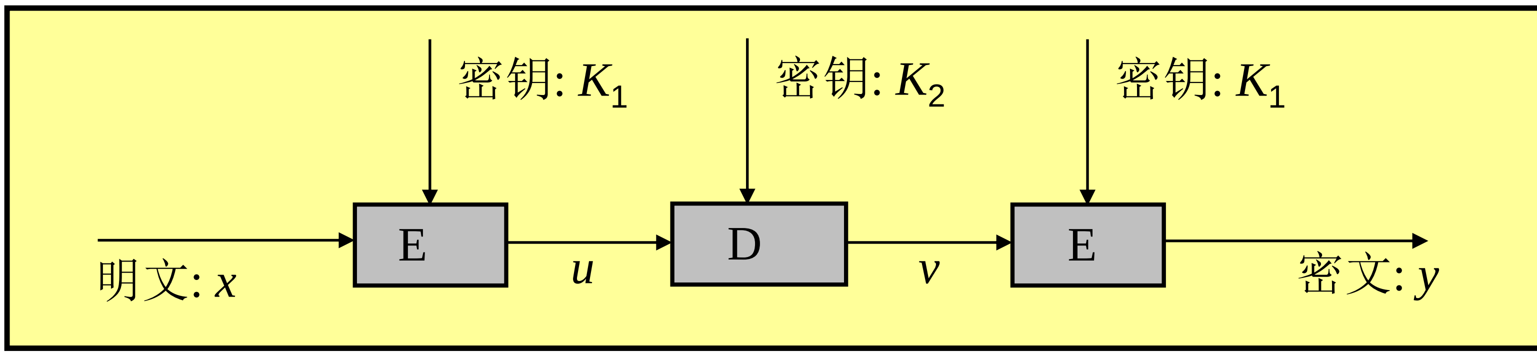
  密钥长度为112bit, 强度极大增加.

二个密钥的三重DES

- **二个密钥的三重DES结构**

  加密：E=DES，解密：D=DES^{-1}

  $y=\mathrm{E}_{K_1}[\mathrm{D}_{K_2}(\mathrm{E}_{K_1}(x))]$

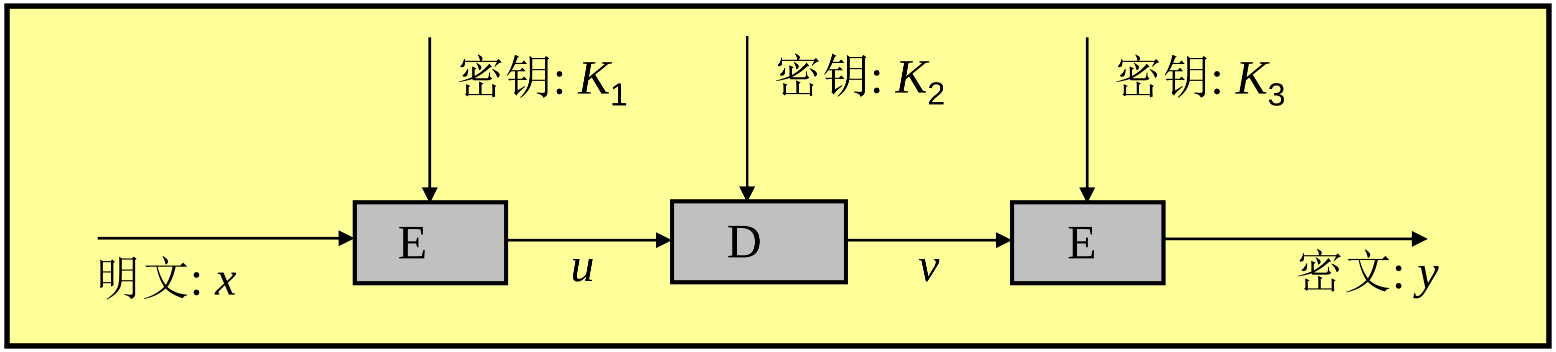


- **加密-解密-加密模式(EDE: encrypt-decrypt-encrypt)**
- **已被密钥管理标准ANS X.917和ISO 8732采用**

三个密钥的三重DES

- **三个密钥的三重DES结构**

  $y = E_{K_3}[D_{K_2}(E_{K_1}(x))]$



- **密钥长度为156bit, 强度进一步增加.**
- **已在Internet的许多应用(如PGP, S/MIME)中被采用**

第 3章　分组密码

- 3.1 分组密码概述
- 3.2 数据加密标准（DES）
- 3.3 国际数据加密算法（IDEA）
- 3.4 高级数据加密标准（AES）
- 3.5 分组密码工作模式

3.3 国际数据加密算法（IDEA）

- 1990年，Xuejia Lai(来学嘉，旅居瑞士中国学者)和J.L.Massey(国际著名密码学家)提出一个建议加密标准(PES: proposed encryption standard)
- 1991年，设计出改进型建议加密标准(IPES),能够抗击差分密码分析
- 1992年，改名为国际数据加密算法

  (IDEA: international data encryption algorithm)
- 是DES之后又一个成功的分组密码，已被用于Internet的E-mail加密系统PGP和其他加密系统
- 分组长度：64
- 密钥长度：128

Xuejia Lai(来学嘉)简介

- 国际著名密码学家
- 1982年获西安电子科技大学学士学位
- 1984年获该校应用数学硕士学位
- 1988年获瑞士苏黎世高工通信技术硕士学位
- 1992年获瑞士苏黎世高工技术科学博士学位
- 1994年加入瑞士r3安全工程中心，该中心于1998年6月成为Entrust(瑞士)公司
- 2001年加入瑞士S.W.I.S.中心
- 2004年到上海交大任教，兼任科学院研究生院名誉教授，西南交通大学顾问教授

- 设计了IDEA加密算法。对Hash 函数的分析和构造研究的成果得到国际上普遍应用,包括最近对MD4,SHA的碰撞攻击。在差分破译法的研究中，提出差分，高阶差分，马尔科夫密码的概念,用马尔科夫链理论将差分破译法公式化，使得推导差分密码分析复杂度的下界成为可能。设计了欧洲Eurochip 电话卡中的认证算法。 审核过欧洲银行Eurocard 智能卡系统的安全性。分析评估欧洲电讯标准局的专用密码。分析及改进付费电视系统中使用的密码及密钥管理系统。参与过中国金融认证中心的建设。主编了ISO-13888 不可抵赖标准、ISO-11770 密钥管理标准及ISO-18033 加密算法标准。并参与欧盟KRISIS，ICE-CAR和PKI Challenge 项目。现任2006亚密会(Asiacrypt)程序委员会主席, 中国密码学会常务理事。

Xuejia Lai(来学嘉)简介

国际著名密码学家来学嘉博士2005年6月在西南交通大学讲学

3.3 国际数据加密算法（IDEA）

运算‘+’：两个长度为16的比特串*x*和*y*。 *x*‘+’*y*表示*x*和*y*做逐位模2加运算（逐比特异或）。

运算“+”：将长度为16的比特串*x*和*y*看作是“≥0，且$<2^{16}$”的整数。 *x*“+”*y*表示*x*和*y*做模2^{16}加运算。

运算×：将长度为16的比特串*x*和*y*看作是“≥1，且$<2^{16}+1$”的整数。 其中将全0串看作是2^{16}。 $x \times y$表示*x*和*y*做模$2^{16}+1$乘运算。 (注意： $2^{16}+1$ 是素数)

3.3 国际数据加密算法（IDEA）

三种运算的代数结构

- 三种运算都是交换群运算（Abel群运算）。
- 运算‘+’的单位元是(0000000000000000)。
- 运算“+”的单位元是(0000000000000000)。
- 运算×的单位元是(0000000000000001)。
- 关于群运算‘+’，任何非单位元的阶数都是2。
- 关于群运算“+”，非单位元的阶数有2，2^2，2^3，…，2^{16}。
- 关于群运算×，非单位元的阶数有2，2^2，2^3，…，2^{16}。

3.3 国际数据加密算法（IDEA）

三种运算的密码学性质

- 设$z=x$‘+’y。则z的一个比特仅仅依赖于x和y在相同位置上的比特值，毫无扩散能力。
- 设$z=x$“+”y。则z的一个比特依赖于x和y在相同位置及以下位置上的比特值，具有单向扩散能力。
- 设$z=x\times y$。则z的一个比特依赖于x和y在所有位置上的比特值，具有双向扩散能力。
- ‘+’与“+”的结构差异很大；“+”与×虽然具有同构关系，但这种同构关系却表现为一种“离散对数”的关系。
- 将它们组合在一起使用，可以破坏任何一种群结构。

