Feedback shift register based stream ciphers

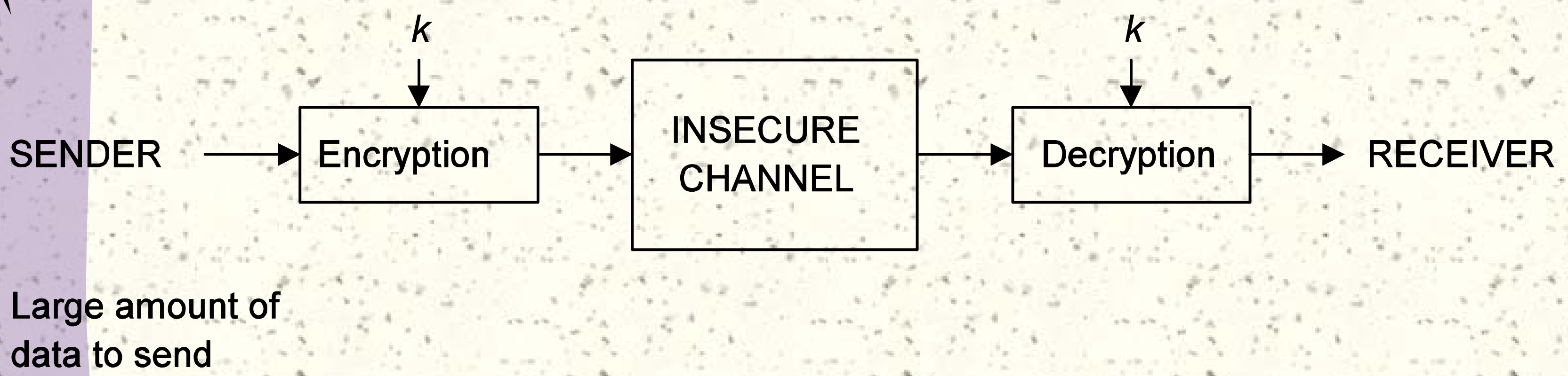
Thomas Johansson,
Lund University,
Lund, Sweden

CONTENTS

- Efficient encryption and possible solutions
- Stream ciphers
- Basic security analysis of stream ciphers

- LFSR sequences
- Design of LFSR based stream ciphers
- NLFSR sequences

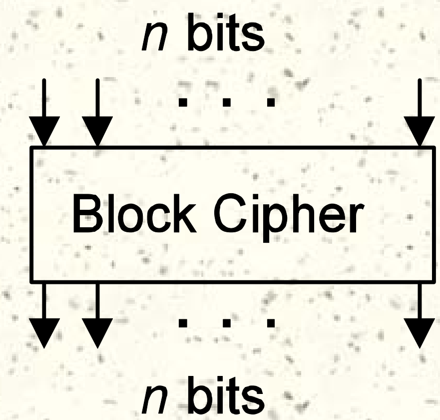
OUR PROBLEM – EFFICIENT ENCRYPTION



- Public key solutions too slow, used only for key setup
- We need ***symmetric encryption***
- *Stream ciphers, Block ciphers*

BLOCK CIPHERS

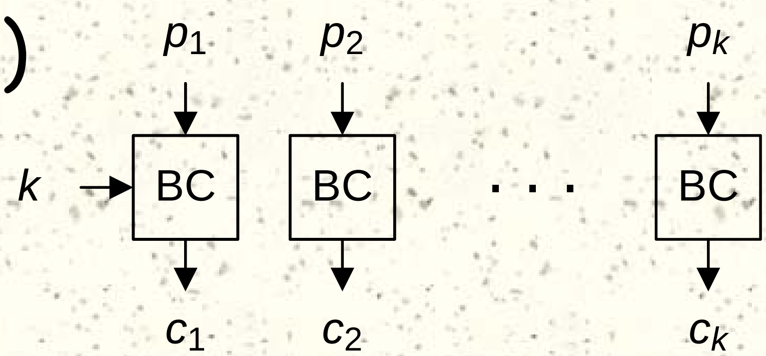
- Ideally, random permutations



Each key defines a permutation on the set of *n*-bit strings

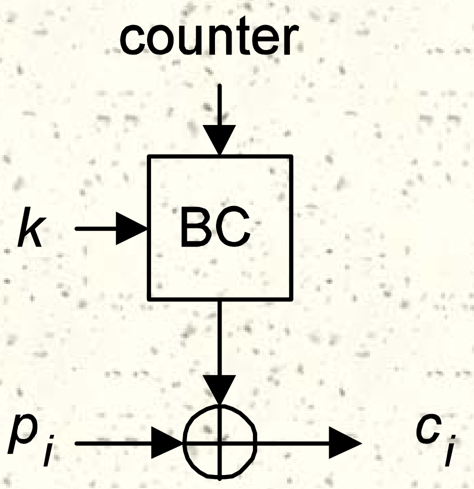
- **One problem**: We cannot encrypt as follows:

(because if $p_i=p_j$ then $c_i=c_j$)



BLOCK CIPHERS

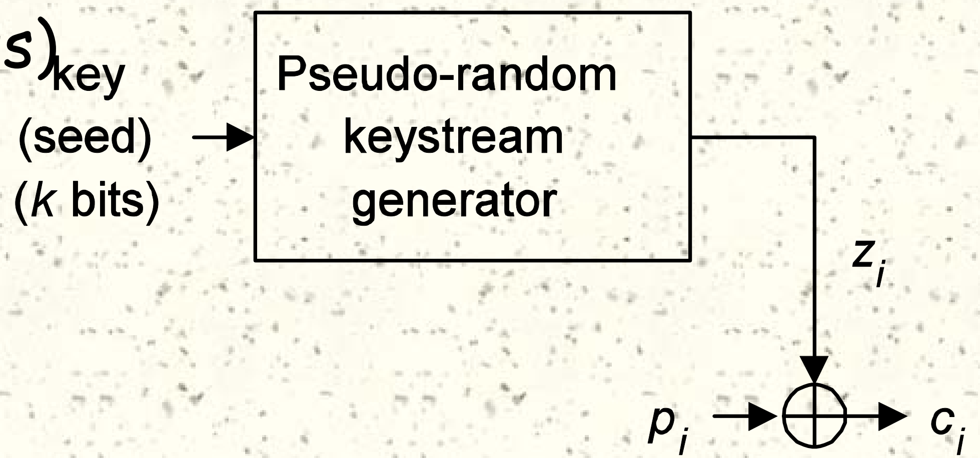
- The block cipher must be used in a *mode of operation*
- For example, counter mode



