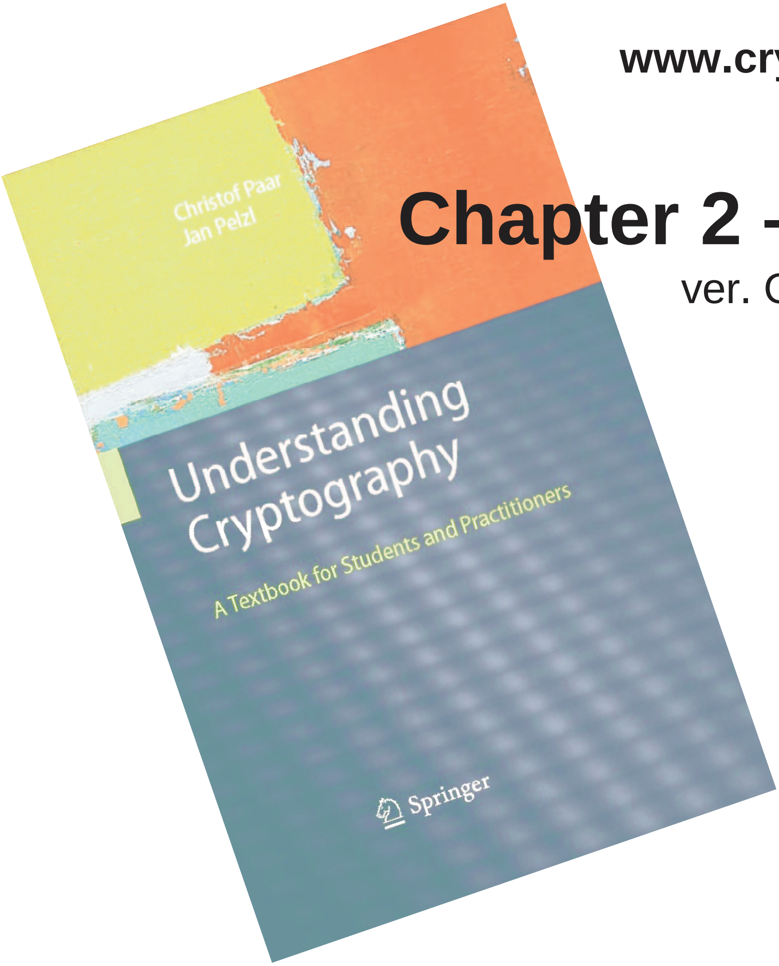
Understanding Cryptography – A Textbook for Students and Practitioners

by Christof Paar and Jan Pelzl

www.crypto-textbook.com



Chapter 2 – Stream Ciphers

ver. October 29, 2009

These slides were prepared by Thomas Eisenbarth, Christof Paar and Jan Pelzl

Some legal stuff (sorry): Terms of Use

- The slides can used free of charge. All copyrights for the slides remain with the authors.
- The title of the accompanying book “Understanding Cryptography” by Springer and the author’s names must remain on each slide.
- If the slides are modified, appropriate credits to the book authors and the book title must remain within the slides.
- It is not permitted to reproduce parts or all of the slides in printed form whatsoever without written consent by the authors.

