On the Synthesis of Side-Channel resistant Cryptographic Modules

TECHNISCHE UNIVERSITÄT DARMSTADT

Sorin A. Huss

A. Israr
K. Rohde
O. Stein
M. Stöttinger
M. Zohner

Integrated Circuits and Systems Lab
Computer Science Department
Technische Universität Darmstadt, Germany

huss@iss.tu-darmstadt.de

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Introduction
2. System Overview
3. Detection of SCA Vulnerabilities
4. Re-Synthesis of Hardware Modules
5. Application Example
6. Conclusions

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Methodology is the right way to significantly rise both productivity and reliability in the digital design domain:

- Modeling and simulation concepts form the foundation of a consistent design methodology
- Logic synthesis is since many years the successful standard approach to digital circuit design automation
- High-level synthesis additionaly boosts the productivity of logic synthesis

So, why not trying to conceive kind of *SCA-related* high-level synthesis?

Power Analysis Attack Methods

TECHNISCHE UNIVERSITÄT DARMSTADT

Type	Name	Comment
Non-profiling	Differential power analysis	First attack on intermediate values
	Correlation power analysis	Most popular attack method
	Mutual information analysis	Considers non-linearity
	Linear regression analysis	Utilizes linear regression
	Power analysis collision	Exploits internal collisions
	Power amount analysis	Considers time intervals
Profiling	Simple power analysis	First control flow attack
	Template attack	Utilizes power distribution
	Stochastic approach	Exploits subspace representation

