《现代密码学》第五讲

流密码（一）

上讲内容回顾

- 分组密码定义
- 分组密码的发展历史
- 保密系统的安全性分析及分组密码的攻击
- 数据加密标准（DES）算法介绍
- 高级加密标准（AES）算法介绍
- 中国无限局域网标准（SMS4）算法介绍
- 分组密码算法的运行模式

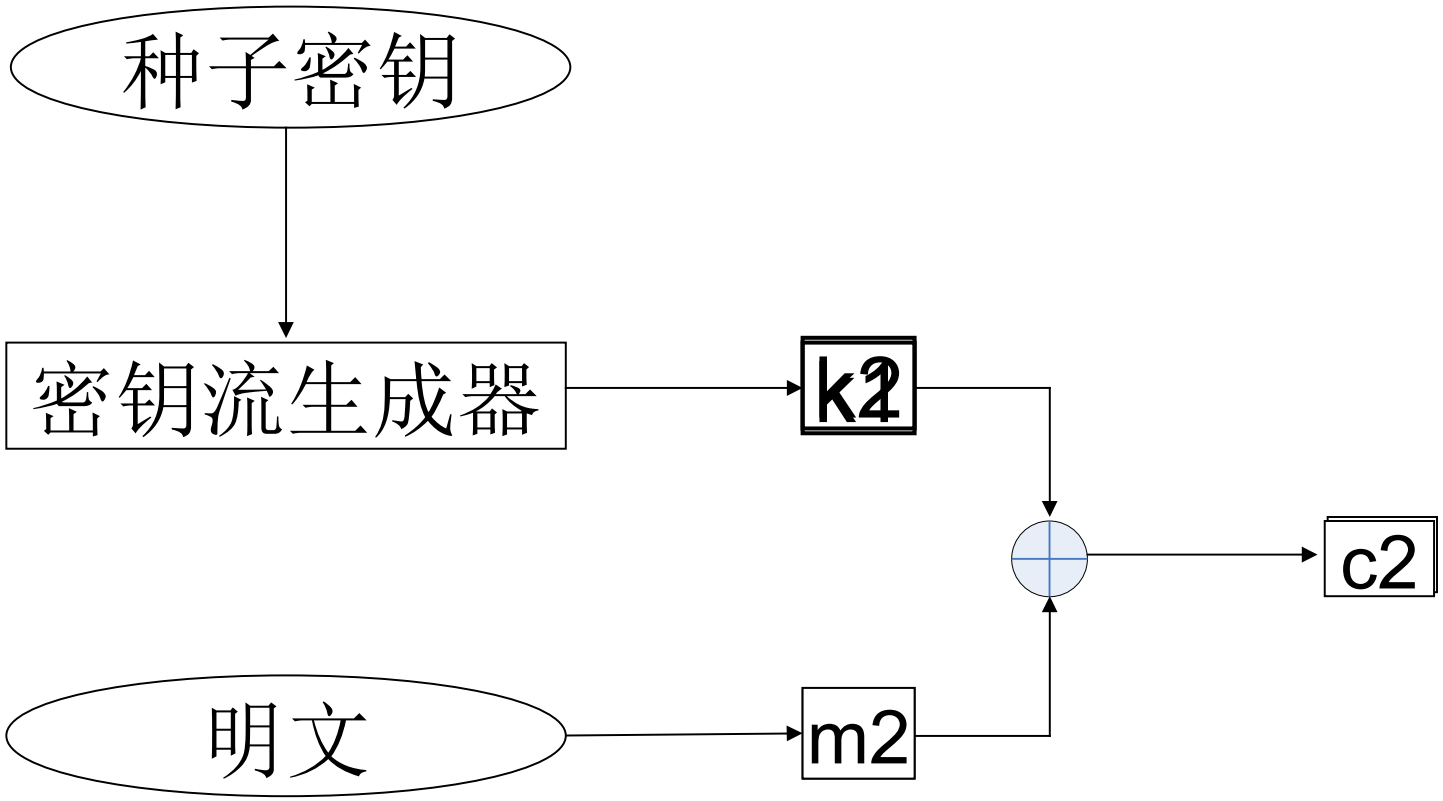
本章主要内容

- 流密码（序列密码）的思想起源
- 流密码技术的发展及分类
- 基于移位寄存器的流密码算法
- 其它流密码算法

本章主要内容

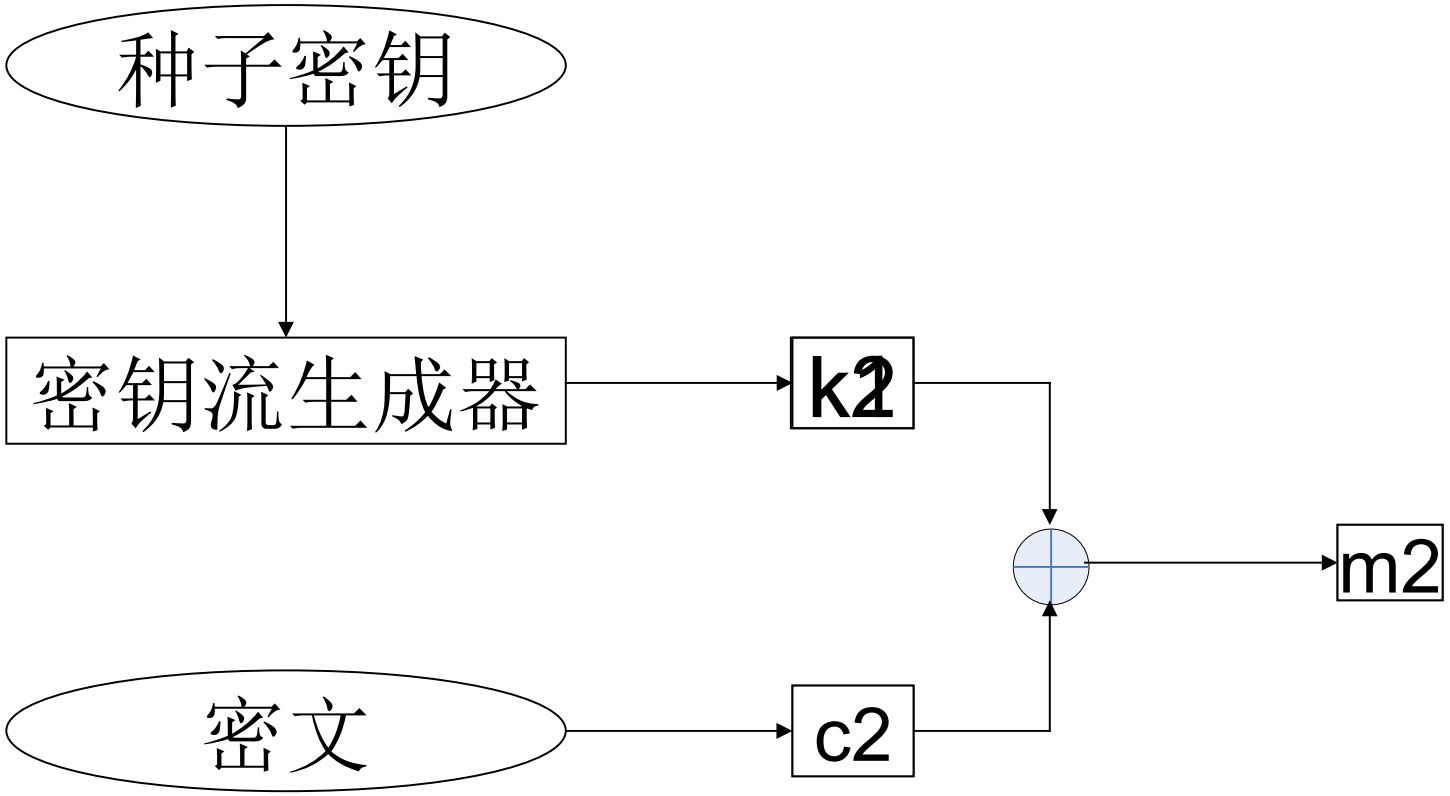
- 流密码（序列密码）的思想起源
- 流密码技术的发展及分类
- 基于移位寄存器的流密码算法
- 其它流密码算法