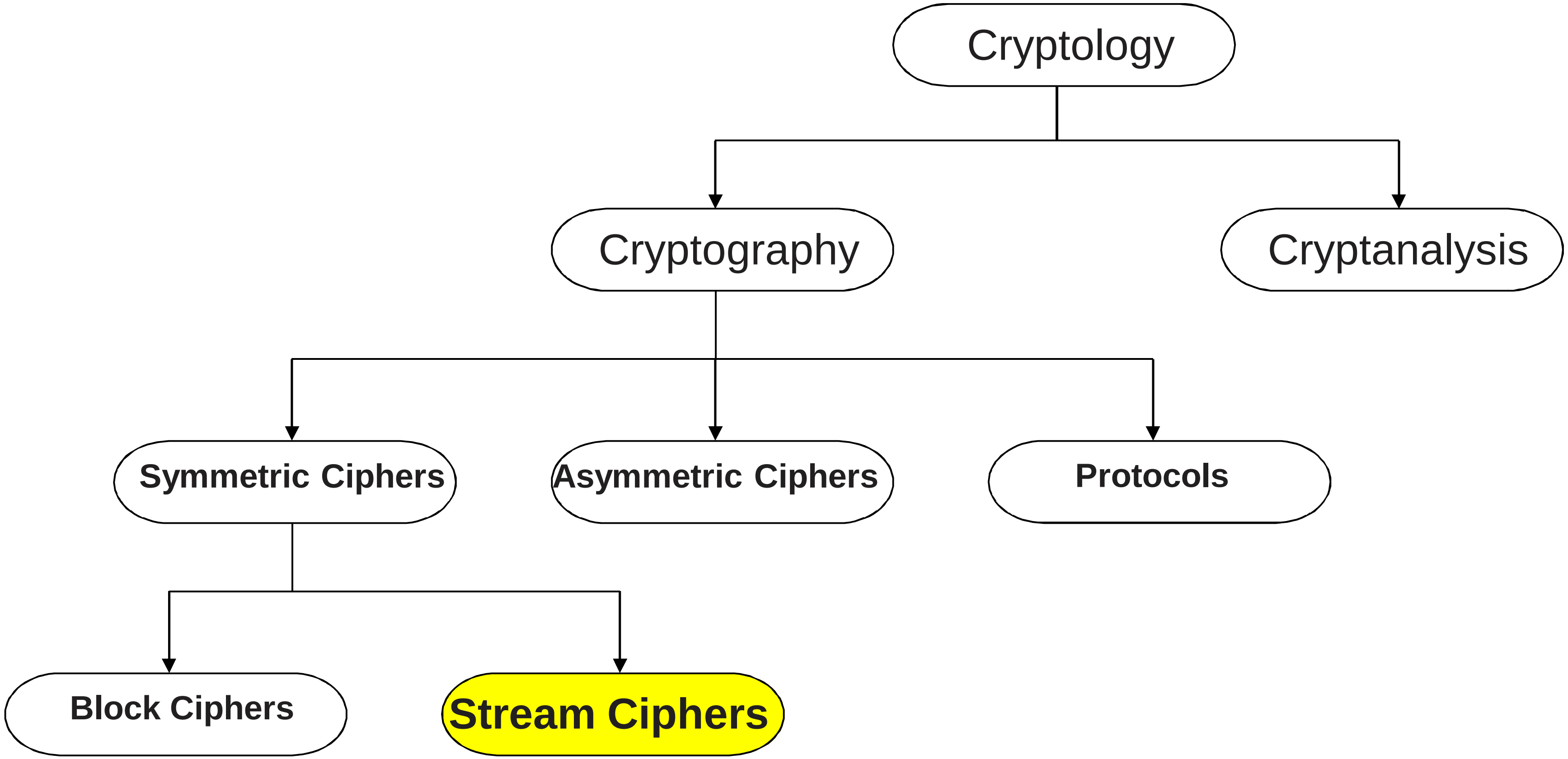
Content of this Chapter

- Intro to stream ciphers
- Random number generators (RNGs)
- One-Time Pad (OTP)
- Linear feedback shift registers (LFSRs)
- Trivium: a modern stream cipher

Content of this Chapter

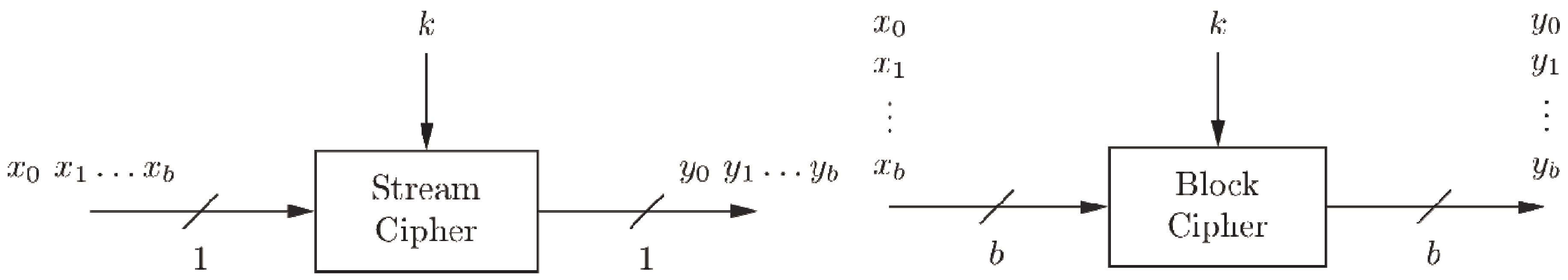
- **Intro to stream ciphers**
- Random number generators (RNGs)
- One-Time Pad (OTP)
- Linear feedback shift registers (LFSRs)
- Trivium: a modern stream cipher

