Glitch-Resistant Masking Revisited
or Why Proofs in the Robust Probing Model are Needed

Thorben Moos¹, Amir Moradi¹, Tobias Schneider² and François-Xavier Standaert²

¹Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
²ICTEAM/ELEN/Crypto Group, Université catholique de Louvain, Belgium

August 27th, 2019
Section 1

Introduction
Physical Attacks

Introduction

- Physical characteristics used to extract secrets:
  - Timing
  - Power
  - EM

- Countermeasures to increase attack complexity:
  - Masking
  - Hiding
  - Re-keying
Concept of Masking

Introduction

- Encode sensitive variables into shares
- Compute securely on shares
- Decode at end to recover result

Thorben Moos, Amir Moradi, Tobias Schneider and François-Xavier Standaert | Glitch-Resistant Masking Revisited | August 27th, 2019
Concept of Masking

Introduction

- Encode sensitive variables into shares
- Compute securely on shares
- Decode at end to recover result

Masking if implemented correctly increases the attack complexity exponentially in the number of shares.

(assuming sufficient noise)
Security Notions

Introduction

- Masked algorithms can be proven secure
- **Common Solution**: Probing model\(^1\)

---

**Definition (t-Probing Security)**

A circuit C is \(t\)-probing secure if and only if every \(t\)-tuple of its intermediate variables is independent of any sensitive variable.

---

\(^1\) Y. Ishai, A. Sahai and D. Wagner, *Private Circuits: Securing Hardware against Probing Attacks*, CRYPTO 2003
Security Notions

Introduction

• Masked algorithms can be proven secure
• **Common Solution:** Probing model\(^1\)

Definition (t-Probing Security)

A circuit C is t-probing secure if and only if every t-tuple of its intermediate variables is independent of any sensitive variable.

Example:

• 3rd-order masking
• Any possible combination of **three** probes should not reveal secret

---

\(^1\) Y. Ishai, A. Sahai and D. Wagner, *Private Circuits: Securing Hardware against Probing Attacks*, CRYPTO 2003
Security Notions

Introduction

- Masked algorithms can be proven secure
- **Common Solution**: Probing model\(^1\)

**Definition (t-Probing Security)**

A circuit C is \(t\)-probing secure if and only if every \(t\)-tuple of its intermediate variables is independent of any sensitive variable.

**Example:**

- 3rd-order masking
- Any possible combination of **three** probes should not reveal secret

---

\(^1\) Y. Ishai, A. Sahai and D. Wagner, *Private Circuits: Securing Hardware against Probing Attacks*, CRYPTO 2003
Security Notions

Introduction

• Masked algorithms can be proven secure

• **Common Solution:** Probing model

---

**Definition (t-Probing Security)**

A circuit $C$ is $t$-probing secure if and only if every $t$-tuple of its intermediate variables is independent of any sensitive variable.

---

**Example:**

• 3rd-order masking

• Any possible combination of **three** probes should not reveal secret

---

Security Notions

Introduction

- Scales badly with number of probes and complexity of algorithm
- Prove smaller sub-gadgets and compose securely

---

Security Notions

Introduction

• Scales badly with number of probes and complexity of algorithm
• Prove smaller sub-gadgets and compose securely

\[ F_1 \quad F_2 \quad F_3 \]

\[ ? \]

---

Security Notions

Introduction

- Scales badly with number of probes and complexity of algorithm
- Prove smaller sub-gadgets and compose **securely**

\[ F_1 \rightarrow F_2 \rightarrow F_3 \]

Common Solution: (Strong) Non-Interference

2

**Definition** (t - (Strong) Non-Interference)

A circuit gadget \( G \) is \( t \)-((S)NI) if and only if for any set of \( t_1 \) probes on its intermediate values and every set of \( t_2 \) probes on its output shares with \( t_1 + t_2 \leq t \), the totality of the probes can be simulated with \( t_1 + t_2 \) (only \( t_1 \)) shares of each input.

---

Security Notions

Introduction

- Scales badly with number of probes and complexity of algorithm
- Prove smaller sub-gadgets and compose **securely**

\[ \text{F}_1 \quad \text{F}_2 \quad \text{F}_3 \quad ? \quad \text{F}_1 \quad \text{F}_2 \quad \text{F}_3 \quad \text{F}_1 \quad \text{F}_2 \quad \text{F}_3 \]

Security Notions

Introduction

• Scales badly with number of probes and complexity of algorithm
• Prove smaller sub-gadgets and compose securely

Common Solution: (Strong) Non-Interference

Definition ($t$−(Strong) Non-Interference)

A circuit gadget $G$ is $t$−(Strong) Non-Interfering ($t$-(S)NI) if and only if for any set of $t_1$ probes on its intermediate values and every set of $t_2$ probes on its output shares with $t_1 + t_2 \leq t$, the totality of the probes can be simulated with $t_1 + t_2$ (only $t_1$) shares of each input.

---

Potential Flaws

Introduction

**Local Flaw:** Probing security of masked **module** is reduced.

**Example:** 2nd-order masking

\[ F_1 \]
Potential Flaws

Introduction

Local Flaw: Probing security of masked module is reduced.

