UNIVERSITÄT DUISBURG ESSEN

Stream ciphers and random number generators

A.J. Han Vinck
University of Duisburg/Essen
June 1, 2013
Vinck@iem.uni-due.de

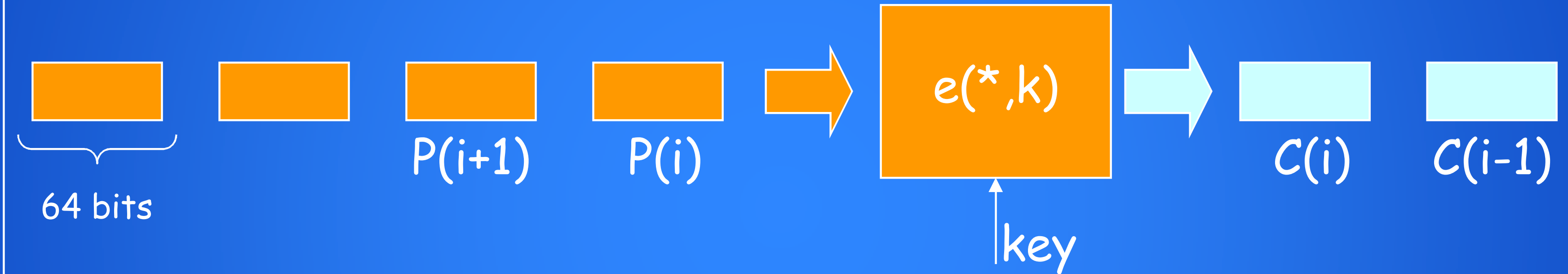
content

- Pseudo random number generation
  - Linear feedback shift register
  - RC4
  - Some examples

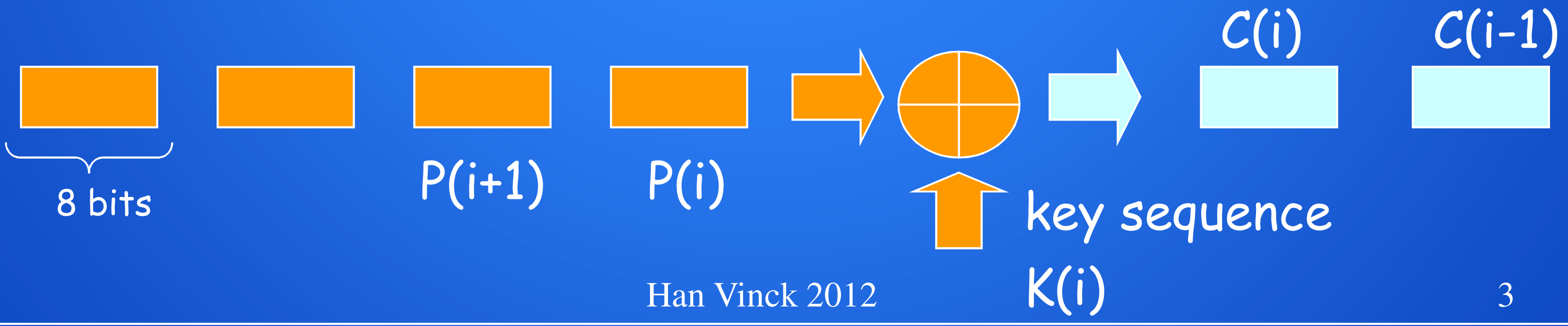
Difference between a block and a stream cipher

Block cipher:

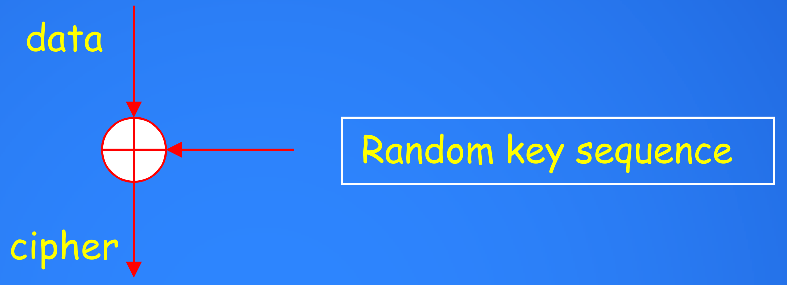
operates on blocks of plaintext , block by block independent from each other



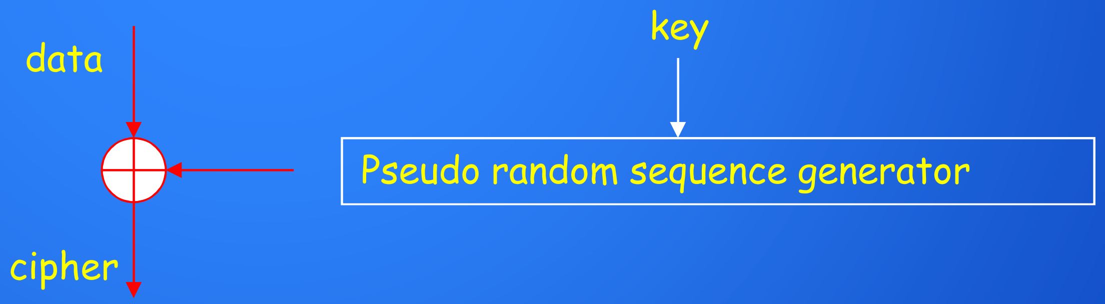
Stream cipher: operates on the message, symbol by symbol



Stream ciphers: Vernam

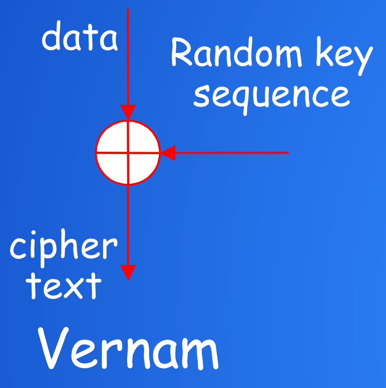


vernam

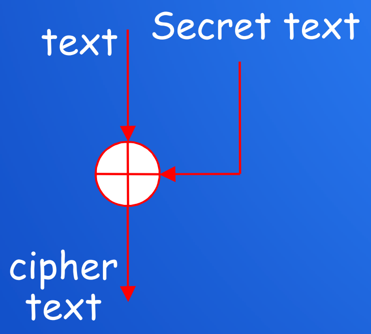


„vernam"