Stream Ciphers in the Field of Cryptology



Stream Ciphers were invented in 1917 by Gilbert Vernam

Stream Cipher vs. Block Cipher

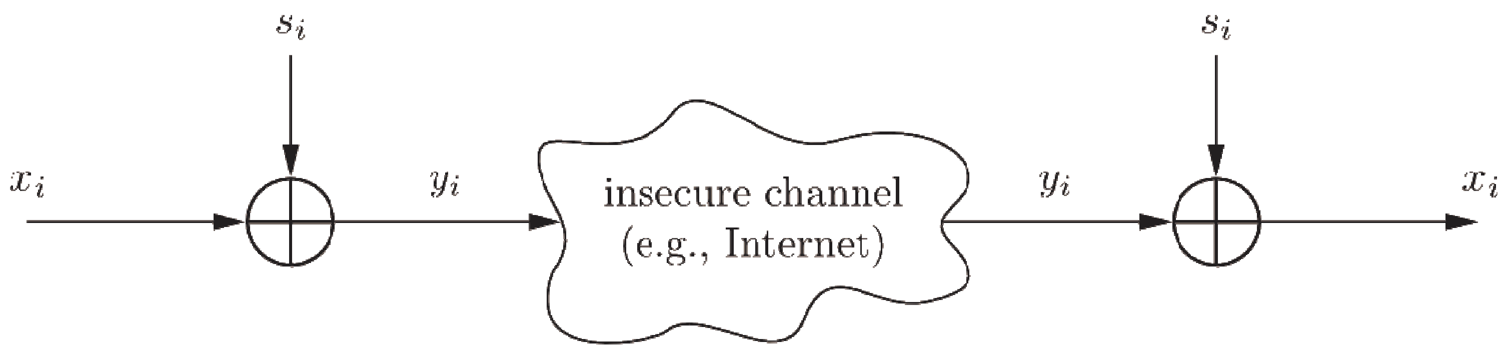


- **Stream Ciphers**
  - Encrypt bits individually
  - Usually small and fast → common in embedded devices (e.g., A5/1 for GSM phones)

- **Block Ciphers:**
  - Always encrypt a full block (several bits)
  - Are common for Internet applications

Encryption and Decryption with Stream Ciphers

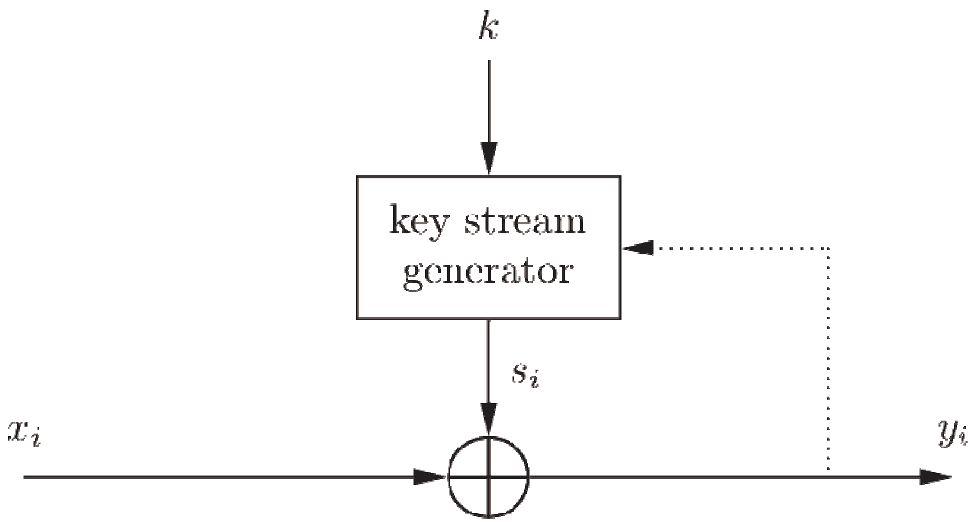
Plaintext x_i, ciphertext y_i and key stream s_i consist of individual bits



- Encryption and decryption are simple additions modulo 2 (aka XOR)
- Encryption and decryption are the same functions

- **Encryption:** $y_i = e_{s_i}(x_i) = x_i + s_i \bmod 2$ $\qquad x_i, y_i, s_i \in \{0,1\}$
- **Decryption:** $x_i = e_{s_i}(y_i) = y_i + s_i \bmod 2$

Synchronous vs. Asynchronous Stream Cipher



- Security of stream cipher depends entirely on the key stream s_i :
  - Should be **random** , i.e., $Pr(s_i = 0) = Pr(s_i = 1) = 0.5$
  - Must be **reproducible** by sender and receiver

- **Synchronous Stream Cipher**
  - Key stream depend only on the key (and possibly an initialization vector IV)

- **Asynchronous Stream Ciphers**
  - Key stream depends also on the ciphertext (dotted feedback enabled)

Why is Modulo 2 Addition a Good Encryption Function?

