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

Sonia Belaïd[1], Gaëtan Cassiers[4], Camille Mutschler[2,5], Matthieu Rivain[1], Thomas Roche[2], François-Xavier Standaert[3], **Abdel Rahman Taleb**[1,6]

[1]CryptoExperts, France

[2]NinjaLab, France

[3]UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium

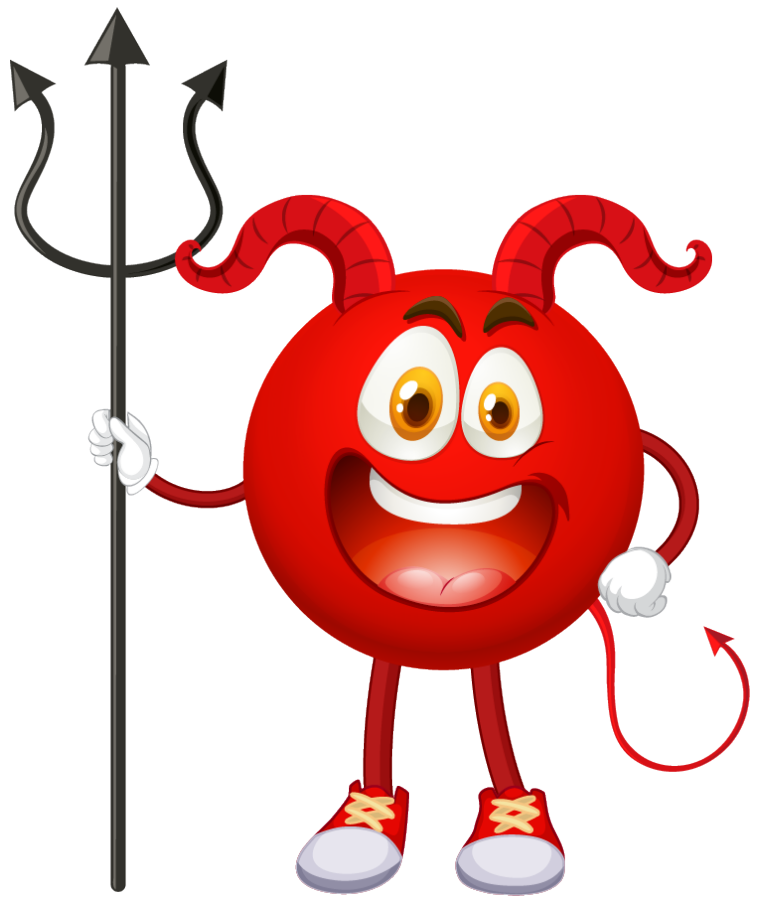
[4]TU Graz

[5]LIRMM, Univ. Montpellier, CNRS, Montpellier, France

[6]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Journées C2
19/10/2023

Side-Channel Attacks



Device (e.g. Smartcard)

Secret Key

Encryption Algorithm

Black box oracle

Plaintext

Ciphertext

Side-Channel « Eavesdropping »

(late 1990s)

Execution Time

Power Consumtion

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)

Encode

shares

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

Countermeasure

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

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

$$\Big\downarrow \text{Encode}$$

shares

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

s.t.

$$x_1 \xleftarrow{\$} \mathbb{F}_2$$

$$\ldots$$

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

Countermeasure

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

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

Encode $\downarrow$

shares

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

s.t.

$x_1 \xleftarrow{\$} \mathbb{F}_2$

$\ldots$

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

$n - 1$ random values

Countermeasure

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

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

Encode

shares

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

s.t.

$x_1 \xleftarrow{\$} \mathbb{F}_2$

$\ldots$

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

$n-1$ 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

shares

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

s.t.

$x_1 \xleftarrow{\$} \mathbb{F}_2$

$\ldots$

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

$n-1$ random values

secret recombination

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

Operations over variables $\mathbb{F}_2$

a, b $+$ $a + b$

a, b $\times$ $a \times b$

Countermeasure

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

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

Encode

shares

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

s.t.

$x_1 \xleftarrow{\$} \mathbb{F}_2$

$\ldots$

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

$n-1$ random values

secret recombination

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

Operations over variables $\mathbb{F}_2$

a, b $+$ $a + b$

a, b $\times$ $a \times b$

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)

$\downarrow$ Encode

shares

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

s.t.

$x_1 \xleftarrow{\$} \mathbb{F}_2$

$\ldots$

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

$n-1$ random values

secret recombination

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

Operations over variables $\mathbb{F}_2$

$a, b \; + \; a + b$

$a, b \; \times \; a \times b$

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

$(a_1, \ldots, a_n), (b_1, \ldots, b_n)$ $\boxed{G_+}$ $(c_1, \ldots, c_n)$ s.t. $c_1 + \ldots + c_n = a + b$

$(a_1, \ldots, a_n), (b_1, \ldots, b_n)$ $\boxed{G_\times}$ $(c_1, \ldots, c_n)$ 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

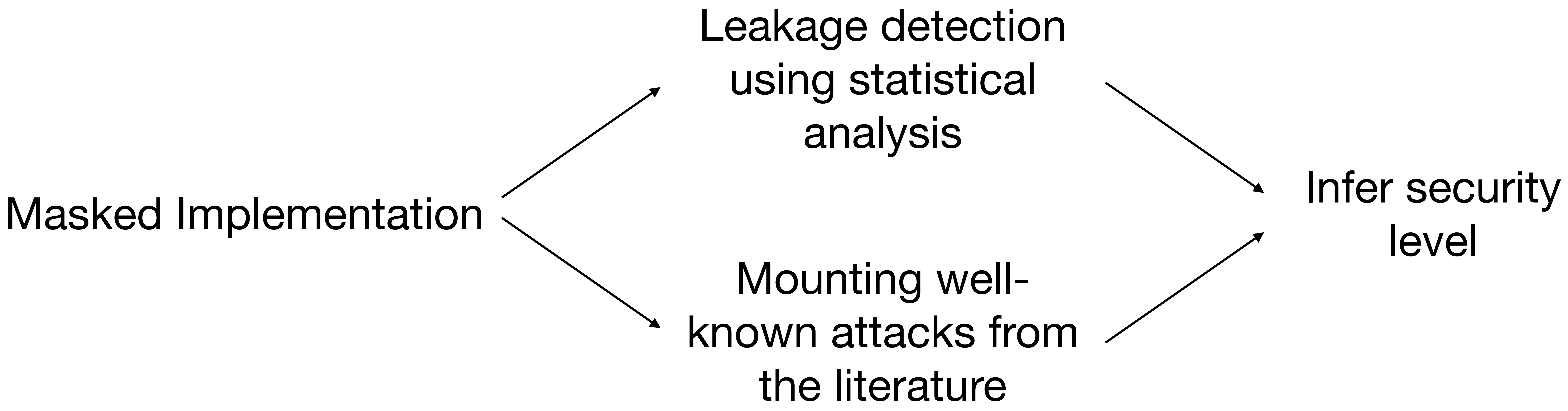
Masked Implementation

Leakage detection using statistical analysis

Mounting well-known attacks from the literature

Security of Masked Implementations

Empirical Approach



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

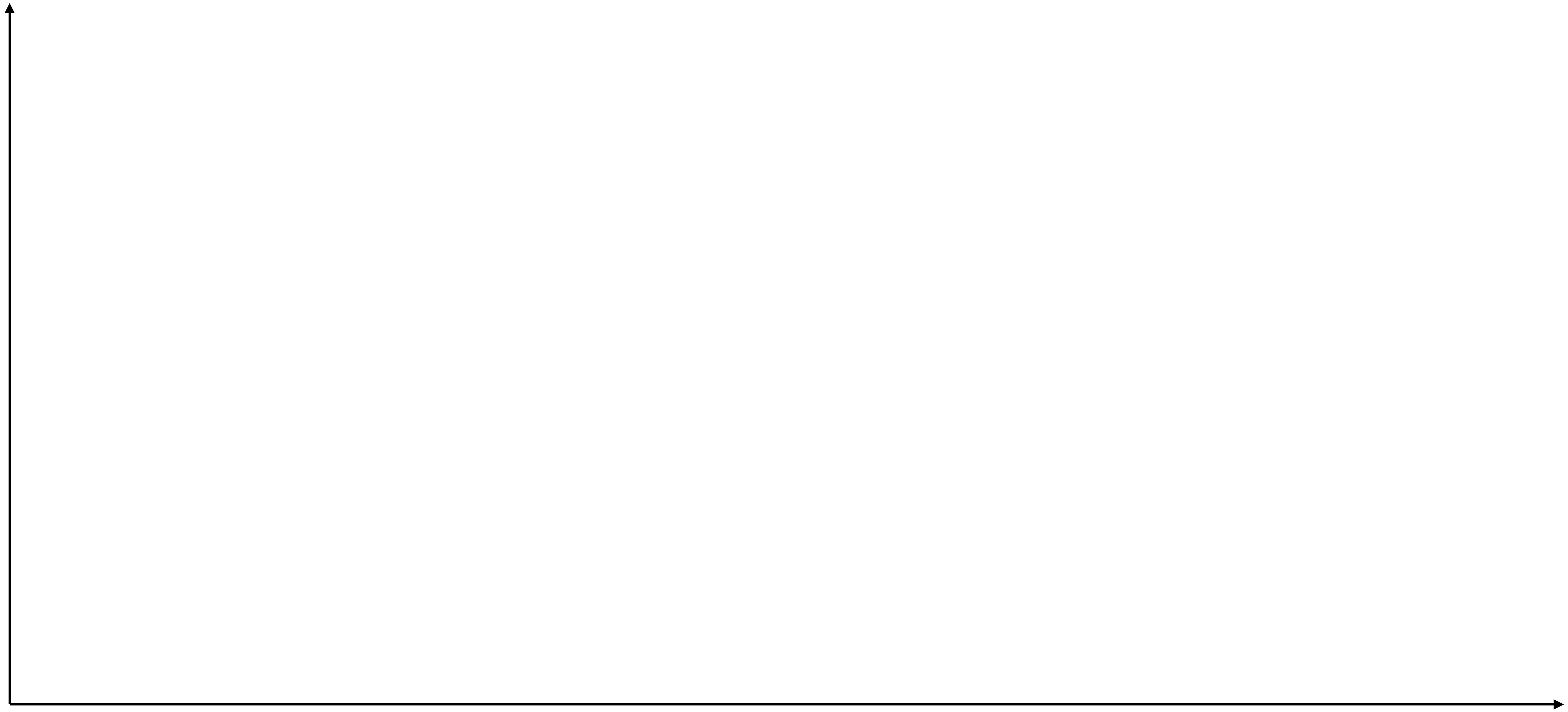
Leakage Models

Formally define side-channel attackers' capabilities

Security of Masked Implementations

Leakage Models

Formally define side-channel attackers' capabilities



Security of Masked Implementations

Leakage Models

Formally define side-channel attackers' capabilities

Easy to use

t-Probing Model

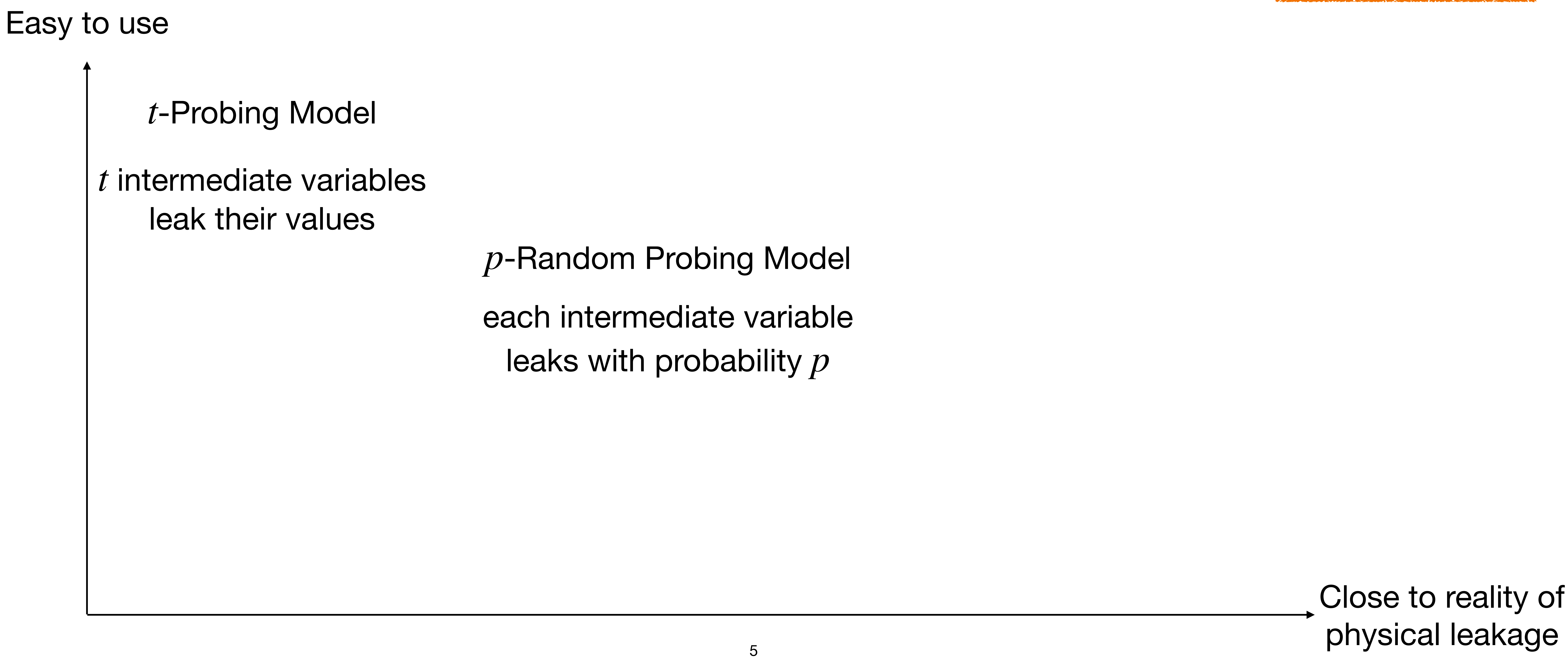
t intermediate variables leak their values

