eurobits
European Competence
Center for IT Security

Ruhr
University
Bochum

Past and Future of Cryptographic Engineering

Hot Chips 2003

Christof Paar

Chair for Communication Security

Ruhr-University of Bochum

www.crypto.rub.de

Acknowledgement

This tutorial would not have been possible without the material and research of the following people:

- Michaël Neve and Jean-Jacques Quisquater
(Université catholique de Louvain, Belgium)
- Gerardo Orlando
(General Dynamics, MA)
- Jan Pelzl
(Ruhr-Universität Bochum, Germany)
- Thomas Wollinger
(Ruhr-Universität Bochum, Germany)

Special thanks also to John Wawrzynek and Nick Weaver from UC Berkeley for their many helpful comments.

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

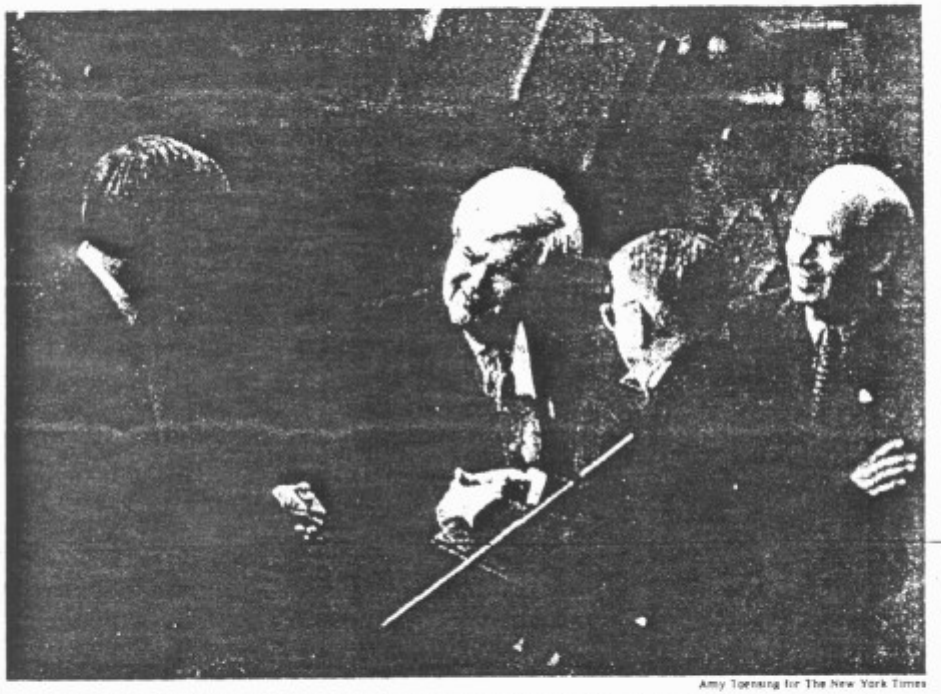
Contents

1. **Cryptography, IT Security, and Crypto Engineering**
   - **Past, present and future crypto applications**
   - What IT security can do for you
   - The cryptographic tool kit
   - What is crypto engineering anyway?
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

Do we really need security?

New York Times

NEW YORK, FRIDAY, JANUARY 10, 1997



Amy Toensing for The New York Times

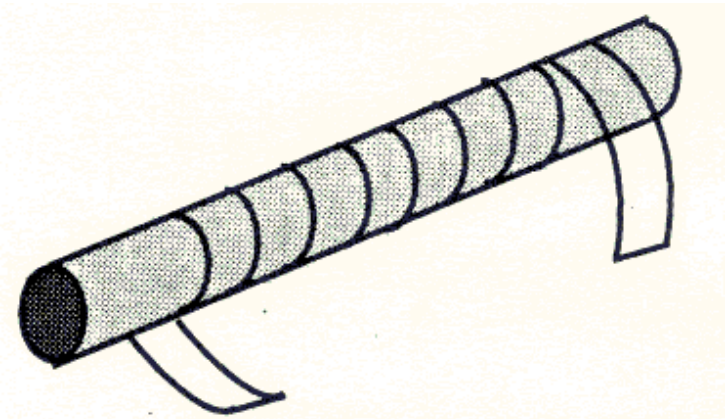
Newt Gingrich greeting House members yesterday before the final count of the Electoral College.

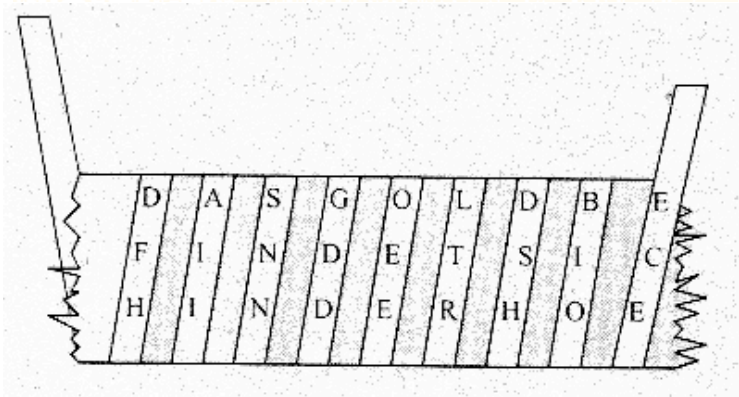
Gingrich Is Heard Urging Tactics in Ethics Case

By ADAM CLYMER

ture," since the subcommittee had

Cryptography, ca. 500 B.C





Skytale of Sparta

Cryptography, ca. 1940

German Enigma

(Polish, British & US break crucial for allied victory in WWII)

Cryptography, ca. 1990

Smart card for banking applications

Cryptography, ca. 2000

Electronic road toll

Cryptography:

- prevents cheating **by** drivers
- protects privacy **of** drivers

Cryptography, ca 2010

Brave new pervasive world

#2 Bridge sensors

#3 Cleaning robots

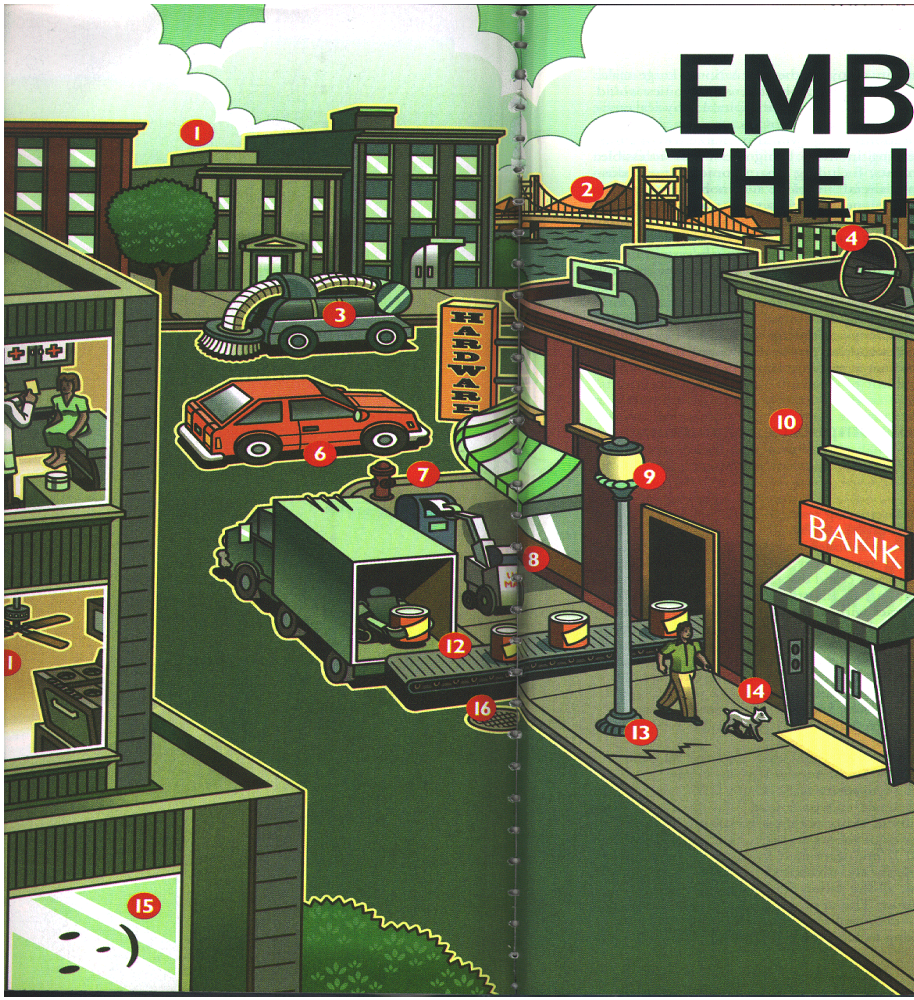
#6 Car with Internet access

#8 Networked robots

#9 Smart street lamps

#14 Pets with electronic sensors

#15 Smart windows



Contents

1. **Cryptography, IT Security, and Crypto Engineering**
   - Past, present and future crypto applications
   - **What IT security can do for you**
   - The cryptographic tool kit
   - What is crypto engineering anyway?
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

What IT security can do for you

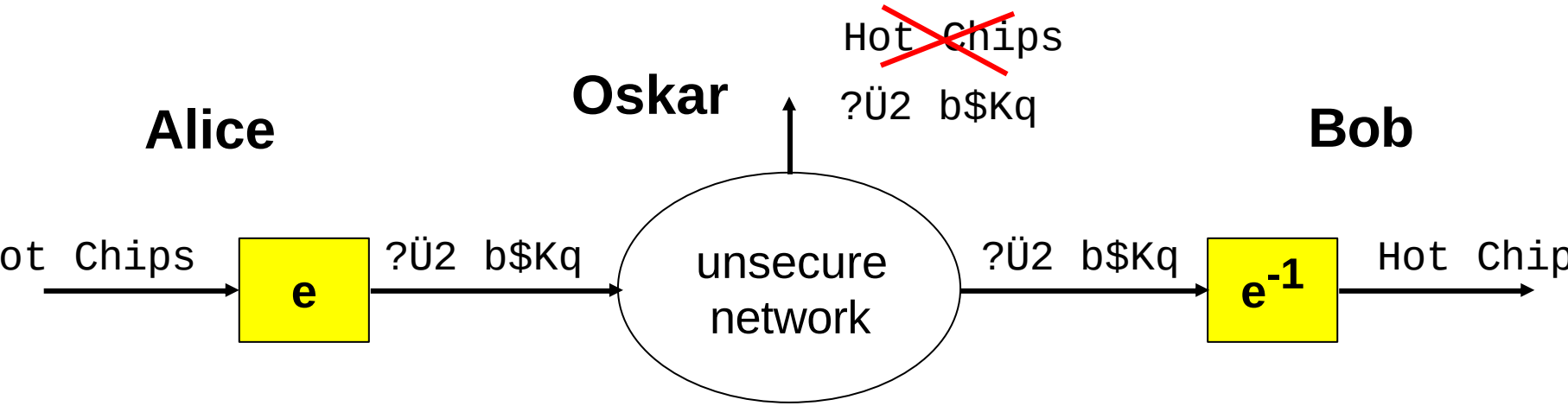
Classification into „security services“

1. **Confidentiality** (of messages)
2. **Integrity** (of messages)
3. **Authentication** (of messages)
4. **Identification** (of users or devices)
5. **Non-repudiation**

And more advanced:

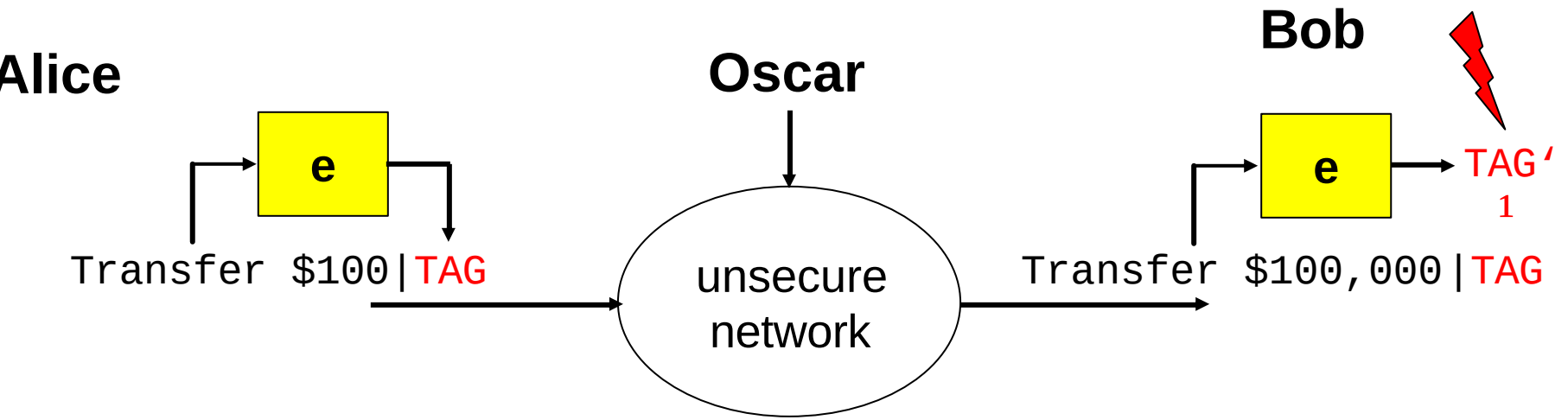
6. **Availability** (good luck)

Confidentiality



Encryption ensures **confidentiality** of messages

Integrity of Messages

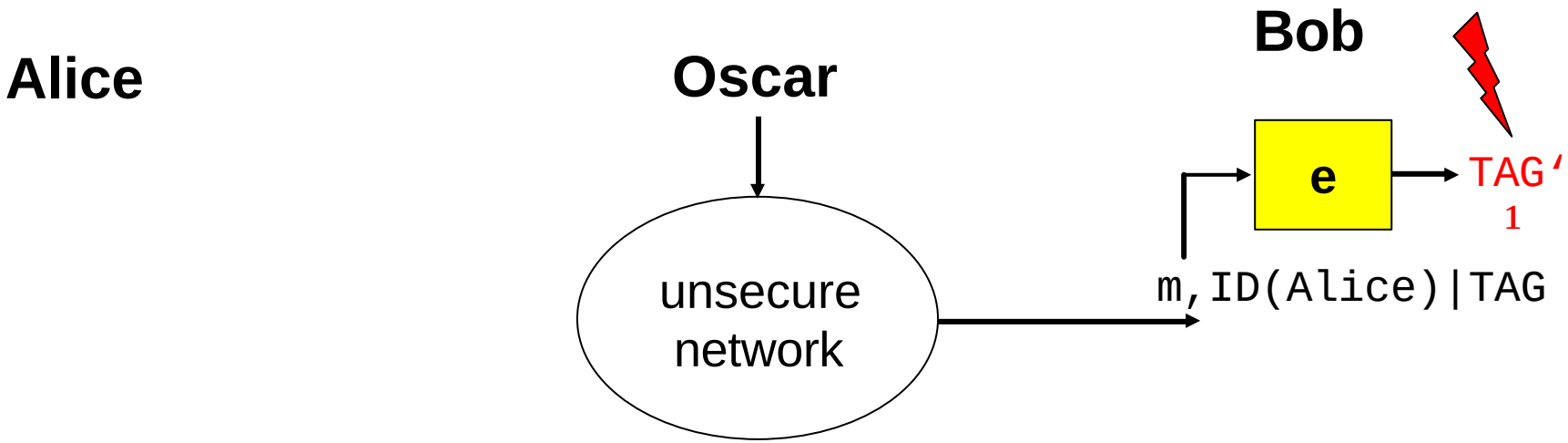


Cryptographic authentication tags:

1. Message Authentication Codes (MAC), or
2. digital signatures

ensure the **integrity** of messages

Sender Authentication

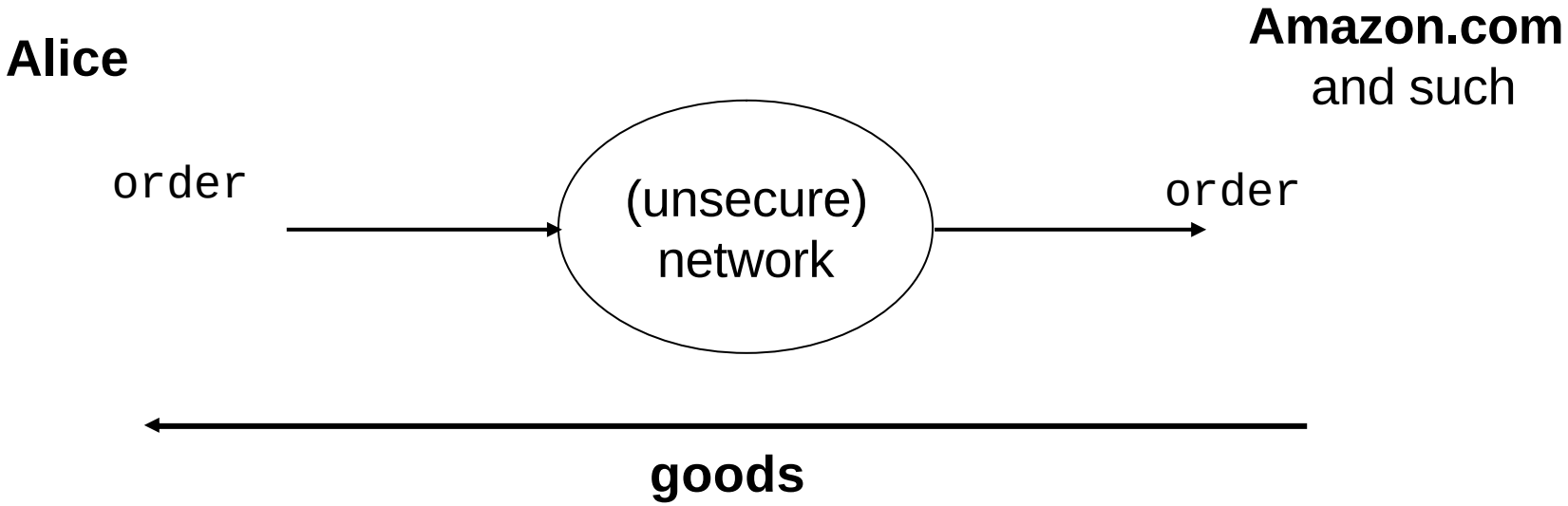


Cryptographic authentication tags:

1. Message Authentication Codes (MAC), or
2. digital signatures

authenticate the origin of a messages

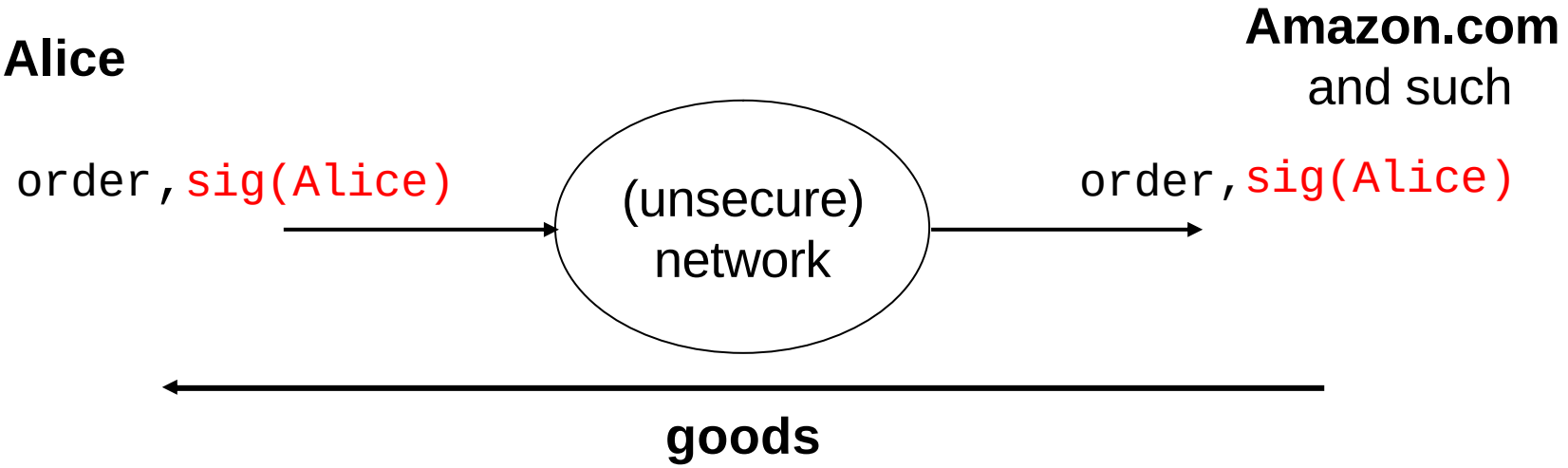
Non-repudiation: Why we need it



without non-repudiation:

1. Alice orders at favorite eCommerce vendor
2. stuff gets delivered
3. Alice doesn‘t feel like buying: „I never ordered this“
4. vendor can not **proof** it (big monetary issue if vendor = BMW.com)

Non-repudiation: How it works



with non-repudiation:

1. Alice orders at favorite eCommerce vendor
2. stuff gets delivered
3. Alice doesn‘t feel like buying: „I never ordered this“
4. vendor sues Alice: **proof** of order through Alice‘s signature

Non-repudiation is strong point of digital signatures

Identification (weak, w/o crypto)

Ex: identification of device (smart card)

Claimant (knows secret k)

k‘

Verifier (knows secret k)

1. Card sends secret („password“)

2. Reader compares k == k‘

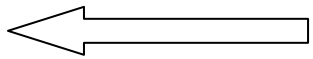
Problem: Eavesdropper gets k and clones smart card! (masquerading)

Identification (strong, with crypto)

Ex: identification of device (smart card)

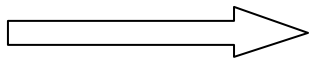
Claimant (knows k)

r



1. reader sends random challenge r

e_k (r) = y



2. Card returns encrypted challenge

3. Reader verifies

e_k (r) = y‘

y == y‘

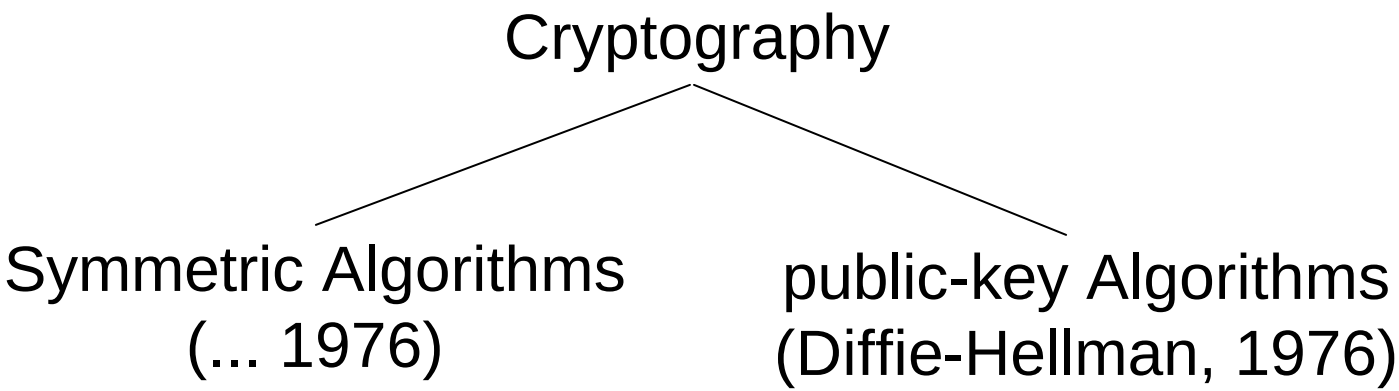
Verifier (knows k)

Contents

1. **Cryptography, IT Security, and Crypto Engineering**
   - Past, present and future crypto applications
   - What IT security can do for you
   - **The cryptographic tool kit**
   - What is crypto engineering anyway?
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

IT Security and Cryptography

1. IT Security ≠ Cryptography
2. **but**: cryptography is an important **tool** for achieving IT security



The cryptographic toolkit

Cryptographic Algorithms

Symmetric (... 1976)

- Stream cipher
- Block cipher

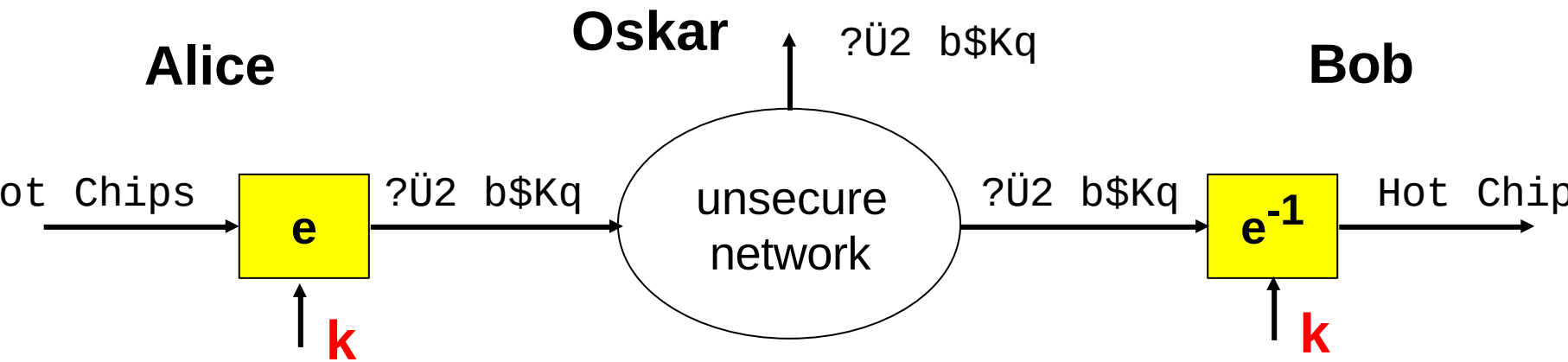
public-key (Diffie-Hellman, 1976)

- Integer Factorization (RSA...)
- Discrete Logarithm (D-H, DSA,...)
- Elliptic Curves (ECDH, ECDSA,...)

Hash fct.

Symmetric Cryptography

Ex: Enigma, DES (Data Encryption Standard)



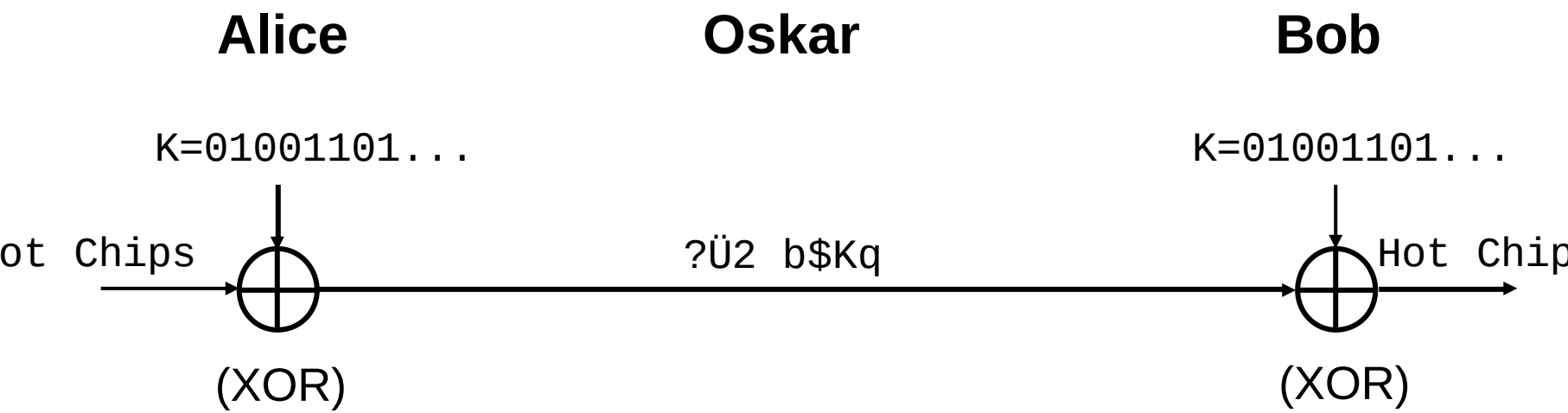
Classical Advantages

1. Confidenitality
2. Message integrity

Classical Shortcomings

1. Key distribution
2. Non-repudiation

Stream ciphers (1): One-time pad



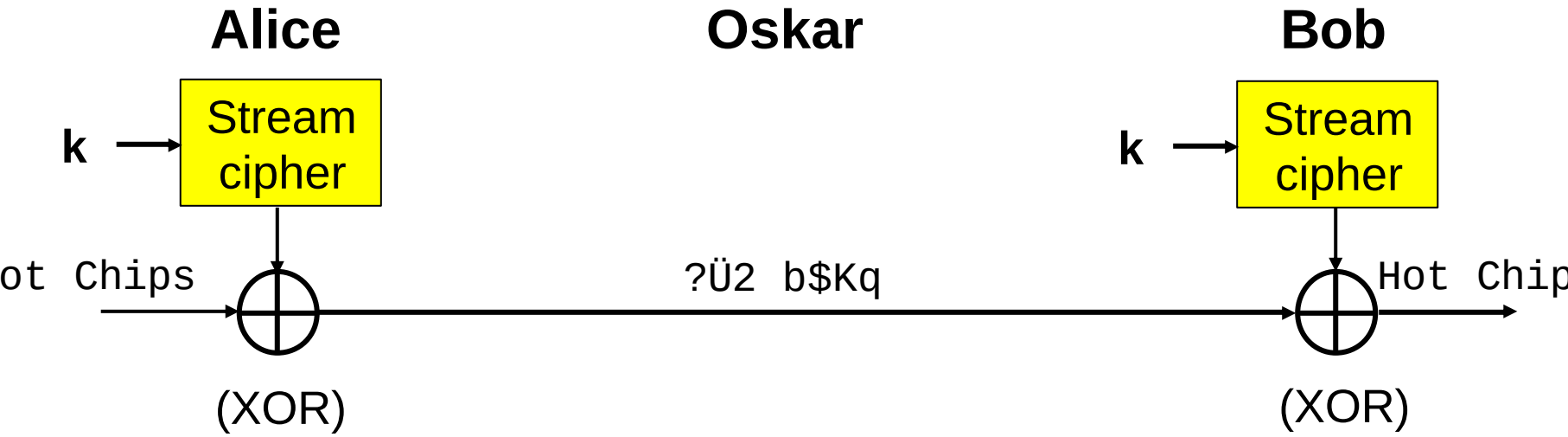
Requirement: Every key bit encrypts only one plaintext bit

(big) **Advantage**: Unbreakable („unconditional security“)

(big) **Disadvantage**: Highly impracticle (key length = message length)

Q: Can we emulate OTP with short key (practical)??

Stream ciphers (2): Practical schemes



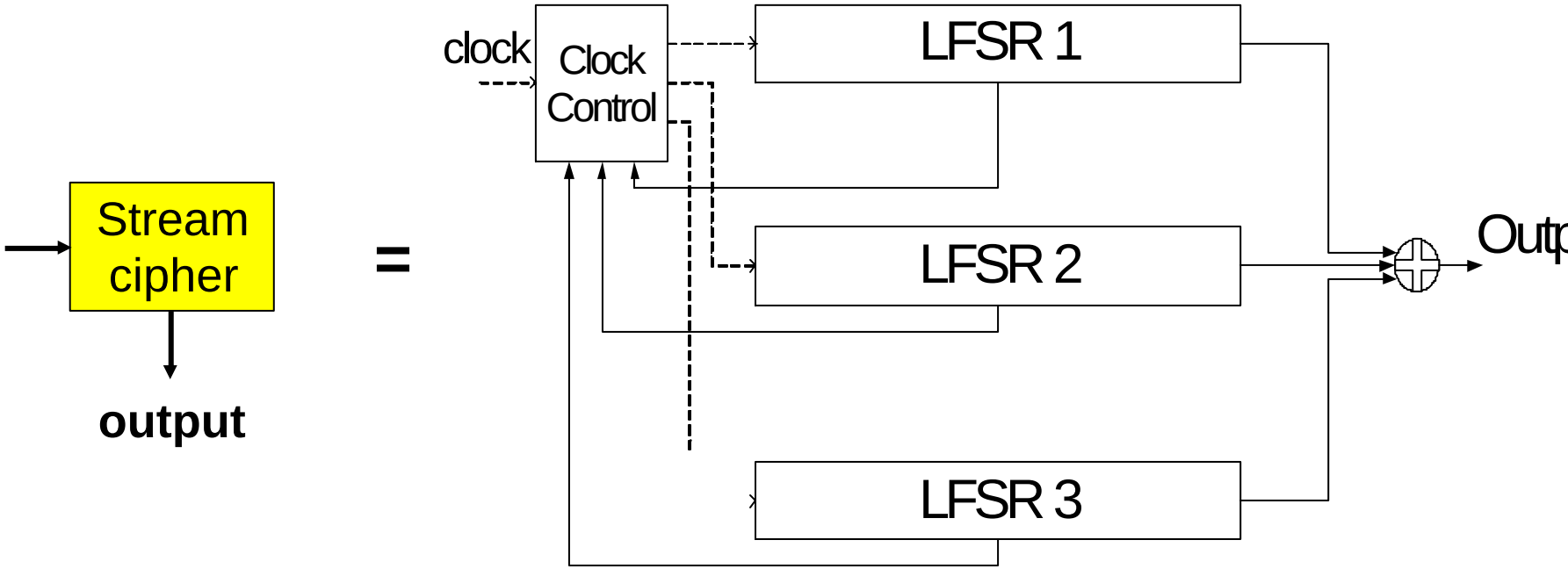
- Key k has **finite length** (e.g. 128 bits)
- Stream cipher output is
  - **pseudo-random &** very **long cycle length**
  - **cryptographically secure** (non-predicitable)
- **Not** unconditionally secure (just like all other practical alg.)