- Modulo 2 addition is equivalent to XOR operation

- For perfectly random key stream s_i , each ciphertext output bit has a 50% chance to be 0 or 1

  → Good statistic property for ciphertext

- Inverting XOR is simple, since it is the same XOR operation

x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

Stream Cipher: Throughput

Performance comparison of symmetric ciphers (Pentium4):

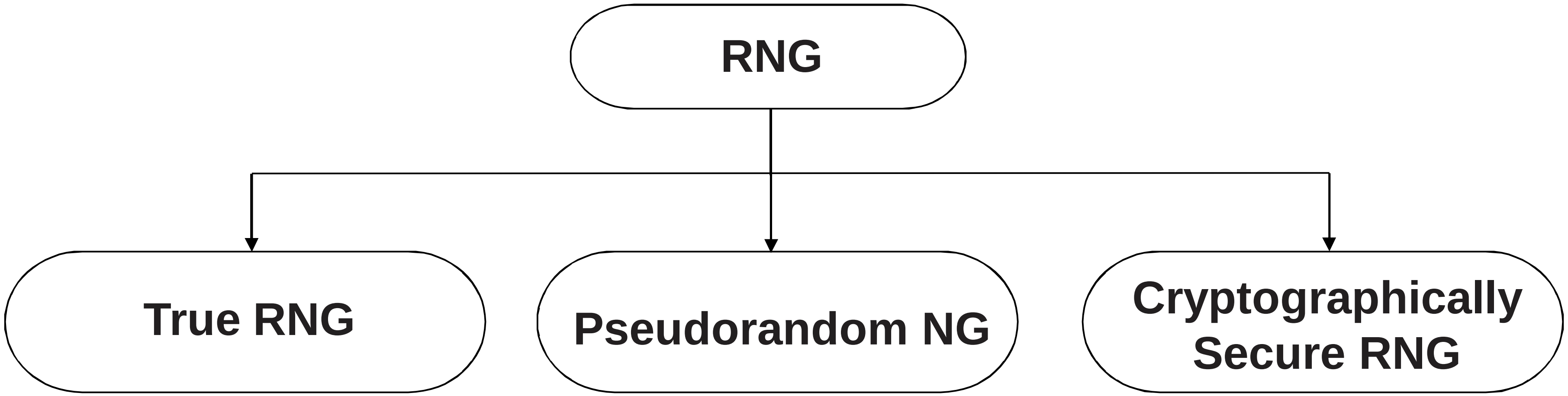
Cipher	Key length	Mbit/s
DES	56	36.95
3DES	112	13.32
AES	128	51.19
RC4 (stream cipher)	(choosable)	211.34

Source: Zhao et al., Anatomy and Performance of SSL Processing, ISPASS 2005

Content of this Chapter

- Intro to stream ciphers
- **Random number generators (RNGs)**
- One-Time Pad (OTP)
- Linear feedback shift registers (LFSRs)
- Trivium: a modern stream cipher

