A Methodology to Achieve Provable Side-Channel Security in Real-World Implementations

Sonia Belaïd¹, Gaëtan Cassiers⁴, Camille Mutschler²,⁵, Matthieu Rivain¹, Thomas Roche², François-Xavier Standaert³, Abdel Rahman Taleb¹,⁶

¹CryptoExperts, France
²NinjaLab, France
³UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
⁴TU Graz
⁵LIRMM, Univ. Montpellier, CNRS, Montpellier, France
⁶Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Journées C2
19/10/2023
Side-Channel Attacks

- Plaintext
- Encryption Algorithm
- Secret Key
- Device (e.g. Smartcard)
- Ciphertext

Side-Channel « Eavesdropping » (late 1990s)

- Black box oracle
- Execution Time
- Power Consumption
- Electromagnetic Radiation
- Memory Cache
Countermeasure

Masking  Chari et al. [CRYPTO'99], Goubin and Patarin [CHES'99]
Countermeasure

Masking  Chari et al. [CRYPTO'99], Goubin and Patarin [CHES'99]

Secret Variable \( x \in \mathbb{F}_2 \) (field)
Countermeasure

Masking  *Chari et al.* [CRYPTO’99], *Goubin and Patarin* [CHES’99]

Secret Variable $x \in \mathbb{F}_2$ (field)

Secret Vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$
Countermeasure

Masking *Chari et al. [CRYPTO’99], Goubin and Patarin [CHES’99]*

Secret Variable $x \in \mathbb{F}_2$ (field)

Encode

Secret Vector $\vec{x'} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ s.t.

\[
x_1 \leftarrow \mathbb{F}_2 \\
\ldots \\
x_{n-1} \leftarrow \mathbb{F}_2
\]
Countermeasure

Masking Chari et al. [CRYPTO'99], Goubin and Patarin [CHES'99]

Secret Variable $x \in \mathbb{F}_2$ (field)

Encode

Secret Vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$

s.t. $x_1 \in \mathbb{F}_2$

$s$ random values

$n - 1$ random values
Countermeasure

Masking Chari et al. [CRYPTO’99], Goubin and Patarin [CHES’99]

Secret Variable $x \in \mathbb{F}_2$ (field)

Secret Vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$

s.t.

$x_1 \leftarrow \mathbb{F}_2$

$\vdots$

$x_{n-1} \leftarrow \mathbb{F}_2$

$n - 1$ random values

Secret recombination

$x_n \leftarrow x - x_1 \ldots - x_{n-1}$
Countermeasure

Masking  Chari et al. [CRYPTO’99], Goubin and Patarin [CHES’99]

Secret Variable $x \in \mathbb{F}_2$ (field)

Encode

Secret Vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$

s.t.

\[ x_1 \leftarrow \mathbb{F}_2 \]
\[ \cdots \]
\[ x_{n-1} \leftarrow \mathbb{F}_2 \]
\[ x_n \leftarrow x - x_1 \cdots - x_{n-1} \]

$\left\lceil \frac{n-1}{2} \right\rceil$ random values

secret recombination

Operations over variables $\mathbb{F}_2$

\[ a, b \bigoplus a + b \]
\[ a, b \bigotimes a \times b \]
Countermeasure

Masking  

Chari et al. [CRYPTO’99], Goubin and Patarin [CHES’99]

Secret Variable $x \in \mathbb{F}_2$ (field)

Encode

Secret Vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$

s.t.

$x_1 \leftarrow \mathbb{F}_2$

... $\leftarrow \mathbb{F}_2$

$n - 1$ random values

secret recombination

$x_n \leftarrow x - x_1 \ldots - x_{n-1}$

Operations over variables $\mathbb{F}_2$

$a, b \oplus a + b$

$a, b \odot a \times b$

Gadgets over masked variables in $\mathbb{F}_2^n$
Countermeasure

Masking Chari et al. [CRYPTO’99], Goubin and Patarin [CHES’99]

Secret Variable \( x \in \mathbb{F}_2 \) (field)

Encode

Secret Vector \( \vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_2^n \)

s.t.

\[ x_1 \leftarrow \mathbb{F}_2 \]

\[ \cdots \]

\[ x_{n-1} \leftarrow \mathbb{F}_2 \]

\[ x_n \leftarrow x - x_1 \ldots - x_{n-1} \]

\( \text{n - 1 random values} \)

\( \text{shares} \)