Stream ciphers (3): Comments

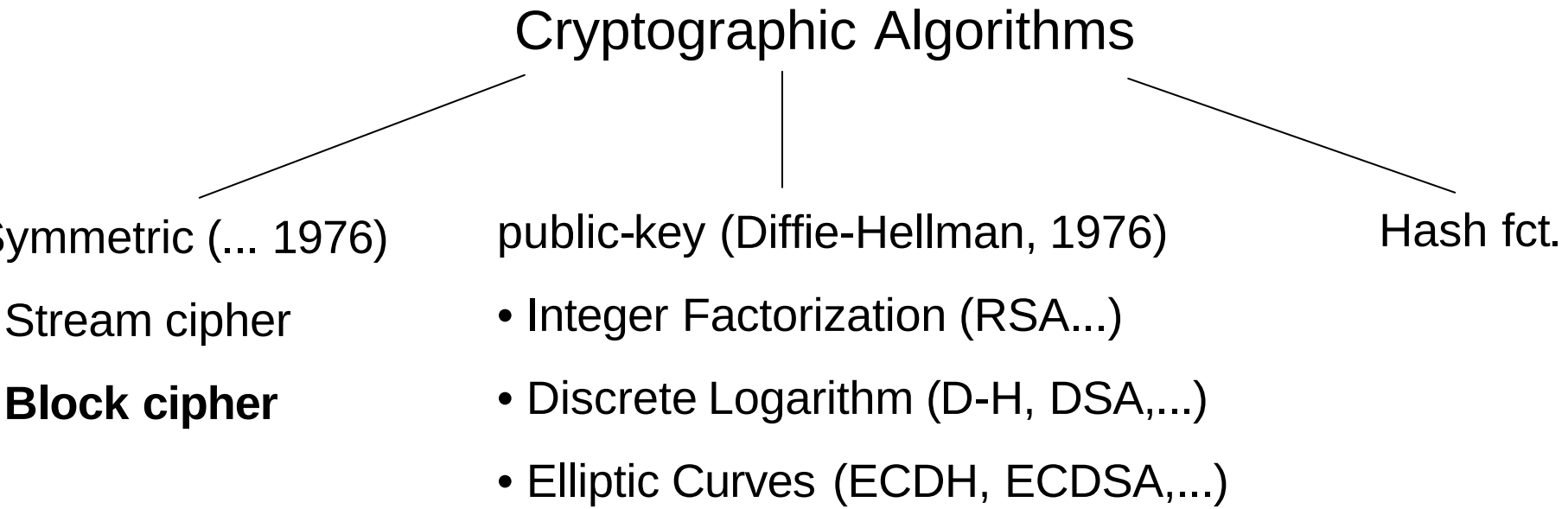
- Stream ciphers tend to be „cheaper“ (faster, smaller) than block ciphers
- Popular in mobile (and military) applications
- Not quite as well understood as block ciphers
  - Many proposed ciphers are unsecure
- Practical stream ciphers:
  - RC4,
  - Based on LFSR (linear feedback shift registers)
  - ...

Stream ciphers (4): An Example

- A5 stream cipher for GSM voice encryption
- based on LFSR (linear feedback shift registers)
- key k is the initial content of the LFSRs

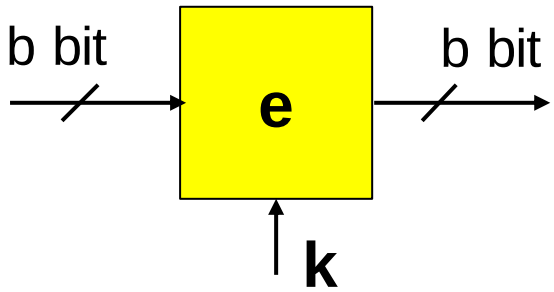


Recall: The cryptographic toolkit



Block ciphers (1): Basics

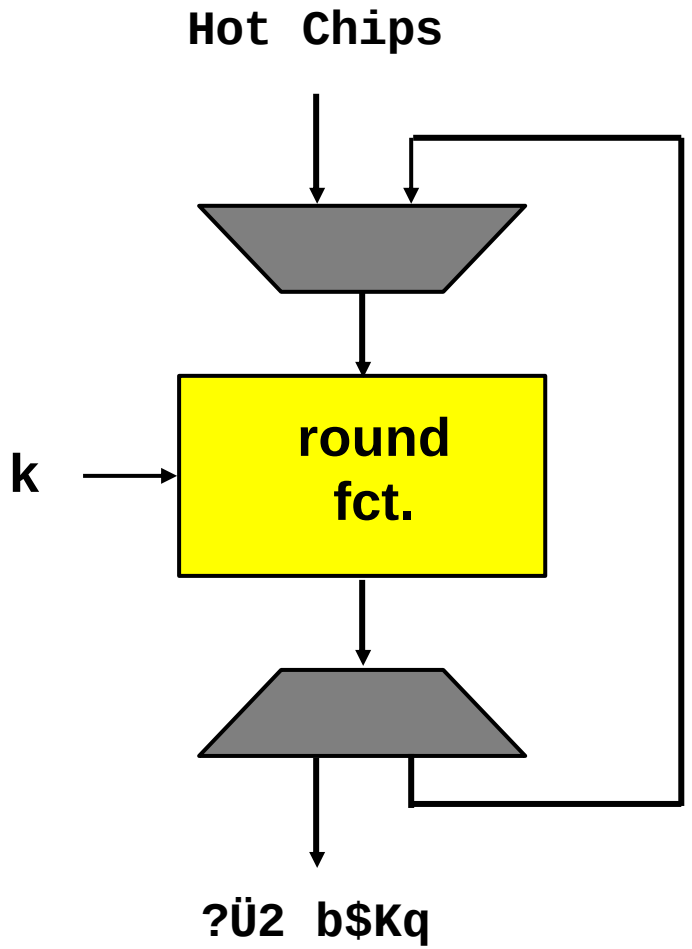
- encrypts b bit at a time
- typically b = 64 or b = 128
- key length: 56 ... 256 bit



- 100-200 or so proposed block ciphers
- 10 or so have commercial relevance
- 2 important standardized ones: DES and AES
- block ciphers are used in the majority of commercial applications

Block ciphers (2): Structure

- All practical block ciphers have an **iterative structure**
- The **components** in the round function **greatly vary** from cipher to cipher, e.g.,
  - table look-up
  - bit or word permutation
  - arithmetic
  - Boolean ops

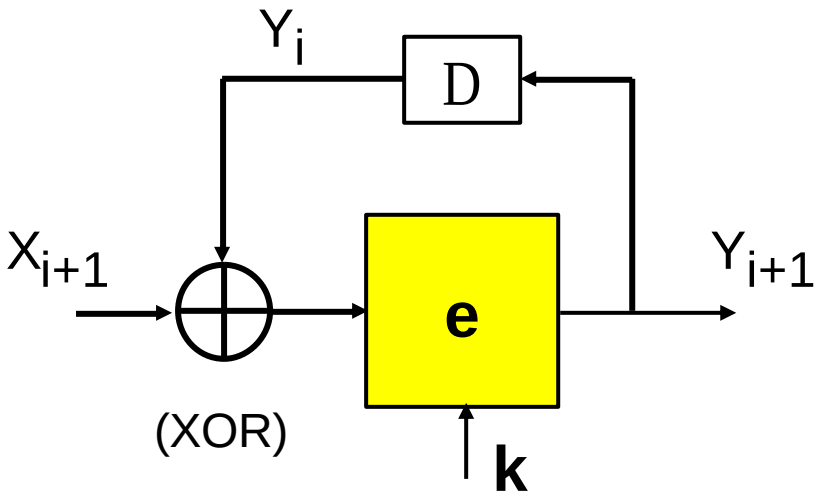


Block ciphers (3): Implementational properties

- Wide data path: 64-128 bits
- Generally well suited for high-speed HW
- Resource needs (area, power) depends on round function and varies greatly: large design space
- Software properties even more cipher-specific
- Parallelization can be a problem (see next slide)

Block ciphers (4): Parallelization

For security reasons often cipher chaining mode (CFB):



- x_{i+1} has to wait until x_i has been processed
- Prohibits parallelization!
- Modes for parallelization do exist too
- Good example for the constraints that crypto imposes

The cryptographic toolkit

Cryptographic Algorithms

Symmetric (... 1976)

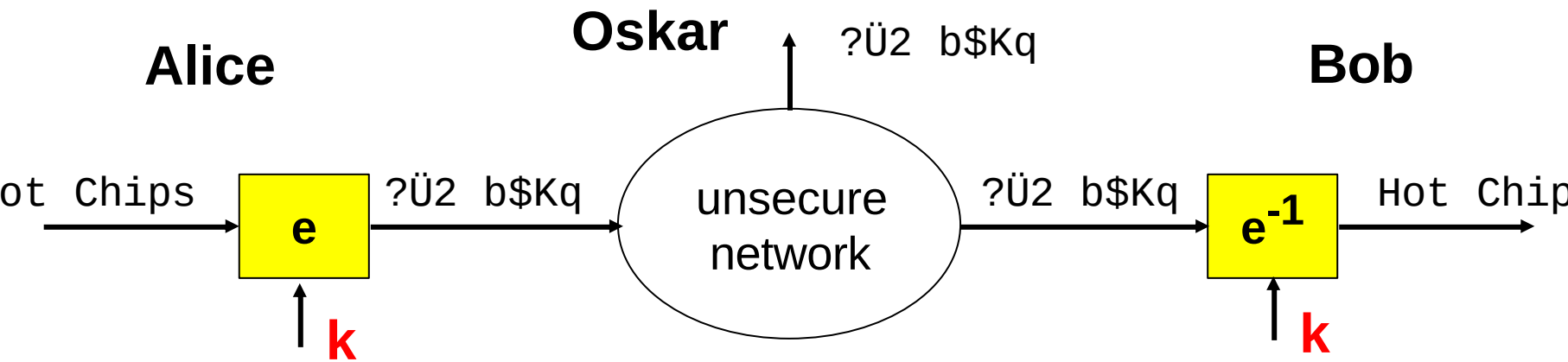
- Stream cipher
- Block cipher

public-key (Diffie-Hellman, 1976)

- Integer Factorization (RSA...)
- Discrete Logarithm (D-H, DSA,...)
- Elliptic Curves (ECDH, ECDSA,...)

Hash fct.

Recall: Shortcomings of symmetric cryptography



Classical Advantages

1. Confidentiality
2. Message integrity

Classical Shortcomings

1. **Key distribution**
2. **Non-repudiation**

Solution: public-key algorithms

What we can do with public-key (or asymmetric) cryptography

1. Key distribution over unsecure channel
2. Digital Signatures (non-repudiation!)
3. Encryption

Q: So, why do we still need symmetric ciphers?

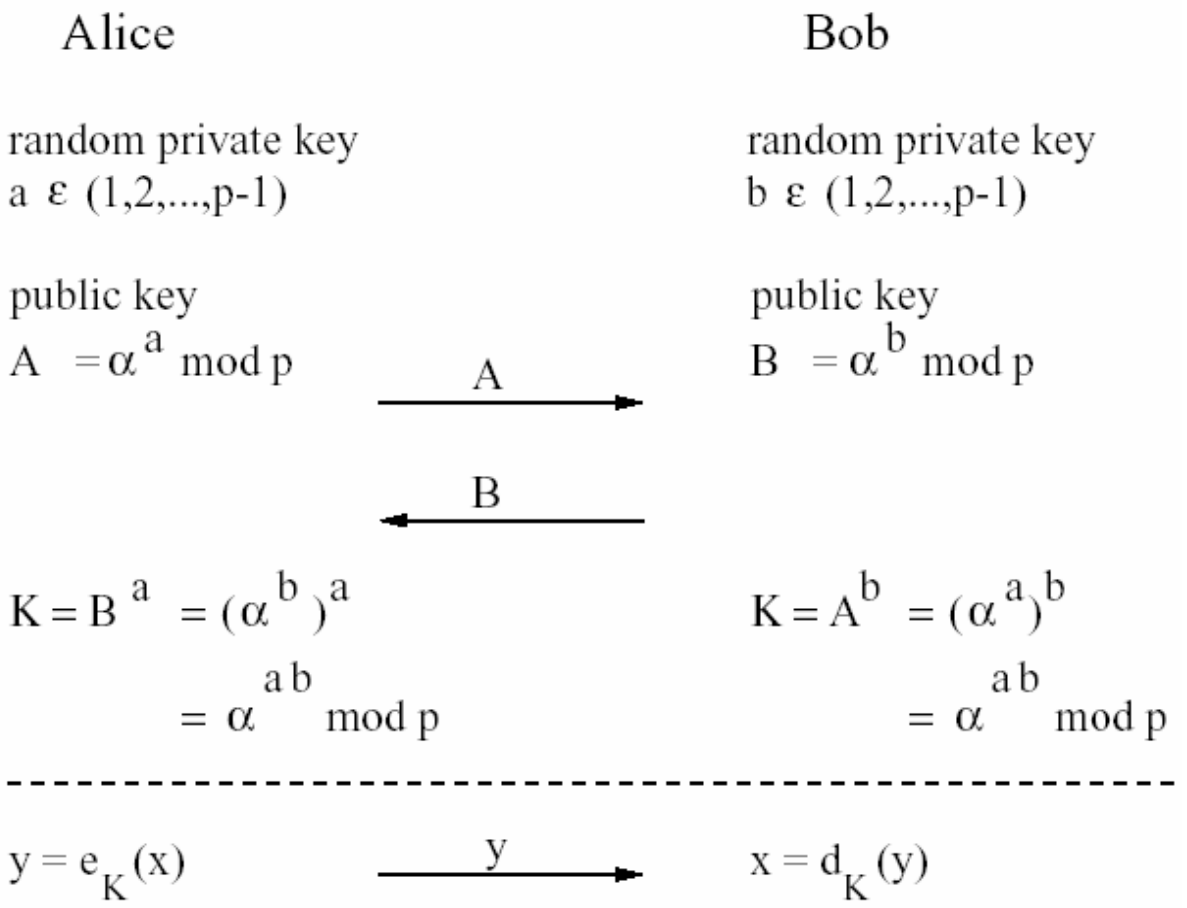
A: Public-key algorithms are awfully slow.

(Note: purely practical/engineering reason)

Public-key Ex: Diffie-Hellman Key Exchange

Given: large prime p
integer α (α generates subgroup of Z^*_p)

Idea: both parties posses 1 secret key and 1 public key



Practical public-key algorithms

3 families of algorithms of practical relevance:

Integer Factorization

Ex: RSA, Rabin, ...

Operands: 1024 – 2048 bits

Discrete Logarithm

Ex: Diffie-Hellman, DSA, ...

Operands: 1024 – 2048 bits

Elliptic Curves (ECC)

Ex: EC Diffie-Hellman, ECDSA, ...

Operands: 160 – 256 Bits

Observation: All asymm. algorithms require heavy computation

Bit length and security

Key lengths for *roughly* equivalent security level

Symmetric ciphers	80 bit
RSA etc.	1024 bit
Discrete log (Diffie-Hellman, …)	1024 bit
Elliptic Curves (EC Diffie-Hellman, …)	160 bit

ECC look promising, but ...

Arithmetic requirements of PK algorithms

Algorithm	*Operand length for multipl.*	*# multipl./ group op*	*# group ops/ crypto fct.*
RSA etc.	1024 bit	1	17 (verify) ≈ 1300 (sign)
Discrete log	1024 bit	1	≈ 200
Elliptic Curves	160 bit	≈ 10	≈ 200

- RSA is „best“ for signature verification
- ECC is „best“ for signature generation
- ECC has other advantages (bandwidth etc)
- RSA by far outnumbers ECC implementations in praxis!
- Are there faster PK algorithms? (big research issue)

How many key bits do I need?

symmetric	*ECC*	*RSA, DL*	*comment*
64 bit	128 bit	≈ 700 bit	only short term security (breakable with some effort)
80 bit	160 bit	≈ 1024 bit	medium term security (excl. government attacks)
128 bit	256 bit	≈ 2048-3072 bits	long term security (not assuming quantum computers)

- Exact complexity of RSA (factorization) and DL (index-calculus) attacks is hard to determine
- Quantum computer would probably be the death of ECC, RSA & DL
- Current assumption is that symmetric ciphers are robust against quantum computers.

Contents

1. **Cryptography, IT Security, and Crypto Engineering**
   - Past, present and future crypto applications
   - What IT security can do for you
   - The cryptographic tool kit
   - **What is crypto engineering anyway?**
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

What is crypto engineering anyway?

Definition:

The **efficient** and **secure** realization of cryptographic algorithms and protocols for applications in practice.

(+ the study of special-purpose cryptanalytical machines)

Where do we need crypto?

- PCs, laptops, workstations
- Network devices
- Smart cards (ATM cars, credit cards etc.)
- Hand helds (PDAs, ...)
- Cars (In-car entertainment, anti-theft etc.)
- Refrigerators, washing machines, ...
- Infrastructure sensors (buildings, windows etc.)
- …

⇒ Cryptography is becoming increasingly pervasive

⇒ Breadth of platforms makes *crypto engineering* increasingly crucial (and challenging)

Why don´t we leave it to the engineers anyway?

