Side Channel-aware Design

Novel Analysis and Construction Concepts

Sorin Alexander Huss

A. Biedermann
L. Chen
A. Heuser
M. Jung
M. Kasper
R. Laue
F. Madlener
W. Schindler
A. Shoufan
M. Stöttinger
Q. Tian
M. Zohner

CASED
Center for Advanced Security Research Darmstadt

and

Technische Universität Darmstadt
Germany

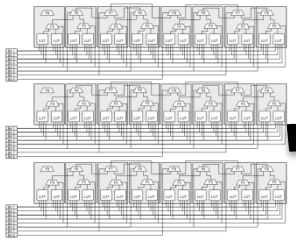
Overview

- Introduction
  - Motivation
  - Generic Attack Scenario
- Power Amount Analysis
  - Outline of DPA
  - AWGN Channel Model
  - Statistical Calculations
  - Analysis Results
- Constructive Methods
  - Localization of Leakage
  - Dynamically Mutating Processing Units
  - Virtualization in Multicore HW-Modules
- Summary

Why Constructive Side Channel Analysis? (1)

CASED

Design

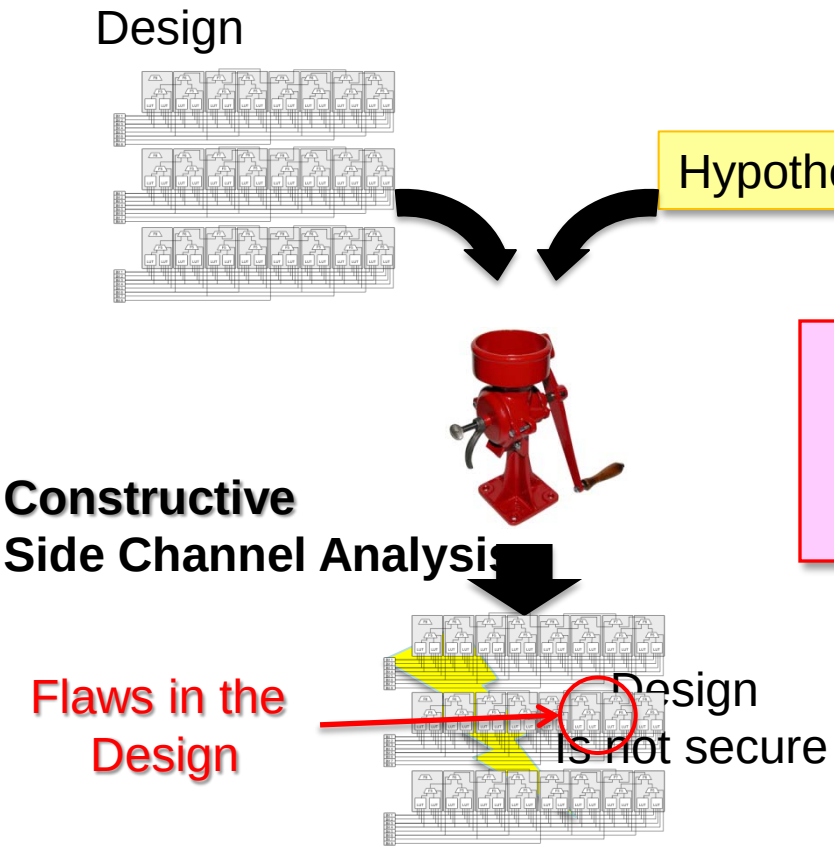


Hypotheses

Usual
Side Channel Analysis

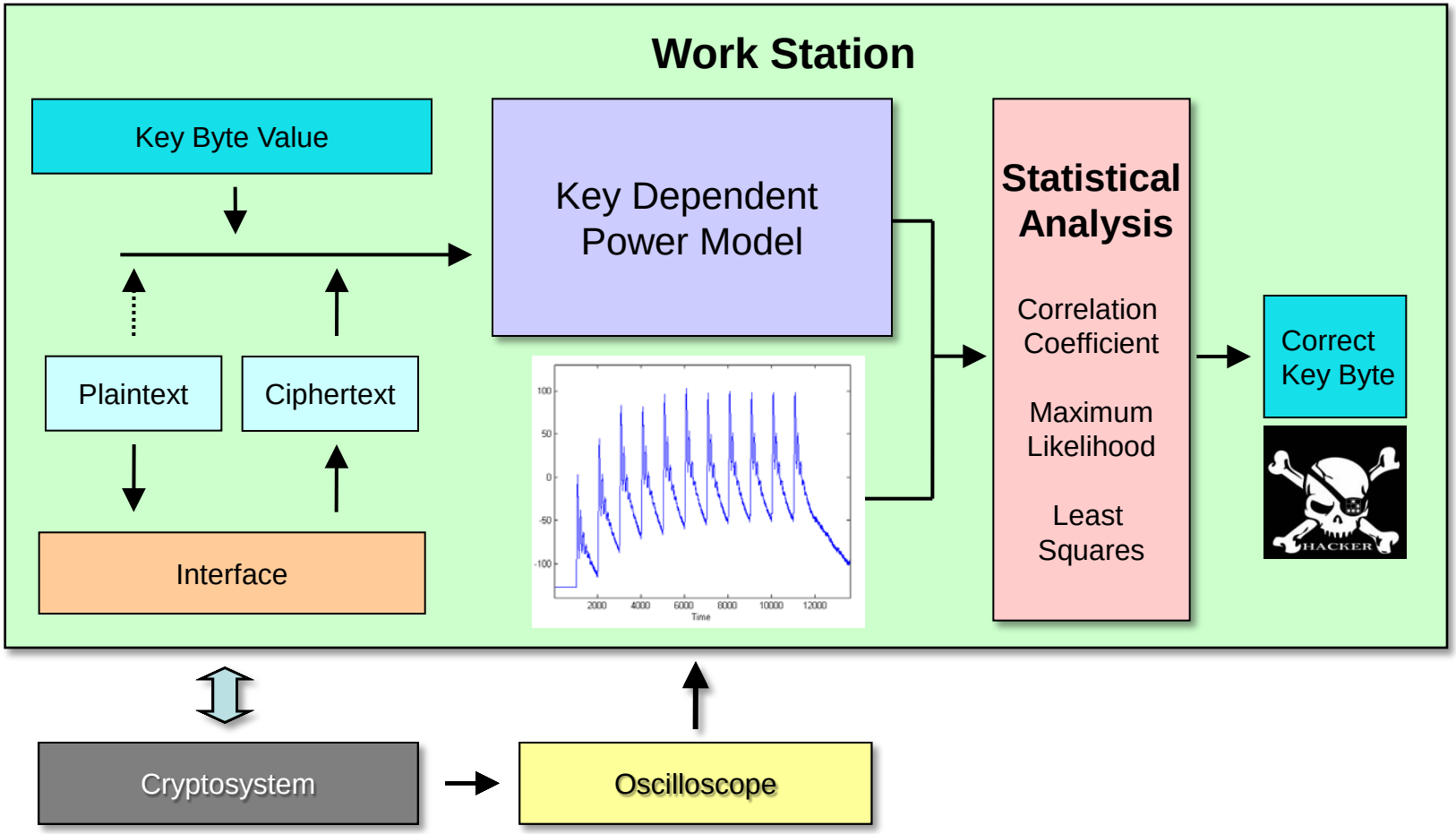
Design
is secure

Why Constructive Side Channel Analysis? (2)



Quantifying side channel in order to support the design phase

Generic Power Attack Scenario



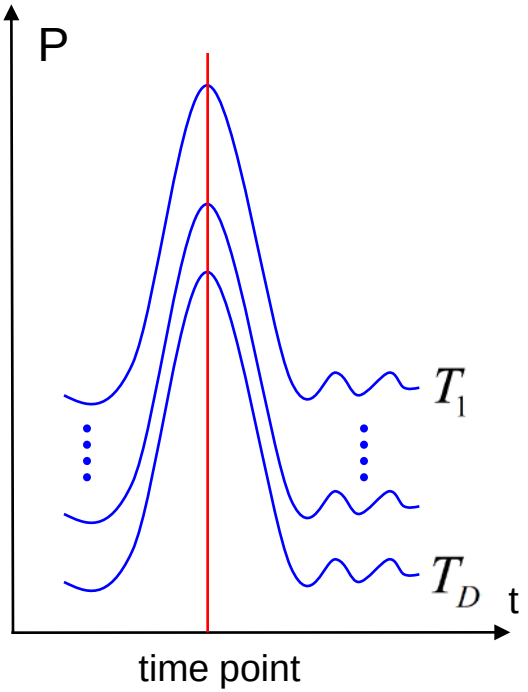
Overview

- Introduction
  - Motivation
  - Generic Attack Scenario
- Power Amount Analysis
  - Outline of DPA
  - AWGN Channel Model
  - Statistical Calculations
  - Analysis Results
- Constructive Methods
  - Localization of Leakage
  - Dynamically Mutating Processing Units
  - Virtualization in Multicore HW-Modules
- Summary

Power Traces

$$P_{total} = P_{op} + P_{Data} + P_{el.noise} + P_{const}$$

P_{op}	Data dependent, but key independent power consumption
P_{Data}	Data and key dependent power consumption
$P_{el.noise}$	Electronic noise of the hardware
P_{const}	Constant power consumption of hardware module



- Functions of time
- Additive property
- Calculated at a certain time point in several traces

Differential Power Analysis (1)
Noise in Power Traces

Distribution of Noise in the Traces:

Perform the same operations with the same input data. The fluctuations of the power value at the same time point in the captured traces are the noise.

Gaussian Distribution $$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2\right)$$

$$P_{el.noise} \sim N(0, \sigma^2)$$

Only for a certain time point considered!

Differential Power Analysis (2)
Power Models

Hamming Weight: $HW = \text{HammingWeight}(c_i \oplus k)$

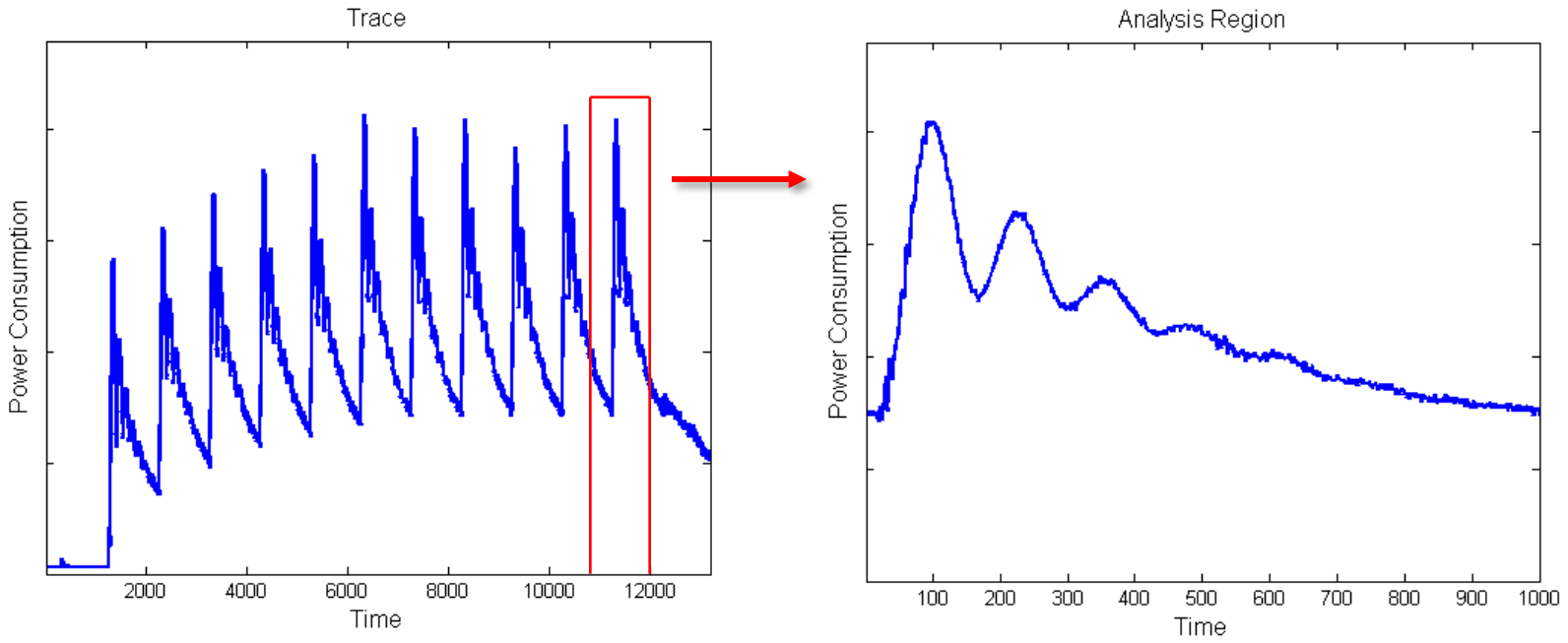
Hamming Distance: $HD = \text{HammingWeight}(c_i \oplus \tilde{d}_i)$

