



密码学

第三章 序列密码算法

网络安全学院

朱丹

zhudan@nwpu.edu.cn



置 换 密 码

01

置换密码算法的原理是**不改变明文字符**，只将字符在明文中的排列顺序改变，从而实现明文信息的加密。置换密码有时又称为**换位密码**

替 代 密 码

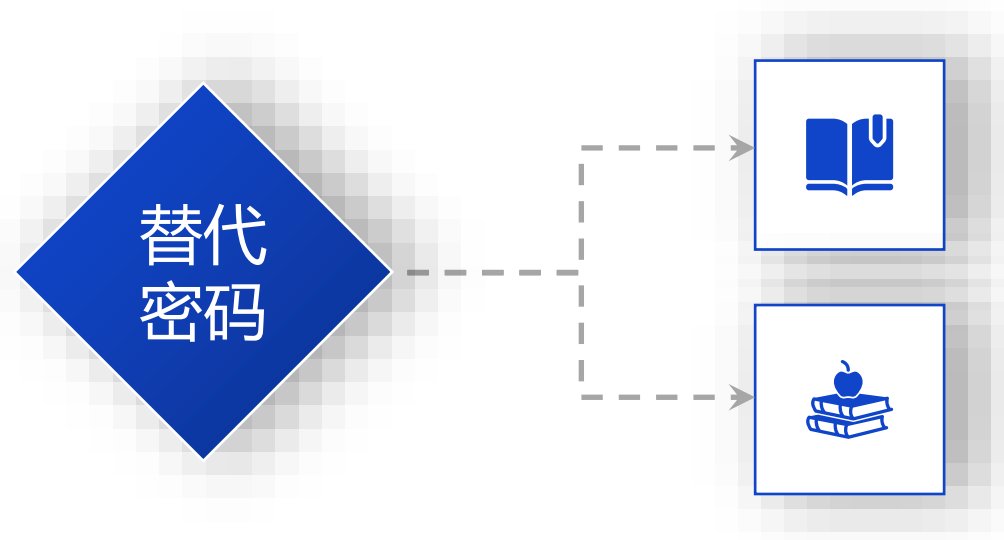
02

替代密码算法的原理是使用替代法进行加密，就是将**明文中的字符用其它字符替代**后形成密文。**加密后明文字符的形态会发生变化**

替代密码是利用预先定义的**代替规则**，对**明文逐字符进行替代**的密码算法

替代密码的**替代规则**就是其**密钥**

替代规则又称为**替代函数**、**替代表**或**S盒**



单表替代密码

只使用一个密文字符表

多表替代密码

使用多组密文字符表或映射函数





替代密码和置换密码相结合

01

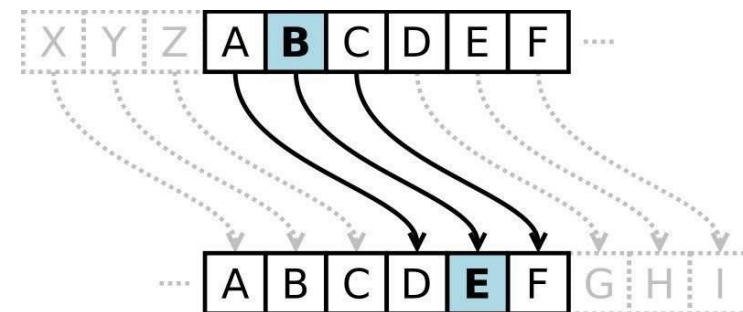
在实际应用中，通常将替代密码和置换密码结合起来，从而设计出安全的密码体制

替代 - 置换模型

02

这就是分组密码的替代-置换模型，即现代密码的设计思想：利用简单的密码变换的组合，设计出抗攻击能力强的密码算法

- ✓ 斯巴达密码
- ✓ 凯撒密码
- ✓ 加法密码
- ✓ 乘法密码
- ✓ 仿射密码
- ✓ Vigenere密码
- ✓ Vernam密码



加密押的日昇昌银票

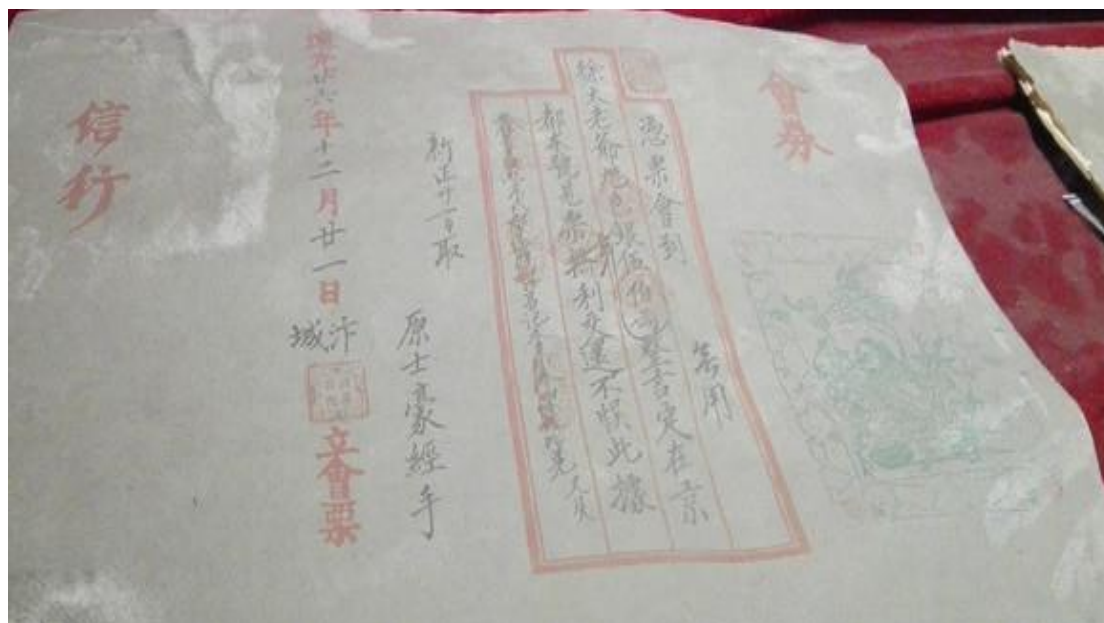
如“日升昌”太原票号留存在中国历史博物馆的一份防假密押是：

“谨防假票冒取，勿忘细视书章”，表示1至12个月；

“堪笑世情薄，天道最公平。昧心图自私，阴谋害他人。善恶终有报，到头必分明”，表示1至30天。

“坐客多察看，斟酌而后行”，表示银两的1至10。

“国宝流通”，表示万千百两。



豪密•1931（周恩来总理）



豪密，由中国共产党初期领导人之一周恩来亲自编制，以周恩来党内化名“伍豪”命名

“豪密”所用的密码从不重复，简单好记，却难以破译，直到1949年中国国民党垮台，都没有被破译出来

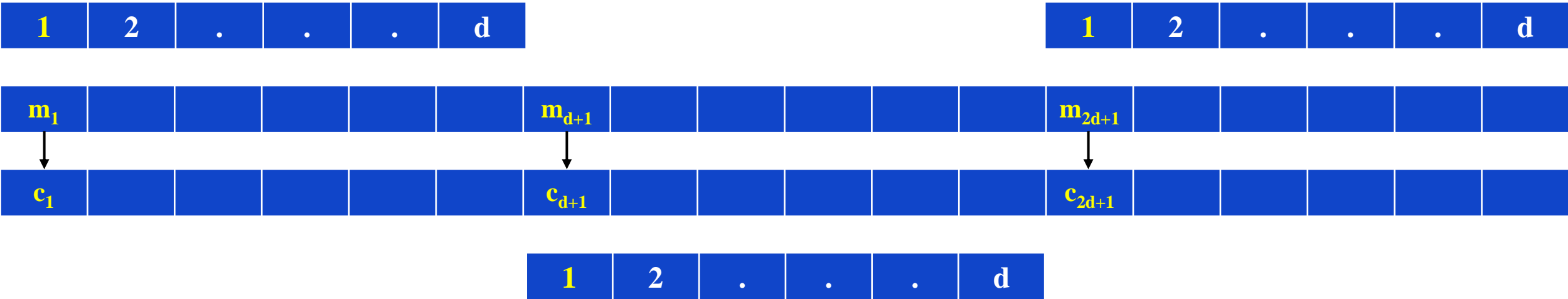
豪密很可能由两部分组成：书名与册码；页码、行数与字序，能够在电报中实现“同字不同码，同码不同字”

单表替代密码统计分析

- 首先，统计密文的各种统计特征
- 其次，分析双字母、三字母密文组，以区分元音和辅音字母