流密码的思想起源



加密过程

流密码的思想起源



解密过程

流密码的思想起源

- 设明文为 $m=m_1m_2\cdots m_i \in GF(2)$， $i>0$
- 设密钥为 $k=k_1k_2\cdots k_i \in GF(2)$， $i>0$
- 设密文为 $c=c_1c_2\cdots c_i \in GF(2)$， $i>0$
- 则加密变换为 $c_i = m_i+k_i \pmod 2$， $i>0$
- 则解密变换为 $m_i = c_i+k_i \pmod 2$， $i>0$

流密码的思想起源

- 思想起源：20世纪20年代的Vernam体制，即“一次一密”密码体制。香农利用信息论证明“一次一密”密码体制在理论上不可破译

- 由有限的种子密钥生成无限长的随机密钥序列

- 流密码研究内容——设计安全高效的伪随机序列发生器

流密码的思想起源

Golomb 伪随机性测试

周期为 r 的 0 — 1 序列的随机性公设如下：

- r 是奇数，则 0 — 1 序列 {si} 的一个周期内 0 的个数比 1 的个数多一个或少一个；若 r 是偶数，则 0 的个数与 1 的个数相等 .
- 在长度为 r 的周期内，长为 1 的游程的个数为游程总数的 1/2，长为 2 的游程的个数占游程总数的 $1/2^2, \cdots,$ 长为 c 的游程的个数占总游程的 $1/2^c$. 而且对于任意长度，0 的游程个数和 1 的游程个数相等 .

例：

0110111101 中，4 个游程长度为 1，1 个游程长度为 2，1 个游程长度为 4

流密码的思想起源

异相自相关函数是一个常数.

设一个周期为 r 的序列

$$a_1, a_2, \cdots, a_r, a_{r+1}, a_{r+2}, \cdots ,$$

将序列平移 T 位得到另外一个序列

$$a_T, a_{T+1}, \cdots a_{r+T}, a_{r+T+1}, \cdots ,$$

若 $a_i = a_{i+T}$, 则称对应第 i 位相等。

设两个序列相同位的个数为 n，不同位的个数为 d，

则 $R(T)=(n-d)/r$ 为自相关函数

本章主要内容

- 流密码（序列密码）的思想起源
- **流密码技术的发展及分类**
- 基于移位寄存器的流密码算法
- 其它流密码算法

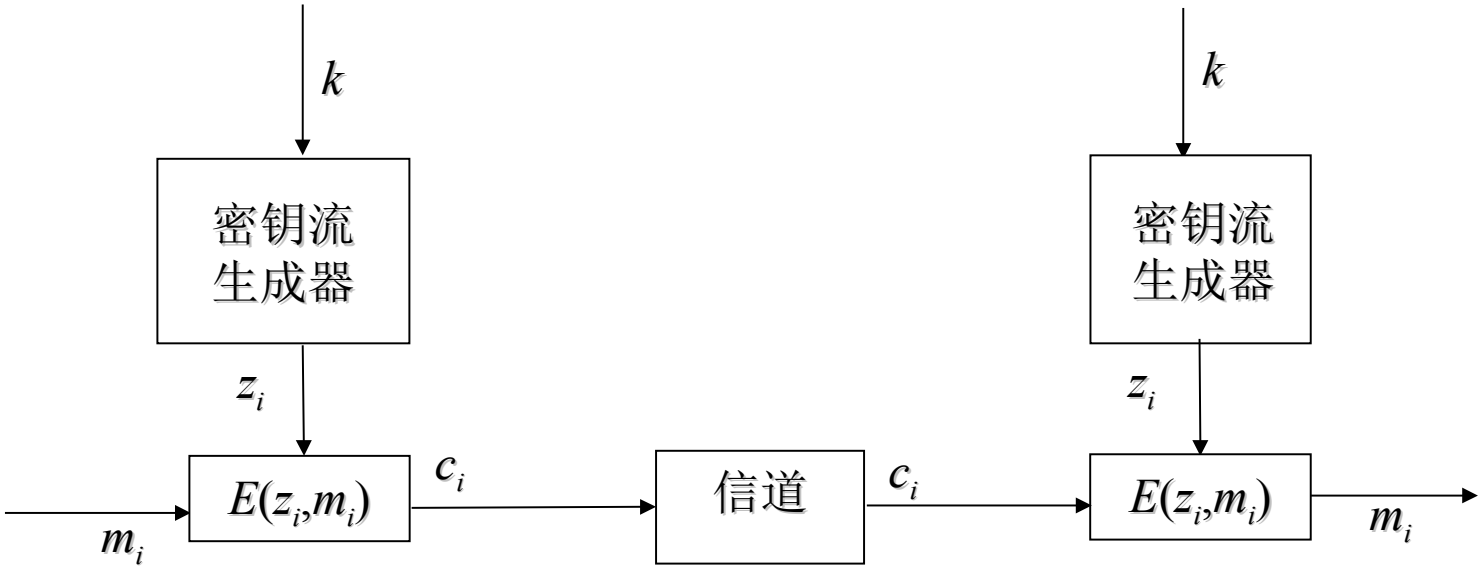
流密码技术的发展及分类

北京郵電大學 BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Profile 1 (SW)	Profile 2 (HW)
Profile 1 (SW)	Profile 2 (HW)
Profile 1 (SW)	Profile 2 (HW)
HC-128	~~P-FCSR H v2~~
Rabbit	Grain v1
Salsa20/12	MICKEY v2
SOSEMANUK	Trivium
Rabbit	**Moustique**
Salsa20	**Pomaranch** (Pomaranch Version 3)
SOSEMANUK	**Trivium**

2008 年 8 月更新为 7 个推荐算法 .

