

密码学

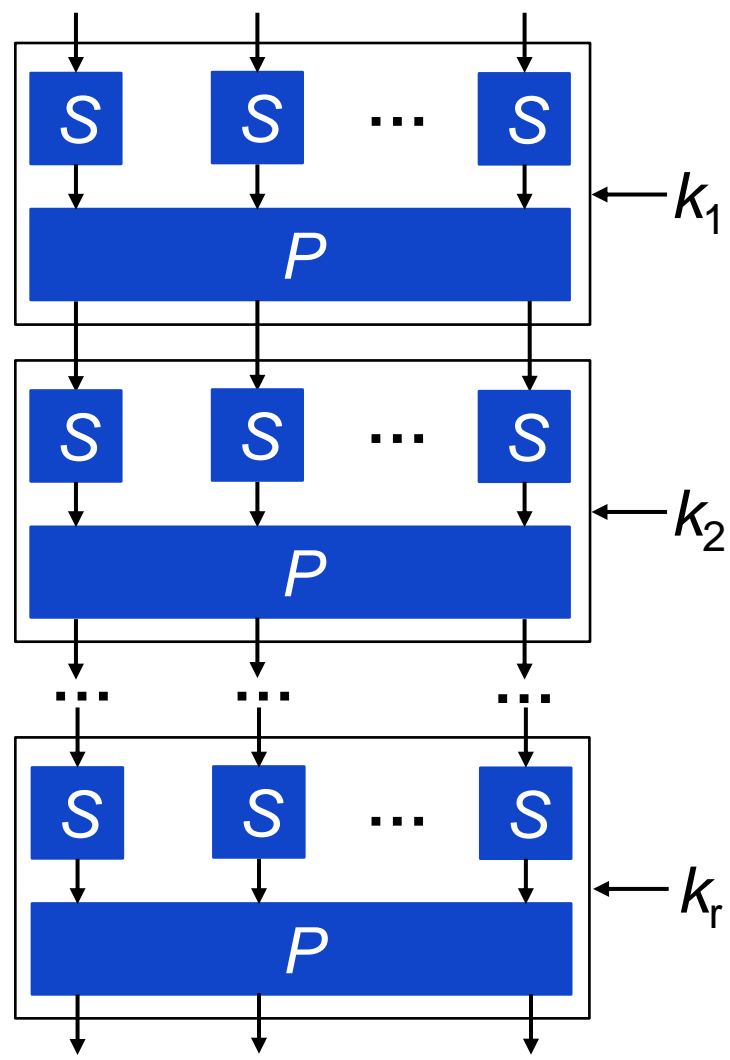
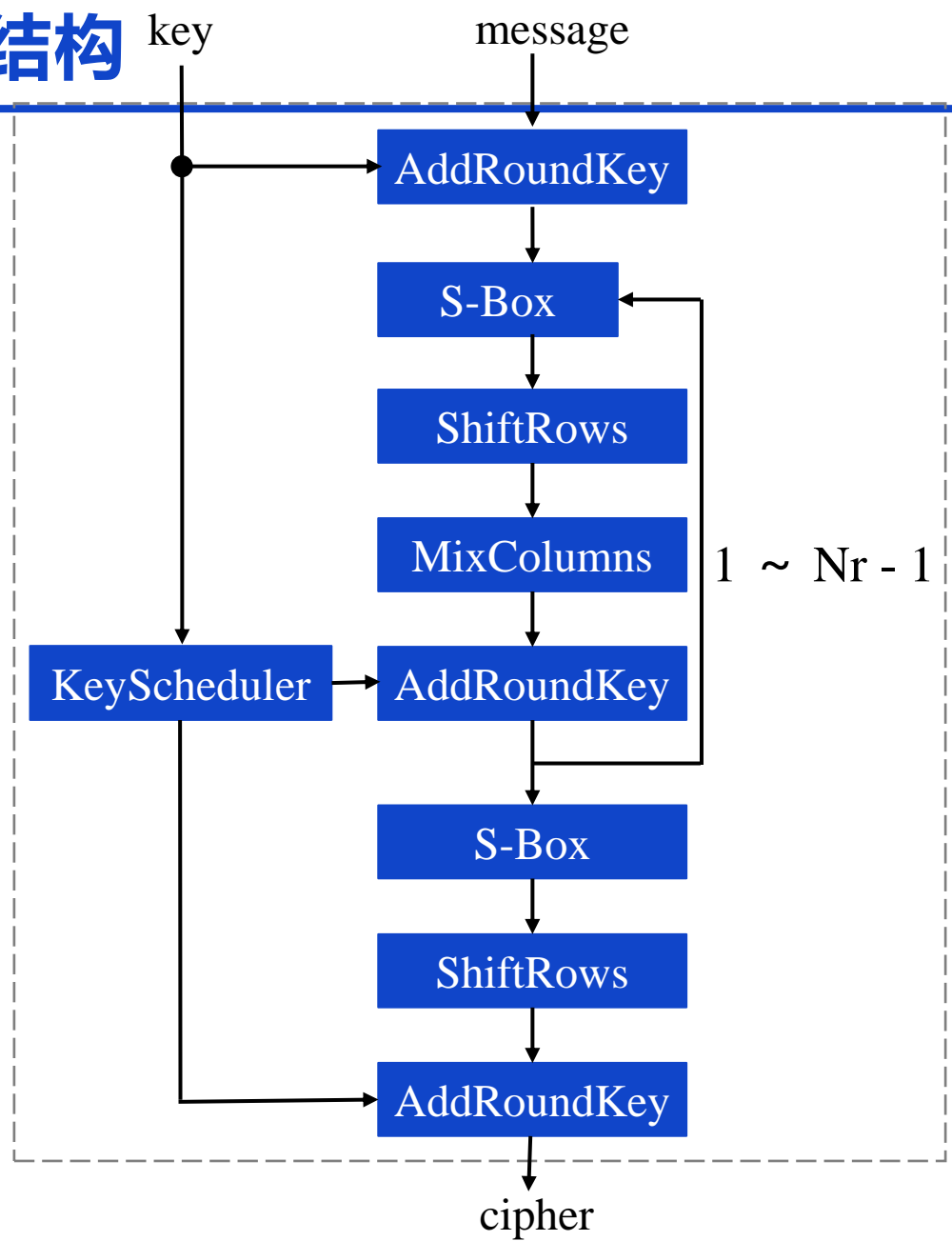
第四章 分组密码

网络安全学院

胡伟 朱丹

weihu/zhudan@nwpu.edu.cn

- ✎ S盒变换 – S-Box
- ✎ 行移位 – ShiftRows
- ✎ 列混合 – MixColumns
- ✎ 密钥扩展 – KeyScheduler
- ✎ 加轮密钥 – AddRoundKey