- 最后，分析字母较多的密文，可以使用猜测法

字母	A	B	C	D	E	F	G	H	I	J	K	L	M
频率	8.167	1.492	2.782	4.253	12.702	2.228	2.015	6.094	6.966	0.153	0.722	4.025	2.406
字母	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
频率	6.749	7.507	1.929	0.095	5.987	6.327	9.056	2.758	0.978	2.360	0.150	1.974	0.074

- ✎ 转化为单表替代密码的破译问题
 - ✎ 确定密钥长度
 - ✎ 根据密钥长度对密文进行分组，每组是一个单表



✎ 多表替代会改变每个字符的频度，破译的关键是密钥的长度

✎ 基本思想：按照正确密钥长度对多表替代的密文进行分组后，每个分组都是一个单表，确定 d 是关键

✎ 转化为单表替代密码的破译问题

- ✎ 如何确定密钥长度d?
- ✎ 根据密钥长度对密文进行分组时，每组是一个单表

粗糙度

$$\begin{aligned} M.R &= \sum_{i=0}^{25} \left(p_i - \frac{1}{26}\right)^2 \\ &= \sum_{i=0}^{25} p_i^2 - \frac{1}{26} = \sum_{i=0}^{25} p_i^2 - 0.0385 \end{aligned}$$

重合指数

$$IC = \sum_{i=0}^{n-1} p_i^2$$

章节安排

Outline



线性反馈序列密码



非线性反馈序列密码

章节安排

Outline



线性反馈序列密码



非线性反馈序列密码



3.1 序列密码—概念

- ✦ 序列密码又称为**流密码** (Stream Cipher)
- ✦ 是一类重要的**对称密码算法**
- ✦ 源于1917年Gilbert Vernam提出的 “**一次一密**” 密码体制
- ✦ **工程易实现，效率高**

明文	1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 1
密钥	1 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 1 0
密文	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 1