Close to reality of physical leakage

Security of Masked Implementations

Leakage Models

Formally define side-channel attackers' capabilities



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

δ-Noisy Leakage Model

each intermediate variable leaks a δ-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

Security Reduction
Duc et al. [EUROCRYPT14]

t-Probing Model

t intermediate variables leak their values

p-Random Probing Model

each intermediate variable leaks with probability p

δ-Noisy Leakage Model

each intermediate variable leaks a δ-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

Security Reduction
Duc et al. [EUROCRYPT14]

t-Probing Model

t intermediate variables leak their values

p-Random Probing Model

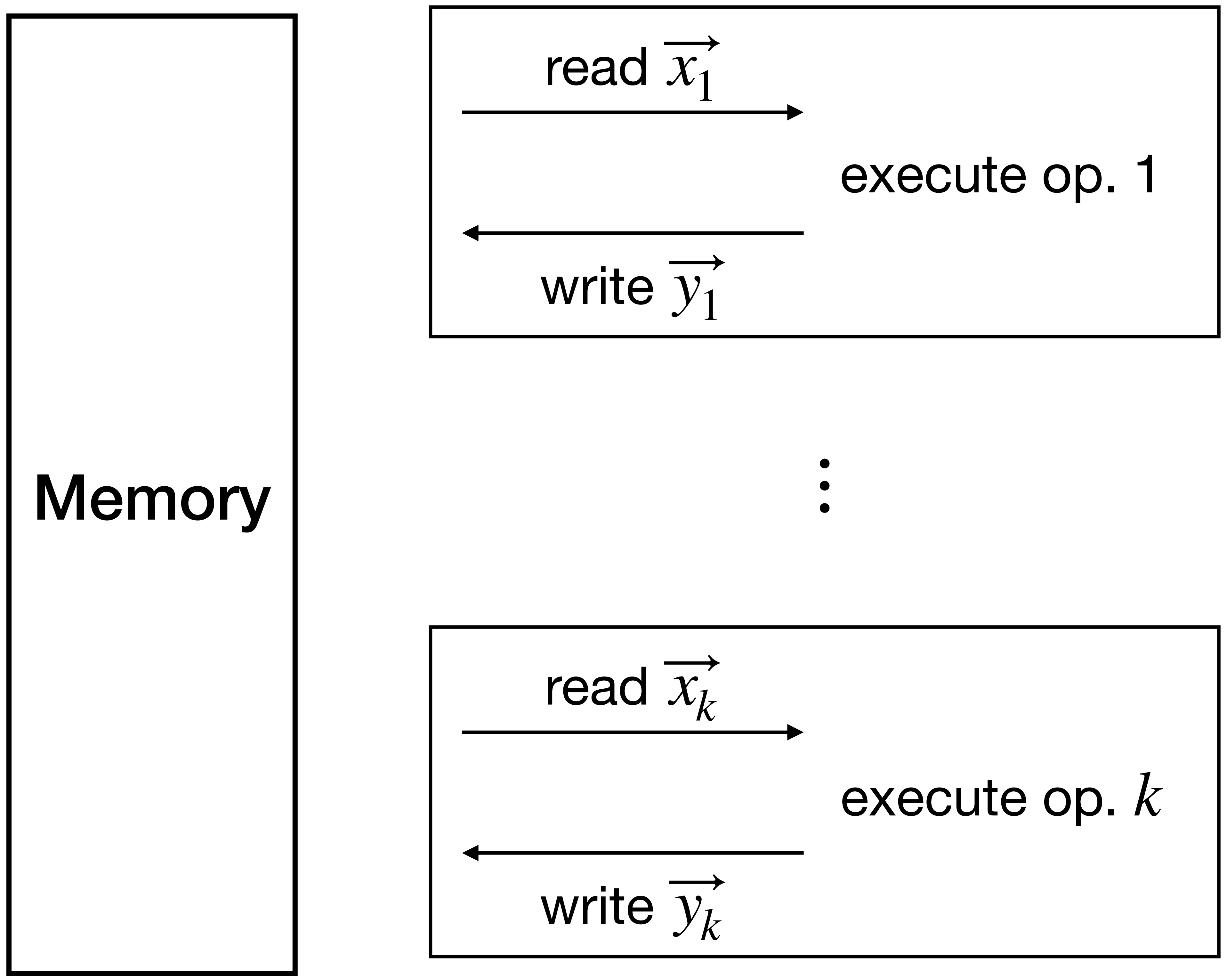
each intermediate variable leaks with probability p

δ-Noisy Leakage Model

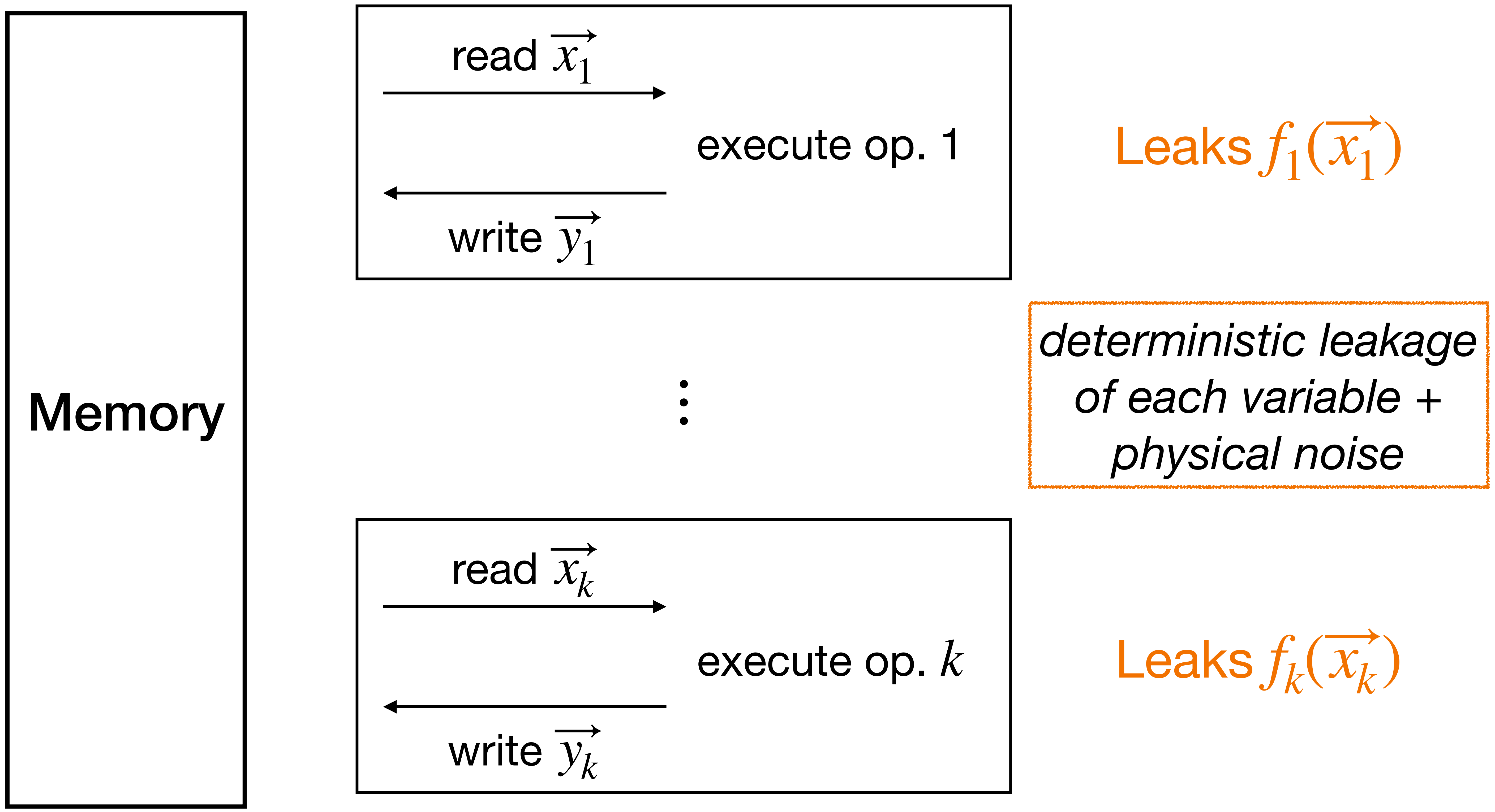
each intermediate variable leaks a δ-noisy function of its value

Close to reality of physical leakage

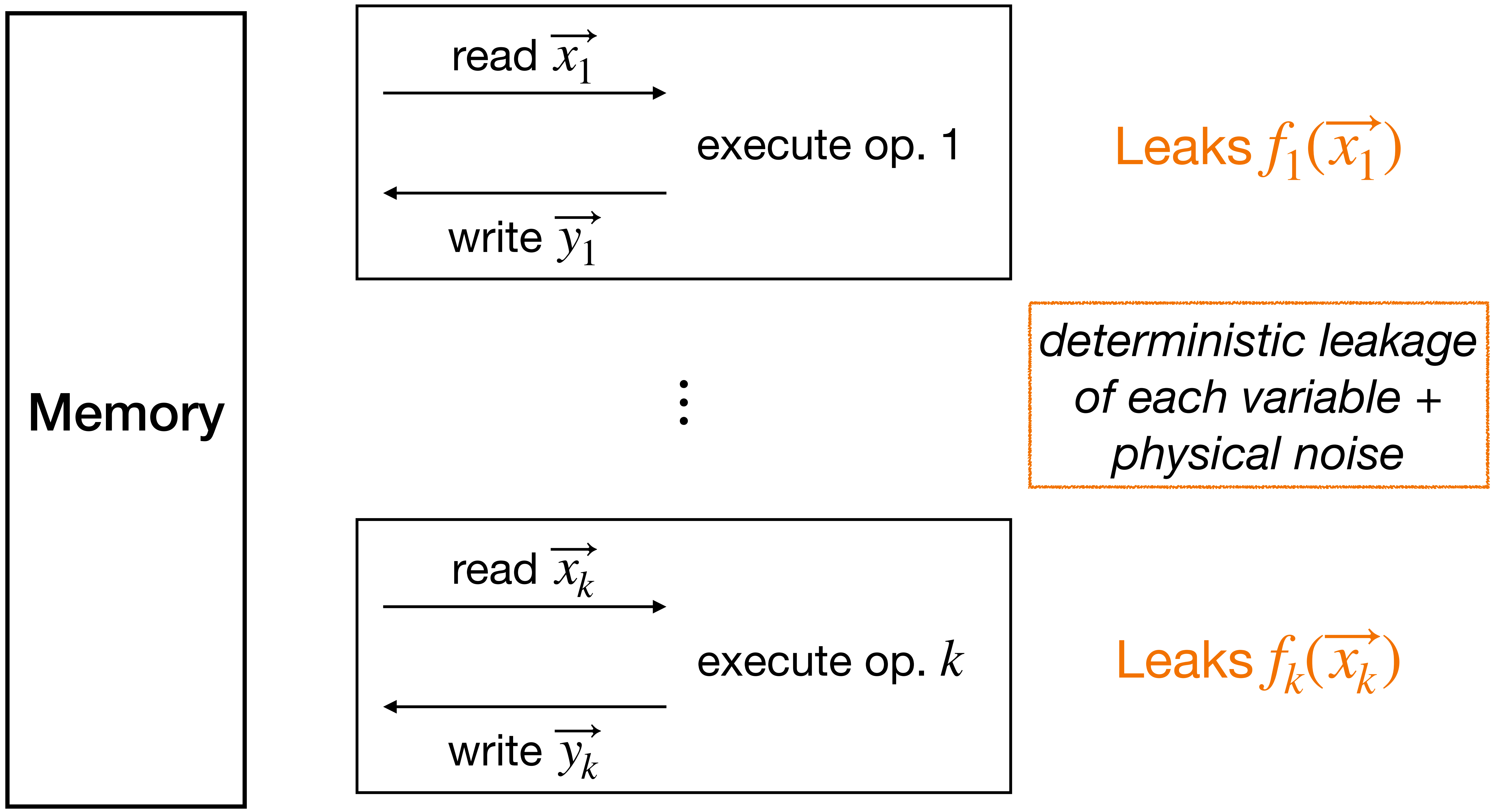
Noisy Leakage Model



Noisy Leakage Model



Noisy Leakage Model



f_i is a δ-noisy function

Low δ $\longrightarrow$ Low leakage

High δ $\longrightarrow$ High leakage

Leakage Models

Physical Assumptions & Issues

Sequential execution
of operations

$\overrightarrow{x_1}$ → op. 1

$\overrightarrow{x_2}$ → op. 2

$\overrightarrow{x_3}$ → op. 3

$\overrightarrow{x_4}$ → op. 4

Leakage Models

Physical Assumptions & Issues

Each operation leaks during execution

Sequential execution of operations $\longrightarrow$

$\overrightarrow{x_1}$	$\overrightarrow{x_2}$	$\overrightarrow{x_3}$	$\overrightarrow{x_4}$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
op. 1	op. 2	op. 3	op. 4