But this is also a stream cipher …

STREAM CIPHERS

(Additive synchronous)



- The PRKG stretches the *k* bit key to some arbitrarily long sequence

$$Z = z_1, z_2, z_3, \ldots$$

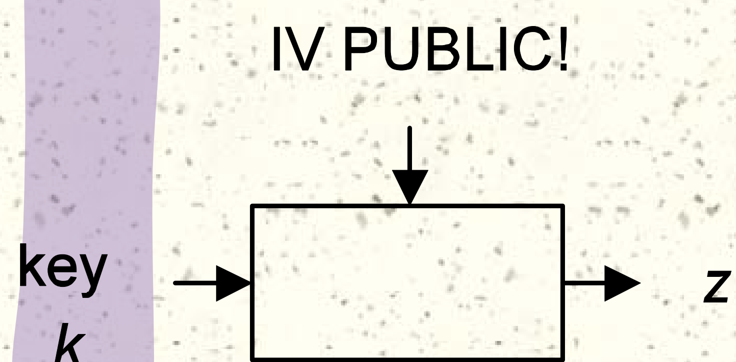
(*keystream, running key*)

DEFINITION OF A GENERATOR

Version 1

key	keystream
00…00	0110100110110100…
00…01	1010111001000010…

Version 2 (with IV):



IV	key	
00…00	00…00	0110100110110100…
"	00…01	1010111001000010…
00…01	00…00	1100101101010101…
"	00…01	0101001100110100…

OPERATION OF A STREAM CIPHER

1. Key initialization

   Set all the internal variables according to the selected key

   IV initialization

   Set all the internal variables according to the IV
2. Run the generator and produce the keystream

   $$Z = z_1, z_2, z_3, \ldots$$
3. Add the keystream to the plaintext

   $$c_i = p_i + z_i$$

MOTIVATION FOR STUDYING STREAM CIPHERS

- We need to bring forward new modern stream ciphers and study them carefully
- A modern stream cipher should be superior to a block cipher in performance (software and hardware)
- A modern stream cipher should provide security similar to a block cipher, for example, the ``best" attack is an exhaustive key search attack

BLOCK CIPHERS VS STREAM CIPHERS

Idea: Since we are already using stream ciphers through *block cipher + some mode of operation* we might gain something through a direct construction

Typical gain: Higher speed in software, smaller complexity in hardware, lower power consumption, …

In some applications this is <u>very</u> important

Security ?

- There are many well known and well studied block ciphers DES, IDEA, RC5, … more recent AES + candidates, Camelia,…
- There are not many equally well known stream ciphers A5, RC4, and definitely not many of them with good security!

Security of a stream cipher

- The standard assumption
  KNOWN PLAINTEXT ATTACK
- This implies knowledge of the keystream
  $Z = z_1, z_2, \ldots , z_N$
- When IV is used the opponent knows
  $Z_1 = z_{1,1}, z_{1,2}, \ldots , z_{1,N}$, for IV = 1
  $Z_2 = z_{2,1}, z_{2,2}, \ldots , z_{2,N}$ for IV = 2
  ...

generated by the same key k. Could be a *chosen IV attack.*

DIFFERENT TYPES OF ATTACKS

- KEY RECOVERY ATTACK

  Recover the secret key *k*.

- **DISTINGUISHING ATTACKS**

  Build a distinguisher that can distinguish

  $Z = z_1, z_2, \ldots, z_N$ from random

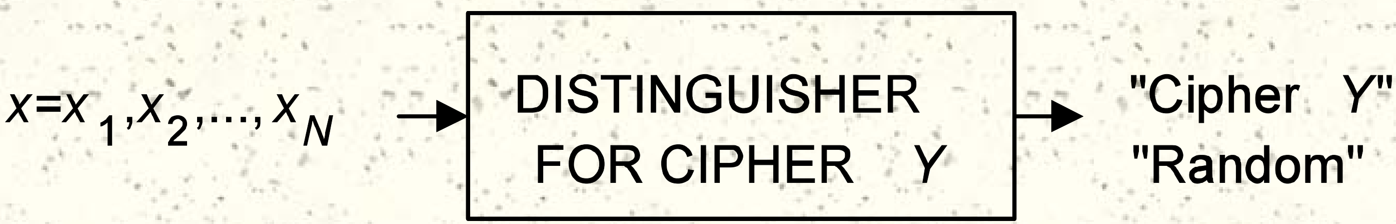
  (or $Z_1; Z_2; \ldots$ in the IV case)

- OTHER ATTACKS

  RELATED: Prediction of the next symbol, …

  UNRELATED: Side-channel attacks (power analysis, timing attacks, etc.), …

DISTINGUISHING ATTACKS

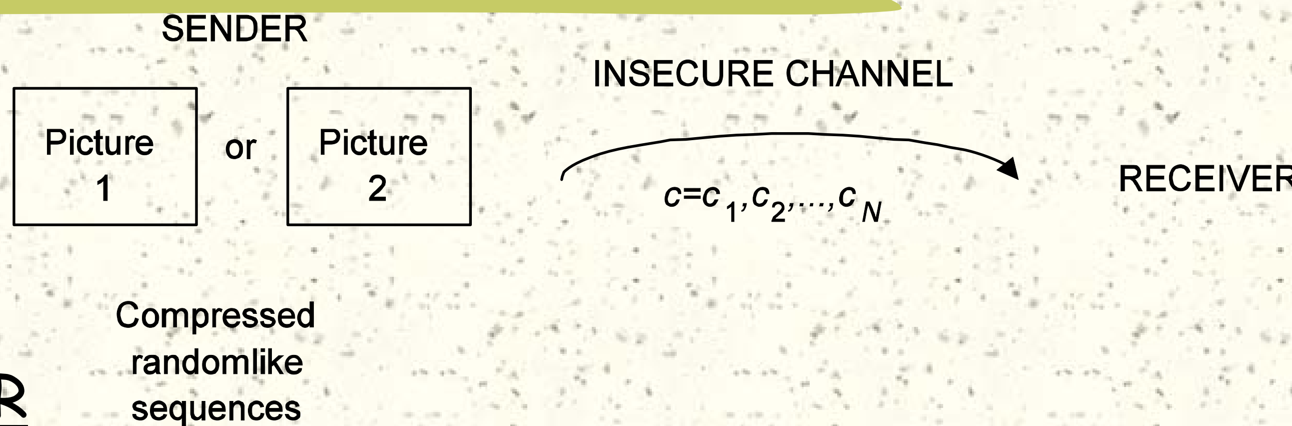


Assume that D is given a truly random X with probability ½.