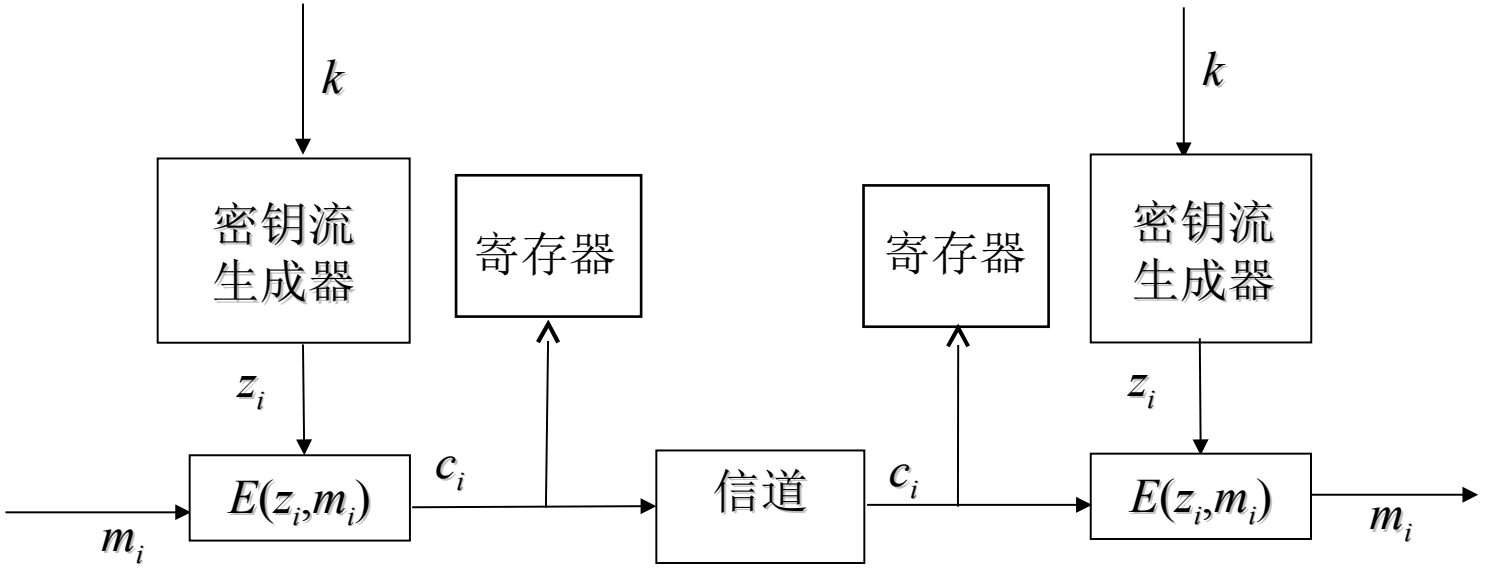
流密码技术的发展及分类



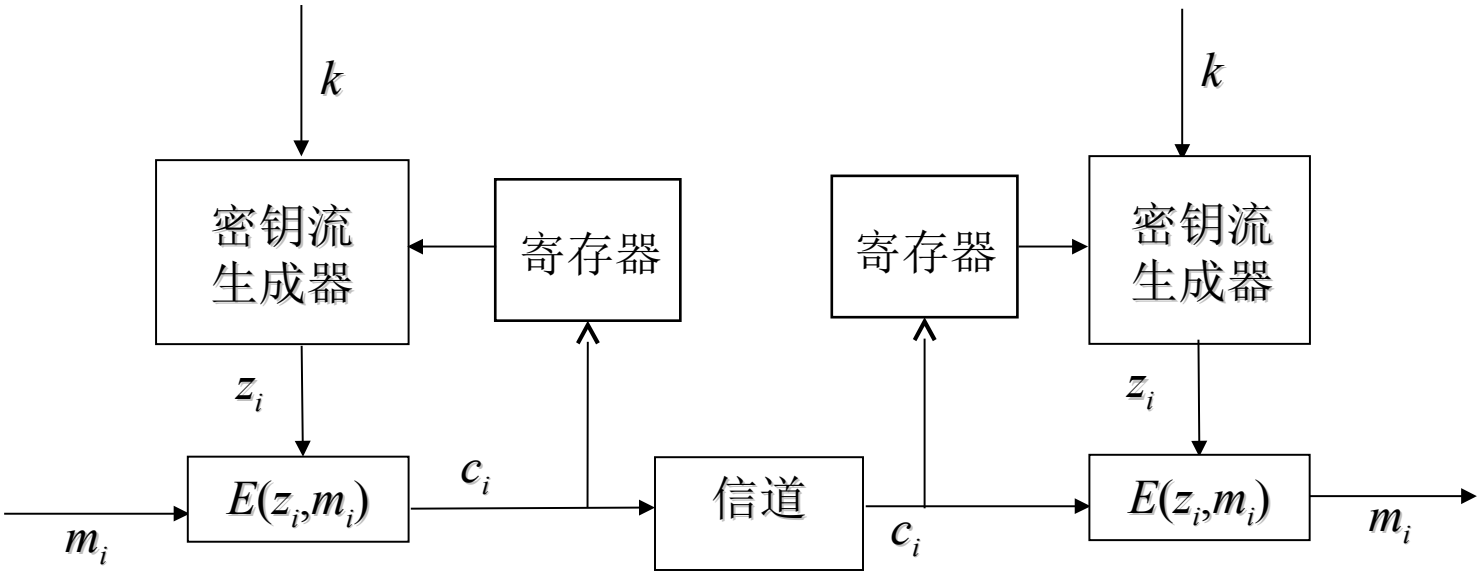
- 在同步流密码中，密（明）文符号是独立的，一个错误传输只会影响一个符号，不影响后面的符号。

- 缺点：一旦接收端和发送端的种子密钥和内部状态不同步，解密就会失败，两者必须立即借助外界手段重新建立同步。

流密码技术的发展及分类



流密码技术的发展及分类



自同步流密码的优点是即使接收端和发送端不同步，只要接收端能连续地正确地接受到 n 个密文符号，就能重新建立同步。因此自同步流密码具有有限的差错传播，且分析较同步流密码的分析困难得多