✎ S盒变换的特点

- ✎ 把输入字节看成 $GF(2^8)$ 上的元素
- ✎ 求出其在 $GF(2^8)$ 上的逆元素

设 $a(x)$ 的逆元为 $b(x)$,

则 $a(x)b(x) = 1 \bmod m(x)$

其中, $m(x) = x^8 + x^4 + x^3 + x + 1$

✎ 第二步：对上面的结果作如下的仿射变换

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

知识回顾—行移位

行移位变换 ShiftRow(128位)

- 行移位变换对状态矩阵的行进行循环左移
- 第 0 行不移位，第 1 行移 1 字节，第 2 行移 2 字节，第 3 行移 3 字节
- 行移位变换属于置换，属于线性变换，本质在于把数据打乱重排，起扩散作用

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	循环左移0位	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	循环左移1位	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	循环左移2位	$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	循环左移3位	$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

✦ 列混合变换 MixColumn (128位), 属于线性变换, 起扩散作用

✦ 把状态的列视为GF(2⁸)上的多项式 $a(x)$, 乘以一个固定的多项式 $c(x)$, 并模 x^4+1 :

$$b(x) = a(x)c(x) \bmod x^4 + 1$$

✦ 其中, $c(x) = c_3x^3 + c_2x^2 + c_1x + c_0 = 03x^3 + 01x^2 + 01x + 02$

✦ 写成矩阵形式

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

$$b_0 = a_0c_0 + a_3c_1 + a_2c_2 + a_1c_3 \quad \text{四次项和常量}$$

$$b_1 = a_1c_0 + a_0c_1 + a_3c_2 + a_2c_3 \quad \text{一次项}$$

$$b_2 = a_2c_0 + a_1c_1 + a_0c_2 + a_3c_3 \quad \text{二次项}$$

$$b_3 = a_3c_0 + a_2c_1 + a_1c_2 + a_0c_3 \quad \text{三次项}$$

- 轮密钥加变换的逆就是其本身

$$(AddRoundKey)^{-1} = AddRoundKey$$

- 行移位变换的逆是状态的后三行分别循环左移3, 2, 1个字节 (或循环右移1, 2, 3个字节)

- 列混合变换把状态的每一列都乘以一个多项式 $c(x)$:

$$b(x) = a(x)c(x) \bmod x^4 + 1$$

- 列混合变换的逆就是状态的每列都乘以 $c(x)$ 的逆多项式 $d(x)$:

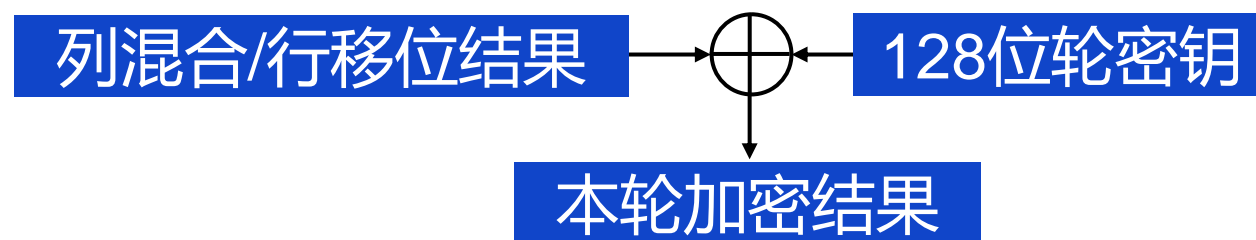
$$d(x) = (c(x))^{-1} \bmod x^4 + 1$$

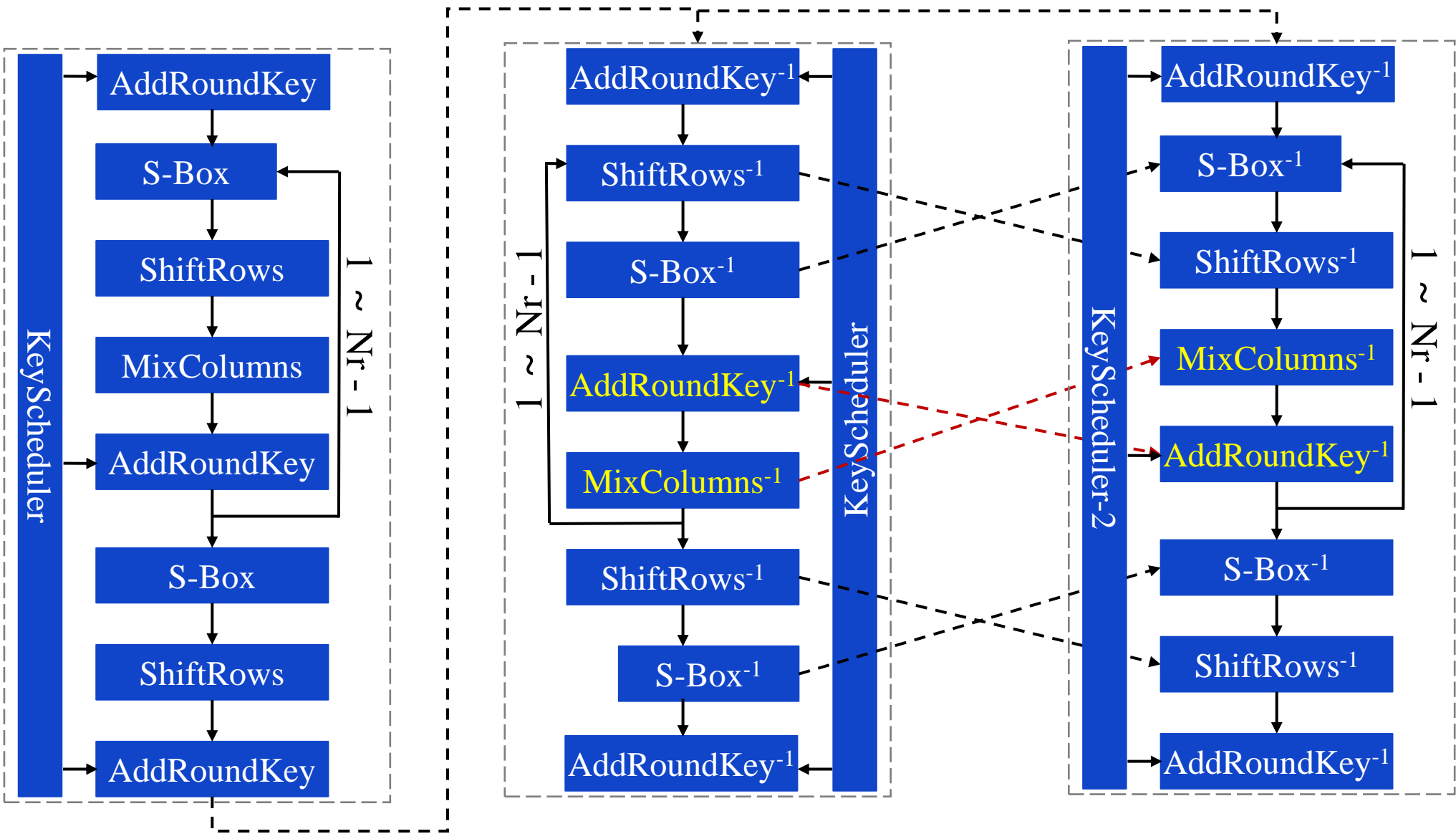
$$c(x) = 03x^3 + 01x^2 + 01x + 02$$

$$d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$$

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

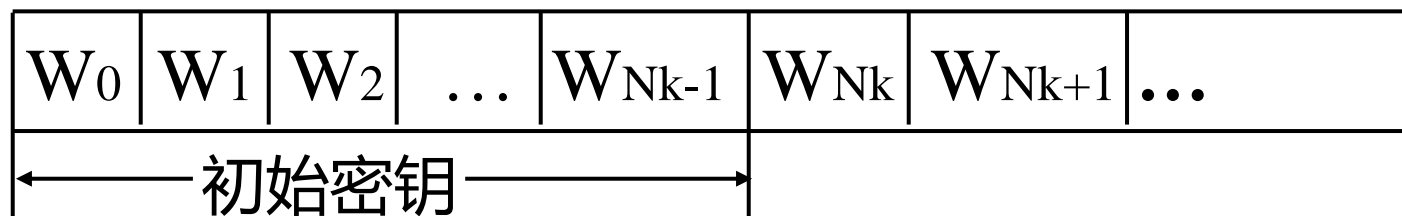
- ✦ 轮密钥加变换AddRoundKey(128位)
 - ✦ 把轮密钥与状态进行模2加
 - ✦ 轮密钥根据密钥产生算法产生
 - ✦ 轮密钥长度等于数据分组长度





❖ 密钥扩展 ($N_k \leq 6$ 的密钥扩展)

- ❖ 最前面的 N_k 个字由用户密钥填充
- ❖ 之后每个字 $W[j]$ 等于 $W[j-1]$ 与 N_k 个位置之前的字 $W[j - N_k]$ 的异或
- ❖ 对于 N_k 的整数倍的位置处的字，在异或之前，对 $W[j-1]$ 进行Rotl变换和ByteSub变换，再异或一个轮常数Rcon

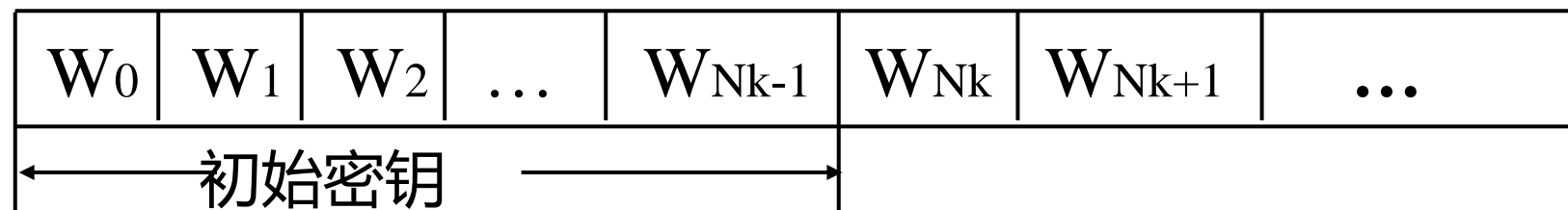


当 j 是 N_k 的整数倍时: $W_j = W_{j-N_k} \oplus \text{ByteSub}(\text{Rotl}(W_{j-1})) \oplus \text{Rcon}[j/N_k]$

否则: $W_j = W_{j-N_k} \oplus W_{j-1}$

✎ $N_k > 6$ 的密钥扩展

✎ 增加: $N_k > 6$ 的密钥扩展与 $N_k \leq 6$ 的密钥扩展不同之处在于: 如果 j 被 N_k 除的余数为 4, 则在异或之前, 对 $W[j-1]$ 进行 SubBytes 变换



当 j 是 N_k 的整数倍时: $W_j = W_{j-N_k} \oplus \text{ByteSub}(\text{Rotl}(W_{j-1})) \oplus \text{Rcon}[j/N_k]$

当 j 被 N_k 除的余数为 4: $W_j = W_{j-N_k} \oplus \text{ByteSub}(W_{j-1})$

否则: $W_j = W_{j-N_k} \oplus W_{j-1}$

- ✎ 数据类型：定义便于做移位和逻辑运算的数据类型
- ✎ S盒：采用查找表方式较为简单
- ✎ 行移位：字节之间的赋值和交换
- ✎ 列混合：[03 01 01 02]采用 x_{time} 乘法实现， $03 = 02 \oplus 01$
- ✎ 密钥扩展：存储常量，调用S盒，按规则进行

- ✎ AES算法中对安全性影响最大的变换是什么？为什么？
- ✎ AES最后一轮与其它轮的差别是什么？对安全性有何影响？
- ✎ AES加密算法和解密算法的区别有哪些？
- ✎ 列举3个DES和AES的代表性不同之处。

章节安排

Outline



SM4密码算法



PRESENT密码算法

章节安排

Outline



SM4密码算法



PRESENT密码算法

- 坚持密码的公开设计原则

 - 密码的安全应仅依赖于密钥的保密，不依赖于算法的保密

- 公开设计原则并不要求使用时公开所有的密码算法

 - 核心密码不能公开算法

 - 核心密码的设计也要遵循公开设计原则

- 商用密码应当公开算法

 - 美国DES开创了公开商用密码算法的先例

 - 美国经历了DES（公开）→ EES（保密）→ AES（公开）的曲折过程，证明公开征集、公布算法的路线是正确的

 - 欧洲也公开商用密码算法






Kerckhoffs

- ✎ 我国商用密码长期以来不公开密码算法，只提供密码芯片
 - ✎ 由少数专家设计，难免有疏漏
 - ✎ 难于标准化，应用成本高，不利于推广应用
- ✎ 近年来我国陆续公布了商用密码算法
 - ✎ 2006年2月公布了分组密码SM4
 - ✎ 2021年，SM4加密算法作为ISO/IEC国际标准
 - ✎ 2011年2月公布了椭圆曲线密码SM2和杂凑算法SM3
 - ✎ 商用密码管理更加科学化、与国际接轨
 - ✎ 这将促进我国商用密码的发展


- ✓ 美国1997年启动的NIST计划
- ✓ 欧洲2000年启动的NESSIE计划
- ✓ 欧洲2004年启动的ECRYPT计划
- ✓ 美国2007年启动的SHA-3计划
- ✓ 我国近年来也形成了国密标准