- If P(D guesses correct) > ½ we have a distinguisher (with some advantage)

<u>Note</u>: We are usually not interested in cases when P(D guesses correct) = ½ + 2^{-n} for too small 2^{-n} .

APPLICATION OF A DISTINGUISHING ATTACK



THE ATTACKER

- Guesses that PLAINTEXT = PICTURE 1 (P_1)
- Calculates $Z' = P_1 + C$
- Give Z' to the distinguisher

  If Z' is recognized as ``CIPHER" the plaintext was PIC. 1

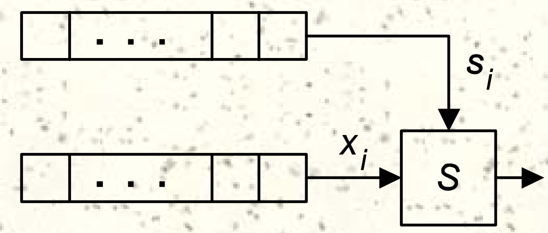
  If Z' is recognized as ``RANDOM" the plaintext was PIC. 2

(A wrong guess would give $Z' = P_1 + C = P_1 + P_2 + Z$)

DIFFERENT TYPES OF STREAM CIPHERS

BIT-ORIENTED: ``ONE BIT ON EACH CLOCK"

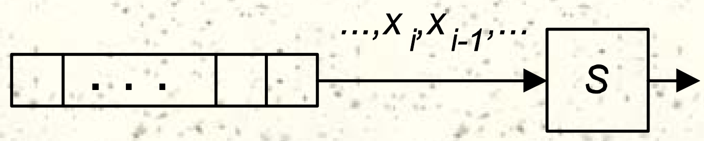
SHRINKING



S: If s_i=1 output x_i

otherwise delete x_i

SELFSHRINKING

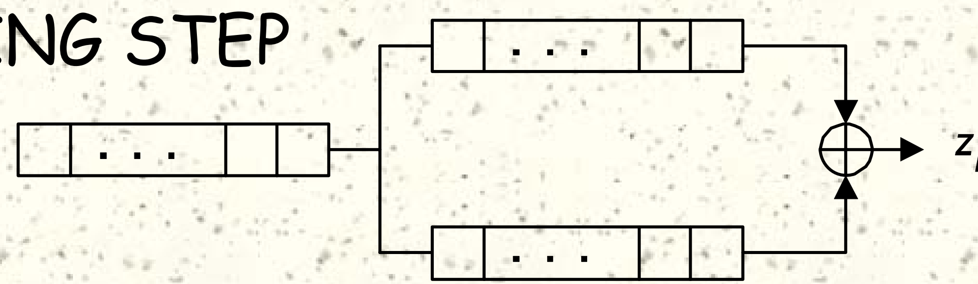


S: If $x_{i-1}x_i$ = 10 $\rightarrow$ 0

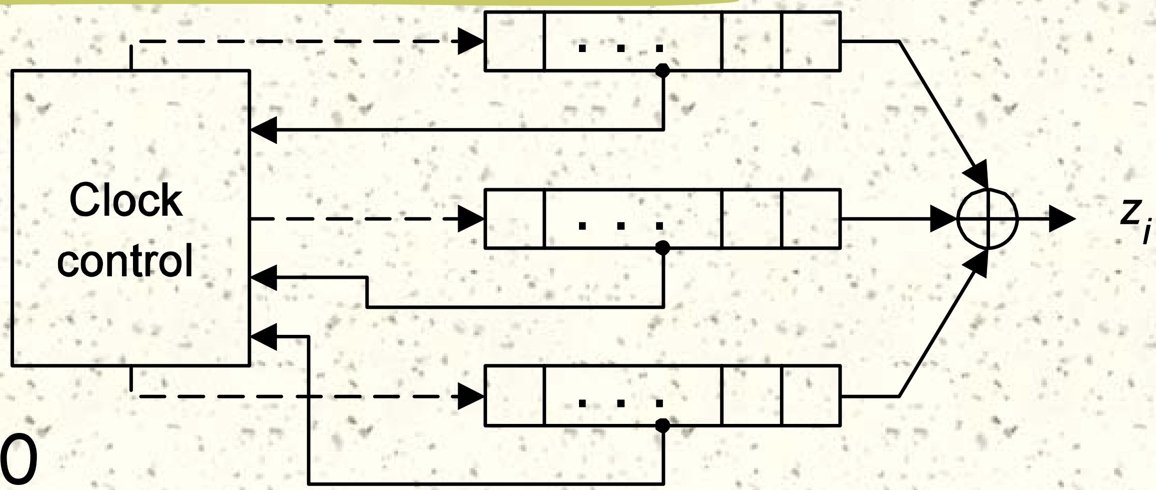
11 $\rightarrow$ 1

00 }
01 } delete

ALTERNATING STEP



- A5/1



- Bluetooth, E0
- Nonlinear combination generators and Filter generators

Very simple to implement in hardware

BUT

in general slow in software

- *In addition, some have security problems*

WORD-ORIENTED STREAM CIPHERS

``Produce a word on each clock/step"

- Word size: 8, 16, 32, 64
- When we are operating on words, things are a bit different...
- Moving closer to block ciphers, using their machinery, e.g.,

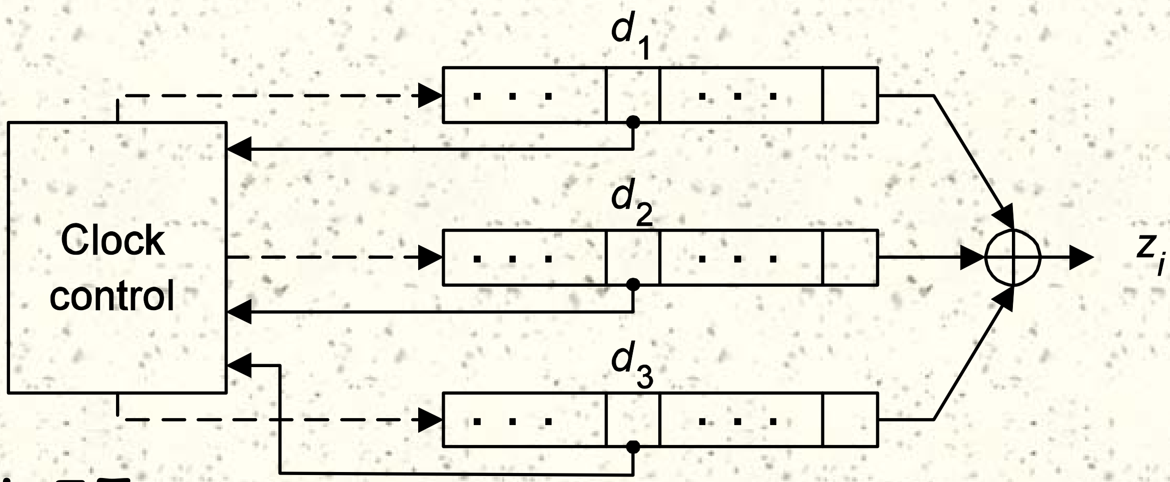
  S-boxes, SP-networks, etc.