Leakage Models

Physical Assumptions & Issues

Each operation leaks during execution

Sequential execution of operations

$\vec{x_1}$ → op. 1 — Leakage only on $\vec{x_1}$

$\vec{x_2}$ → op. 2 — Leakage only on $\vec{x_2}$

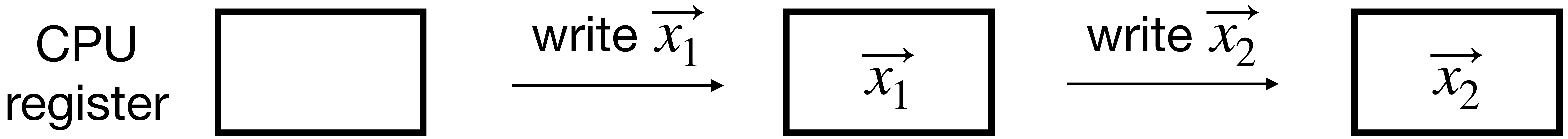
$\vec{x_3}$ → op. 3 — Leakage only on $\vec{x_3}$

$\vec{x_4}$ → op. 4 — Leakage only on $\vec{x_4}$

Data Isolation Assumption: each operation leakage only depends on its inputs

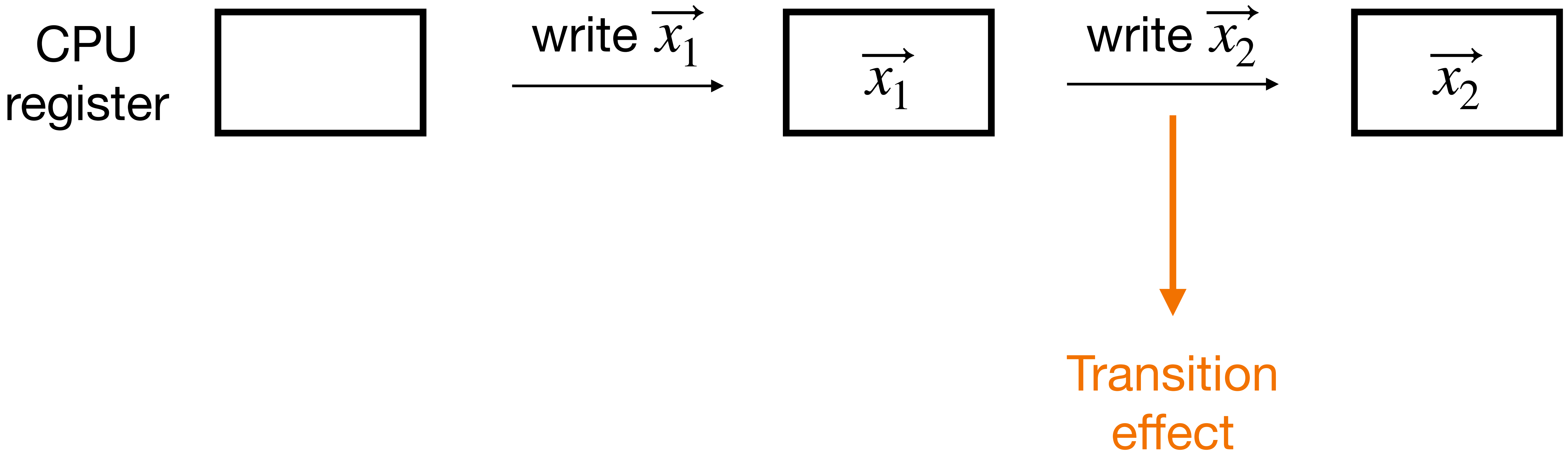
Leakage Models

Physical Assumptions & Issues



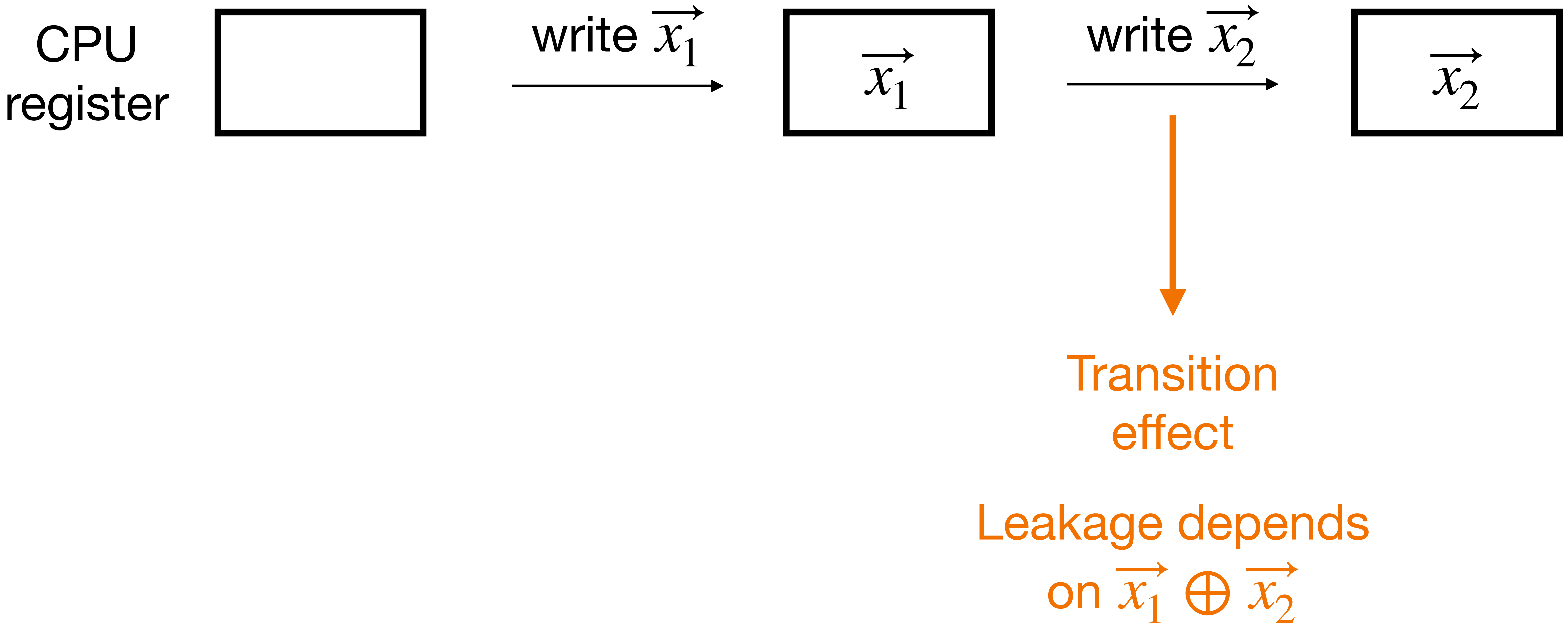
Leakage Models

Physical Assumptions & Issues



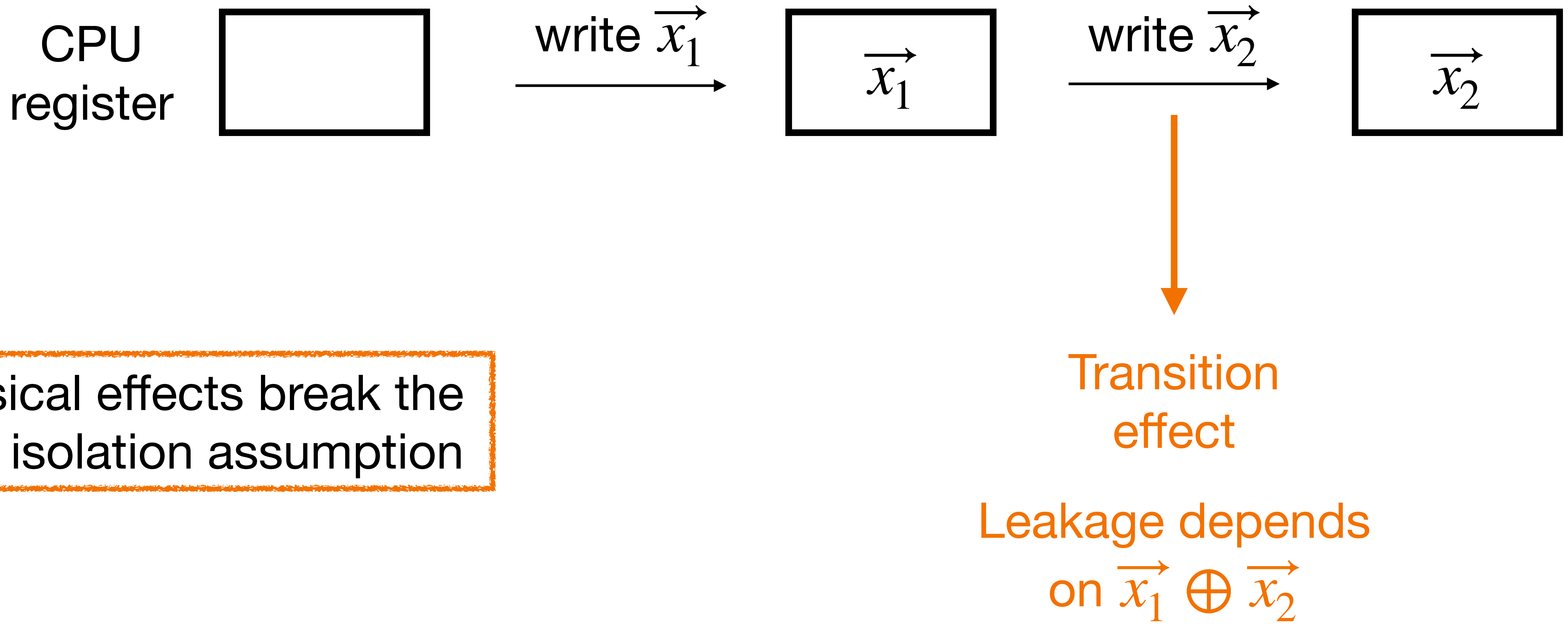
Leakage Models

Physical Assumptions & Issues



Leakage Models

Physical Assumptions & Issues



Physical effects break the data isolation assumption

Leakage Models

Physical Assumptions & Issues

Sequential execution
of operations
$\longrightarrow$

$$
\begin{array}{cccc}
\overrightarrow{x_1} & \overrightarrow{x_2} & \overrightarrow{x_3} & \overrightarrow{x_4} \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\text{op. 1} & \text{op. 2} & \text{op. 3} & \text{op. 4}
\end{array}
$$