Operations over variables \( \mathbb{F}_2 \)

\( a, b \underbrace{+}_{\text{Gadget}} a + b \)

\( a, b \underbrace{\times}_{\text{Gadget}} a \times b \)

Gadgets over masked variables in \( \mathbb{F}_2^n \)

\[ (a_1, \ldots, a_n), \underbrace{G_+}_{\text{Gadget}} (c_1, \ldots, c_n) \text{ s.t.} \]

\[ c_1 + \ldots + c_n = a + b \]

\[ (b_1, \ldots, b_n), \underbrace{G_\times}_{\text{Gadget}} (c_1, \ldots, c_n) \text{ s.t.} \]

\[ c_1 + \ldots + c_n = a \times b \]
Security of Masked Implementations
Empirical Approach
Security of Masked Implementations

Empirical Approach

Masked Implementation
Security of Masked Implementations

Empirical Approach

- Leakage detection using statistical analysis
- Mounting well-known attacks from the literature
Security of Masked Implementations
Empirical Approach

Masked Implementation

Leakage detection using statistical analysis

Mounting well-known attacks from the literature

Infer security level
Security of Masked Implementations
Empirical Approach

Masked Implementation → Leakage detection using statistical analysis

Mounting well-known attacks from the literature

Infer security level

How to have formal security guarantees?
Security of Masked Implementations

Leakage Models
Security of Masked Implementations

Leakage Models

Formally define side-channel attackers’ capabilities
Security of Masked Implementations

Leakage Models

Easy to use

Formally define side-channel attackers’ capabilities

Close to reality of physical leakage
Security of Masked Implementations
Leakage Models

Easy to use

$t$-Probing Model

$t$ intermediate variables leak their values

Formally define side-channel attackers’ capabilities

Close to reality of physical leakage
Security of Masked Implementations

Leakage Models

Easy to use

- $t$-Probing Model
  - $t$ intermediate variables leak their values

- $p$-Random Probing Model
  - Each intermediate variable leaks with probability $p$

Formally define side-channel attackers' capabilities

Close to reality of physical leakage
Security of Masked Implementations
Leakage Models

Formally define side-channel attackers' capabilities

Easy to use

\( t \)-Probing Model
\( t \) intermediate variables leak their values

\( p \)-Random Probing Model
each intermediate variable leaks with probability \( p \)

\( \delta \)-Noisy Leakage Model
each intermediate variable leaks a \( \delta \)-noisy function of its value

Close to reality of physical leakage
Security of Masked Implementations

Leakage Models

Formally define side-channel attackers’ capabilities

Easy to use

$t$-Probing Model
$t$ intermediate variables leak their values

$p$-Random Probing Model
each intermediate variable leaks with probability $p$

$\delta$-Noisy Leakage Model
each intermediate variable leaks a $\delta$-noisy function of its value

Security Reduction
Duc et al. [EUROCRYPT14]

Close to reality of physical leakage
Security of Masked Implementations

Leakage Models

Easy to use

- $t$-Probing Model
- $t$ intermediate variables leak their values

$\delta$-Noisy Leakage Model
- each intermediate variable leaks a $\delta$-noisy function of its value

Security Reduction

Duc et al. [EUROCRYPT14]

- $p$-Random Probing Model
  - each intermediate variable leaks with probability $p$

- Noisy Leakage Model
  - each intermediate variable leaks a $\delta$-noisy function of its value

Close to reality of physical leakage

Formally define side-channel attackers’ capabilities
Noisy Leakage Model

Memory

- read $\vec{x}_1$
- write $\vec{y}_1$
- execute op. 1

... (repeated for $k$)

- read $\vec{x}_k$
- write $\vec{y}_k$
- execute op. $k$
Noisy Leakage Model

Memory

execute op. 1

read $\vec{x}_1$

write $\vec{y}_1$

Leaks $f_1(\vec{x}_1)$

$deterministic$ $leakage$ $of$ $each$ $variable$ $+$ $physical$ $noise$

execute op. $k$

read $\vec{x}_k$

write $\vec{y}_k$

Leaks $f_k(\vec{x}_k)$
Noisy Leakage Model

Memory

read \( \vec{x}_1 \)
execute op. 1
write \( \vec{y}_1 \)

\[ \text{Leaks } f_1(\vec{x}_1) \]

\[ \vdots \]

read \( \vec{x}_k \)
execute op. k
write \( \vec{y}_k \)

\[ \text{Leaks } f_k(\vec{x}_k) \]