- **Instantaneous Model**: Power model based on the state of a certain register *at some time point*
- **Process Model**: Power model based on two states changing *within a time interval*

Differential Power Analysis (3)
Attack Scenario

Analysis Region:

- Large number of time points in area of interest
- Identification takes a lot of computation time

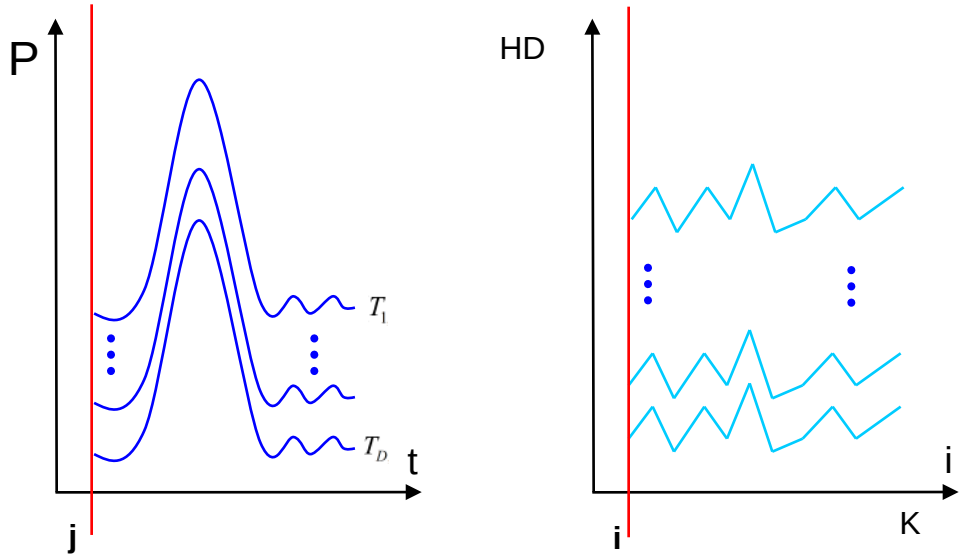


Differential Power Analysis (4)

Attack Outline

- T : Power traces matrix
- H: Hypothesis matrices (mapped from power model)
- R: Results matrix

$$\begin{pmatrix} R_{1,1} & \cdots & R_{1,M} \\ \vdots & \ddots & \vdots \\ R_{K,1} & \cdots & R_{K,M} \end{pmatrix} = StatAnalysis\left(\begin{pmatrix} T_{1,1} & \cdots & T_{1,M} \\ \vdots & \ddots & \vdots \\ T_{D,1} & \cdots & T_{D,M} \end{pmatrix}, \begin{pmatrix} H_{1,1} & \cdots & H_{1,K} \\ \vdots & \ddots & \vdots \\ H_{D,1} & \cdots & H_{D,K} \end{pmatrix}\right) \quad R_{i,j} = CorrCoef\left(T_{D,j}, H_{D,i}\right)$$



Subkey k_l , l = 1, …, N

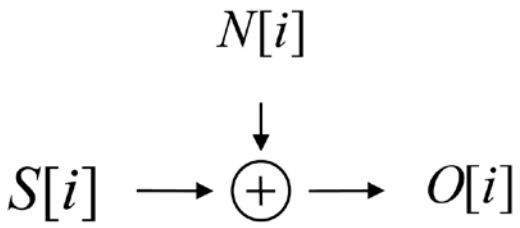
$k_l = \{k_l(i)\}$, i = 1, …, K

Classification of some Power Analysis Methods

Algorithm	Differential Power Analysis	Stochastic Approach	Template Attack
Time Points Usage	One time point	Several time points	Several time points
Profiling Phase	No	Yes	Yes
Comments	Only one time point contributes to information leakage	▪ Identical device is required for the profiling phase ▪ The more time points are being used the more computational time and memory space are needed	
Prerequisite	Power traces must be aligned in time domain		

Power Amount Analysis (1)
AWGN Channel

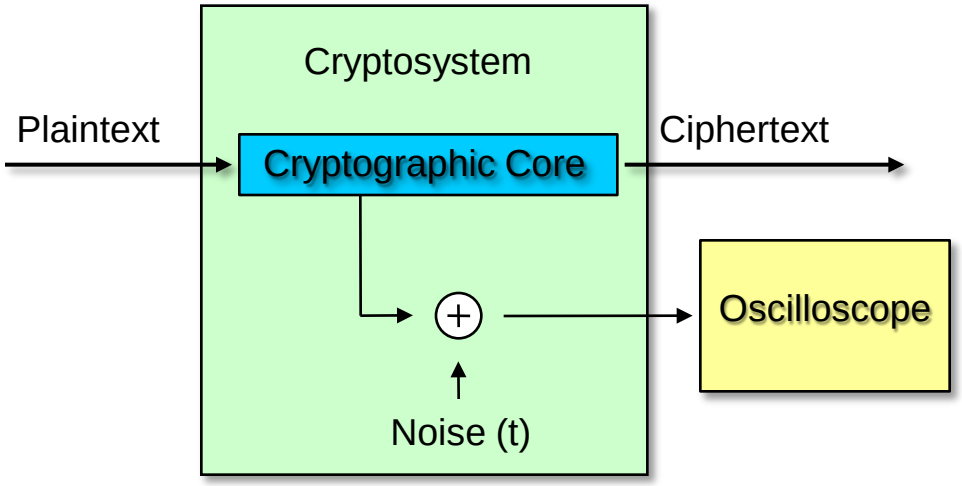
Additive White Gaussian Noise (AWGN) is a simple *channel model* in communication theory, which describes how the white noise adds up when the signal is passing through the channel.



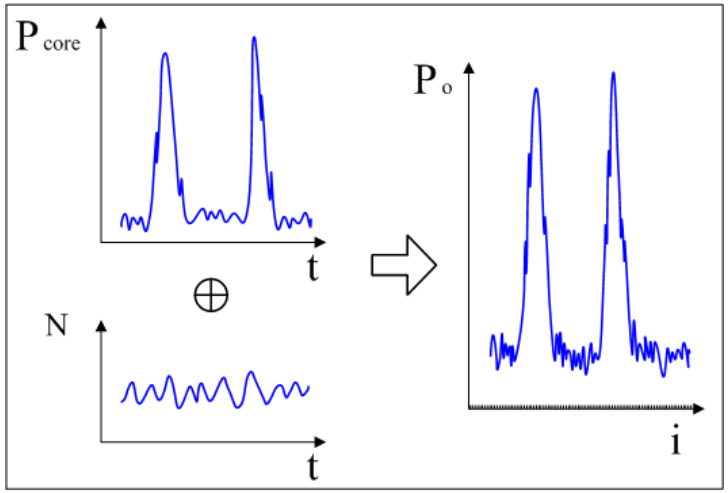
$$O[i] = S[i] + N[i] \qquad N \sim N(0, \sigma^2)$$

Source: Web

Power Amount Analysis (2)
Basic Power Model and Resulting Traces



AWGN based Hardware Model

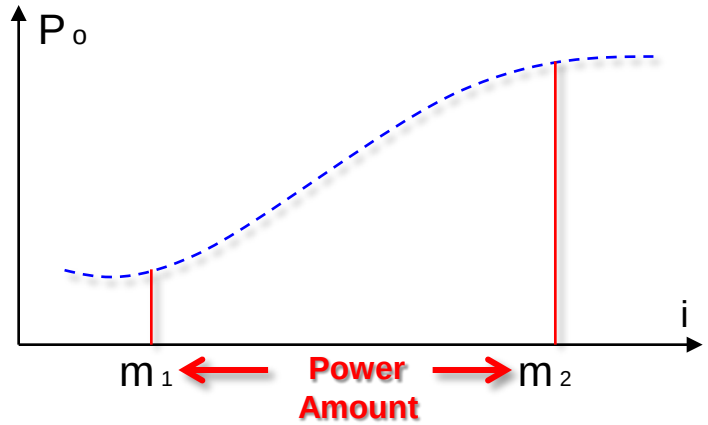


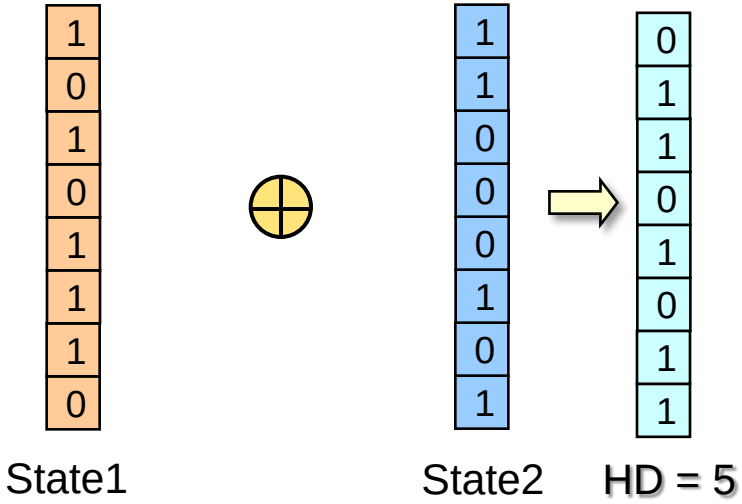
Traces

$$P_o = P_{core}[i] + P_N[i]$$

For any trace $N \sim N(0, \sigma^2)$ holds in time domain

Power Amount Analysis (3)
Properties of Power Model





$$\overline{P_{avg}} = \frac{1}{m_2 - m_1}(P_o[m_1] + \cdots + P_o[m_2]) = \overline{P_o[i]}$$

With: N = P_N

$$E(P_{avg}) = E(P_o) = E(P_{core}) + E(N) = E(P_{core})$$

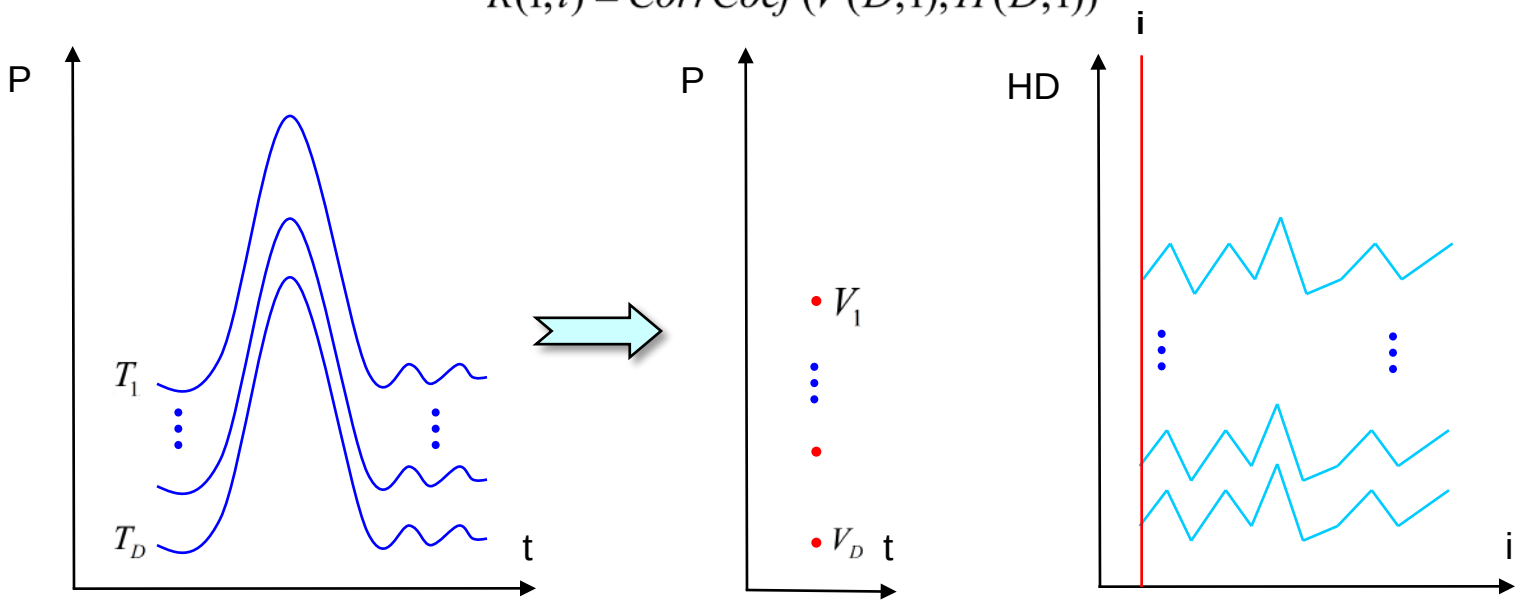
$$Var(P_{avg}) = Var(P_o) = Var(P_{core}) + Var(N) = Var(P_{core}) + \sigma^2$$