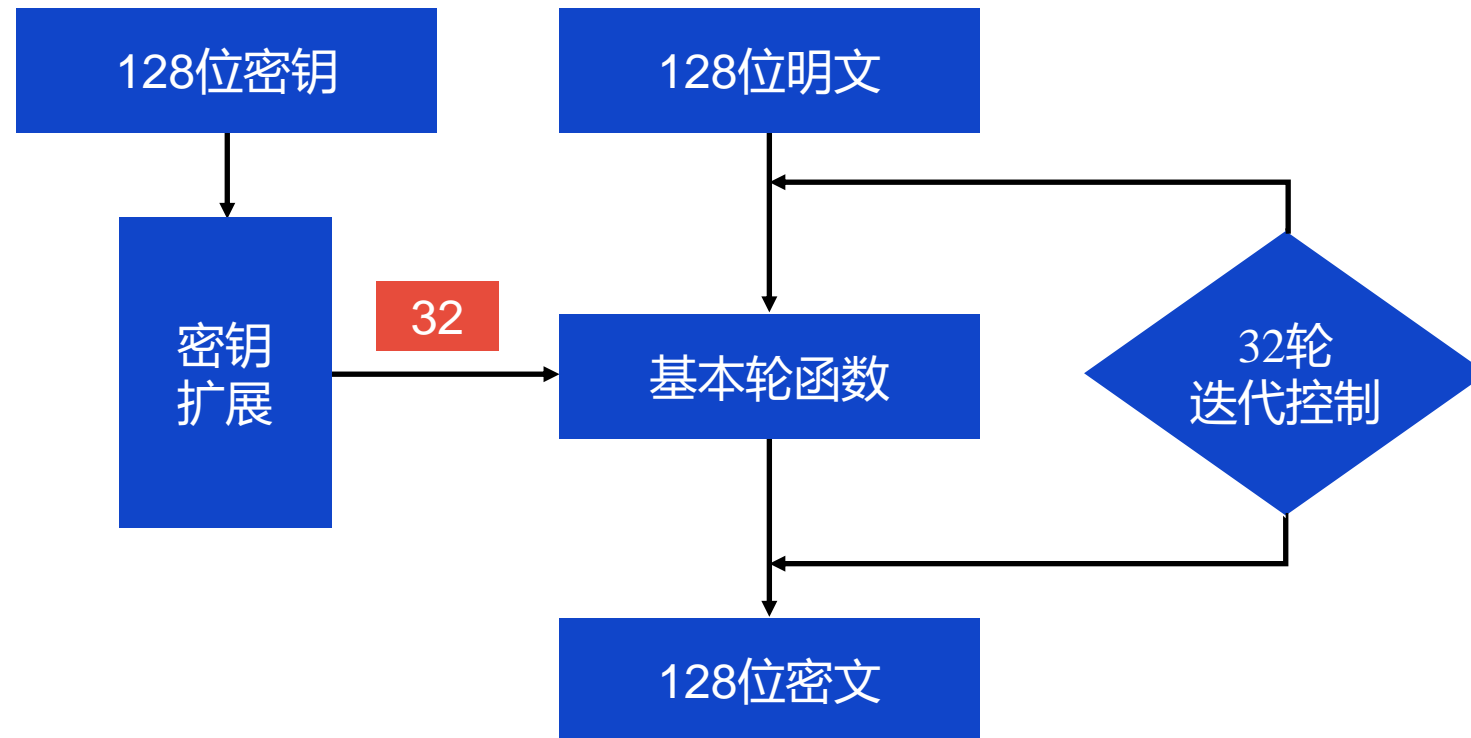
- ✓ 国家商用密码管理办公室制定了一系列密码标准，包括
- ✓ **SM1** (SCB2) - 128位分组密码算法，算法强度与AES相当
- ✓ **SM2** - 椭圆曲线公钥密码算法
- ✓ **SM3** - 杂凑（哈希、散列）算法
- ✓ **SM4** - 128位分组密码算法
- ✓ **SM7** - 128位分组密码算法，适用于非接触式IC卡
- ✓ **SM9** - 标识密码算法
- ✓ 祖冲之密码算法 (**ZUC**) – 流密码算法，运用于移动通信4G网络中的**国际标准**密码算法

分组密码

-  由数据分组（明文，密文）长度为128位、密钥长度为128位
-  迭代轮数：32轮
-  数据处理单位：字节（8位），字（32位）

密码算法特点

-  对合运算：解密算法与加密算法相同
-  密钥生成算法与加密算法结构类似
-  不是SP结构，也不是Feistel结构，属于滑动窗口结构



4.12(2) SM4密码算法概述

✎ 基本运算

✎ 模2加: \oplus , 32 比特异或运算

✎ 循环移位: $\lll i$, 把32位字循环左移 i 位

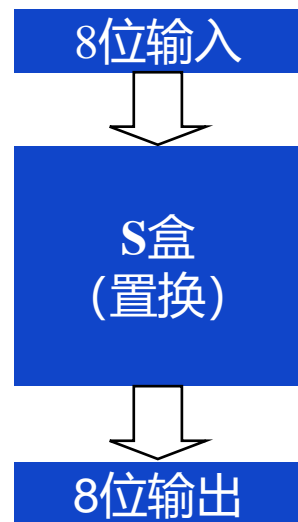
✎ 基本密码部件: 非线性字节变换部件S盒:

✎ 8位输入, 8位输出

✎ 本质上是 8位的非线性置换

✎ 设输入为 a , 输出为 b , S盒运算可表示为:

$$b = S\text{-Box}(a)$$



4.12(2) S盒查找表

- 以输入的前半字节为行号，后半字节为列号，行列交叉点处的数据即为输出
- 与AES的S盒相当
- S-Box(0xAC) = ?