Leakage Models

Physical Assumptions & Issues

Physical noise occurs during side-channel acquisitions

Sequential execution of operations

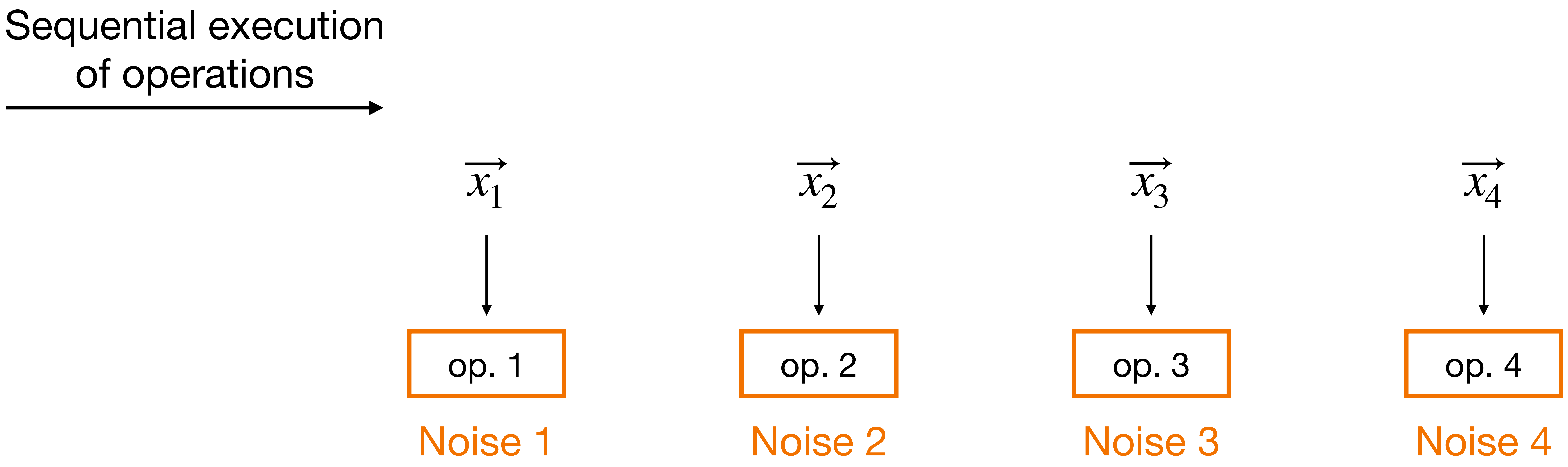
$\overrightarrow{x_1}$ $\overrightarrow{x_2}$ $\overrightarrow{x_3}$ $\overrightarrow{x_4}$

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

Leakage Models

Physical Assumptions & Issues

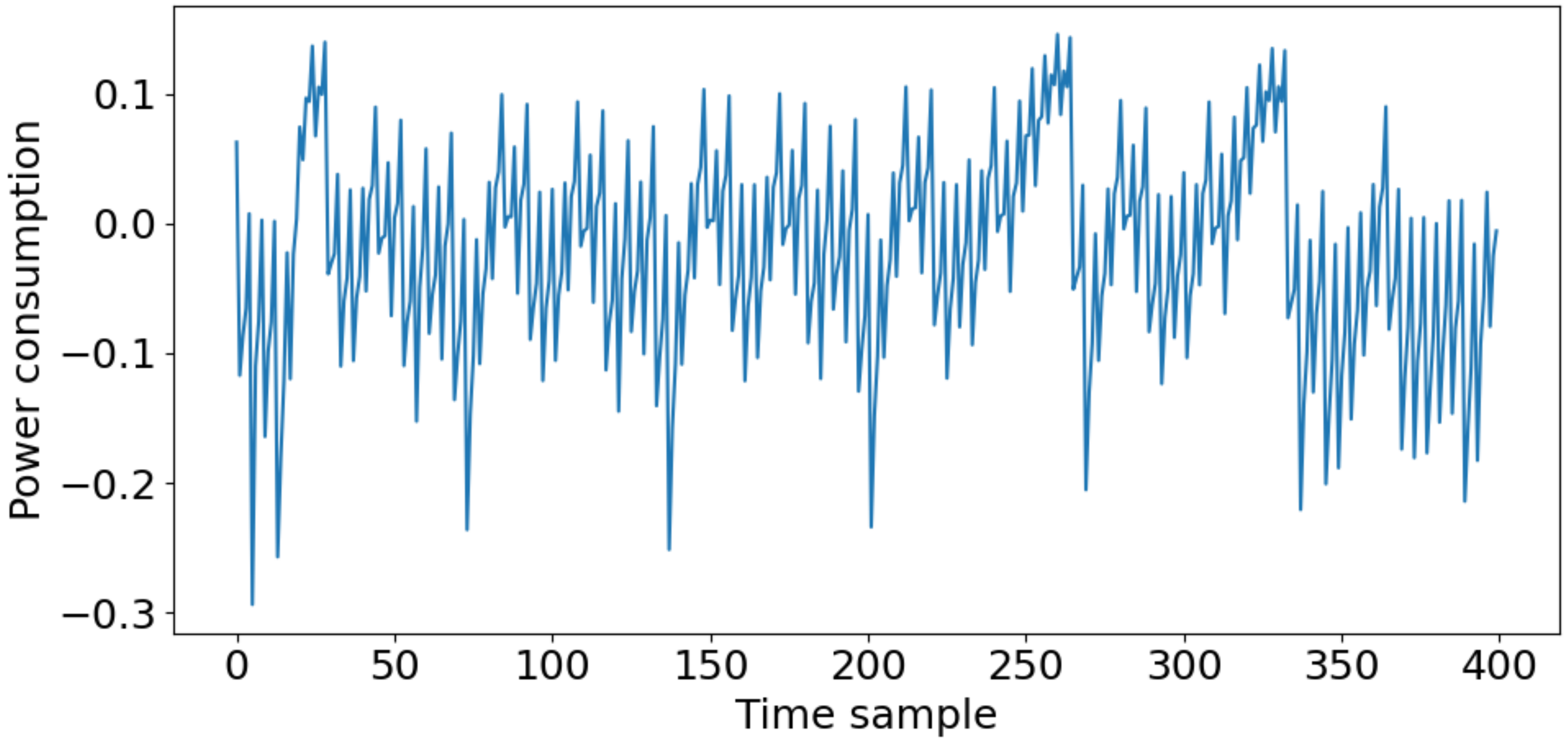
Physical noise occurs during side-channel acquisitions