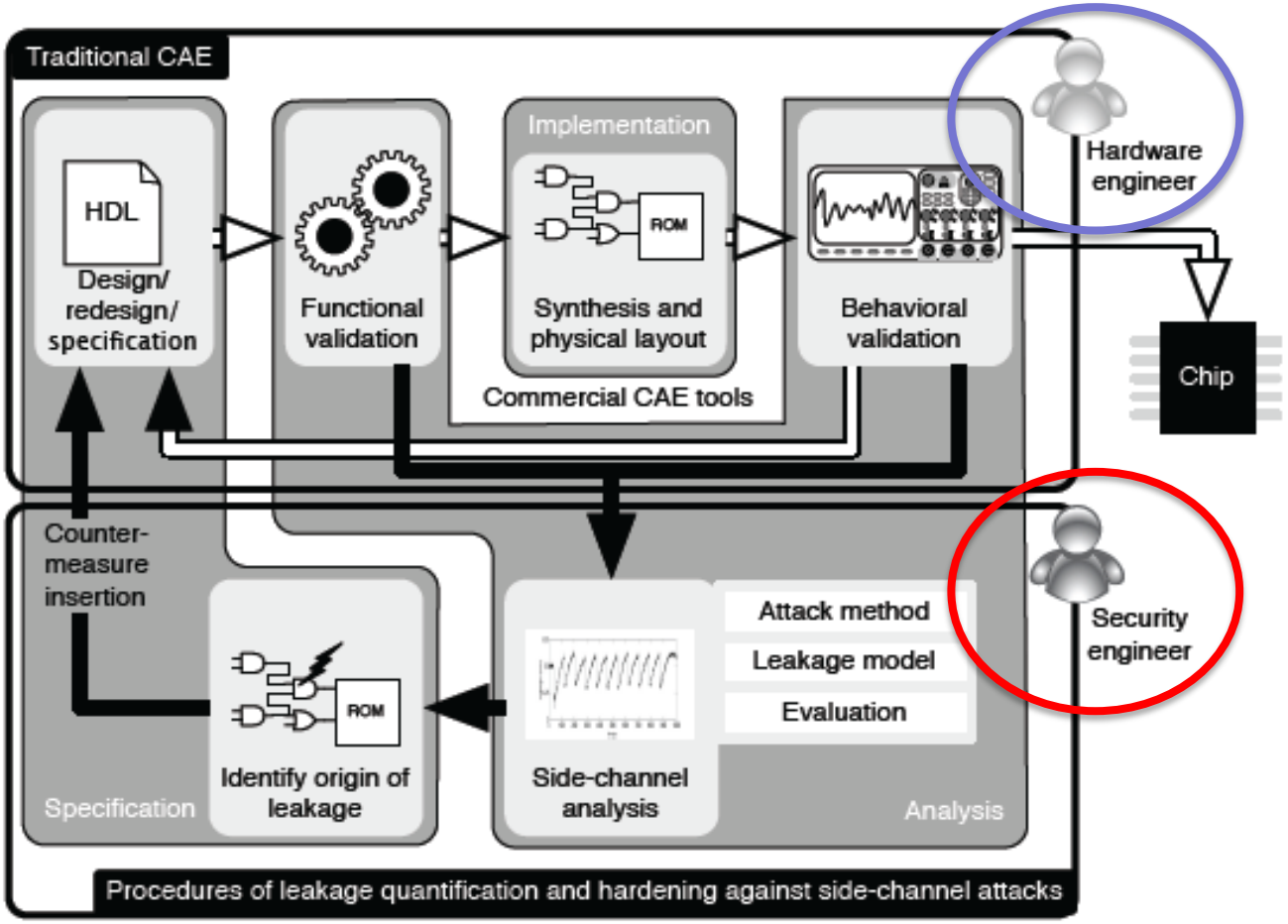
Well-known Countermeasures

TECHNISCHE UNIVERSITÄT DARMSTADT

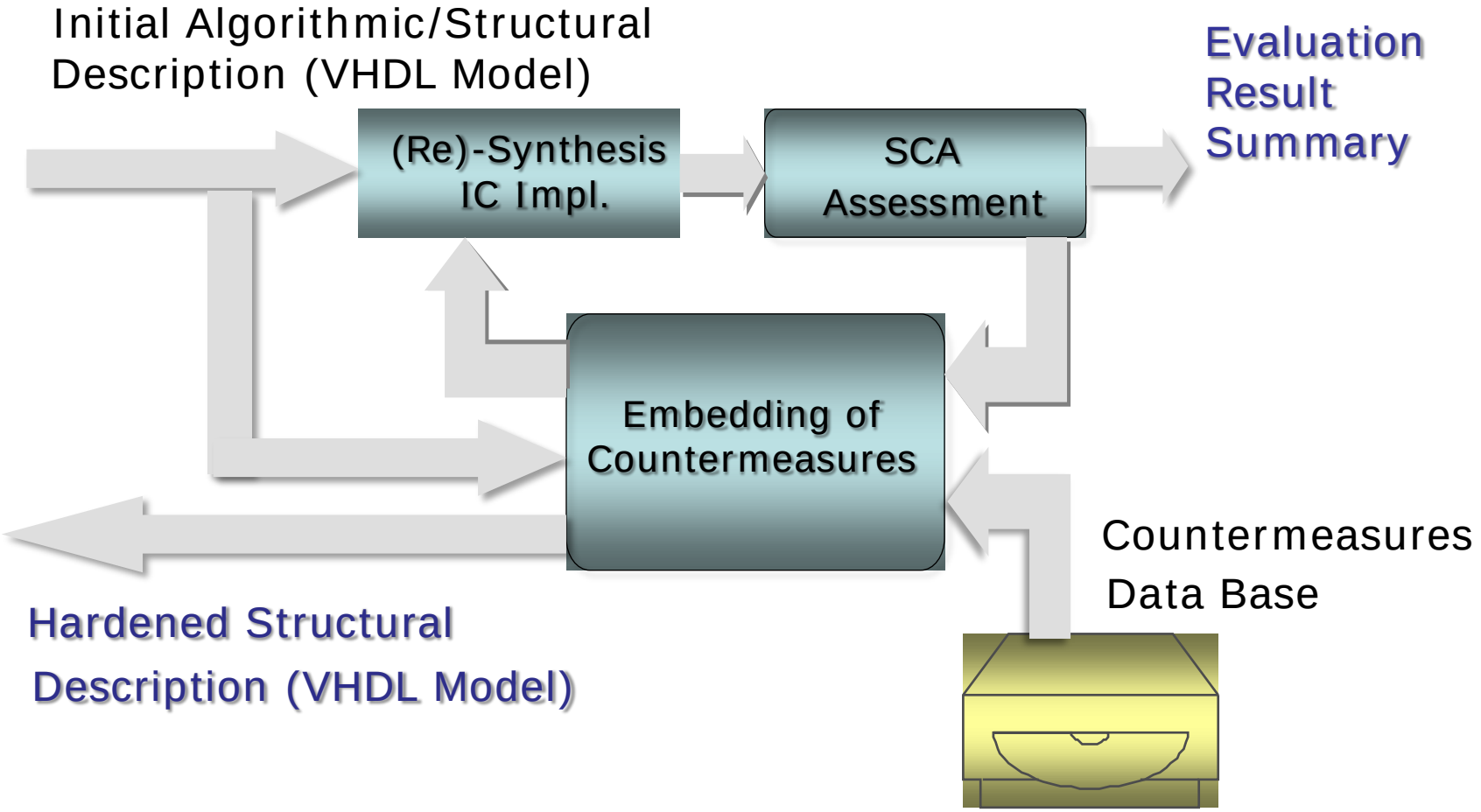
Type	Name	Level
Masking	Masked values	Algorithm
	Masking logic	Cell
	Masked Dual-Rail Pre-Charge Logic (MDPL)	Cell
	Threshold implementation	Algorithm
Hiding	Shuffling	Algorithm
	Random delay, clock	Cell
	Dual-Rail logic	Cell
	Current Load logic	Cell
	Noise, random switching	Cell
	Register pre-charging	Algorithm

