RUHR-UNIVERSITÄT BOCHUM

SILVER: Statistical Independence and Leakage Verification

26th Annual International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2020

David Knichel, Pascal Sasdrich, Amir Moradi

Security Engineering Group
Electrical Engineering and Information Technology
Ruhr University Bochum

Motivation

Problem

- **Fragmented view on security notions of masked implementations**
- **Tool with a wide feature range: maskVerif**
  - **Language-based →not in direct conformity with security notions**

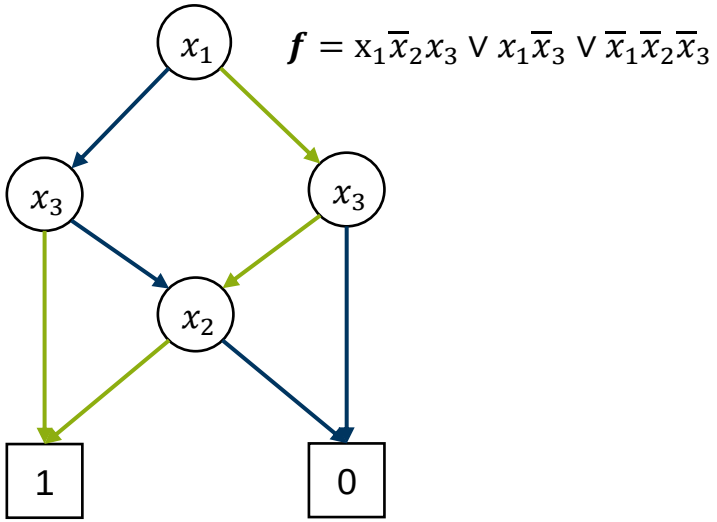
Goal

- **Building a sound and complete verification tool**

Background

ROBDD

- ***Reduced Ordered*** **Binary Decision Diagram**
- **Ordered: Every path from root to leaf traverses every variable only once and in the same order**
- **Reduced: No isomorphic subtrees and no redundant nodes**

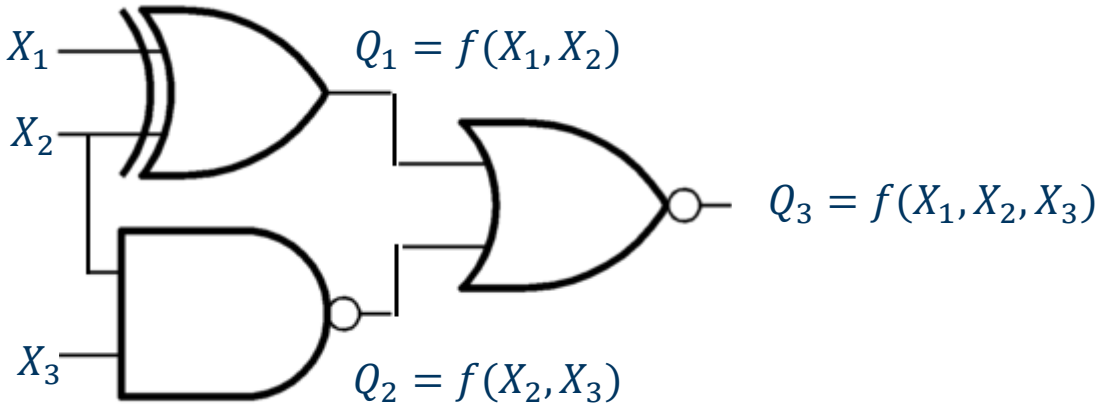


Canonical representation of a boolean function $f: F_2^n \rightarrow F_2$

Background

ROBDD

- **Wires of a circuit can be seen as binary random variables**
- **These wires are functions of the input**
- **Can be efficiently represented as ROBDD**



Background

Statistical Independence

- **Two Events A, B:** $\boldsymbol{Pr}[\boldsymbol{A} \wedge \mathbf{B}] = \mathbf{Pr}[\mathbf{A}] \times \boldsymbol{Pr}[\boldsymbol{B}]$
- **Two Random Variables:** $\boldsymbol{Pr}[\boldsymbol{X} = \boldsymbol{x} \wedge \boldsymbol{Y} = \boldsymbol{y}] = \mathbf{Pr}[\boldsymbol{X} = \boldsymbol{x}] \times \boldsymbol{Pr}[\boldsymbol{Y} = \boldsymbol{y}] \; \forall \boldsymbol{x}, \boldsymbol{y}$