<u>Noise Independence Assumption:</u> each noise is independent of the others

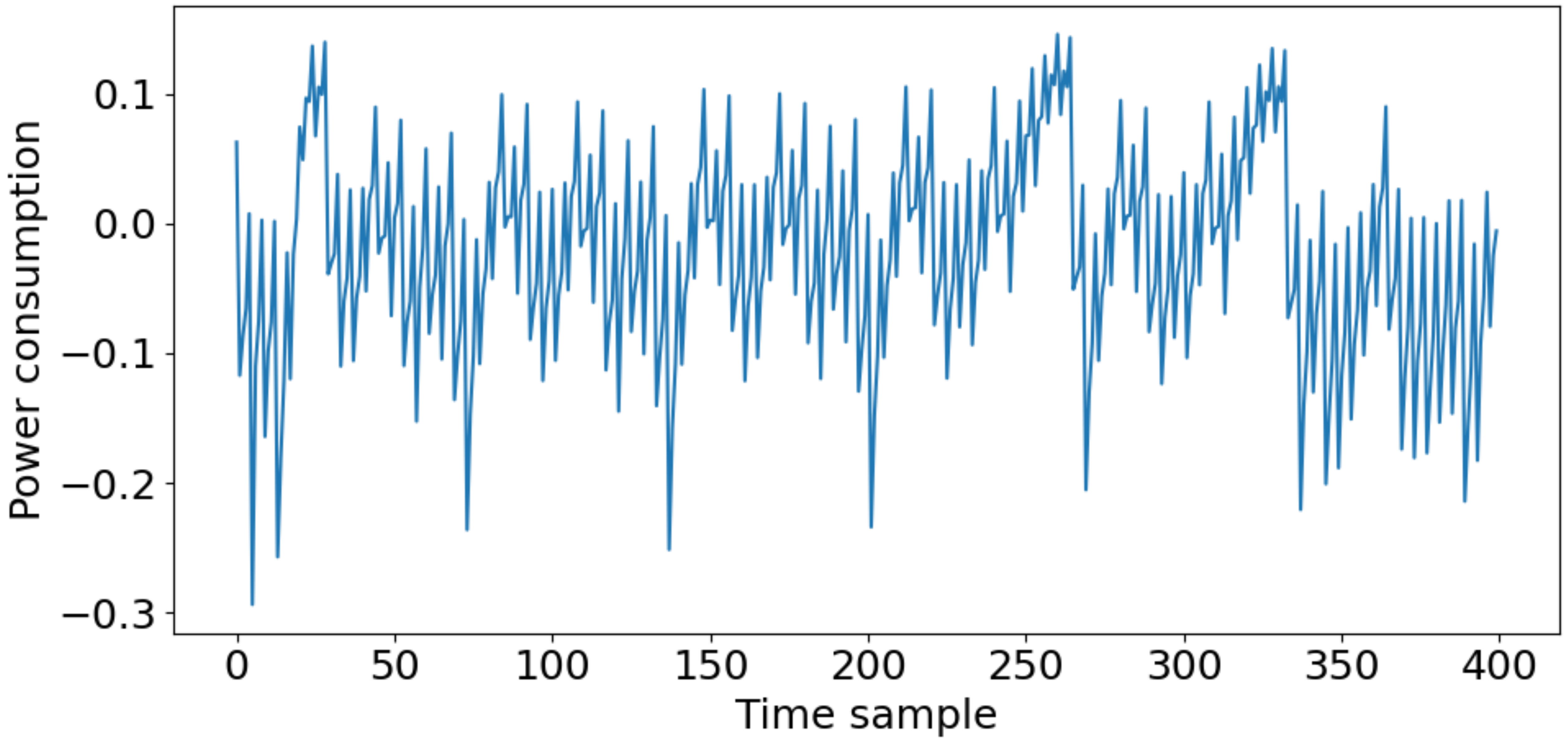
Leakage Models

Physical Assumptions & Issues



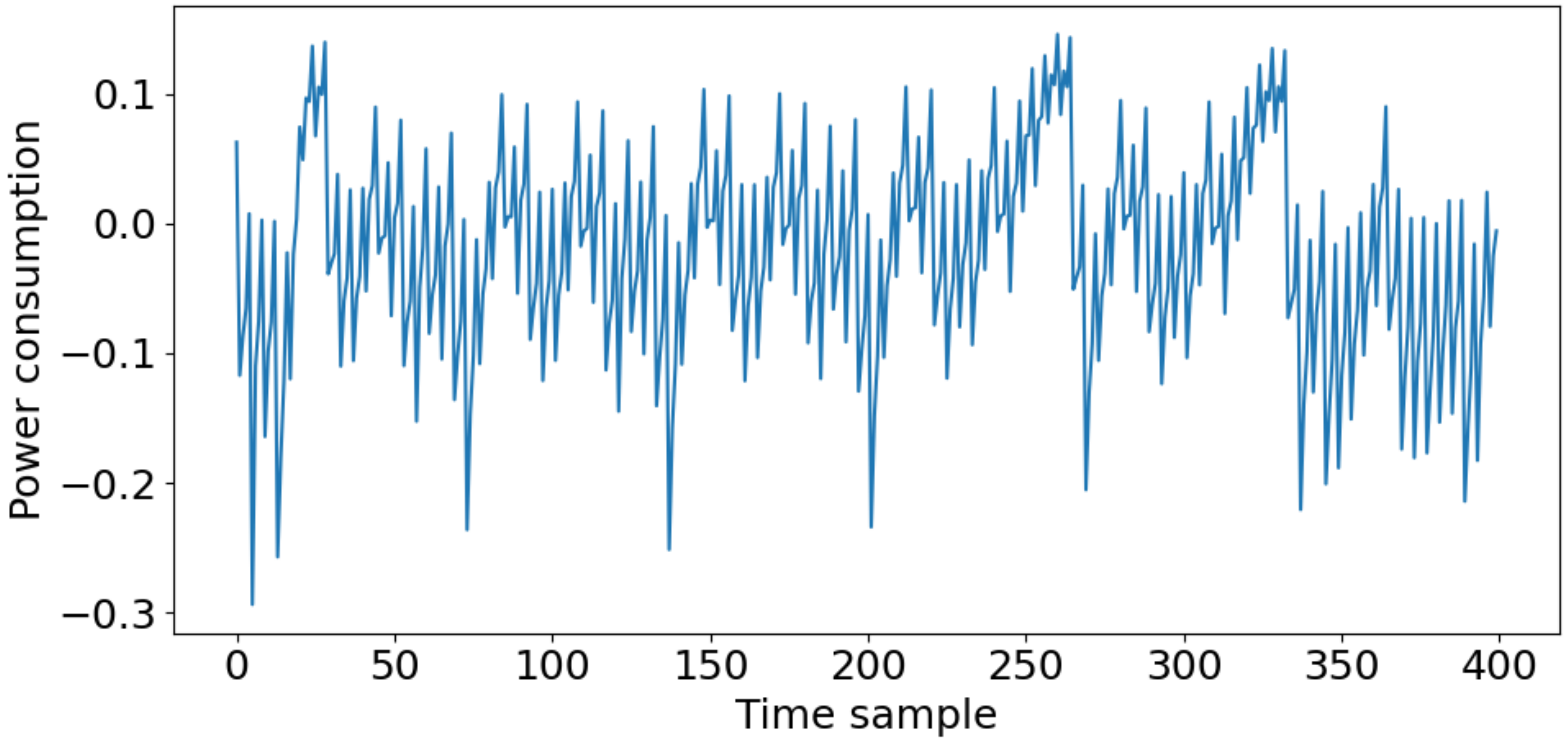
Leakage Models

Physical Assumptions & Issues



Leakage Models

Physical Assumptions & Issues

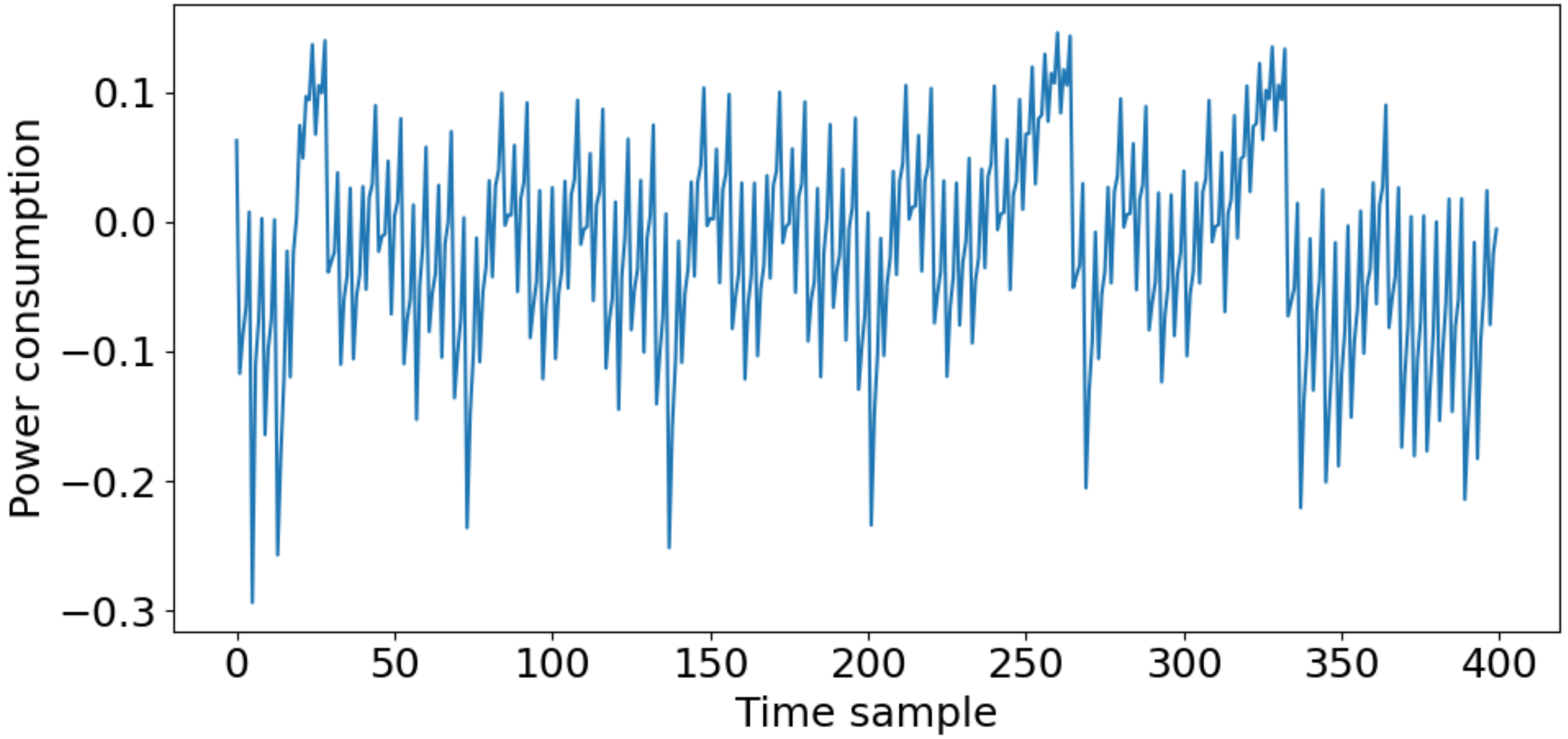


deterministic leakage

$$\overrightarrow{d}(\overrightarrow{x}) = (d_1, \ldots, d_{400})$$

Leakage Models

Physical Assumptions & Issues



deterministic leakage

$\overrightarrow{d}(\overrightarrow{x}) = (d_1, \ldots, d_{400})$

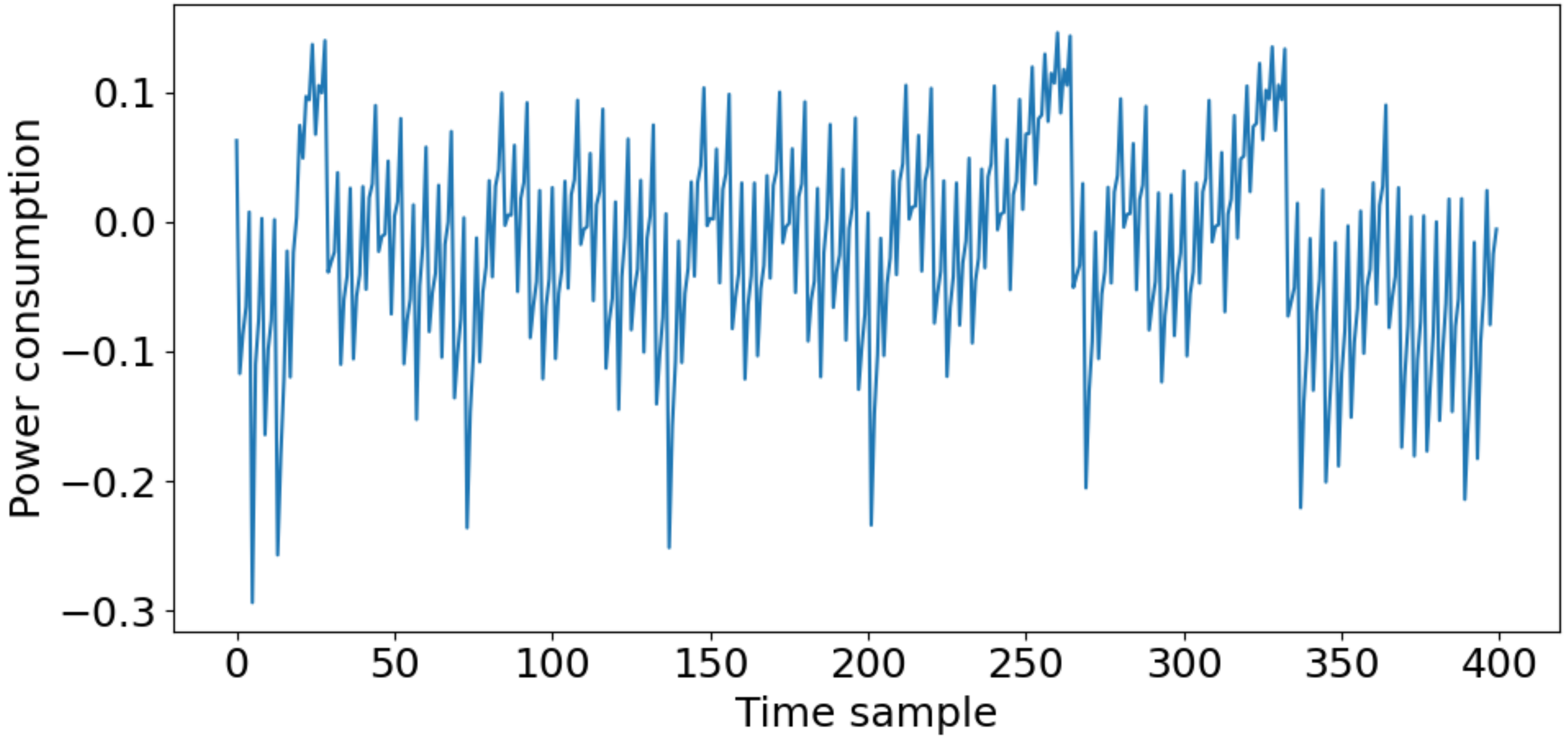
$+$

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

$.\vec{z} = (z_1, \ldots, z_{400})$

Leakage Models

Physical Assumptions & Issues



Multivariate noise breaks the noise independence assumption

deterministic leakage

$\overrightarrow{d}(\overrightarrow{x}) = (d_1, \ldots, d_{400})$

$+$

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

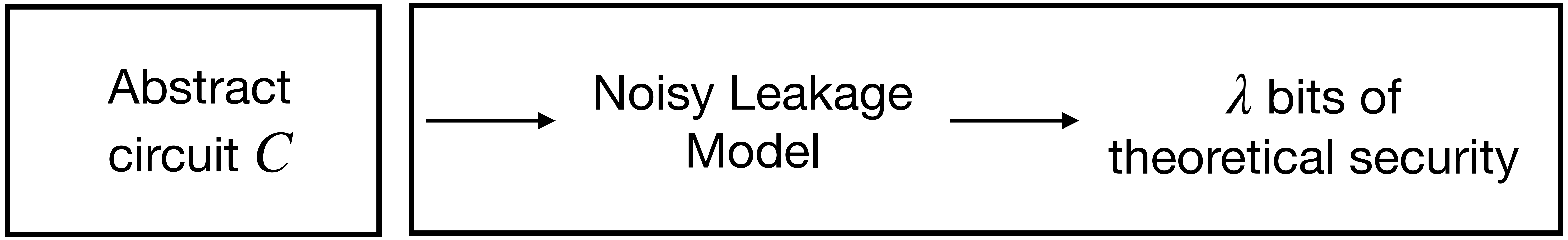
$.\vec{z} = (z_1, \ldots, z_{400})$

Overview

Overview

Abstract
circuit C

Overview



Abstract circuit C
Noisy Leakage Model
λ bits of theoretical security

Overview

Abstract circuit C $\longrightarrow$ ~~Noisy Leakage Model~~ ~~$\longrightarrow$~~ ~~λ bits of theoretical security~~ $\longrightarrow$ Implementation on a device

Loss of theoretical security level

Overview

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

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

Overview

Abstract circuit C → Noisy Leakage Model → λ 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

Overview

Abstract circuit C → Noisy Leakage Model → λ 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 → λ 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**

Overview

Abstract circuit C → Noisy Leakage Model → λ 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** → **Enforce / Relax noise independence**

Overview

Abstract circuit C → Noisy Leakage Model → λ 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** → **Enforce / Relax noise independence** → **Estimate the noisy leakage parameter δ tolerated by the device**

Overview

Abstract circuit C → Noisy Leakage Model → λ 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** → **Enforce / Relax noise independence** → **Estimate the noisy leakage parameter δ tolerated by the device** → **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 $\longrightarrow$ use data whitening

operation_$1(a_1, b_1)$

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_$1(a_1, b_1)$

$\boxed{\text{whitening()}}$

operation_$2(a_2, b_2)$

$\boxed{\text{whitening()}}$

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

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

Methodology

Step 2: Enforce / Relax data isolation

Leakage of an operation must only depend on its inputs $\longrightarrow$ 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

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()

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 ? → 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 $\overrightarrow{x}$ $\longrightarrow$ Leakage $\overrightarrow{y} = \overrightarrow{d}(\overrightarrow{x}) + \mathcal{N}(\overrightarrow{0}, \Sigma)$

Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

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

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

1. Infer $d_i(\cdot)$ for each time sample i
2. Compute the covariance matrix Σ

Methodology

Step 3: Characterize the Leakage Distribution

Extensively studied in the literature

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

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

2. Compute the covariance matrix Σ

Linear Regression
Machine Learning
$\ldots$

Methodology

Step 4: Enforce / Relax Noise Independence

Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice $\longrightarrow$ 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 $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

Methodology

Step 4: Enforce / Relax Noise Independence

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

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

<u>data isolation assumption</u>

each S_i is only the leakage of operation$_i$

start

Leakage trace $\vec{Y}$

operation$_1$

operation$_2$

operation$_3$

end

Methodology

Step 4: Enforce / Relax Noise Independence

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

$$\overrightarrow{Y} = \overrightarrow{S_1} + \overrightarrow{S_2} + \overrightarrow{S_3} + \overrightarrow{N}$$