Example: 2nd-order masking

Compositional Flaw: Probing security of composition of modules is reduced.

Example: 2nd-order masking
Robust Probing

Introduction

- Physical defaults (glitches, transitions, coupling) reduce masking order in practice
- Numerous higher-order hardware-oriented masking schemes:
  - CMS: Consolidated Masking Schemes
  - DOM: Domain-Oriented Masking
  - UMA: Unified Masking Approach
  - GLM: Generic Low-Latency Masking
Robust Probing

Introduction

- Physical defaults (glitches, transitions, coupling) reduce masking order in practice
- Numerous higher-order hardware-oriented masking schemes:
  - CMS: Consolidated Masking Schemes
  - DOM: Domain-Oriented Masking
  - UMA: Unified Masking Approach
  - GLM: Generic Low-Latency Masking
- Due to lack of model: Mostly focused on glitch-resistant (local) probing security
- Dedicated extension of probing model to hardware masking:

  Composable Masking Schemes in the Presence of Physical Defaults and the Robust Probing Model

  Sebastian Faust¹, Vincent Grosso¹,², Santos Merino Del Pozo³, Clara Paglialonga¹, François-Xavier Standaert³
Overview

Introduction

In this paper:

• Analysis of higher-order HW masking schemes
  • CMS - local
  • DOM - local
  • UMA - compositional
  • GLM - local + compositional

• Experiments and evaluation of practical impact of flaws

• Conclusion: Always verify local and compositional security in adequate model

Strong case for unified HW security notion (e.g., robust probing model)
Overview

Introduction

In this paper:

- Analysis of higher-order HW masking schemes
  - CMS - **local**
  - DOM - **local**
  - UMA - **compositional**
  - GLM - **local** + **compositional**

  \[
  \text{Strong case for unified HW security notion} \\
  \text{(e.g., robust probing model)}
  \]

- Experiments and evaluation of practical impact of flaws

- **Conclusion:** Always verify local and compositional security in adequate model

Disclaimer

Most of the flaws are in instantiations/compositions which are not explicitly given in the sources, and their specific instantiations at lower orders should not be affected by our flaws. The discussed flaws can still result in insecure designs when used by others.
Section 2

Local Flaws
Consolidated Masking Scheme

Local Flaws

- First proposed at CRYPTO 2015 as $d+1$ masking scheme
- Then used at CHES 2016 to mask AES with $d+1$ shares for $d=1$ and $d=2$
- "Our construction is generic and can be extended to higher orders"
- "The ring structure of the refreshing in the general, higher-order case..."
Consolidated Masking Scheme

Local Flaws

2nd-order masking

3rd-order masking
Consolidated Masking Scheme

Local Flaws

- **Local Flaw:** Attack with 3 *standard* probes
- Authors already proposed fix
- **Compositional** security is still open issue

3rd-order masking
Consolidated Masking Scheme

Local Flaws

- **Local Flaw**: Attack with 3 *standard* probes
- Authors already proposed fix
- **Compositional** security is still open issue
Consolidated Masking Scheme

Local Flaws

- **Local Flaw**: Attack with 3 *standard* probes
- Authors already proposed fix
- **Compositional** security is still open issue

**In Paper**: Domain-Oriented Masking

\((\lceil d/2 \rceil + 1)\text{th-order flaw with } extended \text{ probes for DOM}-dep \text{ multiplication}\)
Section 3

Compositional Flaws
Generic Low-Latency Masking

Compositional Flaws

- Introduced at CHES 2018
- Proposes to use CMS refresh $\mathcal{R}$
- Suffers from same flaws
  - Local Flaw
  - Compositional Flaw
- Fix requires secure refresh algorithm with low-latency
Generic Low-Latency Masking
Compositional Flaws

- Introduced at CHES 2018
- Proposes to use CMS refresh $\mathcal{R}$
- Suffers from same flaws
  - Local Flaw
  - Compositional Flaw
- Fix requires secure refresh algorithm with low-latency

In Paper: Unified Masking Approach
A systematic composability flaw
On the Need of the Robust Probing Model

Compositional Flaws

- Security depends on combinatorial combinations, refreshs, register stages
- Not sufficient to solve glitch-resistance and composability separately
- Example: Non-completeness and SNI
On the Need of the Robust Probing Model

Compositional Flaws

- Security depends on combinatorial combinations, refreshs, register stages
- Not sufficient to solve **glitch-resistance** and **composability** separately
- Example: Non-completeness and SNI
- Solution: Unified model
- Note: TI can be composable, but hard to formally prove for higher orders
Section 4

Practical Impact
Experiments
Practical Impact

- SAKURA-G (Spartan-6 FPGA), Clock: 6 MHz, Sampling: 500 MS/s
- Leakage detection with fixed-vs-random $t$-test

Results:
- All flaws are practically **detectable** / **Not necessarily reduce** practical security
- Bias caused by the flaws have low amplitude
- All order reductions multivariate
**Experiments**