Background

Statistical Independence of Two Binary Random Variables

- **One can show that for $\boldsymbol{X} \in \mathbb{F}_2, \boldsymbol{Y} \in \mathbb{F}_2$:**
  - $\boldsymbol{X}$ and $\boldsymbol{Y}$ are stat. independent $\Leftrightarrow \Pr[X = 1 \wedge Y = 1] = \Pr[X = 1] \times \Pr[Y = 1]$
- **Simplifies verification to check of one event combination instead of four**

Background

Statistical Independence of Joint Distributions over Binary Random Variables

- **We showed that for $\boldsymbol{X} \in \mathbb{F}_2^{\boldsymbol{n}}$, $\boldsymbol{Y} \in \mathbb{F}_2^{\boldsymbol{m}}$:**
  - $\boldsymbol{X}$ and $\boldsymbol{Y}$ are stat. independent $\Leftrightarrow \forall\, \boldsymbol{X}' \subseteq \mathbf{X}, \mathbf{Y}' \subseteq \mathbf{Y}$: $\Pr[\boldsymbol{X}' = (1, \dots, 1) \wedge \boldsymbol{Y}' = (1, \dots, 1)] = \Pr[\boldsymbol{X}' = (1, \dots, 1)] \times \Pr[\boldsymbol{Y}' = (1, \dots, 1)]$
  - Example: $\boldsymbol{X} = (X_1, X_2), \boldsymbol{Y} = (Y_1, Y_2)$
    - $\Pr[X_i = 1 \wedge Y_j = 1] \stackrel{?}{=} \Pr[X_i = 1] \times \Pr[Y_j = 1]$
    - $\Pr[X_i = 1 \wedge \boldsymbol{Y} = (1,1)] \stackrel{?}{=} \Pr[X_i = 1] \times \Pr[\boldsymbol{Y} = (1,1)]$
    - $\Pr[Y_i = 1 \wedge \boldsymbol{X} = (1,1)] \stackrel{?}{=} \Pr[Y_i = 1] \times \Pr[\boldsymbol{X} = (1,1)]$
    - $\Pr[\boldsymbol{Y} = (1,1) \wedge \mathbf{X} = (1,1) \stackrel{?}{=} \Pr[\boldsymbol{Y} = (1,1)] \times \Pr[\boldsymbol{X} = (1,1)]$

Background

Reduction to Check of Binary Random Variables

- $\Pr[\boldsymbol{X} = (X_1, X_2, X_2) = (1,1,1)] = \Pr[\underbrace{X_1 X_2 X_2}_{\text{simple binary random variable}} = 1]$

- **We reduced the check of independence for *joint distribution of random variables* to the check of *simple binary random variables***
  - *Can be efficiently realized utilizing ROBDDs*

Application to Formal Verification of Masked Implementation

Approach

- **Redefine common security notions by the means of statistical independence of joint distributions over binary random variables**
- **Check this independence utilizing BDDs**

Application to Formal Verification of Masked Implementation

ISW d-probing security

- **Unshared function with multiple sensitive inputs, for example:** (X, Y, Z)
- **Every set of probes** $\boldsymbol{Q} = (Q_1, Q_2, \ldots, Q_d)$ **on wires of the shared circuit:**
  - $\boldsymbol{Q}$ is independent of (X, Y, Z)

joint distributions of binary random variables ⇒ utilizing ROBDDS

Application to Formal Verification of Masked Implementation

d-Non-Interference

- **Idea: For each set of probes $\boldsymbol{Q} = (Q_1, Q_2, \dots, Q_d)$ on the circuit, their must exists a simulation set $\boldsymbol{S}$ of input shares and restricted to max. $\boldsymbol{d}$ shares per input, s.t. $\boldsymbol{Q}$ is perfectly simulatable using $\boldsymbol{S}$.**