<u>data isolation assumption</u>

each S_i is only the leakage of operation_i

start

Leakage trace $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

Split $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$

instead of having time-separated noises

Methodology

Step 4: Enforce / Relax Noise Independence

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

<u>data isolation assumption</u>

each S_i is only the leakage of operation_i

start

Leakage trace $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

$$\overrightarrow{Y} = \overrightarrow{S_1} + \overrightarrow{S_2} + \overrightarrow{S_3} + \overrightarrow{N}$$

$\downarrow$

Split $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$
instead of having time-separated noises

$\downarrow$

Then the leakage is split into
$$\{\overrightarrow{Y_i} = \overrightarrow{S_i} + \overrightarrow{N_i}\}_{i=1,2,3}$$

Methodology

Step 4: Enforce / Relax Noise Independence

Difficult to ensure in practice ⟶ We propose to relax it

$$\overrightarrow{Y} = \overrightarrow{S_1} + \overrightarrow{S_2} + \overrightarrow{S_3} + \overrightarrow{N}$$

start

Leakage trace $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

Split $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$
instead of having time-separated noises

Trivial: $\overrightarrow{N_i} = \dfrac{1}{3}\overrightarrow{N}$

Methodology

Step 4: Enforce / Relax Noise Independence

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

start

Leakage trace $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

$$\overrightarrow{Y} = \overrightarrow{S_1} + \overrightarrow{S_2} + \overrightarrow{S_3} + \overrightarrow{N}$$

$$\downarrow$$

Split $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$
instead of having time-separated noises

Trivial: $\overrightarrow{N_i} = \frac{1}{3}\overrightarrow{N}$

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

Methodology

Step 4: Enforce / Relax Noise Independence

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

start

Leakage trace $\overrightarrow{Y}$

operation_1

operation_2

operation_3

end

$$\overrightarrow{Y} = \overrightarrow{S_1} + \overrightarrow{S_2} + \overrightarrow{S_3} + \overrightarrow{N}$$

$$\downarrow$$

Split $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$
instead of having time-separated noises

Trivial: $\overrightarrow{N_i} = \frac{1}{3}\overrightarrow{N}$

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 $\overrightarrow{N} = \overrightarrow{N_1} + \overrightarrow{N_2} + \overrightarrow{N_3}$, such as to minimize the information leakage of the different operations ?

Methodology

Step 5: Estimate the noisy leakage parameter δ

Methodology

Step 5: Estimate the noisy leakage parameter δ

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

Methodology

Step 5: Estimate the noisy leakage parameter δ

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

Efficient way to compute δ on a device

Methodology

Step 5: Estimate the noisy leakage parameter δ

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

Efficient way to compute δ on a device

$\downarrow$

Infer tolerated leakage probability p by the device

Methodology

Step 5: Estimate the noisy leakage parameter δ

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

Efficient way to compute δ on a device

$\downarrow$

Infer tolerated leakage probability p by the device

$\downarrow$

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

Methodology

Step 5: Estimate the noisy leakage parameter δ

$(p, \varepsilon)-$random probing security $\Longrightarrow$ $\delta-$noisy leakage security