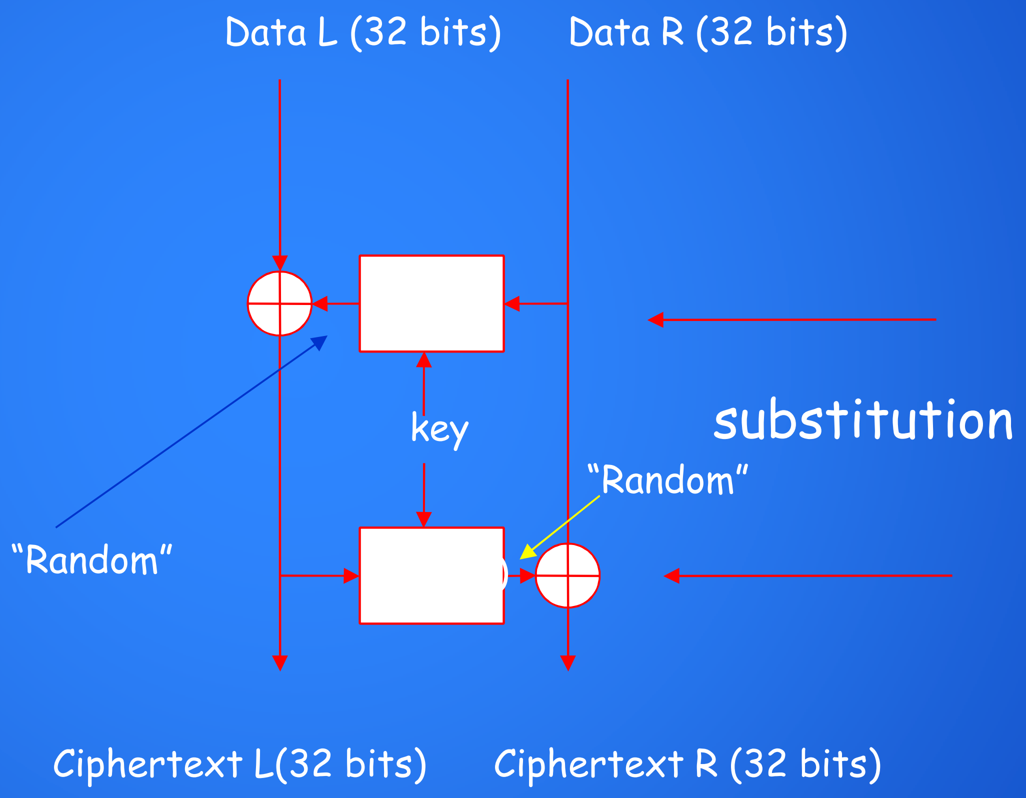
Remember Feistel?



Vernam

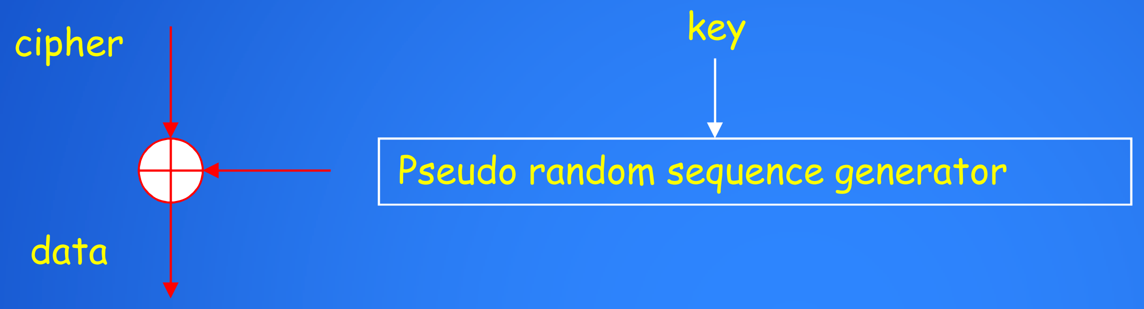


Vigenere



Feistel (IBM, 1974)

Stream ciphers: the decryption



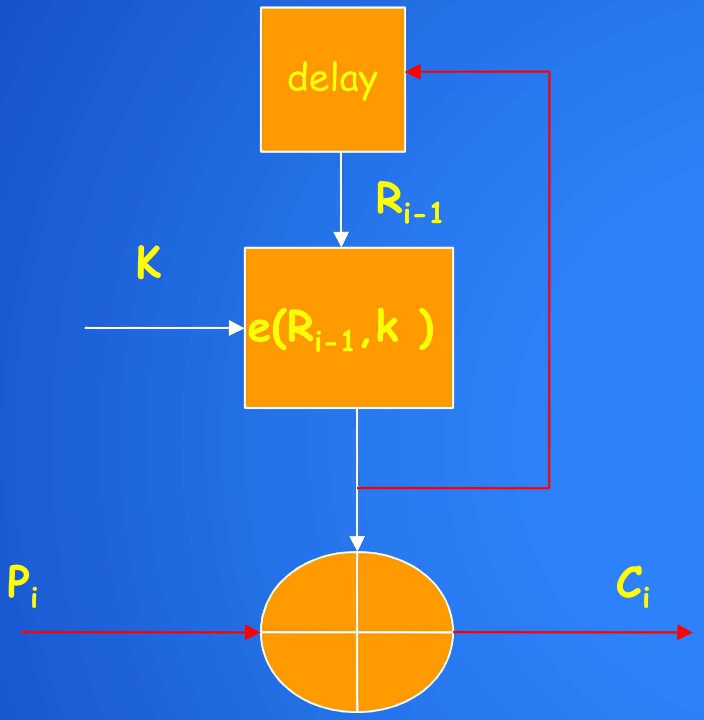
Problem: using the same key (or initial value of the key generator) gives an easy way to find out the original data (if for instance text)

Since:

$$\text{Cipher1} \oplus \text{Cipher2} = \text{text1} \oplus \text{key1} \oplus \text{text2} \oplus \text{key1} = \text{text1} \oplus \text{text2}$$

frequency analysis can solve this problem easily!

example: Output Feedback Mode



R_0 given

$C_i = P_i \oplus e(R_{i-1}, k)$

$P_i = C_i \oplus e(R_{i-1}, k)$

Advantage: no error propagation; easy to start decryption (using the encrpytion function) when time is known

Disadvantage: manipulation possible when parts of plaintext are known by using the XOR to replace P_i

Can you show this?