✦ 序列密码又称为流密码，是最接近 “一次一密” 的密码体制

✦ 序列密码加密环节简单，所以效率高

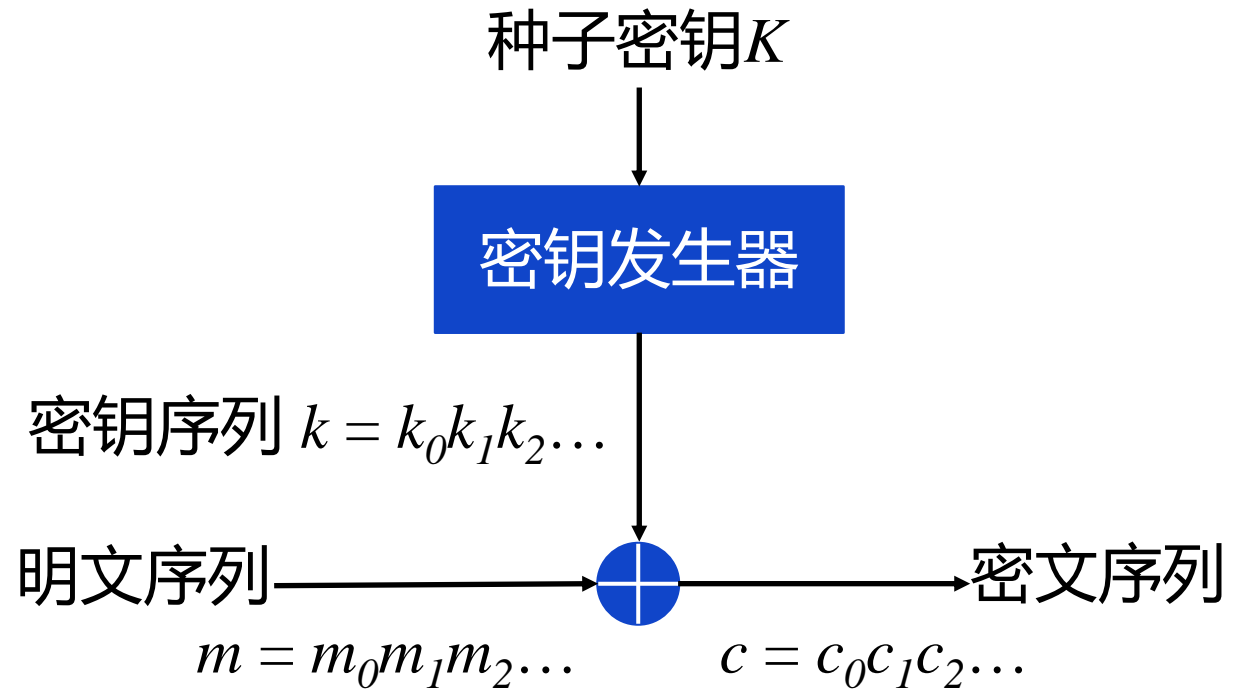
3.1 序列密码—原理

- 将一串较短的**种子密钥** K 通过**密钥流发生器**扩展成足够长的**伪随机**

密钥流 $k = k_0k_1k_2\dots$

- 加密变换**: $c_i = m_i \oplus k_i$

- 解密变换**: $m_i = c_i \oplus k_i$

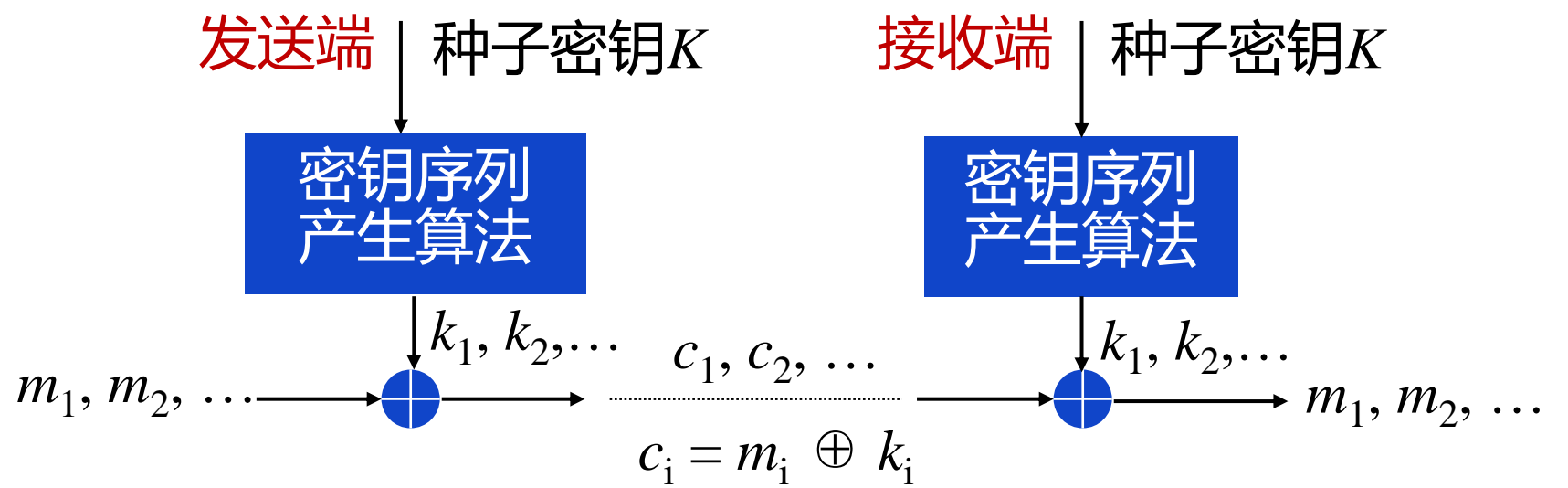




- ✎ 密钥序列产生算法与明文（密文）无关，所产生的密钥序列也与明文（密文）无关
- ✎ 通信双方必须保持精确的同步，收方才能正确解密
- ✎ 例如，若通信中丢失或增加了一个密文字符，则收方的解密将一直错误，直至重新同步

加密变换： $c_i = m_i \oplus k_i$

解密变换： $m_i = c_i \oplus k_i$



设密文失步

$c = c_1, c_3, c_4, \dots c_{n-1}, c_n$	(c_2 丢失)
$\oplus k = k_1, k_2, k_3, \dots k_{n-2}, k_{n-1}$	(密钥同步)
<hr/>	
$m = m_1, \times, \times, \dots \times, \times$	(m_1 后明文全错)

同步的概念：明文、密钥和密文字符的顺序准确对应，不对应即失步

- ✦ 对失步的敏感性，使我们能够容易检测插入、删除、重播等主动攻击
- ✦ 当通信中某些密文字符产生了错误（0翻转成1或1翻转成0），只影响相应字符的解密，不影响其它字符

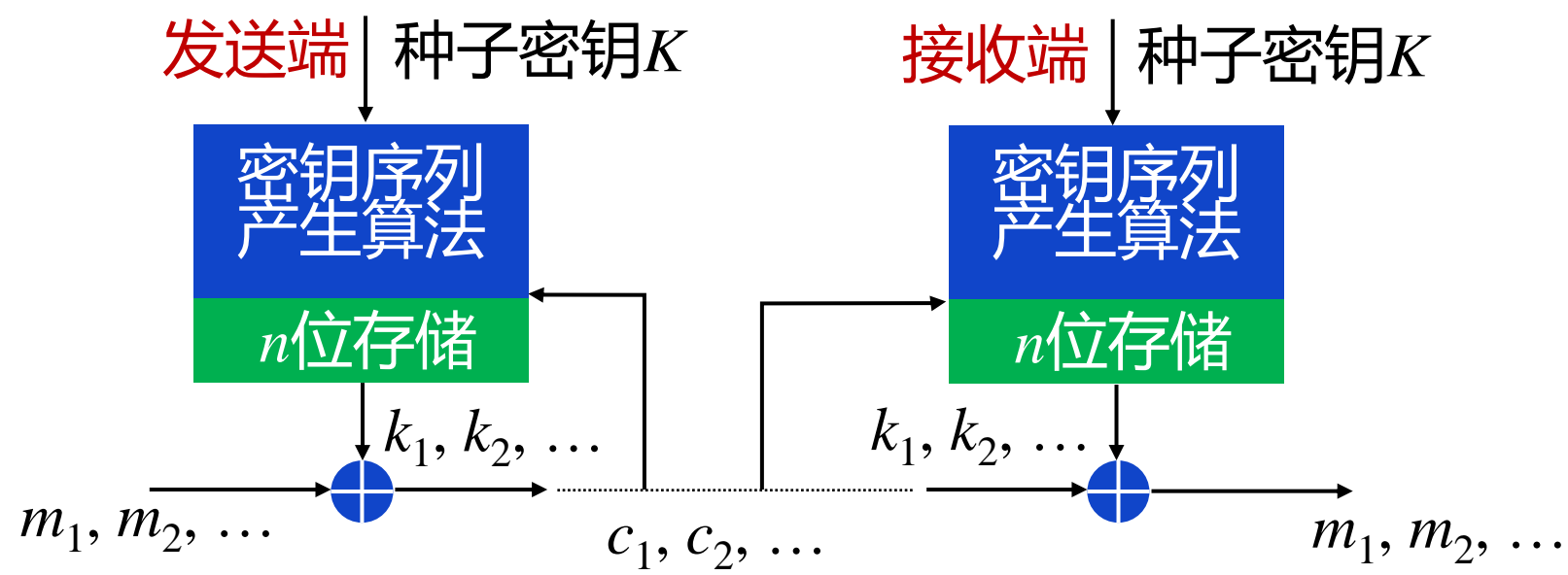
设密文错误 $c = c_1, c_2, c_3, \dots, c_{n-1}, c_n$ (c_2 错)

$\oplus k = k_1, k_2, k_3, \dots, k_{n-1}, k_n$ (密钥同步)

$m = m_1, \times, m_3, \dots, m_{n-1}, m_n$ (仅 m_2 错)

