CRYPTOEXPERTS

WE INNOVATE TO SECURE YOUR BUSINESS

Securing Cryptography Against Side-Channel Attacks

Practical Tools and Proven Countermeasures

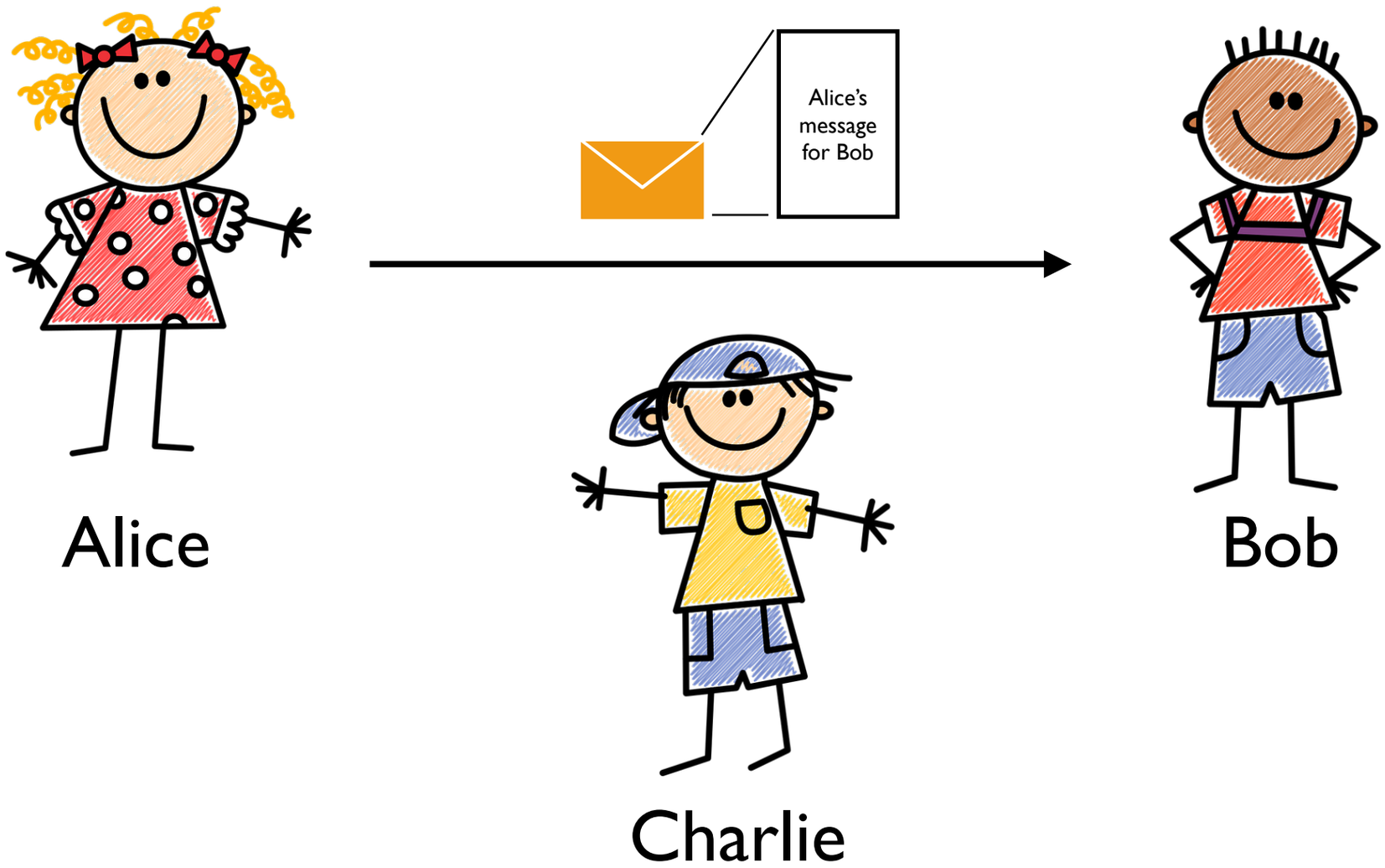
Sonia Belaïd

Overview

- What are side-channel attacks?
  - Definition, examples
- How to thwart side-channel attacks?
  - Masking countermeasure
- How to make sure that you did it?
  - Leakage models, proofs, automatic tools

