# Building Secure Hardware

ECRYPT II Workshop on Physical Attacks

November 27th, Graz, Austria

Stefan Mangard

Infineon Technologies, Munich, Germany


Stefan.Mangard@infineon.com

# Outline

**What is secure hardware?**

- Assets and Requirements
- Threats

**State-of-the Art Research**

- Overview of Countermeasures
- Example: CIPURSE™

# PART I

# Assets and Requirements

# Classical Security Requirements

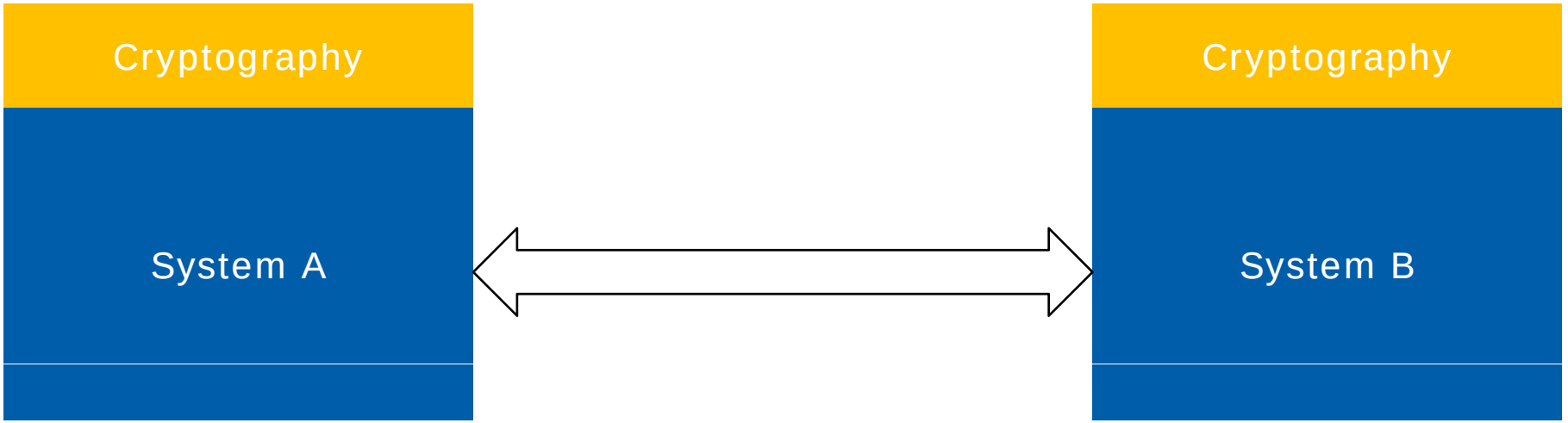
- There are many applications with security needs
- The requirements usually include a communication interface that needs to provide the classical IT security qualities for exchanged messages
  - Authenticity
  - Integrity
  - Confidentiality

→ Implement cryptography to fulfill these requirements

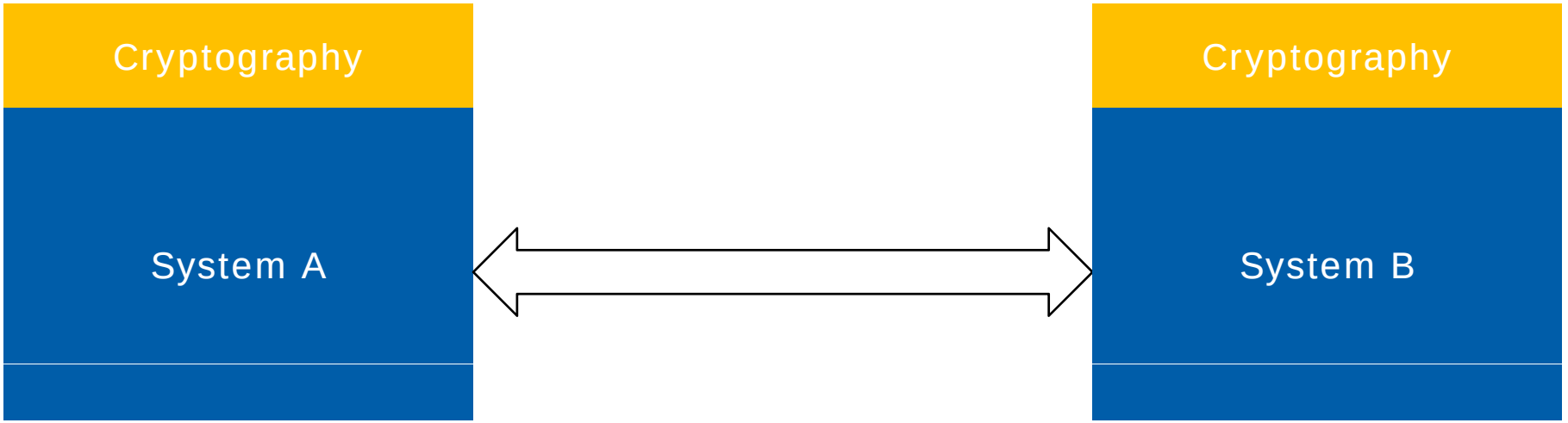
# Implementing Cryptography

- Cryptography is added to the system by implementing corresponding hardware or a corresponding library