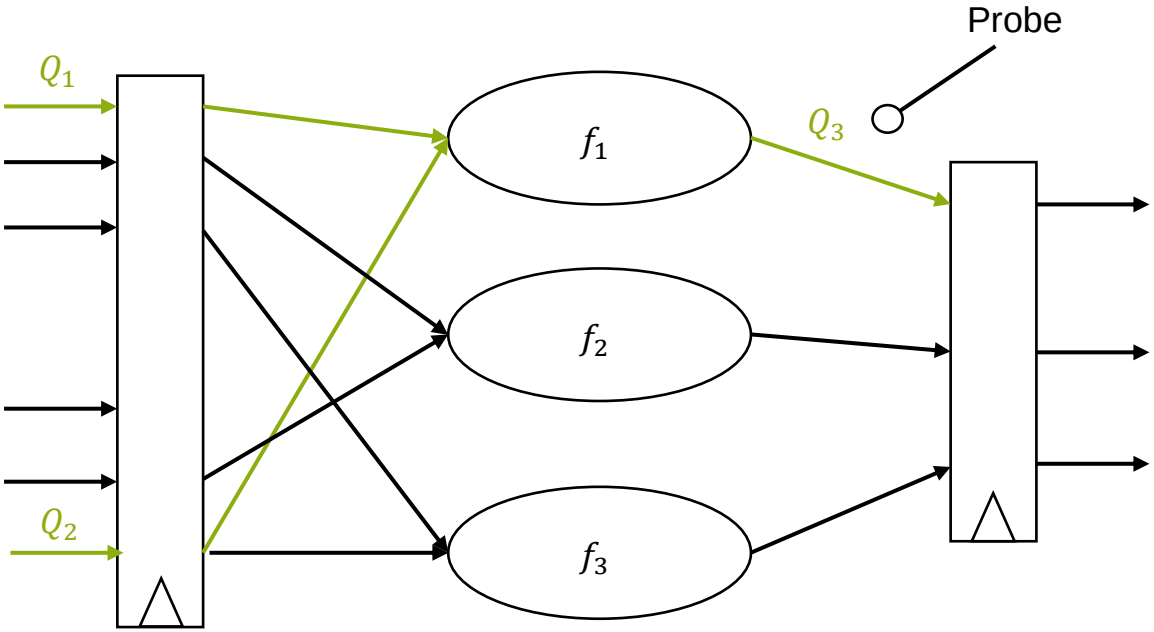
- **Translated into probability distributions:**
  - $\boldsymbol{Pr[Q|S] = Pr[Q|Sh(X)]}$ **where** $\boldsymbol{Sh(X)}$ **consists of all the shared inputs**

- **If all input shares are i.i.d. this simplifies to:**
  - **Independence of** $\boldsymbol{(Q \cup S)}$ **and** $\boldsymbol{Sh(X) \backslash S}$

joint distibutions of binary random variables ⇒ utilizing ROBDDS

Application to Formal Verification of Masked Implementation

Glitch-Extended Probes



$$Q = \{Q_3\} \Rightarrow Q_{extended} = \{Q_1, Q_2\}$$

Application to Formal Verification of Masked Implementation

Uniformity (Single Unshared Output Example, 3 Shares)

- **Uniformity:** $Pr[(Z_1 = z_1, Z_2 = z_2, Z_3 = z_3) \mid Z = z] = p$, $if\ z = z_1 \oplus z_2 \oplus z_3$, **else 0**
- **We showed, that instead we can check:**
  - $(Z_1, Z_2), (Z_1, Z_3), (Z_2, Z_3)$ **are each balanced**
- **Reduces uniformity check to balancedness verification of (joint) output shares**

We generalized this approach even for multiple unshared outputs.

Application to Formal Verification of Masked Implementation

Uniformity

- $Z = (Z_1, Z_2, \dots, Z_m)$ is balanced iff each component function is balanced.

binary random variable ⇒ utilizing ROBDDS

Supported Security Notions

Overview

- **d-probing security**
- **d-Non-Interference (d-NI)**
- **d-Strong-Non-Interference (d-SNI)**
- **d-Probe-Isolating-Non-Interference (d-PINI)**
- **Output Uniformity**

standard (software)
and
glitch-extended (hardware)

Program Flow: SILVER

Overview

- **Netlist is in self-defined format**
- **Result contains for each supported security Notion:**
  - **Security order d**
  - **First failing probe for order (d+1)**

Results

Table of Tested Designs:

Scheme	Pos.†	d	Probing		NI		SNI		PINI		Unif.
			std.	rob.	std.	rob.	std.	rob.	std.	rob.	
Gadgets											
DOM [29]	19	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✗[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM [29]	42	2	✓[3 ms]	✓[4 ms]	✓[6 ms]	✓[19 ms]	✓[8 ms]	✗[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM [29]	74	3	✓[98 ms]	✓[1.2 s]	✓[2.2 s]	✓[23.7 s]	✓[3.2 s]	✗[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM SNI [26]	21	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM SNI [26]	45	2	✓[3 ms]	✓[5 ms]	✓[6 ms]	✓[30 ms]	✓[7 ms]	✓[29 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM SNI [26]	78	3	✓[0.1 s]	✓[1.5 s]	✓[2.4 s]	✓[39.4 s]	✓[3.7 s]	✓[39.4 s]	✗[0.0 s]	✗[0.0 s]	✓[0.0 s]
PARA1 [5]	22	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
PARA2 [5]	45	2	✓[3 ms]	✓[6 ms]	✓[5 ms]	✓[32 ms]	✓[8 ms]	✓[37 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
PARA3 [5]	68	3	✓[61 ms]	✓[0.5 s]	✓[1.2 s]	✓[12.1 s]	✗/✓[0.6 s]	✗[0 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
PARA3 SNI [5]	82	3	✓[0.2 s]	✓[1.4 s]	✓[2.8 s]	✓[35.5 s]	✓[4.1 s]	✓[40.4 s]	✗[0 ms]	✗[0 ms]	✓[0 ms]
PINI1 [17]	21	1	✓[0 ms]	✗[0 ms]	✓[0 ms]	✗[0 ms]	✓[0 ms]	✗[0 ms]	✓[0 ms]	✗[0 ms]	✓[0 ms]
PINI2 [17]	51	2	✓[7 ms]	✗[0 ms]	✓[10 ms]	✗[0 ms]	✓[12 ms]	✗[0 ms]	✓[22 ms]	✗[0 ms]	✓[0 ms]
HPC1 [16]	22	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✗[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
HPC1 [16]	52	2	✓[5 ms]	✓[7 ms]	✓[7 ms]	✓[23 ms]	✓[9 ms]	✗[0 ms]	✓[16 ms]	✓[46 ms]	✓[0 ms]
HPC2 [16]	32	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✗[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
HPC2 [16]	75	2	✓[6 ms]	✓[12 ms]	✓[11 ms]	✓[37 ms]	✓[13 ms]	✗[0 ms]	✓[19 ms]	✓[61 ms]	✓[0 ms]
ISW SNI REF [26]	26	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
ISW SNI REF [26]	65	2	✓[5 ms]	✓[8 ms]	✓[7 ms]	✓[36 ms]	✓[9 ms]	✓[34 ms]	✓[16 ms]	✓[59 ms]	✓[0 ms]
CMS3 [36]	104	3	✗/✓[0.1 s]	✗/✓[0.3 s]	✗/✓[0.8 s]	✗/✓[2.6 s]	✗/✓[1.3 s]	✗/✓[4.4 s]	✗[0 ms]	✗[0 ms]	✓[0 ms]
UMA2 [36]	81	2	✗/✓[2 ms]	✗/✓[0 ms]	✗/✓[7 ms]	✗/✓[4 ms]	✗/✓[6 ms]	✗/✓[3 ms]	✗[0 ms]	✗[0 ms]	✓[0 ms]
DOM2 DEP‡ [36]	56	2	✓[4 ms]	✗/✓[8 ms]	✓[3 ms]	✗/✓[20 ms]	✓[4 ms]	✗[0 ms]	✓[4 ms]	✗/✓[21 ms]	✓[0 ms]

Results

Table of Tested Designs (continued):