Current Side-Channel Analysis aware Hardware Design Flow

TECHNISCHE UNIVERSITÄT DARMSTADT

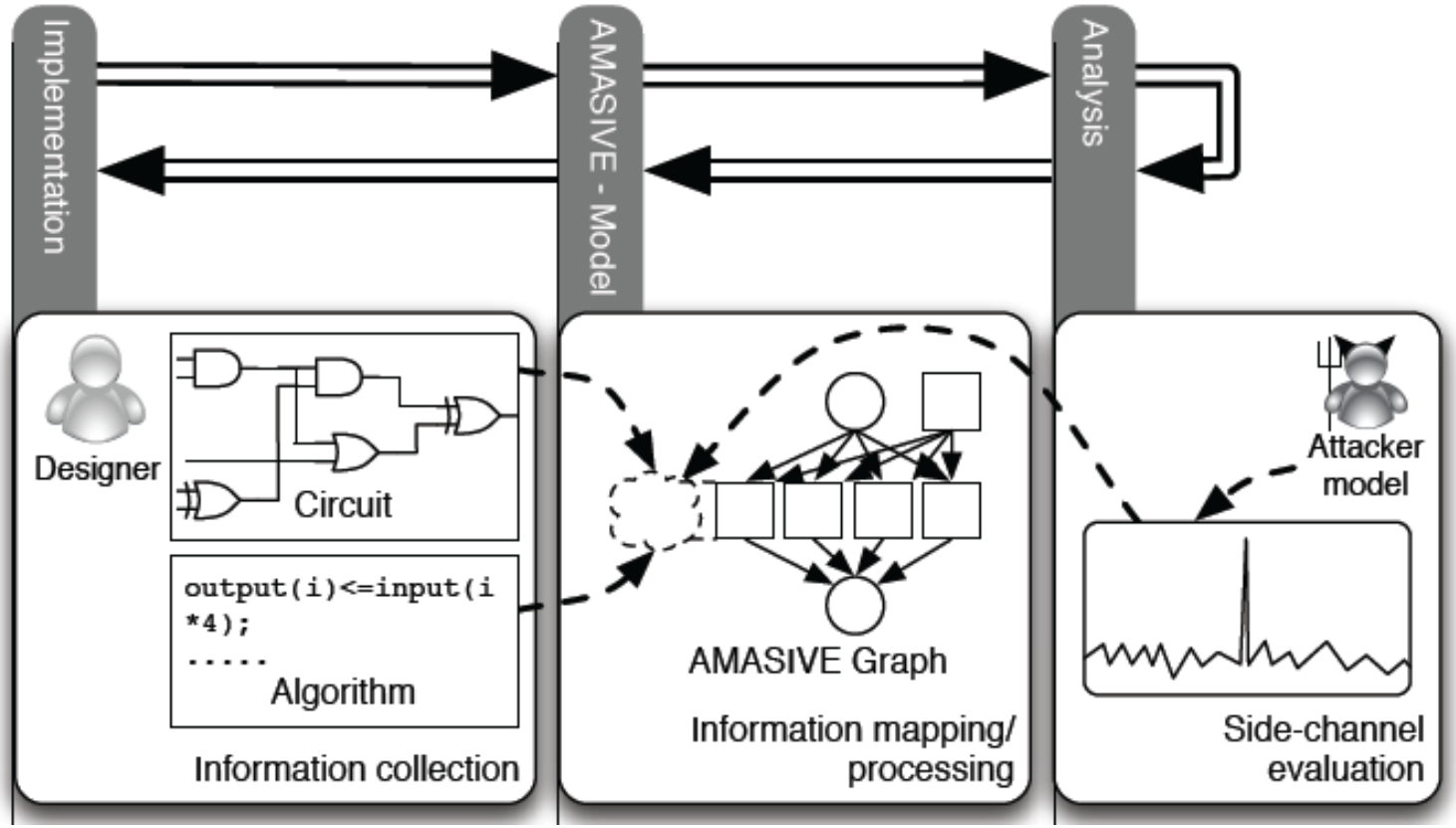


Generic SCA-related High-Level Synthesis Approach



Link of Spec/Impl and SCA by dedicated Graph-based Model

TECHNISCHE UNIVERSITÄT DARMSTADT



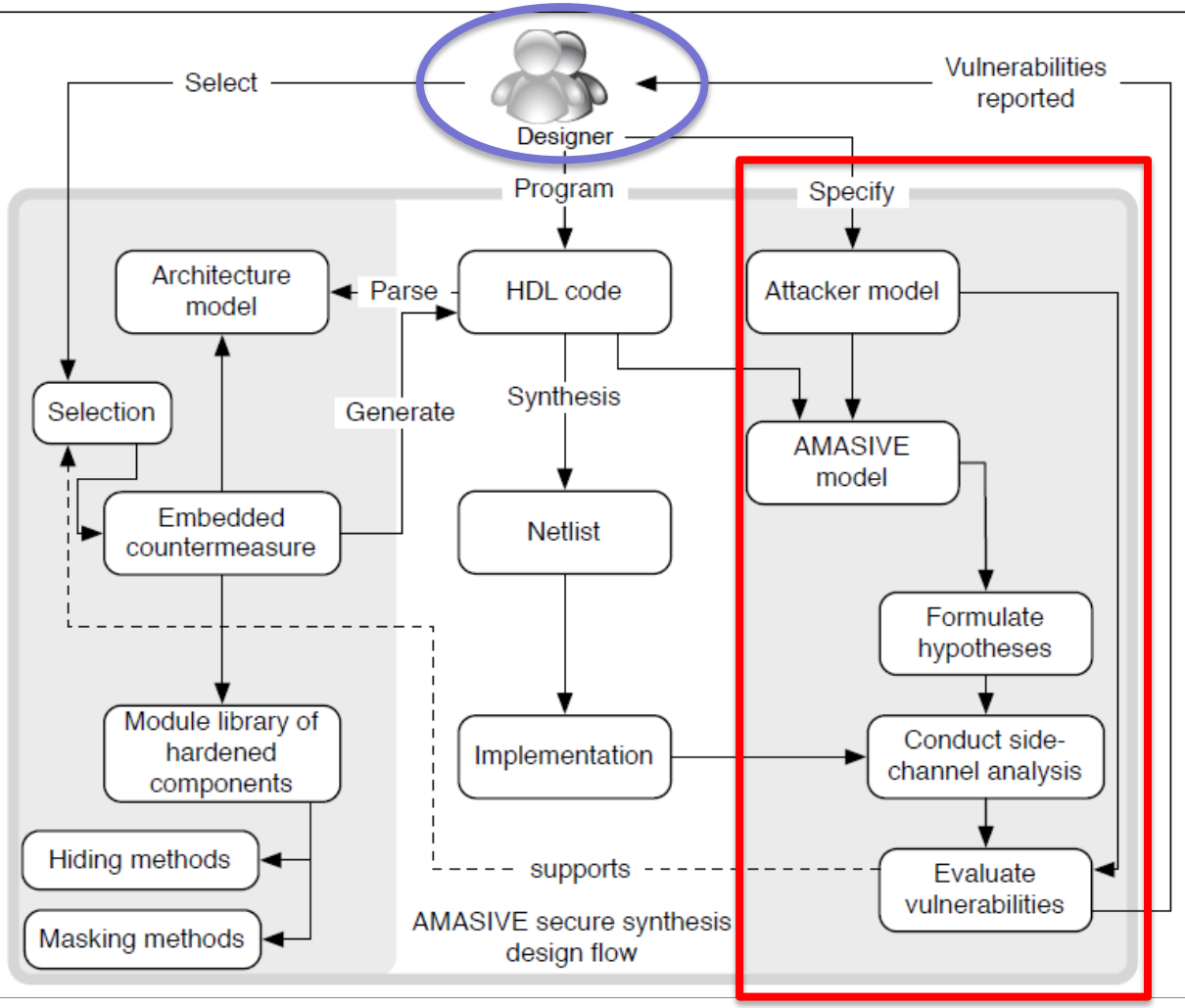
Graph Node Types

TECHNISCHE UNIVERSITÄT DARMSTADT