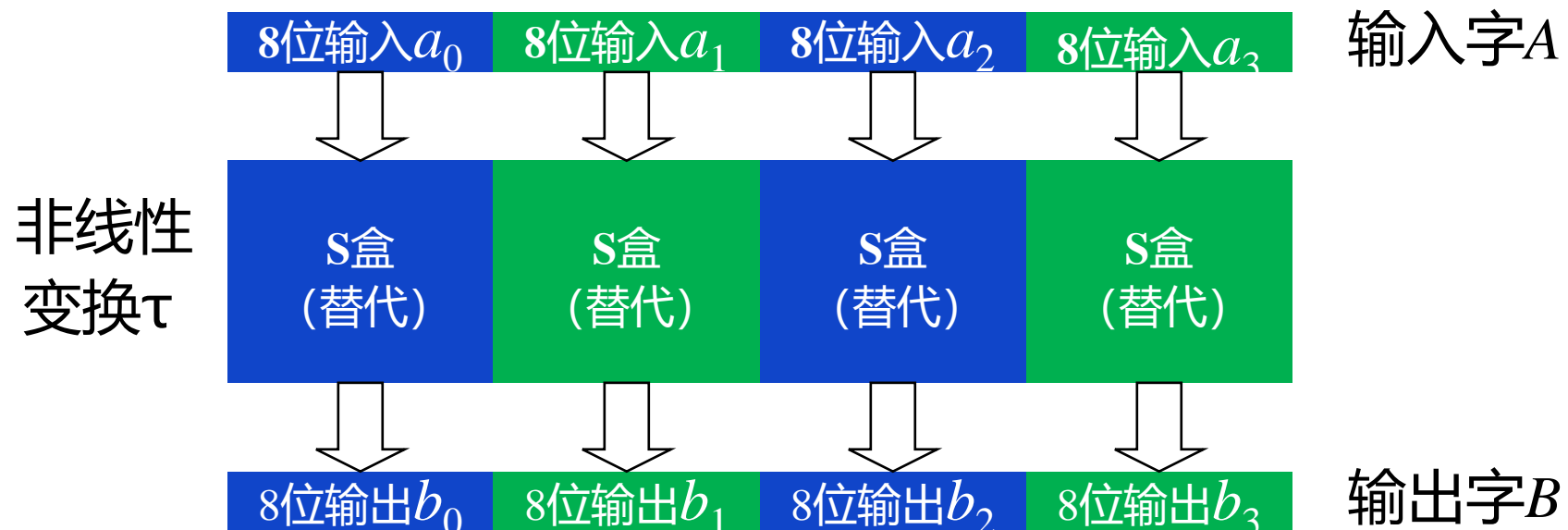
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	d6	90	e9	fe	cc	e1	3d	b7	16	b6	14	c2	28	fb	2c	05
1	2b	67	9a	76	2a	be	04	c3	aa	44	13	26	49	86	06	99
2	9c	42	50	f4	91	ef	98	7a	33	54	0b	43	ed	cf	ac	62
3	e4	b3	1c	a9	c9	08	e8	95	80	df	94	fa	75	8f	3f	a6
4	47	07	a7	fc	f3	73	17	ba	83	59	3c	19	e0	85	4f	a8
5	68	6b	81	b2	71	64	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	ea	60	e0	29	23	ab	8d	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

✎ 非线性字变换：32位字的非线性变换

✎ 4个S盒并行替代

✎ 设输入字 $A = (a_0, a_1, a_2, a_3)$ ，输出字 $B = (b_0, b_1, b_2, b_3)$

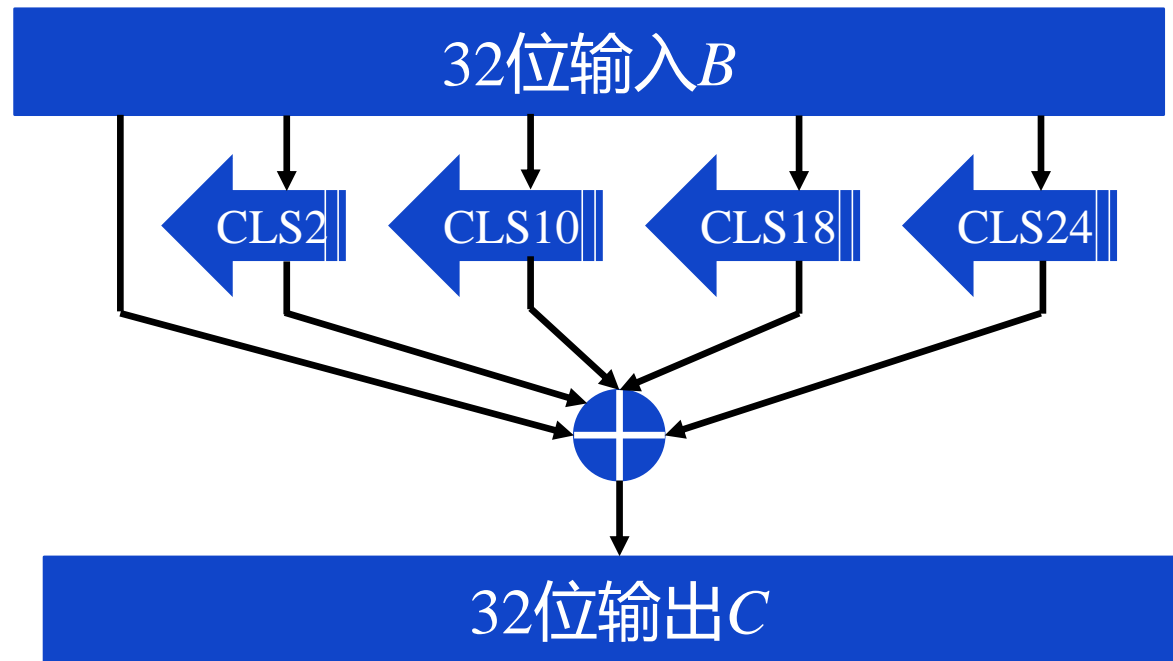
✎ $B = \tau(A) = (\text{S-Box}(a_0), \text{S-Box}(a_1), \text{S-Box}(a_2), \text{S-Box}(a_3))$



✎ 线性变换，32位输入，32位输出

✎ 设输入为 B ，输出为 C ，表为： $C = L(B)$

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



✎ 由非线性变换 τ 和线性变换 L 复合而成

✎ 记 $T(A) = L(\tau(A))$

✎ 先S盒变换, 后 L 变换

$$A = (a_0, a_1, a_2, a_3)$$

$$B = \tau(A) = (\text{S-Box}(a_0), \text{S-Box}(a_1), \text{S-Box}(a_2), \text{S-Box}(a_3))$$