流密码技术的发展及分类

课堂练习： 假设 j=n/4, n 为分组长度

对于 DES， n=64， j=16；对 AES， n=128, j=32

CFB 模式为（　　）流密码？

OFB 模式为（　　）流密码？

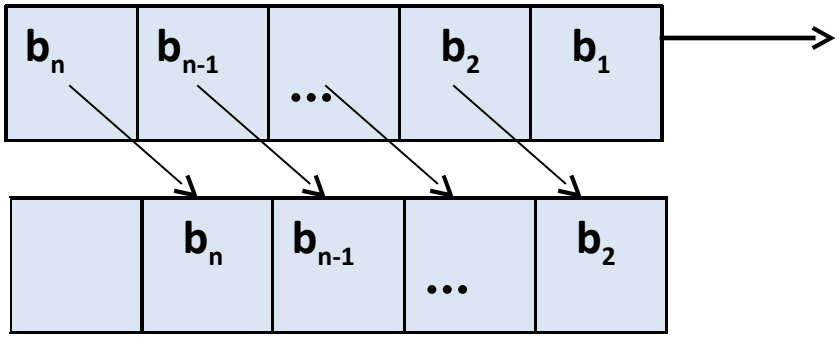
CTR 模式为（　　）流密码？

自同步、同步、同步

本章主要内容

- 流密码（序列密码）的思想起源
- 流密码技术的发展及分类
- **基于移位寄存器的流密码算法**
- 其它流密码算法

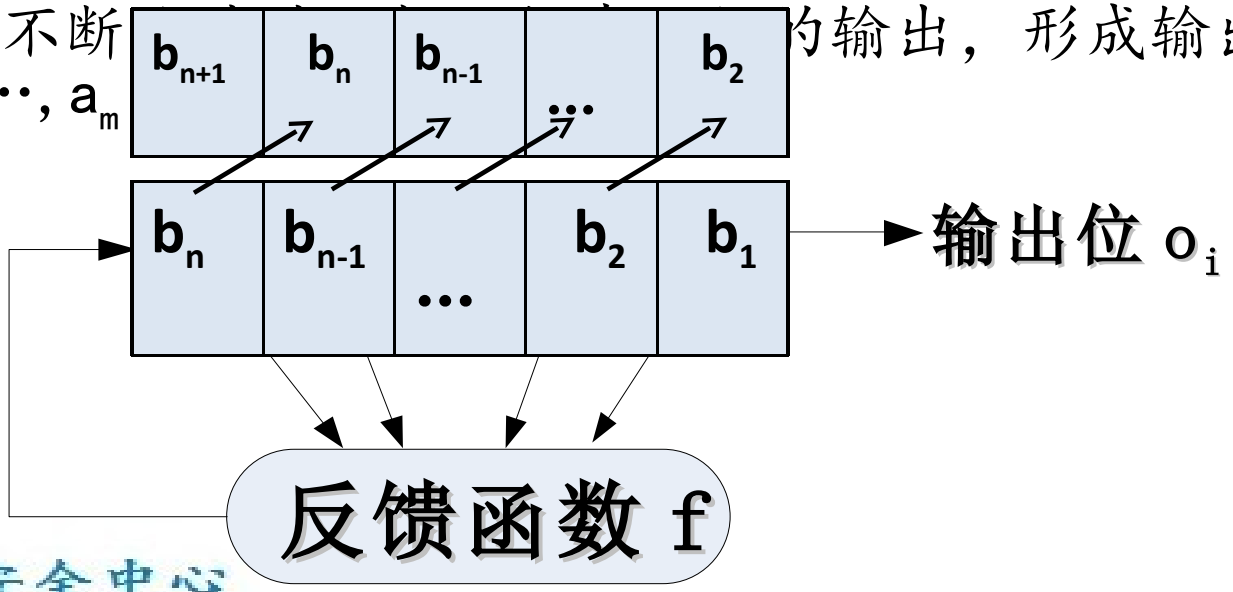
基于移位寄存器的算法



- 挪威政府的首席密码学家 Ernst Selmer 于 1965 年提出了移位寄存器理论，它是序列密码中研究随机密钥流的主要数学工具 .
- 移位寄存器是指有 n 个寄存器（称为 n- 级移位寄存器） $r_1, r_2, \cdots, r_n$ 从右到左排列，每个寄存器中能存放 1 位二进制数，所有寄存器中的数可以统一向右（或向左）移动 1 位，称为进动 1 拍 . 即 r_1 的值 (b_1) 右移 1 位后输出，然后 r_2 的值 (b_2) 送 r_1 ， r_3 的值 (b_3) 送 r_2, …… 最后，r_n 的值 (b_n) 送 r_{n-1}.

基于移位寄存器的算法

- 反馈移位寄存器（feedback shift register,FSR）是由n位的寄存器和反馈函数（feedback function）组成，如下图所示，n位的寄存器中的初始值称为移位寄存器的初态．
- 工作原理：移位寄存器中所有位的值右移1位，最右边的一个寄存器移出的值是输出位，最左边一个寄存器的值由反馈函数的输出值填充，此过程称为进动1拍． 反馈函数f是n个变元$(b_1, b_2, \cdots, b_n)$的布尔函数． 移位寄存器根据需要不断[illegible]的输出，形成输出序列$a_1, a_2, \cdots, a_m$



基于移位寄存器的算法

- 线性反馈移位寄存器 LFSR（linear feedback shift register）的反馈函数为线性函数

- 作为密钥流的序列 $\{a_i\}$ 的周期一定要大，因为密钥流的周期太小，利用很容易得到整个密钥流 $\{a_i\}$

- n 级 LFSR 输出的序列的周期 r 不依赖于寄存器的初始值，而依赖于特征多项式 $p(x)$

基于移位寄存器的算法

定义：

设 n 级 LFSR 的输出序列 $\{a_i\}$ 满足递推关系

$$a_{n+k} = c_n a_{n+k-1} \quad c_{n-1} a_{n+k-2} \quad \cdots \quad c_1 a_k \quad (k \quad 1).$$

这种递推关系可用一个一元高次多项式

$$f(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + 1$$

表示，称这个多项式为 LFSR 的**特征多项式**。

基于移位寄存器的算法

定义 设$f(x)$是$GF(2)$上的多项式，使$f(x)|(x^n-1)$的最小的n称为$f(x)$的**周期**或者**阶**。

例：

$f(x)=x^4+x^3+x^2+x+1$

为 GF(2) 上多项式，

以它为特征多项式的LFSR的输出序列周期。

$(x^5-1)=(x^4+x^3+x^2+x+1)(x-1)=f(x)(x-1)$

$f(x)\nmid x^n-1, \quad n<5$

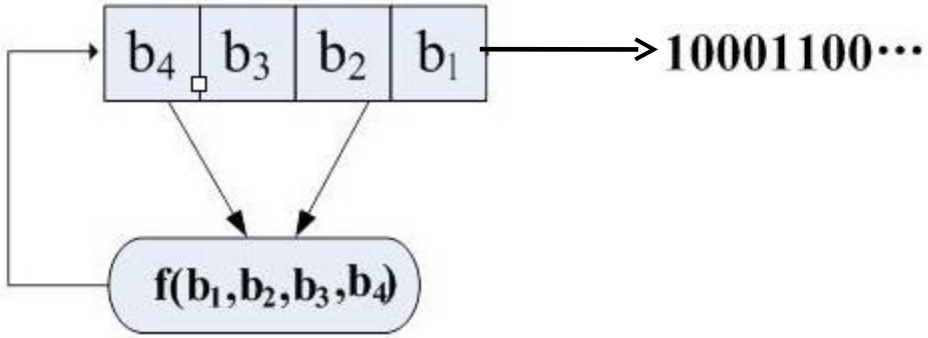
所以 f(x) 的周期为 5

基于移位寄存器的算法

解：对应的 n 级 LFSR 的反馈函数为

$$a_{4+k} = c_4 a_{3+k} \quad c_3 a_{2+k} \quad c_2 a_{1+k} \quad c_1 a_k \quad (k \quad 1).$$