(or: Why crypto engineering really is important)

1. Many real-world attacks exploit implementation weaknesses
   - Ex. Side channel attack, fault injection attack
2. Often, new schemes only practical if eff. implemented
   - Ex. early days of elliptic curves & (until **very** recently) hyperelliptic curves
3. Interaction between implementation and alg.design
   - Ex. Arithmetic choice has major impact on implementation *and* security

⇒ **Crypto engineering is integral part of cryptography**

What's so difficult about crypto engineering?

1. Cultural differences: Cryptographers ↔ Engineers
2. Interdisciplinary knowledge required
   - Cryptography
   - Mathematics (number theory, abstract algebra) & Algorithms
   - Engineering stuff: Computer arch., micro electronic, …
3. Implementation methods often demanding
   - Ex. 2048 bit arithmetic (with low power)
   - Ex. Gbit/sec throughput without parallelization
4. Unusual rules: A „working" implementation is not enough, should also be secure

Trade-offs in crypto engineering

1. Performance
   - throughput
   - latency
2. Cost
   - Area (Hardware)
   - Code size (Software)
3. Power consumption
4. Security level (e.g., bit lengths)
5. Resistance against side channel attacks
6. Flexibility regarding parameter and crypto algorithm swap

(red = crypto specific)

Historical Perspective on Crypto Engineering

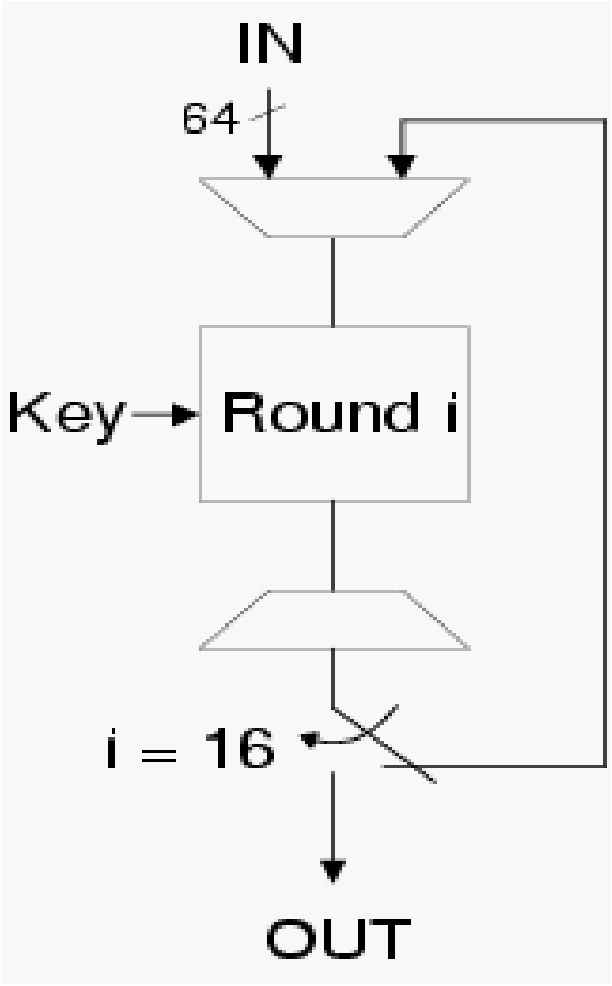
- For a long time (… mid-1990s) a niche area
- Research results scattered across literature
- Focus on high performance implementations, e.g.,
  - High radix RSA architectures
- Generally an after-thought, e.g.,
  - DES software is inefficient (perhaps intended?)
  - AES candidate hardware is area-intensive
- Field has become more mature over last 5 years

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. **Implementing Cryptographic Algorithms**
   - **Implementing symmetric algorithms**
   - Implementing public-key algorithms
   - Case study: A high performance elliptic curve engine
   - Case study: Emerging PK schemes on embedded processors
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

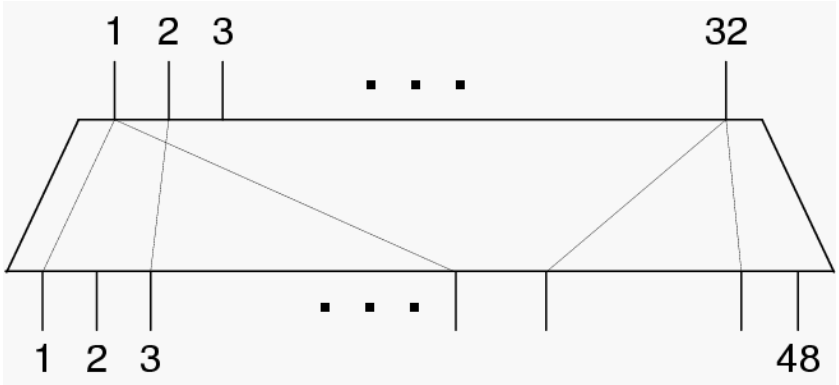
Symmetric case study I: DES

- designed in early/mid 1970s
- dominant cipher until late 1990s
- iterated block cipher
- implementation: strong focus on (1970s) hardware
- 56 bit key
- unsecure: brute-force attack
- but: 3DES very secure & popular

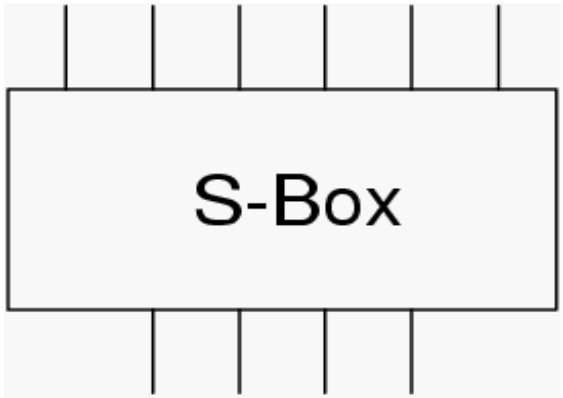


Main DES Round Components

- 32 bit permutation
- 32 $\rightarrow$ 48 bit expansion



- 6 x 4 bit substitution



Implementation of DES Components

Operation	Hardware	Software
32 bit permutation	fast + small	inefficient
48 bit expansion	fast + small	inefficient
S-box look-up	fast + small	inefficient

- One DES round needs ≈ 5k gates in HW
- DES very much designed with HW in mind (slow SW perhaps intended?)

Throughputs of typical implementations

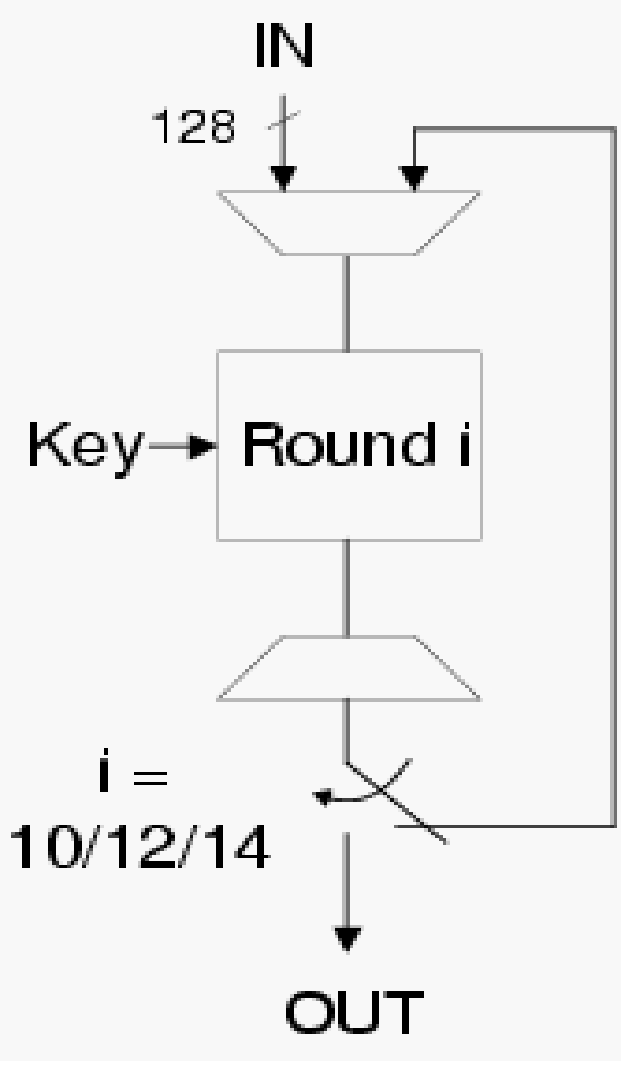
1. **HW, ASIC**
   > 10 Gbit/sec [Wilcox et al. 99]

2. **HW, FPGA**
   ≈ 10 Gbits/sec [Trimberger et al. 2000]

3. **SW, 300MHz DEC Alpha**
   ≈ 100 Mbit/sec [Biham 97]

Symmetric case study II: AES

- Advanced Encryption Standard
- „DES successor“
- Iterated block cipher
- Selected by NIST on Oct 2, 2000
- Focus on SW implementation
- HW implementation still quite reasonable



AES Component 1: ShiftRow

Data path = 128 bit = 16 byte = (a1,a2, ... , a16)

ShiftRow: reordering of bytes

0 shift	a1	a4	a8	a12
3 shifts →→→	a5	a9	a13	a1
2 shifts →→	a10	a14	a2	a6
1 shift →	a15	a3	a7	a11

AES Component 2: MixColumn

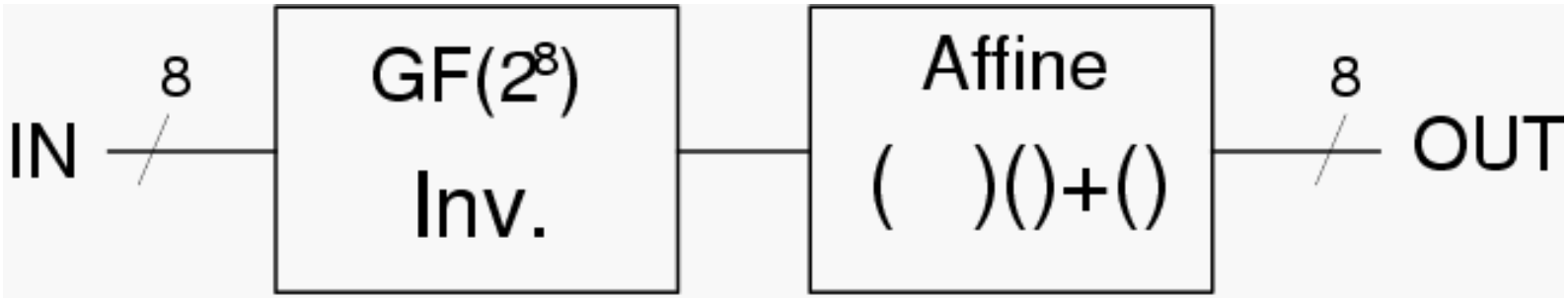
vector-matrix multiplication:

mapping 4 Bytes $\rightarrow$ 4 Bytes

$GF(2^8)^4 \rightarrow GF(2^8)^4$

$$\begin{pmatrix} c0 \\ c1 \\ c2 \\ c3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} x \begin{pmatrix} b0 \\ b1 \\ b2 \\ b3 \end{pmatrix}$$

AES Component 3: Substitution Box



S-Box: 2-steps

1. GF(2^8) Inversion
2. affine mapping (bit matrix mult. + vector add)

AES Implementation: Software

- Individual operations are mainly GF(2^8) ops on bytes
  $\Rightarrow$ straightforward implementation inefficient on
  32 bit µP

- Trick: Precompute look-up table which map entire round („T-Box“)
  - memory: 4 x 256 x 32bit = 4kByte
  - computations: 16 table accesses/round
  - throughput: 400Mbit/sec on 1.2 GHz Intel

AES in Hardware: ShiftRow

0 shift	a1	a4	a8	a12
3 shifts $\rightarrow\rightarrow\rightarrow$	a5	a9	a13	a1
2 shifts $\rightarrow\rightarrow$	a10	a14	a2	a6
1 shift $\rightarrow$	a15	a3	a7	a11

Permutation of 8 bit vectors $\Rightarrow$ **very cheap in HW**

AES in Hardware: MixColumn

$$\begin{pmatrix} c0 \\ c1 \\ c2 \\ c3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} x \begin{pmatrix} b0 \\ b1 \\ b2 \\ b3 \end{pmatrix}$$

per matrix: 8 constant multiplications over GF(2^8)

$\Rightarrow$ **cheap in HW** (152 XORs for entire MixColumn)

AES in Hardware: S-Box (1)

Recall:

- S-Box is mainly GF(2^8) inversion
- 16 S-Boxes per round
- S-box is bottleneck in hardware!

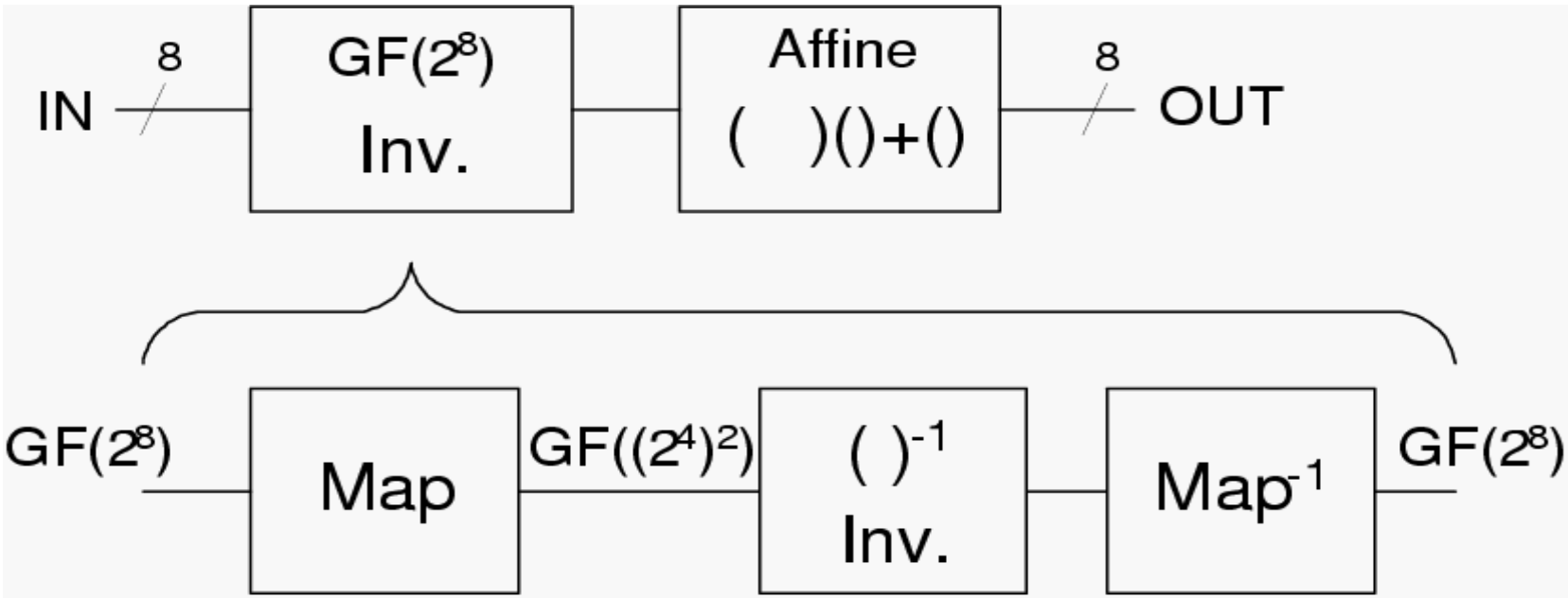
1. **Approach: 8 x 8 bit table look-up**
   pros: fast
   cons: ROM technology on chip needed

2. **Approach: Direct inversion with Boolean logic**
   pros: no ROM technology needed
   cons: lots of gates, bad critical path
   (no coincidence, due to high non-linearity of S-Box)

AES in Hardware: S-Box (2)

3. Approach: **Change of Galois field representation**

isomorphic mapping $GF(2^8) \leftrightarrow GF((2^4)^2)$



AES S-Box (3): Change of Galois field

- main advantage:
  reduction of GF(2^8) inversion to GF(2^4) inversion
- costs:
  - 1 inversion in GF(2^4)
  - 3 multiplications in GF(2^4)
  - Total complexity ≈ **150 gates**
    (vs. ≈ 600 gates for direct inversion)
- further reading: e.g., [Paar 95])

Throughputs of typical AES implementations

1. **HW, ASIC**
   2.3 Gbit/sec [Kuo et al.]

2. **HW, FPGA (efficient low cost implementation)**
   1.3 Gbit/sec [Weaver]

3. **SW, 1,3 GHz Intel**
   ≈ 580 Mbit/sec [Gladman]

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. **Implementing Cryptographic Algorithms**
   - Implementing symmetric algorithms
   - **Implementing public-key algorithms**
   - Case study: A high performance elliptic curve engine
   - Case study: Emerging PK schemes on embedded processors
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

Recall: Practical Public-Key (PK) Algorithms

3 families of algorithms of practical relevance:

Integer Factorization

Ex: RSA, Rabin, ...