$$Dev(P_{avg}) = \sqrt{Var(P_{core}) + \sigma^2}$$

Power Amount Analysis (4)
Statistical Calculations

$$Var(T) = \begin{pmatrix} Var(T_{1,1}) & \cdots & T_{1,M}) \\ & \vdots & \\ Var(T_{D,1}) & \cdots & T_{D,M}) \end{pmatrix} = \begin{pmatrix} V_{1,1} \\ \vdots \\ V_{D,1} \end{pmatrix} \quad \begin{pmatrix} R_{1,1} & \cdots & R_{1,K} \end{pmatrix} = StatAnalysis\left(\begin{pmatrix} V_{1,1} \\ \vdots \\ V_{D,1} \end{pmatrix}, \begin{pmatrix} H_{1,1} & \cdots & H_{1,K} \\ \vdots & \ddots & \vdots \\ H_{D,1} & \cdots & H_{D,K} \end{pmatrix} \right)$$

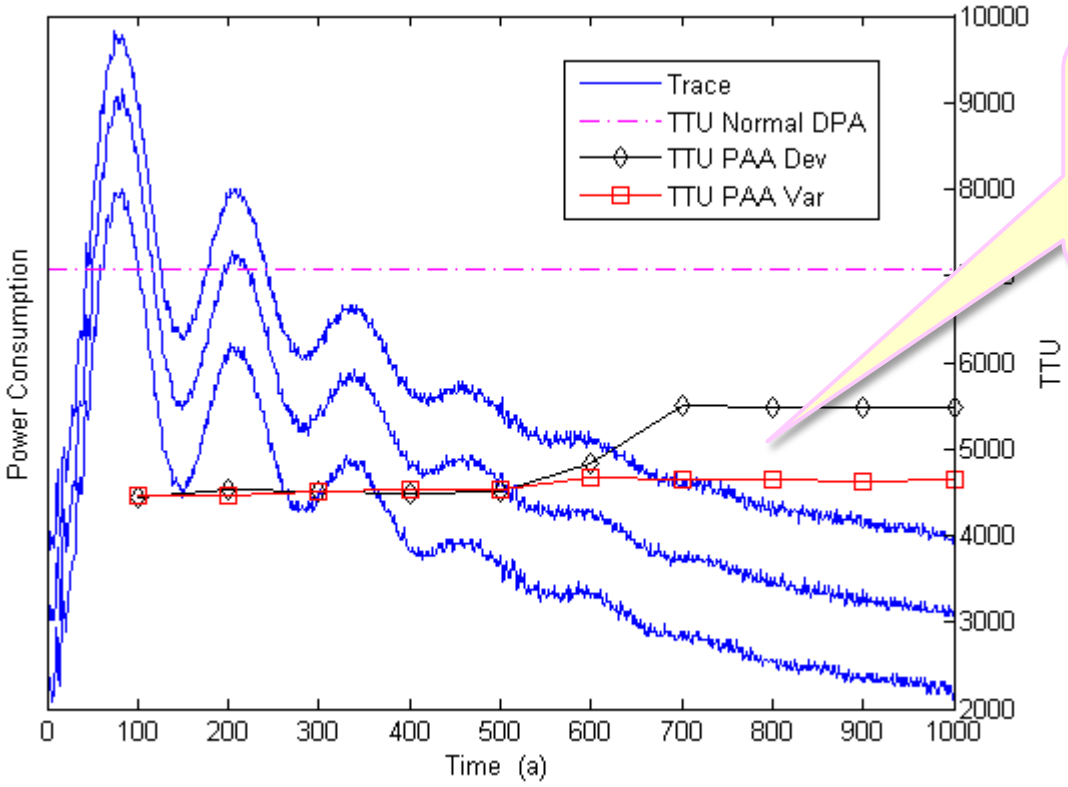
$$R(1,i) = CorrCoef(V(D,1), H(D,1))$$



Compared to DPA the calculation complexity of PAA is quite low.

AES-128: Experimental Results (1)

Total Trace Usage TTU



No. of points within PAA time interval: 100 (100) 1.000

AES-128: Experimental Results (2)

Summary

Property	DPA	PAA	
Total Trace Usage	7.000	4.800 – 5.500	
Total Attack Time	247s	Var	142s
		Dev	146s

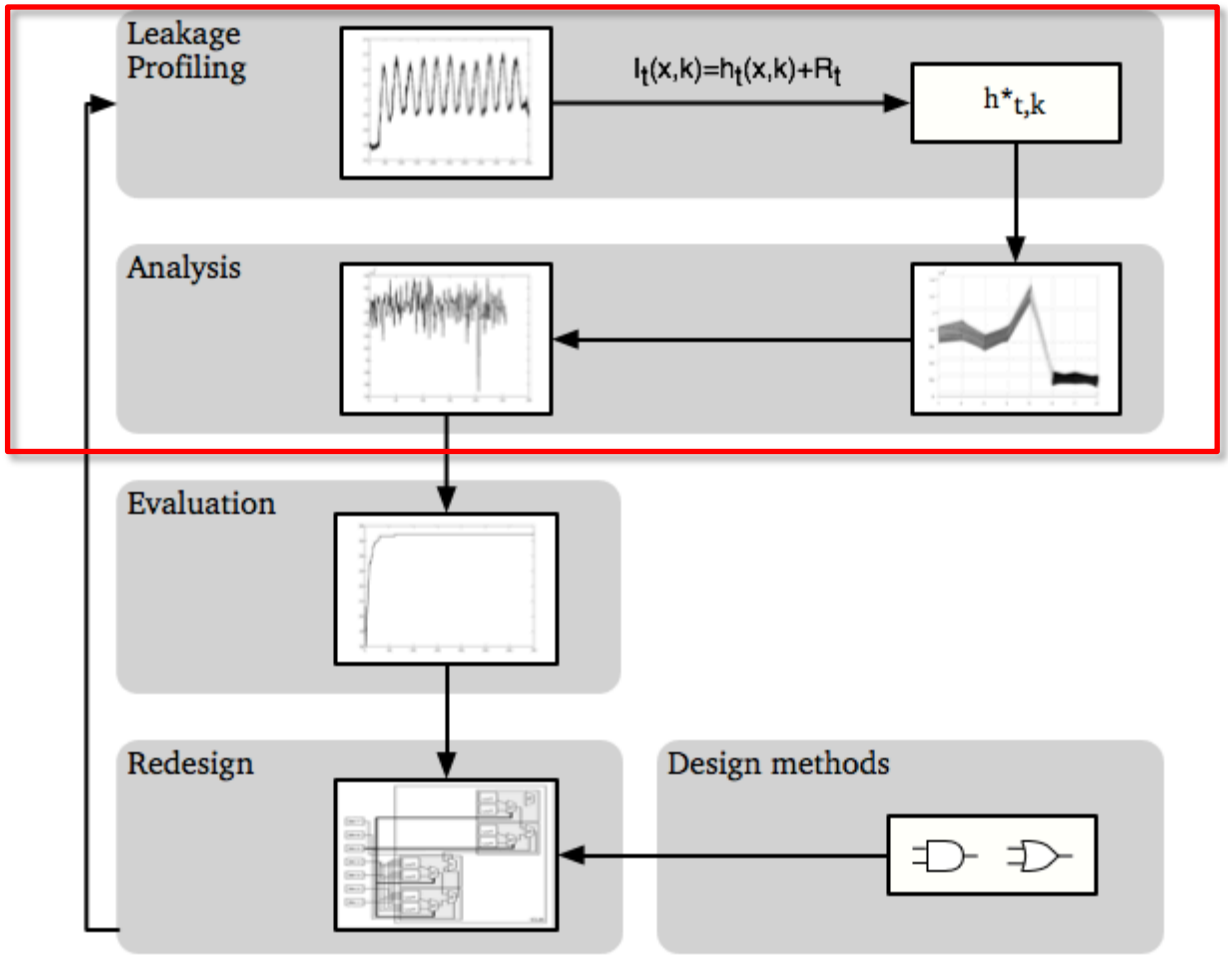
Bottom Line:

- PAA requires both considerably less computation time and power traces than DPA to mount a successful attack.
- PAA features a significant trace misalignment tolerance.
- However, PAA does not identify leakage sources…

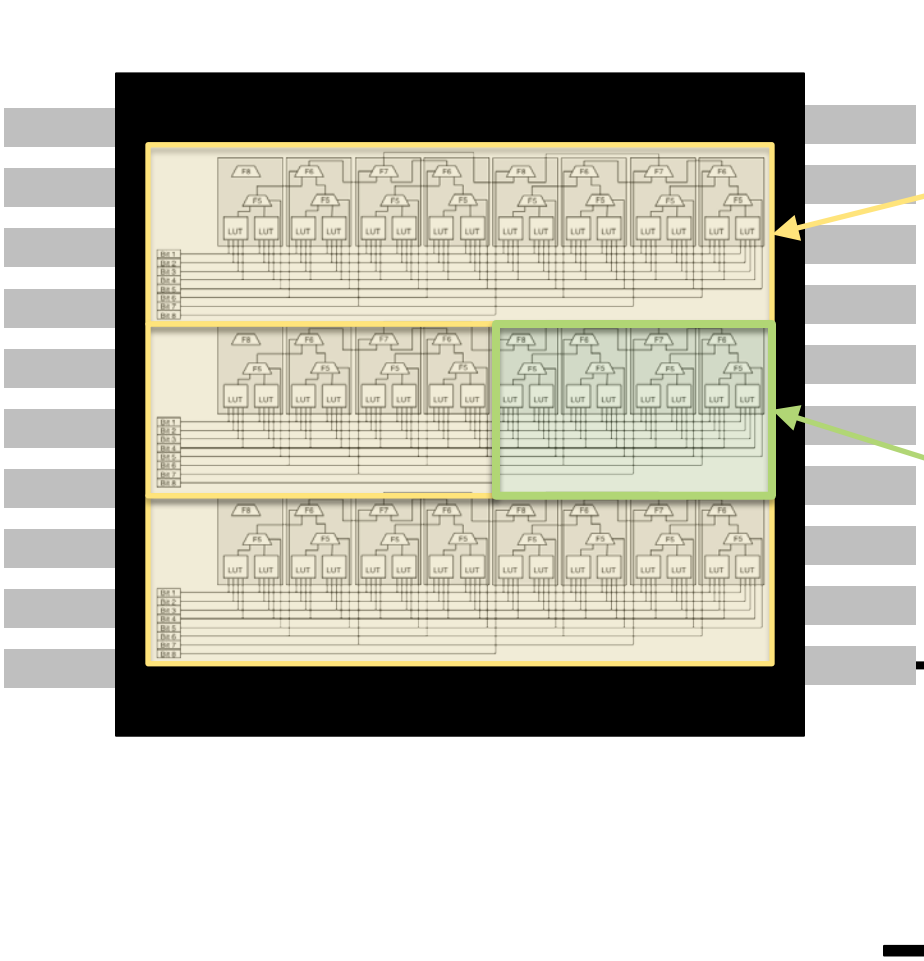
Overview

- Introduction
  - Motivation
  - Generic Attack Scenario
- Power Amount Analysis
  - Outline of DPA
  - AWGN Channel Model
  - Statistical Calculations
  - Analysis Results
- Constructive Methods
  - Localization of Leakage
  - Dynamically Mutating Processing Units
  - Virtualization in Multicore HW-Modules
- Summary