错误与失步是不同的概念！

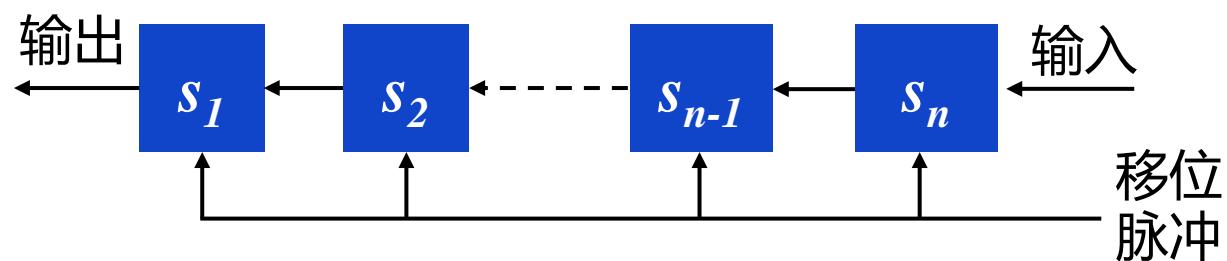
- ✎ 密钥序列产生算法与明文（密文）**相关**，则所产生的密钥序列与明文（密文）**相关**
- ✎ 加解密会造成**错误传播**。但错误传播有界，在错误过去之后恢复正确
- ✎ 例如， n 位密钥发生器：
 - ✎ 若在**加密时**一位密文错误将影响连续后面 n 个密文错误，在此之后恢复正确
 - ✎ 若在**解密时**一位密文错误将影响连续后面 n 个明文错误，在此之后恢复正确



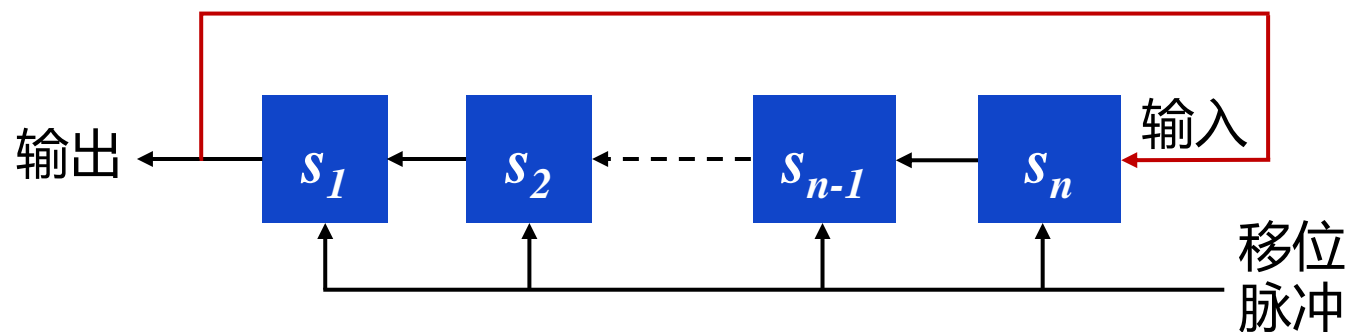
c_i 的错误将影响n位

- ✎ 错误的有界传播，原因在于n位寄存器，n位寄存器中有1比特发生错误，解密结果就会出错
- ✎ n位寄存器的值全部正确时，密钥应该是正确的

例1: 移位寄存器 (Shift Register)

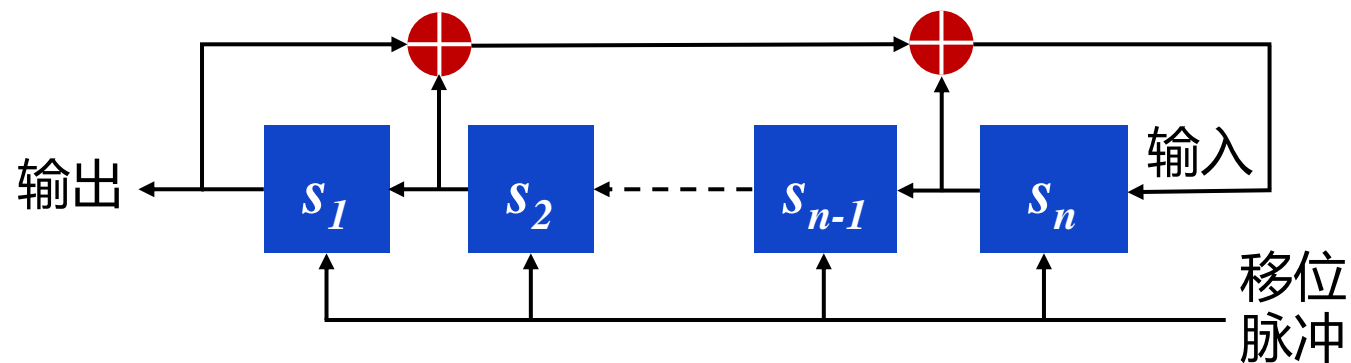


例2: 增加反馈, 反馈移位寄存器 (Feedback Shift Register)

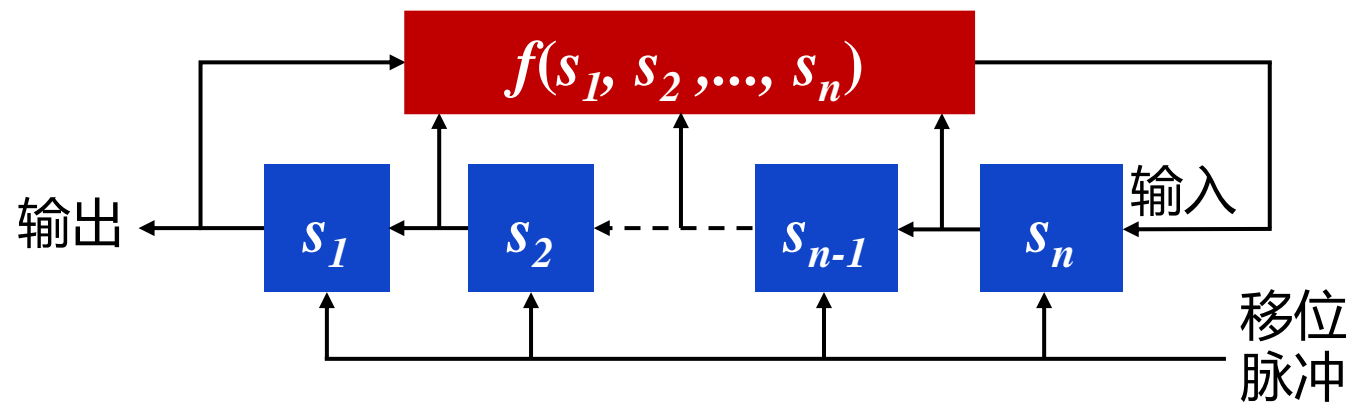


✎ 例3: 线性反馈移位寄存器 (Linear Feedback Shift Register)

✎ 特点: 增加了线性运算部件



- 称每一时刻移位寄存器的取值 $S = (s_1, s_2, \dots, s_n)$ 为一个状态
- 移位寄存器在输出同时将新值要送入 s_n ，其值要通过函数 $f(s_1, s_2, \dots, s_n)$ 计算产生，该函数为反馈函数
- 若反馈函数是 s_1, s_2, \dots, s_n 的线性函数，则称为线性反馈移位寄存器，否则称为非线性反馈移位寄存器