\( f_i \) is a \( \delta \)-noisy function

- Low \( \delta \) \rightarrow Low leakage
- High \( \delta \) \rightarrow High leakage
Leakage Models
Physical Assumptions & Issues

Sequential execution of operations

$\vec{x}_1 \rightarrow \vec{x}_2 \rightarrow \vec{x}_3 \rightarrow \vec{x}_4$

op. 1 \rightarrow op. 2 \rightarrow op. 3 \rightarrow op. 4
Leakage Models
Physical Assumptions & Issues

Sequential execution of operations

$\vec{x}_1 \rightarrow \vec{x}_2 \rightarrow \vec{x}_3 \rightarrow \vec{x}_4$

op. 1 \rightarrow op. 2 \rightarrow op. 3 \rightarrow op. 4

Each operation leaks during execution
Leakage Models
Physical Assumptions & Issues

Sequential execution of operations

Each operation leaks during execution

Data Isolation Assumption: each operation leakage only depends on its inputs
Leakage Models
Physical Assumptions & Issues
Leakage Models
Physical Assumptions & Issues

CPU register → write $\vec{x}_1$ → $\vec{x}_1$ → write $\vec{x}_2$ → $\vec{x}_2$

Transition effect
Leakage Models
Physical Assumptions & Issues

Leakage depends on $\vec{x}_1 \oplus \vec{x}_2$
Leakage Models
Physical Assumptions & Issues

Leakage depends on $\vec{x}_1 \oplus \vec{x}_2$

Physical effects break the data isolation assumption

Transition effect
Leakage Models
Physical Assumptions & Issues

Sequential execution of operations

\( \vec{x}_1 \rightarrow \vec{x}_2 \rightarrow \vec{x}_3 \rightarrow \vec{x}_4 \)

op. 1 \rightarrow op. 2 \rightarrow op. 3 \rightarrow op. 4
Leakage Models

Physical Assumptions & Issues

Sequential execution of operations

$\vec{x}_1 \rightarrow \vec{x}_2 \rightarrow \vec{x}_3 \rightarrow \vec{x}_4$

- op. 1
- op. 2
- op. 3
- op. 4

Physical noise occurs during side-channel acquisitions
Leakage Models

Physical Assumptions & Issues

Sequential execution of operations

$\vec{x}_1 \rightarrow \text{op. 1}$
$\vec{x}_2 \rightarrow \text{op. 2}$
$\vec{x}_3 \rightarrow \text{op. 3}$
$\vec{x}_4 \rightarrow \text{op. 4}$

Physical noise occurs during side-channel acquisitions

**Noise Independence Assumption:** each noise is independent of the others
Leakage Models

Physical Assumptions & Issues

![Graph showing power consumption over time sample]
Leakage Models

Physical Assumptions & Issues

![Graph showing power consumption over time samples](image-url)
Leakage Models

Physical Assumptions & Issues

\[
\vec{d}(\vec{x}) = (d_1, \ldots, d_{400})
\]
Leakage Models

Physical Assumptions & Issues

\[ \vec{d}(\vec{x}) = (d_1, \ldots, d_{400}) \]

Noise drawn from multivariate Gaussian distribution \( \mathcal{N}(\vec{0}, \Sigma) \)

\[ \vec{z} = (z_1, \ldots, z_{400}) \]
Leakage Models

Physical Assumptions & Issues

Multivariate noise breaks the noise independence assumption

Deterministic leakage

\[ \mathbf{d}(\mathbf{x}) = (d_1, \ldots, d_{400}) \]

Noise drawn from multivariate Gaussian distribution \( \mathcal{N}(\mathbf{0}, \Sigma) \)

\[ \mathbf{z} = (z_1, \ldots, z_{400}) \]
Overview
Overview

Abstract
circuit $C$
Overview

Abstract circuit $C$ → Noisy Leakage Model → $\lambda$ bits of theoretical security
Overview

Abstract circuit $C$ → Noisy Leakage Model → $\lambda$ bits of theoretical security → Implementation on a device

Loss of theoretical security level
Overview

Abstract circuit $C$ $\rightarrow$ Noisy Leakage Model $\rightarrow$ $\lambda$ bits of theoretical security $\rightarrow$ Implementation on a device

Methodology to preserve the security level for an implementation on a device
Overview