3.3 国际数据加密算法（IDEA）

轮函数 **轮函数为$M=F(m,z)$。其中m是明文，它是长度为64的比特串；M是密文，它是长度为64的比特串；z是密钥，它是长度为96的比特串。**

将明文m分为4个子块：$m=(m_1,m_2,m_3,m_4)$，其中每个子块长度为16。

将密文M分为4个子块: $M=(M_1,M_2,M_3,M_4)$, 其中每个子块长度为16。

将密钥z分为6个子块：$z=(z_1,z_2,z_3,z_4,z_5,z_6)$，其中每个子块长度为16。

三种运算‘+’、“+”、×分别为前面所述。

3.3 国际数据加密算法（IDEA）

轮函数算法描述如下：

（1） $(m_1,m_2,m_3,m_4)(\times,“+”,“+”,\times)\ (z_1,z_2,z_3,z_4)=(a, b, c, d)$。

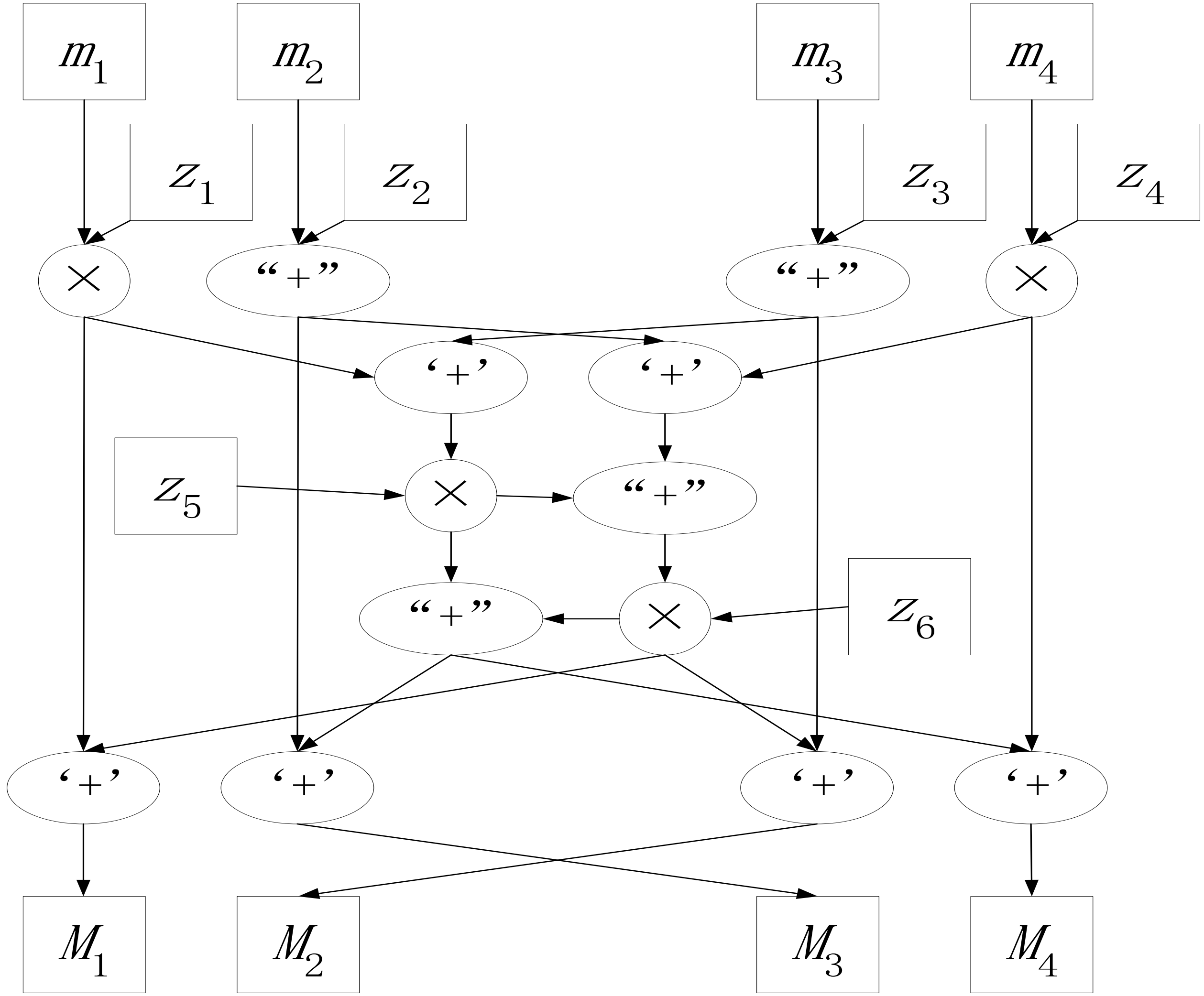
（2） $(a‘+’c, b‘+’d)=(e, f)$。

（3） $((e\times z_5)“+”f\ )\times z_6=u$， $u“+”(e\times z_5)=v$。

（4） $(a, b, c, d)(‘+’,‘+’,‘+’,‘+’)\ (u,v,u,v)=(w_1,w_2,w_3,w_4)$。

（5） $(w_1,w_3,w_2,w_4)=(M_1, M_2, M_3, M_4)$。

3.3 国际数据加密算法（IDEA）



3.3 国际数据加密算法（IDEA）

（1）$(m_1,m_2,m_3,m_4)(\times,\text{“}+\text{”},\text{“}+\text{”},\times)(z_1,z_2,z_3,z_4)=(a, b, c, d)$。

轮函数的第（1）步称为群加密，使用密钥(z_1,z_2,z_3,z_4)，将(m_1,m_2,m_3,m_4)变为(a,b,c,d)。

因为 $(a, b, c, d)(\times,\text{“}+\text{”},\text{“}+\text{”},\times)(z_1^{-1}, -z_2, -z_3, z_4^{-1})=(m_1,m_2,m_3,m_4)$，其中$z_1^{-1}$是$z_1$的×逆元，$-z_2$是$z_2$的“+”逆元，$-z_3$是$z_3$的“+”逆元，$z_4^{-1}$是$z_4$的×逆元。

结论：群加密的逆变换也是群加密，只不过所用的密钥不同。

3.3 国际数据加密算法（IDEA）

（2）$(a`+\text{'}c, b`+\text{'}d)=(e, f)$。

（3）$((e\times z_5)``+\text{''}f)\times z_6=u$，$u``+\text{''}(e\times z_5)=v$。

（4）$(a, b, c, d)(`+\text{'},`+\text{'},`+\text{'},`+\text{'})\ (u,v,u,v)=(w_1,w_2,w_3,w_4)$。

轮函数的第（2）（3）（4）步称为<u>MA变换</u>，使用密钥 (z_5,z_6)，将(a, b, c, d)变为(w_1,w_2,w_3,w_4)。

(2') $(w_1`+\text{'}w_3, w_2`+\text{'}w_4)=(e, f)$，

(3') $(((e\times z_5)``+\text{''}f)\times z_6)=u$，$u``+\text{''}(e\times z_5)=v$，

(4') $(w_1,w_2,w_3,w_4)(`+\text{'},`+\text{'},`+\text{'},`+\text{'})\ (u,v,u,v)=(a, b, c, d)$.