- Using cryptography, systems can exchange messages securely → the asset (the message) is protected against attacks on the communication line



# What About Other Assets and Requirements?

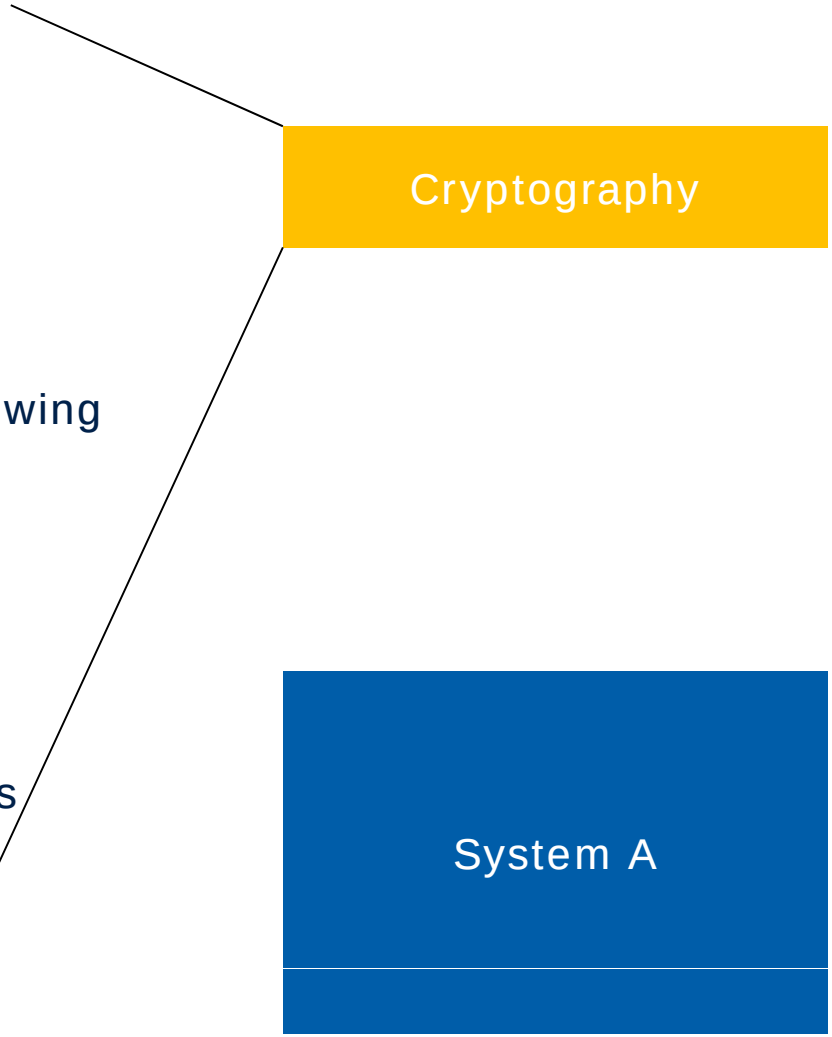


# What About Other Assets and Requirements?

- Requirements for the cryptographic implementation:
  - Confidentiality of private/secret keys
  - Confidentiality of intermediate values

Depending on the application also the following properties might be required:

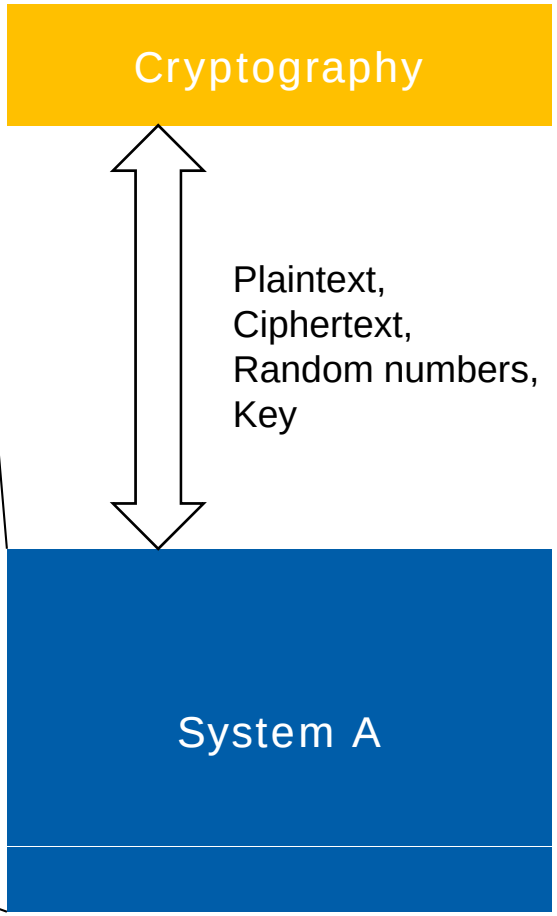
  - Integrity of all intermediate values
  - Integrity of the keys
  - Confidentiality and integrity of plaintexts
  - Integrity of ciphertexts
  - Confidentiality of random numbers
  - Integrity of random numbers



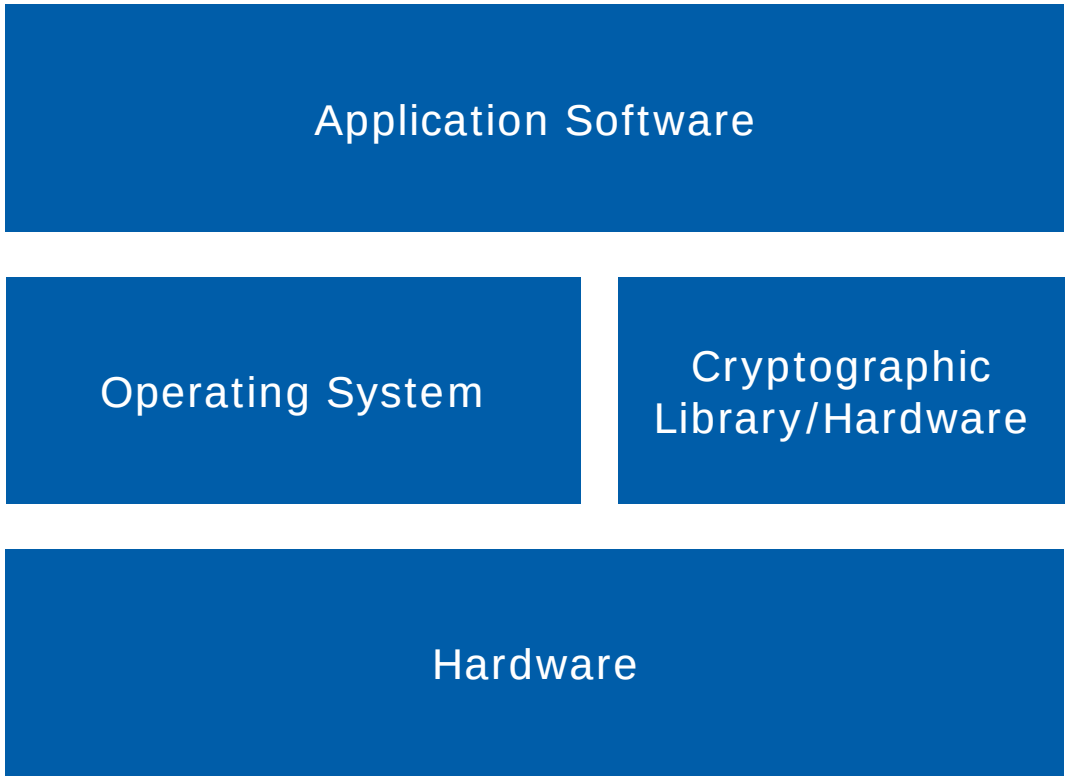
# What About Other Assets and Requirements?