We use a Pseudo Random Number Generator

What do we want?

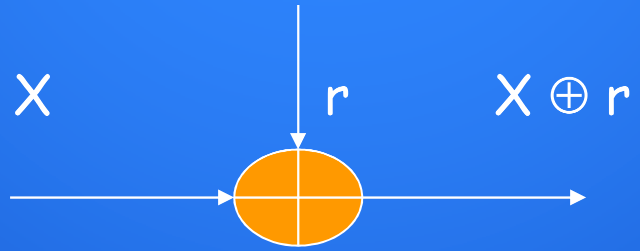
- generate „random" looking sequence

  http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo38.php

- produce an unpredictable sequence with infinite period

  http://en.wikipedia.org/wiki/Diehard_tests

- fast and low complexity



Definition?

A random number generator (RNG) is

a computational or physical device

designed to generate a sequence of numbers or symbols

that lack any pattern, i.e. appear random.

What is random?

$$\boxed{\text{RNG}} \xrightarrow{x \in \{0,1\}}$$

We expect: $P(0) = P(1) = 0.5$

but also: $P(x_i | x_{i-1} \; x_{i-2} \; \cdots \; x_1) = P(x_i)$

Or in experiments $P(0) = 1 - p = 1 - P(1)$

„Random“ Numbers can be generated by:

Hardware (digital building blocks)

Software (algorithms)

Physical device (noise generator)

example

linear congruential generator, which uses the recurrence

$$X_{n+1} = (aX_n + b) \bmod m$$

FROM WIKIPEDIA

Donald Knuth (1997). "Chapter 3 – Random Numbers". *The Art of Computer Programming*. Vol. 2: Seminumerical algorithms (3 ed.).

http://en.wikipedia.org/wiki/The_Art_of_Computer_Programming

Kroese, D. P.; Taimre, T.; Botev, Z.I. (2011). "Chapter 1 – Uniform Random Number Generation". *Handbook of Monte Carlo Methods*. New York: John Wiley & Sons. p. 772. ISBN 0-470-17793-4.

Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Chapter 7. Random Numbers". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8

NIST SP800-90A, B, C series on random number generation

random vs. pseudorandom numbers

1. One measures some physical phenomenon

   that is expected to be random

   - measurements must be correct !

2. Algorithms (pseudorandom number generators are predictable)

   that can produce long sequences of apparently random results,

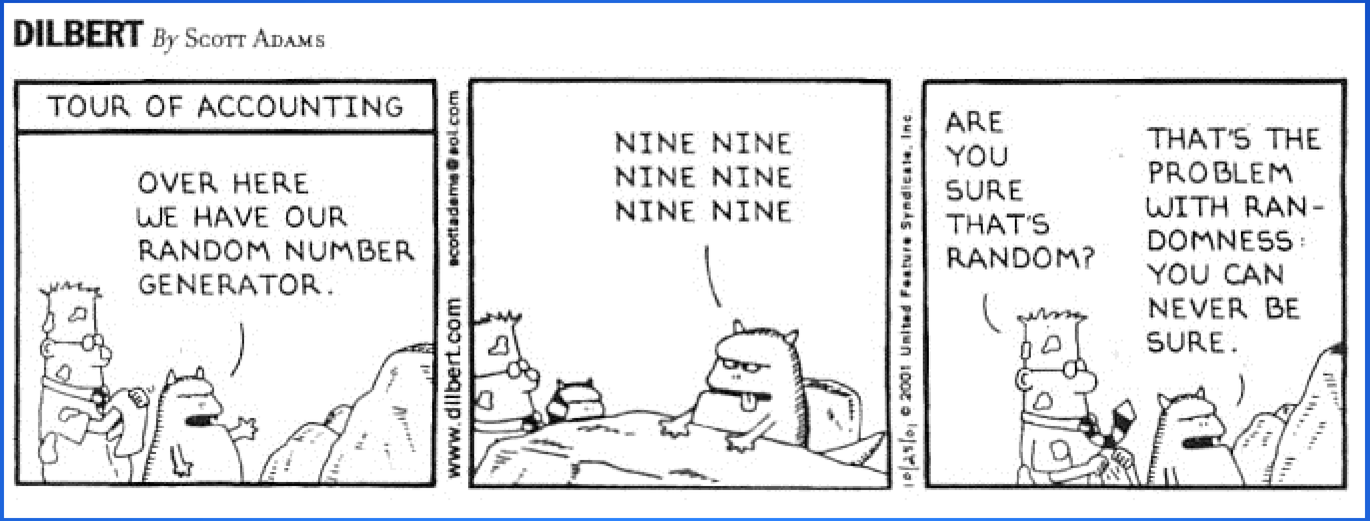
   -statistical analysis of the output is needed

tests

Tests are based on mathematical properties that can be calculated

See: http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html

for instance:	average number of ones
	frequency in a block of length M
	distribution of runs of 0's and 1's
	Fourier analysis
	Linear complexity test
	Entropy test
	etc.


DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.