Abstract circuit $C \rightarrow$ Noisy Leakage Model $\rightarrow \lambda$ bits of theoretical security $\rightarrow$ Implementation on a device

Methodology to preserve the security level for an implementation on a device

Implement abstract gates on a device
Overview

Abstract circuit $C$ → Noisy Leakage Model → $\lambda$ bits of theoretical security → Implementation on a device

Methodology to preserve the security level for an implementation on a device

Implement abstract gates on a device → Enforce / Relax data isolation
Overview

Abstract circuit $C$ → Noisy Leakage Model → $\lambda$ bits of theoretical security → Implementation on a device

Methodology to preserve the security level for an implementation on a device

Implement abstract gates on a device → Enforce / Relax data isolation → Characterize the leakage distribution
Abstract circuit $C$ → Noisy Leakage Model → $\lambda$ bits of theoretical security → Implementation on a device

Methodology to preserve the security level for an implementation on a device:

Implement abstract gates on a device → Enforce / Relax data isolation → Characterize the leakage distribution

Characterize the leakage distribution

Enforce / Relax noise independence
Overview

Abstract circuit $C$ \rightarrow Noisy Leakage Model \rightarrow $\lambda$ bits of theoretical security \rightarrow Implementation on a device

Methodology to preserve the security level for an implementation on a device

Implement abstract gates on a device \rightarrow Enforce / Relax data isolation \rightarrow Characterize the leakage distribution

Estimate the noisy leakage parameter $\delta$ tolerated by the device

Enforce / Relax noise independence
Overview

Methodology to preserve the security level for an implementation on a device

Implement abstract gates on a device

Enforce / Relax data isolation

Characterize the leakage distribution

Estimate the noisy leakage parameter $\delta$ tolerated by the device

Enforce / Relax noise independence

Compile the implementation
Methodology

Step 1: Implement abstract gates
Methodology
Step 1: Implement abstract gates

respect the format from the leakage models
Methodology

Step 1: Implement abstract gates

respect the format from the leakage models

operation_xor:
   ldr r0, [r0]
   ldr r1, [r1]
   eor r0, r1 r0  // For other operations, change ALU instruction.
   str r0, [r2]
Methodology
Step 2: Enforce / Relax data isolation
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs → use data whitening
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs ➔ use data whitening

\[
\text{operation}_1(a_1, b_1)
\]

\[
\text{operation}_2(a_2, b_2)
\]
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs → use data whitening

operation_1(a_1, b_1)

whitening()

operation_2(a_2, b_2)

whitening()
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs

\[
\text{operation}_1(a_1, b_1) \quad \text{use data whitening}
\]

\[
\text{whitening}()
\]

\[
\text{operation}_2(a_2, b_2)
\]

\[
\text{whitening}()
\]

\[
\text{clean data path and registers from previous calls}
\]
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs

use data whitening

operation_1(a₁, b₁)  whitening()  operation_2(a₂, b₂)  whitening()

clean data path and registers from previous calls

Example: call same operation with random inputs
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs → use data whitening

operation_1\((a_1, b_1)\)

whitening()

whitening()

operation_2\((a_2, b_2)\)

clean data path and registers from previous calls

Example: call same operation with random inputs

Effectiveness depends on CPU micro-architecture
Methodology
Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs

use data whitening

operation_1(a_1, b_1)

whitening()

operation_2(a_2, b_2)

whitening()

clean data path and registers from previous calls

Example: call same operation with random inputs

Effectiveness depends on CPU micro-architecture

How to check if it works?
Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs

use data whitening

operation_1\((a_1, b_1)\)

whitening()

clean data path and registers from previous calls

operation_2\((a_2, b_2)\)

whitening()

Example: call same operation with random inputs

Effectiveness depends on CPU micro-architecture

How to check if it works? we propose a novel statistical test to (in)validate the assumption on a device
Methodology

Step 3: Characterize the Leakage Distribution
Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature
Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

Operation with input $\vec{x}$

$$\vec{y} = d(\vec{x}) + \mathcal{N}(\vec{0}, \Sigma)$$
Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

Operation with input $\vec{x}$

Leakage $\vec{y} = d(\vec{x}) + \mathcal{N}(\vec{0}, \Sigma)$

1. Infer $d_i(\cdot)$ for each time sample $i$
Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

Operation with input $\vec{x}$

$\vec{y} = \vec{d}(\vec{x}) + \mathcal{N}(\vec{0}, \Sigma)$

1. Infer $d_i(\cdot)$ for each time sample $i$

2. Compute the covariance matrix $\Sigma$
Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

Operation with input \( \overrightarrow{x} \)