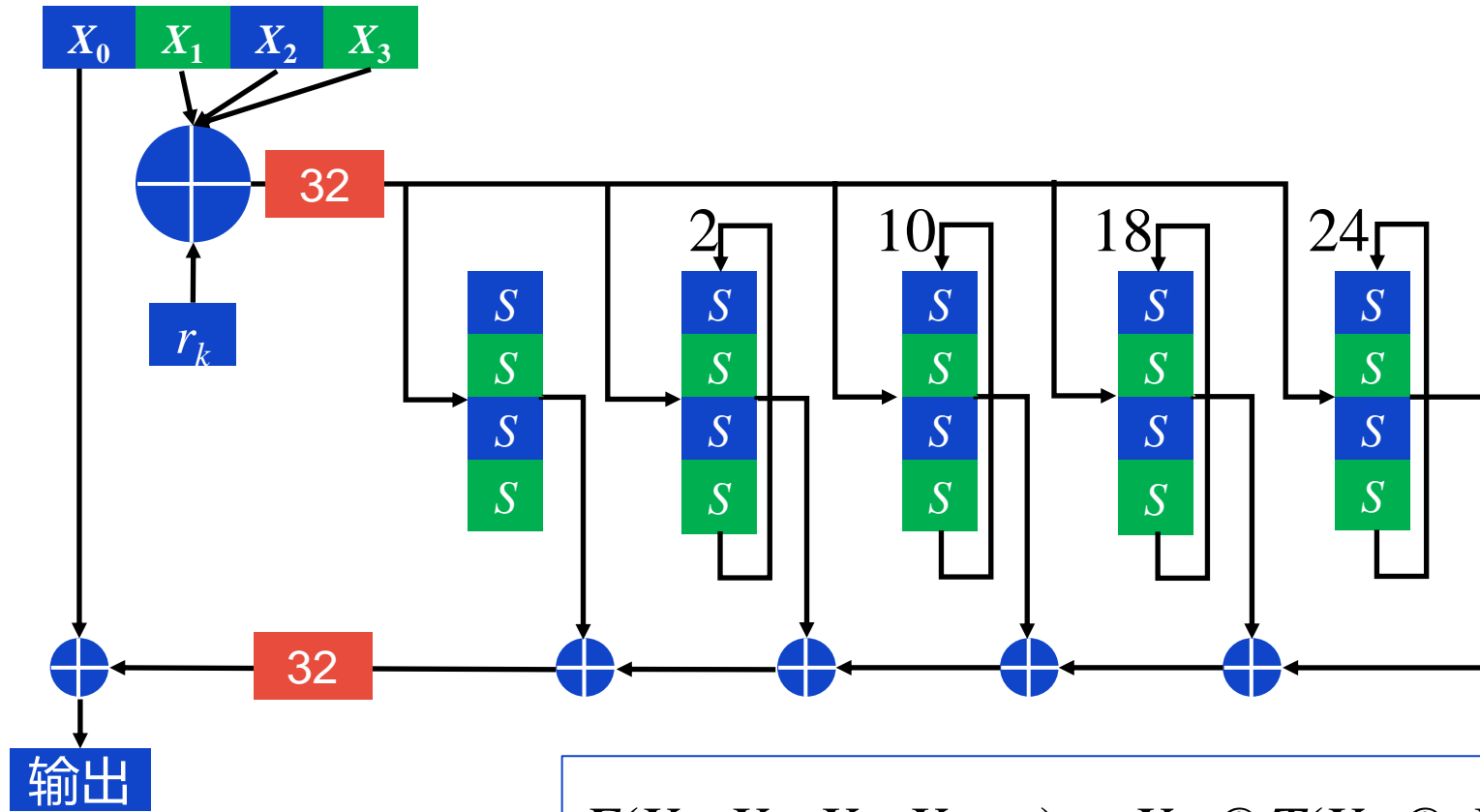
$$C = L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24)$$

- ✎ 输入数据: (X_0, X_1, X_2, X_3) , 128位, 四个32位字
- ✎ 输入轮密钥: r_k , 32位字
- ✎ 输出数据: 32位字
- ✎ 轮函数 F 的定义:

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

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

4.12(2) 轮函数



$$F(X_0, X_1, X_2, X_3, r_k) = X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus r_k), T(X) = L(\tau(X))$$

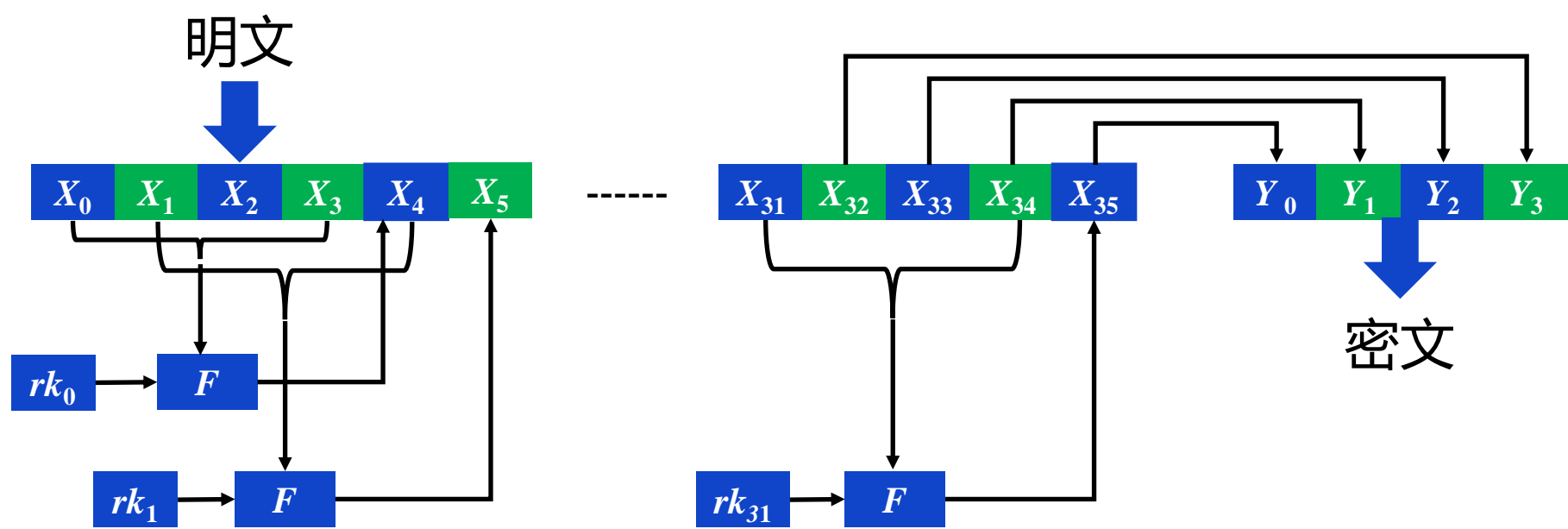
$$B = \tau(A) = (\text{S-Box}(a_0), \text{S-Box}(a_1), \text{S-Box}(a_2), \text{S-Box}(a_3))$$

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

- ✎ 输入明文: (X_0, X_1, X_2, X_3) , 128位, 四个32位字
- ✎ 输入轮密钥: $r_{ki} (i = 0, 1, \dots, 31)$, 32位字, 共32个轮密钥
- ✎ 输出密文: (Y_0, Y_1, Y_2, Y_3) , 128位, 四个32位字
- ✎ 算法结构: 轮函数32轮迭代, 每轮使用一个轮密钥

$$\begin{cases} X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, r_{ki}) \\ \quad = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus r_{ki}), \quad i = 0, 1, \dots, 31 \\ \quad = X_i \oplus L(\tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus r_{ki})), \quad i = 0, 1, \dots, 31 \\ (Y_0, Y_1, Y_2, Y_3) = (X_{35}, X_{34}, X_{33}, X_{32}) \end{cases}$$

✎ 滑动窗口迭代结构 (广义 Feistel结构)