National Institute of Standards and Technology

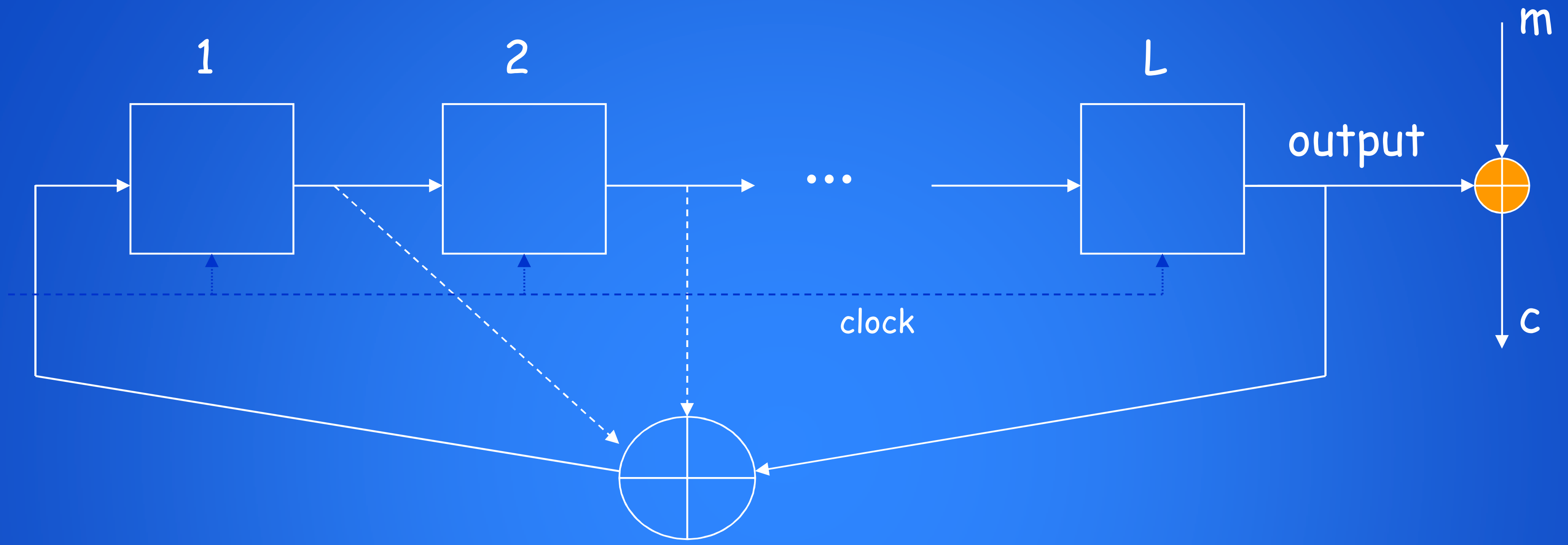
NIST National Institute of Standards and Technology U.S. Department of Commerce

Since 1997, the Random Number Generation Technical Working Group (RNG-TWG) has been working on developing a battery of statistical tests suitable in the evaluation of random number generators and pseudo-random number generators used in cryptographic applications. Currently, we are finalizing the documentation and software in preparation for public release.

We look at the basic structure of a linear feedback shift register

LFSR

Linear Feedback Shift Register



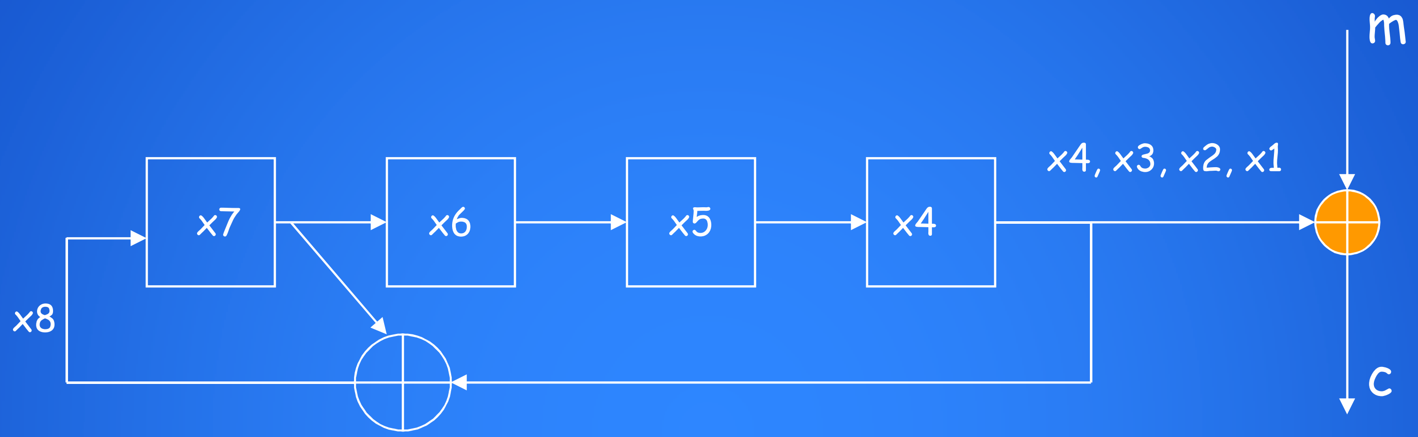
Every time unit:

binary content of the delay elements shifted 1 position to the right

the XOR of a subset of the cell contents is placed in the leftmost cell

example

Note: we omit the drawing of the clock



(x4 x3 x2 x1) = (1 1 0 0)

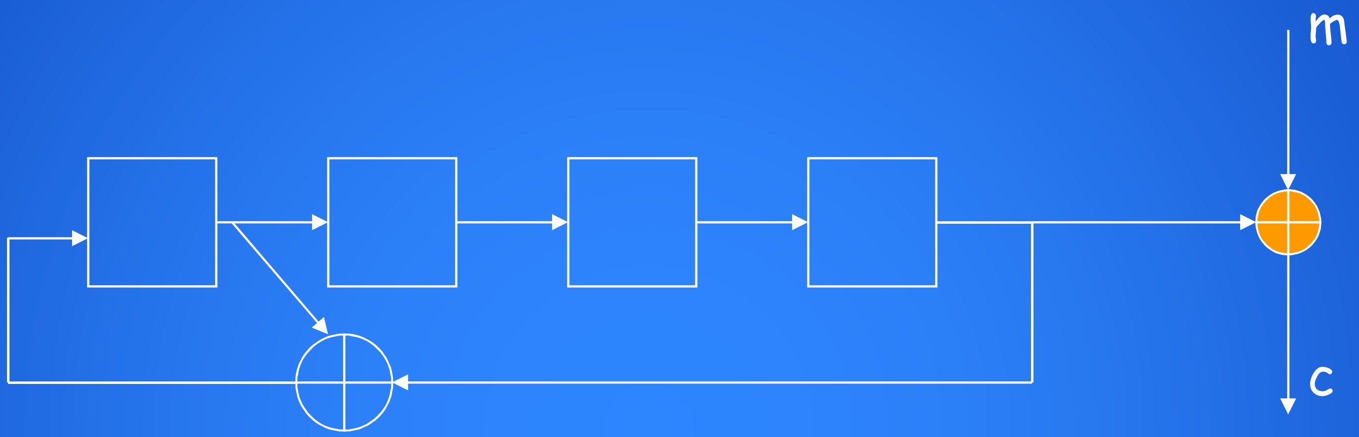
(x5 x4 x3 x2) = (1 1 1 0)

(x6 x5 x4 x3) = (1 1 1 1)

(x7 x6 x5 x4) = (0 1 1 1)

(x8 x7 x6 x5) = (1 0 1 1)

Example (cont'd)



Shift register content: ... **0100** 0010 0001 1000 1100 1110 1111 0111
1011 0101 1010 1101 0110 0011 1001

0100 ..., hence, period 15 = $2^4 - 1$

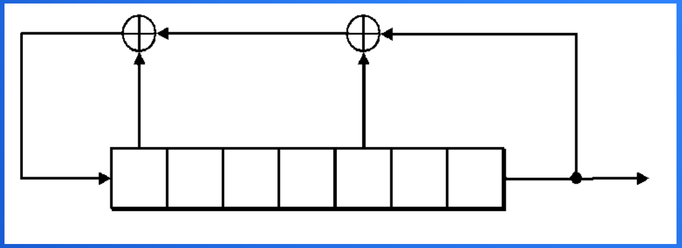
The output sequence:= ...001000111101011...