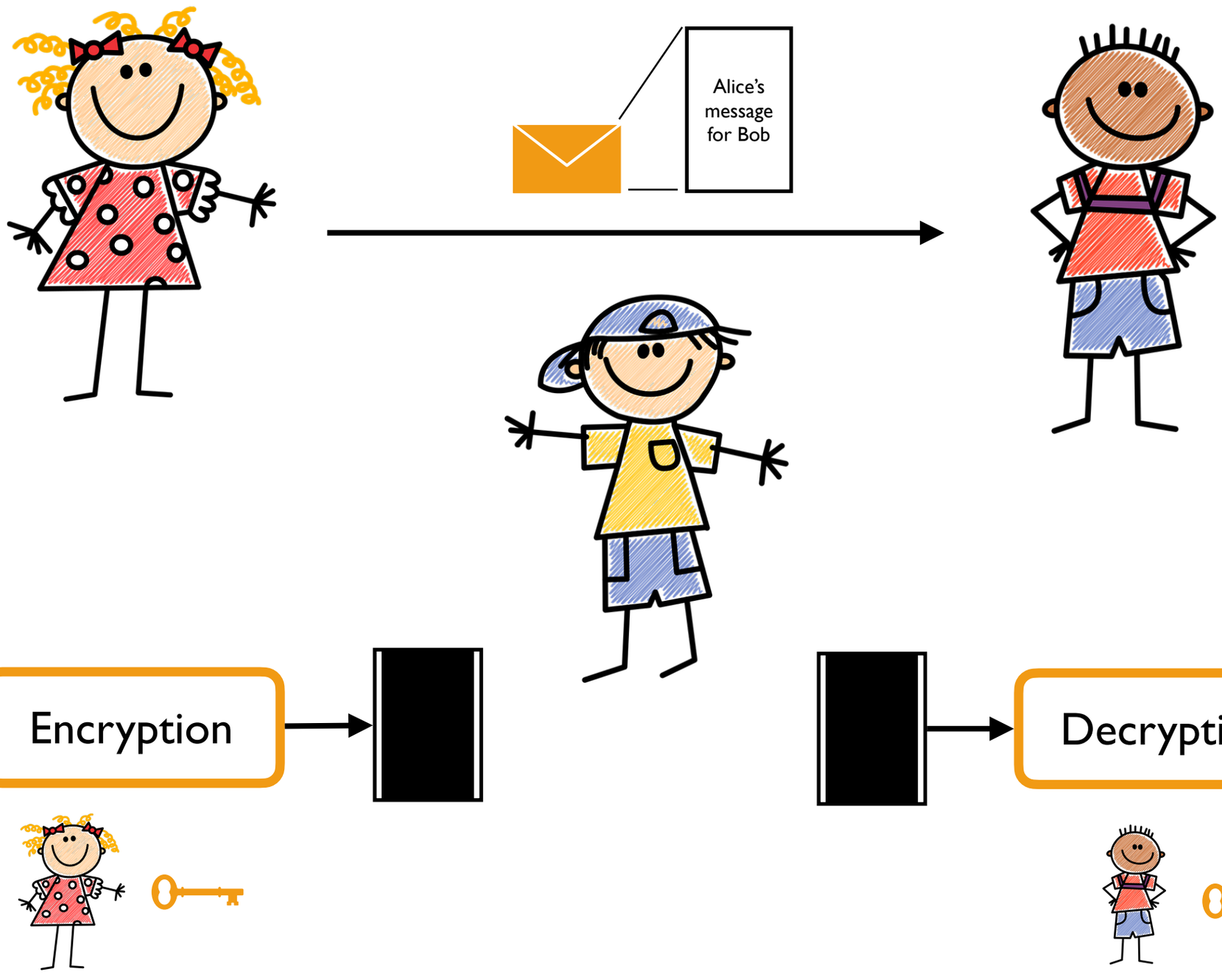
Side-Channel Attacks

Side-Channel Attacks

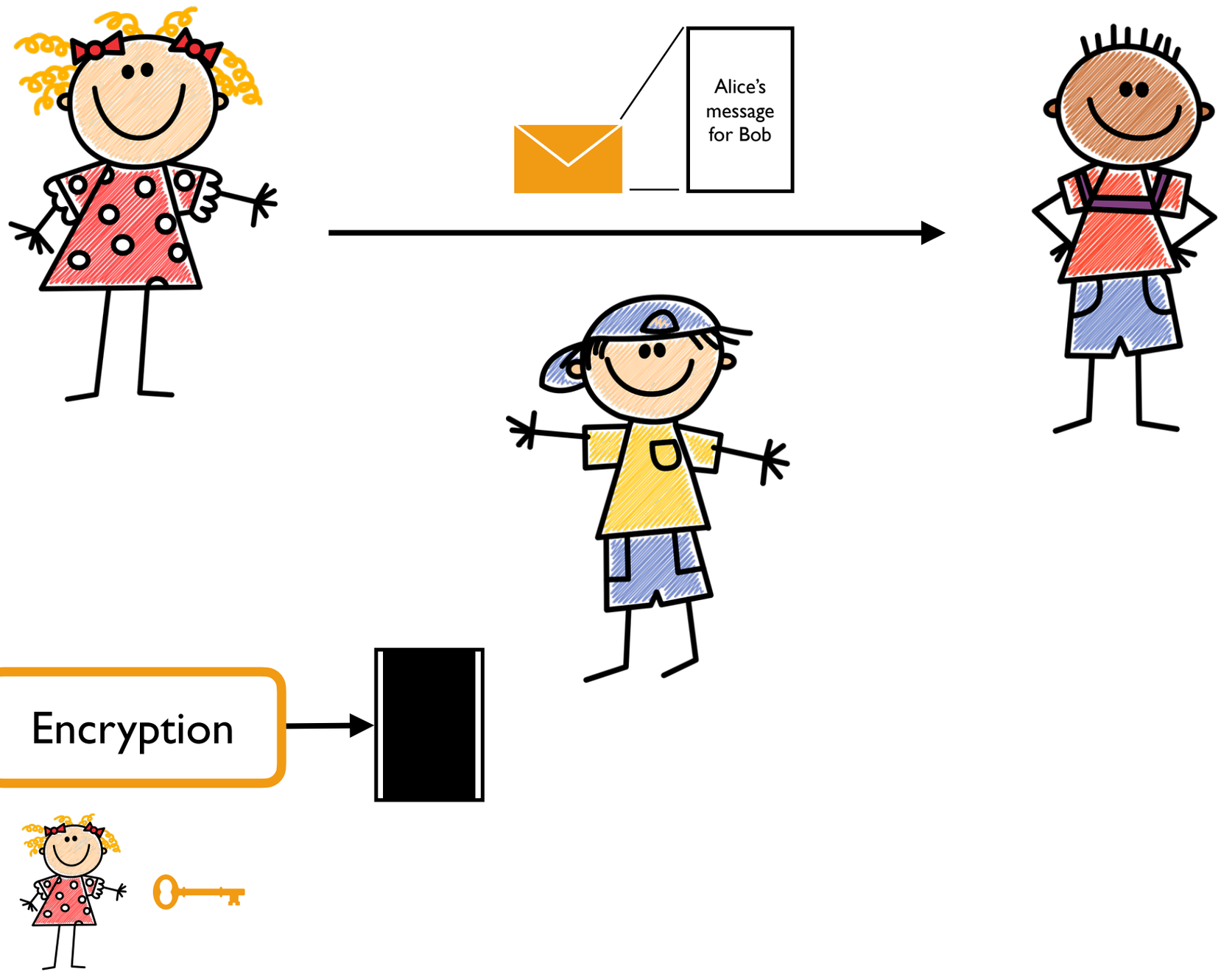


Definition: Cryptography is the science and art of protecting information despite external attacks.