状态		输出位
0001	→	1
1000	→	0
1100	→	0
0110	→	0
0011	→	1
0001	→	1
1000	→	0
1100	→	0



输出序列的周期为 5

基于移位寄存器的算法

- n 级 LFSR 输出的序列的最大周期是 2^n-1
- LFSR 的寄存器状态遍历 2^n-1 个非零状态
- 初始状态为全零，则输出序列为 0 的循环

定义 当 LFSR 的寄存器状态遍历 2^n-1 个非零状态时，序列的周期达到最大 2^n-1，这种

基于移位寄存器的算法

定义 若 n 次不可约多项式 $f(x)$ 的阶为 2^n-1，则称 $f(x)$ 为 n 次**本原多项式**。

定理 $\{a_i\}$ 是周期为 2^n-1 的 m- 序列的充要条件是其特征多项式 f(x) 为 n 阶本原多项式

基于移位寄存器的算法

一个 3- 级的反馈移位寄存器，反馈函数 $f(x)= b_3 \oplus b_1$ ，初态为 100，输出序列？

生成多项式为： $f(x) = x^3 + x + 1$

$$(x^7 - 1) = (x^4 + x^2 + x + 1)(x^3 + x + 1) = (x^4 + x^2 + x + 1) \bullet f(x)$$

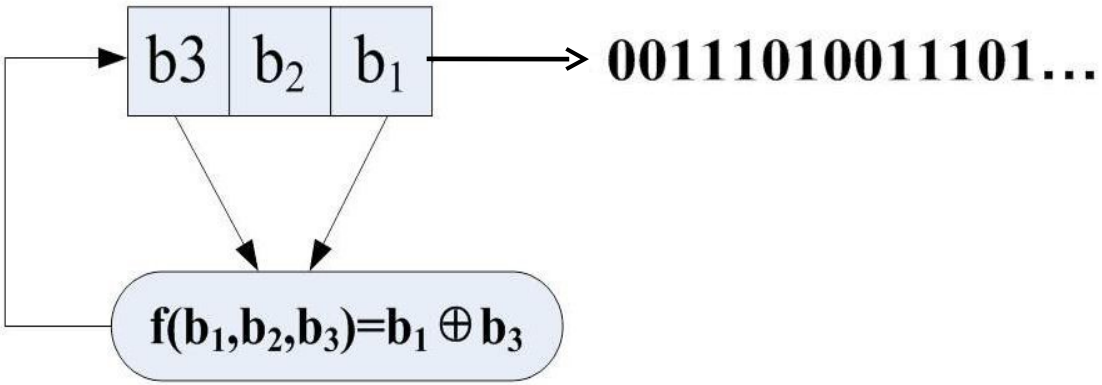
$$f(x) \nmid x^n - 1, \quad n < 7$$

所以 f(x) 的周期为 7

基于移位寄存器的算法

初态为 100 放入寄存器，输出序列情况如下：

状态位		输出
100	→	0
110	→	0
111	→	1
011	→	1
101	→	1
010	→	0
001	→	1
100	→	0



输出序列的周期为 7，是 m 序列

基于移位寄存器的算法

- 流密码的攻击
  - 攻击目的：确定整个密钥流 $\{k_i\}$
  - 攻击手段：
    - ✓ 惟密文
    - ✓ 已知明文
    - ✓ 选择明／密文
    - ✓ 自适应选择明／密文

基于移位寄存器的算法

1　若 LFSR 的反馈函数已知，破译者已知连续 n 位明密文对 $\{m_1, m_2, \cdots, m_n\}$ 和 $\{c_1, c_2, \cdots, c_n\}$，则可以推导出 n 比特密钥流 $\{k_1, k_2, \cdots, k_n\}$，

继而由反馈函数得到整个密钥流 $\{k_i\}$

2　已知明文攻击下，假设破译者已知了 2n 位明密文对 $K = \{k_1, k_2, \cdots, k_{2n}\}$，则可确定一段 2n 位长的密钥序列，由此可以完全确定 n 级反馈多项式的系数。

基于移位寄存器的算法

$$
\begin{aligned}
k_{n+1} &= k_1 b_n + k_2 b_{n-1} + \cdots + k_n b_1 \\
k_{n+2} &= k_2 b_n + k_3 b_{n-1} + \cdots + k_{n+1} b_1 \\
&\cdots \\
k_{2n} &= k_n b_n + k_{n+1} b_{n-1} + \cdots + k_{2n-1} b_1
\end{aligned}
$$

n 个线性方程包含 n 个未知数： $b_1, b_2, \ldots, b_n$ ，所以可惟一解出 b_i ($i=0$ ， 1 ， …， n)

从而可确定该线性反馈移位寄存器接下来的状态，也就能够得到余下的密钥序列。

基于移位寄存器的算法

例：

5 级线性反馈移位寄存器产生的密钥序列加密得到的明文串为 0110011111111001，对应的密文串为 1011010111110011 。求该 LFSR 的反馈函数。

解： 由明密文得相应的密钥序列为

1	1	0	1	0	0	1	0	0	0	0	1	0	1	0
k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}

