Sicherheit in Kommunikationsnetzen
(Network Security)

Stream Ciphers

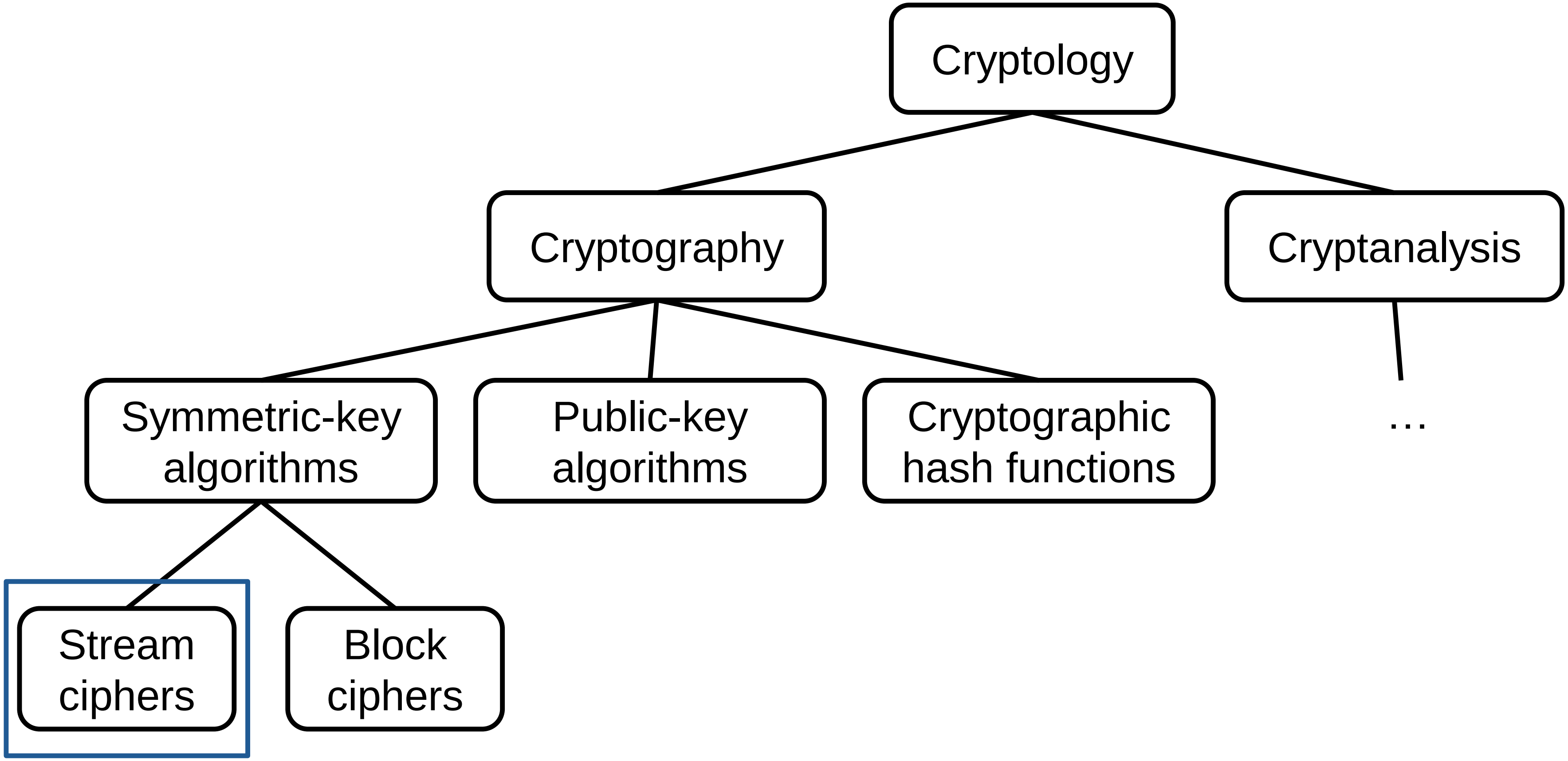
Dr.-Ing. Matthäus Wander

Universität Duisburg-Essen

Motivation

- Vigenère with short key is insecure
  - Plain: E S S E N K E N N T E S S E N
  - Key: V E N U S V E N U S V E N U S
  - Cipher: Z W F Y F F I A H L Z W F Y F
- Vigenère with long key appears secure
  - Plain: E S S E N K E N N T E S S E N
  - Key: T N P X U G T T Q E G C F R Y
  - Cipher: X F H B H Q X G D X K U X V L
- Idea: apply a long keystream to plaintext

Cryptography: Overview



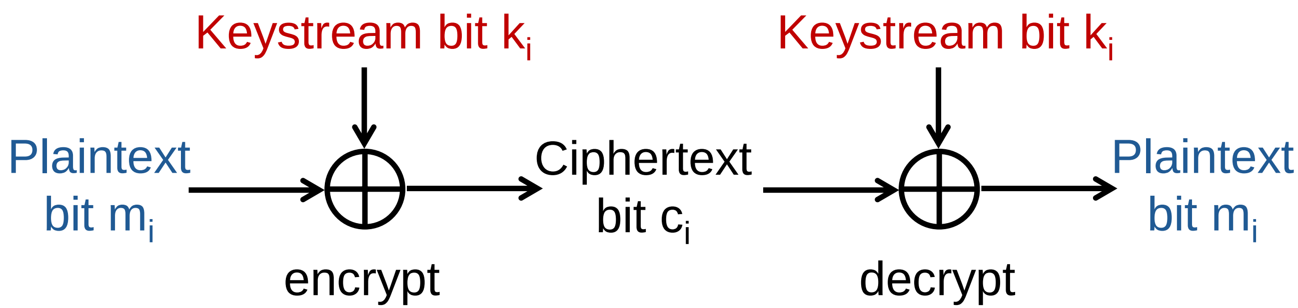
Vernam Cipher

- Gilbert Vernam invented a stream cipher for teleprinters („Fernschreiber“) in 1917
  - Characters are encoded in 5-bit Baudot code
  - En/decryption uses exclusive-or (XOR) operation
  - XOR (we write: ⊕) is identical to addition modulo 2