Homework: change the connections and comment on the effect

Properties LFSR:

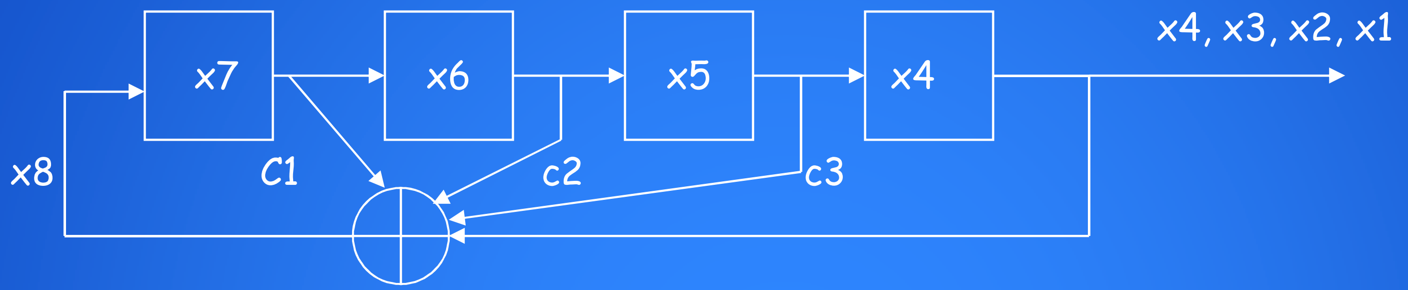
- maximum period of the output sequence = $2^L - 1$ bits long
  - Note that the all zero content cannot occur with any nonzero starting values
- good (?) connections are listed („primitive“ sequences)

  equivalent to generating GF(2^L)
- easy to find connections from 2L output observations



Another example, L = 7

Example L = 4



relation

$$\underbrace{\begin{pmatrix} c1 & c2 & c3 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} x7 & x6 & x5 & x4 \\ x6 & x5 & x4 & x3 \\ x5 & x4 & x3 & x2 \\ x4 & x3 & x2 & x1 \end{pmatrix}}_{X1} = \underbrace{\begin{pmatrix} x8 & x7 & x6 & x5 \\ x7 & x6 & x5 & x4 \\ x6 & x5 & x4 & x3 \\ x5 & x4 & x3 & x2 \end{pmatrix}}_{X2}$$

NOTE: The matrix C is non-singular (has inverse), if X1 is, then also X2 is.

Conclusion: $C = X2 * X1^{-1}$ and 8 components suffice to calculate C !

Example L = 4

Suppose that we observe: (1 1 0 0 1 0 0 0) = (x8, x7, x6, x5, x4, x3, x2 , x1)

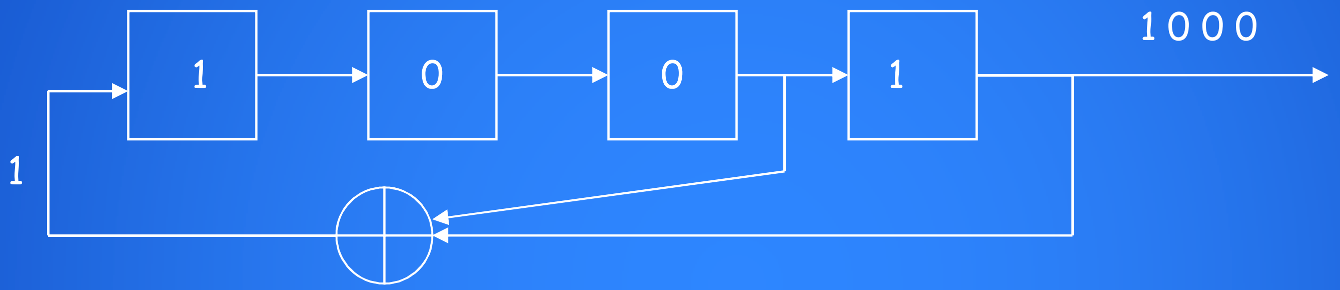
Construct: $X1 = \boxed{\begin{matrix} 1&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{matrix}}$ $X2 = \boxed{\begin{matrix} 1&1&0&0 \\ 1&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \end{matrix}}$

From this: $X1^{-1} = \boxed{\begin{matrix} 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&1 \end{matrix}}$ $X2\,X1^{-1} = \boxed{\begin{matrix} 0&0&1&1 \\ 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \end{matrix}}$ c3 = 1!

Use Cramer's rule

(or Gauss-Jordan, easier)

Example L = 4, c3 = 1



Content

1000	0100	0010	1001	1100	0110	1011
0101	1010	1101	1110	1111	0111	0011
0001	1000	...				

How to calculate *C* in general?

Assume that the shift register of length L generates a sequence of maximum period $2^L - 1$. (all possible contents $\neq$ 0 must occur)