3.3 国际数据加密算法（IDEA）

所以第（2）（3）（4）步的逆运算仍然是MA变换，所使用的密钥仍然是(z_5,z_6)，将(w_1,w_2,w_3,w_4)变为(a, b, c, d)。

结论：MA变换是对合变换（自反变换）；

3.3 国际数据加密算法（IDEA）

（5）$(w_1,w_3,w_2,w_4)=(M_1, M_2, M_3, M_4)$。

轮函数的第（5）步称为块置换，不使用密钥，仅仅把第二子块与第三子块对调，将(w_1,w_2,w_3,w_4) 变为(M_1, M_2, M_3, M_4)。

结论： 块置换是对合变换；

综上所述，有：

- 群加密的逆运算还是群加密，只是使用的密钥不同；
- MA变换的逆运算还是MA变换，使用的密钥也相同；
- 块置换的逆运算还是块置换。

3.3 国际数据加密算法（IDEA）

完整的加密算法：8轮迭代

分组密码IDEA的完整加密算法是连续8次使用轮函数，不过第8轮与前7轮有所不同。前7轮是普通轮，轮函数的运算步骤如前所述为：

群加密→MA变换→块置换。

第8轮是特殊轮，轮函数的运算步骤为：

群加密→MA变换→群加密。

因此，分组密码IDEA的完整加密算法如下：

3.3 国际数据加密算法（IDEA）

群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→块置换→
群加密→MA变换→群加密。

3.3 国际数据加密算法（IDEA）

解密算法

考察在加密算法中的“块置换→群加密”组合结构。设该组合结构的输入为(g_1, g_2, g_3, g_4)，群加密的密钥为(q_1, q_2, q_3, q_4)，输出为(r_1, r_2, r_3, r_4)，则组合结构的算法描述如下：

$(g_1,g_3,g_2,g_4)= (h_1, h_2, h_3, h_4)$；

$(h_1, h_2, h_3, h_4)(\times,“+”,“+”,\times)(q_1, q_2, q_3, q_4)$

$=(r_1, r_2, r_3, r_4)$。

3.3 国际数据加密算法（IDEA）

这个结构还可以描述如下：

(g_1,g_2,g_3,g_4) (×,“+”,“+”,×)(q_1, q_3, q_2, q_4)

$=(r_1, r_3, r_2, r_4)= (h_1, h_2, h_3, h_4)$；

$(h_1, h_3, h_2, h_4) =(r_1, r_2, r_3, r_4)$。

这就是说，“块置换→群加密”组合结构中的块置换和群加密的顺序可以颠倒，描述为“群加密→块置换”。只不过在颠倒顺序时，密钥四个子块中的第二、第三子块交换位置，从(q_1, q_2, q_3, q_4) 变为(q_1, q_3, q_2, q_4) 。

3.3 国际数据加密算法（IDEA）

再注意到块置换和群加密各自的逆运算也是它们自己。

因此，结论如下：

- “块置换→群加密”组合结构的逆运算还是“块置换→群加密”组合结构。
- 如果“块置换→群加密”的群加密的密钥是(q_1, q_2, q_3, q_4)，且逆运算也采用“块置换→群加密”顺序，则逆运算群加密的密钥就是

$$(q_1^{-1}, -q_3, -q_2, q_4^{-1})。$$

3.3 国际数据加密算法（IDEA）

IDEA的解密算法是加密算法的逆运算。从前面的叙述，我们已经得到如下事实：

- 群加密的逆运算还是群加密；
- MA变换的逆运算还是MA变换；
- 块置换的逆运算还是块置换；
- “块置换→群加密”组合结构的逆运算还是“块置换→群加密”组合结构。

因此， IDEA的解密算法与加密算法完全相同，仅仅是密钥有所不同。以下是加解密算法密钥对照。

3.3 国际数据加密算法（IDEA）

加密算法	密钥
第一轮	$(z_{11}, z_{12}, z_{13}, z_{14})$；$(z_{15}, z_{16})$
第二轮	$(z_{21}, z_{22}, z_{23}, z_{24})$；$(z_{25}, z_{26})$
第三轮	$(z_{31}, z_{32}, z_{33}, z_{34})$；$(z_{35}, z_{36})$
第四轮	$(z_{41}, z_{42}, z_{43}, z_{44})$；$(z_{45}, z_{46})$
第五轮	$(z_{51}, z_{52}, z_{53}, z_{54})$；$(z_{55}, z_{56})$
第六轮	$(z_{61}, z_{62}, z_{63}, z_{64})$；$(z_{65}, z_{66})$
第七轮	$(z_{71}, z_{72}, z_{73}, z_{74})$；$(z_{75}, z_{76})$
第八轮	$(z_{81}, z_{82}, z_{83}, z_{84})$；$(z_{85}, z_{86})$； $(z_{91}, z_{92}, z_{93}, z_{94})$

3.3 国际数据加密算法（IDEA）

解密算法	密钥
第一轮	$(z_{91}^{-1}, -z_{92}, -z_{93}, z_{94}^{-1})$； (z_{85}, z_{86})
第二轮	$(z_{81}^{-1}, -z_{83}, -z_{82}, z_{84}^{-1})$； (z_{75}, z_{76})
第三轮	$(z_{71}^{-1}, -z_{73}, -z_{72}, z_{74}^{-1})$； (z_{65}, z_{66})
第四轮	$(z_{61}^{-1}, -z_{63}, -z_{62}, z_{64}^{-1})$； (z_{55}, z_{56})
第五轮	$(z_{51}^{-1}, -z_{53}, -z_{52}, z_{54}^{-1})$； (z_{45}, z_{46})
第六轮	$(z_{41}^{-1}, -z_{43}, -z_{42}, z_{44}^{-1})$； (z_{35}, z_{36})
第七轮	$(z_{31}^{-1}, -z_{33}, -z_{32}, z_{34}^{-1})$； (z_{25}, z_{26})
第八轮	$(z_{21}^{-1}, -z_{23}, -z_{22}, z_{24}^{-1})$； (z_{15}, z_{16})； $(z_{11}^{-1}, -z_{12}, -z_{13}, z_{14}^{-1})$

3.3 国际数据加密算法（IDEA）

密钥扩展算法

IDEA加密算法中所使用的密钥共有52个子块，即加密密钥长度为16×52=832（比特）。用户密钥实际上只有128 （比特），因此需要一个密钥扩展算法。密钥扩展算法如下。

- 将128 比特的用户密钥分为8个子块，作为加密密钥的第一个“8个子块”；
- 将128 比特密钥循环左移25位，再分为8个子块，作为加密密钥的第二个“8个子块”；

3.3 国际数据加密算法（IDEA）

密钥扩展算法

- 将128 比特密钥再次循环左移25位，再分为8个子块，作为加密密钥的第三个“8个子块”；
- …；直到有52个子块作为加密密钥。

IDEA特性

- IDEA算法软、硬件实现容易，速度快。软件实现比DES快两倍
- 安全性好：用穷举攻击要试探$2^{128}=10^{38}$个密钥，如用每秒运行100万次的计算机进行搜索，大约需要10^{13}年.
- IDEA能抵抗差分攻击和线性攻击
- IDEA的安全缺陷：存在大量的弱密钥
- IDEA的设计适合于16位CPU, 对于32CPU实现不太方便

第3章　习 题

1. 设 DES 算法中，明文 M 和密钥 K 分别为

M=0011 1000 1100 0100 1011 1000 0100 0011　　1101 0101 0010 0011 1001 1110 0101 1110

K=1010 1001 0011 0101 1010 1100 1001 1011　　1001 1101 0011 1110 1101 0101 1100 0011

求 L_1 和 R_2。

2. 设 DES 算法中 S_4 盒的输入为 010101，求其输出。

3. 在图 3.18 所示的 IDEA 算法 MA 部件中，已知输入数据为

X_1= 0000100110010011　　X_2= 1010001001001011

K_1= 1100001000100110　　K_2= 1000011010000110

求输出求 Y_1 和 Y_2。

4. 在 IDEA 算法中，已知明文 M 和密钥 K 分别为

M=10101010 11100110 01010101 00001111 11001100 00110011 10011001 01100110

K= 00000000 11111111 00000000 11111111 11111111 00001111 11110000 11111111

00001111 11110000 11111111 00000000 00001111 11111111 11110000 00001111

求第 1 轮的输出和第 2 轮的输入。

第 3章　分组密码

- 3.1 分组密码概述
- 3.2 数据加密标准（DES）
- 3.3 国际数据加密算法（IDEA）
- 3.4 高级数据加密标准（AES）
- 3.5 分组密码工作模式

3.4 高级数据加密标准（AES）

- 1997年4月5日,美国NIST(美国国家标准和技术协会)开始征集和评估新的高级数据加密标准AES (advanced encryption standard)
- 1998年,NIST从21个提交算法中选出15个作为AES候选算法
- 1999年,NIST从15个候选算法中选出5个作为新一轮评估，让社会公开评价
- 2000年10月, NIST宣布Rijndael算法作为AES算法
  - Rijndael（读音: rain doll）算法由比利时密码专家Joan Daeman博士和Vincent Rijmen博士后开发
- 2001年11月26日, NIST宣布AES 为美国政府的新加密标准
- 2002年5月26日正式生效

有限域GF(2^8)

- **GF(2^8)的元素**
  - 8bit的字节=8维二元向量

    $b=b_7b_6b_5b_4b_3b_2b_1b_0=(b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$.
  - 次数不超过8的二元多项式

    $b \Rightarrow b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$.
  - 例: $b=10011011=(1,0,0,1,1,0,1,1) \Rightarrow b(x)=x^7+x^4+x^3+x+1$.
- **GF(2^8)的加法**$\oplus$: 对应位mod 2相加
  - 例: $a=01101111 \Rightarrow a(x)=x^6+x^5+x^3+x^2+x+1$,

    $a \oplus b=01101111 \oplus 10011011=11110100$,

    $a(x) \oplus b(x)= x^6+x^5+x^3++x^2+x+1 \oplus x^7+x^4+x^3+x+1$

    $= x^7+ x^6+x^5+x^4+x^2$.