General Workflow of Constructive Side Channel Analysis



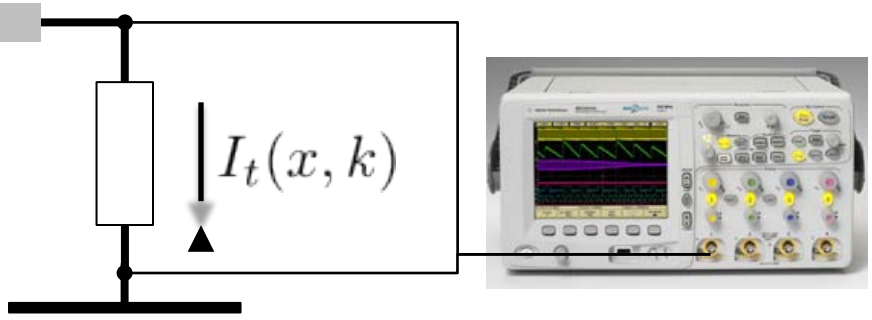
Stochastic Approach (1)
Basic Model



Noise
(normal distribution)

$$I_t(x,k) = h_t(x,k) + R_t$$

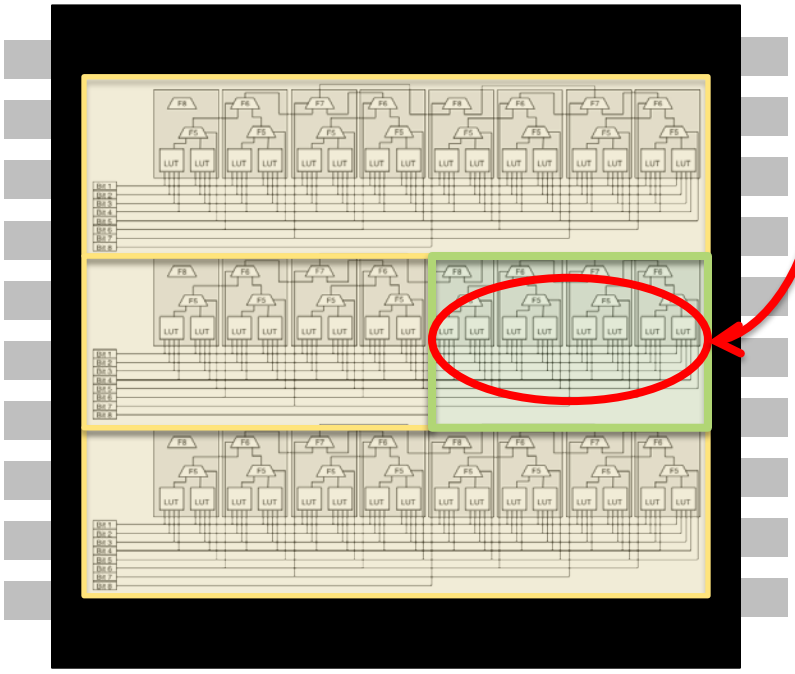
Deterministic Part
(x: message, k: key)



Stochastic Approach (2)

Deterministic Part $h_t(x, k)$

CASED



$h_t(x, k)$ represents the physical leakage

$h_t(\cdot, k)$ is unknown, but by $\widetilde{h}_t^*(\cdot, k)$ it may be approximated from an initial training phase

Hypotheses

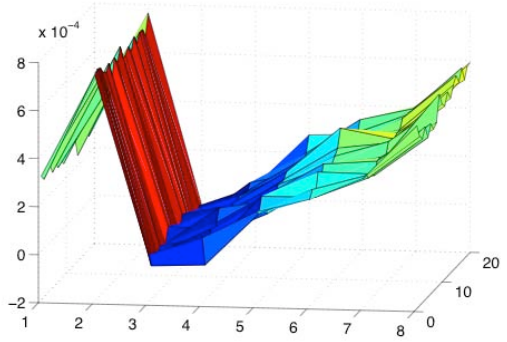
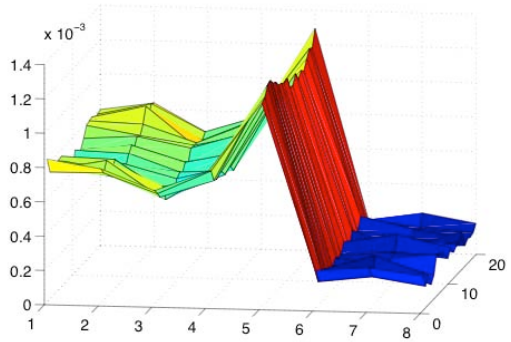
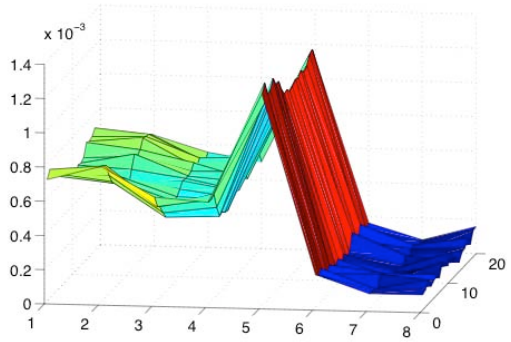
$$\widetilde{h}_t^*(\cdot, k) = \sum_{j=0}^{u-1} \widetilde{\beta}_{j,t;k} \cdot g_{j,t;k}(\cdot, k)$$

Weighting coefficients to model the impact of the basis function on the leakage

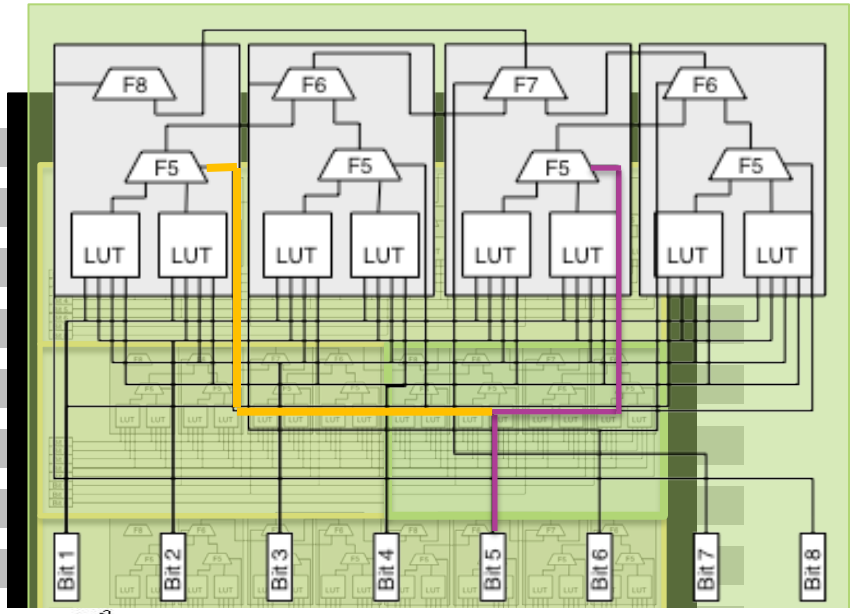
Example (1)
Leakage of Different AES Implementations

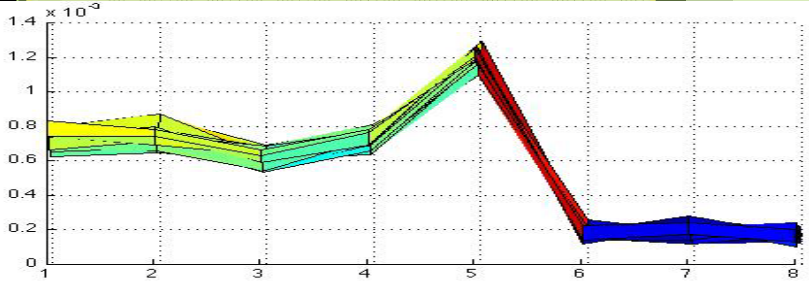
CASED

Experiments from SASEBO platform

AES-128 COMP	AES-128 TBL	
		$Key_{(1)}=19$
Beta Characteristic ($\|\beta_1\|,...,\|\beta_8\|$) X-Axis: Leakage Y-Axis: Bit Lanes Z-Axis: Time Instances ($t_1,..,t_{20}$)		$Key_{(1)}=208$

Example (2)
Reason for Characteristic Leakage in TBL





Coefficients per Bit Line

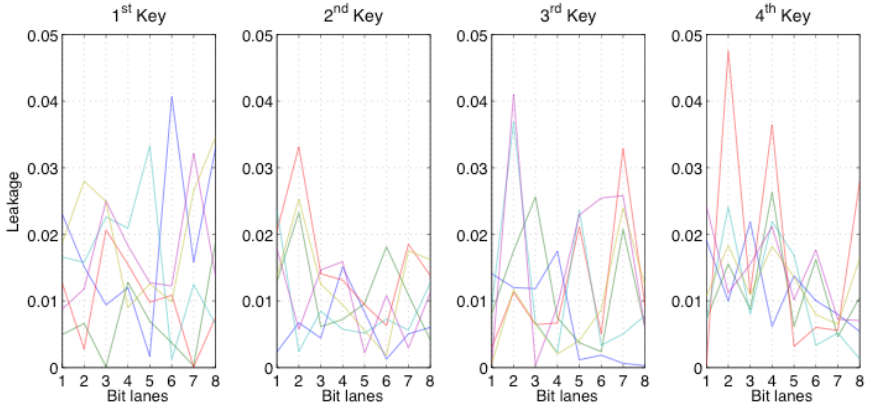
- Different propagation delays between some components may cause **data dependent glitches**
- The Stochastic Approach can be used as a **tool** in order to support the development of secure designs by **identifying leakage** sources

Stochastic Approach (4)

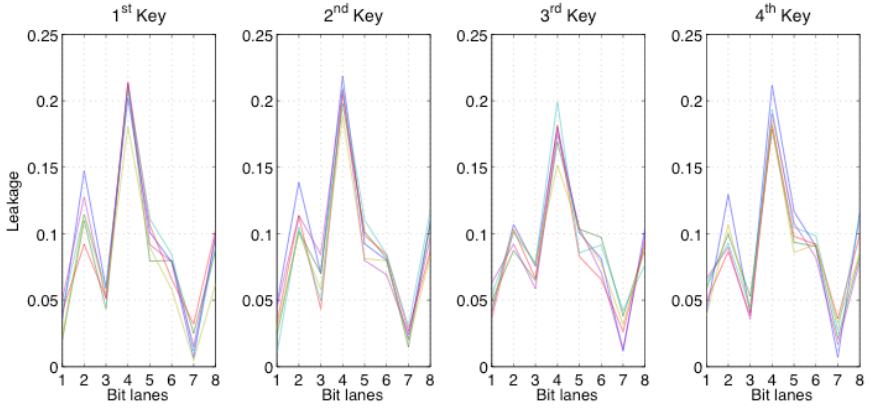
Symmetry Properties

- Weight coefficients may be used to **identify** SCA design flaws
- Implementation issues are a **deterministic** and independent of the sub key value
- **Inappropriate** models may lead to sub key value dependent weight coefficients
- Checking the **suitability** of the model in the profiling phase is mandatory

Model A:

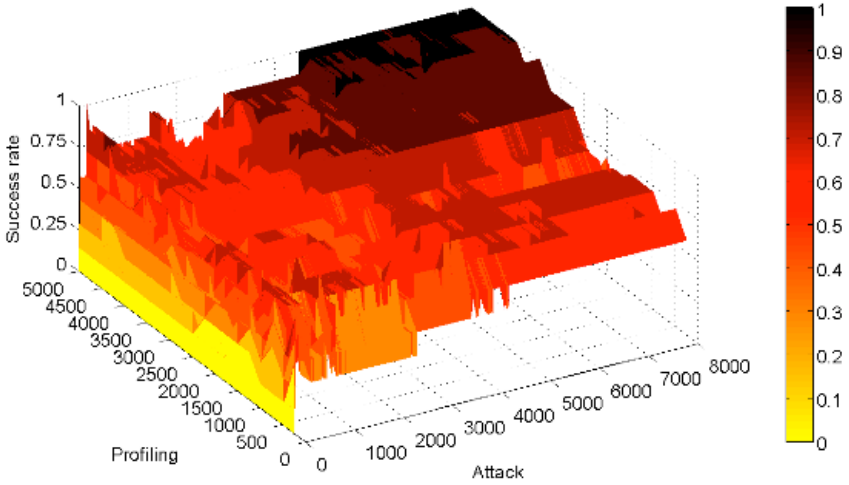


Model B:

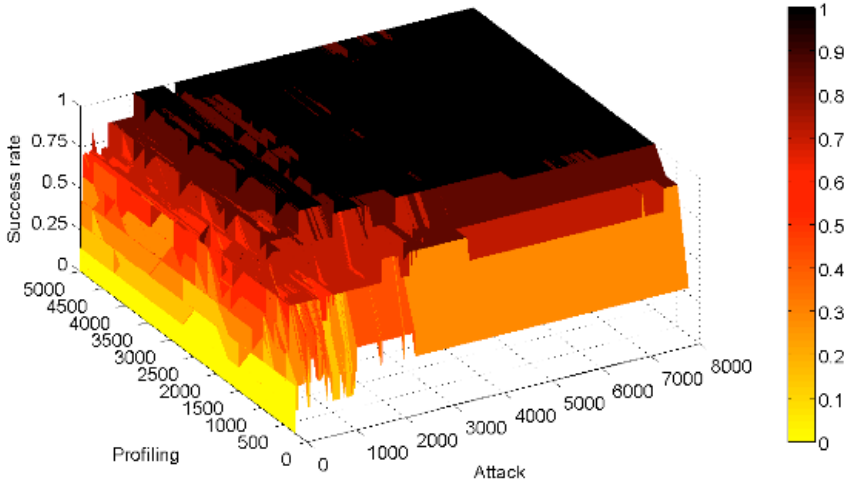


Stochastic Approach (5)
Results on the Impact of Symmetries

Attack on one set of power traces with two different models and variable amounts of traces in the profiling phase

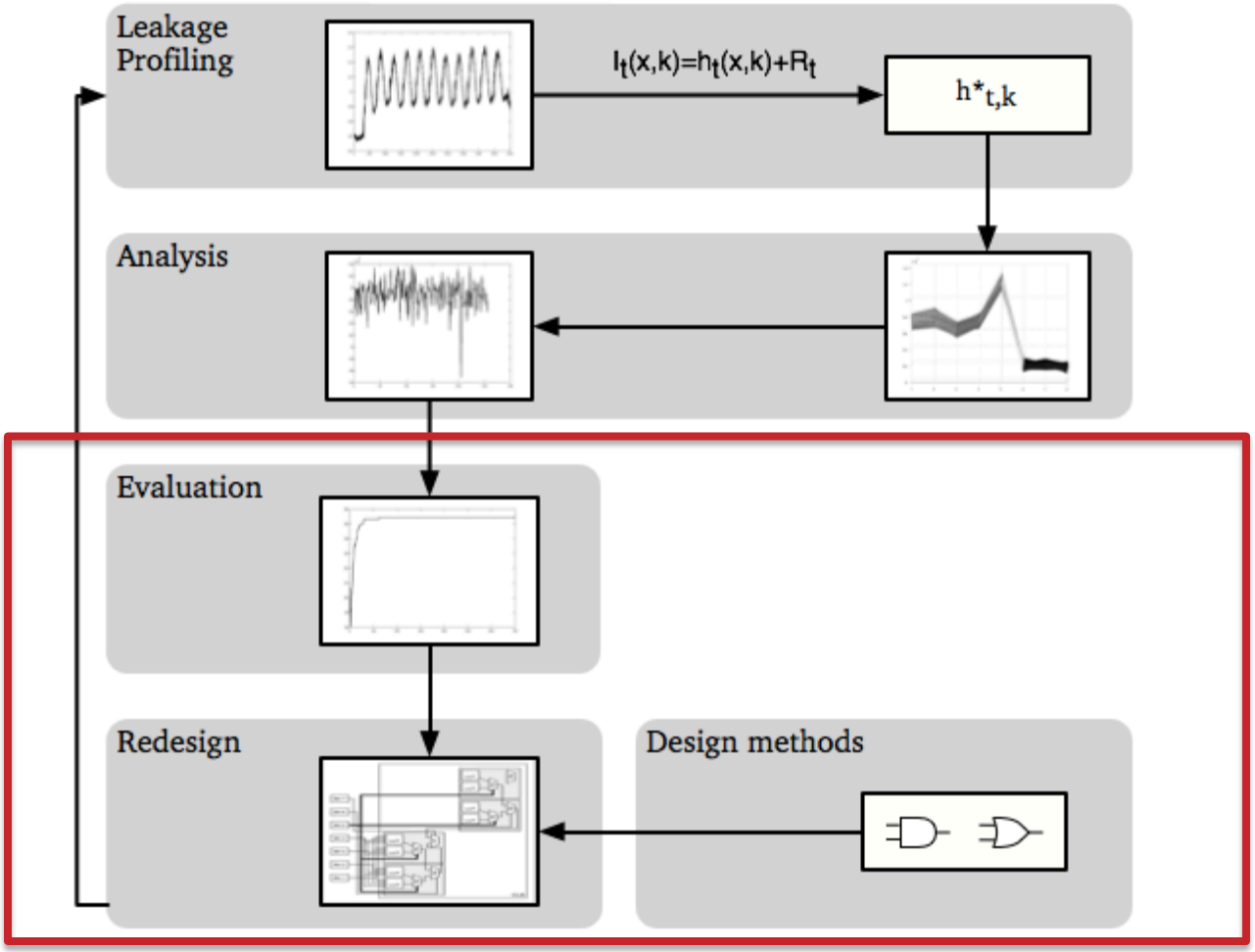


Success Rate with Model A

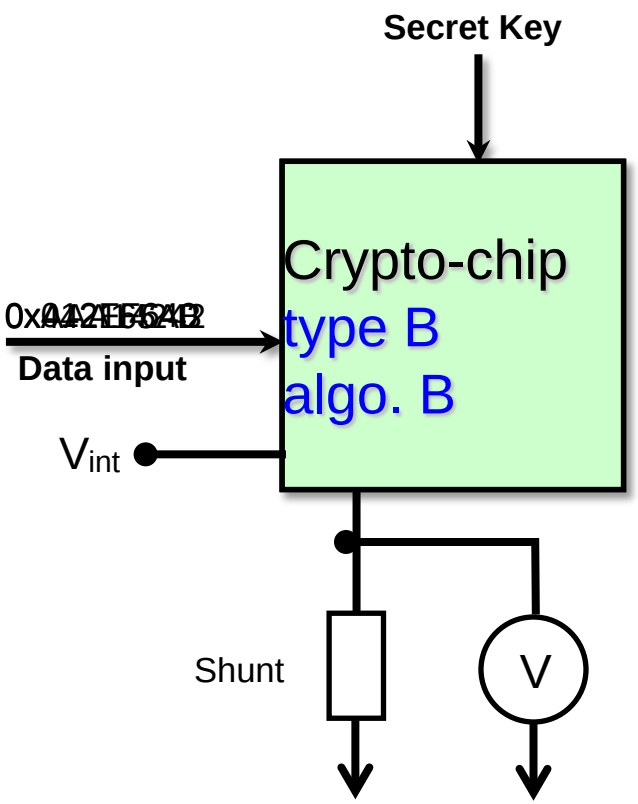


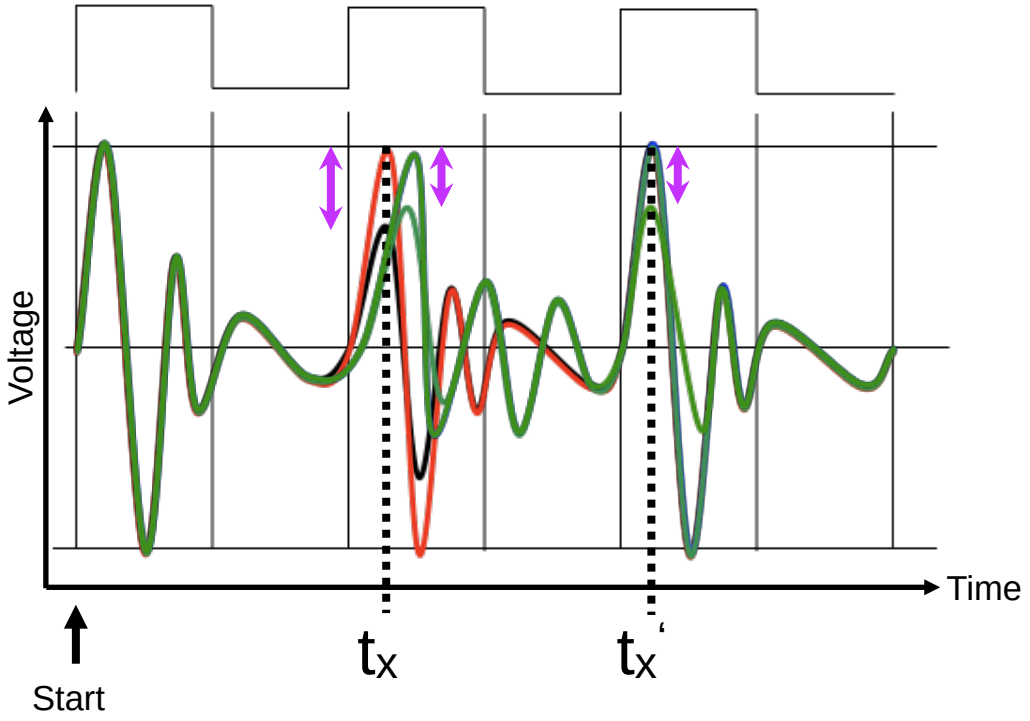
Success Rate with Model B

Countermeasures



Dynamically Mutating Processing Units (1)
Basic Phenomena

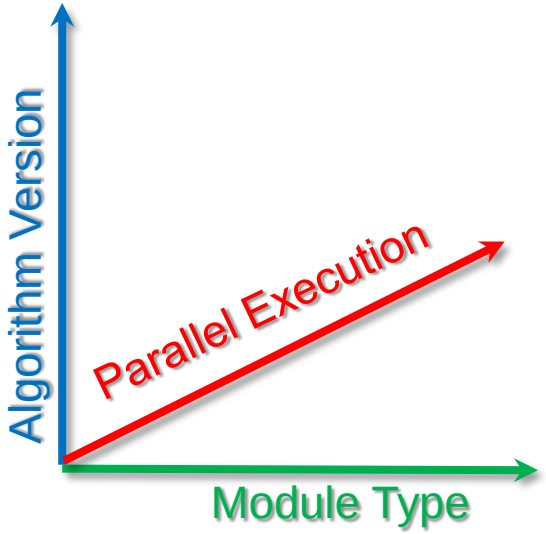




Dynamically Mutating Processing Units (2)
Detailed Concept

Many-dimensional design space for processing units

- Type, i.e., circuit variant of dedicated processing unit
- Parallelization degree within a distinct unit
- Algorithm version for envisaged functionality