Node type	Component	AMASIVE identifier
Register	Register element/flip flop	*register(clk,rst,en)*
Switch	Multiplexer	*mux(sel)*
Permutation	Permutation module	*permutation*
Non-linear function	Substitution module	*non-linear*, *bilinear*,*invertible*
Data path	Top level module of data path	*data path*

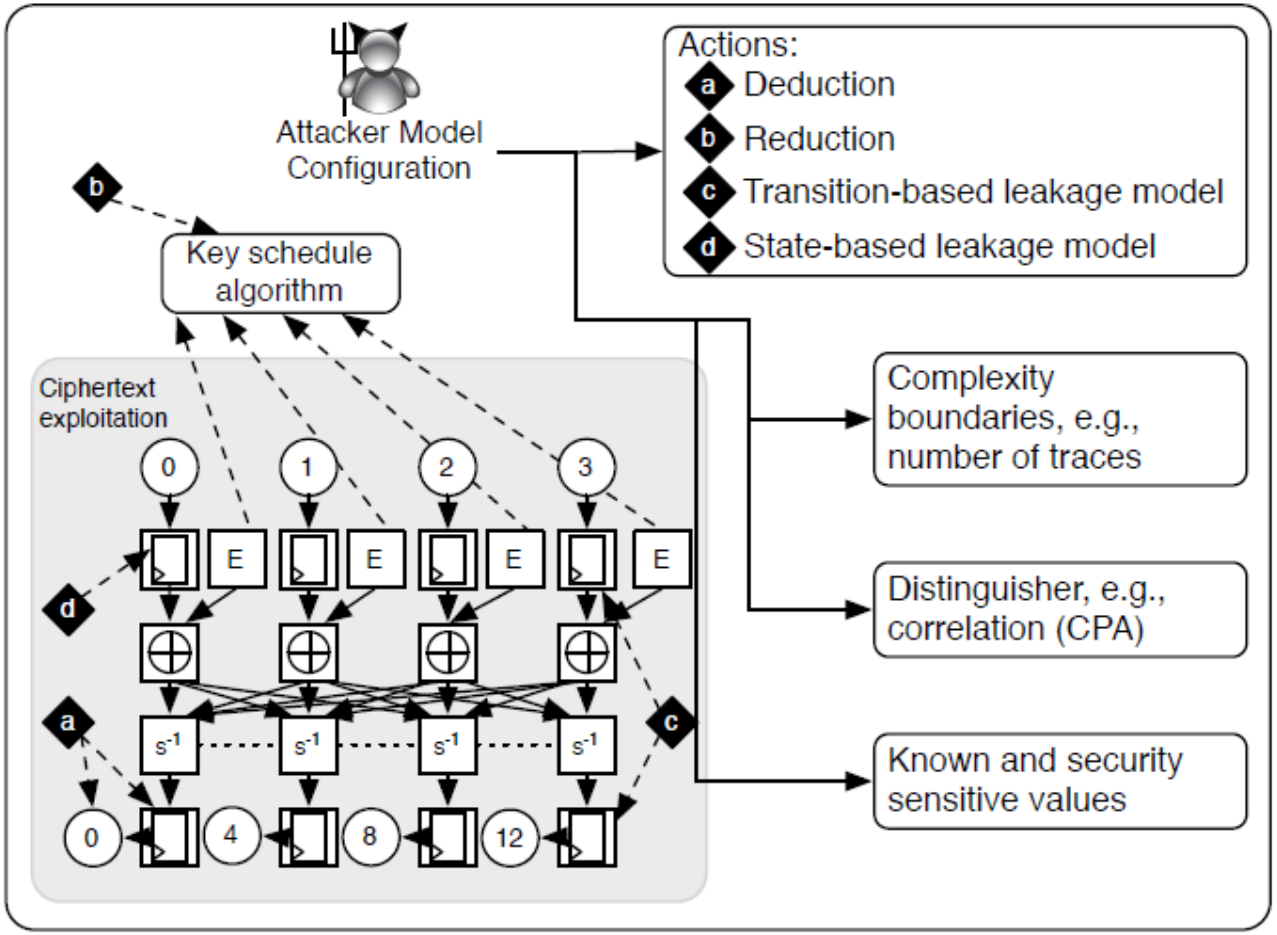
AMASIVE: A novel Re-Synthesis based Hardware Design Flow