Then, (xL, ···, x2, x1) = (1, ···, 0,0) is a valid content,

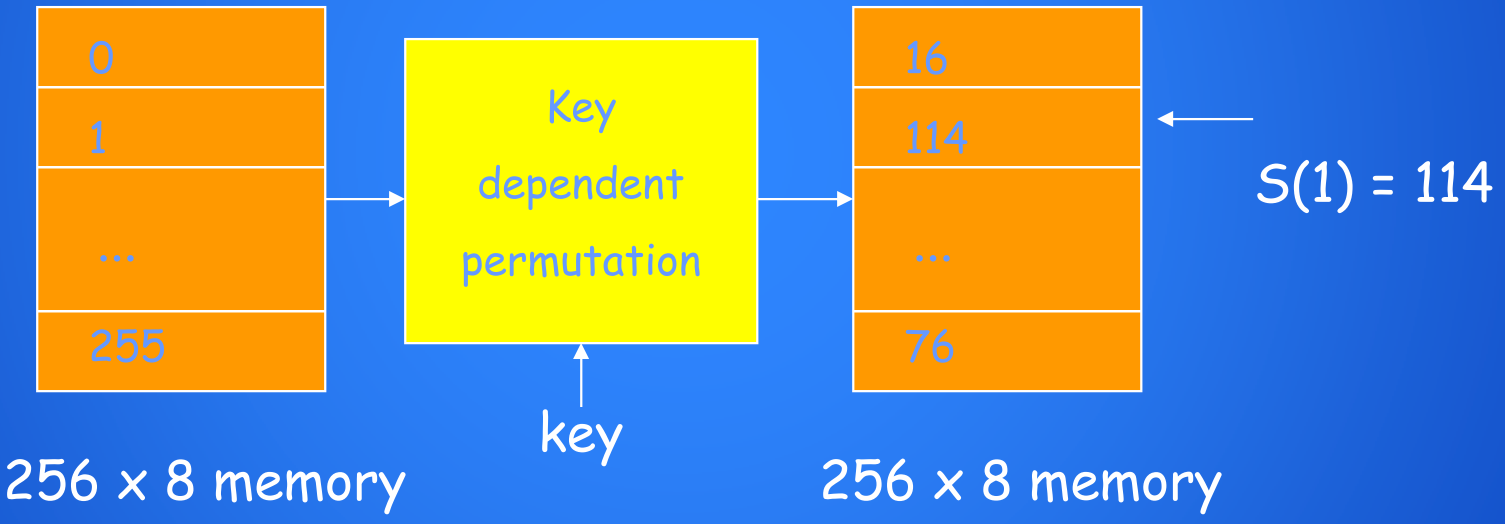
- as a consequence, X1 is nonsingular and so is X2
- since *C* * X2 = X3, it follows that X3 is also nonsingular, etc.
- we need 2L coefficients to construct X(t-1) and Xt and

  calculate *C* = Xt * $[X(t-1)]^{-1}$

Hence, any 2L -consecutive- coefficients can be used to calculate *C* !

Random Byte Generation RC4 (Ron Rivest) step 1

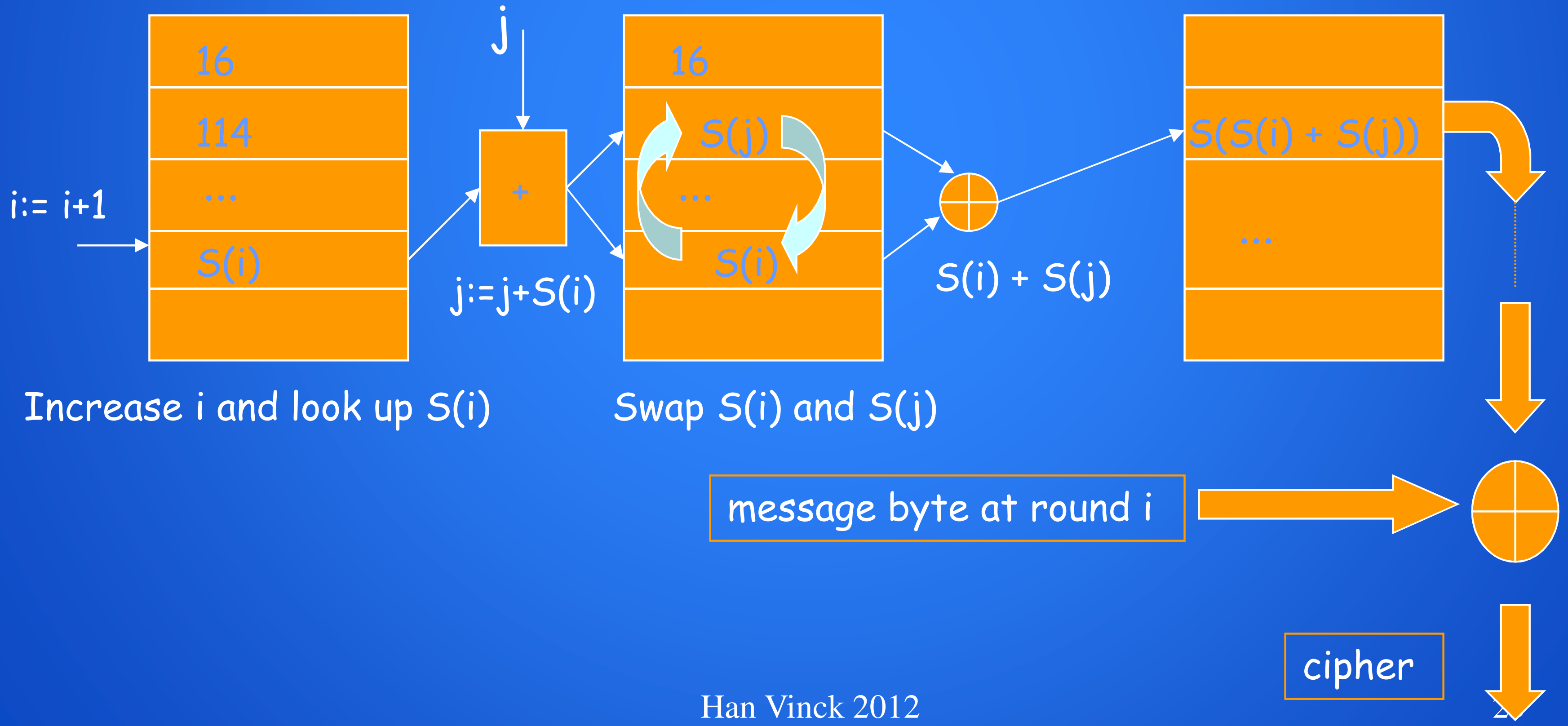
initialize



Random Byte Generation
RC4 (Ron Rivest) step 2

Algorithm: start i = 0 and j = 0, all additions modulo 256

for round i



RC4 (Ron Rivest), some facts

- simple and efficient pseudo-random generator.
- recent attacks relate only to the key-scheduling algorithm, not to the generator.
- no known practical attacks against this generator when initialized with a randomly-chosen initial state.
  - i.e. no two messages with same key! (see WEP protocol attack)
- Reference:
  - FMS01] Fluhrer, Scott, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4
  - http://www.rsa.com/rsalabs/node.asp?id=2009

Schneier on Security
A blog covering security and security technology

Microsoft RC4 Flaw, January 18, 2005

One of the most important rules of stream ciphers is to never use the same keystream to encrypt two different documents.
If someone does, you can break the encryption by XORing the two ciphertext streams together.
The keystream drops out, and you end up with plaintext XORed with plaintext –
and you can easily recover the two plaintexts using letter frequency analysis and other basic techniques.