- SM4密码算法是对合的，因此解密与加密算法相同，只是轮密钥的使用顺序相反
- 输入密文： (Y_0, Y_1, Y_2, Y_3) ，128位，四个32位字
- 输入轮密钥： r_{ki} ($i = 31, 30, \dots, 1, 0$)，32位字，共32个轮密钥
- 输出明文： (X_0, X_1, X_2, X_3) ，128位，四个32位字

$$\left\{ \begin{array}{l} X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, r_{ki}) \\ \quad = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus r_{ki}), \quad i = 0, 1, \dots, 31 \\ \quad = X_i \oplus L(\tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus r_{ki})), \quad i = 0, 1, \dots, 31 \\ (Y_0, Y_1, Y_2, Y_3) = (X_{35}, X_{34}, X_{33}, X_{32}) \end{array} \right.$$

✎ 常数FK：在密钥扩展中使用的一些常量

✎ $FK_0 = (A3B1BAC6)$

✎ $FK_1 = (56AA3350)$

✎ $FK_2 = (677D9197)$

✎ $FK_3 = (B27022DC)$

✎ 固定参数C: 32 个固定参数 C_{ki} , $i = 0, 1, \dots, 31$

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

4.12(5) SM4密钥扩展算法

✎ 输入加密密钥: $MK = (MK_0, MK_1, MK_2, MK_3)$

✎ 输出轮密钥: $rk_i, i = 0, 1, 2, \dots, 30, 31$

✎ 中间数据: $K_i, i = 0, 1, 2, \dots, 34, 35$

✎ 扩展算法流程:

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

✎ For $i = 0, 1, 2, \dots, 30, 31$ Do

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

✎ 说明: T' 变换与加密算法轮函数中的 T 相似, 只将其中的线性变换 L 修改为 L'

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

$FK_0 = (A3B1BAC6)$, $FK_1 = (56AA3350)$, $FK_2 = (677D9197)$, $FK_3 = (B27022DC)$, CK_i 也是常数

- ✦ 国家专业机构设计，专业机构进行了充分的密码分析，是安全的
- ✦ 已有23轮SM4的差分密码分析
- ✦ 已有20轮SM4的线性密码分析
- ✦ SM4抵御差分故障注入攻击的能力较弱，平均需要47个错误即可恢复128位密钥

章节安排

Outline



SM4密码算法



PRESENT密码算法

✦ 主要面向物联网、RFID系统、无线传感网络、智能卡等计算资源受限的应用

NAME (<i>Nb/Nk</i>)	Reference	Struct.	Nb rounds
AES-128* (128/128)	[18]	SPN	10
CLEFIA-128* (128/128)	[48]	Feistel	18
DESXL (64/184)	[37]	Feistel	16
HIGHT (64/128)	[23]	Feistel	32
IDEA* (64/128)	[34]	Lai-Massey	8.5
KATAN & KTANTAN (32, 48, 64/80)	[9]	Stream	254
KLEIN (64/64, 80 and 96)	[19]	SPN	12, 16, 20
LBLOCK (64/80)	[57]	Feistel	32
LED (64/64 and 128)	[20]	SPN	32/48
mCrypton (64/64, 96 and 128)	[39]	SPN	12
MIBS (64/64 and 80)	[25]	Feistel	32
Noekeon* (128/128)	[14]	SPN	16
Piccolo (64/80 and 128)	[47]	Feistel	25/31
PRESENT (64/80 and 128)	[8]	SPN	31
TEA & XTEA (64/128)	[56]	Feistel	64
TWINE (64/ 80 and 128)	[51]	Feistel	36
SEA (96/96,...)	[50]	Feistel	Var.
SKIPJACK* (64/80)	[44]	Feistel	32

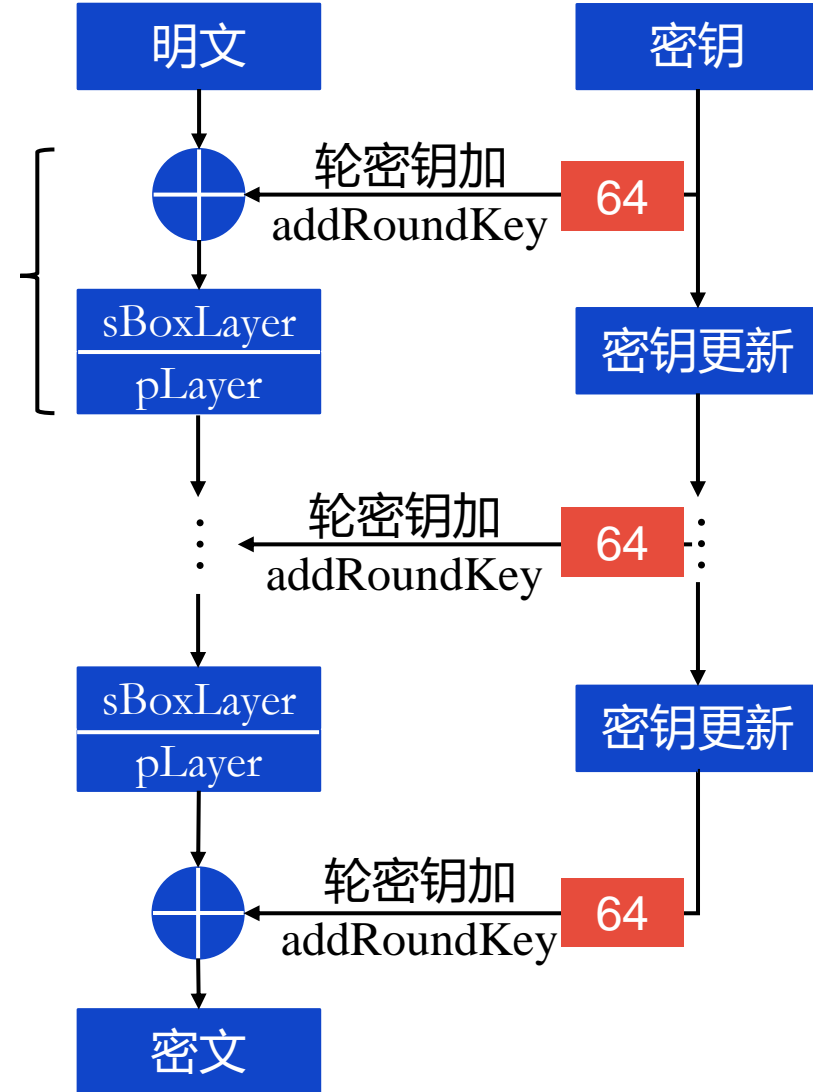
Mickaël Cazorla, et al. Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks. IDEA, 2013, 64(128)