TECHNISCHE UNIVERSITÄT DARMSTADT



Detection of Vulnerabilities

Attacker Model



Security Analysis Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Require: all nodes $n \in V$ of the AMASIVE graph $G = (V, E)$ are contained in the view $\mathbb{V}$
Require: set of security sensitive nodes $\mathbb{S} \subseteq \mathcal{P}(V)$
Require: ordered list of N actions to traverse the graph $\mathbb{A} = \{a_1^G, a_2^G, ..., a_N^G\}$ with corresponding requirement function $R_G : (\mathbb{V}, \mathbb{A}) \mapsto \{0, 1\}$
Require: computational complexity boundary C

1: $i = 1$
2: **repeat**
3: **if** $R(\mathbb{V}, a_i^G) = 1$ **then**
4: $\mathbb{V} = a_i^G(\mathbb{V})$
5: $i = 1$
6: **else**
7: $i = i + 1$
8: **end if**
9: **until** $i > N$
10: Attacker wins if $\exists \{s_1, s_2, ..., s_p\} \in \mathbb{S}$ with $(s_j, \epsilon_j) \in \mathbb{V}$ such that $(\sum_{j=1}^{p} \epsilon_j) < C$

Main Characteristics of Vulnerability Evaluation Module

TECHNISCHE UNIVERSITÄT DARMSTADT

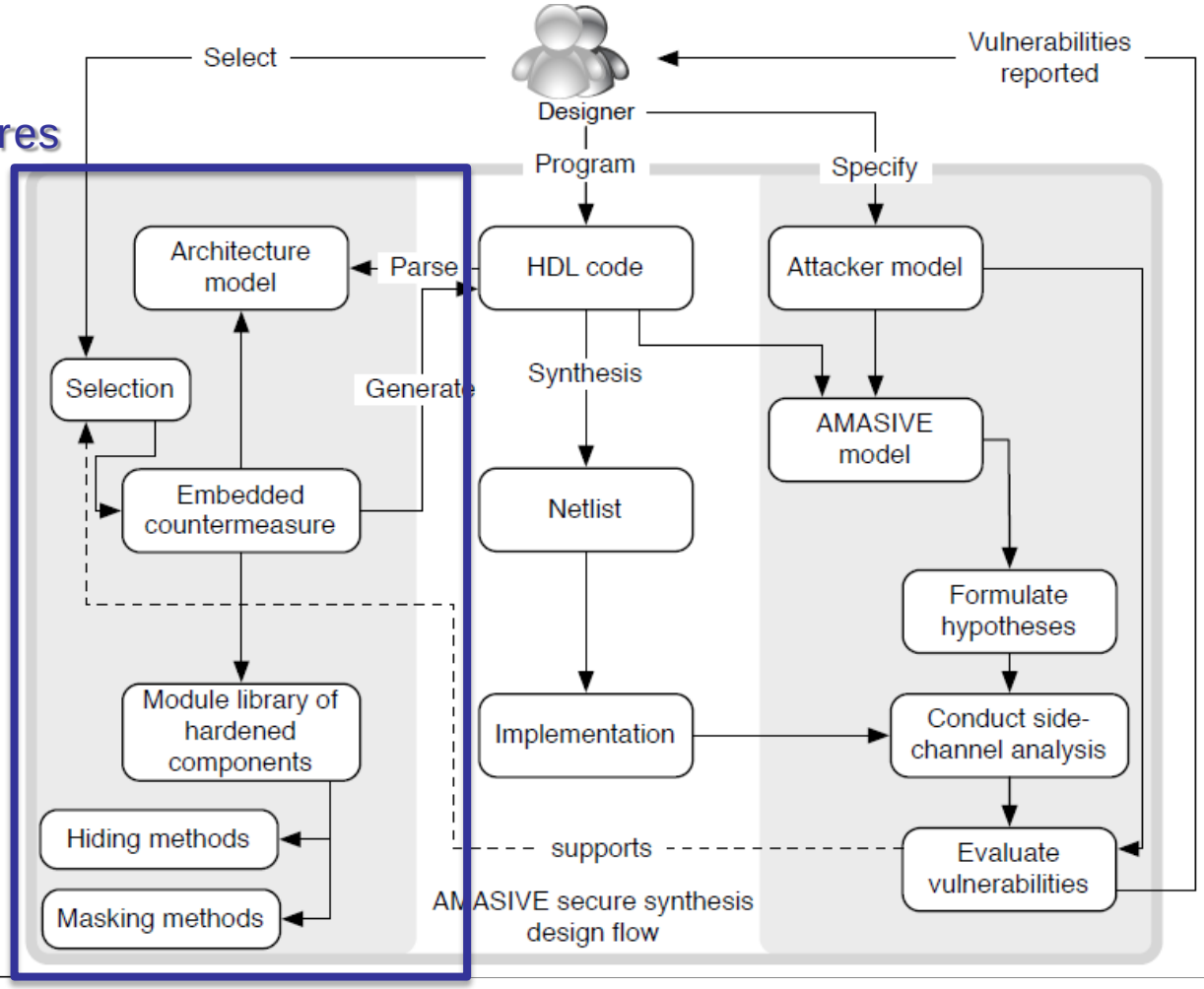
- Variable strength level of attacker achievable by adding/removing known nodes and actions or by changing complexity boundaries
- Attacker model is indepentent of the constructed graph
- Security analysis as a game with the goal of the attacker to yield a set of security sensitive nodes
- Identification of suitable hypothesis functions currently available for both HW and HD models within CPA of symmetric ciphers
- Easily extendable in terms of both distinguishers and SCA attacks

M. Zohner, M. Stöttinger, S. Huss, O. Stein: An Adaptable, Modular, and Autonomous Side-Channel Vulnerability Evaluator. IEEE HOST Conf., 2012

AMASIVE: A novel Re-Synthesis based Hardware Design Flow

TECHNISCHE UNIVERSITÄT DARMSTADT

Hardening by Countermeasures



Generation of hardened Circuits

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithm 1 Generate a side-channel hardened circuit

Require: VHDL code attributed by AMASIVE identifiers and the inputs of the security analysis