Operands: 1024 – 2048 bits

Discrete Logarithm (DL)

Ex: Diffie-Hellman, DSA, ...

Operands: 1024 – 2048 bits

Elliptic Curves (EC)

Bsp.: EC Diffie-Hellman, ECDSA, ...

Operands: 160 – 256 Bits

Note: All public-key algorithms require heavy computation

Implementing Public-Key (PK) Algorithms: General Remarks

- Efficient implementation = high speed arithmetic
- Very wide operands: 160 ... 2048 bits
- Unusual arithmetic
  1. mod m (modular integer)
  2. GF(p^n) (Galois field)

Arithmetic requirements of PK algorithms

1. **RSA and Discrete Log** (Diffie-Hellman, DSA, ...)
   - simple algorithm: $y = x^a \bmod m$
   - wide operands: 1024 ... 2048 bits
   - mod m arithmetic (mostly)
2. **Elliptic curves** (ECDiffie-Hellman, ECDSA, ...)
   - complex algorithm (≈ 10-20 ops/group operation)
   - medium-size operands: 160 ... 256 bits
   - mod p or $GF(p^n)$ arithmetic

Let‘s look at arithmetic in finite fields, i.e., mod p and $GF(p^n)$

What are finite fields (or Galois fields)?

Engineering definition:

Galois fields are finite sets in which the four basis operations (+, -, $\times$, $\div$) hold.

Existence and notations:

Finite fields have always the form **GF(p^m),** where

p = prime,

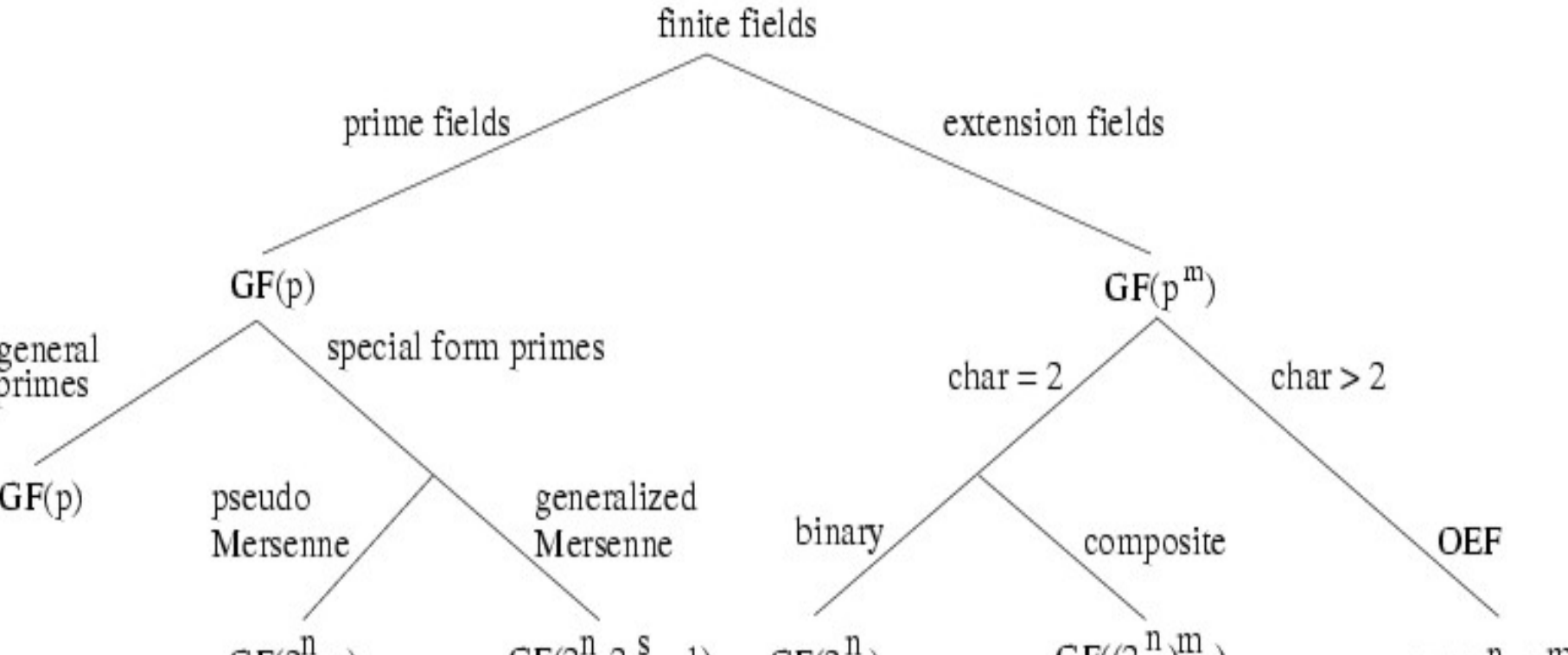
m = integer

Ex. GF(2^8) or GF(31)

In crypto practice mainly:

1. „prime fields“ GF(p)
2. „binary fields“ GF(2^m)

Finite Fields Proposed for Use in PK Schemes



Prime fields GF(p)

- conceptionally simple: modulo p computations
- wide-spread use in practice
- used for discrete log and ECC
- arithmetic:
  - addition is cheap
  - inversion is costly but can often be avoided
  - „remaining“ problem: efficient multiplication

Multipl. in prime fields GF(p): Software

Let A, B in GF(p), p = n-bit prime, word size = w

Ex: n = 1024 bit, w =16 bit

1. **C = A x B** (multi-precision multiplication)
   complexity: $(n/w)^2$ = 4096 int. mult.
2. **C mod p** (modular reduction)
   complexity: $(n/w)^2$ = 4096 int. mult.

Total Complexity: $\approx 2\,(n/w)^2 = 8192$ integer mult.

- Single GF(p) mult. is very costly
- Step 1 & step 2 are often interleaved

Multipl. in prime fields GF(p): Software

- Long-number multiplication: Complexity of $(n/w)^2$ int mult for can be reduced (Karatsuba algorithm)
- Several techniques available for division-free modular reduction:
  - Montgomery reduction
  - Barrett reduction
  - Sedlack reduction
- Best studied approach: Montgomery reduction
- Further reading: [Koc et al.]

Multipl. in prime fields GF(p): Hardware

Ex: n = 1024 bit, r = radix

Idea: Compute n/r inner products in parallel w/o division!

Montgomery multiplication (best studied architecture)

Input: B, A = å a_i
Output: A B mod N /* N is auxiliary modulus
R_0 ¬ 0
  For i = 0 to (n/r+2) do /* main loop
  q_i ¬ R_i(0)
  R_{i+1}=(R_i+ a_i B + q_i N)/2 /* parallel digit multiplier

time compl. = n/r clocks

area compl. = O(r n) gates

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. **Implementing Cryptographic Algorithms**
   - Implementing symmetric algorithms
   - Implementing public-key algorithms
   - **Case study: A high performance elliptic curve engine**
   - Case study: Emerging PK schemes on embedded processors
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

Recall: Practical public-key algorithms

Integer Factorization

Ex: RSA, Rabin, ...

Operands: 1024 – 2048 bits

Discrete Logarithm

Ex: Diffie-Hellman, DSA, ...

Operands: 1024 – 2048 bits

Elliptic Curves (EC)

Bsp.: EC Diffie-Hellman, ECDSA, ...

Operands: 160 – 256 Bits

⇒ Elliptic curves look promising for high performance applications

Casestudy: An Elliptic Curve Processor on Reconfigurable HW

see also [Orlando/Paar 2000]

Design Goals

1. (Very) high performance
2. Flexible security levels
3. Performance/cost (= speed/area) scalability
4. Moderate costs for medium-volume applications

Why reconfigurable platform for elliptic curve processor?

Idea: Exploit capabilities of modern commercial FPGAs

- **High performance**: Speed-optimized architecture for every parameter set
- **Flexible security levels**: Compile architecture for every bit length (field order) & field polynomial
- **Performance/cost scalability**: Choose slow + small or fast + large arithmetic units
- **Moderate costs**: Development using HDL and unit costs in the $100 range

Core function: EC Point multiplication over GF(2^m)

"Point multiplication":

$$k\,P := P + P + \ldots + P \text{ (k times)}$$

- P is a point on the elliptic curve: P=(x,y)
- k is integer, with $k \approx 2^m$
  - in practice m = 160 … 256
- core operation in elliptic curve cryptosystems, e.g.
  - digital signature
  - key exchange

Arithmetic requirements for point multiplication

underlying field: GF(2^m), m = 160 … 256

GF(2^m) ops	**Montgomery** (proj. coord.)	**IEEE P1363** (proj. coord.)
# Square	5m	7m (ave.)
# Multiply	6m	10.5 (ave.)
# Inverse	1	1

- squaring can be almost free
- main cost (m ≈ 160): ≈ 1000 mult's with 160 bit ops

How costly is squaring in GF(2^m)?

m = 160 … 256

field supported	# clocks	# gates
fixed m	1	≤ m (trinom.) ≤ 4m (pentan.)
any m	m/2	O (m)

- flexible architectures (i.e. arbitrary m) are slow
- reconfig. HW allows fast arch. for every bit length m

Digit-serial multiplier for GF(2^m)

- D = digit size, e.g. D = 1…16
- mult. time: $K_D = \lceil m/D \rceil$ clocks
- multiplier complexity
  - O(D m) gates
  - O(m) flip-flops
- incorporates adder: A+B := A*1 + B*1
  - Addition time: 1-2 clocks
- crucial for over-all performance!

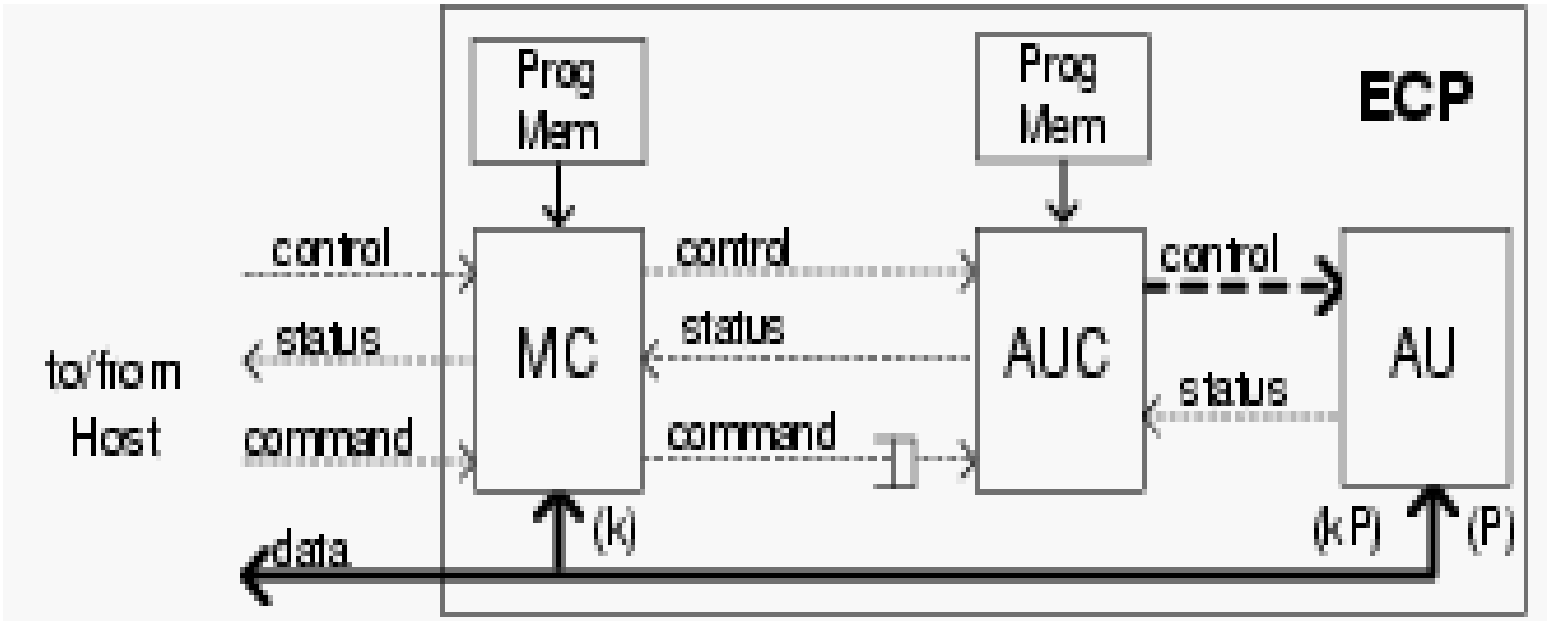
Complexity of GF(2^m) operations

m = 160 … 256

GF(2^m) ops	# clocks
addition	1
square	1
multiply	$\lceil m/D \rceil$

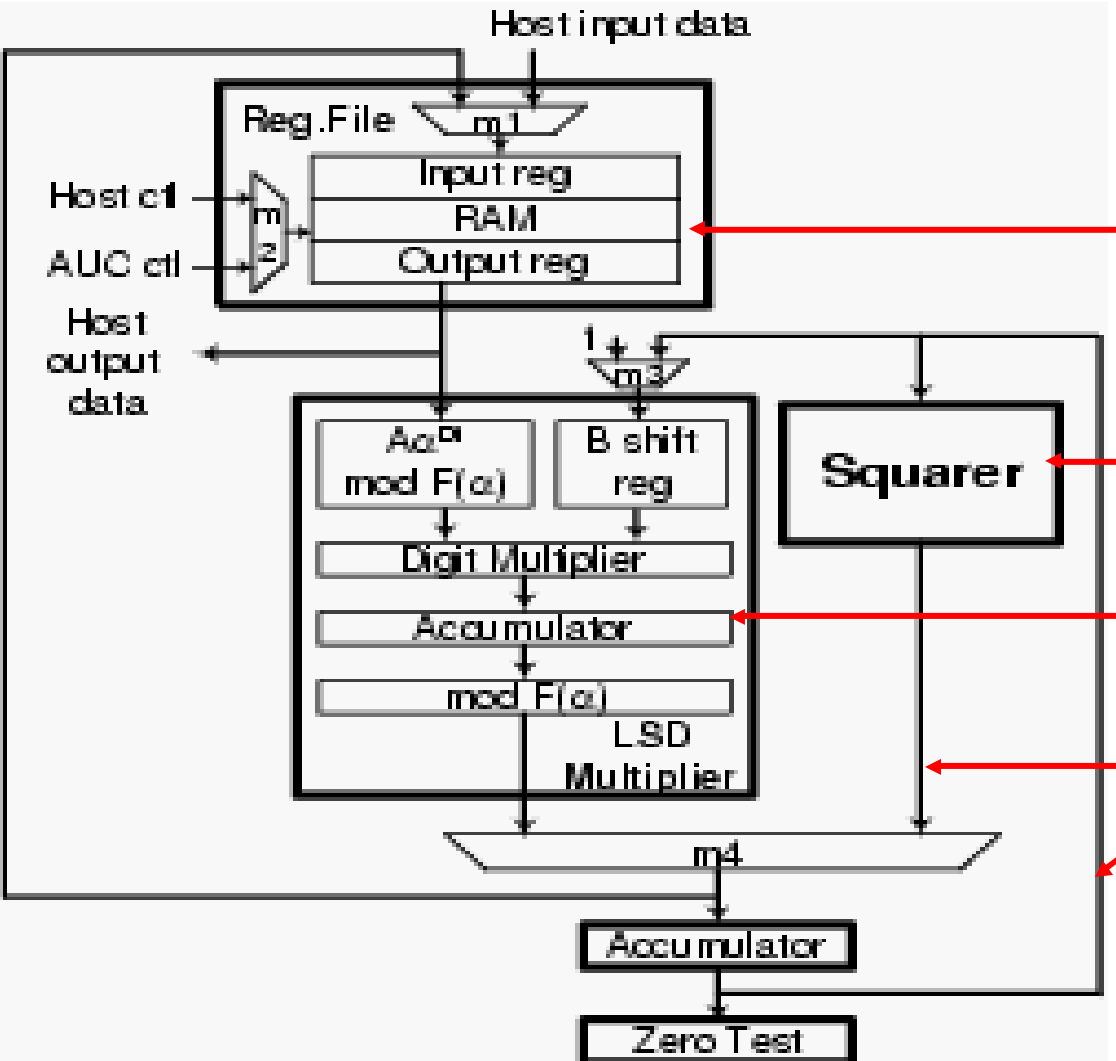
- D is the multiplier's digit size, e.g. D=1…16
- field multiplication dominates over-all complexity

Block Diagram EC Processor



- MC --- Main Ctrl: Point multiplication alg.
- AUC --- Arithmetic Unit Ctrl: Group operation and arithmetic control
- AU --- Arithmetic Unit: add, square, multiply in GF(2^m)

Arithmetic Unit



- crucial for over-all performance!
- large register file allows precomp. algorithms
- bit parallel squarer (fast!)
- scalable, digit-serial multiplier
- m bit buses

Performance of EC Engine Prototypes

Arithmetic in GF(2^{167}) (security equiv. to RSA 1024)

Digit Size (multiplier)	**Clock** (MHz)	**Point Mult.** (msec)	**Speed** (norm.)
4	86	**0.55**	1
8	75	**0.35**	1.8
16	77	**0.21**	3.0

- Xilinx Virtex XCV400 FPGA
- almost linear speed-up with digit size increase
- 10 x faster than best published result at design time

Complexity of EC Engine Prototypes

Arithmetic in GF(2^m), m = 167

Digit Size (multiplier)	**# RAM blocks**	**# FF**	**# LUT**	**# LUT (norm.)**
4	10	**1745**	**1627** (9.7 m)	1
8	10	**1753**	**2136** (12.8 m)	1.3
16	10	**1769**	**3002** (17.8 m)	1.8

- #RAM, #FF independent of multiplier speed
- logic (# LUT) scales sub-linearly with digit size
- best time-area product with fast multiplier (i.e. D large)

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. **Implementing Cryptographic Algorithms**
   - Implementing symmetric algorithms
   - Implementing public-key algorithms
   - Case study: A high performance elliptic curve engine
   - **Case study: Emerging PK schemes on embedded processors**
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

History of some public-key schemes with practical relevance

1976 Diffie-Hellman

1977 RSA

1985 Elliptic curves (practical relevance since mid 1990s)

1988 Hyperelliptic curves (practical relevance since **2002**)

Recall: Arithmetic requirements of PK algorithms

Algorithm	*Operand length for multipl.*	*# multipl./ group op*	*# group ops/ crypto fct.*
RSA etc.	1024 bit	1	17 (verify) ≈ 1300 (sign)
Discrete log	1024 bit	1	≈ 200
Elliptic Curves	160 bit	≈ 16	≈ 200

Q: Are there other (faster) PK algorithms, esp. for embedded applications?