A	B	A⊕B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

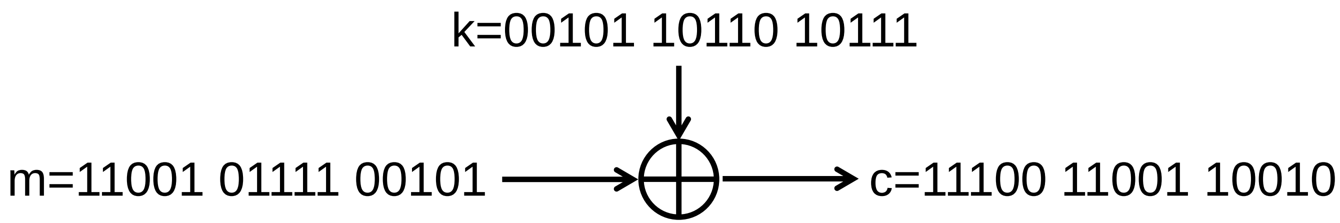
Source: ciphermachines.com

Vernam Cipher (2)



- Stream ciphers encrypt each bit individually
  - $m_i \oplus k_i = c_i$ is very simple, efficient, and self-inverse
  - e_k and d_k are identical, thus $e_k(e_k(m)) = m$
- Challenge: how to generate the keystream?
  - Vernam suggested a key tape running in loop
  - Repeating key vulnerable with sufficient ciphertext

One-Time Pad (OTP)



- Idea: use a random key as long as the plaintext
  - Such a cipher is called one-time pad („Einmalblock“)
- Key should be a truly random bit sequence
  - Bit values b ∈ {0, 1} are distributed uniformly, i.e. occur with probability of ½ each
  - **and** key bits are unpredictable
- Key should be only used once

Cryptanalysis of One-Time Pad

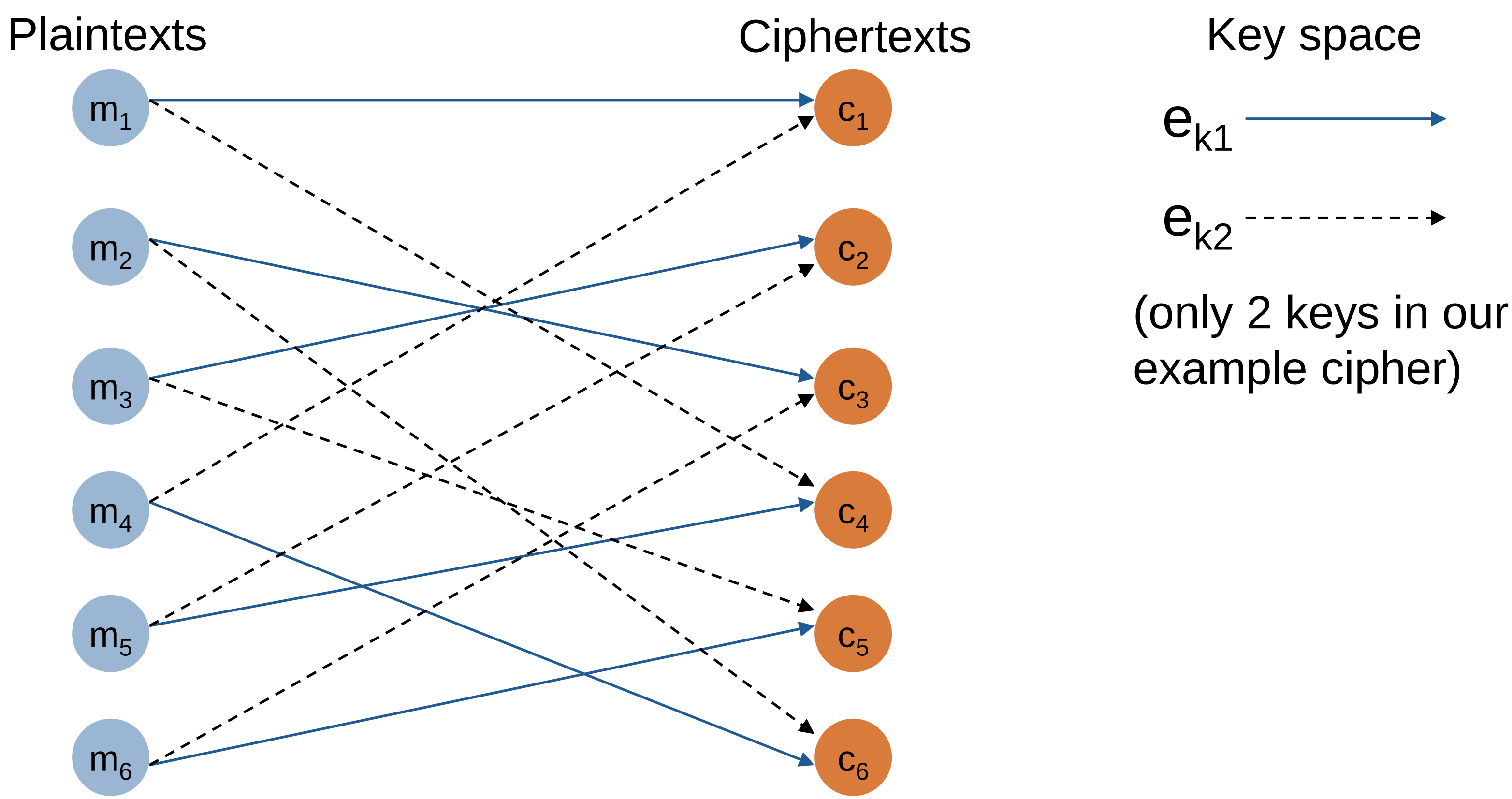
- Size of key space $\mathcal{K}$?
  - 2^n for a message of n bits
- Example: 9 × 5-bit chars encrypted with OTP
  - $|\mathcal{K}| = 2^{45} \approx 3.5 \times 10^{13}$ (not too much for computers)
- Perform ciphertext-only brute-force attack
  - yields $d_{k1}(c)$ = „**ATTACKNOW**"
  - but also $d_{k2}(c)$ = „**SURRENDER**"
  - and any other plaintext, e.g. $d_{k3}(c)$ = „**KARTOFFEL**"
- Which key is the right one? We don‘t know