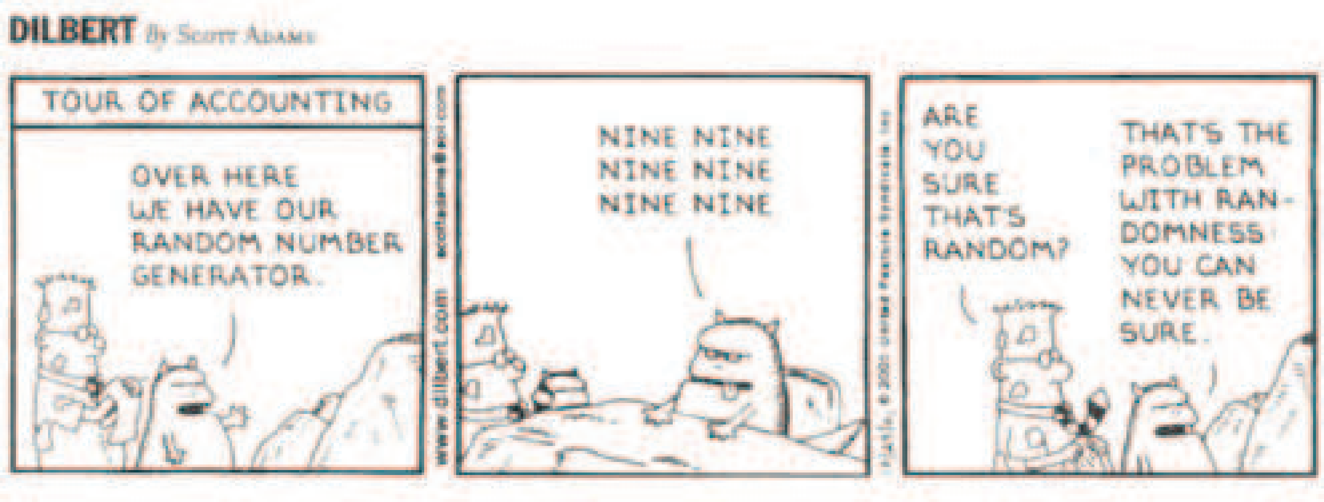
- **Random number generators (RNGs)**



True Random Number Generators (TRNGs)

- Based on physical random processes: coin flipping, dice rolling, semiconductor noise, radioactive decay, mouse movement, clock jitter of digital circuits
- Output stream s_i should have good statistical properties: $Pr(s_i = 0) = Pr(s_i = 1) = 50\%$ (often achieved by post-processing)
- Output can neither be predicted nor be reproduced

Typically used for generation of keys, nonces (used only-once values) and for many other purposes



Pseudorandom Number Generator (PRNG)

- Generate sequences from initial seed value
- Typically, output stream has good statistical properties
- Output can be reproduced and can be predicted

Often computed in a recursive way:

$$s_0 = seed$$

$$s_{i+1} = f(s_i, s_{i-1}, \ldots, s_{i-t})$$

Example: *rand() function* in ANSI C:

$$s_0 = 12345$$

$$s_{i+1} = 1103515245 s_i + 12345 \bmod 2^{31}$$

Most PRNGs have bad cryptographic properties!

Cryptanalyzing a Simple PRNG

Simple PRNG: **Linear Congruential Generator**

$$S_0 = seed$$

$$S_{i+1} = AS_i + B \bmod m$$

Assume

- unknown A, B *and* S_0 as key
- Size of A, B and S_i to be 100 bit
- 300 bit of output are known, i.e. S_1, S_2 and S_3

Solving

$$S_2 = AS_1 + B \bmod m$$

$$S_3 = AS_2 + B \bmod m$$

…directly reveals A and B. All S_i can be computed easily!

Bad cryptographic properties due to the linearity of most PRNGs

Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

- Special PRNG with additional property:
  - Output must be **unpredictable**

More precisely: Given *n* consecutive bits of output s_i , the following output bits s_{n+1} cannot be predicted (in polynomial time).

- Needed in cryptography, in particular for stream ciphers

- Remark: There are almost no other applications that need unpredictability, whereas many, many (technical) systems need PRNGs.

Content of this Chapter

- Intro to stream ciphers
- Random number generators (RNGs)
- **One-Time Pad (OTP)**
- Linear feedback shift registers (LFSRs)
- Trivium: a modern stream cipher

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

- A cryptosystem is unconditionally secure if it cannot be broken even with *infinite* computational resources

One-Time Pad

- A cryptosystem developed by Mauborgne that is based on Vernam's stream cipher:
- Properties:

  Let the plaintext, ciphertext and key consist of individual bits $x_i, y_i, k_i \in \{0,1\}$.

Encryption:	$e_{k_i}(x_i) = x_i \oplus k_i$.
Decryption:	$d_{k_i}(y_i) = y_i \oplus k_i$

OTP is unconditionally secure if and only if the key k_i. is used once!

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

$$y_0 = x_0 \oplus k_0$$

$$y_1 = x_1 \oplus k_1$$

$$\vdots$$

Every equation is a linear equation with two unknowns

⇒ for every y_i are $x_i = 0$ and $x_i = 1$ equiprobable!