有限域GF(2^8)

- **GF(2^8)的乘法“·”**
  - 模多项式: $m(x)=x^8+x^4+x^3+x+1$.
  - 两个多项式相乘: 将积按$m(x)$取模
  - 乘法逆元: 如果$a(x)$与$b(x)$满足: $a(x)\cdot b(x)=1 \pmod{m(x)}$

    则称$b(x)$是$a(x)$的乘法逆元，记为$b(x)=a(x)^{-1}$.
  - 例: 设$a(x)=x^6+x^4+x^2+x+1$, $b(x)=x^7+x+1$,则

    $a(x)\cdot b(x)=(x^6+x^4+x^2+x+1)(x^7+x+1)$

    $=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1=x^7+x^6+1 \pmod{m(x)}$
  - 又例: 用16进制表示字节

    $(B\underline{2})\cdot(8\underline{4})=(1100\ \underline{0010})\cdot(1000\ \underline{0100})$

    $=(x^7+x^6+x)\cdot(x^7+x^2)=x^{14}+x^{13}+x^9+x^3$

    $=x^7+x^6+x^5+x^3+1 \pmod{m(x)}=1110\ 1001=(D9)$.

- 扩展欧几里德算法伪代码
- r1←a; r2←b;s1←1; s2←0; t1←0; t2←1;
- while(r2>0)
- { q=r1/r2;
- r=r1-q×r2; r1← r2; r2←r;
- s=s1-q×s2; s1← s2; s2←s;
- t=t1-q×t2; t1← t2; t2←t;
- }
- gcd(a, b)←r1, s ←s1, t ←t1;
- 满足gcd(a, b)=s×a+t×b

有限域GF(2^8)

$$x^6+x^5+x^2+x+1$$

$$x^8+x^4+x^3+x+1 \,\overline{\big)\, x^{14}+x^{13}+x^9+x^3}$$

$$\underline{x^{14}+x^{10}+x^9+x^7+x^6}$$

$$x^{13}+x^{10}+x^7+x^6+x^3$$

$$\underline{x^{13}+x^9+x^8+x^6+x^5}$$

$$x^{10}+x^9+x^8+x^7+x^5+x^3$$

$$\underline{x^{10}+x^6+x^5+x^3+x^2}$$

$$x^9+x^8+x^7+x^6+x^2$$

$$\underline{x^9+x^5+x^4+x^2+x}$$

$$x^8+x^7+x^6+x^5+x^4+x$$

$$\underline{x^8+x^4+x^3+x+1}$$

$$x^7+x^6+x^5+x^3+1$$

有限域GF(2^8)

- **倍乘函数(x乘)**

设$b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0= b_7b_6b_5b_4b_3b_2b_1b_0$,

则 $x\cdot b(x)=b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x^1$

$x\cdot b(x)= b_6b_5b_4(b_3+b_7)\ (b_2+b_7)b_1(b_0+b_7)(0+b_7)$

$= b_6b_5b_4b_3b_2b_1b_00+000b_7b_70b_7b_7$

  - 记$x\cdot b(x)=\mathrm{xtime}(b(x))=\mathrm{xtime}(b)$
  - xtime是AES的基本运算,已做成专用芯片, 任意常数乘法都可以用xtime来实现

有限域GF(2^8)

- **倍乘函数(x乘)**
  - $(02)\cdot b(x)=x\cdot b(x)=\mathrm{xtime}(b)$,

    $(04)\cdot b(x)=x^2\cdot b(x)=x\cdot(x\cdot b(x))=x\cdot(\mathrm{xtime}(b))=\mathrm{xtime}(\mathrm{xtime}(b))$,

    $(08)\cdot b(x)=x\cdot(x^2\cdot b(x))=\mathrm{xtime}(\mathrm{xtime}(\mathrm{xtime}(b)))$.
  - **例: 计算57·13**

    $13=0001\ 0011=x^4+x+1$,

    $x\cdot 57=\mathrm{xtime}(57)=\mathrm{AE}$,

    $x^4\cdot 57=\mathrm{xtime}(\mathrm{xtime}(\mathrm{xtime}(\mathrm{xtime}(57))))$

    $=\mathrm{xtime}(\mathrm{xtime}(\mathrm{xtime}(\mathrm{AE})))$

    $=\mathrm{xtime}(\mathrm{xtime}(47))$

    $=\mathrm{xtime}(8\mathrm{E})=07$.

    $\Rightarrow 57\cdot 13=57\oplus x\cdot 57\oplus x^4\cdot 57=57\oplus \mathrm{AE}\oplus 07=\mathrm{FE}$.

有限域GF(2^8)

- **GF(2^8)上次数小于4的多项式全体**
  - 形式: $a(x)=a_3x^3+a_2x^2+a_1x+a_0$ ($a_3,a_2,a_1,a_0\in$ GF(2^8)), 用于表示4个字节!
  - 两个多项式相加: 对应系数相加
  - 模多项式: $M(x)=x^4+1$.
  - 两个多项式相乘⊗: 将积按$M(x)$取模

  设 $a(x)=a_3x^3+a_2x^2+a_1x+a_0$, $b(x)=b_3x^3+b_2x^2+b_1x+b_0$

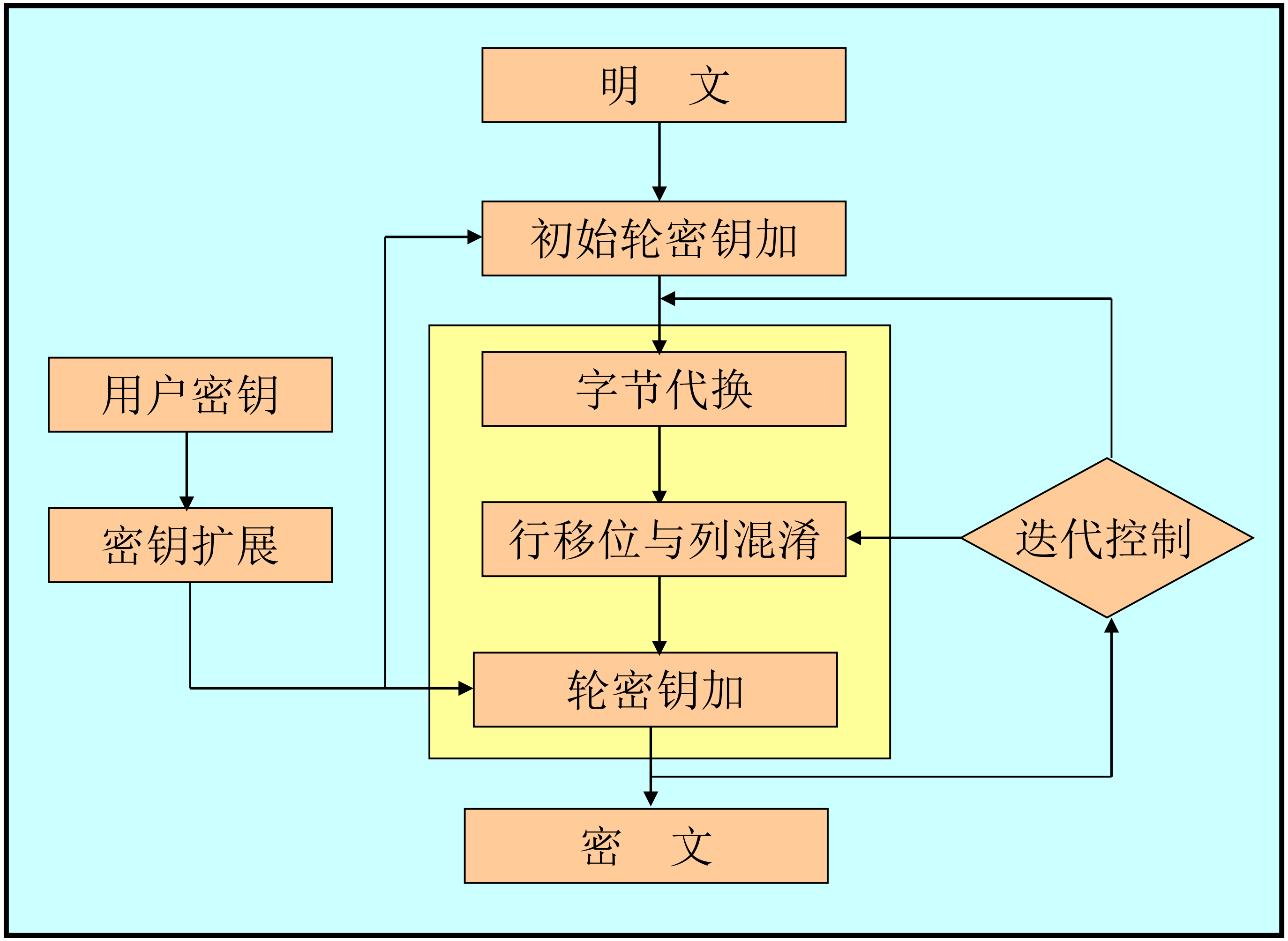
  则有: $a(x)\otimes b(x)=c_3x^3+c_2x^2+c_1x+c_0 \pmod{M(x)}$