Cryptanalysis of One-Time Pad (2)

- With a truly random key, the ciphertext has no statistical relationship to the plaintext
- One-time pad provides perfect secrecy
  - If used correctly, OTP cannot be deciphered
- Perfect secrecy is unconditionally secure
  - Even against attacks with unlimited computing power
- Most ciphers are only computationally secure
  - They can be broken with a brute-force attack, but with infeasible costs

Perfect Secrecy

- Example cipher maps a set of plaintexts $\mathcal{M}$ to a set of ciphertexts $\mathcal{C}$ with $|\mathcal{K}|=2$



Perfect Secrecy (2)

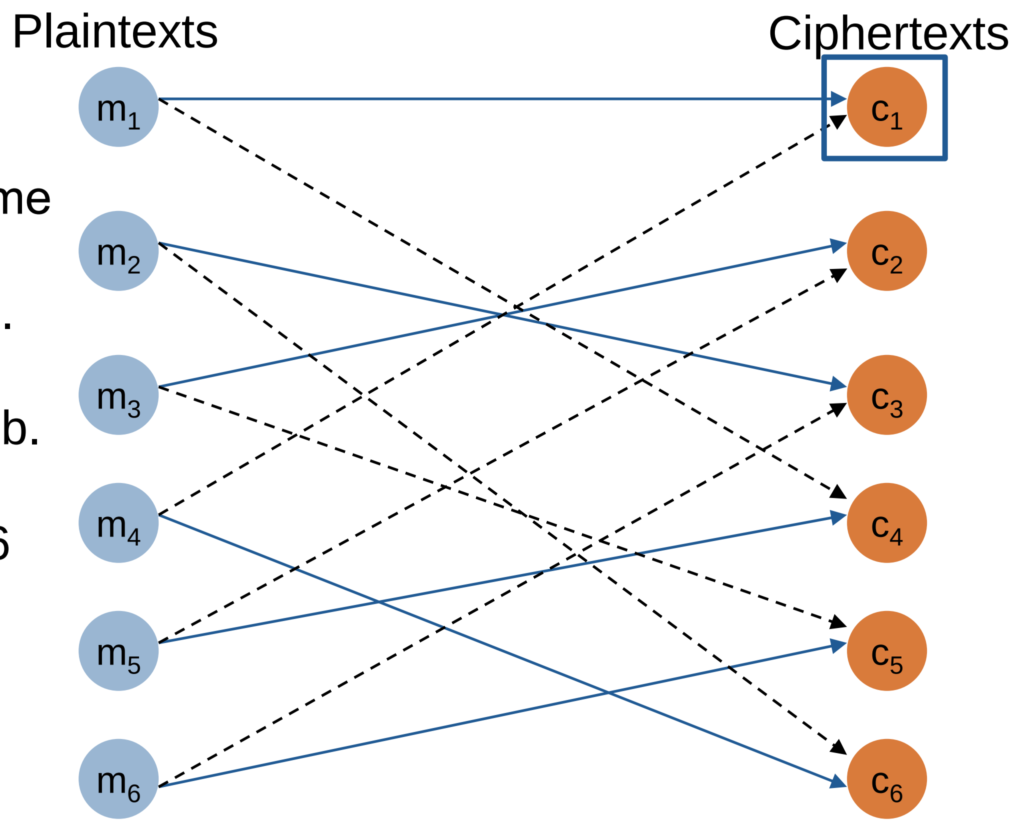
- A priori probability
  - What is the probability that a certain plaintext m $\in \mathcal{M}$ occurs?
  - „HALLO“ may be more likely than „XRZDY“
  - Let Pr(m) be the a priori probability of plaintext m
- A posteriori probability
  - Given a certain ciphertext c $\in \mathcal{C}$, what is the probability that c originates from plaintext m?
  - Some plaintext-ciphertext pairs may be improbable
  - Let Pr(m|c) be the probability that m maps to c

Perfect Secrecy (3)

- Example cipher maps a set of plaintexts $\mathcal{M}$ to a set of ciphertexts $\mathcal{C}$ with $|\mathcal{K}|=2$

Let's assume uniform distribution.

A priori prob. for all m: Pr(m) = 1/6



A posteriori prob. for c_1:

$Pr(m_1|c_1) = 0.5$
$Pr(m_2|c_1) = 0$
$Pr(m_3|c_1) = 0$
$Pr(m_4|c_1) = 0.5$
$Pr(m_5|c_1) = 0$
$Pr(m_6|c_1) = 0$

Perfect Secrecy (4)

- Our example does not provide perfect secrecy
- Given a ciphertext c_1, the attacker derives that m_1 or m_4 is the plaintext (out of six plaintexts)
  - $Pr(m_1|c_1) > Pr(m_1)$ ⇨ likely plaintext
  - $Pr(m_2|c_1) < Pr(m_2)$ ⇨ unlikely/impossible plaintext
- Perfect secrecy: $Pr(m|c)=Pr(m)$ for all m and c
  - A posteriori probability equals a priori probability
  - The attacker does not learn anything from c
  - That is, m and c are statistically independent

Properties of Perfectly Secret Ciphers

- Properties of a cipher with perfect secrecy
- Lemma 1: There exists a $k \in \mathcal{K}$ so that $e_k(m)=c$ for every pair (m, c) with $m \in \mathcal{M}$, $c \in \mathcal{C}$
  - Every plaintext can be mapped to every ciphertext
- Proof
  - With perfect secrecy $Pr(m|c)=Pr(m)$ for all m, c
  - $Pr(m) > 0$ because every plaintext is possible
  - Thus $Pr(m|c) > 0$,
    i.e. there is a key k, which maps every m to every c

Properties of Perfectly Secret Ciphers (2)