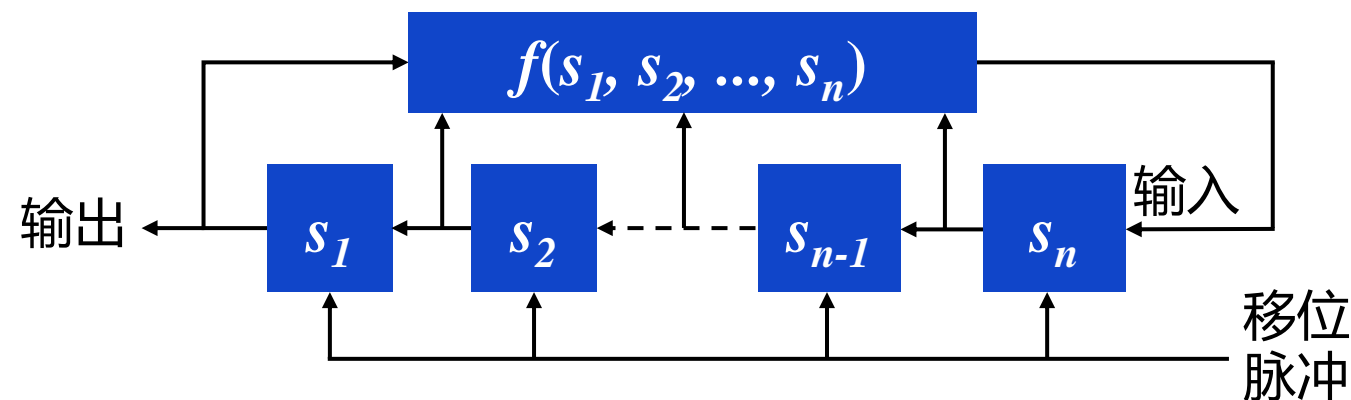


✎ 设 $f(s_1, s_2, \dots, s_n)$ 为线性函数，则可写成

$$f(s_1, s_2, \dots, s_n) = g_n s_1 + g_{n-1} s_2 + \dots + g_1 s_n$$

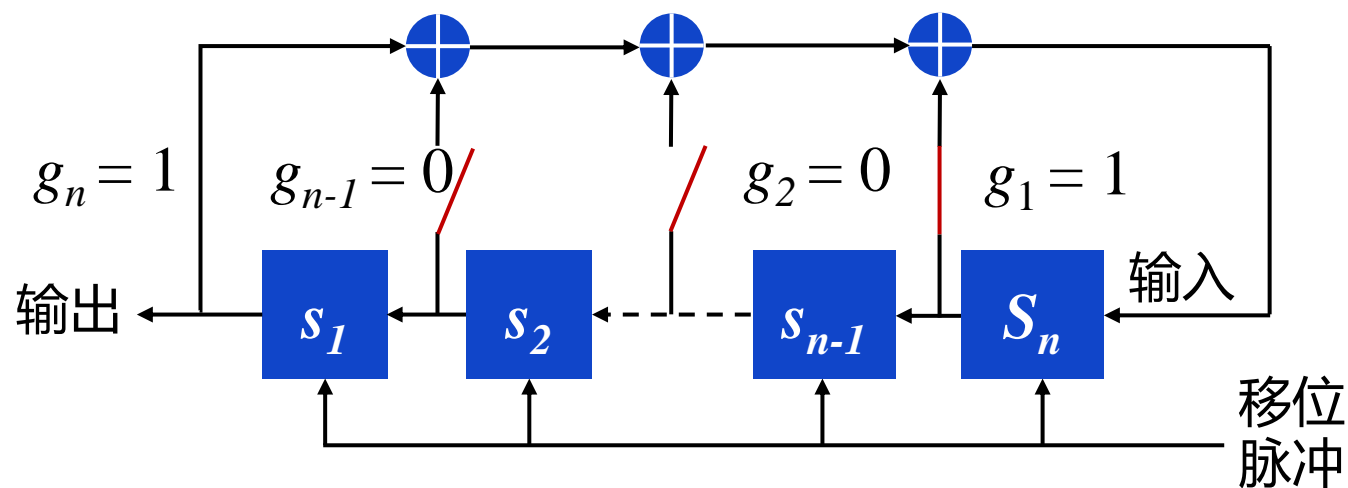
其中， g_1, g_2, \dots, g_n 称为反馈系数

✎ 在 $GF(2)$ 的情况下，式中的+即为 \oplus ，反馈系数 $g_i \in GF(2)$



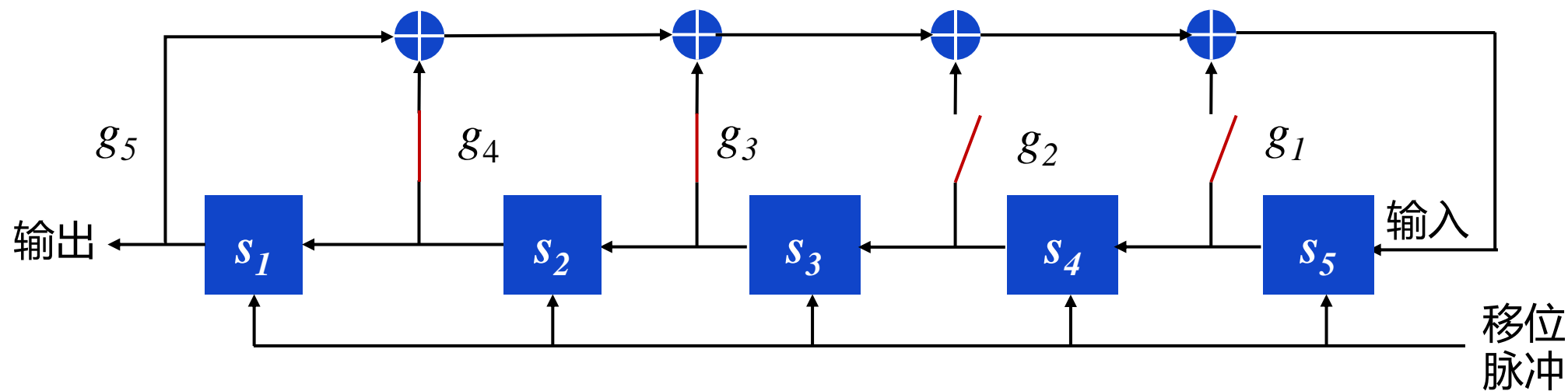
- ✎ 若 $g_i = 0$, 则表示式中的 $g_i s_i$ 项不存在, 即 s_i 不连接。同理 $g_i = 1$ 表示 s_i 连接
- ✎ g_i 相当于一个选择开关
- ✎ 连接多项式, 其中 $g_n = 1$

$$g(x) = g_n x^n + g_{n-1} x^{n-1} + \dots + g_1 x + 1$$



连接多项式

$$g(x) = g_n x^n + g_{n-1} x^{n-1} + \dots + g_1 x + 1 \quad \longrightarrow \quad g(x) = x^5 + x^4 + x^3 + 1$$