$$c_0=a_0\cdot b_0\oplus a_3\cdot b_1\oplus a_2\cdot b_2\oplus a_1\cdot b_3,$$
$$c_1=a_1\cdot b_0\oplus a_0\cdot b_1\oplus a_3\cdot b_2\oplus a_2\cdot b_3,$$
$$c_2=a_2\cdot b_0\oplus a_1\cdot b_1\oplus a_0\cdot b_2\oplus a_3\cdot b_3,$$
$$c_3=a_3\cdot b_0\oplus a_2\cdot b_1\oplus a_1\cdot b_2\oplus a_0\cdot b_3.$$

  注意: 这是GF(2^8)中的运算.

AES加密算法

- **AES加密算法结构**



四、分组密码Rijndael

Rijndael算法的总体结构：

初始变换——密钥加

第一轮： 字节替换→行移位→列混合→密钥加

第二轮： 字节替换→行移位→列混合→密钥加

第三轮： 字节替换→行移位→列混合→密钥加

第四轮： 字节替换→行移位→列混合→密钥加

…………………………………………

………

第N_r-1轮：字节替换→行移位→列混合→密钥加

第N_r轮： 字节替换→行移位→密钥加

四、分组密码Rijndael

Rijndael算法的总体结构也可以看做如下结构：

初始变换——密钥加

第一轮： 字节替换→行移位→列混合→密钥加

第二轮： 字节替换→行移位→列混合→密钥加

第三轮： 字节替换→行移位→列混合→密钥加

第四轮： 字节替换→行移位→列混合→密钥加

……………………………………………………………

…

第N_r-1轮：字节替换→行移位→列混合→密钥加

第N_r轮： 字节替换→行移位

末尾变换——密钥加

AES加密算法

- 数据长度可变
  - 明文、密文分组长度: l_m=128,192,256
  - 密钥长度: l_k=128,192,256
- 加密过程的中间结果称为“状态(state)”
  - 将每次变换的状态以字节为单位表示成一个4行N_b列的矩阵. $N_b=l_m/32=4,6,8$.
  - l_m=192, N_b=6时的状态表示

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}

AES加密算法

- **密钥k的表示**
  - **将密钥k以字节为单位表示成一个4行N_k列的矩阵.** $N_k=l_k/32=4,6,8$.
  - $l_k=128$, $N_k=4$**时的密钥表示**

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}

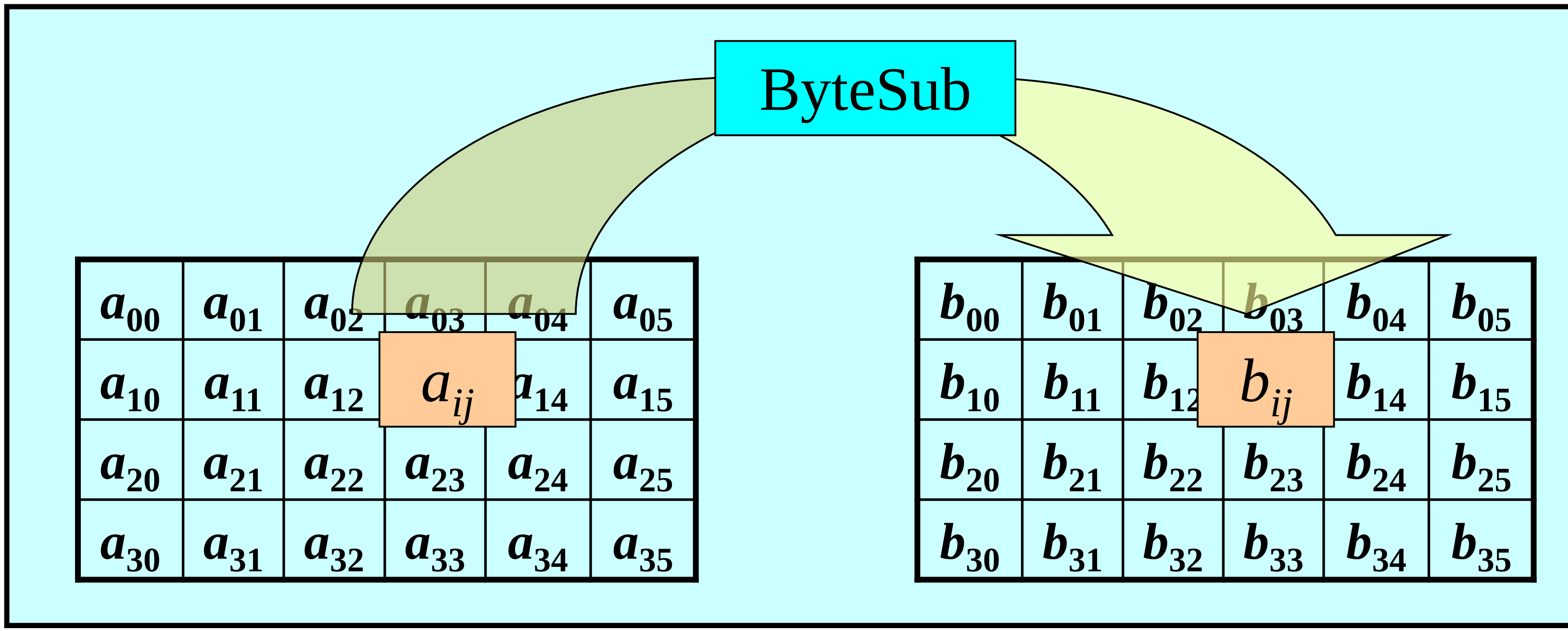
- **迭代轮数N_r.**

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

AES的轮函数

- **字节代换: ByteSub**
  - **对状态的每个字节独立进行代换，是字节的非线性变换，也称为<u>S盒变换</u>。设**

$$\text{ByteSub}(a_{ij})=b_{ij}.$$



AES的轮函数

- **字节代换：ByteSub(a_{ij})=b_{ij}.**

$$a_{ij} \xrightarrow{\text{求逆}} a_{ij}^{-1} \xrightarrow{\text{仿射变换}} b_{ij}.$$

- **第1步: 求乘法逆元(GF(2^8)中的乘法·)， ‘00’的逆元为自己 ‘00’**

  $a_{ij} \rightarrow a_{ij}^{-1}$.

- **第2步: 对a_{ij}^{-1}作仿射变换.**

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

AES的轮函数

- **行移位: ShiftRow**
  - **将状态矩阵的每行进行循环左移:**
    第0行不移;第i行循环左移C_i个字节(i =1,2,3).

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

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}	$\xrightarrow{0位}$	a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	$\xrightarrow{1位}$	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{10}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	$\xrightarrow{2位}$	a_{22}	a_{23}	a_{24}	a_{25}	a_{20}	a_{21}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	$\xrightarrow{3位}$	a_{33}	a_{34}	a_{35}	a_{30}	a_{31}	a_{31}

AES的轮函数

- **列混合: MixColumn**
  - **将状态矩阵每一列的4个字节表示成一个3次多项式，再与多项式 $c(x)$ 相乘.**

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

- 例如: 对于第2列,

  $a(x)=a_{31}x^3+a_{21}x^2+a_{11}x+a_{01}$,

  则 MixColumn($a(x)$)

  $=a(x)\otimes c(x)$ (mod $M(x)$)

  $=b_{31}x^3+b_{21}x^2+b_{11}x+b_{01}$.