- Requirements for the system:
  - Many of the assets (keys, messages, random numbers) are also processed by other components of the system and not just the cryptographic implementation

Hence, all the requirements imposed by these assets are also requirements for the system



# Assets Are Located in All Basic Parts of the System




The requirement essentially is to build a system (hardware and software) that is able to provide confidentiality and integrity of all the data (this includes keys and random numbers) it stores and processes and that provides true random numbers

# High-Level Security Requirements for Security ICs in Certification

- The “Security IC Platform Protection Profile” (BSI-PP-0035) that is used to evaluate security ICs and smart card products defines four high-level security concerns:

- **SC1:** Integrity of User Data and of the Security IC Embedded Software (while being executed/processed and while being stored in the TOE’s memories)

- **SC2:** Confidentiality of User Data and of the Security IC Embedded Software (while being processed and while being stored in the TOE’s memories)

- **SC3:** Correct operation of the security services provided by the TOE for the Security IC Embedded Software

- **SC4:** Deficiency of random numbers

# PART II

# Threats

# The Two Main Threat Scenarios

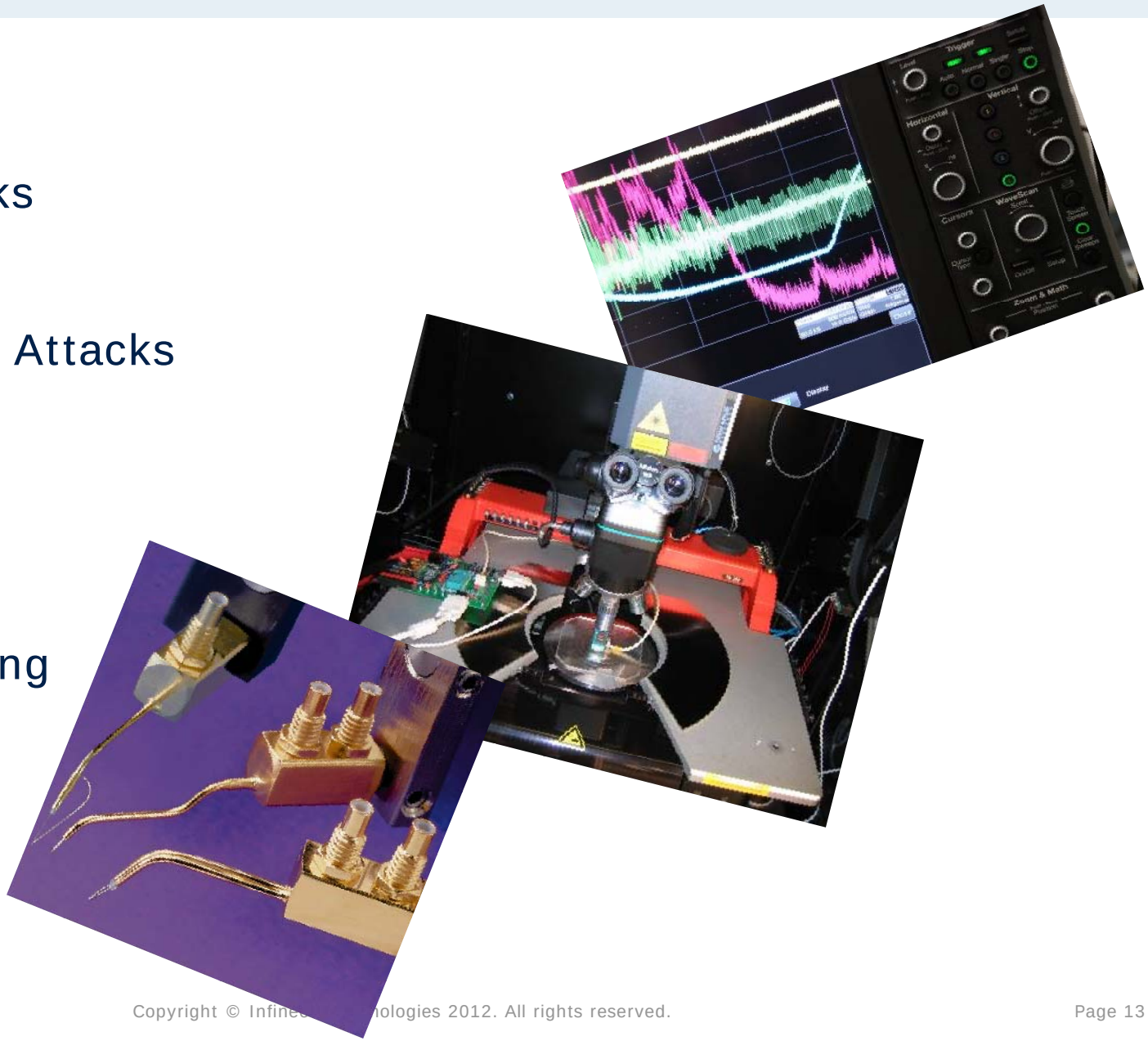
- Secure Environment
  - Logical attacks
  - Examples: The classical Internet communication scenario, where the attacker does not have physical access to the device

- Non-Secure Environment
  - Logical and physical attacks
  - Examples: all kinds of embedded devices, USB sticks, smart cards, RFID tags, …