Leakage

\[ \overrightarrow{y} = \overrightarrow{d}(\overrightarrow{x}) + \mathcal{N}(\overrightarrow{0}, \Sigma) \]

1. Infer \( d_i(\cdot) \) for each time sample \( i \)

2. Compute the covariance matrix \( \Sigma \)

Linear Regression

Machine Learning

...
Methodology

Step 4: Enforce / Relax Noise Independence
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice → We propose to relax it
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice  We propose to relax it

start

Leakage trace \( \hat{Y} \)

operation_1

operation_2

operation_3

dend
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice → We propose to relax it

\[ \vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N} \]

Data isolation assumption
each \( S_i \) is only the leakage of operation \( _i \)

Leakage trace \( \vec{Y} \)

start

operation_1

operation_2

operation_3

end
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice → We propose to relax it

\[ \vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N} \]

data isolation assumption
each \( S_i \) is only the leakage of operation\(_i\)

Split
\[ \vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3 \]

instead of having time-separated noises

Leakage trace \( \vec{Y} \)

start

operation_1

operation_2

operation_3

end
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice → We propose to relax it

\[ \vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N} \]

-split instead of having time-separated noises

\[ \vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3 \]

Then the leakage is split into

\[ \{ \vec{Y}_i = \vec{S}_i + \vec{N}_i \}_i=1,2,3 \]

data isolation assumption

each \( S_i \) is only the leakage of operation\(_i\)
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice \[\Rightarrow\] We propose to relax it

\[\vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N}\]

Split \[\vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3\]

instead of having time-separated noises

Trivial: \[\vec{N}_i = \frac{1}{3} \vec{N}\]
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice  →  We propose to relax it

\[ \vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N} \]

Split \[ \vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3 \]

instead of having time-separated noises

Trivial: \[ \vec{N}_i = \frac{1}{3} \vec{N} \]

Drawback: more operations  →  less noise on each operation  →  more leakage  →  lower security level in the leakage models
Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice $\rightarrow$ We propose to relax it

\[ \vec{Y} = \vec{S}_1 + \vec{S}_2 + \vec{S}_3 + \vec{N} \]

Split \( \vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3 \)

instead of having time-separated noises

Trivial: \( \vec{N}_i = \frac{1}{3} \vec{N} \)

Drawback: more operations $\implies$ less noise on each operation $\implies$ more leakage $\implies$ lower security level in the leakage models

Optimization problem: how to rewrite \( \vec{N} = \vec{N}_1 + \vec{N}_2 + \vec{N}_3 \), such as to minimize the information leakage of the different operations?
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$

$$(p, \varepsilon)$$—random probing security $\implies$ $\delta$—noisy leakage security
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$

$(p, \varepsilon)$—random probing security $\implies \delta$—noisy leakage security

Efficient way to compute $\delta$ on a device
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$

$(p, \epsilon)$—random probing security $\implies$ $\delta$—noisy leakage security

Efficient way to compute $\delta$ on a device

Infer tolerated leakage probability $p$ by the device
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$

$$(p, \varepsilon)$$—random probing security $\implies$ $\delta$—noisy leakage security

Efficient way to compute $\delta$ on a device

Infer tolerated leakage probability $p$ by the device

Which random probing secure gadgets from the literature can be used on the device
Methodology

Step 5: Estimate the noisy leakage parameter $\delta$

$(p, \varepsilon)$—random probing security $\implies$ $\delta$—noisy leakage security

Efficient way to compute $\delta$ on a device

Infer tolerated leakage probability $p$ by the device

Which random probing secure gadgets from the literature can be used on the device

Best gadgets from the literature tolerate $p \approx 2^{-7}$
Methodology
Wrap-up
Methodology