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}

$\Downarrow$

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

AES的轮函数

- **轮密钥加: AddRoundKey(State, RoundKey)**
  - 将状态矩阵与子密钥矩阵的对应字节进行逐比特异或

    AddRoundKey(a_{ij}, k_{ij})=$a_{ij} \oplus k_{ij}$.
  - 例： $a_{21} \oplus k_{21}=b_{21}$.

$$
\begin{bmatrix}
a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\
a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\
a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\
a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35}
\end{bmatrix}
\oplus
\begin{bmatrix}
k_{00} & k_{01} & k_{02} & k_{03} & k_{04} & k_{05} \\
k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\
k_{20} & k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \\
k_{30} & k_{31} & k_{32} & k_{33} & k_{34} & k_{35}
\end{bmatrix}
=
\begin{bmatrix}
b_{00} & b_{01} & b_{02} & b_{03} & b_{04} & b_{05} \\
b_{10} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\
b_{20} & b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\
b_{30} & b_{31} & b_{32} & b_{33} & b_{34} & b_{35}
\end{bmatrix}
$$

AES的轮函数

- **轮函数的伪C代码**

第1到$N_r - 1$轮

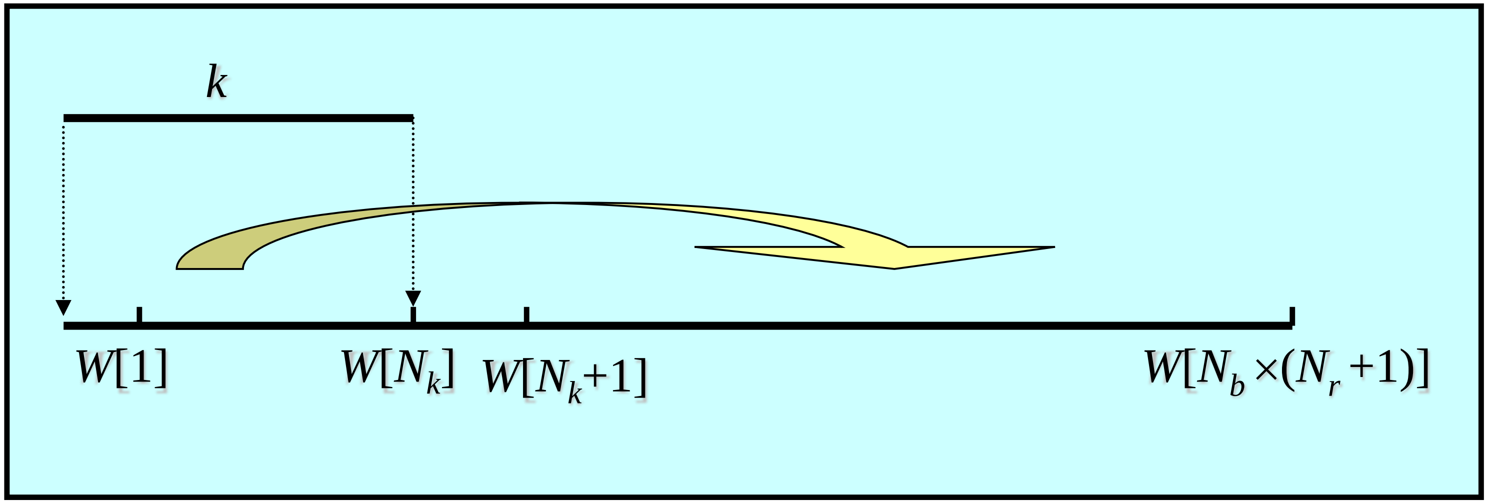
```
Round(State, RoundKey)
{
   ByteSub(State);
   ShiftRow(State);
   MixColumn(State);
   AddRoundKey(State,
               RoundKey);
}
```

第N_r轮(最后轮)

```
FinalRound(State, RoundKey)
{
   ByteSub(State);
   ShiftRow(State);
   AddRoundKey(State,
               RoundKey);
}
```

AES的轮密钥生成算法

- **轮密钥生成算法=密钥扩展+轮密钥选取**
- **密钥扩展: 将密钥k扩展为扩展密钥$W[N_b\times(N_r+1)]$**
  - W是以4个字节为元素的一维数组,共有$N_b\times(N_r+1)$个元素
  - W的前N_k个元素正好是密钥k, 其他元素由扩展算法求出



AES的轮密钥生成算法

◆$N_k \le 6$时的扩展算法

Key: 密钥

"$i \% N_k = =0$": I 整除N_k

^: 异或

RotByte: 循环左移一个字节

SubByte: 对每个字节作代换

(ByteSub)

Rcon[i]=(RC[i],'00','00','00')

RC[1]=1,

RC[i]=x·RC[i-1]= x^{i-1}.

```
KeyExpansion(byte Key[4*Nk], W[Nb*(Nr+1)])
{
  for (i=0; i< Nk; i+ +)
      W[i]=(Key[4*i], Key[4*i+1], Key[4*i+2],
                Key[4*i+3] );
  for (i= Nk; i < Nb*(Nr+1); i+ +)
  {
    temp=W[i-1];
    if  (i %Nk = =0)
        temp=SubByte(RotByte(temp))^Rcon[i/Nk];
        W[i]=W[i-Nk]^temp;
  }
}
```

AES的轮密钥生成算法

◆ N_k>6时的扩展算法

```
KeyExpansion(byte Key[4*Nk], W[Nb*(Nr+1)])
{
    for (i=0; i< Nk; i+ +)
         W[i]=(Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3] );
   for (i= Nk; i < Nb*(Nr+1); i+ +)
    {
      temp=W[i-1];
      if  (i %Nk = =0)
           temp=SubByte(RotByte(temp))^Rcon[i/Nk];
       Else if (i %Nk = =4)
            temp=SubByte(temp);
       W[i]=W[i-Nk]^temp;
   }
}
```

AES的轮密钥生成算法

- **轮密钥选取**

  第i个轮密钥由$W[N_b \times i)]$到$W[N_b \times (i+1)]$给出.

AES加密算法的伪C代码

```
Rijndael(State, CipherKey)
{
    KeyExpansion(CipherKey, ExpandedKey);
    AddRoundKey(State, ExpandedKey);
    for (i=1; i< Nr; i+ +) Round(State, ExpandedKey[Nb*i]);
    FinalRound(State, ExpandedKey[Nb*Nr])
}
```

AES的解密算法

- **字节代换ByteSub的逆变换InvByteSub**
  - 首先作仿射变换的逆变换
  - 再求每个字节在GF(2^8)中的逆元

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

AES的解密算法

- **行移位ShiftRow的逆变换InvShiftRow**

  对状态矩阵的第i行循环左移N_b-C_i 个字节(i =1,2,3).

- **列混合MixColumn的逆变换InvMixColumn**

  将状态矩阵每一列的4个字节表示成一个3次多项式，再与多项式$d(x)$相乘.

$$d(x)=(0B)x^3+(0D)x^2+(09)x+(0E).$$

- **轮密钥加AddRoundKey的逆变换是其自身**

AES的解密算法

- **轮函数的逆变换**

第1到*Nr*– 1轮

```
InvRound(State, InvRoundKey)
{
   InvByteSub(State);
   InvShiftRow(State);
   InvMixColunm(State);
   AddRoundKey(State, InvRoundKey); }
```

第*Nr*轮

```
InvFinalRound(State, InvRoundKey)
{
   InvByteSub(State);
   InvShiftRow(State);
   AddRoundKey(State, InvRoundKey); }
```

AES的解密算法

- **解密密钥生成算法: InvKeyExpansion**

  设加密算法的初始密钥,第1轮,第2轮,…,第N_r轮子密钥依次为:

  $k(0)$, $k(1)$, $k(2)$,…,$k(N_r\text{-}1)$, $k(N_r)$.

  则解密算法的初始密钥,第1轮,第2轮,…,第N_r轮子密钥依次为:

  $k(N_r)$, InvMixColumn($k(N_r\text{-}1)$), InvMixColumn($k(N_r\text{-}2)$), …, InvMixColumn($k(2)$), InvMixColumn($k(1)$), $k(0)$.