利用前 10 个密钥序列比特建立如下方程：

基于移位寄存器的算法

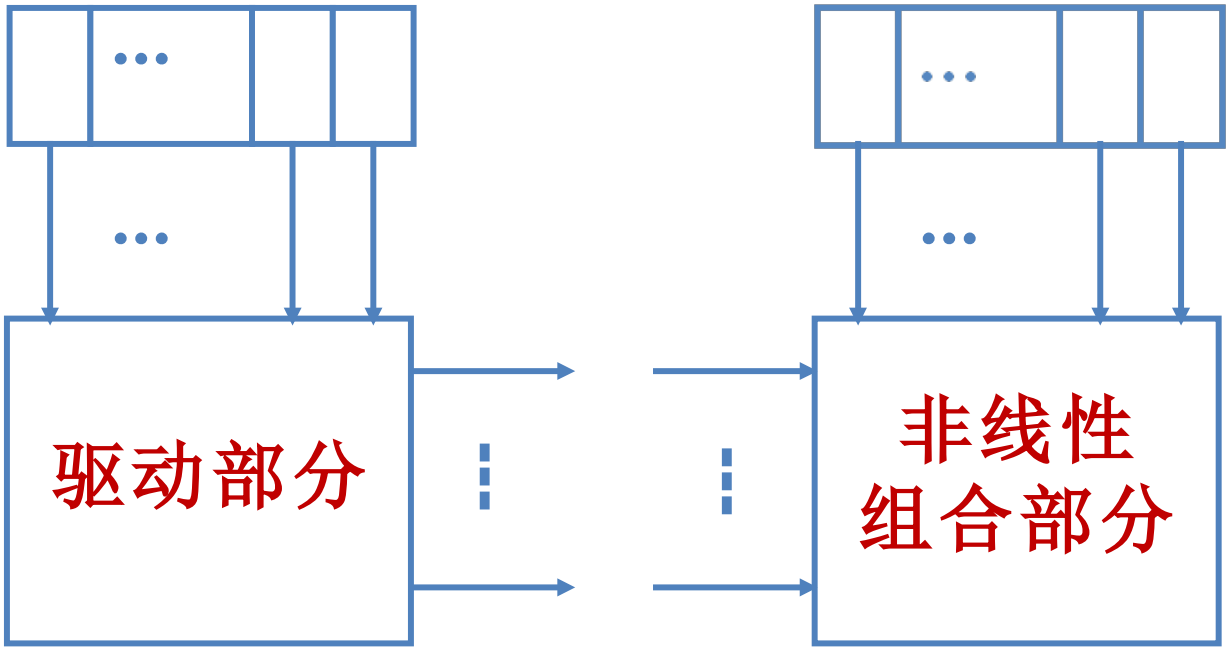
$$\begin{pmatrix} k_6 \\ k_7 \\ k_8 \\ k_9 \\ k \end{pmatrix} = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 \\ k_2 & k_3 & k_4 & k_5 & k_6 \\ k_3 & k_4 & k_5 & k_6 & k_7 \\ k_4 & k_5 & k_6 & k_7 & k_8 \\ k & k & k & k & k \end{pmatrix} \begin{pmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a \end{pmatrix} \Leftrightarrow \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \end{pmatrix}$$

反馈函数为 $k_{i+5} = a_5 k_i + a_2 k_{i+3} = k_i + k_{i+3}$

$$\Rightarrow \begin{pmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Leftrightarrow \begin{pmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

基于移位寄存器的算法

为了提高密钥流序列的线性复杂度，密钥生成器重中必须使用非线性函数． 为了便于分析， Ruppe 将密钥流生成器分成
两部分：驱动部分和非线性组合部分．



基于移位寄存器的算法

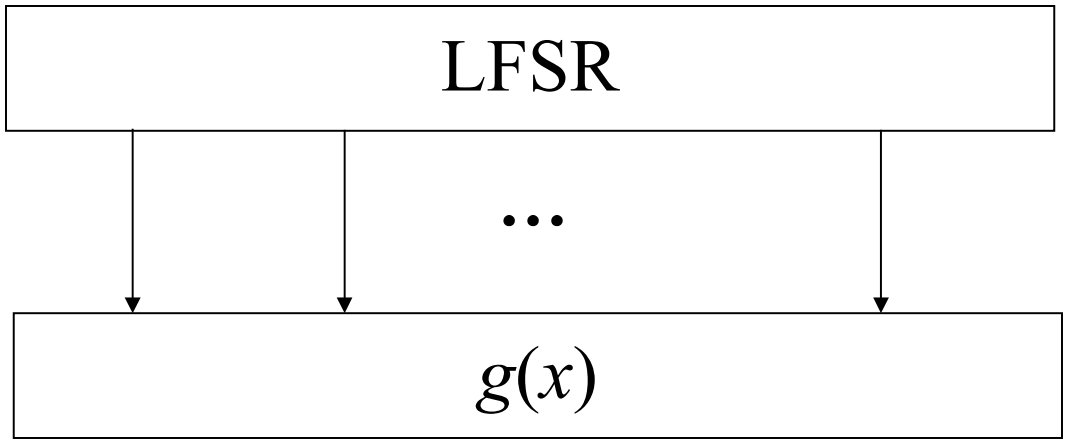
为了提高密钥流序列的线性复杂度，密钥生成器重中必须使用非线性函数． 为了便于分析， Ruppe 将密钥流生成器分成
两部分：驱动部分和非线性组合部分．

一般来说，驱动部分可由 m- 序列或其他长周期的 LFSR 序列组
成，用于控制密钥流生成器的状态序列，并为非线性组合部分提供伪随机性质良好的序列；非线性组合部分利用驱动部分生成的状态序列生成满足要求的密码特性好的密钥流序列．

密钥流生成器机理符合 Shannon 的“扩散”和“混淆”两条密码学的基本原则． 驱动部分利用 LFSR 将密钥 k 扩散成周期
很大的状态序列，而状态序列与密钥 k 间的关系经非线性组