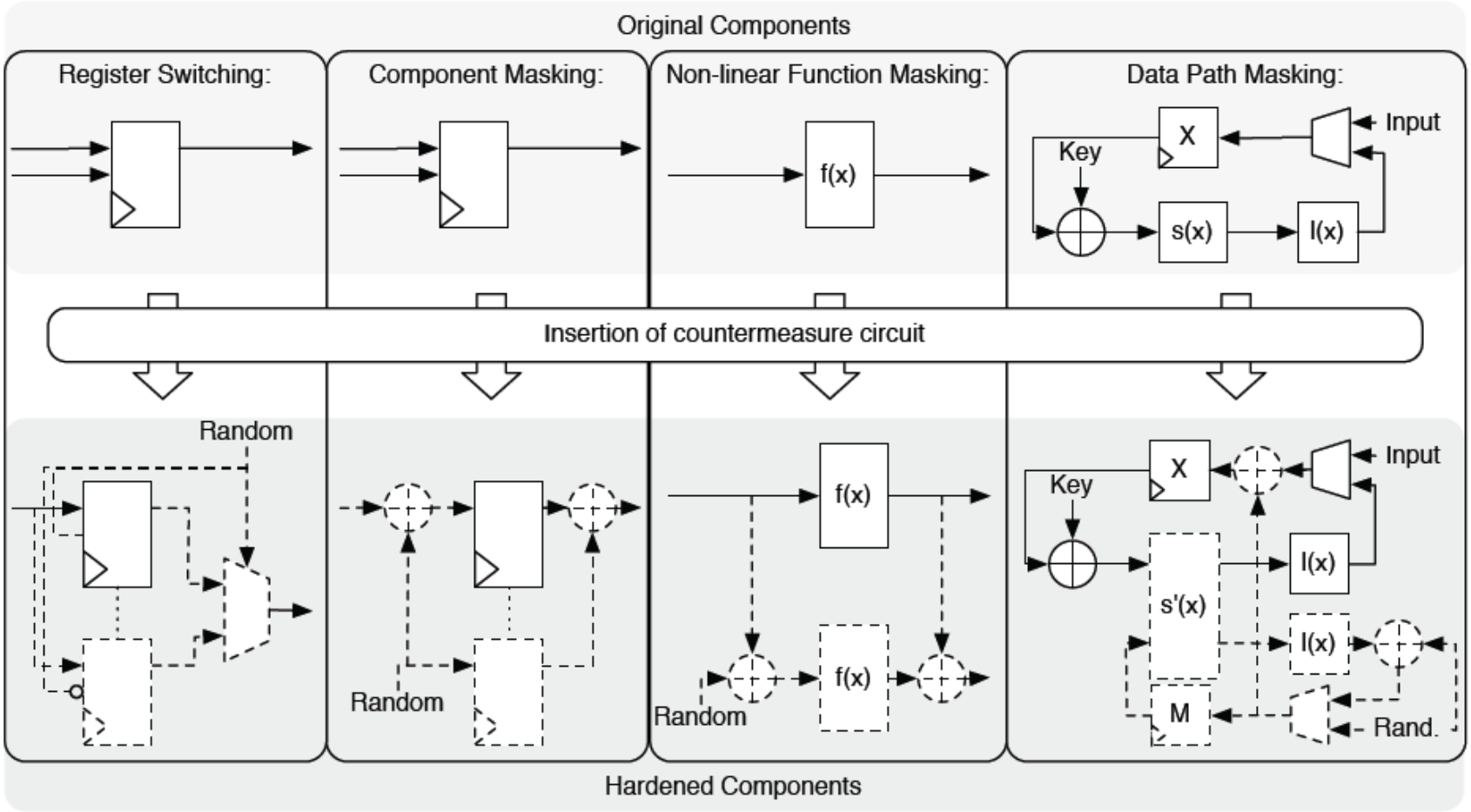
1: Parse the given design code hierarchically beginning with the data path module, which is marked with the AMASIVE identifier *data path*
2: Generate $G_A(V, E)$ based on VHDL code
3: Enhancement of $G_A(V, E)$:
4: Select a node V, which should be manipulated by embedding a side-channel countermeasure
5: Manipulate $G_{A,Sec}(V, E) \leftarrow G_A(V, E)$ by establishing new edges E_+ to embedded node V_+ in $G_A(V, E)$
6: Generate new VHDL code (VHDL_{Sec}), based on the original VHDL code, secured VHDL macros, and the new $G_{A,Sec}(V, E)$