A: Yes, **hyperelliptic curves cryptosystems (HECC)** look promising, but many open issues...

Why use hyperelliptic curve cryptosystems (HECC)?

- Really cool name
- Shorter operand length than ECC (and certainly RSA & DL) looks promising for constrained processors
- Hopefully as secure as ECC
- but open questions
  - Is the over-all performance really better?
  - Are HECC secure??
  - What are hyperelliptic curves???

What are hyperelliptic curve cryptosystems?

1. Generalization of elliptic curves
2. Come in different genii g
   - g = 1 elliptic curves
   - g = 2,3,... hyperelliptic curves
3. **group size = (field size)^{g}**
4. Ex: group size = 2^{160} (commercial security level)
   - ECC (g=1): arithmetic bit length = 160 bit
   - HECC (g=2): arithmetic bit length = 80 bit
   - HECC (g=4): arithmetic bit length = 40 bit

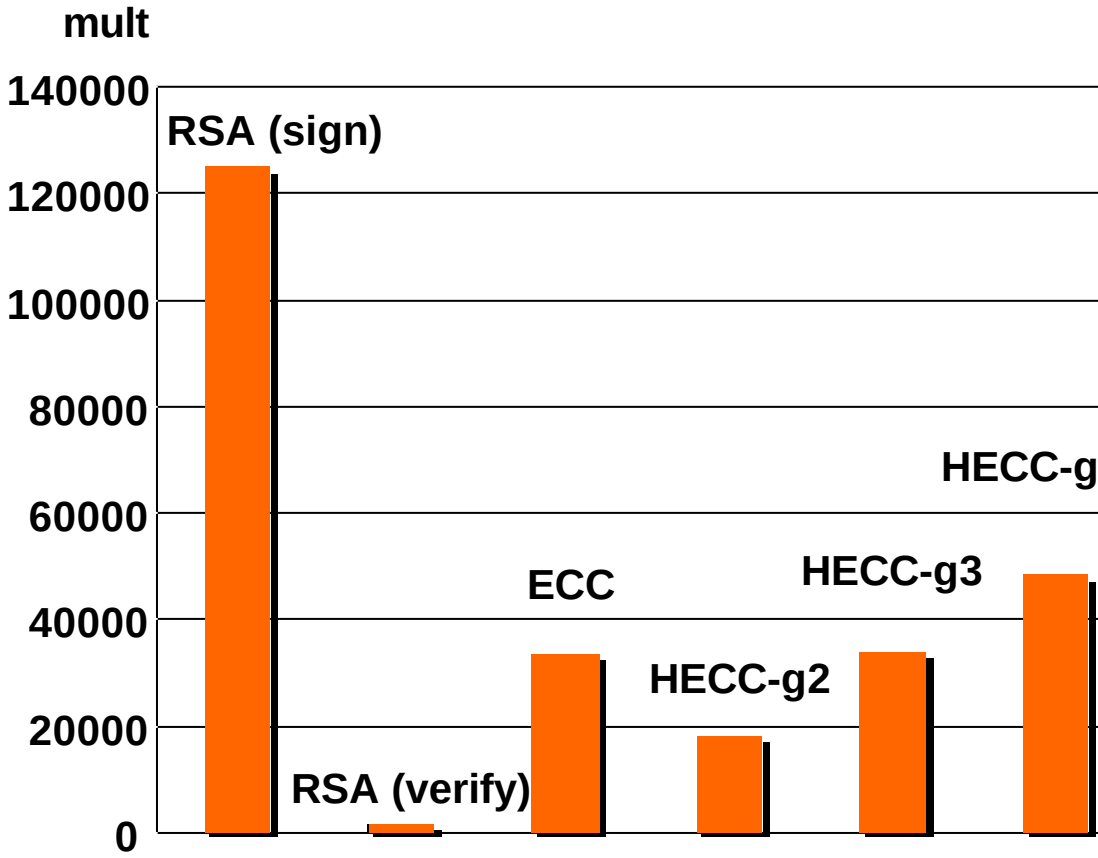
HECC: So, where is the catch?

Trade-off: „group operation“ becomes much more complex as genus increases

genus	*arithm. size (example)*	*# mult.+ #sq/ group ops*	*# invers./ group ops*
1 (ECC)	160 bit	16	0
2 (HECC)	80 bit	25	1
3 (HECC)	53 bit	76	1
4 (HECC)	40 bit	164	2

Theoretical Complexity Comparison of RSA vs. ECC vs. HECC

- Metric: # integer mult.
- crypto systems at equal security level
- ECC, HECC over GF(p)
- **Q**: Influence of processor word size?



An interesting design option: Genus-4 HECC with group size 2^{128}?

Allows 128 / 4 = **32 bit field arithmetic!**

- 1 field element = 1 processor word
- no carries, easy data types
- great for (embedded) 32 processors

(big) **but**: Are HEC with group size 2^{128} secure?

Security of 128-bit HECC

Hard data on attacks (outside government agencies)

1. DES (56-bit) Challenge III – 22 hours (1999)
2. ECCp-109 challenge – 1.5 years, 10.000 computers (2002)

? HECC with a group order of 2^{128}

- are 724 times harder to break than ECCp-109
- far more secure than DES or RSA 512 (still widely used)
- **sufficient for many embedded applications** (short-medium term security)

Light weight security on the ARM7 @ 80MHz

HECC Genus	*Group order*	*Arithm. size*	*Crypto ops (divisor multipl.)*
2	2^{128}	64 bit	**71.54ms**
3	2^{129}	43 bit	**47.13ms**
4	2^{128}	32 bit	**49.07ms**

Note: g = 4 curves are very competitive despite poor theoretical complexity

Conclusions HECC

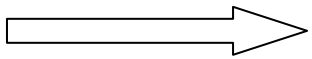
- HECC show good performance on real-world embedded platforms
- g = 4 curves are an interesting option for light-weight crypto
- further research on group operation formulae
  - reduced complexity (= faster)
  - parallelization
  - how realistic are attacks against HECC with g=4, 5, …
- further reading: [Pelzl et al. 2003a, 2003b]

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. **Selected Aspects of Crypto Engineering**
   - **Side channel attacks**
   - Reconfigurable hardware and cryptography
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

Another become-rich-quick scheme

attacker



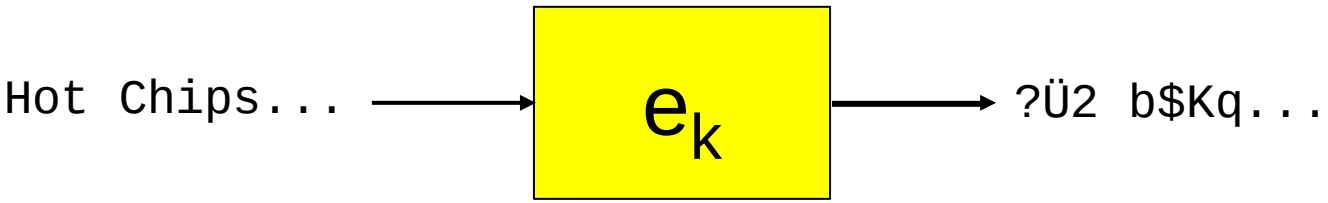
k = ?

Knowledge of secret key k on smart card allows:

- Cloning of card (pay TV, ...)
- Manipulation of card (reloading of payment cards, ...)
- ...

Side channel vs. algorithmic attacks

Classical attack scenario (Enigma break etc.):



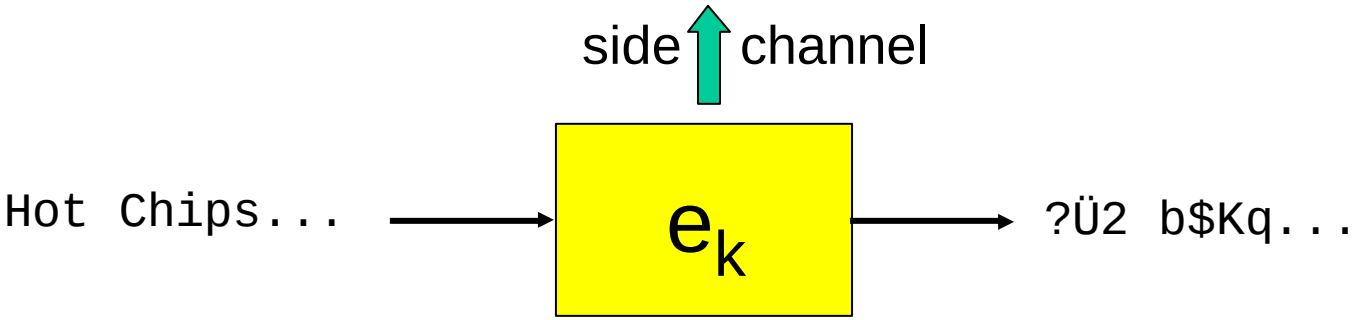
Attacker knows

1. ciphertext
2. some plaintext

and tries to deduct key k

Side channel vs. algorithmic attacks

In the real world, the *implementation* might leak information



Attacker knows

1. ciphertext
2. some plaintext
3. *side channel information*

and tries to deduct key k

Which side channels leak information?

- Power signal
  - SPA: simple power analysis
  - DPA: differential power analysis
- Timing behavior of algorithms
- EM: electromagnetic radiation
- Temperature
- … (probably several other)

red = successfully exploited for attacks

Very brief history of side channel attacks

1992 - TNO (Holland) discovers relationship between smart card program code and power

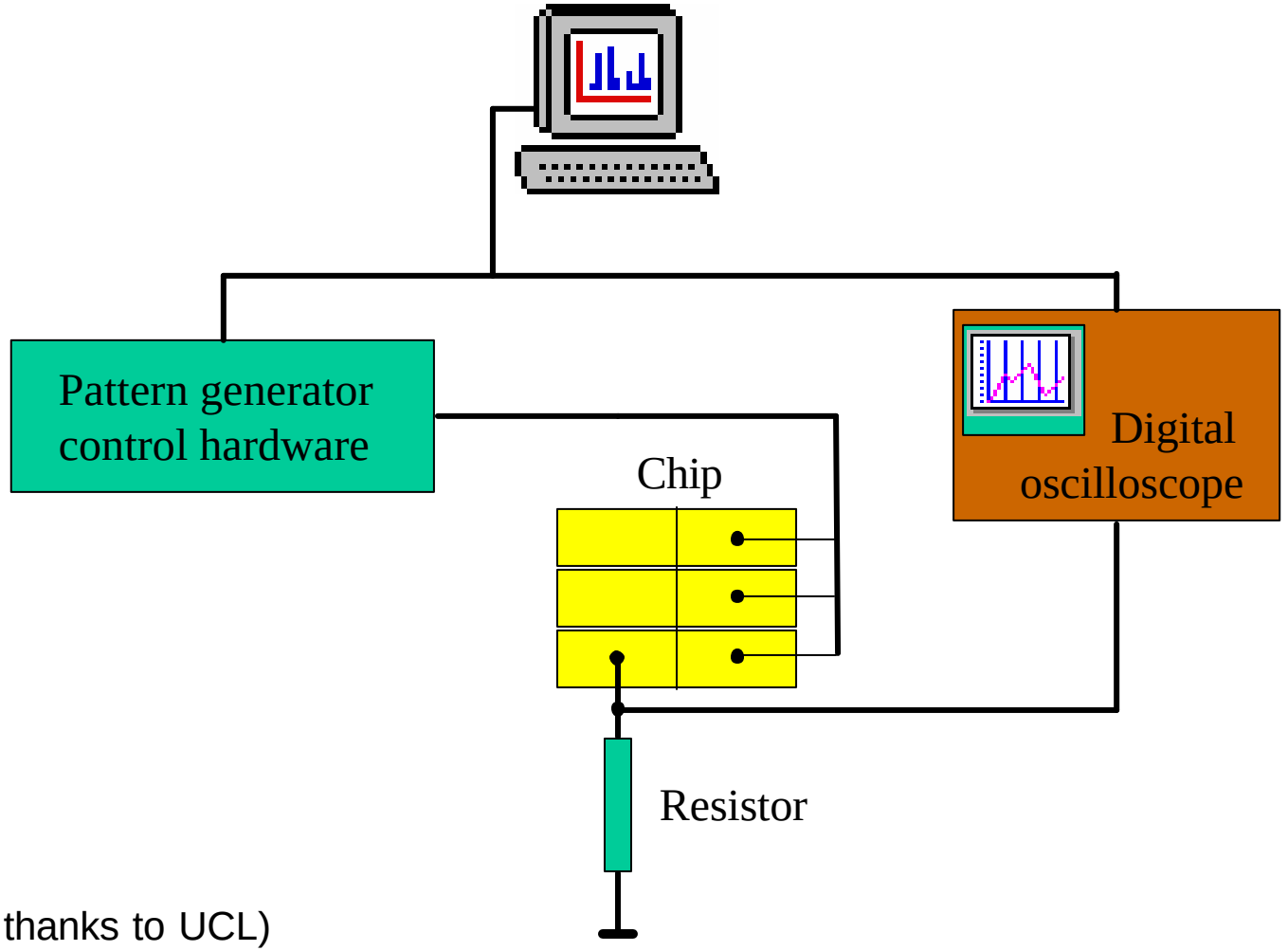
1995 – BellCore develops fault analysis attack

1995 – P. Kocher (US) develops timing attack

1997 – P. Kocher (US) develops diff. power analysis

2000 – J.-J. Quisquater (UC Louvain) presents electromagnetic analysis

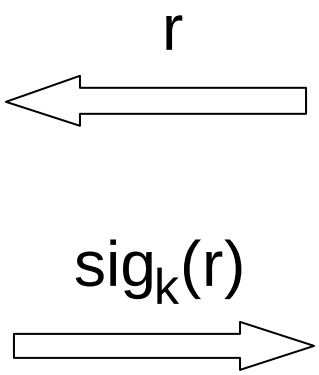
Power and timing analysis: Measurement set-up



(graphic: thanks to UCL)

RSA digital signature: Timing attack

Typical smart card protocol (banking etc.)