ATTACK TECHNIQUES

- ``UNIVERSAL DISTINGUISHERS''
  NIST statistical test suite, DIEHARD, …
- GUESS AND DETERMINE
  Guess unknown things on demand
- ``CORRELATION ATTACKS''
  Dependence between output and internal unknown variables
- LINEAR ATTACKS
  Apply linear approximations
- ``ALGEBRAIC ATTACKS''
  View your problem as the solution to a system of nonlinear equations
- ``TIME-MEMORY TRADEOFF ATTACKS''

GUESS_AND DETERMINE

Example: ``GUESS AND DETERMINE"

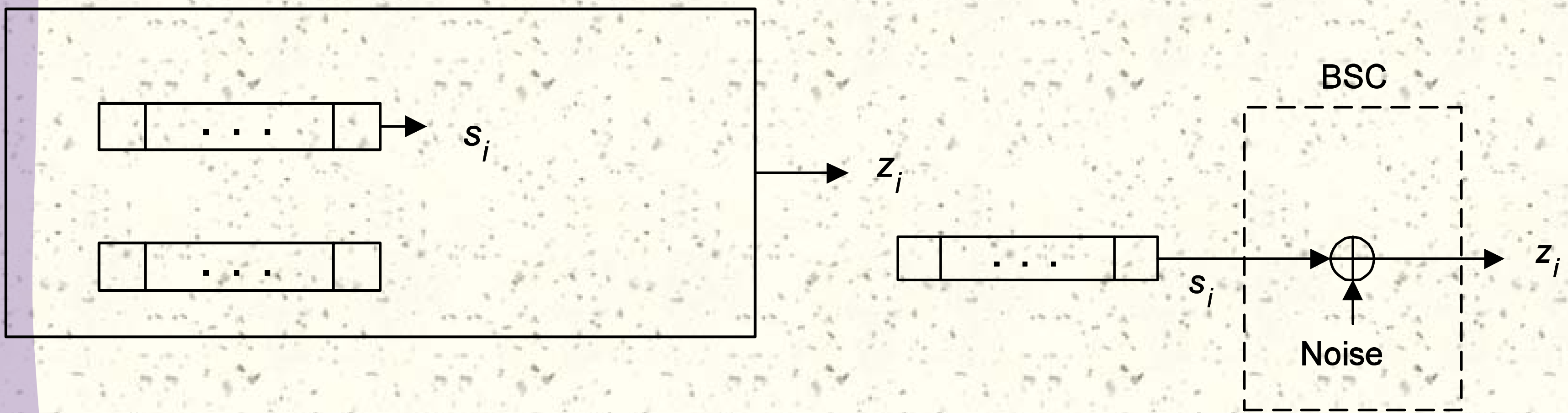


$s_1+t_1+u_1=z_1$

$s_{d_1}=x,\ t_{d_2}=x,\ u_{d_3}=x+1$

$s_2+t_2+u_1=z_2,\ldots$

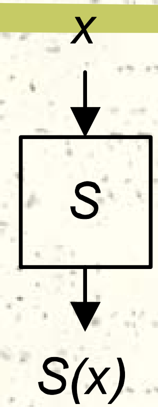
CORRELATION ATTACKS



- All possible LFSR sequences are codeword in a linear code C
- Reconstructing the initial state is the problem of decoding the code C on BSC $(1/2 + \varepsilon)$.

LINEAR ATTACKS

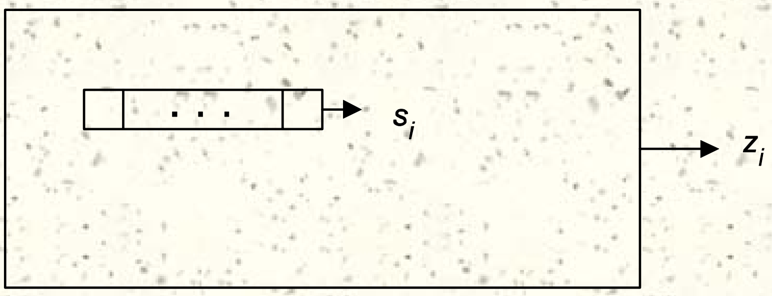
- Replace nonlinear parts by a linear approximation



$S(x) = \alpha \cdot x \; (+ N(x))$

- Find an expression where all unknown variables are eliminated, $\sum c_i z_{n+i} = 0$
- Binary case, let $B_n = \sum c_i z_{n+i}$. Then
  $P(B_n = 0) = \frac{1}{2} + \varepsilon$.
- Collect as many samples as we need to distinguish the sequence $B_1, B_2, \ldots$ from random.

ALGEBRAIC ATTACKS



- Find a <u>low</u> degree algebraic expression relating Z and S,

  $F(z_n, z_{n+1}, \ldots, s_n, s_{n+1}, \ldots) = 0$
- Valid for all n!
- Generate a system of nonlinear equations
- <u>Simplest case</u>: If the number of equations we can generate is very large we may solve the system by relinearization.

RECENTLY PROPOSED STREAM CIPHERS

Some proposed stream ciphers 2000-2003

SNOW 2.0	Lund Univ.
SOBER –t16, t32, 128	Qualcomm
TURING	"
SCREAM	IBM
MUGI	Hitachi
RABBIT	Cryptico

- Word-oriented, fast in software
- Use of LFSR or buffers
- One linear part/update and one nonlinear

eSTREAM project (2004-2008)

- 34 stream ciphers submitted (2005)
- Software: CryptMT, Dragon, HC, LEX, NLS, Rabbit, Salsa20, Sosemanuk
- Hardware: DECIM, Edon80, F-FCSR, Grain, Mickey, Moustique, Pomaranche, Trivium
- A lot of new ideas and techniques being evaluated…

DISCUSSION ISSUE

Where should the level of required security be?

Note: An n-bit block cipher in use is usually distinguished from random using $2^{n/2}$ output blocks and the same complexity.