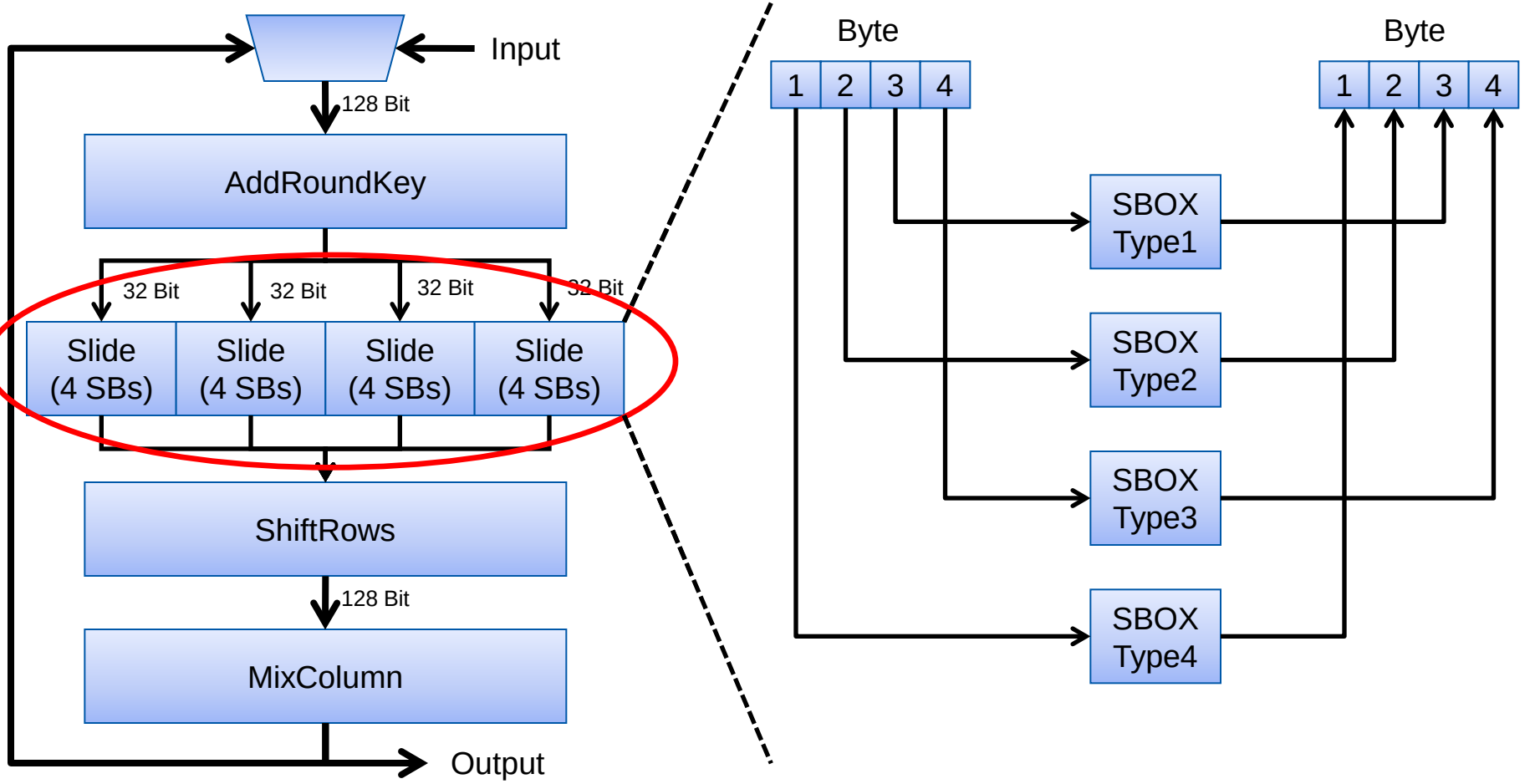


Dynamically Mutating Processing Units
In addition to known masking and hiding methods

Properties

- Inner state not observable during cryptographic processing
- Hiding concepts on different layers to gain maximum resistance effects
- Randomly selected processing unit instance at runtime

Module Type (1)
AES SubBytes

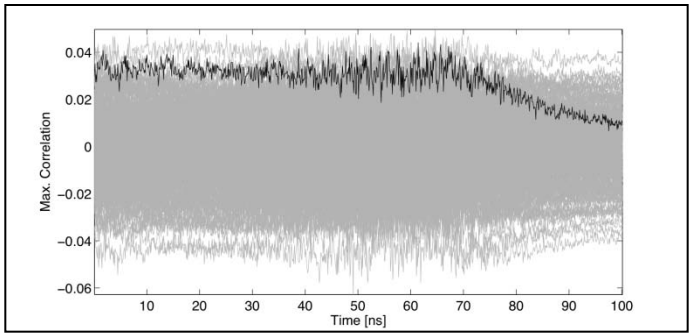


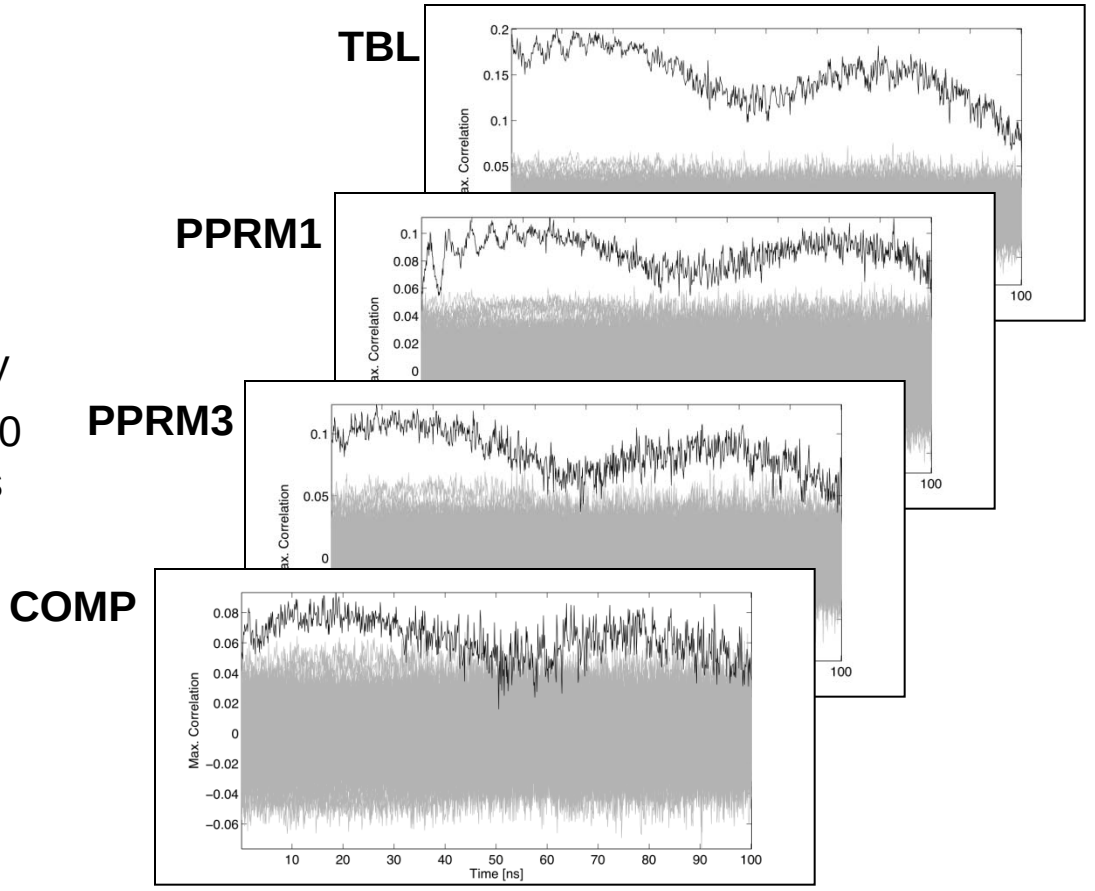
Module Type (2)
DPA Resistance of SBox Groups

CASED

Mutated SBox group:

- Dynamically mutated, heterogenous architecture of grouped SBoxes
- Selection of bytes and SBoxes by a random number
- Additional costs by switching logic only
- More resistant to DPA attack with 5.000 traces than homogenous architectures



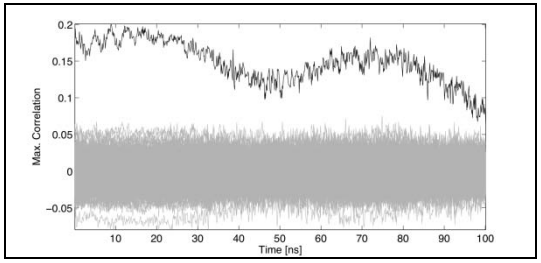


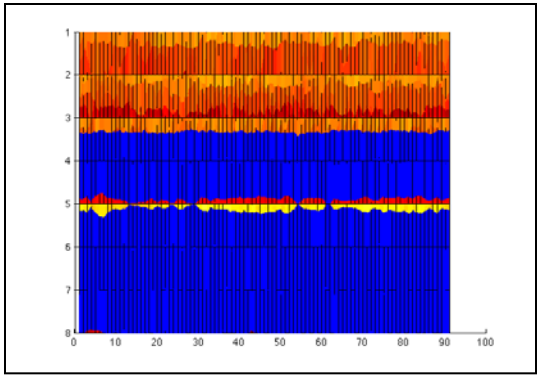
TECHNISCHE UNIVERSITÄT DARMSTADT
Fraunhofer SIT
h.da HOCHSCHULE DARMSTADT UNIVERSITY OF APPLIED SCIENCES
LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz

Module Type (3)
Leakage of Module Instances

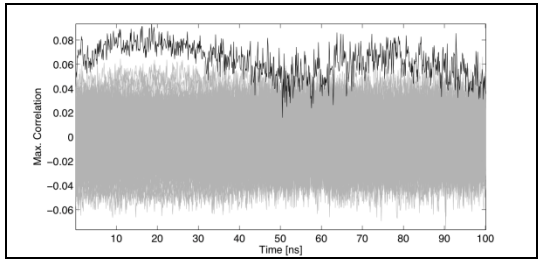
CASED

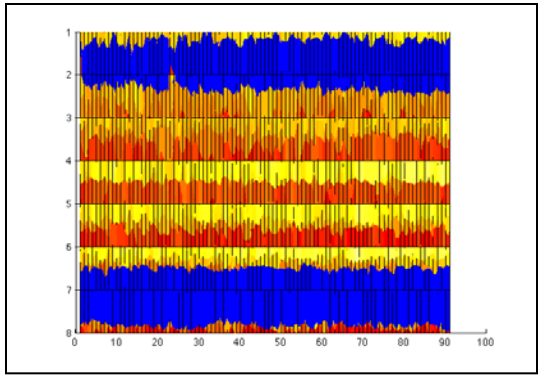
AES TBL



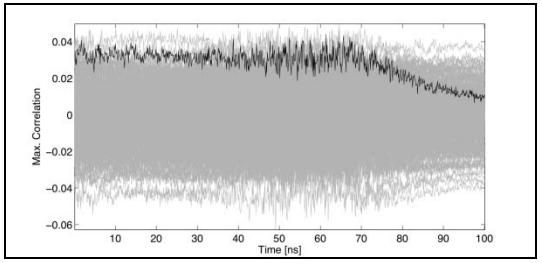


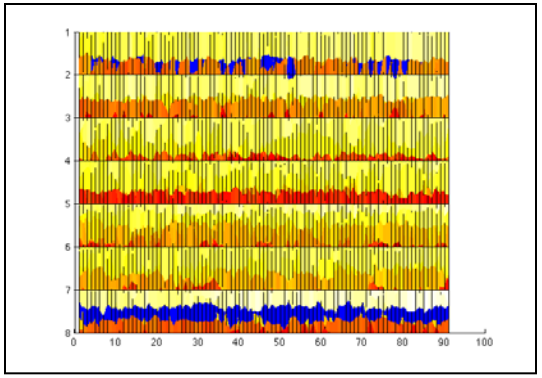
AES COMP





AES Mutated

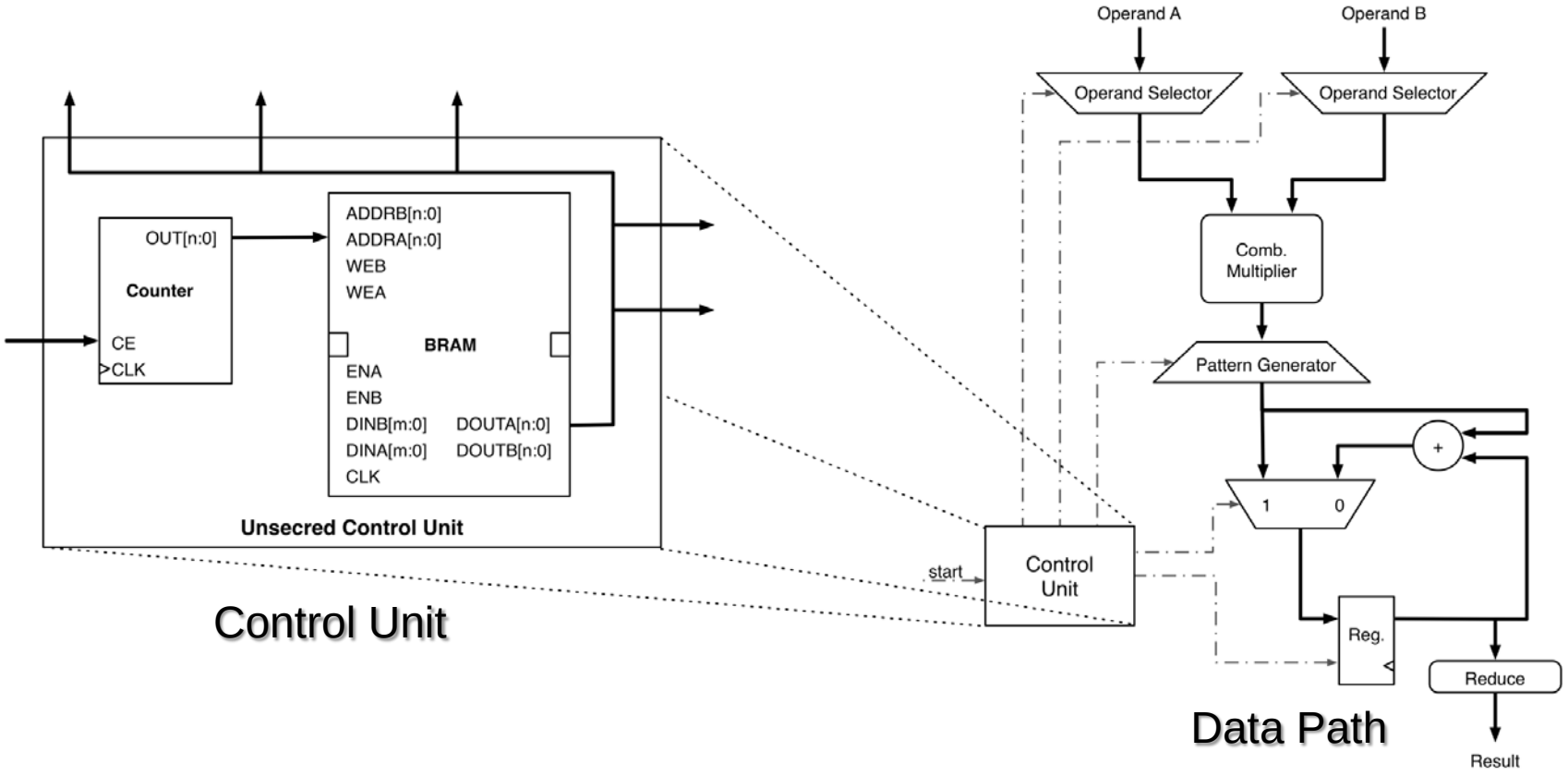




Leakage
(blue areas in Leakage Monitor)

Nearly no leakage

Parallization Degree (1)
Architecture of eMSK Multiplier



eMSK: enhanced Multi-Segment-Karatsuba

Parallelization Degree (2)

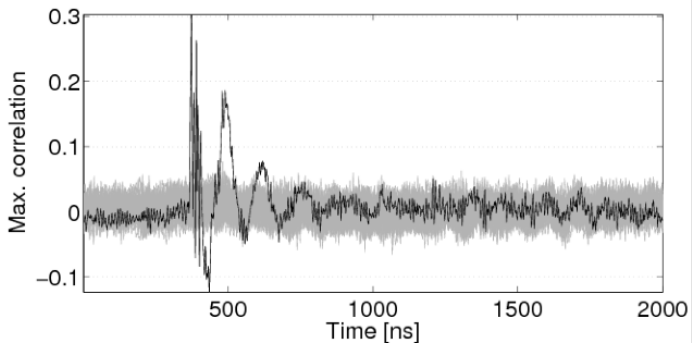
SC Resistance of $eMSK_{24}$ 192 Bit MULT

CASED

Power Analysis without Countermeasure:

$$P(t_i) = P_{process}(t_i) + P_{noise}(t_i)$$

$$Var(P(t_i)) = Var(P_{data}(t_i)) + \underbrace{Var(P_{Op}(t_i))}_{Var(P_{Op}(t_i)) \to 0} + \sigma$$



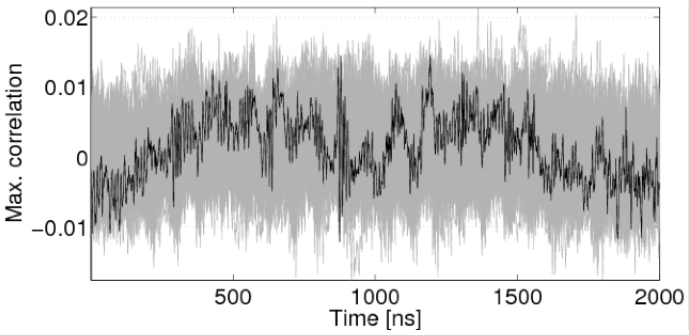
DPA result: 500 traces on unsecured eMSK

Power Analysis with Countermeasure:

$$Var(P(t_i)) = Var(P_{data}(t'_i)) + Var(P_{Op}(t'_i)) + \sigma \quad \text{Time}$$

$$r_i = \bigoplus_{j=1}^{j=i-1} op(intermediate\ result(t_i, r_j)) \quad \text{Amplitude}$$

$$Var(P(t_i)) = Var(P_{data}(t'_0, ..., t'_i)) + Var(P_{Op}(t'_i)) + \sigma$$



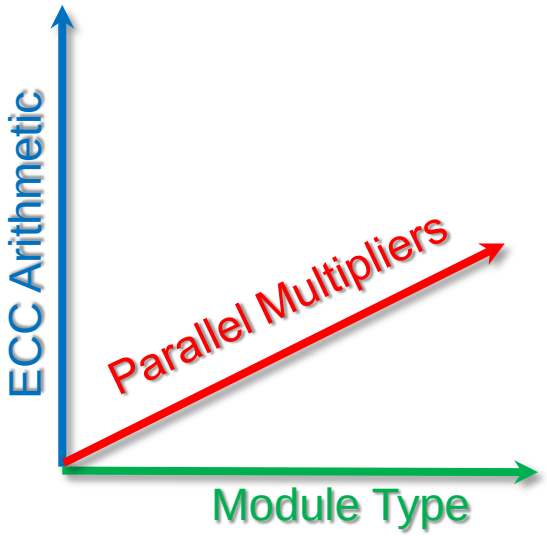
DPA result: 50.000 traces on secured eMSK

Dynamically Mutating Processing Units (3)
Application to ECC (I)

CASED

Information Leakage of ECC Implementation

- Finite Field Multiplication
  - With respect to the application
- Point-addition and -double Operation
  - Runtime dependency based on parallelism
- ECC Arithmetic
  - Point-addtion and -double algorithm
  - Point-multiplication algorithm

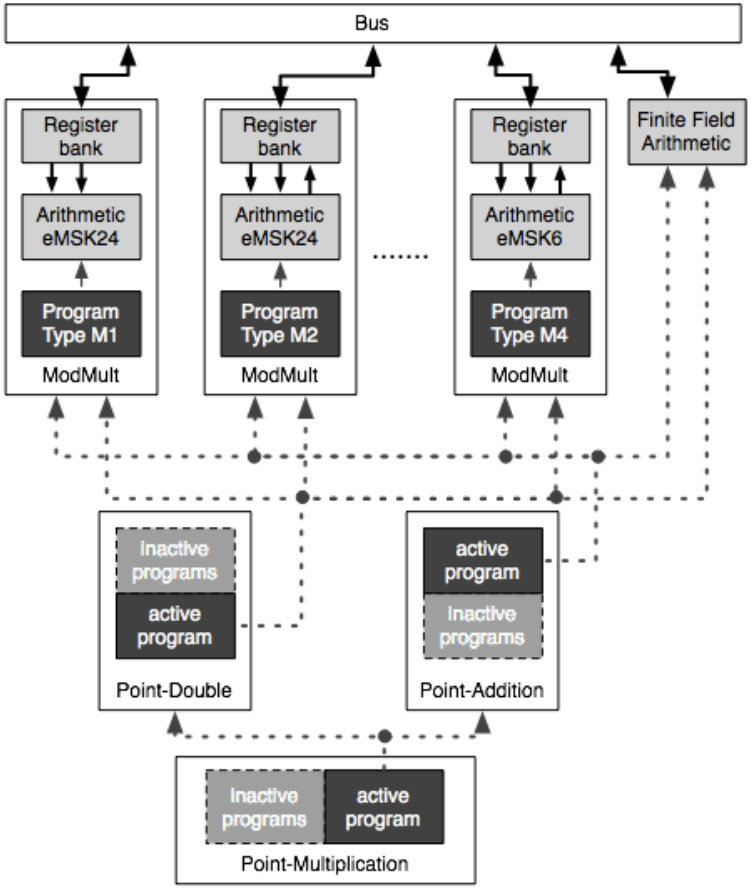


Dynamically Mutating Processing Units (4)
Application to ECC (II)

CASED

Focus on Second Axis of Design Space
Parallelism

- Heterogenous, parallel multiplier scheme
- Different power signature of ECC arithmetic
- Different point-multiplication algorithms
- Virtualization methods in hardware
- Architectures with dynamical concurrency



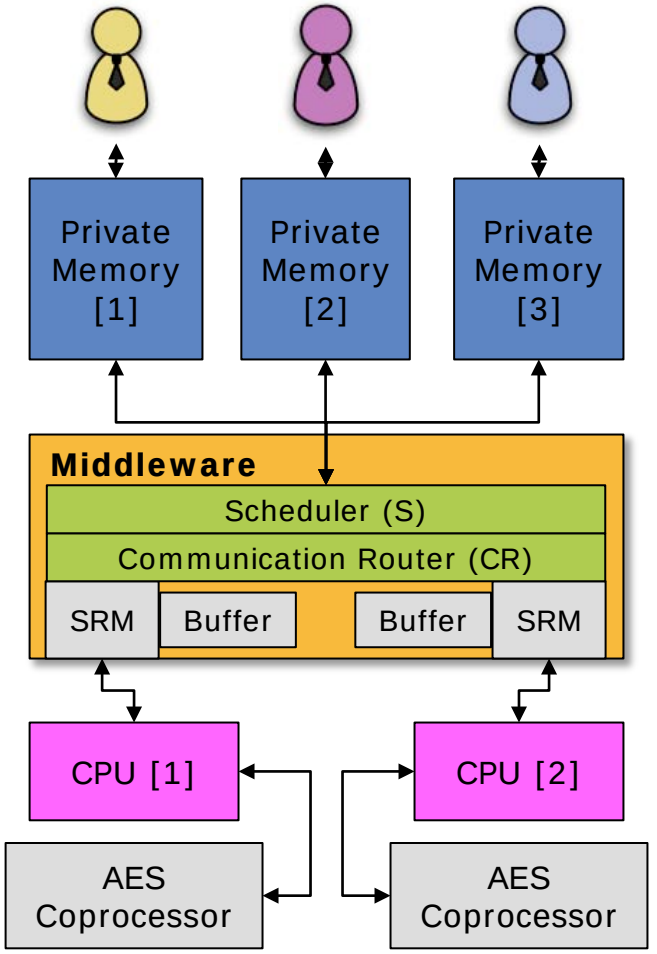
TECHNISCHE UNIVERSITÄT DARMSTADT
Fraunhofer SIT
h.da HOCHSCHULE DARMSTADT UNIVERSITY OF APPLIED SCIENCES
LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz

Virtualization (1)
Memory with Integrity (I)

CASED

Private Memory Space

- Workload balancing with respect to the integrity of each user space
- Status register manager stores and recovers the context switching data
- Intermediate state information and data of each CPU can not be transferred to another user space