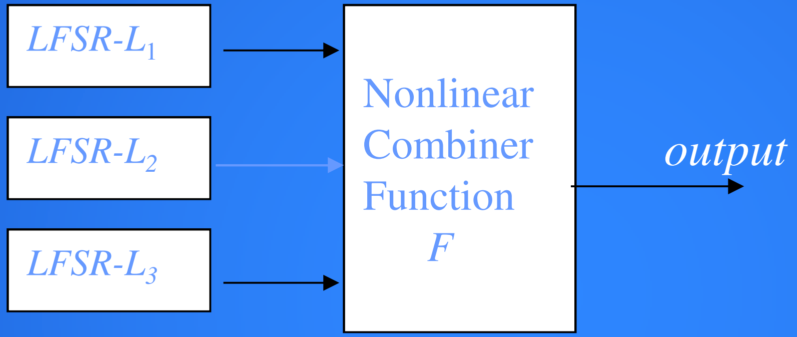
It's an amateur crypto mistake.

The easy way to prevent this attack is to use a unique initialization vector (IV) in addition to the key whenever you encrypt a document.
Microsoft uses the RC4 stream cipher in both Word and Excel.
And they make this mistake. Hongjun Wu has details (link is a PDF).
In this report, we point out a serious security flaw in Microsoft Word and Excel.
The stream cipher RC4 [9] with key length up to 128 bits is used in Microsoft Word and Excel to protect the documents.
But when an encrypted document gets modified and saved, the initialization vector remains the same and thus the same keystream generated from RC4 is applied to encrypt the different versions of that document.
The consequence is disastrous since a lot of information of the document could be recovered easily.
This isn't new. Microsoft made the same mistake in 1999 with RC4 in WinNT Syskey.
Five years later, Microsoft has the same flaw in other products.

Posted on January 18, 2005 at 09:00 AM • 23 Comments • View Blog Reactions

.

Nonlinear combination Generators (1)



example: $F(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3$

maximum period:

$$T = (2^{L_1} - 1)\cdot(2^{L_2} - 1)\cdot(2^{L_3} - 1)$$

Nonlinear combination Generators (1)

$$F(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3 \oplus x_3$$

x_1	x_2	x_3	$z = F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- The combiner function is balanced (#1's = #0's).
- However, the correlation probability,

$$P(z = x_1) = 3/4.$$

- Geffe generator is not secure.

Some examples

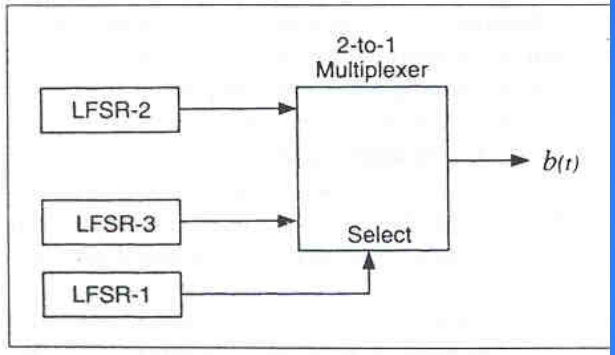


Figure 16.6 Geffe generator.

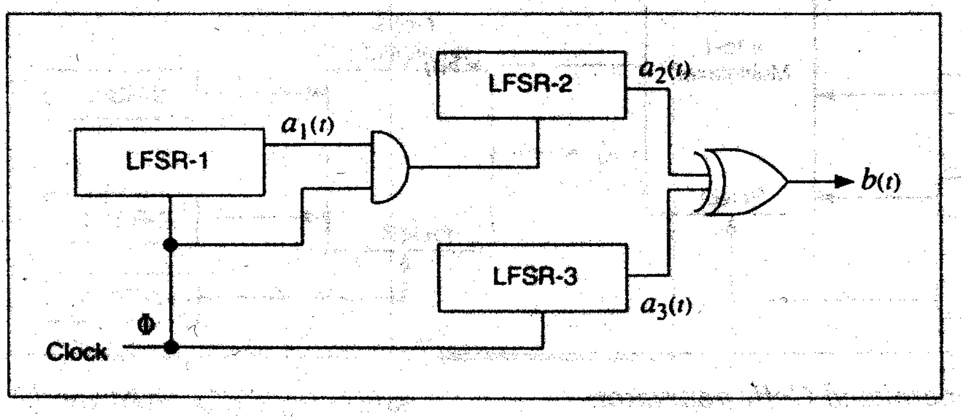


Figure 16.9 Beth-Piper stop-and-go generator.

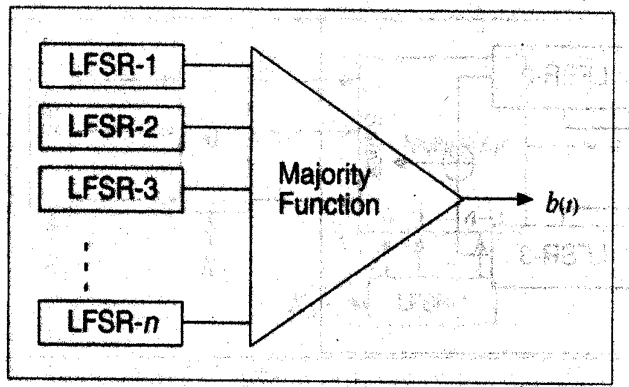


Figure 16.12 Threshold generator.