Ex. AES is distinguished from random using ~ 2^{64} blocks of output

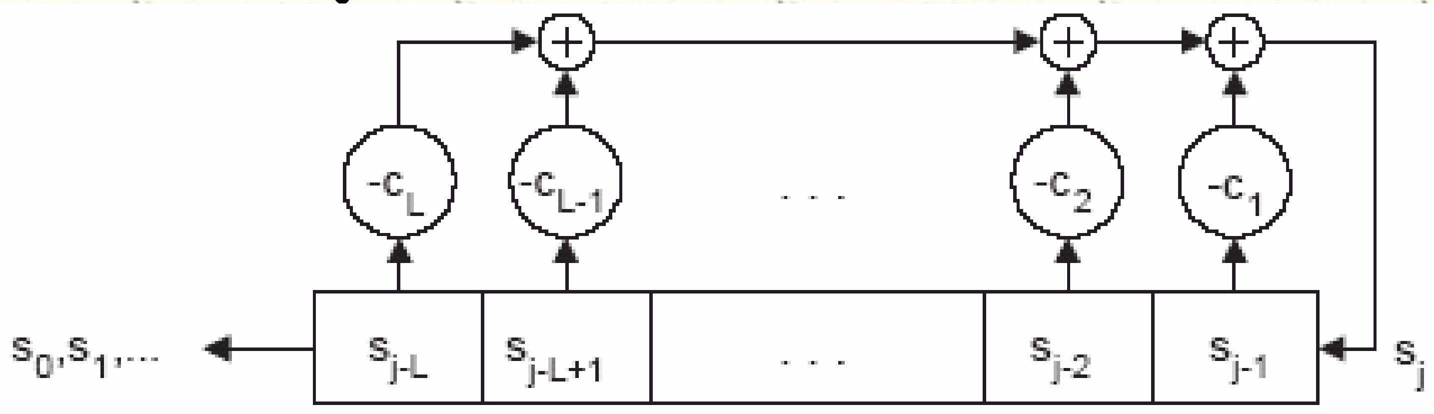
DES is distinguished from random using ~ 2^{32} blocks of output

LFSR BASED APPROACH TO STREAM CIPHER DESIGN

- LFSR sequences have nice statistical properties.
- The idea is to combine or modify LFSR sequences to completely destroy the linear property of them.
- This is the old classic way of constructing stream ciphers.

LFSR sequences

- LFSR $s_j \in GF(q)$



- Connection polynomial

  $C(D) = 1 + c_1 D + c_2 D^2 + \ldots + c_L D^L$

Alternative representations

- Linear recurrence relation

$$s_j=-c_1 s_{j-1} -c_2 s_{j-2}-\ldots-c_L s_{j-L},$$

Characteristic polynomial of the recurrence,

$$f(x)= x^L +c_1 x^{L-1} +c_2 x^{L-2} \ldots +c_{L-1}x+c_L$$

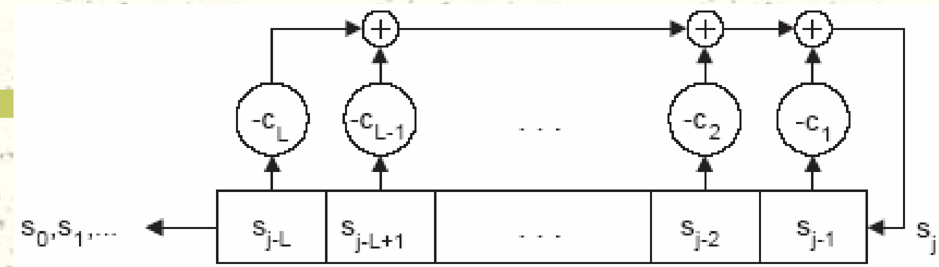
- If the polynomial is irreducible we can also write

$$s_j = Tr(\beta\alpha^j),$$

where $\alpha, \beta \in GF(q^L)$, and

$Tr(x) = x + x^q + x^{q^2} + \ldots + x^{q^{L-1}}$ is the trace map from $GF(q^L)$ to $GF(q)$.

Multiplication in $GF(q^L)$



- The LFSR basically implements multiplication with α in $GF(q^L)$
- A state-transition graph gives a number of different cycles.
- C(D) irreducible 1[1]+ $(q^L-1)/T$ [T]
- C(D) primitive 1[1]+ 1 $[q^L-1]$
- C(D) reducible cycles of different lengths

Primitive connection polynomials, q=2

- m-sequences (period 2^L-1)
- Statistical properties

$P(s_j=0)\approx 1/2$, $P((s_j,s_{j+1})=(a,b))\approx 1/4$, …

$P(s_{j_1}+s_{j_2}+\ldots+s_{j_n}=0)\approx 1/2$ unless

$s_{j_1}+s_{j_2}+\ldots+s_{j_n}$ obeys the recurrence relation.

- Adding two m-sequences results in a new m-sequence

Summary of statistical properties

m-sequences have almost ideal statistical properties, *except for*

the linear parity checks described by the connection polynomial

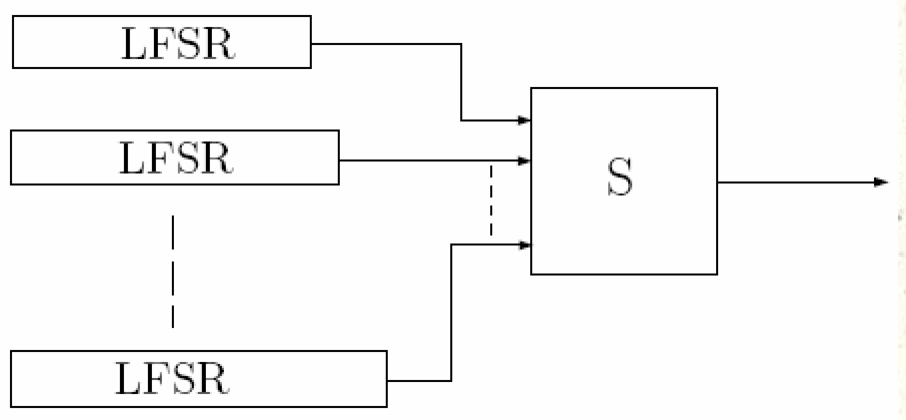
$$C(D)= 1 +c_1D+c_2D^2+\ldots+c_LD^L$$

and all its multiples P(D)=Q(D) C(D).