**Practical Impact**

- SAKURA-G (Spartan-6 FPGA), Clock: 6 MHz, Sampling: 500 MS/s
- Leakage detection with fixed-vs-random t-test

**Results:**

- All flaws are practically **detectable** / **Not** necessarily **reduce** practical security
- Bias caused by the flaws have low amplitude
- All order reductions multivariate

![Graph](c) 3rd-order multivariate (CMS)

![Graph](d) 4th-order univariate (CMS)
Composability in Hardware - A Matter of Registers

Practical Impact

- Register placement is essential
- Used by TI glitch propagation
- For DOM initially claimed that the DOM-\textit{indep} multiplier does not require output registers
- Without output registers (red) the construction is not composable
- Pipeline registers can be important
Section 5

Conclusion
Summary

Conclusion

- Extensive security proofs not yet established in HW masking
- Lack of appropriate model for higher orders and composability
Summary

Conclusion

- Extensive security proofs not yet established in HW masking
- Lack of appropriate model for higher orders and composability

Our results show:

- No HW masking provides **local and compositional** higher-order security
- Practical impact could be **limited**, flaws are still an **undesirable** source of risk
- **Currently:** Only adapted DOM-\textit{indep} multiplication was robustly proven secure
Summary

Conclusion

• Extensive security proofs not yet established in HW masking
• Lack of appropriate model for higher orders and composability

Our results show:

• No HW masking provides \textit{local and compositional} higher-order security
• Practical impact could be \textit{limited}, flaws are still an \textit{undesirable} source of risk
• \textbf{Currently:} Only adapted DOM-\textit{indep} multiplication was robustly proven secure

In the future:

• Fix flaws and prove existing schemes
• Design new (improved) schemes
Thank you for your attention.

Any questions?
Section 6

Backup
Security Notions

Example:

\[ F \]

\[
\begin{align*}
\{ x_1, x_2, \ldots, x_n \} & \quad \rightarrow \quad \{ y_1, y_2, \ldots, y_n \} \\
\text{input shares} & \quad \rightarrow \quad \text{output shares}
\end{align*}
\]

- Simulate with:
  - NI: \( 2 + 1 = 3 \)
  - SNI: \( 2 = 2 \)
- Enables reasoning about secure composition of modules
- Has been used to prove various SW-oriented masked algorithms/gadgets
- Alternative notions allow trade-offs, e.g., PINI

---

Security Notions

Backup

Example:

\[
\begin{align*}
\text{input shares:} & \quad \{x_1, x_2, \ldots, x_n\} \\
\text{output shares:} & \quad \{y_1, y_2, \ldots, y_n\}
\end{align*}
\]

\[
F \quad t_1 \quad t_2
\]

• NI: 2 + 1 = 3
• SNI: 2 = 2

Simulate with

• Enables reasoning about secure composition of modules
• Has been used to prove various SW-oriented masked algorithms/gadgets
• Alternative notions allow trade-offs, e.g., PINI

---

\[3\] G. Cassiers, F.-X. Standaert, *Trivially and Efficiently Composing Masked Gadgets with Probe Isolating Non-Interference*, eprint 2018/438
Security Notions

Example:

\[ \begin{align*}
  \text{input shares} & \quad \{ x_1, x_2, \ldots, x_n \} \\
  \text{output shares} & \quad \{ y_1, y_2, \ldots, y_n \}
\end{align*} \]

Simulate with

- **NI:** $2 + 1 = 3$
- **SNI:** $2 = 2$

- Enables reasoning about secure composition of modules
- Has been used to prove various SW-oriented masked algorithms/gadgets
- Alternative notions allow trade-offs, e.g., PINI

---

Security Notions
Backup

Example:

\[
\begin{align*}
\text{input shares} & : \left\{ x_1, x_2, \ldots, x_n \right\} \\
\text{output shares} & : \left\{ y_1, y_2, \ldots, y_n \right\}
\end{align*}
\]

Simulate with
- **NI**: \(2 + 1 = 3\)
- **SNI**: \(2 = 2\)

\[\text{input shares.}\]

---

Security Notions
Backup

Example:

\[\begin{align*}
X_1 & \rightarrow t_1 \\
X_2 & \rightarrow t_1 \\
\cdots & \rightarrow \cdots \\
X_n & \rightarrow t_1 \\
\end{align*}\]

\[\begin{align*}
F & \rightarrow y_1 \\
F & \rightarrow y_2 \\
\cdots & \rightarrow \cdots \\
F & \rightarrow y_n \\
\end{align*}\]

Simulate with
- **NI**: \(2 + 1 = 3\)
- **SNI**: \(2 = 2\)

- Enables reasoning about secure composition of modules
- Has been used to prove various SW-oriented masked algorithms/gadgets
- Alternative notions allow trade-offs, e.g., PINI\(^3\)

\[^3\] G. Cassiers, F.-X. Standaert, *Trivially and Efficiently Composing Masked Gadgets with Probe Isolating Non-Interference*, eprint 2018/438