→ In those scenarios, where there the strongest limitations of resources (power, energy, and area), typically logical and physical attacks need to be considered

# The Threat Classes in Non-Secure Environments

- Logical Attacks
- Side-Channel Attacks
- Fault Attacks
- Probing/Forcing



# Power/EM Analysis Today

- Attack Strategies
  - Profiled attacks are an established tool
  - Exploitation more and more focuses on multiple points and their relationship (higher-order attacks)
  - Almost any statistical tool that can be used to measure dependencies between random variables has meanwhile been applied to power analysis
- Measurement Setup
  - Measurements of the power consumption is often done via the electromagnetic field
  - Small coils allow local attacks on the chip
  - Basic DPA attack can be conducted with simple and cheap USB oscilloscopes
  - Storage and processing power of modern PCs and oscilloscopes allows to do attacks with more and more traces

# Fault Attack Today

- Attack Strategies
  - There are published fault attacks on many cryptographic algorithms
    - A single fault induction is sufficient to break RSA implemented with CRT
    - A single fault induction is sufficient to reveal an AES-128 key
- Attack Setup
  - Lasers are the most effective method to produce controlled and localized faults in an IC
  - Attack setups can contain a laser to perform attacks from the front- as well as from the backside of the chip
  - Setups being able to induce multiple faults are becoming more and more prominent

# Probing/Forcing Attacks

- Attack Strategies
  - A needle at the right position can be devastating for a system (e.g. putting a needle on a wire of an Sbox in an AES design with only one Sbox)
  - Open source tools for reverse engineering (degate)

- Attack Setup
  - As the technology becomes smaller and smaller, also the analysis tools (FIB, probing stations) become more and more advanced

# Component vs. Threat Matrix

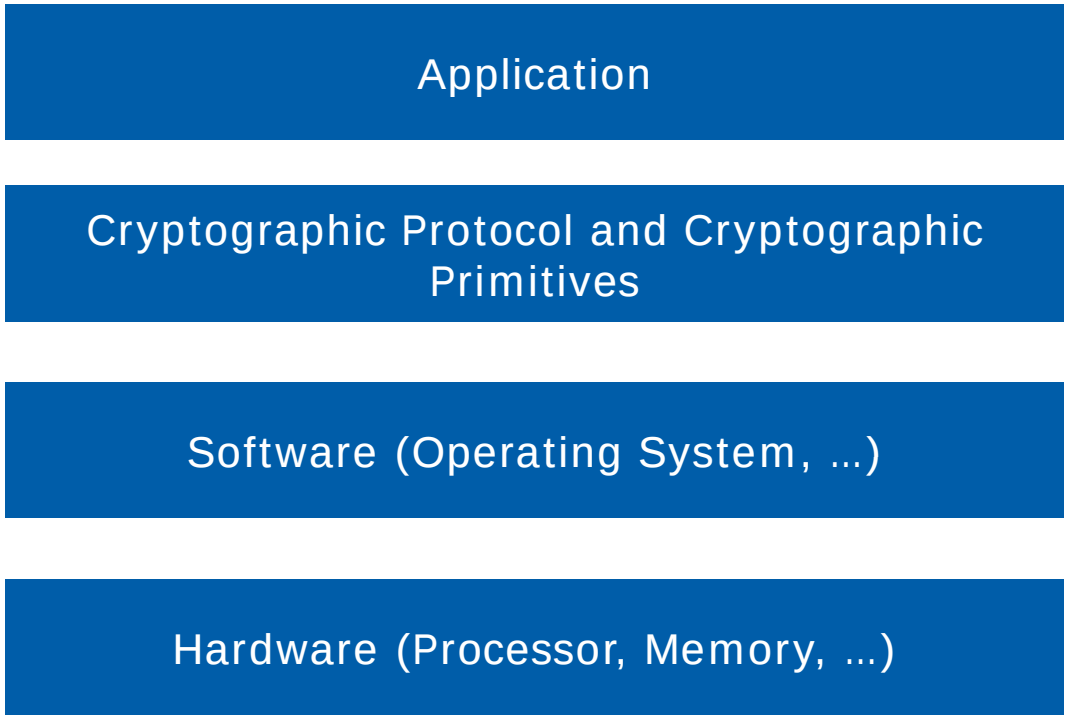
| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | | | | |
| Cryptographic Libraries | | | | |
| Processors | | | | |
| Memories and Key Storage | | | | |
| Random Number Generators | | | | |
| Operating Systems | | | | |
| Application Software | | | | |

# Component vs. Threat Matrix

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

Systems are usually attacked are their weakest link → If one threat is not addressed, this is the likely spot that will be attacked