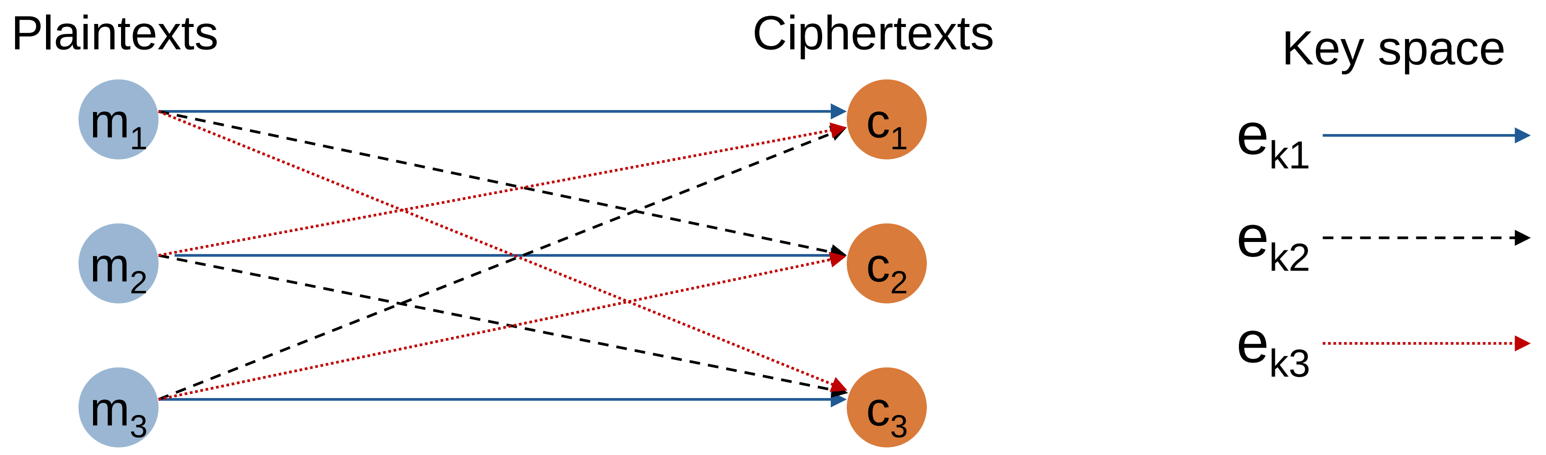
- <u>Lemma 2</u>: $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}|$
  - There are at least as many keys as ciphertexts as plaintexts
- Proof
  - $|\mathcal{C}| \geq |\mathcal{M}|$ is necessary,
    otherwise $e_k(m)=c$ and $e_k(m')=c$ with $m \neq m'$,
    which means $d_k(c)$ would be ambiguous
  - $|\mathcal{K}| \geq |\mathcal{C}|$ follows from Lemma 1:
    $e_k(m)=c$ for every m and c,
    thus we need at least as many keys as ciphertexts

Properties of Perfectly Secret Ciphers (3)

- Assume a cipher with $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{M}|$
- Lemma 3: Each key $k \in \mathcal{K}$ is used with equal probability
  - Keys are chosen randomly
- For a plaintext $m \in \mathcal{M}$ and ciphertext $c \in \mathcal{C}$, there is exactly one $k \in \mathcal{K}$, so that $e_k(m)=c$
- We skip the proof

Example Ciphers with Perfect Secrecy

- Example cipher with perfect secrecy



Example Ciphers with Perfect Secrecy (2)

- Shift ciphers can have perfect secrecy
  - If we restrict $\mathcal{M}$ strictly to single-character messages
- Even if a priori probabilities are non-uniform
  - e.g. „E“ $\in \mathcal{M}$ with Pr(„E“) = 17%
- We receive $c \in \mathcal{C}$, how likely is Pr(„E“ | c)?
  - Keys chosen randomly: $Pr(k) = 1 / |\mathcal{K}|$ for all $k \in \mathcal{K}$
  - All ciphertexts are equally likely to originate from „E“: Pr(„E“ | c_1) = Pr(„E“ | c_2) for all $c_1, c_2 \in \mathcal{C}$
  - Pr(„E“ | c) = Pr(„E“) = 17%
  - We knew that before (a priori) ⇨ no information leak

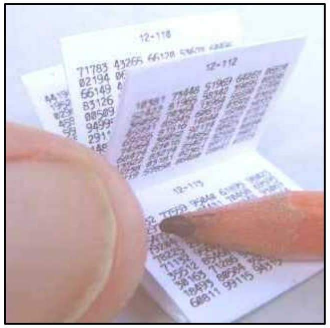
Example Ciphers with Perfect Secrecy (3)

- One-time pad has perfect secrecy
- $\mathcal{K} = \mathcal{C} = \mathcal{M} = \{0, 1\}^n$
  - Sequence of up to n bits
  - $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{M}|$
- Let $k \in \mathcal{K}$ be a random key, which is used once
  - For each $m \in \mathcal{M}$ and $c \in \mathcal{C}$ there is exactly one k, namely $k=m \oplus c$
- Variant of OTP with letters instead of bits:
  - Vigenère with a random, non-repeating key

Number Stations

- Use case for one-time pad: number stations
- Intelligence agencies broadcast messages to their agents over shortwave radio („Kurzwelle“)
  - Shortwave radio travels over long distance (worldwide)
- Message consists of numbers spoken by a voice
  - Intelligible by everyone with a common world band receiver („Weltempfänger“)
  - Unknown encoding, but certainly encrypted with a one-time pad

Author: Oona Räisänen



Source: ciphermachines.com

Security of One-Time Pad

- OTP is rarely used for network communication
  - Impractical and only worth for high security scenarios
- Problem: exchange of key as long as plaintext
  - Generate lots of unpredictable randomness
  - What is the bandwidth of the secure channel?
  - How much key material can we store securely?
  - Key quickly consumed, e.g. with video chat
- Perfect secrecy $\neq$ perfect security
  - A security system might fail to provide confidentiality even with a perfectly secret cipher