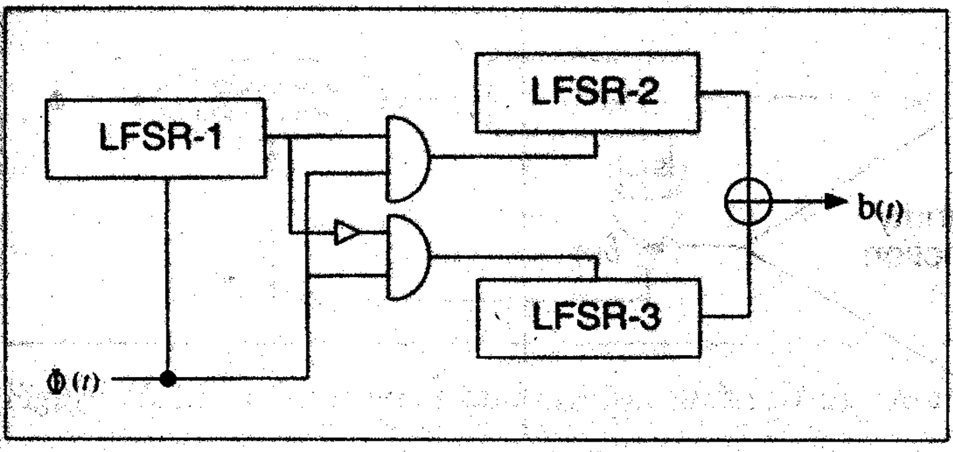
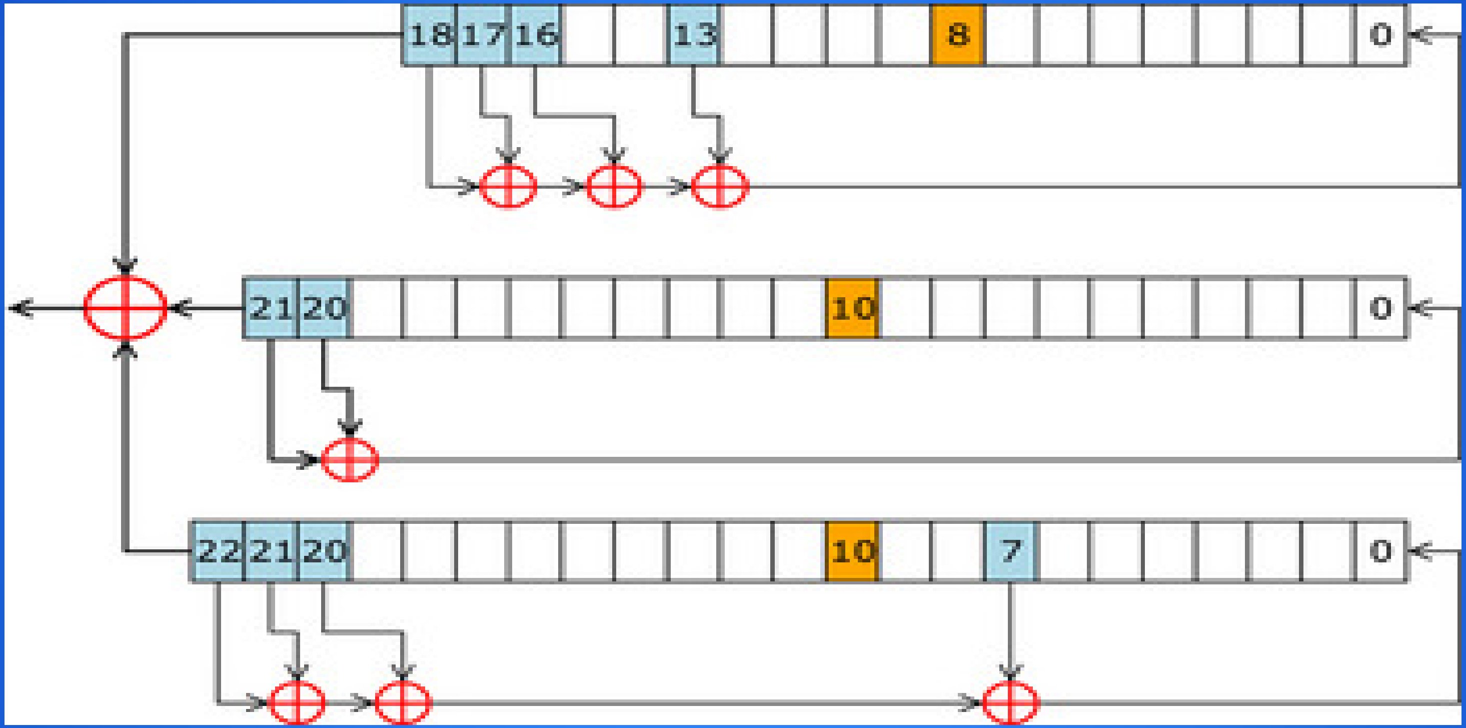


Figure 16.10 Alternating stop-and-go generator.

GSM randomizer A5/1

http://en.wikipedia.org/wiki/A5/1



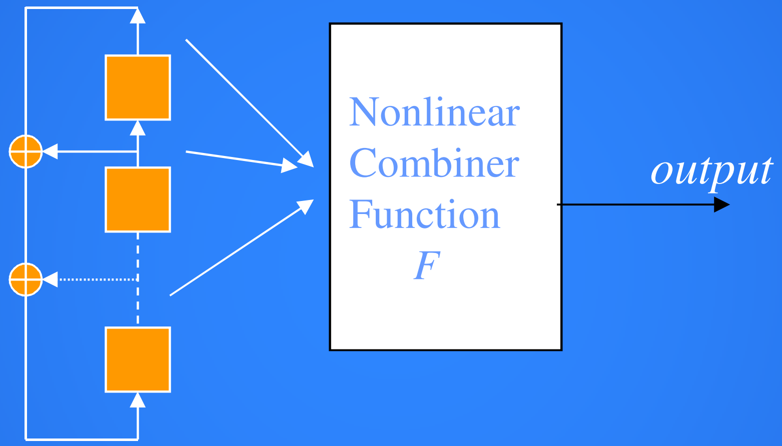
The A5/1 stream cipher uses three LFSRs. A register is clocked if its clocking bit (orange) agrees with the majority of the clocking bits of all three registers.

Hence at each step two or three registers are clocked, and each register steps with probability 3/4.

Ekdahl and Johannson (2003) published an attack on the initialisation procedure which breaks A5/1 in a few minutes using 2–5 minutes of conversation plaintext.

Nonlinear combination Generators (2)

One more option:



Input to the combiner is the content of some of the memory elements

A simple generator with good properties?

```
Start:                    $0 < p, x_0 < 1$;
    for $i \geq 0$,
        if $0 < x_i < p$
                    $R = 1$;   $x_{i+1} := x_i / p$
        if $p < x_i < 1$
                    $R = 0$;   $x_{i+1} := (x_i - p)/(1 - p)$
```

Q: Implement this generator and

- report the problems encountered
- measure a statistical property and compare with the theory