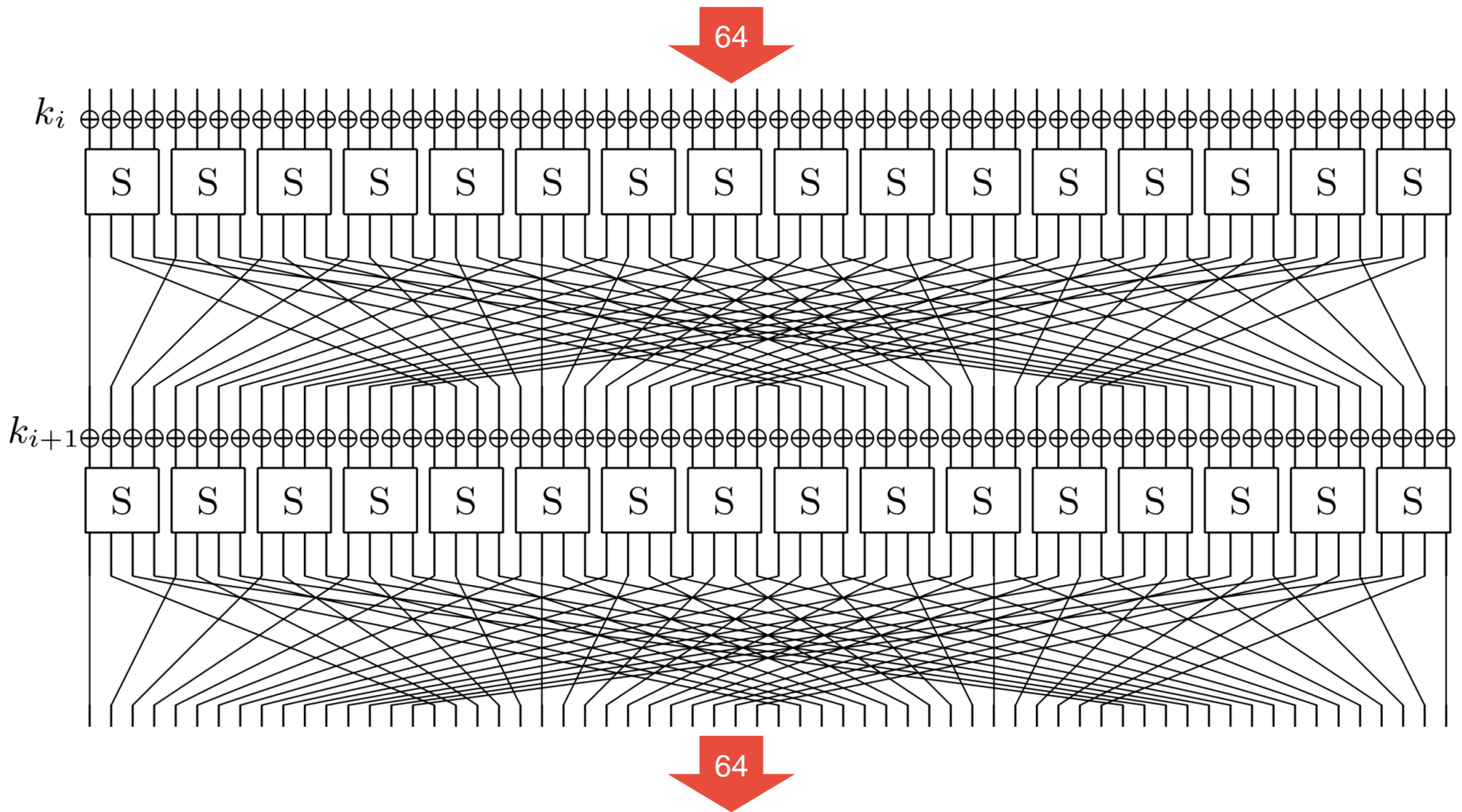
- ✎ 超轻量级分组密码，2007年在CHES会议上报道
 - ✎ 算法采用S-P网络结构
 - ✎ 支持80位和128位两种密钥长度
 - ✎ 分组长度为64位
 - ✎ 共迭代31轮
- ✎ 具有出色的硬件实现性能和简洁的轮函数设计
- ✎ 非常适合于物联网、RFID系统、无线传感网络、智能卡等资源受限的环境

✎ 基本运算

- ✎ 轮密钥加 (addRoundKey)
- ✎ S盒代换层 (sBoxLayer)
- ✎ P置换层 (pLayer)
- ✎ 32个子密钥(加密31轮),
目的是使结果白化

```
generateRoundKey(key)
for i = 1 to 31 do
  addRoundKey(State,  $K_i$ )
  sBoxLayer(State)
  pLayer(State)
end for
addRoundKey(State,  $K_{32}$ )
```





- ✎ 轮密钥加 (addRoundKey)
 - ✎ 64位状态和密钥对应位做异或 (模二加)
- ✎ S盒代换层 (sBoxLayer)
 - ✎ PRESENT使用了一个简单的4位S盒
 - ✎ 64位的状态划分为16个分组

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

✎ P置换层(pLayer): 对S盒替换的64位输出重新排列 (按比特)

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
P(i)	0	10	20	30	1	11	21	31	2	12	22	32	3	13	23	33
i	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
P(i)	4	14	24	34	5	15	25	35	6	16	26	36	7	17	27	37
i	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
P(i)	8	18	28	38	9	19	29	39	A	1A	2A	3A	B	1B	2B	3B
i	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
P(i)	C	1C	2C	3C	D	1D	2D	3D	E	1E	2E	3E	F	1F	2F	3F

- PRESENT支持80位和128位两种密钥长度
- 以80位的密钥为例
 - 维持一个寄存器 K ，存储80位的输入密钥
 - 第 i 轮子密钥由寄存器的最左64位组成 ($k_{79}, k_{78}, \dots, k_{16}$)
- 密钥更新（由第 i 轮子密钥计算第 $i+1$ 轮子密钥）：
 - 当前密钥（第 i 轮子密钥）循环左移61位
 - 最高4位进行S盒替换
 - 将当前加密轮数与 $[k_{19}, k_{18}, k_{17}, k_{16}, k_{15}]$ 进行异或操作

循环左移61位: $[k_{79}, k_{78}, \dots, k_1, k_0] \rightarrow [k_{18}, k_{17}, \dots, k_{20}, k_{19}]$

S盒替换: $[k_{79}, k_{78}, k_{77}, k_{76}] = S([k_{79}, k_{78}, k_{77}, k_{76}])$

异或操作: $[k_{19}, k_{18}, k_{17}, k_{16}, k_{15}] = [k_{19}, k_{18}, k_{17}, k_{16}, k_{15}] \oplus \text{round}$

- ✦ Hatzivasilis G , et al. A review of lightweight block ciphers. Journal of Cryptographic Engineering, 2017
- ✦ Mickaël Cazorla, et al. Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks. IDEA, 2013, 64(128)
- ✦ PRESENT: An Ultra-Lightweight Block Cipher,
https://link.springer.com/chapter/10.1007%2F978-3-540-74735-2_31
- ✦ 大作业：SM4的编程实现



感谢聆听!

THANK YOU FOR YOUR ATTENTION!