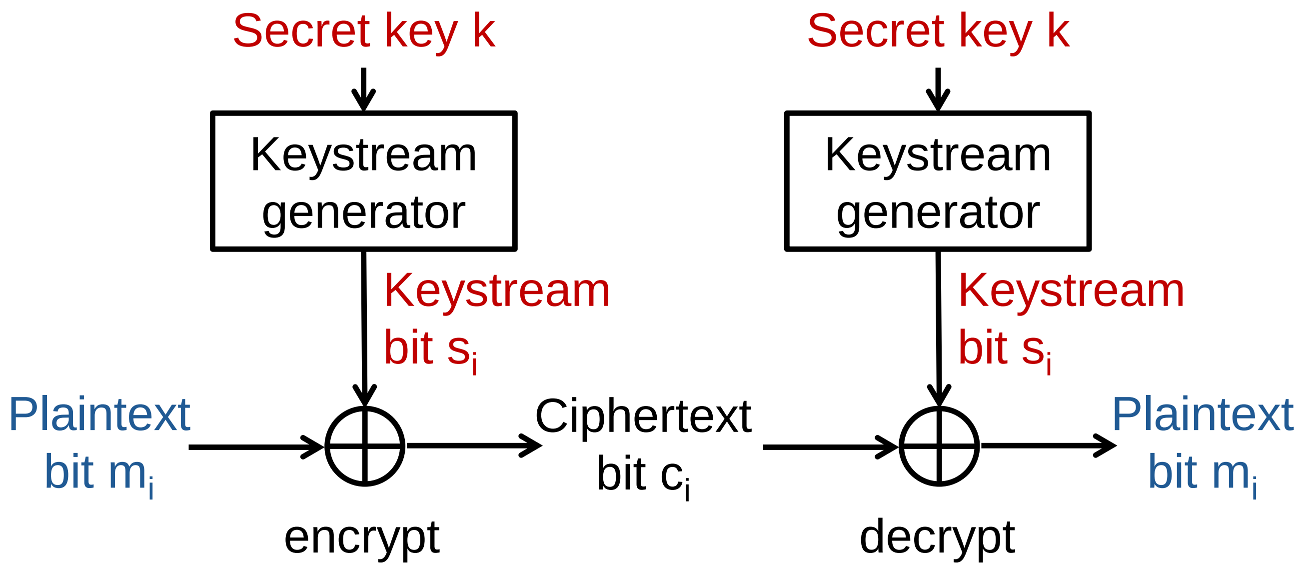
Security of One-Time Pad (2)

- Lack of data integrity (like most ciphers)
  - Known-plaintext attack: attacker can flip bits in ciphertext without knowledge of key

```
               „$“       „1“       „0“       „0“
Plain:    00100100  00110001  00110000  00110000
Key:      unknown
Cipher:   10111110  01100011  10111111  10110110
```

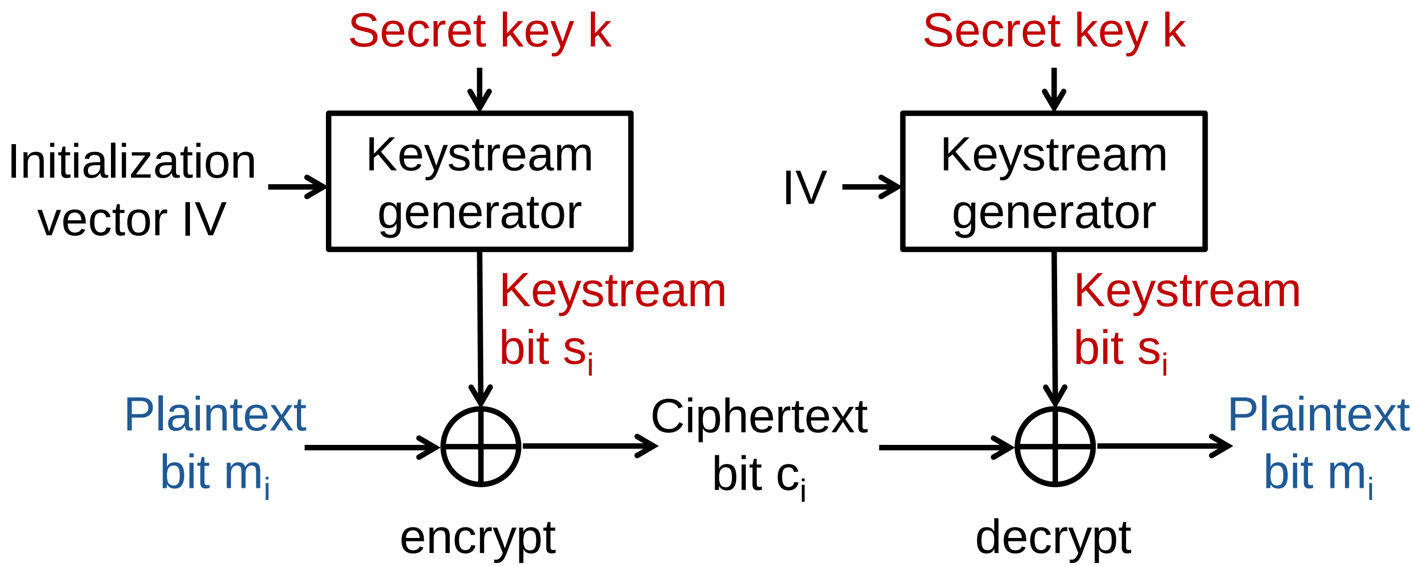
- Can we reuse the keystream?
  - $m_1 \oplus k = c_1$, $m_2 \oplus k = c_2$
  - $c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2$
  - With further analysis, attacker can reveal the plaintext

Stream Ciphers



- Idea: generate keystream with a fixed-size key
- Such a cipher does not have perfect secrecy
  - But is more practical and computationally secure

Stream Ciphers (2)