- **AES解密算法的伪C代码**

```
InvRijndael(State, CipherKey)
{
   InvKeyExpansion(CipherKey, InvExpandedKey);
   InvAddRoundKey(State, InvExpandedKey);
    for (i=0; i< Nr; i+ +) InvRound(State, InvExpandedKey[Nb*i]);
    InvFinalRound(State, InvExpandedKey[Nb*Nr])
}
```

第 3章　分组密码

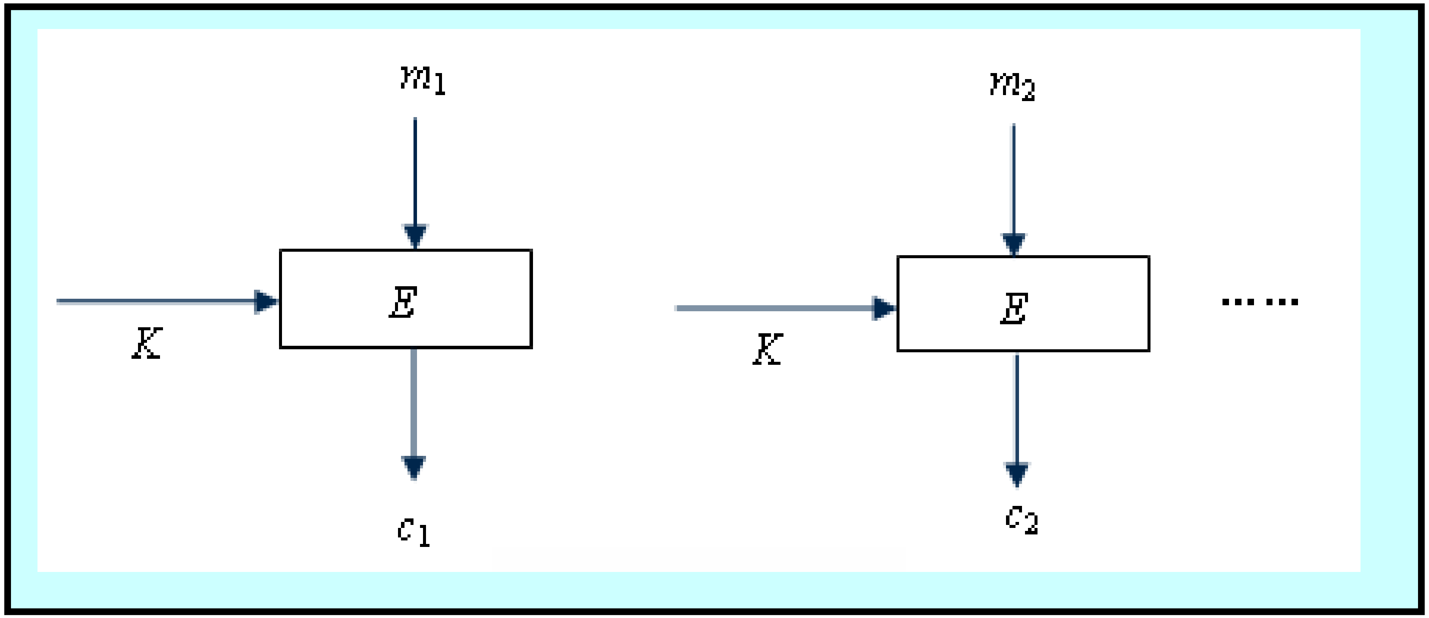
- 3.1 分组密码概述
- 3.2 数据加密标准（DES）
- 3.3 国际数据加密算法（IDEA）
- 3.4 高级数据加密标准（AES）
- 3.5 分组密码工作模式

3.5 分组密码工作模式

- **数据**
  - **密钥:** K
  - **输入:** t**个长度为**n**的明文块**$x_1 x_2 \ldots x_t$
  - **输出:** t**个长度为**n**的密文块**$y_1 y_2 \ldots y_t$

3.5 分组密码工作模式

- **电子密码本模式(ECB)**
  - **电码本模式ECB (electronic codebook mode)**
    - 加密: $y_i=E_K(x_i)$
    - 解密: $x_i=E_K^{-1}(y_i)$

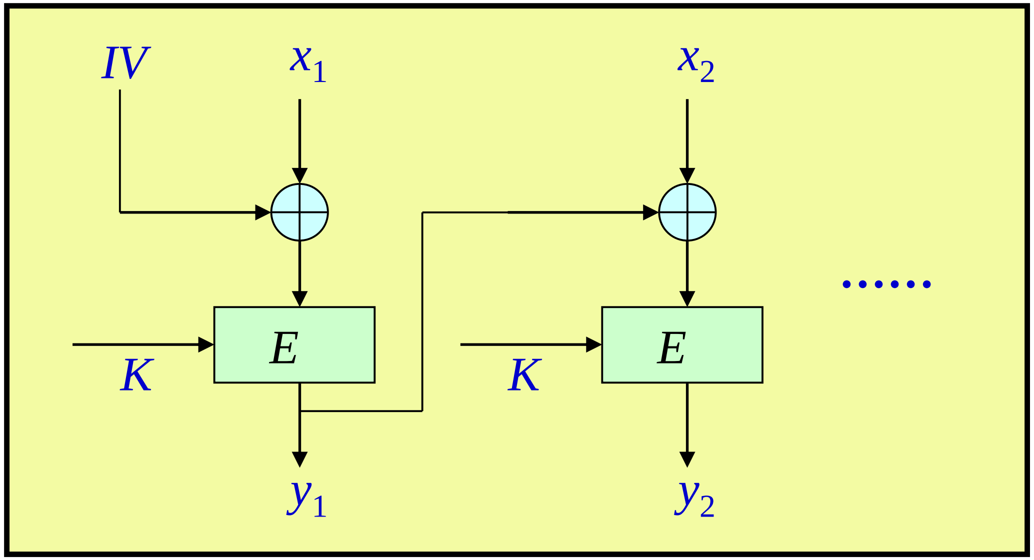


3.5 分组密码工作模式

- **电子密码本模式(ECB)**
  - **特性**
    - 在相同密钥的情况下，相同的明文产生相同的密文。容易暴露明文的数据模式。
    - 明文块x_i 的改变只引起密文块y_i的改变,其他密文块不变
    - 密文块在传输中一位出错，只影响该块的解密

3.5 分组密码工作模式

- **密文分组链接模式(CBC)**
  - **密文分组链接模式CBC(cipher block chaining mode)**
    - 给定初始向量:$y_0=IV$
    - 加密: $y_i=E_K(y_{i-1}\oplus x_i)$
    - 解密: $x_i=y_{i-1}\oplus E_K^{-1}(y_i)$



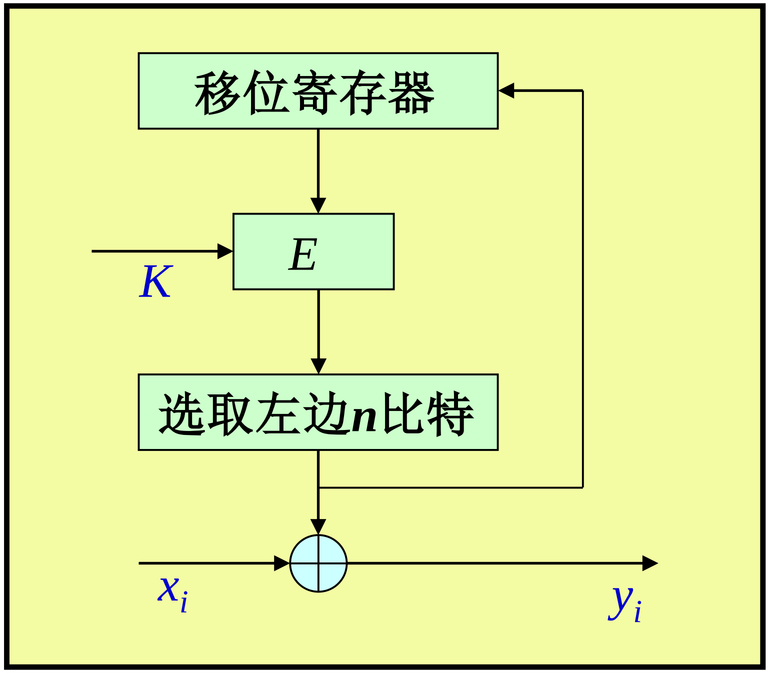
3.5 分组密码工作模式

◆密文分组链接模式CBC特性

- 在相同密钥和初始向量情况下, 相同的明文块加密产生的密文块不同. 能够掩盖明文的数据模式.
- 密文y_i依赖于x_i及之前所有的明文。所以，对密文的重新排序将影响正确解密，对消息的重发、插入、删除敏感.
- 密文块y_i在传输中一位出错，将影响y_i与y_{i+1}两块的正确解密
- 具有自同步功能: 密文块y_i在传输中一位出错, y_{i+2}能正确解密
- 某明文块x_i 发生变化将引起后面的所有密文块发生改变