- ✎ n 级线性移位寄存器最多有 2^n 个不同的状态
- ✎ 若其初始状态为零，则其后续状态恒为零。若其初始状态不为零，则其后续状态也不为零
- ✎ 因此， n 级线性移位寄存器的状态周期 $\leq 2^n - 1$ ，其输出序列的周期 $\leq 2^n - 1$
- ✎ 选择合适的本原多项式（对应于反馈系数）可使线性移位寄存器的输出序列周期达到最大值 $2^n - 1$
- ✎ 称此时的输出序列为最大长度线性移位寄存器输出序列，简称为 m 序列

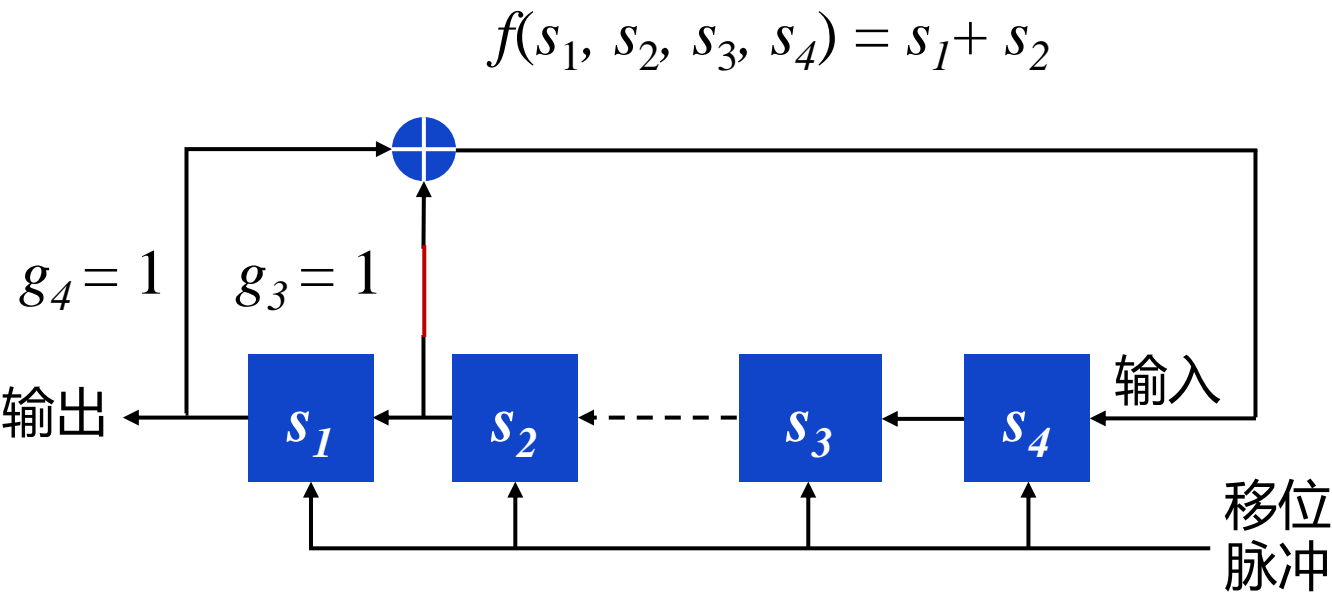
- 仅当连接多项式 $g(x)$ 为本原多项式时，其线性移位寄存器的输出序列为 m 序列

$$g(x) = g_n x^n + g_{n-1} x^{n-1} + \dots + g_1 x + 1$$

- 已经证明，对于任意的正整数 n ，至少存在一个 n 次本原多项式，而且存在有效的产生算法

设 $f(x)$ 为 $GF(2)$ 上的多项式，使 $f(x) \mid x^p - 1$ 的最小正整数 p 称为 $f(x)$ 的周期。如果 $f(x)$ 的次数为 n ，且其周期为 $2^n - 1$ ，则称 $f(x)$ 为本原多项式

✎ 如下图所示的线性反馈移位寄存器的输出序列为000100110101111..., 它的周期为 $2^4 - 1 = 15$



状态序列

1	0001	9	0101
2	0010	10	1011
3	0100	11	0111
4	1001	12	1111
5	0011	13	1110
6	0110	14	1100
7	1101	15	1000
8	1010		


```
// this is 32-bit long LFSR
```

```
module lfsr32(clk, reset, lfsr);
```

```
  input clk, reset;
```

```
  output reg [31:0] lfsr;
```

```
  wire d0;
```

```
  xnor(d0, lfsr[31], lfsr[21], lfsr[1], lfsr[0]);
```

```
  always @(posedge clk, posedge reset) begin
```

```
    if(reset) begin
```

```
      lfsr <= 32'h00000001;
```

```
    end
```

```
    else begin
```

```
      lfsr <= {lfsr[30:0], d0};
```

```
    end
```

```
  end
```

```
endmodule
```

xnor为同或运算

lfsr <= {lfsr[30:0], d0}是移位操作

✎ 设 m 序列线性移位寄存器的状态为

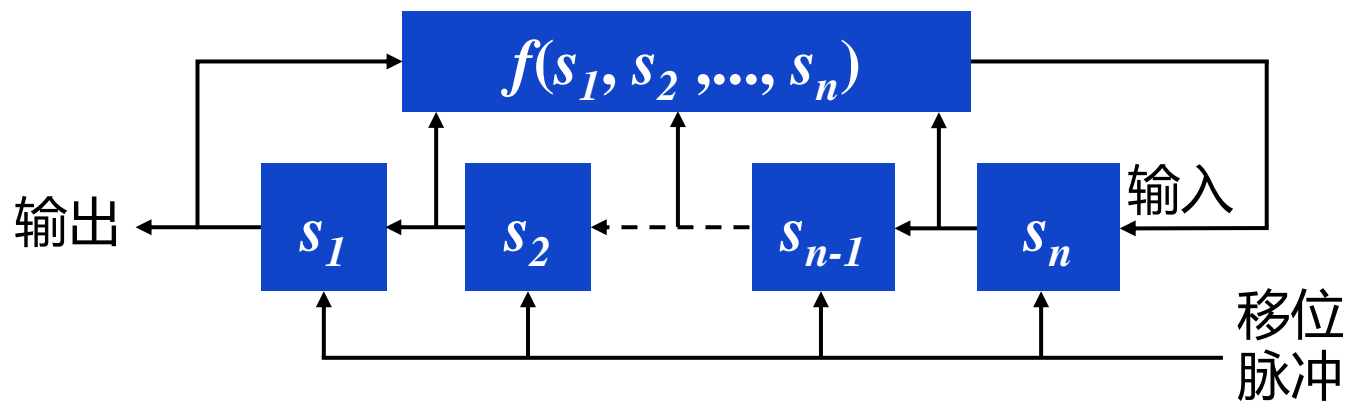
$$S = (s_1, s_2, \dots, s_n)$$

✎ 下一状态为 $S' = (s'_1, s'_2, \dots, s'_n)$, 其中

$$\begin{aligned} s'_1 &= s_2 \\ s'_2 &= s_3 \\ &\dots \end{aligned}$$

$$s'_{n-1} = s_n$$

$$s'_n = g_n s_1 + g_{n-1} s_2 + \dots + g_1 s_n$$



✎ 线性移位寄存器序列, 写成矩阵形式: $S' = H \times S \bmod 2$

$$S' = \begin{pmatrix} s'_1 \\ s'_2 \\ \vdots \\ s'_n \end{pmatrix} \quad H = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \dots & & & & \\ 0 & 0 & 0 & \dots & 1 \\ g_n & g_{n-1} & \dots & g_1 \end{pmatrix} \quad S = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix}$$