S-boxes											
$\text{PRESENT}_{\text{TI}}$ [40]	177	2	✓[4 ms]	✓[8 ms]	^{1}✗[4 ms]	^{1}✗[0 ms]	^{1}✗[3 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[2 ms]
$\text{PRESENT}_{\text{TI}}$ [25]	377	2	✓[15 ms]	✓[6 ms]	^{1}✗[2 ms]	^{1}✗[0 ms]	^{1}✗[2 ms]	^{1}✗[0 ms]	^{1}✗[1 ms]	^{1}✗[0 ms]	✓[0 ms]
$\text{PRESENT}_{\text{TI}}$ [25]	161	2	^{1}✗[3 ms]	^{1}✗[4 ms]	^{1}✗[32 ms]	^{1}✗[0 ms]	^{1}✗[26 ms]	^{1}✗[0 ms]	^{1}✗[2 ms]	^{1}✗[0 ms]	✗[0 ms]
$\text{PRINCE}_{\text{TI}}$ [37]	150	2	✓[2 ms]	✓[10 ms]	^{1}✗[2 ms]	^{1}✗[0 ms]	^{1}✗[2 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]
$\text{PRINCE}_{\text{CMS}}$ [14]	261	1	✓[3 ms]	✓[97 ms]	✓[7 ms]	✓[2.8 s]	✓[9 ms]	^{1}✗[1 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]
$\text{SKINNY8}_{\text{TI}}$ [7]	240	2	✓[51.2 s]	✓[2 min]	^{1}✗[2 min]	^{1}✗[2.0 s]	^{1}✗[2 min]	^{1}✗[2.0 s]	^{1}✗[77 ms]	^{1}✗[1.3 s]	✓[29.6 s]
$\text{SKINNY8}_{\text{CMS}}$ [8]	192	1	✓[20 ms]	✓[0.3 s]	^{1}✗[0.3 s]	^{1}✗[17 ms]	^{1}✗[0.3 s]	^{1}✗[15 ms]	^{1}✗[1 ms]	^{1}✗[1 ms]	✓[1 ms]
AES_{DOM} [29]	884	1	✓[3.3 s]	✓[21 min]	^{1}✗[0.8 s]	^{1}✗[0.4 s]	^{1}✗[0.8 s]	^{1}✗[0.4 s]	^{1}✗[0.2 s]	^{1}✗[40 ms]	✓[0.1 s]
AES_{CMS} [19]	938	1	✓[9.4 s]	✓[2.9 h]	^{1}✗[0.9 s]	^{1}✗[0.5 s]	^{1}✗[0.9 s]	^{1}✗[0.5 s]	^{1}✗[0.2 s]	^{1}✗[42 ms]	✓[1.8 s]
Functions											
A_{in} [37]	18	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
A_{m} [37]	20	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
A_{out} [37]	20	1	✓[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]	✓[0 ms]	✓[0 ms]
$\mathcal{Q}_{12}^{4}$ [42]	48	1	✓[0 ms]	✓[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	^{1}✗[0 ms]	✓[0 ms]

† Number of possible probe positions, i.e., output wires of gates. ‡ Assuming identical inputs, i.e., $a = b$.

Discussion

Advantage of Avoiding False Negatives

Q_{12}^4:

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 \\
x_2 &= F_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 \\
y_1 &= G_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_1 \oplus b_1 \\
y_2 &= G_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_2 \\
y_3 &= G_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_1 \oplus b_2 \\
y_4 &= G_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_2 \\
z_1 &= H_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus a_1c_1 \oplus c_1 \\
z_2 &= H_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 \oplus a_1c_2 \\
z_3 &= H_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 \oplus a_2c_1 \\
z_4 &= H_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus a_2c_2 \oplus c_2 \\
t_1 &= K_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = d_1 \\
t_2 &= K_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = d_2
\end{aligned}
\qquad
\begin{aligned}
\bar{x}_1 &= x_1 \\
\bar{x}_2 &= x_2 \\
\bar{y}_1 &= y_1 \oplus y_2 \\
\bar{y}_2 &= y_3 \oplus y_4 \\
\bar{z}_1 &= z_1 \oplus z_2 \\
\bar{z}_2 &= z_3 \oplus z_4 \\
\bar{t}_1 &= t_1 \\
\bar{t}_2 &= t_2
\end{aligned}
$$

Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating Masking Schemes.

Discussion

Advantages

- **Completeness: SILVER supports all established security and composability notions**
- **Soundness: Due to direct conformity to the security notions, SILVER avoids false negatives and thus avoids introducing unnecessary overhead into the design.**
- **Returning failing probe: Knowing which probe fails to fulfill the security notion allows an easier fix of masked design.**

Disadvantages

- **Due to direct conformity with security notions, SILVER is slower than language-based approaches (like for example maskVerif).**

Demo

Conclusion

With SILVER, we developed an easy-to-use tool for sound and complete verification of the security and composability of masked Implementations.