We need to do something about that…

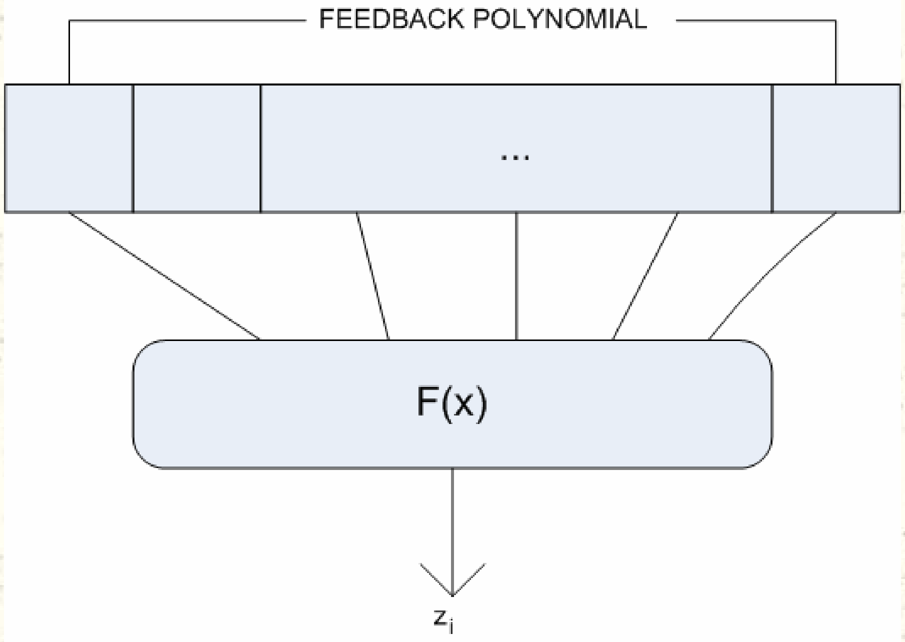
The nonlinear combination generator

- Combine several m-sequences using a Boolean function.



The filter generator

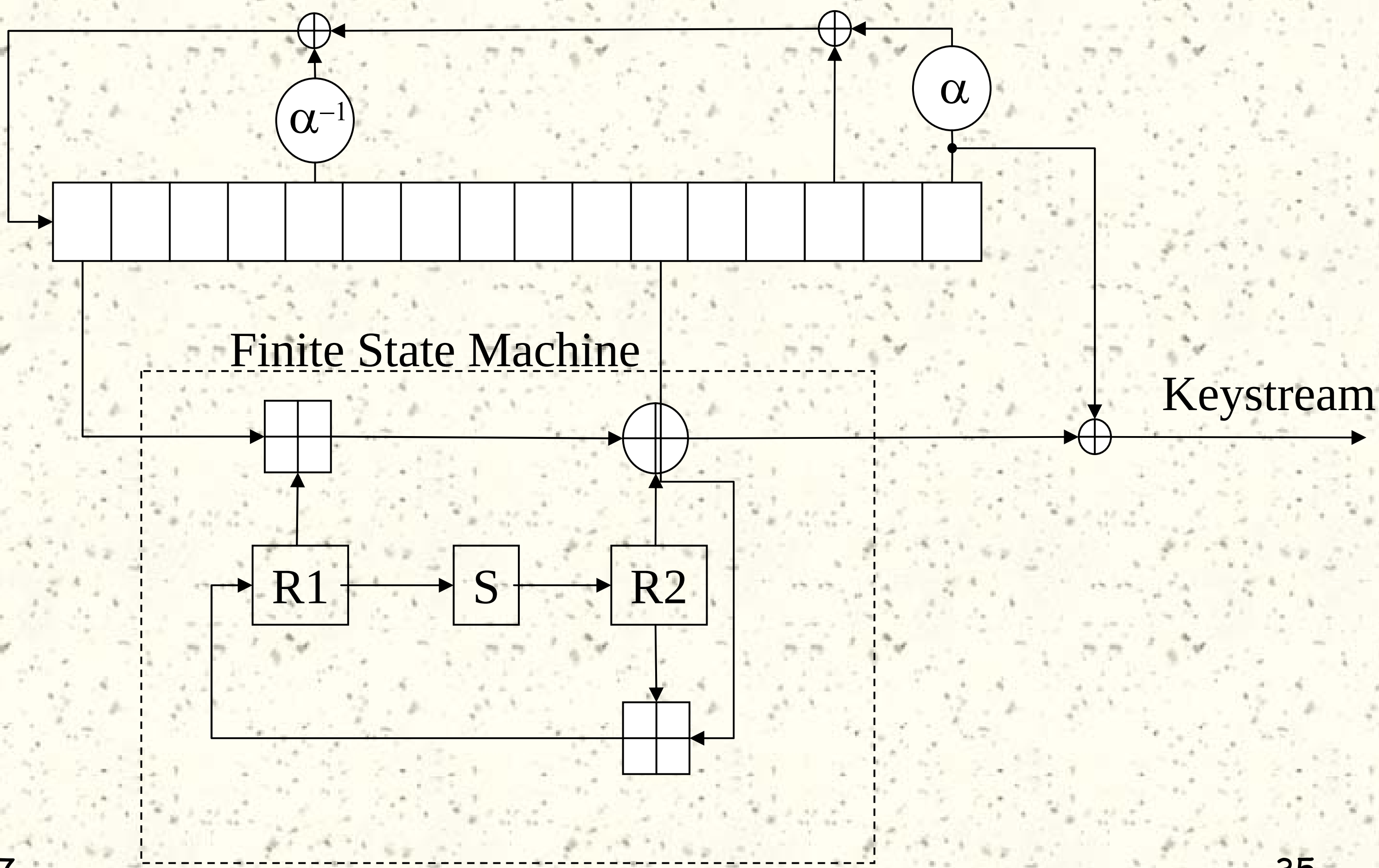
- An m-sequence is filtered by a nonlinear function F(x)