矩阵 H 称为连接多项式的伴侣矩阵

✎ 进一步假设攻击者知道了一段长 $2n$ 位的明密文对, 即

$$M = m_1, m_2, \dots, m_{2n}$$

$$C = c_1, c_2, \dots, c_{2n}$$

✎ 于是可求出一段长 $2n$ 位的密钥序列,

$$K = k_1, k_2, \dots, k_{2n}$$

$$k_i = m_i \oplus (m_i \oplus k_i) = m_i \oplus c_i$$

✎ 由此可以推出线性移位寄存器连续 $n+1$ 个状态 $S_1 \sim S_{n+1}$:

$$S_1 = (k_1, k_2, \dots, k_n)^T$$

$$S_2 = (k_2, k_3, \dots, k_{n+1})^T$$

...

$$S_{n+1} = (k_{n+1}, k_{n+2}, \dots, k_{2n})^T$$

✎ 构造矩阵

$$X = (S_1, S_2, \dots, S_n)$$

$$Y = (S_2, S_3, \dots, S_{n+1})$$

✎ 根据 $S' = H \times S \bmod 2$, 有

$$S_2 = H \times S_1$$

$$S_3 = H \times S_2$$

...

$$S_{n+1} = H \times S_n$$

✎ 于是

$$Y = H \times X \bmod 2$$

- 因为 m 序列的线性移位寄存器连续 n 个状态向量彼此线性无关，因此 X 矩阵为满秩矩阵，故存在逆矩阵 X^{-1} ，于是

$$H = Y \times X^{-1} \bmod 2$$

- 求出 H 矩阵，便确定出本原多项式 $g(x)$ ，从而完全确定线性移位寄存器的结构

- 例：m序列 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0

- 连续 4 个状态 1001, 0011, 0110, 1101 线性无关

求逆矩阵 X^{-1} 的计算复杂度为 $O(n^3)$ 。对于 $n = 1000$ 的线性移位寄存器序列密码，用每秒100万次的计算机，一天之内便可破译

章节安排

Outline



线性反馈序列密码



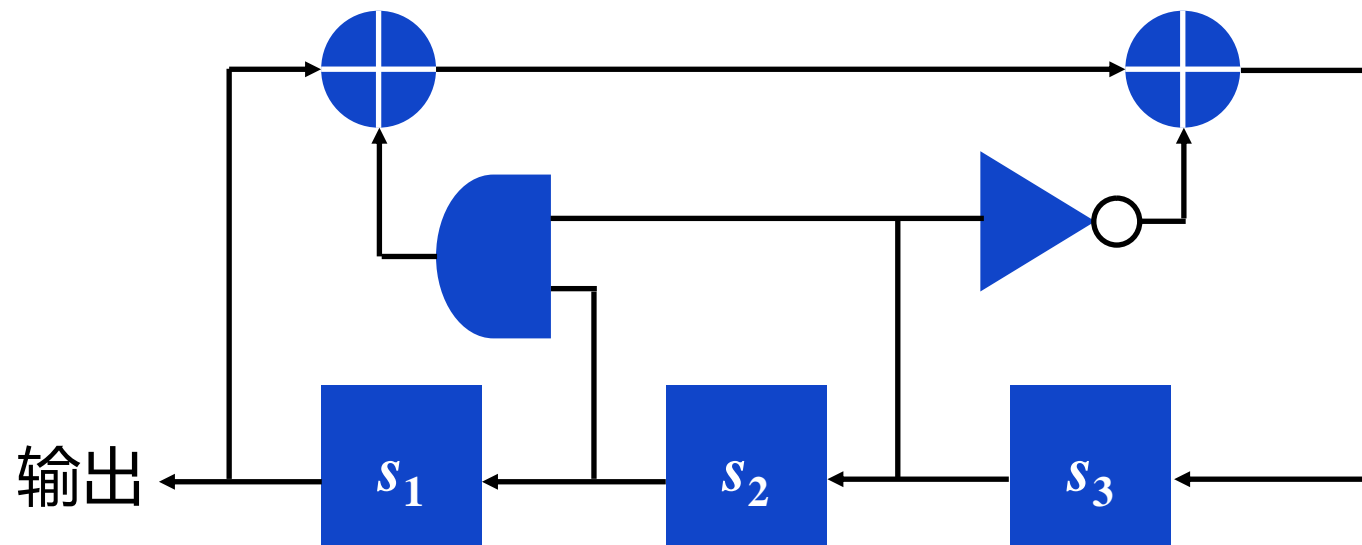
非线性反馈序列密码

- ✎ 线性移位寄存器序列密码在**已知明文攻击**下是可破译的，可破译的根本原因在于线性移位寄存器序列是**线性的**，这一事实促使人们向**非线性**领域探索
- ✎ 目前研究得比较充分的方法包括：
 - ✎ 非线性移位寄存器序列
 - ✎ 对线性移位寄存器序列进行非线性组合
 - ✎ 利用非线性分组码产生非线性序列

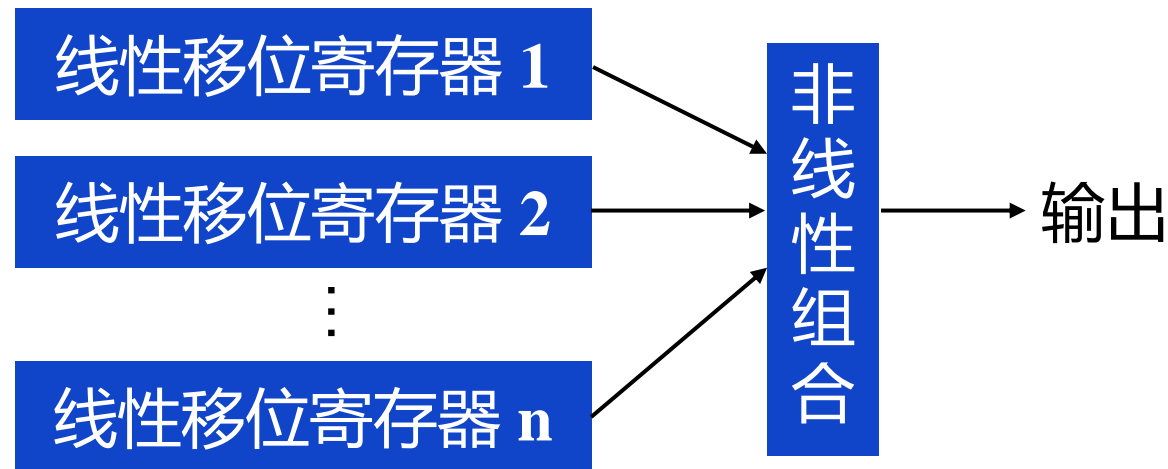
- ✎ 线性移位寄存器序列密码在已知明文攻击下是可破译的，可破译的根本原因在于线性移位寄存器序列是线性的，这一事实促使人们向非线性领域探索
- ✎ 目前研究得比较充分的方法包括：
 - ✎ 非线性移位寄存器序列
 - ✎ 对线性移位寄存器序列进行非线性组合
 - ✎ 利用非线性分组码产生非线性序列