7: (Re-)Synthesis of the side-channel attack hardened circuit based on new VHDL_{Sec}

Generic Countermeasures currently available in Data Base

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Random Register Switching**
- **Component Masking**
- **Boolean Masking of Data Path**

SCA Hardening of Components



Insertion of additional Nodes for Countermeasure Embedding

TECHNISCHE UNIVERSITÄT DARMSTADT

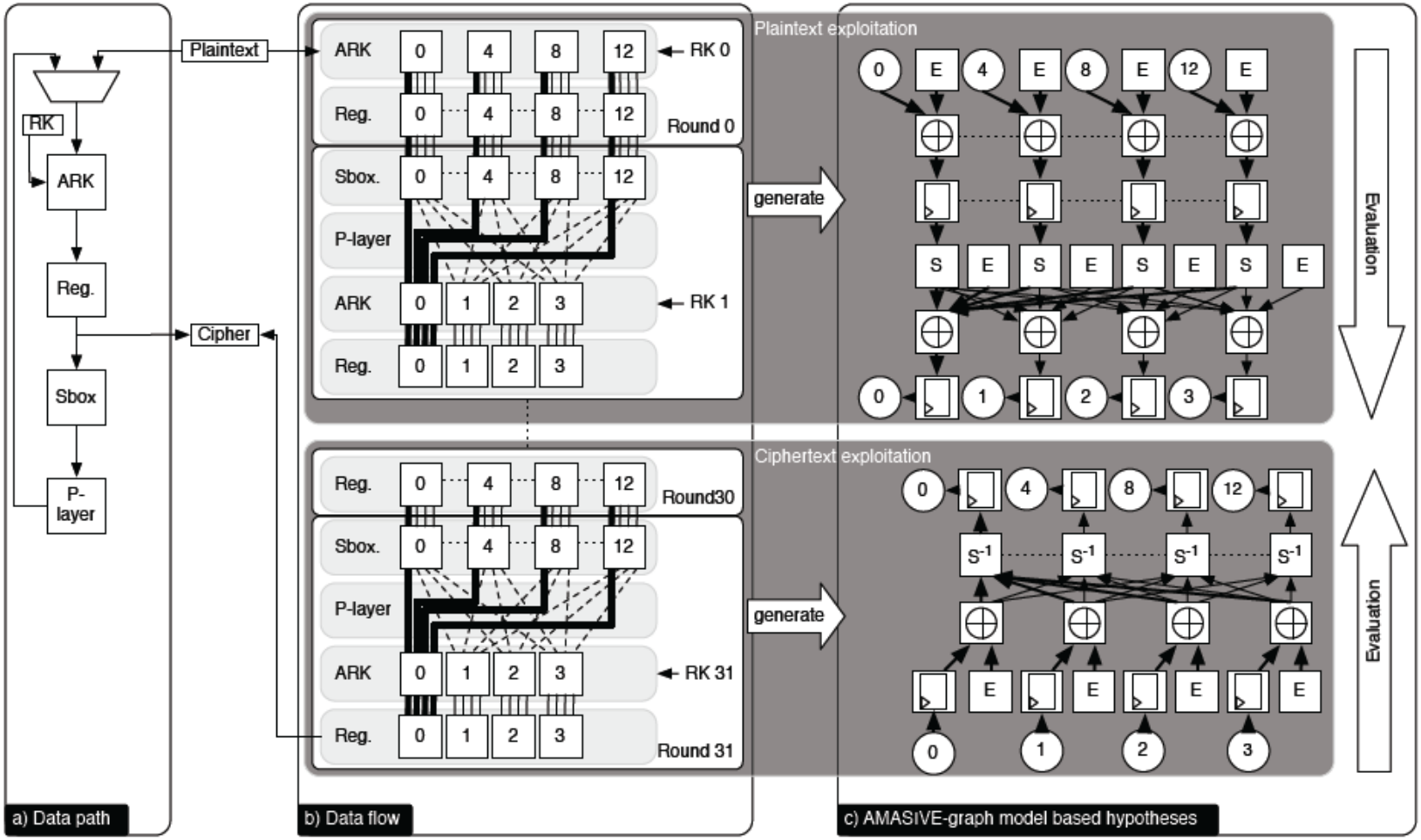
Algorithm 2 Add_Seq: Add new k nodes $V_{+s,i}$, $i = 1, \ldots, k$ sequentially between existing nodes V_{pre} and V_{succ}