⇒ This is true iff $k_0, k_1, \ldots$ are independent, i.e., all k_i have to be generated truly random

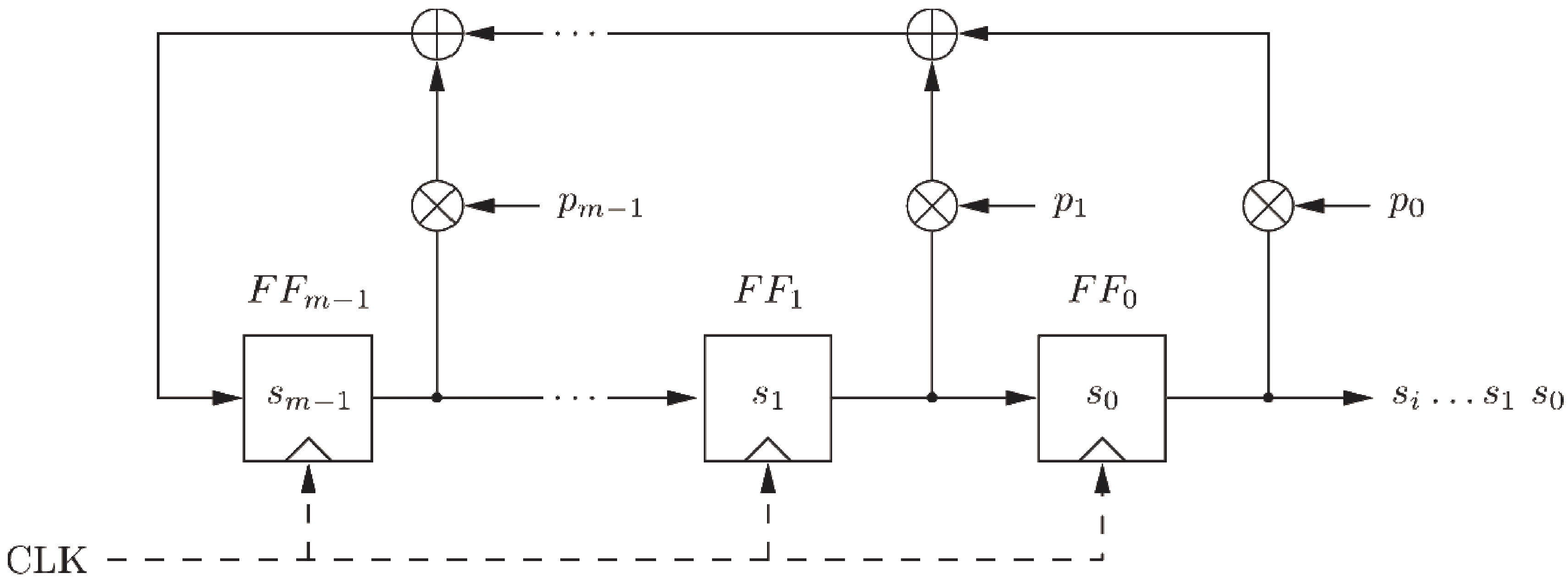
⇒ It can be shown that this systems can *provably* not be solved.

Disadvantage: For almost all applications the OTP is **impractical** since the key must be as long as the message! (Imagine you have to encrypt a 1GByte email attachment.)

Content of this Chapter

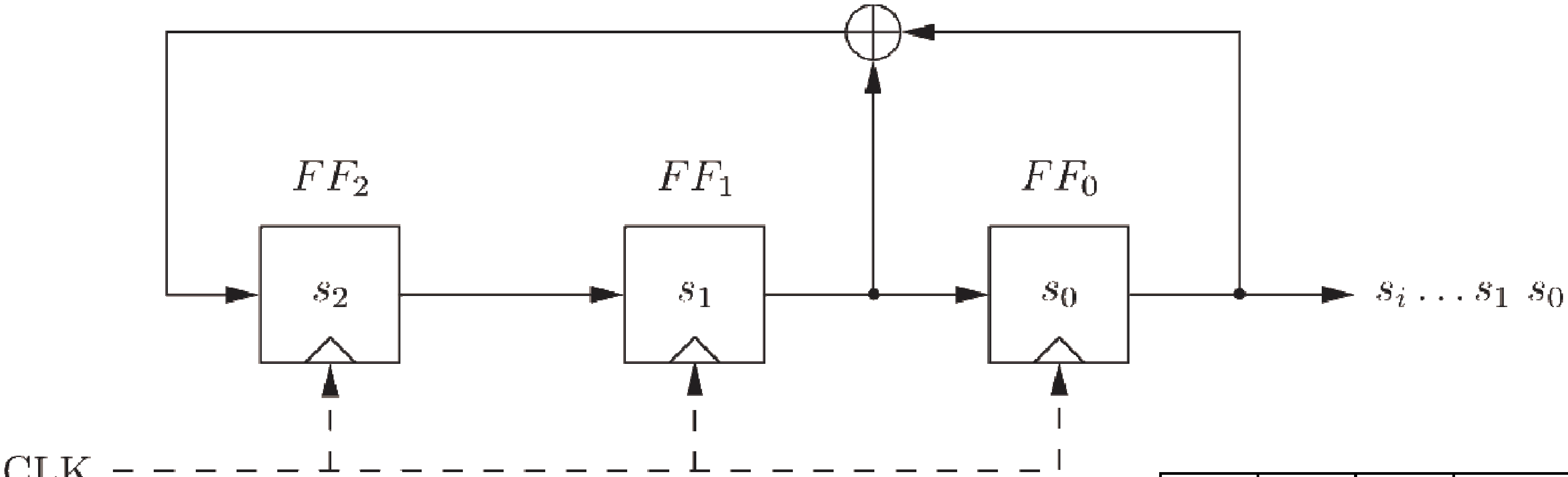
- Intro to stream ciphers
- Random number generators (RNGs)
- One-Time Pad (OTP)
- **Linear feedback shift registers (LFSRs)**
- Trivium: a modern stream cipher