# Countermeasures Against Everything at Each Level?




Not everything needs to be solved at every level. However, whatever has not been solved in one abstraction level is a requirement for the next one

→ The hardware takes all the rest

# Minimum Requirements of Cryptographic Primitives/Protocols for Hardware/Software

- Bounded leakage for all data
  - Protection against probing (Hardware)
  - Protection against side-channel attacks (Hardware/Software)
- Integrity protection
  - Protection against forcing (Hardware)
  - Protection against fault attacks (Hardware/Software)
- Generation of random numbers

# PART III

# Overview of Countermeasure

# Overview

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

# Cryptographic Implementations (I/II)

- By far the area with most publications

- However, most publications only address the threats separately
  → a fundamental problem is not captured

  Countermeasures against side-channel attacks and fault attacks are inherently conflicting

  - The more redundancy is added to a system in order to prevent fault attacks, the more side-channel information leaks
  - The more randomness is used to mask or blind operations, the more targets exist to perform fault attacks

# Cryptographic Implementations (II/II)

- Although a lot of research has been done, fundamental questions are still open
  - Quantification of the information contained in a leakage trace
  - Practical masking schemes against higher-order attacks
  - No efficient redundancy scheme except for duplication for block ciphers like AES
  - ...

# Overview

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

# Processors (I/II)

- There is only very limited research published in the field of building secure processors; However,

**→ The processor is where it is all happening ←**

- The processor needs to provide confidentiality and integrity of the data it processes

- In particular, fault attacks are a highly important field of research

# Processors (II/II)

- Examples of critical fault attacks on a processor:
  - The processor sends confidential data instead of public data due to
    - Faulty pointers
    - induced branches from critical functions to I/O functions
    - …
  - The processor skips critical operations and accesses confidential data, e.g.
    - Password or authentication functions are skipped
    - Encryption functions are bypassed
    - …

# Fault Attacks

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

# Memories

**Where and how do you store your keys and other confidential data?**

Typically, secret keys are temporarily assembled in the chip just before use. Secrets are not stored in plaintext, but assembled from different sources and different design elements.

- PUFs are a strong trend to “securely” store keys
  - However, when a PUF is powered all the well-known attacks can be done: side-channel, fault, probing, …

    → Only few publications look at the big picture

- Memory encryption
  - Protecting the software and the user data is a fundamental requirement

  → Only few publications exist that address the topic of memory encryption

# Overview

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

# Random Number Generators

- There are many publications about noise sources
- However, the important question is to build an RNG that is resistant against all the threats at the same time
- In particular the robustness of the noise source and the random number generator is crucials

# Overview

| Component | Side-Channel | Fault | Probing/ Forcing | Software/ Logical Attacks |
|---|---|---|---|---|
| Cryptographic Coprocessors | X | X | X | X |
| Cryptographic Libraries | X | X | X | X |
| Processors | X | X | X | X |
| Memories and Key Storage | X | X | X | X |
| Random Number Generators | X | X | X | X |
| Operating Systems | X | X | X | X |
| Application Software | X | X | X | X |

# Software Security

- Main research direction is to provide protection against software attacks

  → there are almost no research papers on building operating systems secure against fault and side-channel attacks

- There is a large research field on how to secure software in particular against fault attacks

# PART IV

# Example – CIPURSE™

# Background

- The work started in 2008
- The basic motivation was to build a low-cost system for contactless applications that provides authentication and secure messaging based on AES
- The approach to achieve this goal was to consider all attacks at all abstraction levels and to reduce the requirements for the subsequent levels to a minimum
- It turned out that parts of our work are similar to the research on fresh re-keying which was done later within ECRYPT

# The Core Problem: Session Key Derivation

The basic idea:

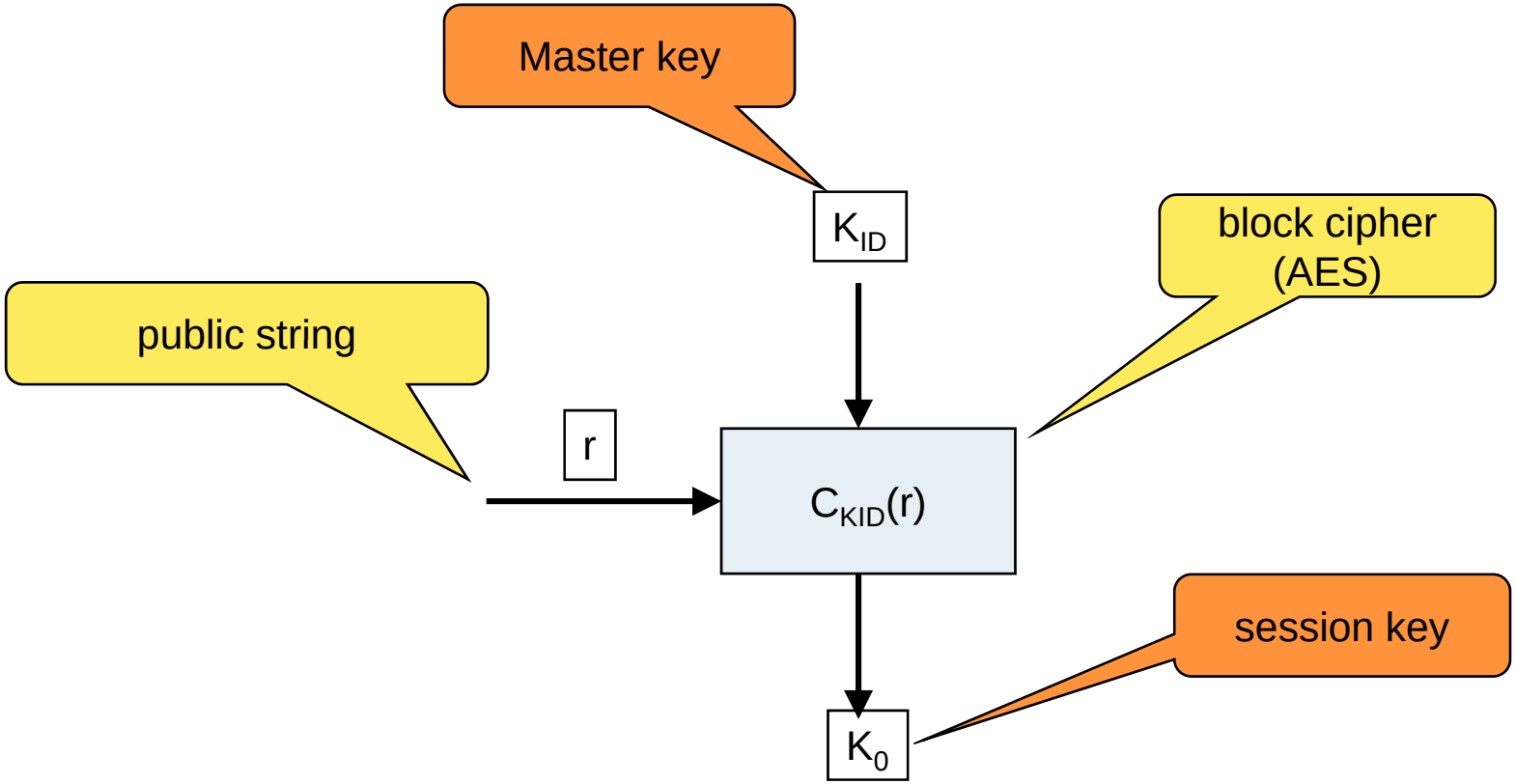
Do not solve the independent problems of

- cryptography
- side-channel resistance
- fault attack resistance

*at once within* the crypto algorithm, because highly complex confusion/ diffusion/... operations are notoriously hard to protect against all kinds of SCA (i.e. DPA,DFA,...).