Require: Architecture graph $G_A = (V, E)$ and node V_j that becomes the successor V_{succ} after $V_{+s,i}$ are placed

1: $V_{succ} \leftarrow V_j$ and V_{pre} are all preceding nodes V, which are connected to V_j
2: Collect all E_i between V_{succ} and V_{pre}
3: Reroute E_i to connect V_{pre} and $V_{+s,i}$
4: Create new edges $E_{+,i}$ to connect $V_{+s,i}$ and V_{succ}
5: **return** $G_{A,+} = (V, E) = (\{V, V_{+s,i}\}, E)$

Example: PRESENT Block Cipher

TECHNISCHE UNIVERSITÄT DARMSTADT



Example: Resource Consumption and Performance of Cipher Variants

TECHNISCHE UNIVERSITÄT DARMSTADT

	Resource consumption					Performance	
	Register	BRAM	LUT	Slices	Overhead[%]	Clock [MHz]	Throughput [MB/s]
Unprotected	414	0	540	255	0	431	862
Random register switching	622	0	652	389	53	321	642
Masked register	607	0	749	312	22	381	762
Masked data path	624	0	842	339	32	415	830
Combined countermeasures	696	0	1036	401	57	256	512

Implementation platform:

SASEBO II board

Example: Result of CPA on both unprotected and hardened Ciphers

TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Summary of SCA Attack Results, KeyLength = 80 bit

TECHNISCHE UNIVERSITÄT DARMSTADT

Effort to get the secret key by brute force	Number of correct found sub keys	Minimal number of needed traces				
		Unprotect. design	Hardened design			
			Rand. reg. sw.	Masked reg.	Masked data path	Combined
2^{80}	0	0	0	0	0	0
2^{76}	1	5	9	14	25	34
2^{72}	2	6	38	140	55	100,000
2^{68}	3	22	112	263	2,223	>100,000
2^{64}	4	85	162	321	22,194	>100,000
2^{60}	5	137	437	565	24,527	>100,000
2^{56}	6	423	475	705	27,660	>100,000
2^{52}	7	449	732	8,511	35,811	>100,000
2^{48}	8	527	957	11,330	51,154	>100,000
2^{44}	9	543	1,185	22,560	55,168	>100,000
2^{40}	10	645	1,274	27,472	77,785	>100,000
2^{36}	11	664	1,342	29,835	96,480	>100,000
2^{32}	12	688	2,112	30,561	>100,000	>100,000
2^{28}	13	1,358	2,269	31,066	>100,000	>100,000
2^{24}	14	1,381	4,258	>100,000	>100,000	>100,000
2^{20}	15	1,497	6,016	>100,000	>100,000	>100,000
2^{16}	16	1,548	11,824	>100,000	>100,000	>100,000

Conclusions (I)

TECHNISCHE UNIVERSITÄT DARMSTADT

Specific characteristics of AMASIVE:

- Parsing of initial VHDL design description of cryptographic modules and automatic extraction of their data and control flows
- Construction of intermediate graph as the foundation of subsequent analysis and synthesis steps
- Sophisticated power SCA featuring configurable attacker models, various distinguishers, and hypothesis function generation

Conclusions (II)

TECHNISCHE UNIVERSITÄT DARMSTADT

- User-controlled insertion of first-order countermeasures, VHDL code generation, and logic re-synthesis
- Applicable to various implementation platforms such as FPGA, ASIC, and Full-Custom IC
- Proven SCA resistance quality of the proposed module hardening methodology

… A big step towards high-level synthesis of side-channel resistant cryptographic modules