3.5 分组密码工作模式

- **输出反馈模式(OFB)**
  - **输出反馈模式OFB(output feedback mode)**

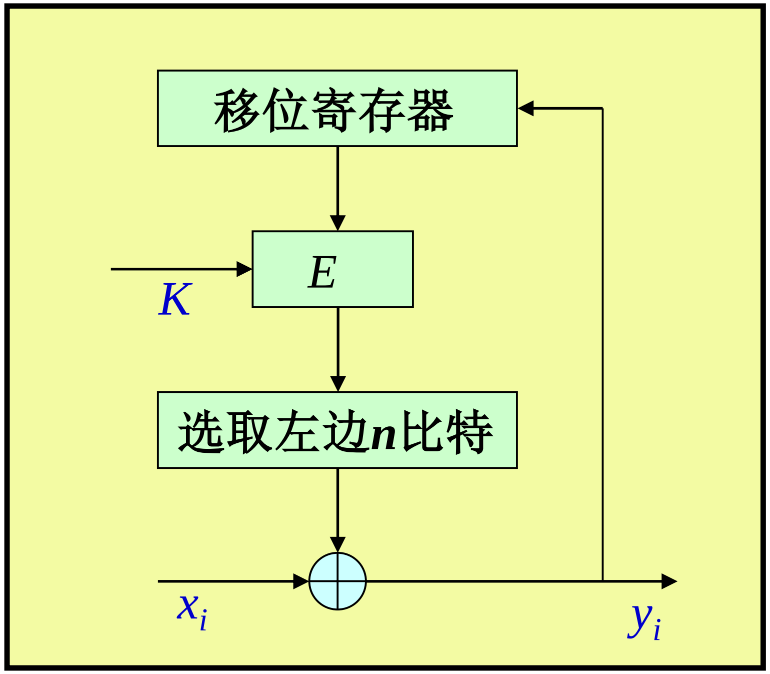


3.5 分组密码工作模式

- **输出反馈模式(OFB)**
  - **输出反馈模式OFB特性**
    - 将分组加密算法作为一个密钥流发生器，构建一个流密码系统
    - 通过改变初始向量使相同明文加密产生不同密文.
    - 明文块x_i改变只引起密文块y_i改变,其他密文块不变.
    - 对密文篡改问题难于发现。如果密文y_i在传输中一位出错，解密仅仅是对应位不对，错误不会传播.
    - 无自同步功能，系统必须严格保持同步.

3.5 分组密码工作模式

- **密文反馈模式(CFB)**
  - **密文反馈模式CFB(cipher feedback mode)**

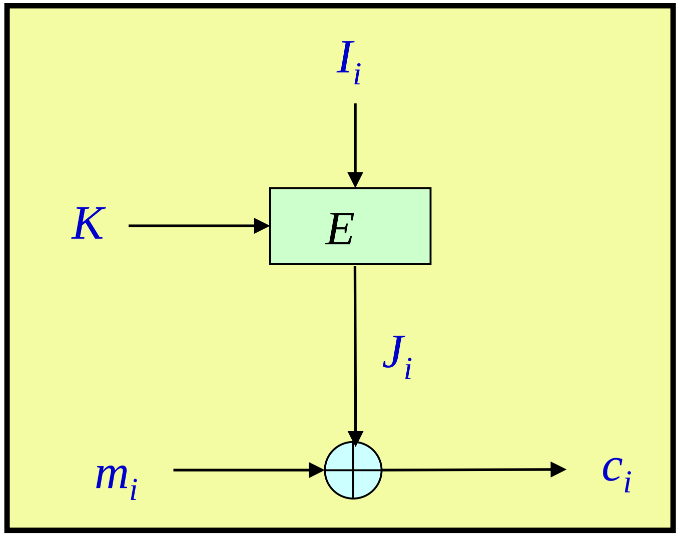


3.5 分组密码工作模式

- **密文反馈模式(CFB)**
  - **密文反馈模式CFB特性**
    - 将分组加密算法作为一个密钥流发生器，构建一个流密码系统
    - 通过改变初始向量使相同明文加密产生不同密文.
    - 密文y_i依赖于x_i及之前若干明文块。对密文的重新排序将影响正确解密.
    - 密文块y_i在传输中一位或多位出错，将影响后面若干块的正确解密. 直到y_i移出寄存器时才能恢复正确解密.
    - 具有自同步功能.

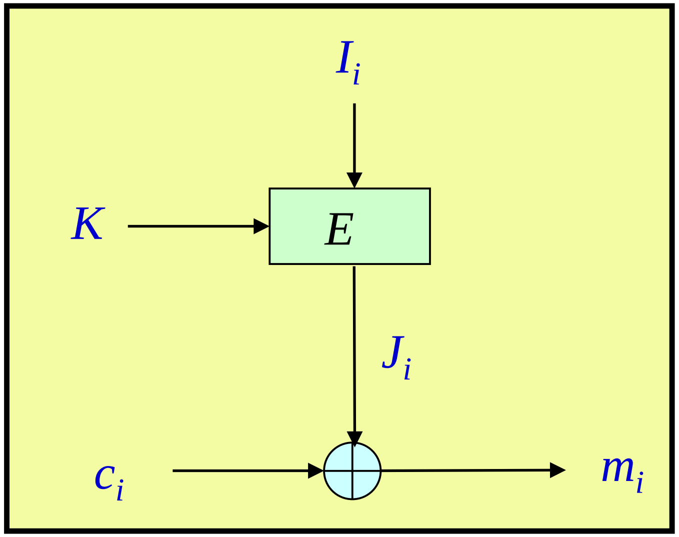
3.5 分组密码工作模式

- **计数器模式(CTR)**
  - **已知**
    - 计数序列: I_1、I_2、…、I_t。
    - 明文块: m_1、m_2、…、$m_{t\text{-}1}$、m_t，其中m_1、m_2、…、$m_{t\text{-}1}$的长度为n，m_t的长度是L，$L \le n$。
  - **加密算法**
    - $J_i=E_K(I_i)$ $(i=1,2,\ldots,t)$
    - $c_i=m_i \oplus J_i$ $(i=1,2,\ldots,t\text{-}1)$
    - $c_t=m_t \oplus \mathrm{MSB}_L(J_t)$
    - 其中$\mathrm{MSB}_L(J_t)$表示J_t中的高L位



3.5 分组密码工作模式

- **计数器模式(CTR)**
  - **解密算法**
    - $J_i=E_K(I_i)\ \ (i=1,2,\ldots,t)$
    - $m_i=c_i\oplus J_i\ \ (i=1,2,\ldots,t-1)$
    - $m_t=c_t\oplus \mathrm{MSB}_L(J_t)$
    - 其中$\mathrm{MSB}_L(J_t)$表示J_t中的高L位



- **计数器模式(CTR)特性**
  - 优点: 安全、高效、可并行、适合加密任意长度的明文，J_i的计算可通过预处理高速进行，加解密过程仅涉及加密运算，不用实现解密算法
  - 缺点: 没有错误传播，因而不易确保数据完整性。

第3章 习 题

1. 设 DES 算法中，明文 M 和密钥 K 分别为

M=0011 1000 1100 0100 1011 1000 0100 0011 1101 0101 0010 0011 1001 1110 0101 1110

K=1010 1001 0011 0101 1010 1100 1001 1011 1001 1101 0011 1110 1101 0101 1100 0011

求 L_1 和 R_2。

2. 设 DES 算法中 S_4 盒的输入为 010101，求其输出。

3. 在图 3.18 所示的 IDEA 算法 MA 部件中，已知输入数据为

X_1= 0000100110010011　　X_2= 1010001001001011

K_1= 1100001000100110　　K_2= 1000011010000110

求输出求 Y_1 和 Y_2。

4. 在 IDEA 算法中，已知明文 M 和密钥 K 分别为

M=10101010 11100110 01010101 00001111 11001100 00110011 10011001 01100110

K= 00000000 11111111 00000000 11111111 11111111 00001111 11110000 11111111 00001111 11110000 11111111 00000000 00001111 11111111 11110000 00001111

求第 1 轮的输出和第 2 轮的输入。

第3章 习 题

5. 计算 $GF(2^8)$上多项式乘法

（1）'57' · '9D'

（2）'66' · 'D5'

（3）'8C' · 'B5'

（4）'F4' · '3C'

6. 设 AES 算法分组长度为 128，输入的明文 *M* 和密钥 *K* 分别为

M=32 6C A8 F6 42 31 8C D6 43 72 64 E0 98 89 07 C3

K=A3 61 89 B5 54 12 D8 90 F4 14 FC AB 81 70 AE 3F

求 AES 的第 1 轮输出。