Linear Feedback Shift Registers (LFSRs)



- Concatenated *flip-flops (FF*), i.e., a shift register together with a feedback path
- Feedback computes fresh input by XOR of certain state bits
- *Degree m* given by number of storage elements
- If $p_i = 1$, the feedback connection is present ("closed switch), otherwise there is not feedback from this flip-flop ("open switch")
- Output sequence repeats periodically
- Maximum output length**:** 2^m-1

Linear Feedback Shift Registers (LFSRs): Example with m=3



clk	*FF₂*	*FF₁*	*FF₀=sᵢ*
0	1	0	0
1	0	1	0
2	1	0	1
3	1	1	0
4	1	1	1
5	0	1	1
6	0	0	1
7	1	0	0
8	0	1	0

- LFSR output described by recursive equation:

$$s_{i+3} = s_{i+1} + s_i \bmod 2$$

- Maximum output length (of 2^3-1=7) achieved only for certain feedback configurations, .e.g., the one shown here.

Security of LFSRs

LFSRs typically described by polynomials:

$$P(x) = x^m + p_{l-1}x^{m-1} + \ldots + p_1 x + p_0$$

- Single LFSRs generate highly predictable output
- If 2*m* output bits of an LFSR of degree *m* are known, the feedback coefficients p_i of the LFSR can be found by solving a system of linear equations*

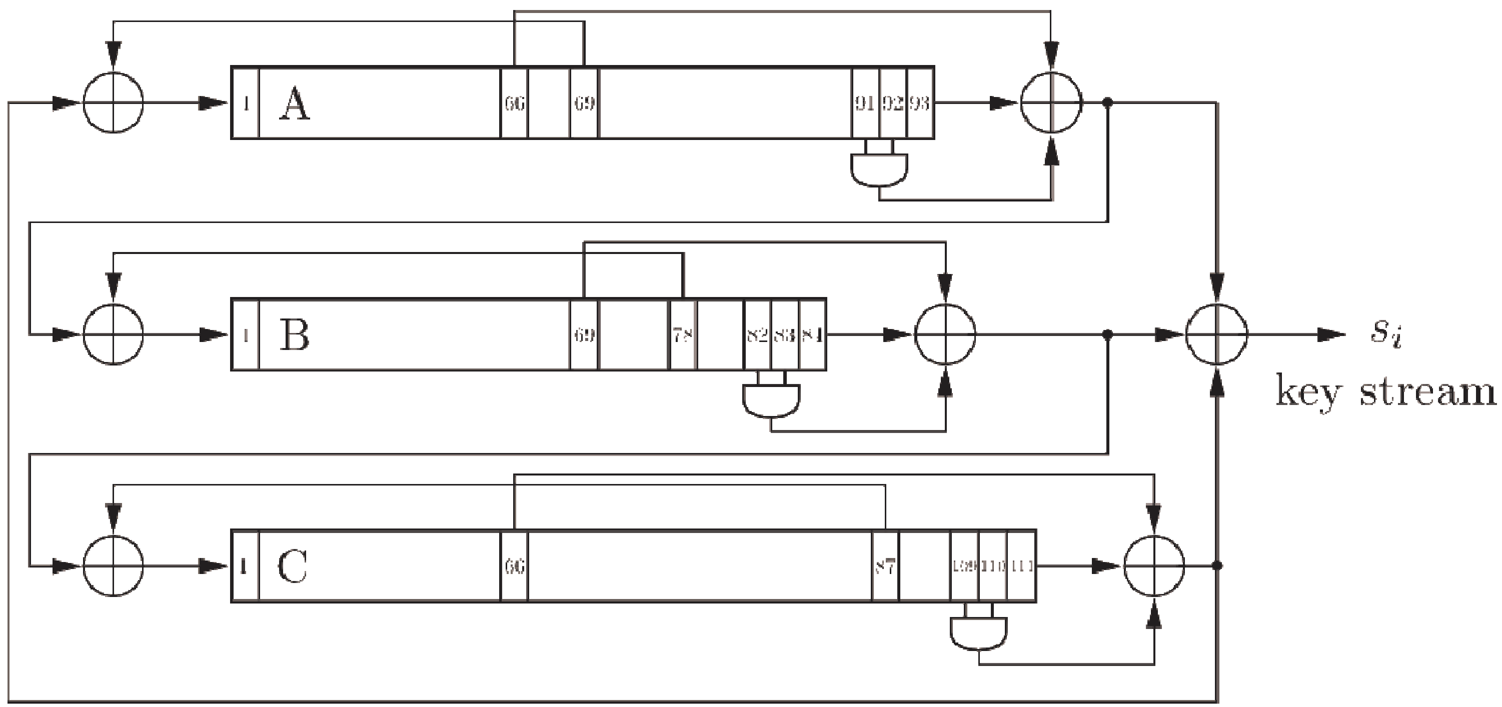
- Because of this many stream ciphers use **combinations** of LFSRs