令反馈函数 $f(s_1, s_2, \dots, s_n)$ 为非线性函数便构成非线性移位寄存器，其输出序列为非线性序列

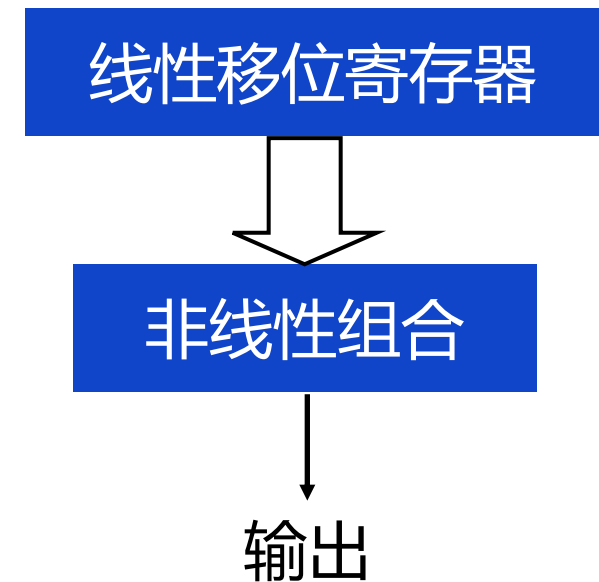
- ✎ 例：令 $n = 3$, $f(s_1, s_2, s_3) = s_1 \oplus (s_3 \oplus 1) \oplus s_2 \cdot s_3$, 由于与运算 \cdot 为非线性运算, 故反馈函数为非线性反馈函数
- ✎ 其输出序列为10110100..., 为M序列(周期达到 $2^n = 8$)





- ✎ 对线性移位寄存器序列进行非线性组合
 - ✎ 非线性移位寄存器序列的研究比较困难
 - ✎ 线性移位寄存器序列的研究却比较充分和深入
 - ✎ 利用线性移位寄存器序列设计容易、随机性好等优点，**对一个或多个线性移位寄存器序列进行非线性组合可以获得良好的非线性序列**



- ✎ 对线性移位寄存器序列进行非线性组合
- ✎ 用线性移位寄存器作为驱动源，来驱动非线性电路产生非线性序列。用线性移位寄存器序列来确保所产生序列的长周期和均匀性
- ✎ 用非线性电路来确保输出序列的非线性和其它密码性质。通常称这里的非线性电路为前馈电路，称这种输出序列为前馈序列



-  http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf
-  <https://www.cnblogs.com/yhsy1002/p/6888895.html>

```
// this is 138-bit long LFSR
module lfsr(clk, reset, lfsr);
    input clk, reset;
    output reg [137:0] lfsr;
    wire d0;

    xnor(d0, lfsr[137], lfsr[136], lfsr[130], lfsr[129]);

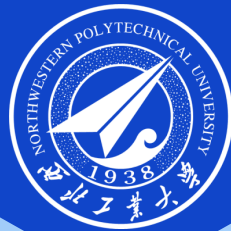
    always @(posedge clk, posedge reset) begin

        if(reset)
            lfsr <= 0;
        else
            lfsr <= {lfsr[136:0], d0};
        end

    endmodule
```

Table 3: Taps for Maximum-Length LFSR Counters

n	XNOR from	n	XNOR from	n	XNOR from	n	XNOR from
3	3,2	45	45,44,42,41	87	87,74	129	129,124
4	4,3	46	46,45,26,25	88	88,87,17,16	130	130,127
5	5,3	47	47,42	89	89,51	131	131,130,84,83
6	6,5	48	48,47,21,20	90	90,89,72,71	132	132,103
7	7,6	49	49,40	91	91,90,8,7	133	133,132,82,81
8	8,6,5,4	50	50,49,24,23	92	92,91,80,79	134	134,77
9	9,5	51	51,50,36,35	93	93,91	135	135,124
10	10,7	52	52,49	94	94,73	136	136,135,11,10
11	11,9	53	53,52,38,37	95	95,84	137	137,116
12	12,6,4,1	54	54,53,18,17	96	96,94,49,47	138	138,137,131,130
13	13,4,3,1	55	55,31	97	97,91	139	139,136,134,131
14	14,5,3,1	56	56,55,35,34	98	98,87	140	140,111
15	15,14	57	57,50	99	99,97,54,52	141	141,140,110,109
16	16,15,13,4	58	58,39	100	100,63	142	142,121
17	17,14	59	59,58,38,37	101	101,100,95,94	143	143,142,123,122
18	18,11	60	60,59	102	102,101,36,35	144	144,143,75,74
19	19,6,2,1	61	61,60,46,45	103	103,94	145	145,93
20	20,17	62	62,61,6,5	104	104,103,94,93	146	146,145,87,86
21	21,19	63	63,62	105	105,89	147	147,146,110,109
22	22,21	64	64,63,61,60	106	106,91	148	148,121
23	23,18	65	65,47	107	107,105,44,42	149	149,148,40,39
24	24,23,22,17	66	66,65,57,56	108	108,77	150	150,97
25	25,22	67	67,66,58,57	109	109,108,103,102	151	151,148
26	26,6,2,1	68	68,59	110	110,109,98,97	152	152,151,87,86
27	27,5,2,1	69	69,67,42,40	111	111,101	153	153,152
28	28,25	70	70,69,55,54	112	112,110,69,67	154	154,152,27,25
29	29,27	71	71,65	113	113,104	155	155,154,124,123
30	30,6,4,1	72	72,66,25,19	114	114,113,33,32	156	156,155,41,40
31	31,28	73	73,48	115	115,114,101,100	157	157,156,131,130
32	32,22,2,1	74	74,73,59,58	116	116,115,46,45	158	158,157,132,131
33	33,20	75	75,74,65,64	117	117,115,99,97	159	159,128
34	34,27,2,1	76	76,75,41,40	118	118,85	160	160,159,142,141
35	35,33	77	77,76,47,46	119	119,111	161	161,143
36	36,25	78	78,77,59,58	120	120,113,9,2	162	162,161,75,74
37	37,5,4,3,2,1	79	79,70	121	121,103	163	163,162,104,103
38	38,6,5,1	80	80,79,43,42	122	122,121,63,62	164	164,163,151,150
39	39,35	81	81,77	123	123,121	165	165,164,135,134
40	40,38,21,19	82	82,79,47,44	124	124,87	166	166,165,128,127
41	41,38	83	83,82,38,37	125	125,124,18,17	167	167,161
42	42,41,20,19	84	84,71	126	126,125,90,89	168	168,166,153,151
43	43,42,38,37	85	85,84,58,57	127	127,126		
44	44,43,18,17	86	86,85,74,73	128	128,126,101,99		



感谢聆听!

THANK YOU FOR YOUR ATTENTION!