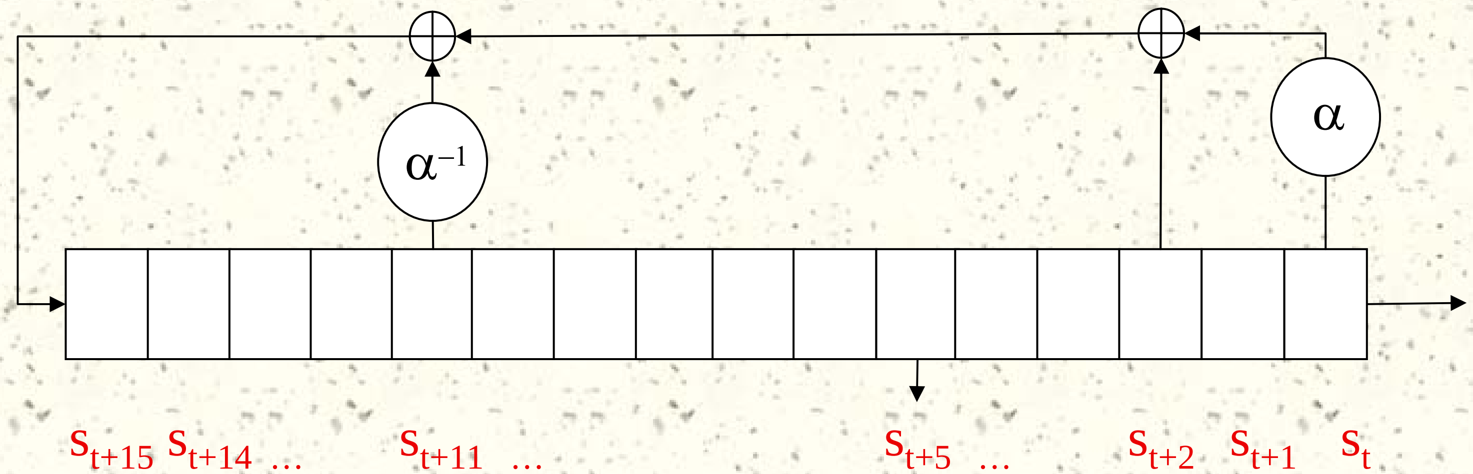
THE SNOW STREAM CIPHERS

- Designed at Lund University, Sweden (Johansson, Ekdahl)
- SNOW 2.0
  - ISO standard ISO/IEC 18033-4:2005
  - DPCP (DisplayPort Content Protection)
  - Reference stream cipher in eSTREAM
- SNOW 3G
  - UMTS

SNOW 2.0



THE LFSR



Feedback polynomial $\pi(x) = \alpha x^{16} + x^{14} + \alpha^{-1} x^{5} + 1 \in \mathrm{F}_{2^{32}}[x]$

More byte oriented structure:

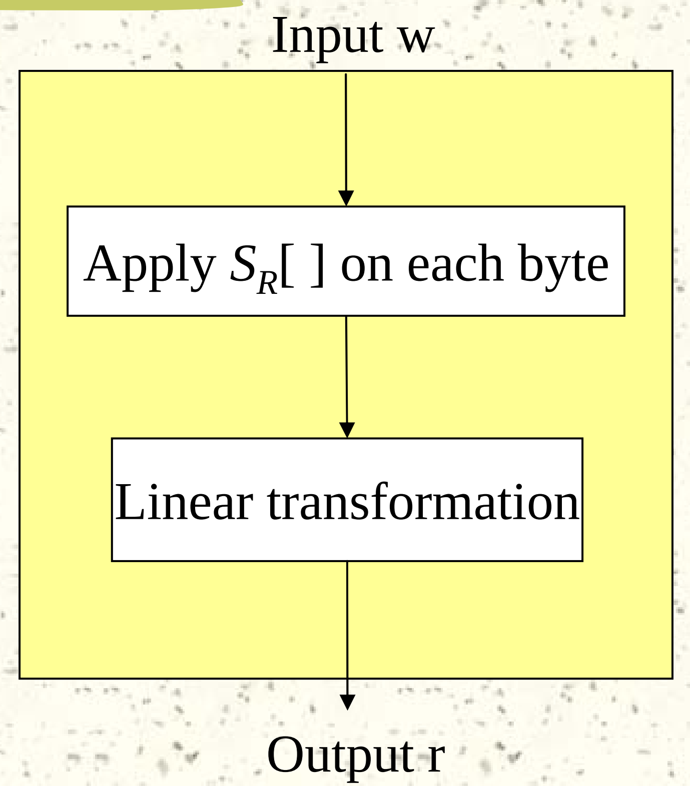
$\mathrm{F}_{2^{32}}$ is built from $\mathrm{F}_{2^{8}}$.

α is a root of primitive polynomial over $\mathrm{F}_{2^{8}}$.

THE S-BOX

Based on the round function of AES.
Let $r = S(w)$ be the output of the S-Box.

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} S_R[w_0] \\ S_R[w_1] \\ S_R[w_2] \\ S_R[w_3] \end{pmatrix}$$



Where S_R[ ] is the S-Box in AES, and each byte is considered an element in $\mathbb{F}_{2^8}$, defined by $x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x]$

KEY INITILIZATION

Two input variables:

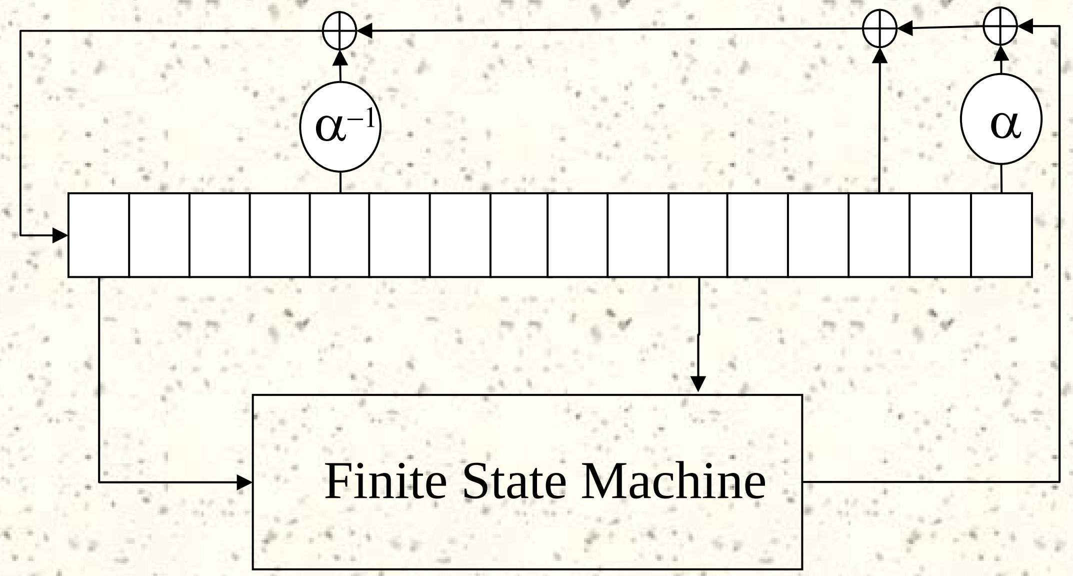
- Secret key of 128 or 256 bits, $(k_3,\ldots,k_0)$ or $(k_7,\ldots,k_0)$
- Publicly known IV of 128 bits, $(IV_3,\ldots,IV_0)$

Denote the register $(s_{15},\ldots,s_0)$

128 bit key: Load the register $(s_{15},\ldots,s_0)$ with a mix of key bits and IV bits.