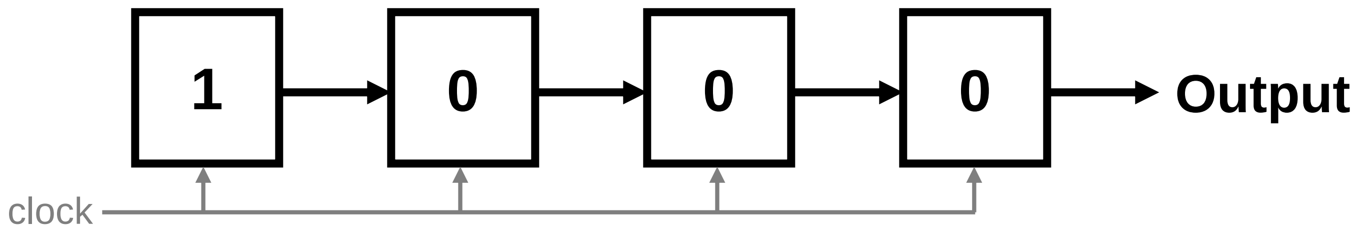
- Additional parameter: initialization vector (IV)
  - Identical key produces identical keystream
- Purpose of IV: randomize the keystream
  - IV does not need to be secret but unique

Design Consideration for Stream Ciphers

- Key stream should have a long period
  - Pseudorandom keystream will repeat eventually
  - Long period ⇨ cryptanalysis harder ⇨ more secure
- Keystream should appear randomly
  - Unpredictable bit sequence
  - Equal distribution of 0 and 1
- Generator initialized with a sufficiently long key
  - Short key space is vulnerable to brute-force attacks
  - State of the art: ≥128-bit keys

Shift Registers

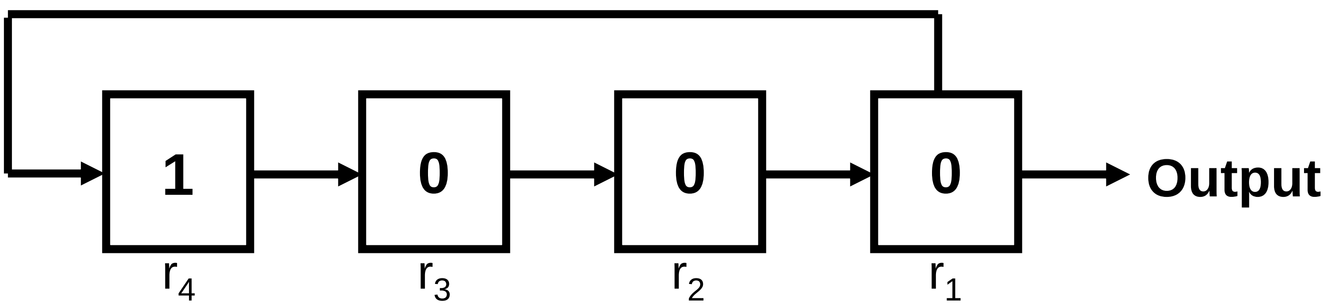
- Idea: generate keystream with shift registers
- Series of flip–flops (1–bit storage elements)
  - In hardware: efficient digital circuit
  - In software: can be simulated with computer program



- With each clock tick, bits are right–shifted
  - One output bit per clock cycle (the right–most bit)

Shift Registers (2)

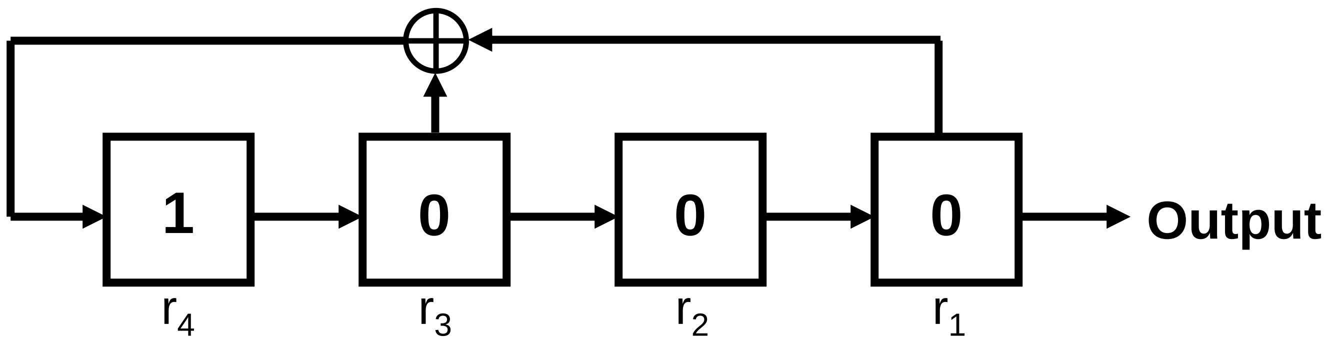
- How to generate new bits?
  - Idea: Create a feedback path



- Output sequence: 0001 0001 0001 ...
- Problem: small period, will repeat after 4 bits
  - Easily predictable

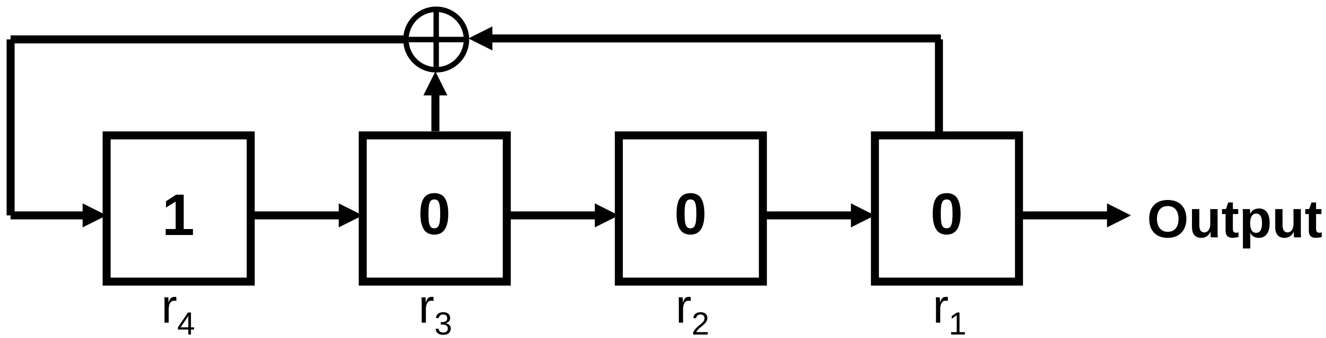
Linear Feedback Shift Registers

- Idea: combine multiple bits for feedback
  - Linear feedback with XOR operation