基于移位寄存器的算法

滤波生成器又叫前馈生成器，一般由 LFSR 和滤波前馈）函数两部分组成．LFSR 可以是一个，也可以是几个，它们输出的序列共同作为滤波函数的输入．

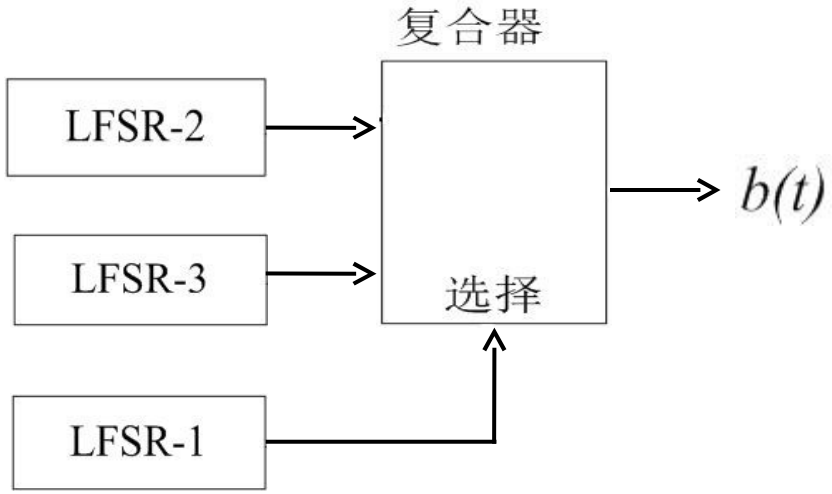


滤波函数要求具有很好的非线性性质，以增强生成器的抗攻击能力．

基于移位寄存器的算法

- **Geffe 序列发生器**

两个 LFSR 作为复合器的输人，第三个 LFSR 控制复合器的输出



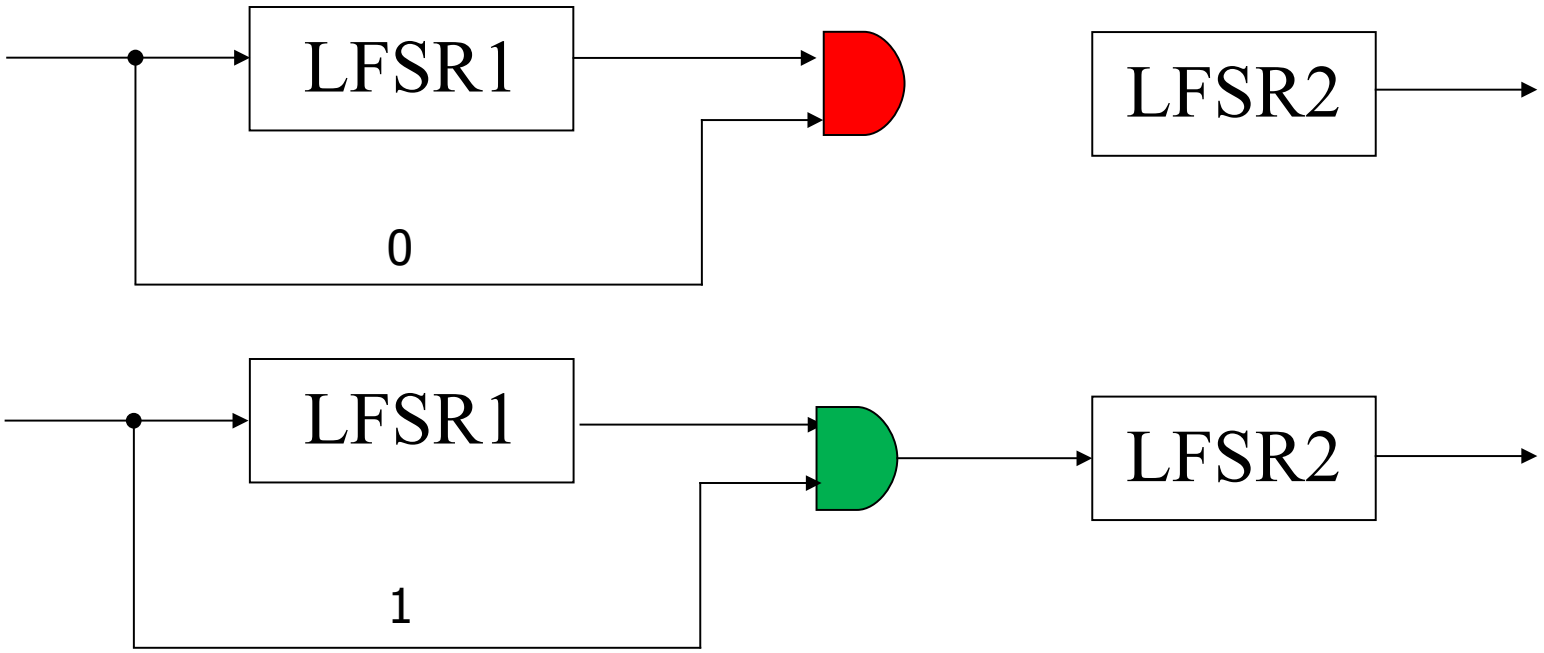
如果 a_1，a_2，和 a_3 是三个 LFSR 的输出，则 Geffe 发

生器的输出表示为：

$$b = (a_1 \wedge a_2) \oplus (\neg a_1 \wedge a_3) = (a_1 \wedge a_2) \oplus (a_1 \wedge a_3) \oplus a_3$$

基于移位寄存器的算法

- **钟控生成器**是由一个或几个 FSR 输出序列，控制一个 FSR 的时钟。最简单的钟控生成器是用一个 LFSR 控制另一个 LFSR 的时钟脉冲，如图：



当 LFSR1 输出 1 时，时钟脉冲通过与门使 LFSR2 进行一次移位，从而生成下一位；当 LFSR1 输出 0 时，时钟脉冲无法通过与门使 LFSR2 移位（走），从而 LFSR2 重复输出前一位（停）。因此，这种钟控生成器也被形象地称之为**走停生成器**（Sstop-and-Ggo generator）。