Wrap-up

Characterization

Device
Methodology

Wrap-up

Characterization

Device →

1. Implementing elementary operations
Methodology

Wrap-up

Characterization

Device

① Implementing elementary operations

② Enforcing / relaxing data isolation

Implemented gates
Methodology

Wrap-up

Characterization

1. Implementing elementary operations

2. Enforcing / relaxing data isolation

Test of data isolation for each pair

Device

Implemented gates

Side-channel acquisition tool
Methodology

Wrap-up

Characterization

1. Implementing elementary operations

2. Enforcing / relaxing data isolation

Test of data isolation for each pair

Addition of whitening

Nok

Device ➔ Implemented gates ➔ Side-channel acquisition tool
Methodology

Wrap-up

Characterization

1. Implementing elementary operations
2. Enforcing / relaxing data isolation

Device → Implemented gates → Test of data isolation for each pair → Gates

Addition of whitening

Side-channel acquisition tool
Methodology

Wrap-up

Characterization

Device

1. Implementing elementary operations

2. Enforcing / relaxing data isolation

Test of data isolation for each pair

Addition of whitening

Gates

Implemented gates

Ok

③ Characterizing the leakage

Side-channel acquisition tool
Methodology

Wrap-up

Characterization

1. Implementing elementary operations

2. Enforcing / relaxing data isolation

3. Characterizing the leakage

4. Enforcing / relaxing noise independence

Test of data isolation for each pair

Ok Gates

Addition of whitening

Nok

Device

Implemented gates

Side-channel acquisition tool
Methodology

Wrap-up

Characterization

1. Implementing elementary operations
   → Test of data isolation for each pair
   → Addition of whitening

2. Enforcing / relaxing data isolation

3. Characterizing the leakage

4. Enforcing / relaxing noise independence

5. Estimating the noisy leakage parameter

Device

Implement gates

Ok

Nok

Side-channel acquisition tool
Methodology

Wrap-up

Characterization

Device

① Implementing elementary operations

Implemented gates

② Enforcing / relaxing data isolation

Test of data isolation for each pair

Ok

Gates

Addition of whitening

Nok

③ Characterizing the leakage

④ Enforcing / relaxing noise independence

⑤ Estimating the noisy leakage parameter

Side-channel acquisition tool

⑥ Compilation

Probability $p$
Methodology

Wrap-up

Characterization

1. Implementing elementary operations
2. Enforcing / relaxing data isolation
3. Characterizing the leakage
4. Enforcing / relaxing noise independence
5. Estimating the noisy leakage parameter
6. Compilation

Device → Implemented gates → Test of data isolation for each pair → Characterizing the leakage → Enforcing / relaxing noise independence → Estimating the noisy leakage parameter

Side-channel acquisition tool

Probability \( p \)

Circuit \( C \) → Security level \( \lambda \)
Methodology

Wrap-up

**Characterization**

1. Implementing elementary operations
2. Enforcing / relaxing data isolation
3. Characterizing the leakage
4. Enforcing / relaxing noise independence
5. Estimating the noisy leakage parameter

**Compilation**

6. Random probing compiler

Random probing compiler: replaces each gate by a \( n \)-share \((p, \varepsilon)\) — random probing secure gadget.
Methodology

Wrap-up

Characterization

1. Implementing elementary operations
2. Enforcing / relaxing data isolation
3. Characterizing the leakage
4. Enforcing / relaxing noise independence
5. Estimating the noisy leakage parameter

Compilation

Circuit $C$
Security level $\lambda$

Random probing compiler

Implementation on device

Side-channel acquisition tool

Random probing compiler: replaces each gate by a $n$-share $(p, \varepsilon)$—random probing secure gadget
Random probing compiler: replaces each gate by a $n$-share $(p, \varepsilon)$—random probing secure gadget
Limitations / Questions
• Noise levels are critical for security levels → we test a component and show that it is not suited for the use-case
Limitations / Questions

• Noise levels are critical for security levels → we test a component and show that it is not suited for the use-case
  • How to achieve high physical noise when designing hardware?
Limitations / Questions

• Noise levels are critical for security levels → we test a component and show that it is not suited for the use-case
  • How to achieve high physical noise when designing hardware?
• Can we make leakage models and security proofs tighter?
Limitations / Questions

• Noise levels are critical for security levels → we test a component and show that it is not suited for the use-case
  • How to achieve high physical noise when designing hardware?
• Can we make leakage models and security proofs tighter?
• Can we solve the remaining limitations of the approach?
Limitations / Questions

• Noise levels are critical for security levels → we test a component and show that it is not suited for the use-case
  • How to achieve high physical noise when designing hardware?
• Can we make leakage models and security proofs tighter?
• Can we solve the remaining limitations of the approach?
• …
Thank you!
Any questions?

https://eprint.iacr.org/2023/1198