1. reader sends random challenge r

2. Card returns signed challenge

3. Reader verifies signature with public key

RSA digital signature algorithm

Signing of challenge r (inside card)

$y = sig_k(r) = r^k \bmod n$

where k = private key

Exponentiation algorithm

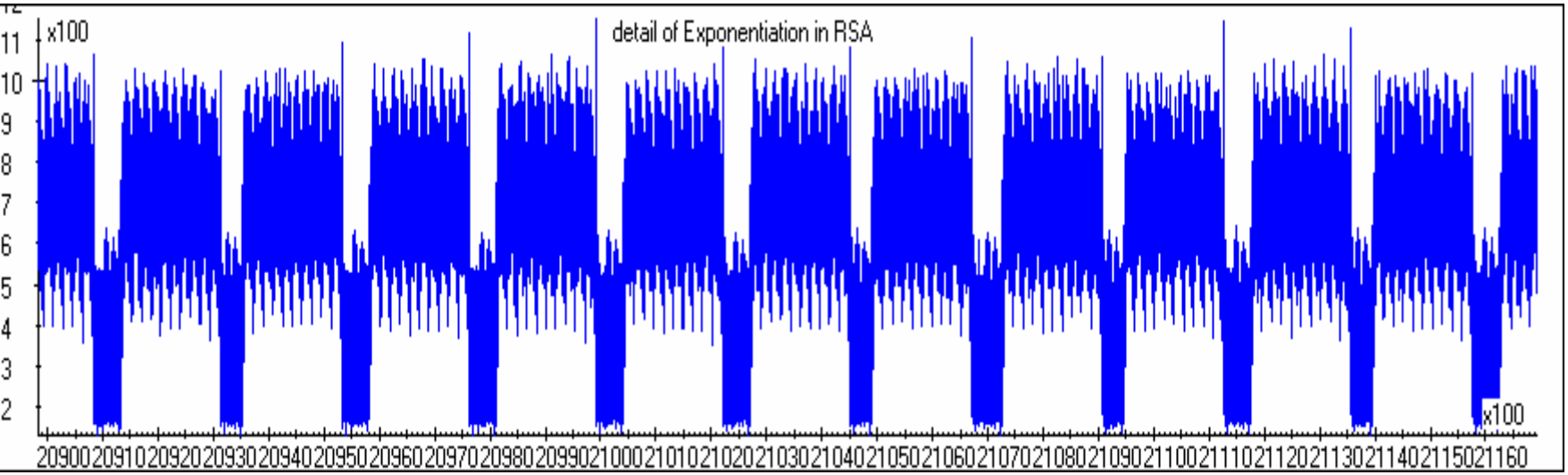
let $k = (k_{1024}, \ldots, k_2, k_1)$

```
1)  y ¬ r
For i=1023 Downto 1
2a) y ¬ y² mod n      /*
    SQUARE
    if k_i=1
2b) y ¬ y ´ r mod n   /* MULT
```

Note: SQR if bit $k_i = 0$

SQR + MUL if bit $k_i = 1$

RSA digital signature: Timing attack



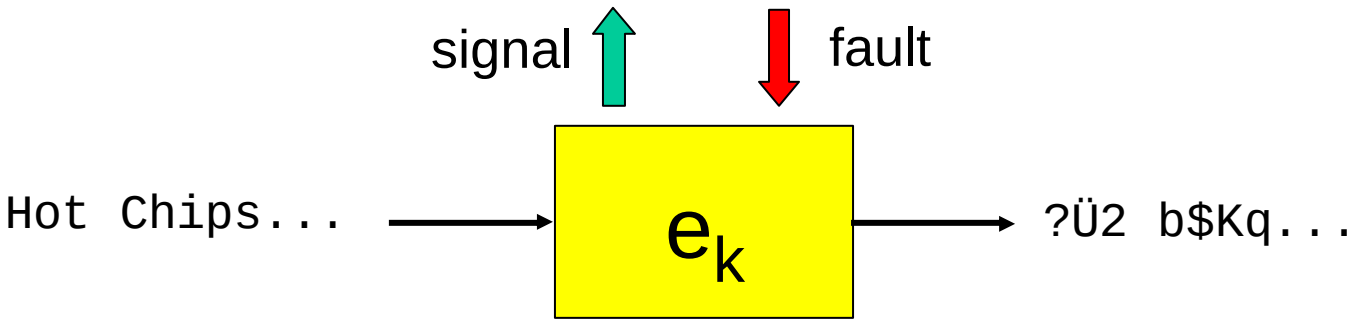
1 0 0 0 1 1 1

Power trace immediately reveals all bits of secret key!

(graphic: thanks to UCL)

Active side channel attacks

Fault-injection (via side channels)



- Attacker actively causes malfunction in device

Attacker has access to

1. ciphertext
2. some plaintext
3. side channel info

and tries to deduct key k

Fault injection attacks

Measurement set up for optical fault injection

(graphic: thanks to UCL)

Fault Injection Attacks

- Malfunction can lead to key leakage (often by comparing faulty and correct ciphertext)
- Fault types used thus far:
  - over-clocking
  - power spikes
  - heat
  - magnetic fields
  - optical
  - ...

Conclusions side channel attacks

- Major concern if the attacker has physical access to crypto device
  - Big issue for smart cards and pervasive computing appl.
  - No big issue for your average ecommerce server
- Counter measures (SW, HW) are available, but scores between attacker and designers are not settled yet
- Side channel attacks **must** be considered if one builds commercial crypto HW
- Excellent example for the need of crypto engineering

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. **Selected Aspects of Crypto Engineering**
   - Side channel attacks
   - **Reconfigurable hardware and cryptography**
4. The Future of Crypto Engineering
5. Crypto Engineering at U Bochum

Why crypto algorithms in hardware?

Two main advantages:

1. **Higher physical security**
   - Tamper resistant key access and algorithm modification
   - Restricting memory access on a processor is tricky and heavily OS dependent (trusted computing …)
2. **Software can be too slow**
   (effective encryption rates often < 100Mbit/sec)

Q: But why reconfigurable hardware??

Advantages of Reconfigurable Hardware (RCHW) in Cryptography

1. Architecture Efficiency
2. Algorithm Agility
3. Algorithm Upload
4. Parameter-Specific Architectures
5. Resource Efficiency
6. Algorithm Modification

RCHW & Crypto (1): Architecture-Efficiency

1. **Block cipher**
   - Block cipher atomic functions vary enormously (very large design space)
   - ASIC for more than 1 or 2 block ciphers very inefficiently
   - Wide data path (64 or 128 bit) are well-suited for custom design
2. **Public-key**
   - Switch between mod p and GF(2^m) arithmetic easy
   - Large range of bit length efficiently possible through reconfiguration (e.g., 160-256 bit arithmetic)

RCHW & Crypto (2): Algorithm-Agility

Observation: Modern security protocols are defined to be algorithm independent

- per-session negotiation of crypto algorithm
- wide variety of ciphers can be required
- Ex: IPSec algorithms
  - DES, 3DES, RC4, Blowfish, CAST, IDEA, ...
  - Diffie-Hellman, elliptic curves, ...
  - future extensions possible
- Run-time configuration attractive

RCHW & Crypto (3): Algorithm-Upload

Fielded application may need upgrade to a new algorithm because:

- Current algorithm was broken (A5 in GSM)
- Standard expired or new one created (DES, AES)
- List in algorithm independent protocol was extended
- Compatibility with new applications

Rem: Upload in ASIC-based devices is very costly or impossible (e.g., satellites)

RCHW & Crypto (4):
Parameter-Specific Architectures

1. **Ex: IDEA block cipher**
   - main ops: integer multiplication with sub-keys
   - degenerates into a constant multiplication if architecture for each key [Taylor/Goldstein 99]
2. **Ex: Arithmetic architecture in Galois fields**
   - are (far) more efficient if parameters (field order, irreducible polynomial) are fixed [Orlando/Paar 2000]
   - $GF(2^m)$ squaring, m variable: m/2 clocks
   - $GF(2^m)$ squaring, m fixed: 1 clock

RCHW & Crypto (5): Resource-Efficiency

Observation: The majority of security protocols uses

- symmetric as well as
- public-key algorithms

during one session, but not simultaneously.

⇒ Same FPGA device can be used for both through run-time reconfiguration

RCHW & Crypto (6): Algorithm-Modification

- Certain application domains prefer non-standard algorithms:
  - government applications
  - pay-TV etc. (to prevent fraud)
- Often realized as variations of commercial algorithms (e.g. DES with proprietary S-boxes)
- Protocol functions can be dynamically included in algorithm (e.g. change of mode of operation)

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. **The Future of Crypto Engineering**
   - **Pervasive computing and cryptography**
   - Challenges for crypto engineers
   - Opportunities for the VLSI community
5. Crypto Engineering at U Bochum

What are Embedded Systems?

- „A computer that doesn‘t look like a computer“, or
- Processor hidden in a product

+

=
Embedded System

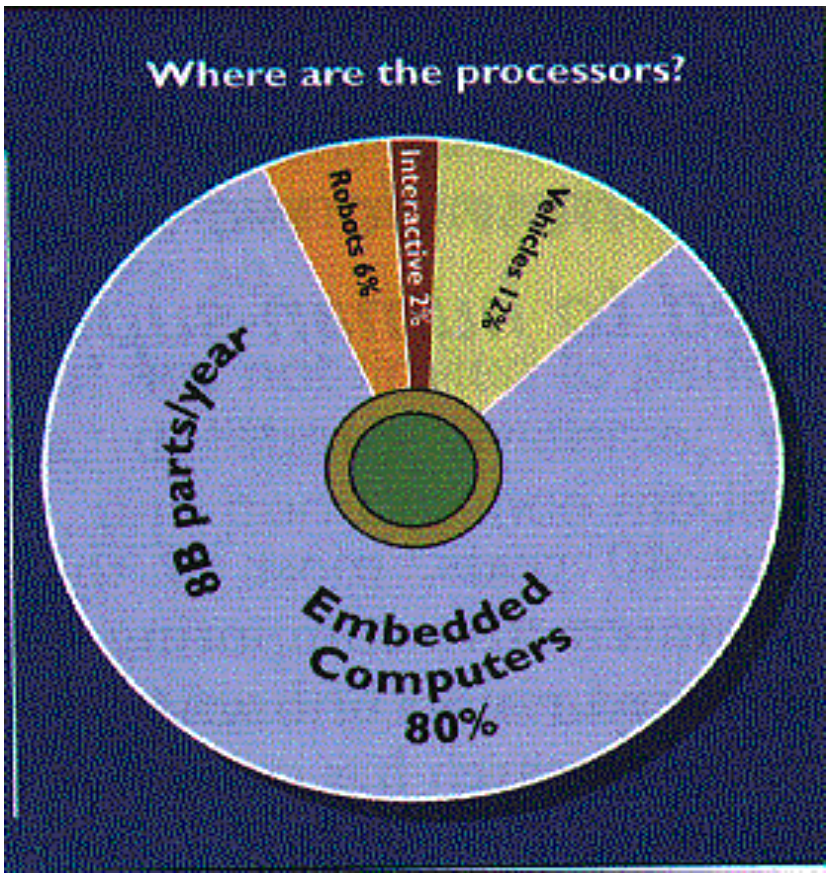
Characteristics of Embedded Systems

- Single purpose device

- Not general purpose like PC!
- Interacts with the world
- Primitive or no user interface

Is this really important?

Depends on your viewpoint, but: CPUs sold in 2000



HotChips 2003

From: Estrin et al.“Embedding the Internet,“ Communications of the ACM, no 5, 2000

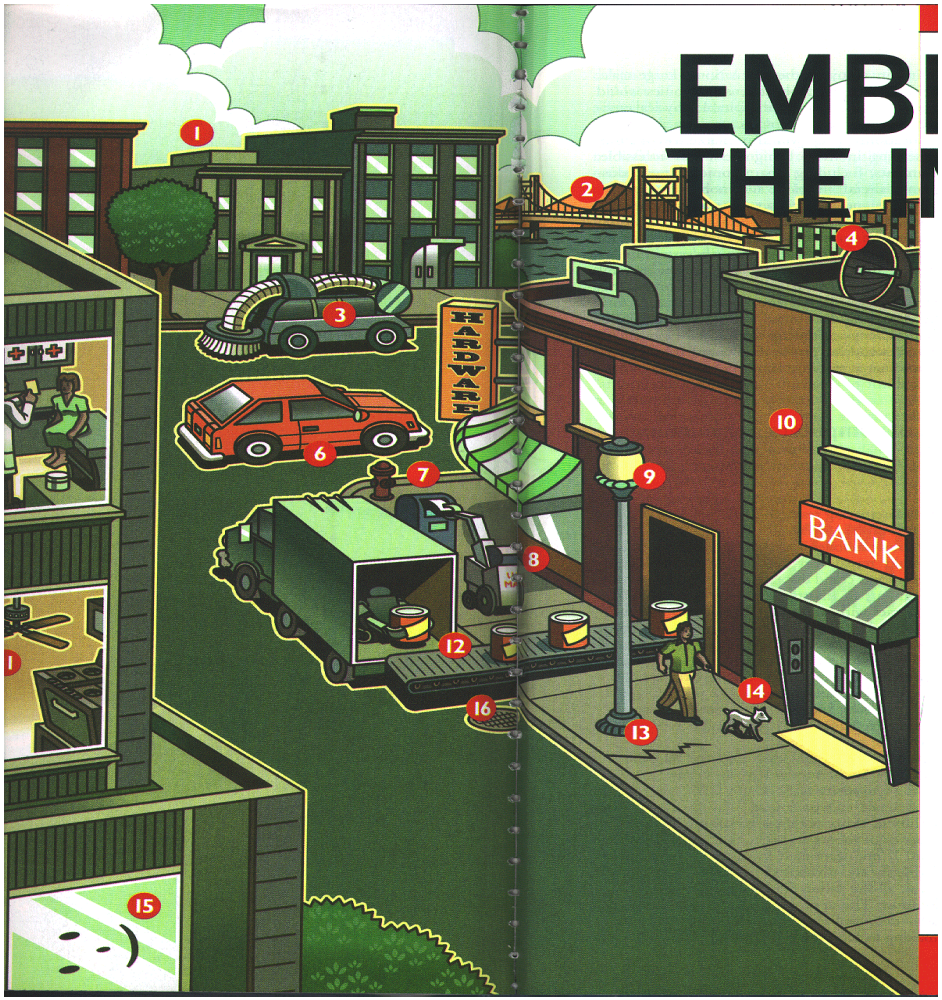
Characteristics of Traditional IT Applications

- Mostly based on interactive (= traditional) computers
- „One user – one computer“ paradigm
- Static networks
- Large number of users per network

Q: How will the IT future look?

eurobit
European Competence
Center for IT Security

Brave new pervasive world



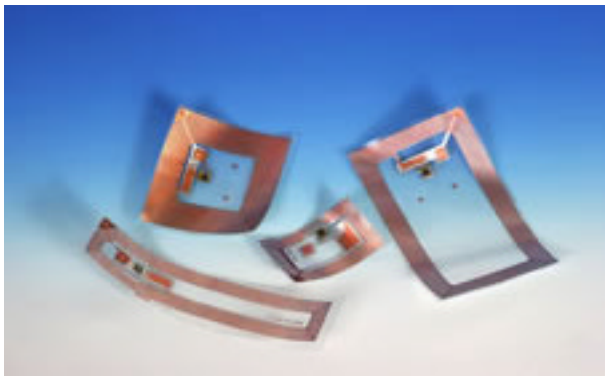
HotChips 2003

Communications of th
ACM, no 5, 2000

Examples for Pervasive Computing

- PDAs, 3G cell phones, ...
- Living spaces will be stuffed with nodes (audio/video)
- Refrigerators will communicate
- as will milk bottles
- Smart sensors in infrastructure (windows, roads, bridges, etc.)
- “Smart Dust”
- *Smart bar codes (autoID)*
- *Wearable computers* (clothes, eye glasses, etc.)
- ...

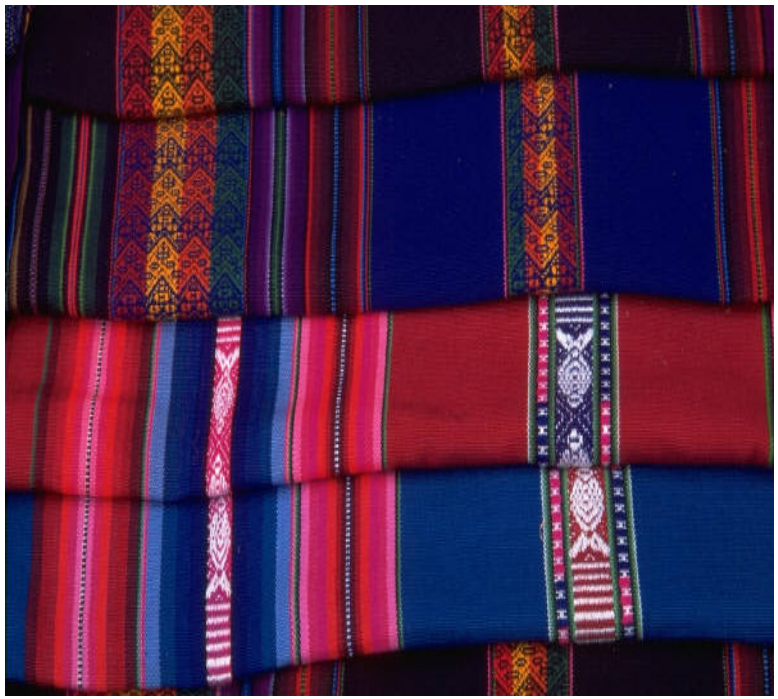
Pervasive Computing Case Study I: **Radio Frequency ID (RFID)**



- Smart tags with receiver & some processing
- Many applications in logistics, consumer products, ...