Additional Information

Tool:

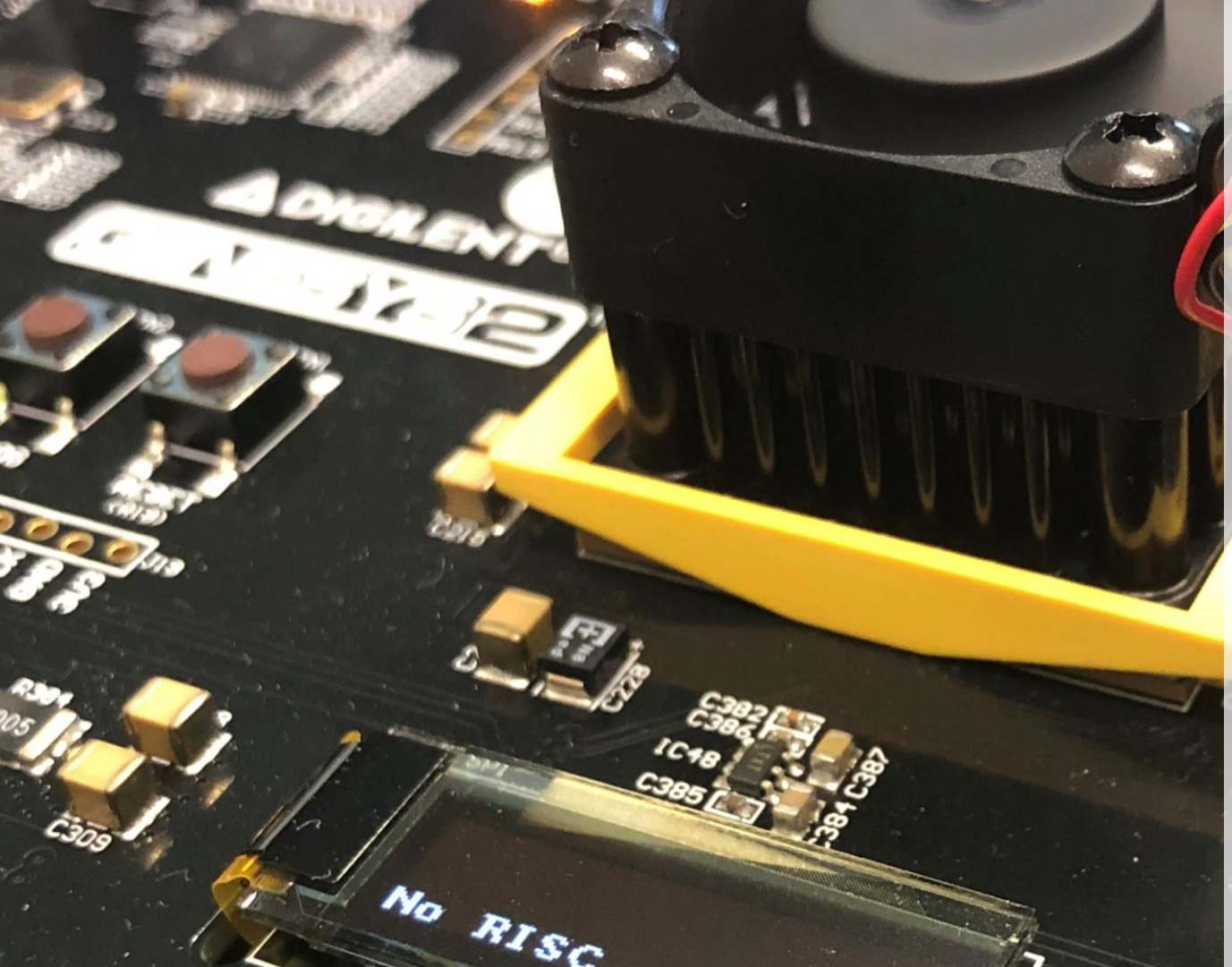
- https://github.com/Chair-for-Security-Engineering/SILVER

ePrint Link:

- https://eprint.iacr.org/2020/634/20200629:134004

Contacts:

- David Knichel: David.Knichel@rub.de
- Pascal Sasdrich: Pascal.Sasdrich@rub.de
- Amir Moradi: Amir.Moradi@rub.de



Thank you!

David Knichel

Security Engineering Group
Electrical Engineering & Information Technology
Ruhr University Bochum