*See Chapter 2 of *Understanding Cryptography* for further details.

Content of this Chapter

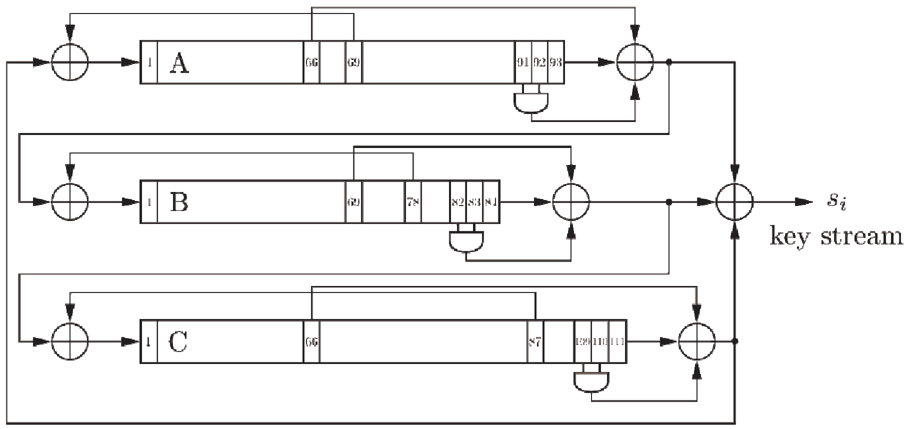
- Intro to stream ciphers
- Random number generators (RNGs)
- One-Time Pad (OTP)
- Linear feedback shift registers (LFSRs)
- **Trivium: a modern stream cipher**

A Modern Stream Cipher - Trivium



- Three *nonlinear* LFSRs (NLFSR) of length 93, 84, 111
- XOR-Sum of all three NLFSR outputs generates key stream s_i
- Small in Hardware:
  - Total register count: 288
  - Non-linearity: 3 AND-Gates
  - 7 XOR-Gates (4 with three inputs)

Trivium



Initialization:

- Load 80-bit IV into A
- Load 80-bit key into B
- Set c_{109} , c_{110} , c_{111} =1, all other bits 0

Warm-Up:

- Clock cipher 4 x 288 = 1152 times without generating output

Encryption:

- XOR-Sum of all three NLFSR outputs generates key stream s_i

Design can be parallelized to produce up to 64 bits of output per clock cycle

	Register length	Feedback bit	Feedforward bit	AND inputs
A	93	69	66	91, 92
B	84	78	69	82, 83
C	111	87	66	109, 110

Lessons Learned

- Stream ciphers are less popular than block ciphers in most domains such as Internet security. There are exceptions, for instance, the popular stream cipher RC4.
- Stream ciphers sometimes require fewer resources, e.g., code size or chip area, for implementation than block ciphers, and they are attractive for use in constrained environments such as cell phones.
- The requirements for a *cryptographically secure pseudorandom number generator* are far more demanding than the requirements for pseudorandom number generators used in other applications such as testing or simulation
- The One-Time Pad is a provable secure symmetric cipher. However, it is highly impractical for most applications because the key length has to equal the message length.
- Single LFSRs make poor stream ciphers despite their good statistical properties. However, careful combinations of several LFSR can yield strong ciphers.