- MIT's AutoID Center: smart bar codes
- $500 \cdot 10^9$ bar codes scans **per day**
- Cost goal: 5 cents

Pervasive Computing Case Study II: **Smart Textiles (by Infineon)**



- Sensors in textiles
- Self-organizing network: fabric can be cut etc.
- Appl.: fire, motion, and anti-theft sensor
- Future version will incorporate LEDs

Pervasive Computing Case Study III: **Smart Dust**

- massively distributed sensor network
- goal size of grain of sand: 1 mm^3
- contains:
  - sensors
  - bi-directional wireless communications
  - computational ability
- inexpensive enough to deploy by the hundreds

eurobit
European Competence Center for IT Security

Security and Economics of Pervasive Applications

- „One-user many-nodes“ paradigm (e.g. 10^2-10^3 processors per human)
- Many new applications we don‘t know yet
- Very high volume applications
- Very cost sensitive
- People won‘t be willing to pay for security per se
- People won‘t buy/use products without security

Security Concerns in Pervasive Applications

- Often wireless channels ⇒ vulnerable
- Hacking into home devices, cars, …
- Contents protection in many applications
- Pervasive nature and high-volume of nodes increase risk potential
- Privacy issues (geolocation, medical sensors, monitoring of home activities, etc.)
- Stealing of services (sensors etc.)
- …

Why is Security in Pervasive Networks Difficult?

- Designers worry about IT functionality, security is ignored or an afterthought
- Security infrastructure (PKI etc.) is missing: Protocols?
- Secure embedded OS are difficult
- Attacker has easy access to nodes (side channel & tamper attacks)
- Computation/memory/power constrained

(red = crypto engineering issues)

Do we really need *crypto* in pervasive networks?

- crypto ops for **identification** is fundamental for embedded security
- almost all ad-hoc protocols (even **routing**!) require crypto ops *for every hop*
- **at least symmetric** alg. are needed
- **fancier** protocols with **public-key** alg.

Q. What type of crypto can we do?

Classification by Processor Power

Very rough classification of embedded processors

Class	*speed : high-end Intel*
Class 0: few 1000 gates	?
Class 1: 8 bit µP, ≤ 10MHz	≈ 1: 10^3
Class 2: 16 bit µP, ≤ 50MHz	≈ 1: 10^2
Class 3: 32 bit µP, ≤ 200MHz	≈ 1: 10

Case Study Class 0: RFID for Bar Codes

Recall: Class 0 = no μP, few 1000 gates

- Goal: RFID as bar code replacement
- AutoID tag: security “with 1000 gates” [CHES 02]
  - Ell. curves (asymmetric alg.) need > 20,000 gates
  - DES (symmetric alg.) needs > 5,000 gates
  - Lightweight stream ciphers might work

Status Quo: Crypto for Class 1

Recall: Class 1 = 8 bit µP, ≤ 10MHz

Symmetric alg: possible at low data rates
(e.g., micro-coded AES might work)

Asymm.alg: very difficult without coprocessor

Status Quo: Crypto for Class 2

Recall: Class 2 = 16 bit µP, ≤ 50MHz

Symmetric alg: possible

Asymm.alg: possible if

- carefully implemented, and
- algorithms carefully selected (ECC feasible; RSA & DL still hard)

Status Quo: Crypto for Class 3

Recall: Class 1 = 32 bit µP, ≤ 200MHz

Symmetric alg: possible

Asymm.alg: full range (ECC, RSA, DL) possible, some care needed for implementation

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. **The Future of Crypto Engineering**
   - Pervasive computing and cryptography
   - **Challenges for crypto engineers**
   - Opportunities for the VLSI community
5. Crypto Engineering at U Bochum

Future Challenges for Crypto Engineering

1. Challenges in pervasive applications
2. Speed optimization is not everything
3. Side channel attacks
4. Interdisciplinary work
5. Dissemination of results

Challenges (1): Crypto in Pervasive Applications

1. Symmetric algorithm for class 0 (e.g., 1000 gates) which are **secure and well understood***?*
2. **Alternative asymm. alg**. for class 0 and class 1 (8 bit µP) with **10x** time-area improvement over ECC?
3. Protocols with **symmetric crypto but asymmetric functionality**
4. Ad-hoc protocols **without long-term security** needs (e.g., for using ECC with 100 bits) ?
5. **Side channel protection** at very low costs?

Chaellenges (2): Speed Optimization is not everything

Past attitude:

As fast as possible, costs did not matter
(e.g., RSA modular multipl. Arch., DES hardware)

But:

1. Moore´s Law makes speed easy in SW and HW
2. Wide-spread commercial use of crypto makes cost optimization (power, code size, area, bandwidth) crucial

Research Challenge:

Develop techniques which optimize cost-performance ratio for given platform (SW, embedded, ASIC, FPGA)

Challenges (3): Side Channel Attacks

(very brief) Status Quo:

- Timing, fault induction, power analysis attacks, etc. proved powerful against unprotected hardware
- Software countermeasure work reasonably well

Research Challenges

1. Some important side channels (e.g., RF) and fault induction (e.g., optical) are poorly understood
2. Are there other side channels?
3. Hardware counter measures are just emerging

Challenges (4): Interdisciplinary Work

Crypto engineering benefits from other disciplines, e.g.,

- TRNG are poorly understood
- HW / SW co-design has barely been addressed

Challenges

1. Educate crypto people about other disciplines (e.g., novel VLSI technologies)
2. Entice people from other disciplines (e.g., novel VLSI technologies) to do crypto work
3. Encourage Ph.D. students to work interdisciplinary

Challenges (5): Dissemination of Results

Observations

- More and more products integrate cryptography
- Often non-optimum methods are used
- The wheel tends to get re-invented in industry

at the same time:

- More and more researchers are working on implementations (110 submissions @ CHES 2003)

Challenges

1. Make research results accessible for engineers without training in pure mathematics!
2. Organize the research results (books, courses)

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. **The Future of Crypto Engineering**
   - Pervasive computing and cryptography
   - Challenges for crypto engineers
   - **Opportunities for the VLSI community**
5. Crypto Engineering at U Bochum

Crypto Engineering and the VLSI Community

General thoughts:

- **Great opportunity**! Cryptography can greatly benefit from the knowledge in the VLSI community.
- **Big obstacles #1**: Requires interdisciplinary work! Ex: learning weired math, or accepting unusual rules (e.g., when is a system secure?)
- **Big obstacle #2**: Knowing where the real problems are

The following slides try to address obstacle #2.

Homework for the VLSI folks: (1) Power analysis (PA) resistance

Much work can be done on **hardware** countermeasures.

Some ideas for nice research:

1. Develop logic which is inherently resistant to PA
   see [Tiri/Verbauwhede]
2. Develop chip-level countermeasure which defeat PA.
3. Incorporate software countermeasures in tools
   (randomization of execution sequences etc.)

Homework for the VLSI folks: (2) Other side channel resistance

1. Develop countermeasures against EM-side channel leakage.
2. Develop countermeasures against timing attacks (see also #3 previous slide).

Homework for the VLSI folks: (3) Fault-injection resistance

Fault injection (over-clocking, power spikes, heat, etc) can lead to leakage of keys on „secure“ hardware.

1. Develop hardware robust against changes in the environment.
2. Develop sensor which sense attacks.

Homework for the VLSI folks: (4) FPGAs for crypto

1. How can keys
   - be stored securely
   - be deleted without residues
2. Uploading of encrypted configuration stream is available but we need good key management.
3. FPGAs with public-key arithmetic kernel

Further reading: [Wollinger/Paar 2003]

Homework for the VLSI folks: (5) Low-power crypto

Many future applications will need computationally expensive crypto ops in constrained environments.

1. Develop low-power arithmetic for public-key algorithms
2. Ultra-low power implementations of symmetric algorithms

Homework for the VLSI folks: (6) True random number generators

TRNG are needed in most crypto applications. Designing them is considerable trickier than one may assumes.

1. Which (physical) sources of randomness can be used?
2. How can those sources be exploited (amplification etc.)?
3. High-speed TRNG?

Contents

1. Cryptography, IT Security, and Crypto Engineering
2. Implementing Cryptographic Algorithms
3. Selected Aspects of Crypto Engineering
4. The Future of Crypto Engineering
   - Pervasive computing and cryptography
   - Challenges for crypto engineers
   - Opportunities for the VLSI community
5. **Crypto Engineering at U Bochum**

eurobits
European Competence
Center for IT Security

Center for Excellence in IT Security
„ E U R O B I T S “

+

Horst Görtz Institute for IT Security
(research)

GITS AG
(commercial)

eurobit
European Competence Center for IT Security

Horst Görtz Institute for IT Security

Chair for ITS & Cryptogr.	Chair for Commun. Sec.	Chair for Network Sec.
Prof. Hans Dobbertin	Prof. Christof Paar	Prof. Jörg Schwenk

Institute for E-Business Security

7 Chairs in Economics Dept.

HGI by the numbers

- founded in 2001
- 5 technical faculty (ECE, math)
- 7 business faculty
- ≈ 25 PhD students
- 5 years program „Dipl.-Ing. IT Security“
- 2 years program „Master‘s in IT Security“
- 4-5 workshops/conferences annually
- interdisciplinary: ECE, math, business, social science

HGI Research Focus Points

1. Technical research areas
   - Embedded security (Paar, Dobbertin)
   - Cryptographic Algorithms (Dobbertin)
   - Network Security (Schwenk, Paar)
   - Contents Protections (Sadeghi, Schwenk)
   - Trusted Computing (Sadeghi)
2. Non-technical research areas
   - ITS in the automotive supply chain, in logistics, ...
   - Social aspects: Critical infrastructres, ITS for KMUs, ...
   - Legal aspects

Crypto Engineering Research: Lightweight Crypto

1. Elliptic curves on smart card µP (8051) without coprocessor
2. Hyperelliptic curves on large range of embedded µP (ARM, DSP, PDA µP)
3. Public-key enabling instruction set extension for low-end 8 bit µP

Crypto Engineering Research

1. **Side channel attacks against embedded µP**
   - Ex: New collision attack against DES, AES, ...
2. **Security in ad-hoc networks**
   - Ex: New protocol family
3. **Contents protection**
   - Ex: Digital rights management (DRM) on embedded platforms
4. **New application domains for IT security & crypto**
   - IT security in **cars** (theft protection, telematics, contents protections, …)
   - IT security in **geoinformation** systems (contents protection, privacy, …)

Research Collaboration: ECRYPT

- ECRYPT = European *Network of Excellence (NoE)*
- 30+ crypto groups in Europe (and a few outside)
- Funded by EU Commission in 2003
- Structured in 5 Virtual Labs
- VAMPIRE (Virtual Applications and Implementations Research) Lab:
  1. Focus on future crypto engineering issues
  2. Workshops, summer schools, exchange of researchers
  3. Coordinated by U. Bochum

Related HGI Events

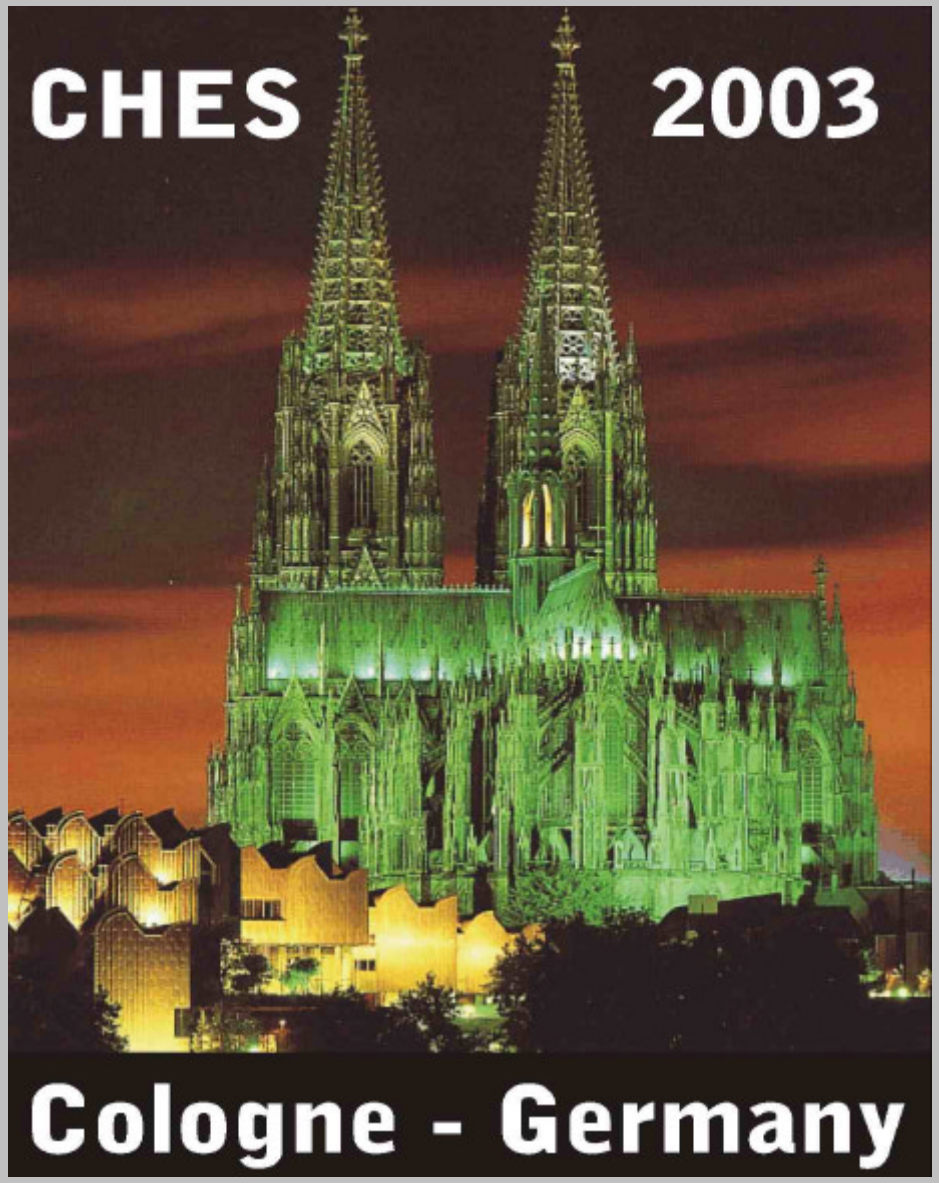
(see also www.crypto.rub.de)

- Workshop **Security in Ad-Hoc Networks**
  December 2002
- Workshop **Side Channel Attacks on Smart Cards**
  January 2003
- Conference **ESCAR (Embedded Security in Cars)**
  November 2003

  (first world-wide conference on this topic)
- and, of course, **CHES**

Cryptographic Hardware and Embedded Systems

Sept. 7-10
2003



chesworkshop.or

Further Reading

- Biham „A Fast New DES Implementation in Software, FSE 97, LNCS 1267, Springer-Verlag, 1997.
- Blum, Paar „High radix Montgomery modular exponentiation on reconfigurable hardware,“ IEEE Tr. on Computers, 50(7): 759-764, July 2001.
- Eldridge, Walter „Hardware implementation of Montgomery's modular multiplication algorithm,“ IEEE Tr. on Computers, 42(6):693--699, July 1993.
- Gladman “Implementations of AES (Rijndael) in C/C++ and Assembler,” http://fp.gladman.plus.com/cryptography_technology/rijndael/ (on July 19th, 2003).
- Kuo et al. “A 2.29 Gb/s, 56 mW non-pipelined Rijndael AES Encryption IC in a 1.8 V, 0.18 um CMOS Technology”, Custom Integrated Circuits Conference 2002.
- Koc, Paar et al. „Proceedings of CHES 99-2003“, Springer-Verlag LNCS.
- Koc et al. „Analyzing and comparing Montgomery multiplication algorithms,“ IEEE Micro, 16:26--33, 1996.

Further Reading (2)

- Orlando, Paar „A high performance elliptic curve processor for GF(2^m)," CHES 2000, LNCS 1965, WPI, Springer-Verlag, August 2000.
- Pelzl et al. „Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves," CHES 2003, LNCS, Cologne, Springer-Verlag, August 2003.
- Pelzl et al. „Low Cost Security: Explicit Formulae for Genus-4 Hyperelliptic Curves," SAC 2003, LNCS, Springer-Verlag, August 2003.
- Paar „Some remarks on efficient inversion in finite fields," 1995 IEEE International Symposium on Information Theory, Whistler, B.C., Canada, September 17-22, 1995.
- Solinas "Generalized Mersenne numbers", Technical report CORR-39, Dept. of C&O, University of Waterloo, 1999.
- Tiri, Verbauwhede "Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology", CHES 2003, LNCS 2003, Cologne, Springer-Verlag, September 2003.
- Taylor, Goldstein „A High-Performance Flexible Architecture for Cryptography,", CHES 99, LNCS 1717, WPI, Springer-Verlag, August 1999.
- Trimberger et al. „A 12 Gbps DES encryptor/decryptor core in an FPG," CHES 2000, LNCS 1965, WPI, Springer-Verlag, August 2000.

Further Reading (3)

- Weaver „A High Performance, Compact Irondale (AES) core for the Virtex Family FPGA," http://www.cs.berkeley.edu/~nweaver/rijndael/ (on July 19th, 2003).
- Wilcox et al. „A DES ASIC Suitable for Network Encryption at 10 Gbps and Beyond," CHES 99, LNCS 1717, WPI, Springer-Verlag, August 1999.
- Wollinger, Paar „How secure are FPGAs in cryptographic applications?", FPL 2003, Lisbon, Portugal, September 1-3, 2003.