- Output sequence: $s_1, s_2, s_3, s_4, s_5, s_6, \ldots, s_n$
  - $s_4=1,\ s_3=0,\ s_2=0,\ s_1=0$
  - $s_5 \equiv s_3 + s_1 \bmod 2$ (linear function)
  - $s_6 \equiv s_4 + s_2 \bmod 2$ $s_{i+4} \equiv s_{i+2} + s_i \bmod 2$

Linear Feedback Shift Registers (2)



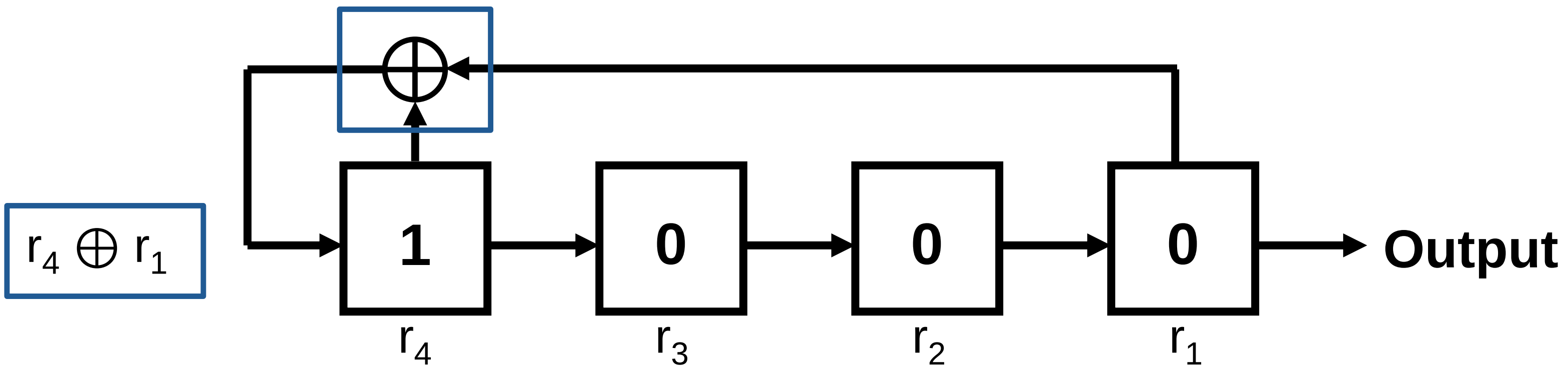
$r_3 \oplus r_1$

r_4	r_3	r_2	r_1
1	0	0	0
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
0	0	0	1
1	0	0	0

Output: 000101 000101 ...

← output repeats with period of length 6

Linear Feedback Shift Registers (3)



r_4	r_3	r_2	r_1
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	0

r_4	r_3	r_2	r_1
1	1	0	1
0	1	1	0
0	0	1	1
1	0	0	1
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

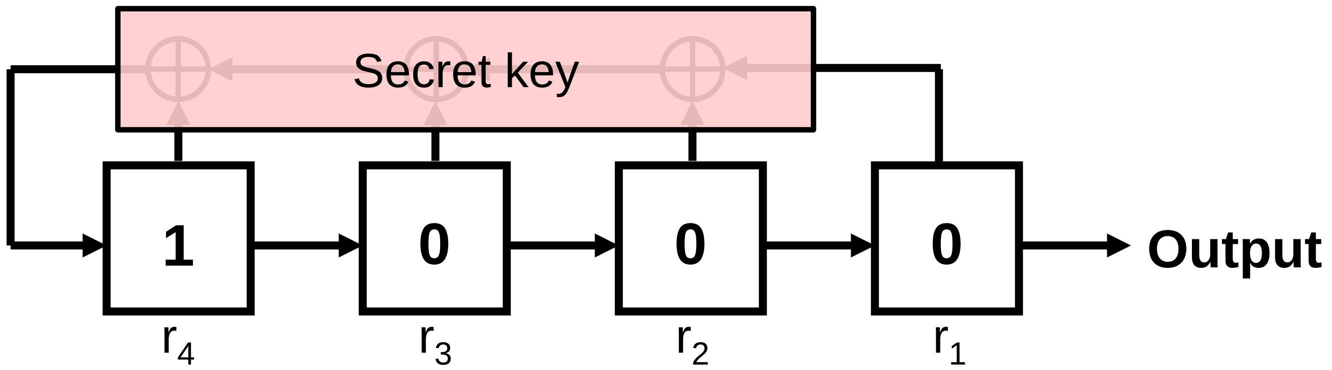
← period length: 15

Period Length of LFSR

- What is the maximum period length?
  - Register of length m has up to 2^m different states
- Special case: $s_m = \ldots = s_4 = s_3 = s_2 = s_1 = 0$ (all zeros)
  - LFSR with an all zero state will only generate „0“ bits
  - Reason: $0 \oplus 0 = 0$
- Maximum period for LFSR of degree m: $2^m - 1$
  - Not all LFSR yield the maximum period (see example)
  - But there are maximum-length LFSR for any degree m