Efficient way to compute δ 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

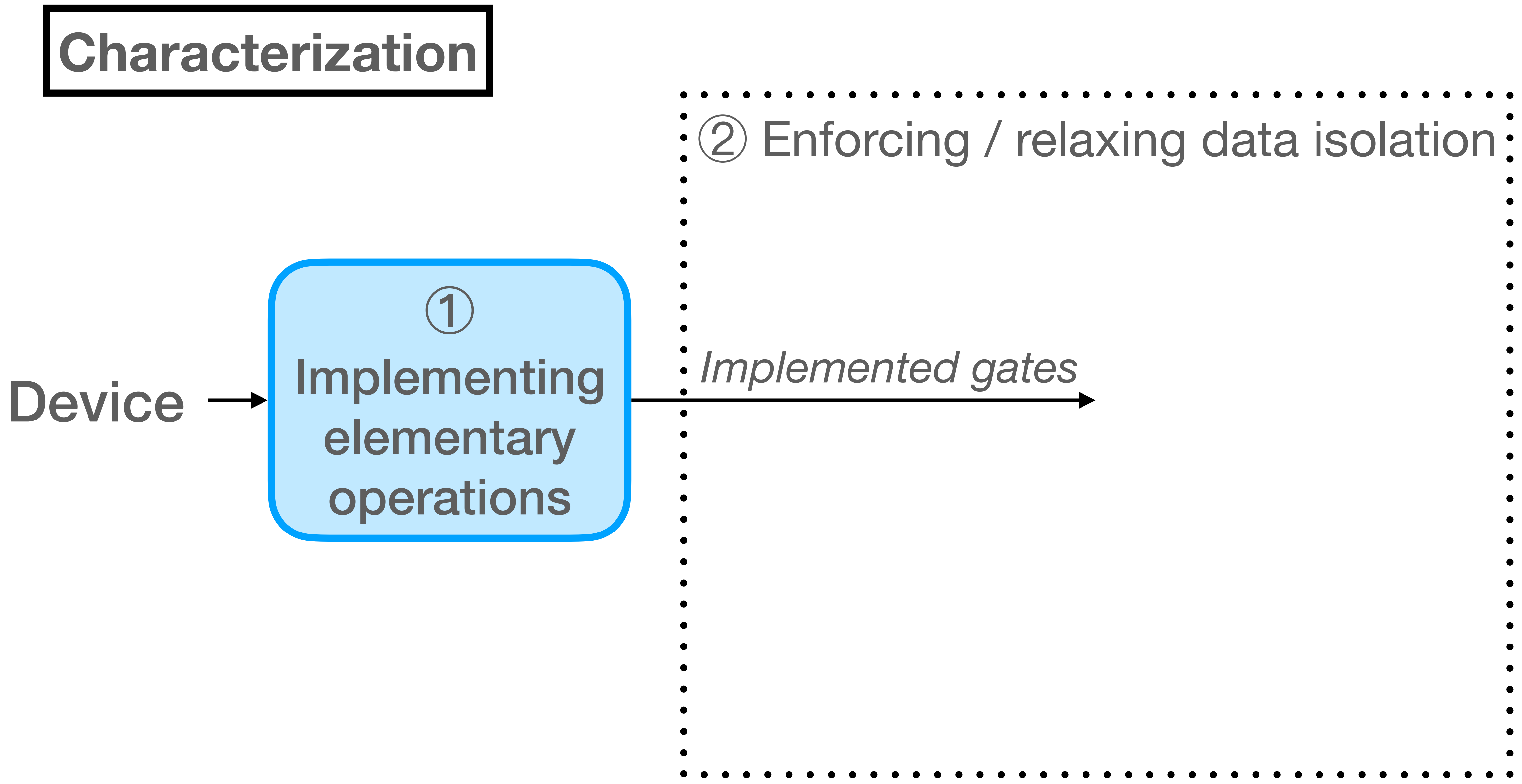
Wrap-up

Characterization

Device → ① Implementing elementary operations

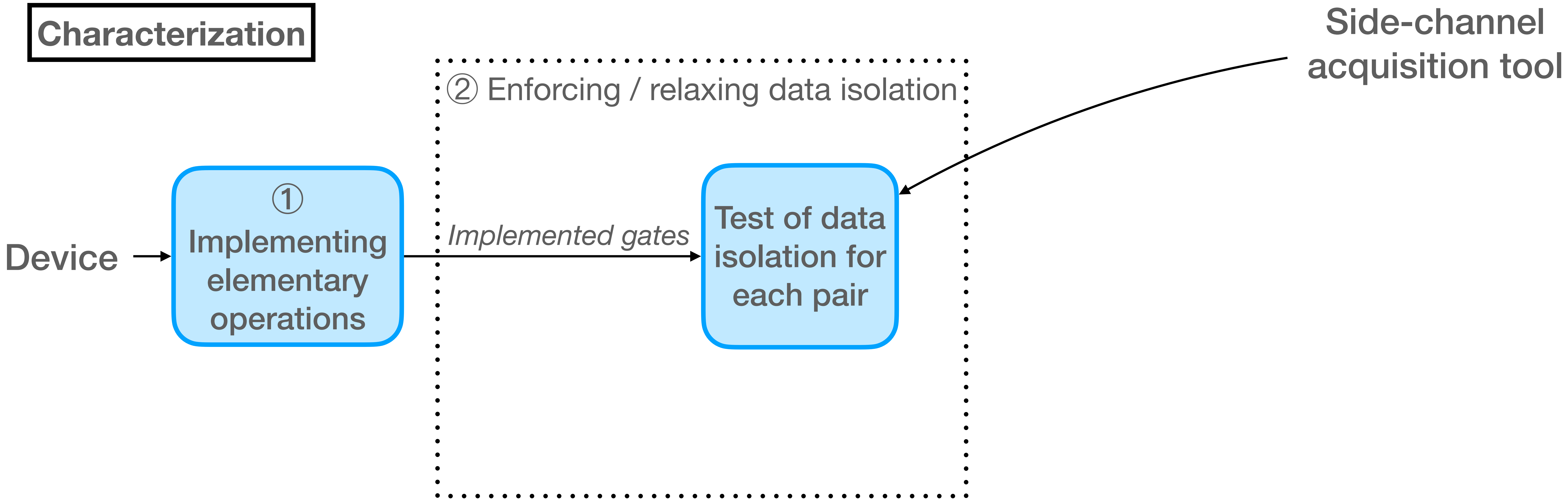
Methodology

Wrap-up



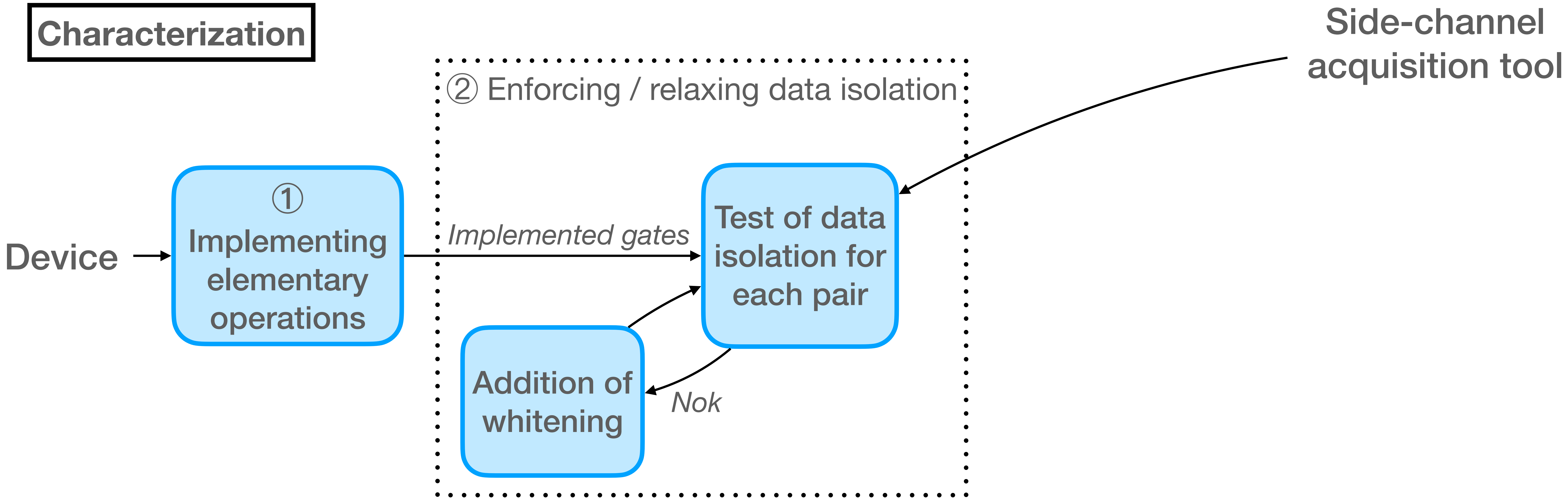
Methodology

Wrap-up



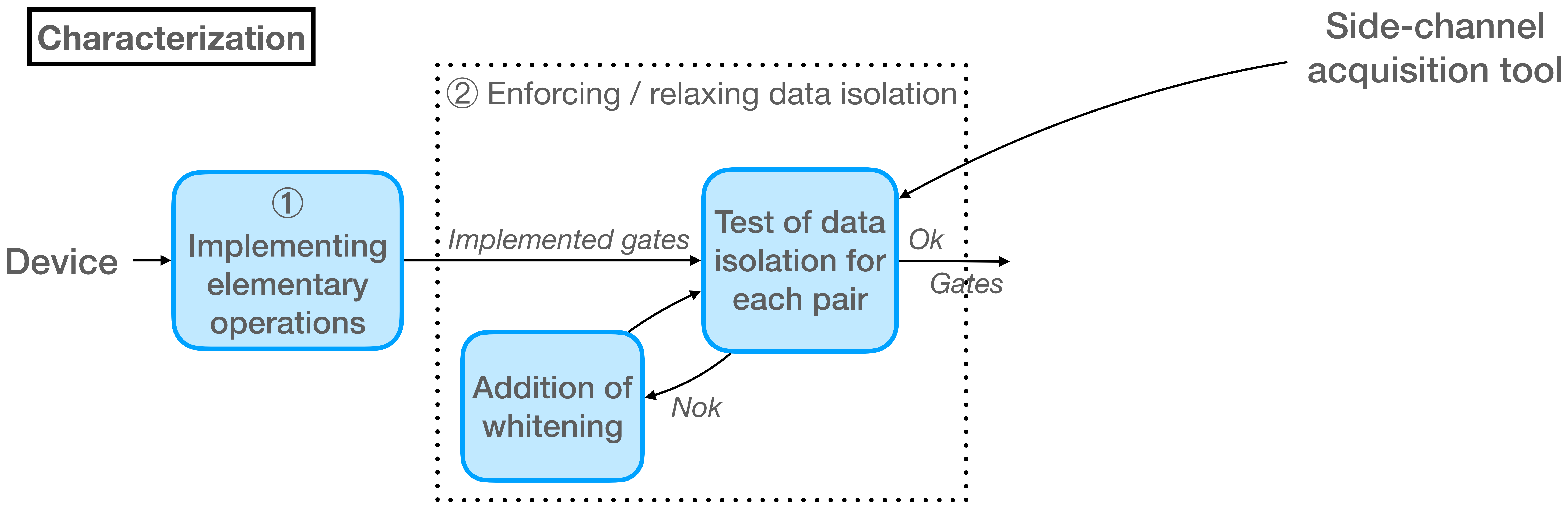
Methodology

Wrap-up



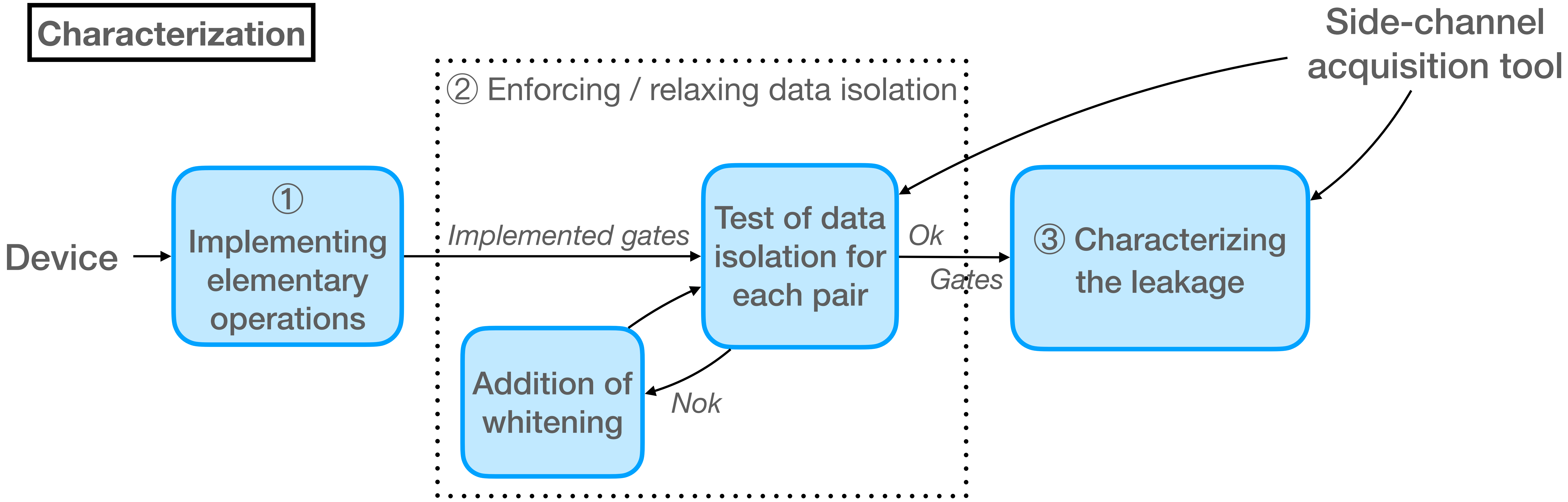
Methodology

Wrap-up



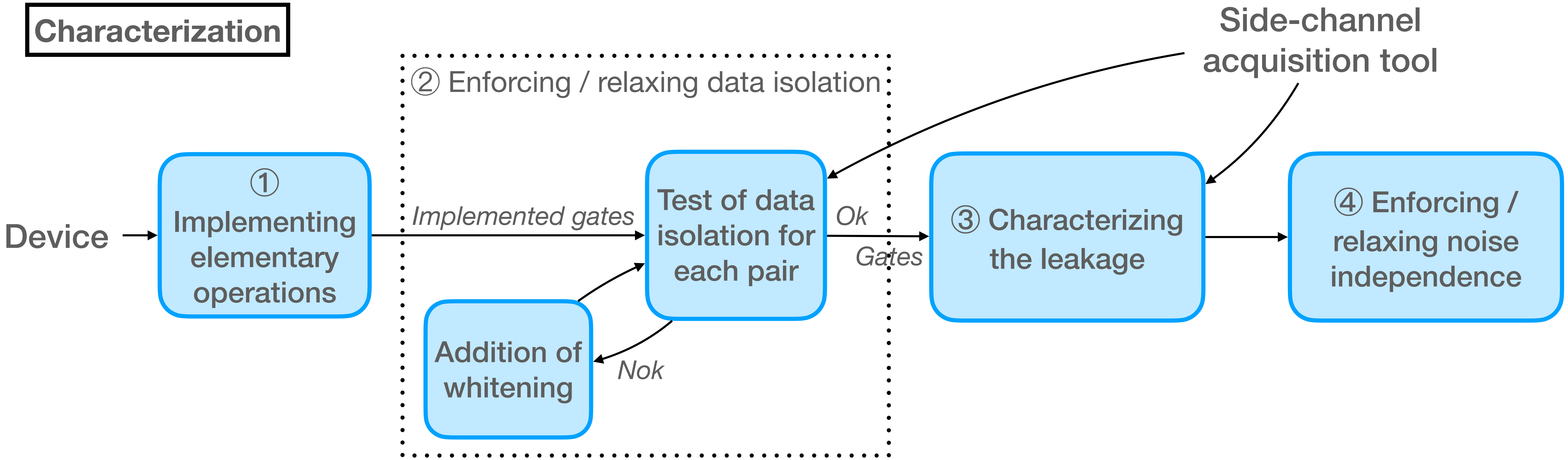
Methodology

Wrap-up



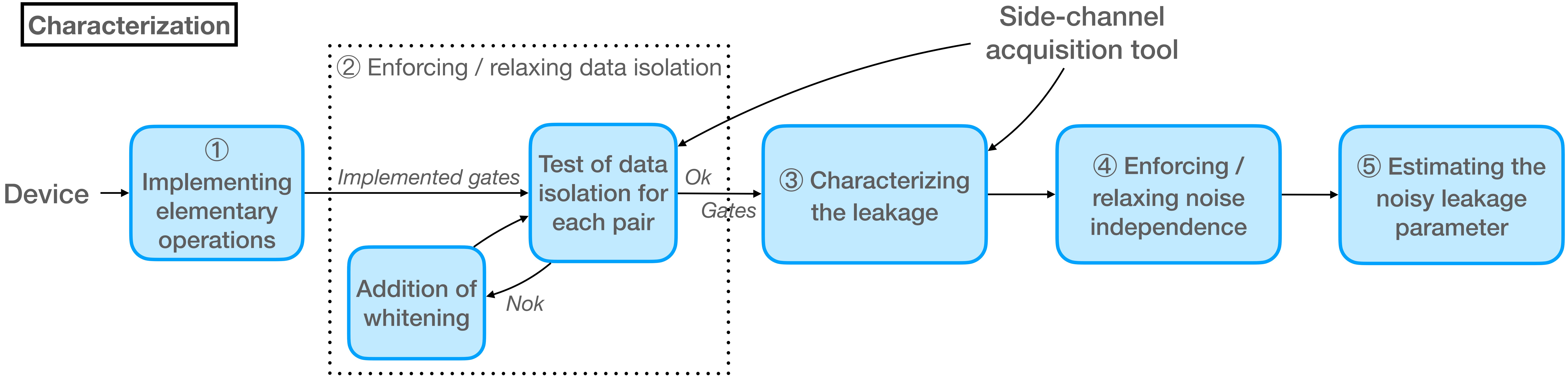
Methodology

Wrap-up



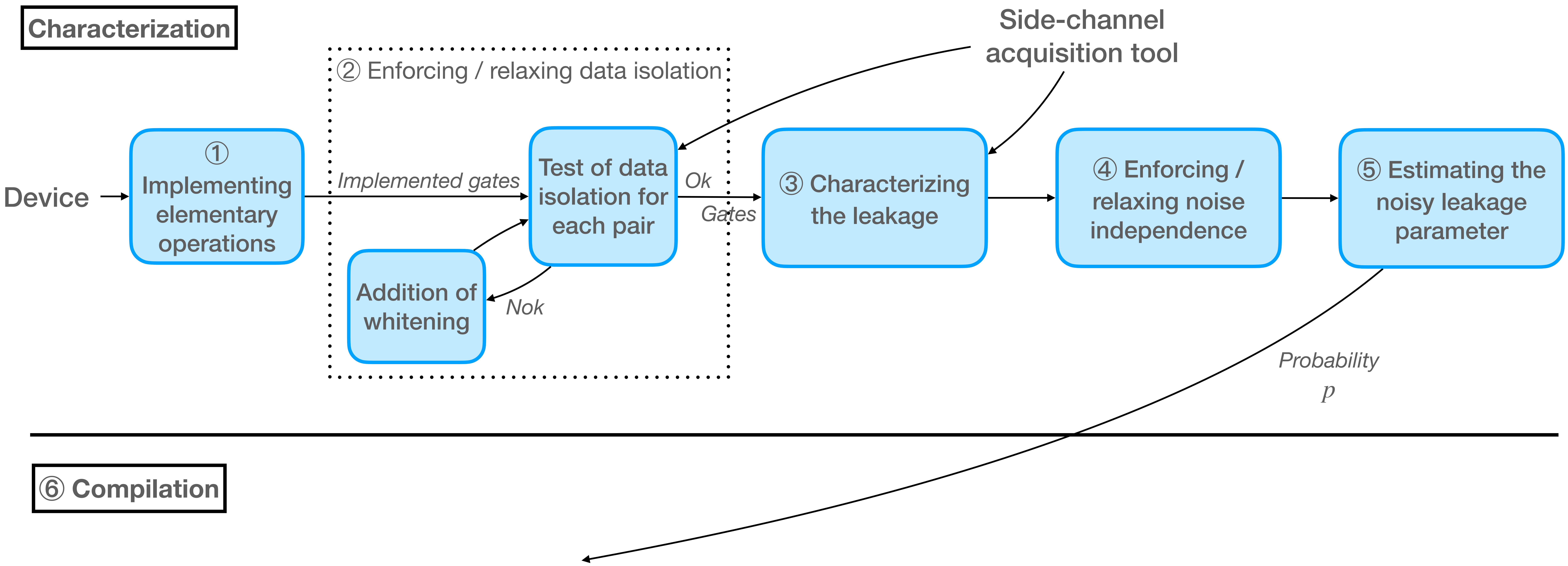