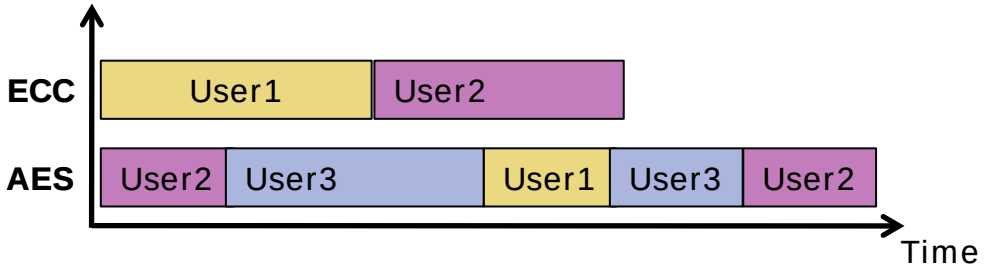


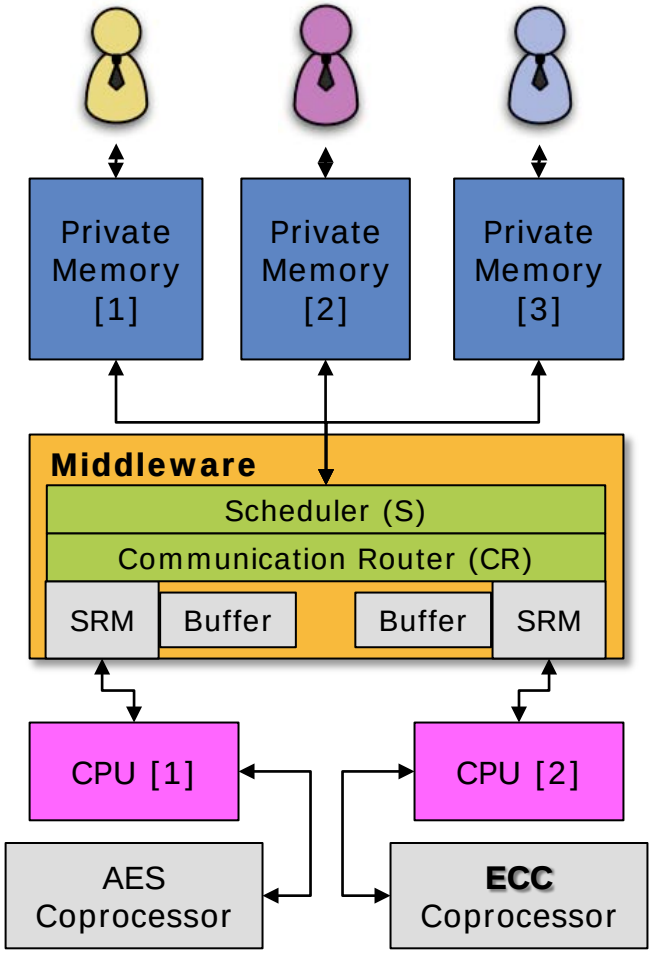
Virtualization (2)
Memory with Integrity (II)

CASED

Private Memory Space

- Middleware encapsulates the processing units from the memory
- Different users can share resources - without losing the integrity of their data



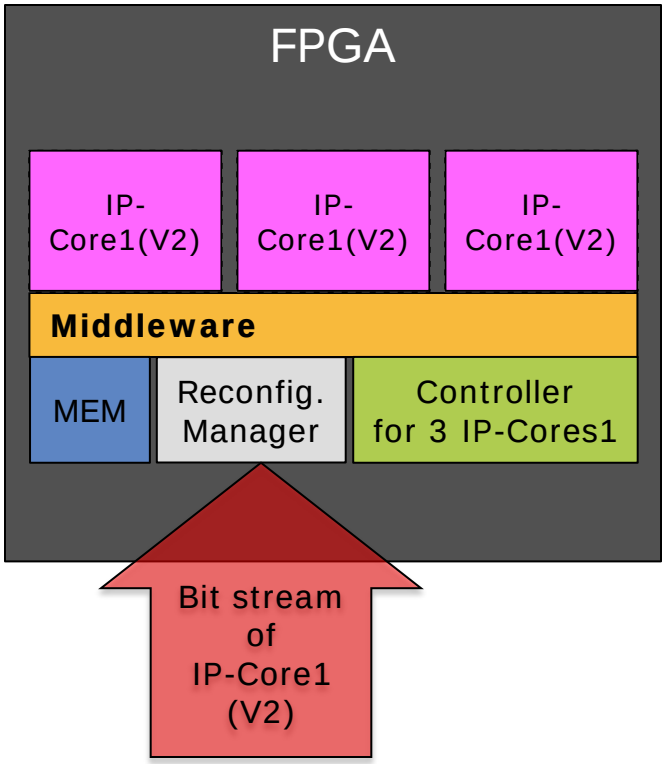


Virtualization (3)
Runtime System Reconfiguration

CASED

Reconfiguration Architecture

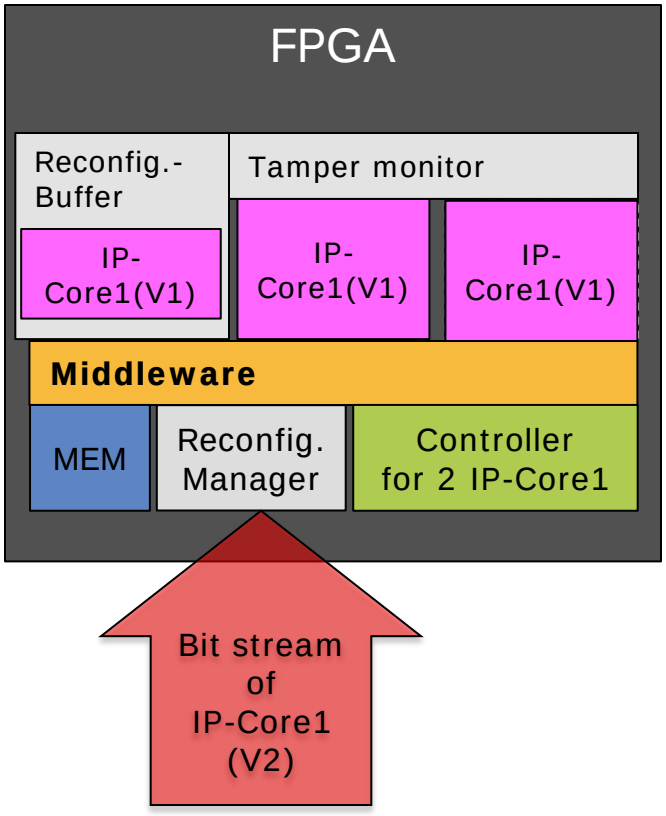
- Reconfiguration without stopping current running processes
- Abstraction between control flow and data flow
- Hiding IP-Core configuration time (*up to 100 ms*) by workload balancing



Virtualization (4)
Tamper Resistance of FPGAs

Tamper Monitoring

- Detect malicious IP-Core reconfiguration during runtime - *without any loss of throughput*
- Self-healing of tampered IP-Core via reconfiguration
- During the healing process the middleware uses a not reconfigured IP-Core for work balancing

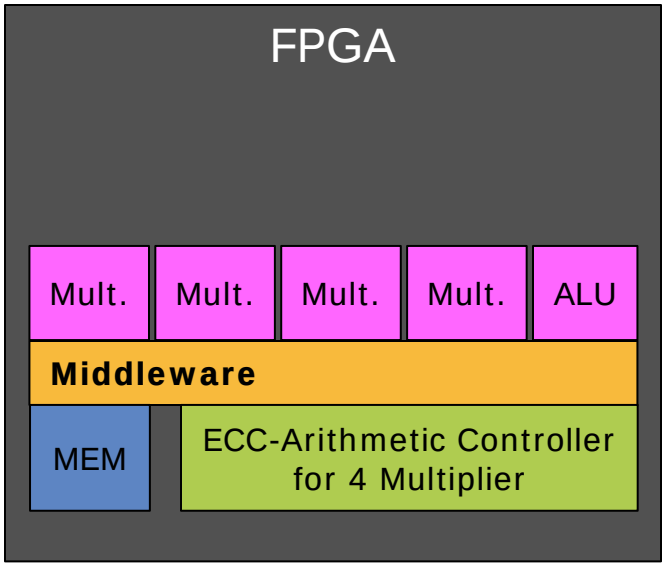


Virtualization (5)
Side Channel Resistance (I)

CASED

Concurrency

- Executes different control flows for point multiplication without changing the program code
- Data-independent runtime variation
- Different amounts of noise caused by varying parallelism of Finite Field multiplications

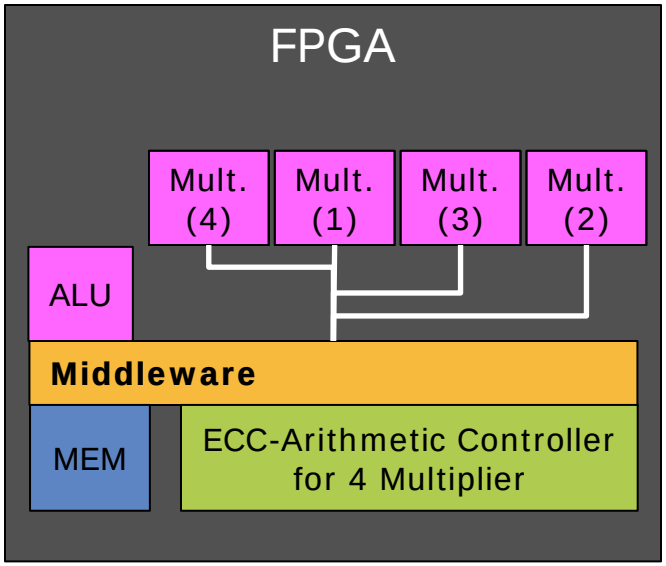


Virtualization (6)
Side Channel Resistance (II)

CASED

Topology

- Change of the binding of each multiplication process
- Effect on the propagation delay of each multiplication - *from execution to execution*
- Varying power consumption caused by different glitch situations



Overview

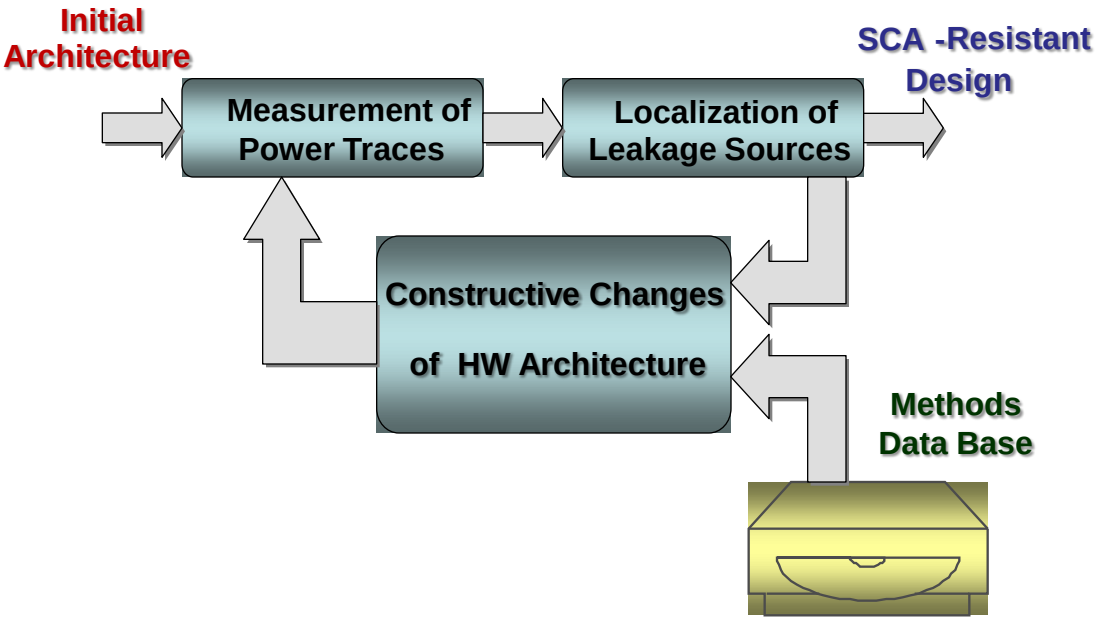
CASED

- Introduction
  - Motivation
  - Generic Attack Scenario
- Power Amount Analysis
  - Outline of DPA
  - AWGN Channel Model
  - Statistical Calculations
  - Analysis Results
- Constructive Methods
  - Localization of Leakage
  - Dynamically Mutating Processing Units
  - Virtualization in Multicore HW-Modules
- Summary

Conclusions

- Fast power trace analysis methods are mandatory for design space exploration purposes.
- Localization of leakage sources from power trace analysis is a prerequisite to SCA-aware module construction.
- The novel concept of dynamically mutating processing elements and virtualization techniques is a viable foundation to a new SCA-aware construction methodology.

Ongoing Work:
Automatic SCA-aware Hardware Synthesis



The Engineer‘s Destiny…