Security of LFSR

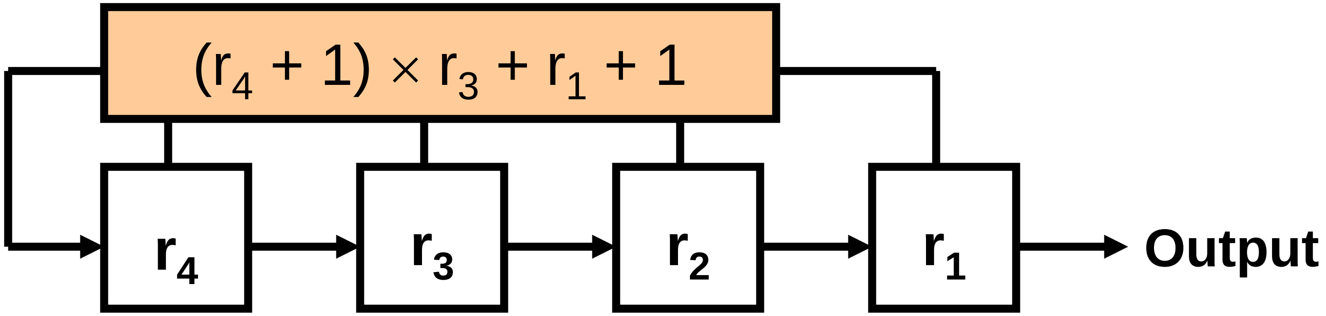
- LFSR are insecure
  - Known-plaintext attack reveals part of keystream
  - Output keystream reveals register state
  - Remaining keystream predictable from register state
- Idea: define feedback function as secret key



  - Still insecure: derive function with 2×m output bits

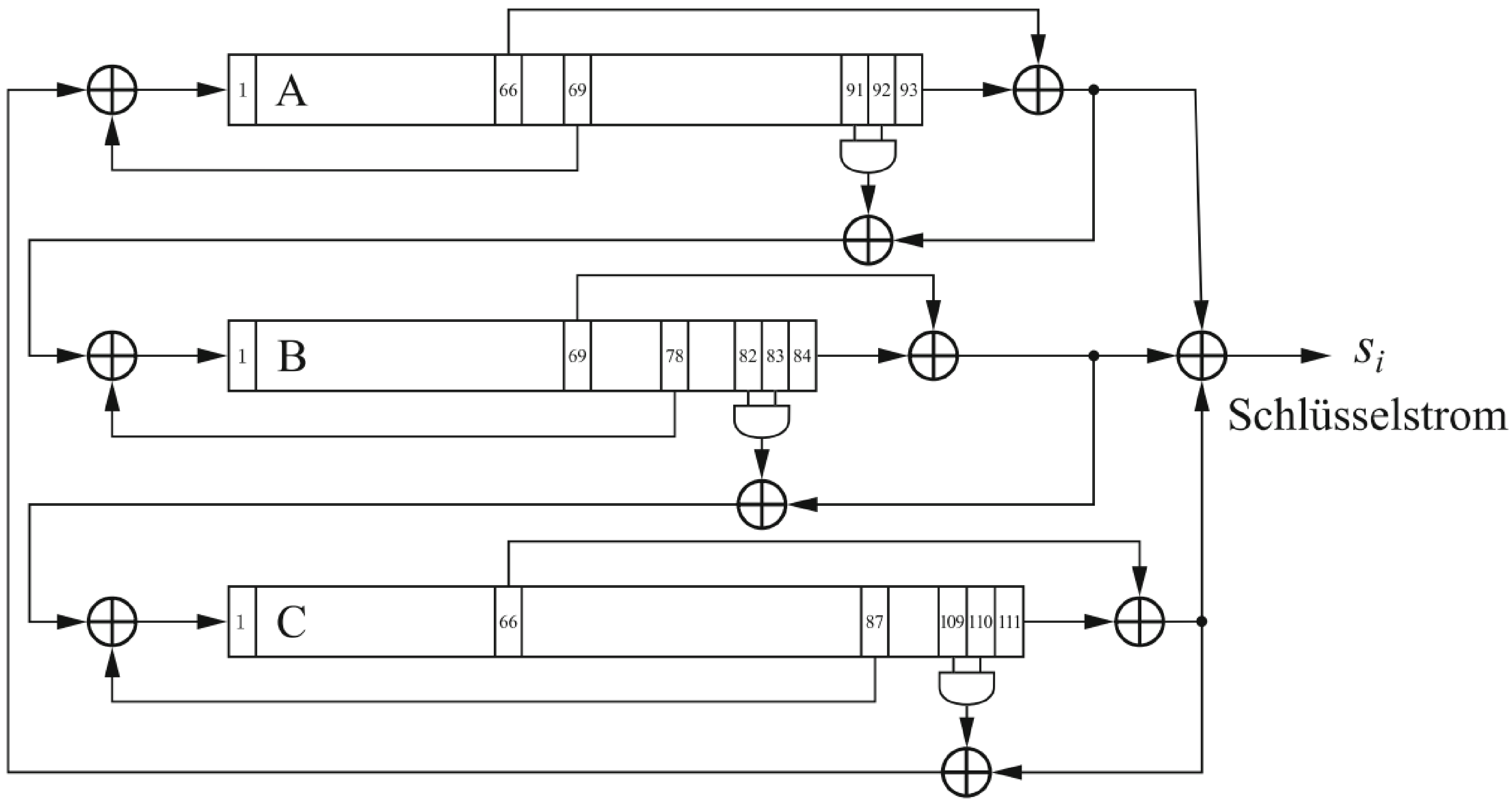
Security of LFSR (2)

- Problem: linear relationship between input and output allows mathematical analysis
- Idea: use non-linear feedback function



- NLFSR are more resistant against cryptanalysis
  - Though not automatically immune

Trivium



Source: Christof Paar, Jan Pelzl

Trivium (2)

- Modern stream cipher by De Cannière/Preneel
  - 80-bit key and 80-bit initialization vector
  - Generates keystream of 2^{64} bits
- Combines three shift registers of various length
  - Feedback and feedforward elements, 288 bit state
  - AND operation crucial for security (non-linear)
- Very fast hardware implementation
- Security: so far computationally secure
  - But: attacks on variants exist (thin security margin)

Conclusions

- Perfect secrecy: no statistical relationship between ciphertext and plaintext
- One-time pad is unconditionally secure
  - But unpractical due to long keys
- Most ciphers are only computationally secure
  - Secure enough for practical purposes
- Stream ciphers generate a pseudorandom keystream from a fixed-length secret key
  - Simple, fast, especially in hardware implementations
  - But security is not as well proven as for block ciphers