# The Core Problem: Session Key Derivation

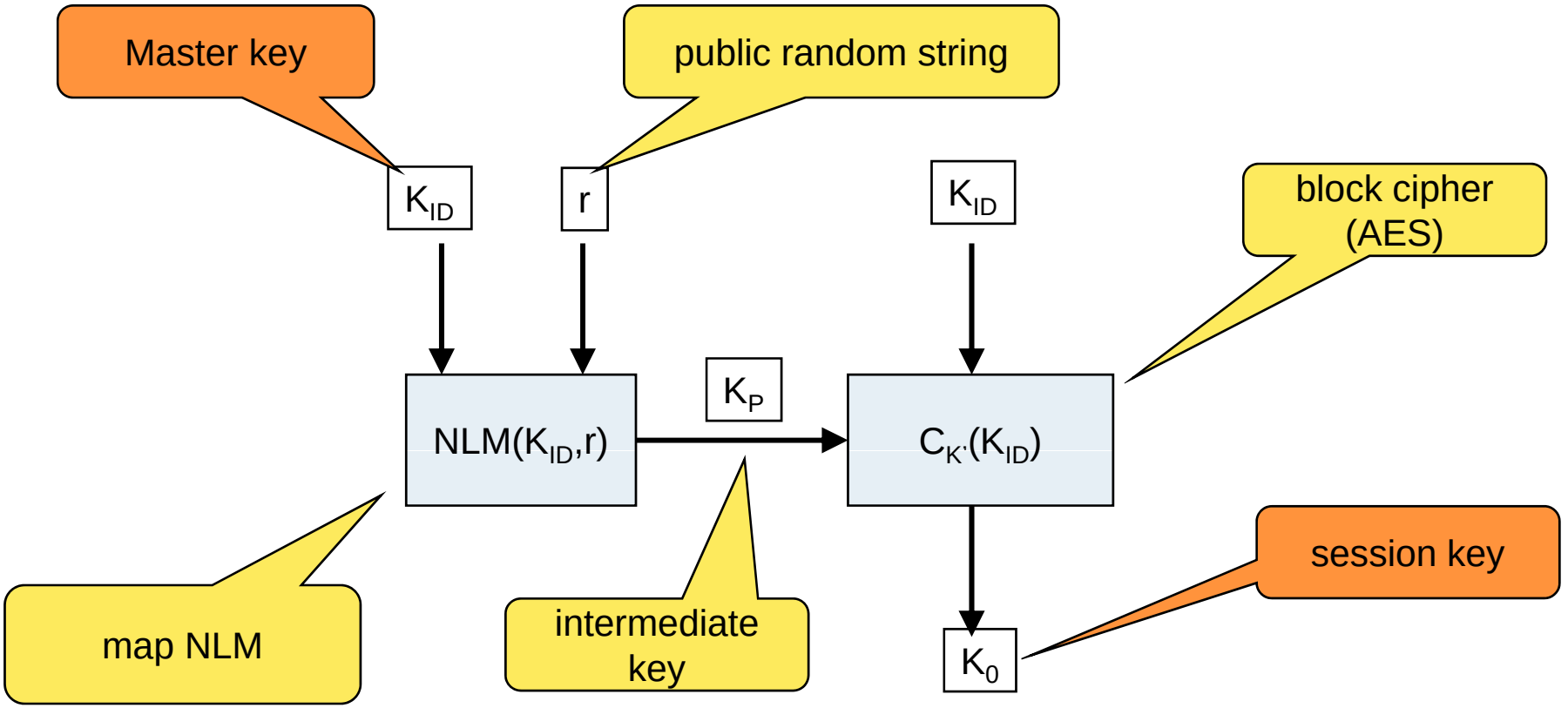
The classical DPA problem:



# The Core Problem: Session Key Derivation

Divide-and-conquer approach:

- Transform input of cipher using a map which can be easily protected against SCA.
- Separate the requirements on cryptography and SCA



# New Protocol's Core: Side-Channel Resilient Session Key Derivation

- Requirements and properties of the map NLM and Cipher C:
  - C: Must be secure against SPA (only).
  - NLM: Must be secure against SPA and DPA.
    *Hence it is called **N**on-**L**eaking **M**ap.*
    - need not be cryptographically strong
    - strong diffusion (prevent divide-and-conquer in DPA)
    - small in hardware implemention
    - fast in software implementation
    - stateless (otherwise synchronization problem like Kocher's approach)
    - simple to protect against DPA, e.g. regular algebraic structure that can be easily masked/randomized

- We found several solutions:
  - based on finite field multiplication

# NLM example

NLM: $GF(2)[x]^n \times GF(2)[x]^n \rightarrow GF(2)[x]^n / P(x)$

- Properties of polynomial P(x):
  - irreducible
  - high weight (good diffusion)

- One option to protect the NLM is the following:generate two secret values $r_1$, $r_2$ from which the public value $r = r_1 * r_2$ and a secret value $k_P$ is deduced:

$$k_P = r * k_{ID} = (k_{ID} * r_1) * r_2$$

# The Complete Protocol and Implementation

- After having derived a session key, the protocol supports a MAC and an encryption mode
  → both are designed in such a way that no AES encryption uses the same key twice (inherent protection against DFA and DPA attacks)

- The protocol significantly reduces the implementation costs for the implementation of the countermeasures for the cryptographic part

- Also the implementation of this protocol has requirements on the processor (probing/forcing, fault attacks on control flow, …); Yet they are significantly smaller than in conventional approaches

# Conclusions

- Building secure hardware has many aspects

- In many publications (including many of mine ;-) the focus too much on single aspects and not on the “big picture”

  → There are many interesting research fields that have not received significant attention yet

- When addressing security requirements at the highest possible levels of abstraction, the requirements for the hardware can be reduced

  →However, attacks like probing/forcing and fault attacks need to be addressed in hardware

  Developing secure hardware is always in interesting job ☺

# ENERGY EFFICIENCY
# MOBILITY
# SECURITY

Innovative semiconductor solutions for energy efficiency, mobility and security.