KEY INITILIZATION

Premix with 32 clocks using:



Switch to normal operation, clock once, and read out the first keystream symbol.

SECURITY ASPECTS

- ✓ The feedback polynomial has two constants.

  Better spreading of the bits in the feedback loop.
  No known method to derive a linear recurrence that hold for each bit, and has reasonably low weight.

- ✓ The FSM takes two words as input.

  Harder to invert the FSM, takes more guessing.
  Suggests that correlations in the FSM is small.

- ✓ The S-Box has good spreading of the bits.

  Each output bit depends on each input bit.

IMPLEMENTATION ASPECTS

Simple instructions:

- XOR
- Integer addition
- Byte shift of a word
- Table lookup

LFSR:

Byte oriented feedback polynomial.
Multiplication with a and a^{-1} implemented as a byte shift and an XOR with a pattern.

$$mul_{\alpha}[c] = (c\beta^{23}, c\beta^{245}, c\beta^{48}, c\beta^{239})$$

$$mul_{\alpha^{-1}}[c] = (c\beta^{16}, c\beta^{39}, c\beta^{6}, c\beta^{64})$$

for all $c \in F_{2^8}$

```
// multiplication w·alpha
result=(w<<8) xor mul_a[w>>24];
```

The S-Box: Same method used in AES.

$$T_0[a] = \begin{pmatrix} xS_R[a] \\ S_R[a] \\ S_R[a] \\ (x+1)S_R[a] \end{pmatrix}, \quad T_1[a] = \begin{pmatrix} (x+1)S_R[a] \\ xS_R[a] \\ S_R[a] \\ S_R[a] \end{pmatrix}$$

$$T_2[a] = \begin{pmatrix} S_R[a] \\ (x+1)S_R[a] \\ xS_R[a] \\ S_R[a] \end{pmatrix}, \quad T_3[a] = \begin{pmatrix} S_R[a] \\ S_R[a] \\ (x+1)S_R[a] \\ xS_R[a] \end{pmatrix}$$

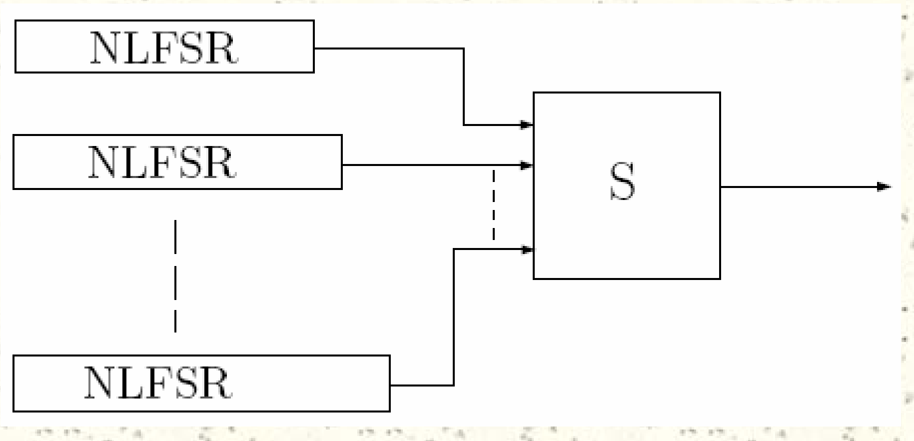
```
//Calculate r=S-Box(w)
r=T0[byte0(w)] xor T1[byte1(w)] xor T2[byte2(w)] xor T3[byte3(w)];
```

PERFORMANCE OF SOME STREAM CIPHERS

Primitive	Profile	Key	IV	MAC	Stream	40 bytes	5
COPY	SW & HW	80	80		0.33	4.36	
ABC-v1	SW	128	128		3.43	13.57	
Py	SW	128	64		3.66	232.71	
Py	SW	256	128		3.66	230.28	
Py6	SW	128	64		3.82	78.93	
Py6	SW	256	128		3.83	83.32	
ABC-v2	SW	128	128		4.15	15.53	
HC-256	SW	256	128		4.95	2426.98	
HC-256	SW	128	128		4.97	2409.40	
SNOW-2.0	SW	128	128		5.19	39.04	
SNOW-2.0	SW	256	128		5.22	42.69	
Phelix	SW	128	128	64	5.53	23.62	
Phelix	SW	256	128	128	5.58	23.59	
SOSEMANUK	SW	128	64		5.72	48.02	
SOSEMANUK	SW	256	128		5.72	39.29	
NLS	SW	128	64		5.75	42.44	
NLS	SW	128	128		5.76	39.05	
Rabbit	SW & HW	128	64		7.71	28.22	
TRIVIUM	HW	80	64		8.53	55.22	
TRIVIUM	HW	80	80		8.54	56.42	
LEX	SW & HW	128	128		9.90	20.83	
MAG-v3	SW	256	64		10.59	713.58	
RC4	SW	256	0		10.98	581.88	
RC4	SW	128	0		11.01	581.85	
NLS	SW	128	128	128	12.25	107.11	
Dragon	SW	128	128		12.27	78.87	
Dragon	SW	256	128		12.27	82.96	
NLS	SW	128	64	64	12.30	95.28	
Salsa20	SW & HW	128	64		13.85	39.20	
Salsa20	SW & HW	256	64		13.85	42.10	
DICING	SW	128	128		14.67	409.99	
DICING	SW	256	128		14.69	414.70	
CryptMT	SW	128	128		16.06	997.48	
CryptMT	SW	256	128		16.07	1064.72	
Yamb	SW & HW	256	128		16.48	1221.48	
Yamb	SW & HW	128	64		16.56	1205.22	
Mir-1	SW	128	64		18.13	59.29	
AES-CTR	SW & HW	128	128		24.13	33.91	
MAG-v1	SW & HW	128	32		30.79	251.83	
Polar-Bear	SW & HW	128	64		30.87	61.36	

Nonlinear shift register sequences

- **De Bruijn sequences (period 2^L)**
- The Achterbahn stream ciphers



NLFSR is implemented as an LFSR but with nonlinear feedback.
Now we do not necessarily have $P(s_{j_1}+s_{j_2}+\ldots+s_{j_n}=0)\approx 1/2$.

SUMMARY

- ✓ Overview of stream ciphers.
- ✓ Using LFSR sequences in stream ciphers.

Research issues:

- ✓ Security analysis of LFSR based stream ciphers.
- ✓ Efficient implementation of sequence generation.
- ✓ Stream ciphers in constrained environments.