基于移位寄存器的算法

A5 算法是 GSM 系统中主要使用的序列密码加密算法，

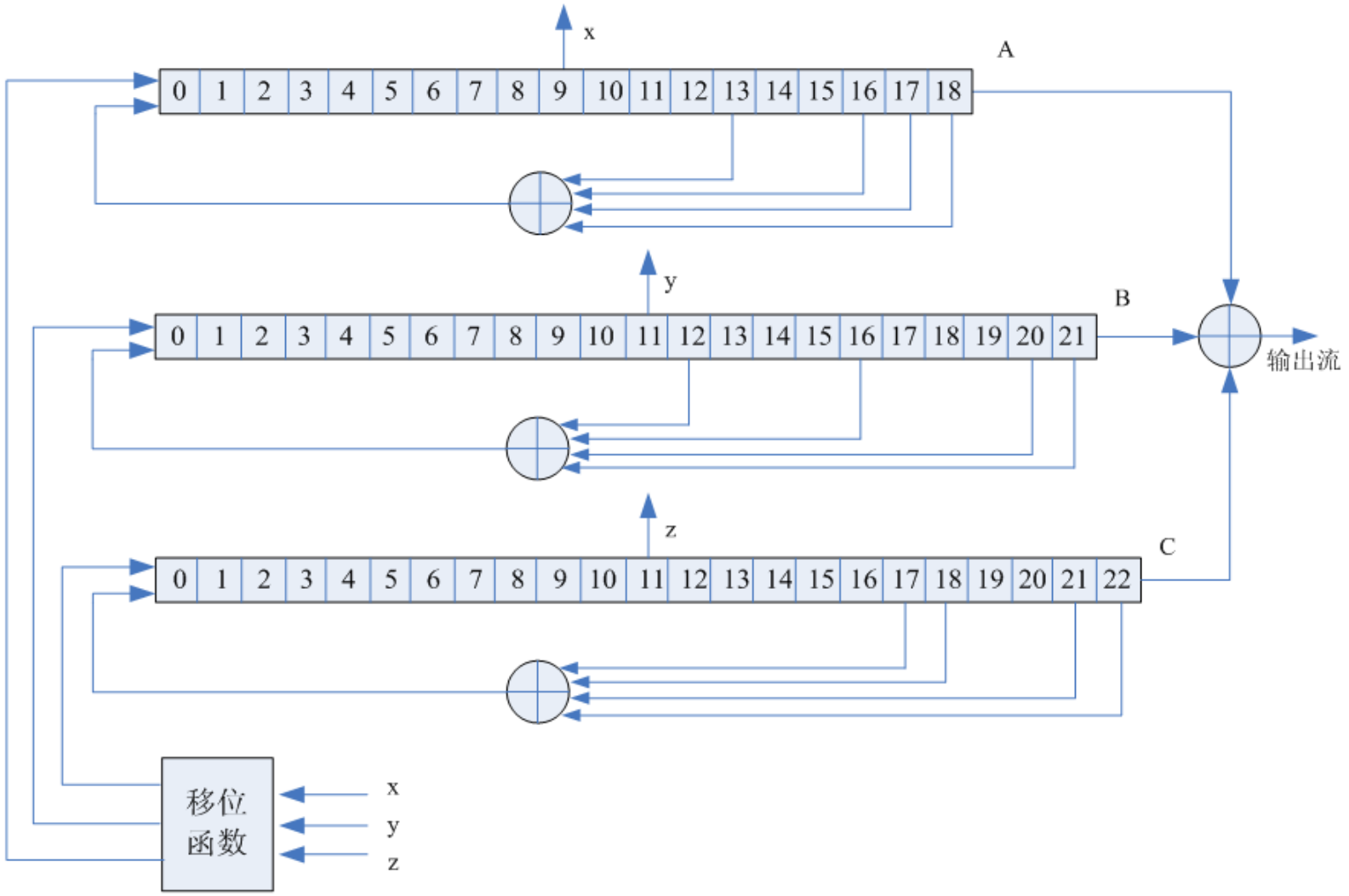
用来保护从基站到移动设备之间传输的信息 。

A5 算法有三种版本：

- A5/1 算法限制出口，保密性较强；
- A5/2 算法没有出口限制，但保密性较弱；
- A5/3 算法则是更新的版本，它基于 KASUMI 算法，

  尚未被 GSM 标准采用。

基于移位寄存器的算法


x
A
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
y
B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
输出流
z
C
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
移位
函数
x
y
z

基于移位寄存器的算法

实际使用中由一个 22 比特长的参数（帧号码， Fn ）和 64 比特
长的参数（会话密钥， Kc ）生成两个 114 比特长的序列（密
钥流）的黑盒子。这样设计的原因是 GSM 会话每帧含 228 比特，
通过与 A5 算法产生的 228 比特密钥流进行异或实现保密。

x 、 y 、 z （位置分别为 A 、 B 、 C 的第 9 、 11 、 11 位）进行钟控，
若三个位中间至少有两个为“ 1”，则为“ 1”的寄存器进行
一次进动，而为“ 0”的不移。

反过来，若三个位中至少有两个为“ 0”，则为“ 0”的寄

本章主要内容

- 流密码（序列密码）的思想起源
- 流密码技术的发展及分类
- 基于移位寄存器的流密码算法
- 其它流密码算法

RC4 算法

- RC4 是由 Rivest 于 1987 年开发的一种序列密码，它已被广泛应用于 Windows 、 Lotus Notes 和其它软件，还被用于安全套接字（ SSL ）和无线通信系统等。 RC4 算法最初没有被公布，但其源代码在 1994 年被人匿名发布，在这种情况下 RSA 数据公司于 1997 年公开了 RC4 算法 .
- RC4 不是基于 LFSR 的序列密码，它使用了一个 256 字节大小的非线性数据表（简称 S 表），依据表进行非线性变换，得到密钥流 . S 表的值 S_0 ， S_1 ，…， S_{255} 是数字 0 到 255 的一个排列， RC4 有两个计数器 I 和 J ，初值都为 0.
- RC4 的优点是算法简单、高效，特别适于软件实现，加密速度比 DES 大约快 10 倍 . RC4 可以支持不同密钥长度，美国政府特别限定，用于出口的 RC4 的密钥长度不得超过 40 位 .

RC4 算法

- RC4 首先进行 S 表的初始化，过程如下：
- 对 S 表进行填充： $S_i = i$ ， $0 \leqslant i < 255$ ；
- 用密钥填充另一个 256 字节的数组 K，如果密钥长度小于 256 字节，则依次重复填充，直至填满这个数组： K_0 ， K_1 ，…， K_{255} ；
- $J=0$ ；
- 对于 $I=0$ 到 255 重复以下步骤：
  - $J=J+ S_I+K_I$ （mod 256）；
  - 交换 S_I 和 S_J 。
- RC4 按下列步骤输出密钥流的一个字节 z ：
- $I=0$ ， $J=0$ ；
- $I = I + 1$ （mod 256）；
- $J = S_I + S_J$ （mod 256）；
- 交换 S_I 和 S_J ；
- $t=S_I+S_J$ （mod 256）；
- $z=S_t$ 。

RC4 算法

假如使用 3 位（从 0 到 7）的 RC4，其操作是对 8 取模（而不是对 256 取模）。数据表 S 只有 8 个元素，初始化为：

S	0	1	2	3	4	5	6	7
	0	1	2	3	4	5	6	7

选取一个密钥，该密钥是由 0 到 7 的数以任意顺序组成的。例如选取 5、6 和 7 作为密钥。该密钥如下填入密钥数据表中：

K	5	6	7	5	6	7	5	6
	0	1	2	3	4	5	6	7

RC4 算法

密钥调度算法 KSA

然后利用如下循环构建实际的 S 数据表：

```
j:=0;
for  i=0  to  7  do
      j:=(j+s(i)+k(i))  mod  8;
      swap(S(i),S(j));
```

该循环以 j=0 和 i=0 开始。使用更新公式后 j 为：

j=(0+S(0)+K(0)) mod 8=5

S 数据表的第一个操作是将 S(0) 与 S(5) 互换。

5	1	2	3	4	0	6	7
0	1	2	3	4	5	6	7

RC4 算法

索引 i 加 1 后， j 的下一个值为：

$$j=(5+S(1)+K(1)) \bmod 8=(5+1+6) \bmod 8=4$$

即将 S 数据表的 S(1) 和 S(4) 互换：

5	4	2	3	1	0	6	7
0	1	2	3	4	5	6	7

当该循环执行完后，数据表 S 就被随机化为：

5	4	0	7	1	6	3	2
0	1	2	3	4	5	6	7

RC4 算法

伪随机数生成算法 PRGA

这样数据表 S 就可以用来生成随机的密钥流序列。

从 j=0 和 i=0 开始， RC4 如下计算第一个密钥字：

i=(i+1) mod 8=(0+1) mod 8 =1

j=(j+s(i)) mod 8=(0+s(1)) mod 8=(0+4) mod 8=4

5	1	0	7	4	6	3	2
0	1	2	3	4	5	6	7

RC4 算法

然后如下计算 t 和 k:

5	1	0	7	4	6	3	2
0	1	2	3	4	5	6	7

t=(S(j)+S(i)) mod 8=(S(4)+S(1)) mod 8=(1+4) mod 8=5

k=S(t)=S(5)=6

第一个密钥字为 6，其二进制表示为 110 。反复进行

该过程，直到生成的二进制的数量等于明文位的数量。

本章主要内容

- 流密码（序列密码）的思想起源
- 流密码技术的发展及分类
- 基于移位寄存器的流密码算法
- RC4 算法

THE END !