Methodology

Wrap-up



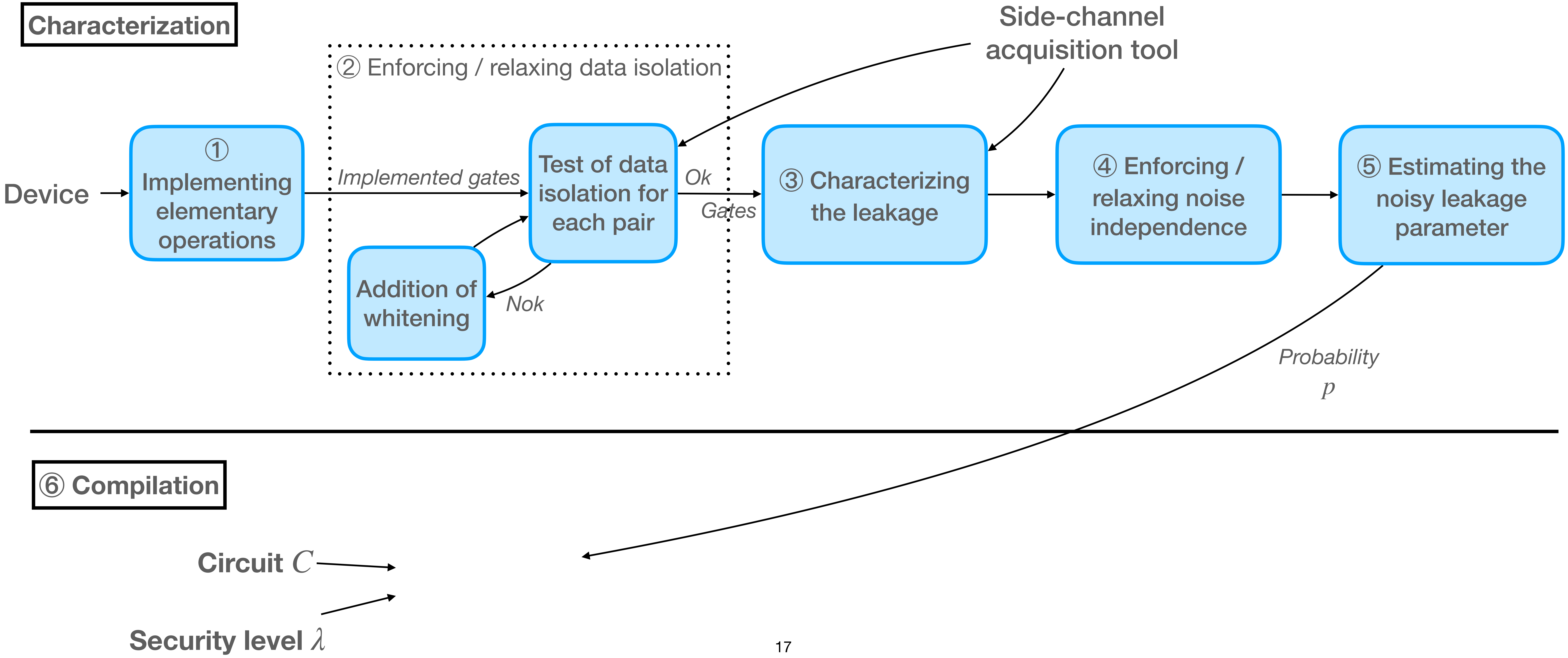
Methodology

Wrap-up



Methodology

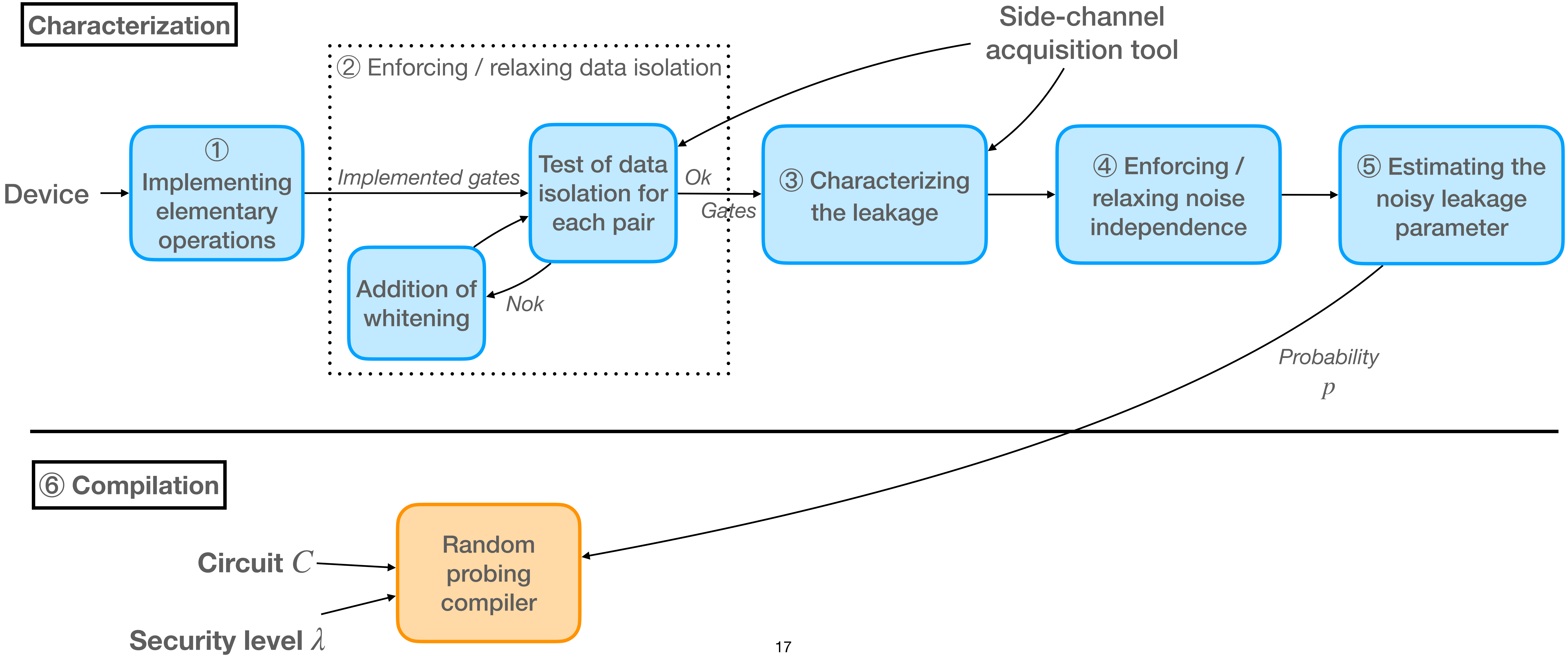
Wrap-up



Methodology

Wrap-up

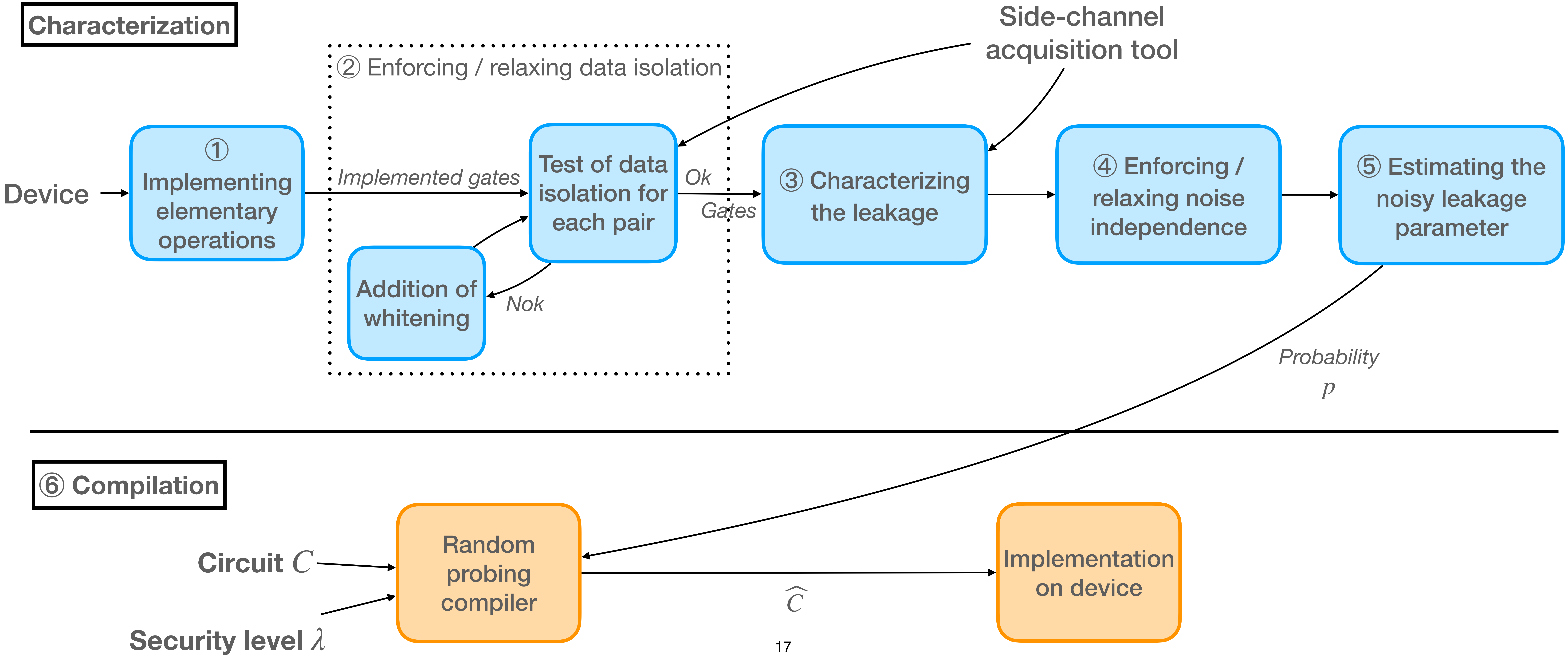
Random probing compiler: replaces each gate by a n-share $(p, \varepsilon)-$random probing secure gadget



Methodology

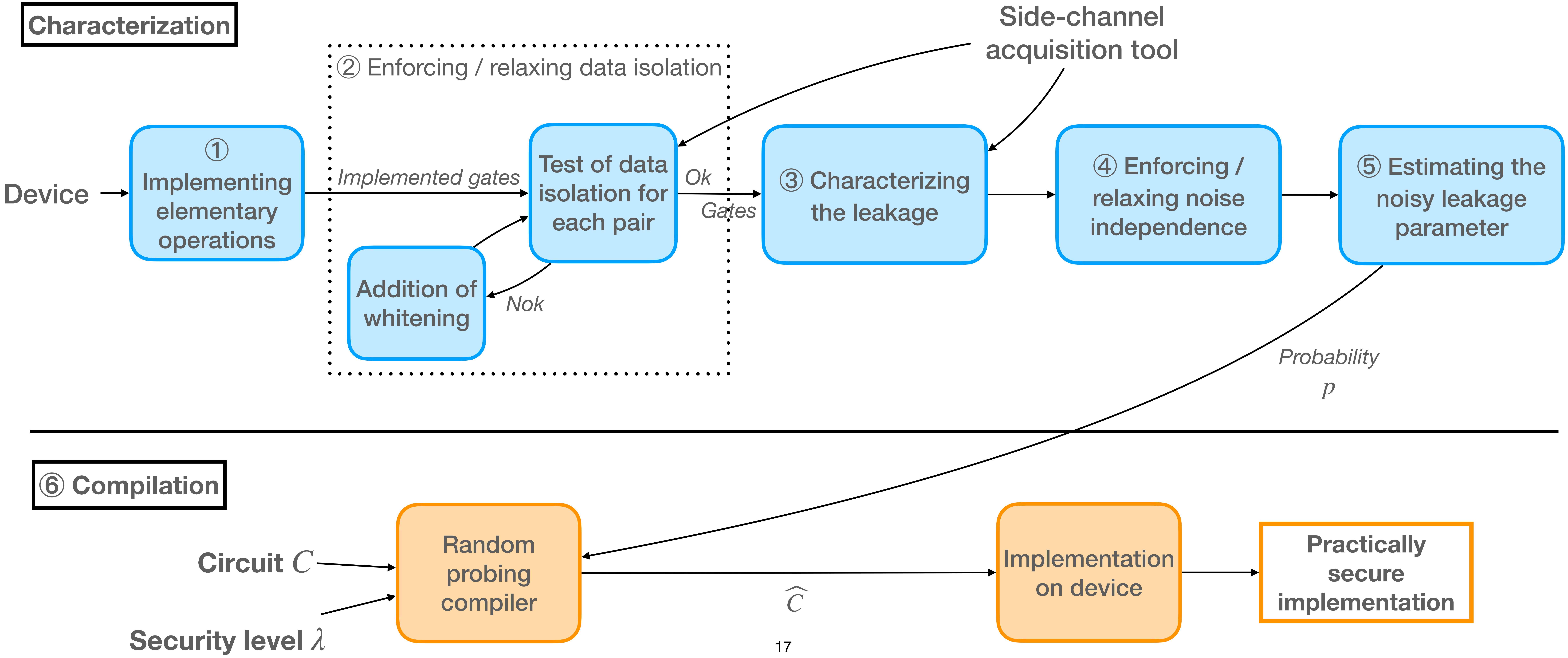
Wrap-up

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



Methodology
Wrap-up

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



Limitations / Questions

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