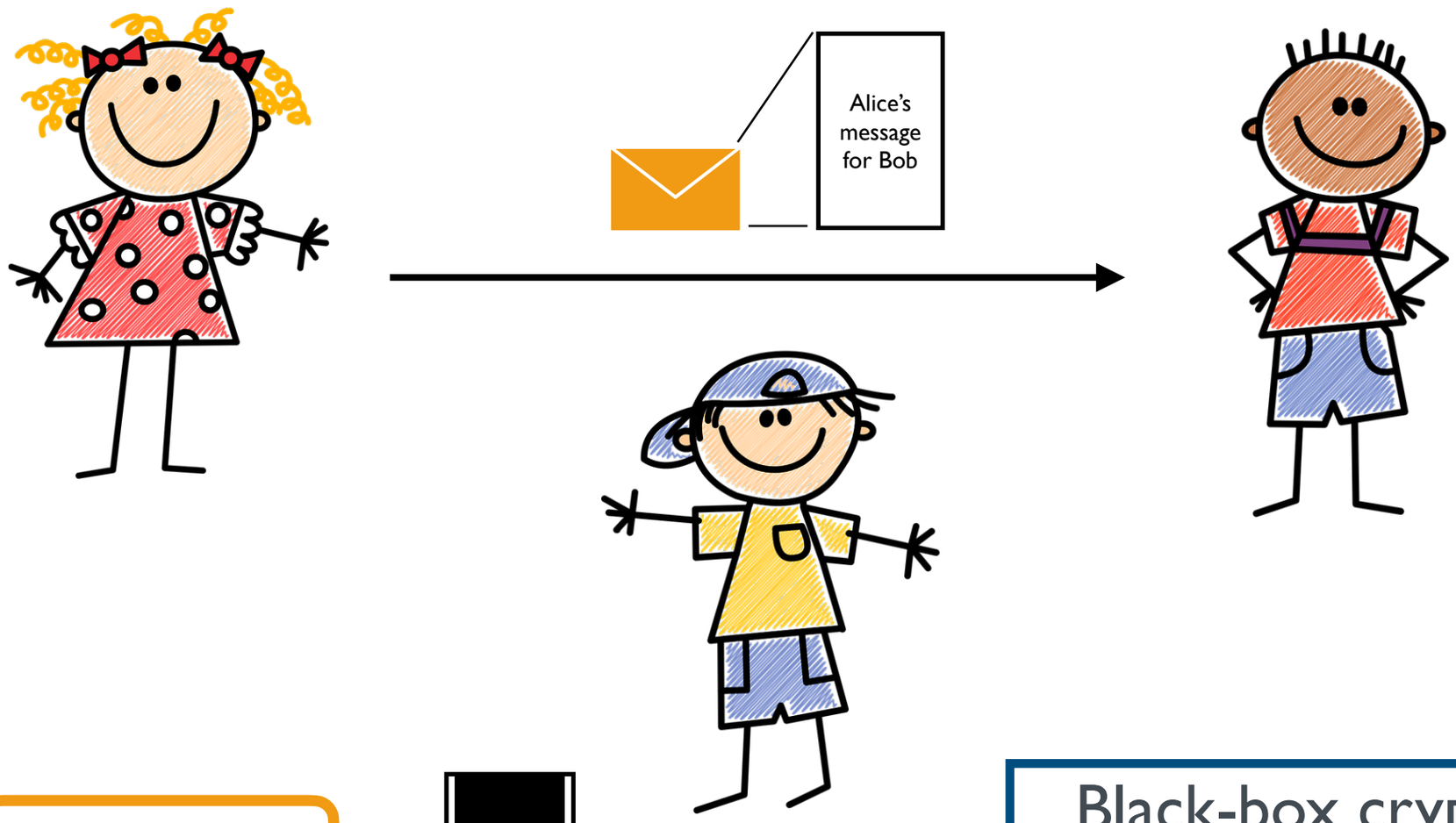
Side-Channel Attacks

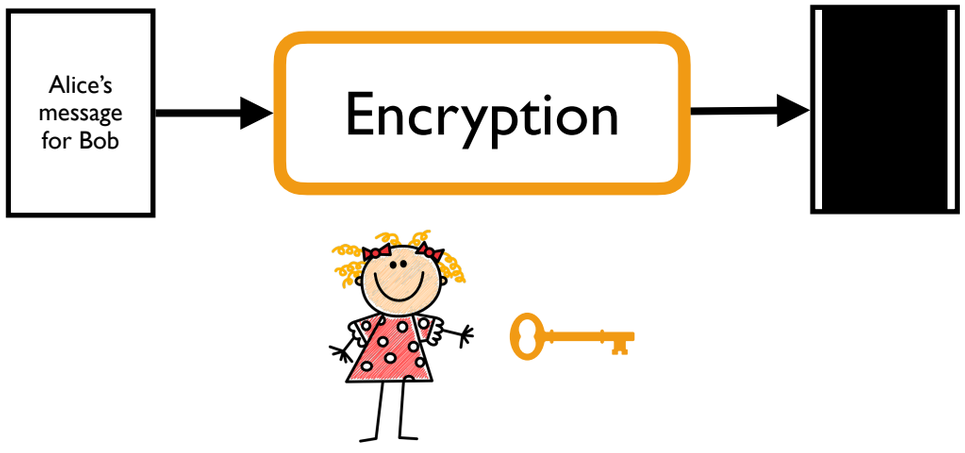


Side-Channel Attacks



Side-Channel Attacks

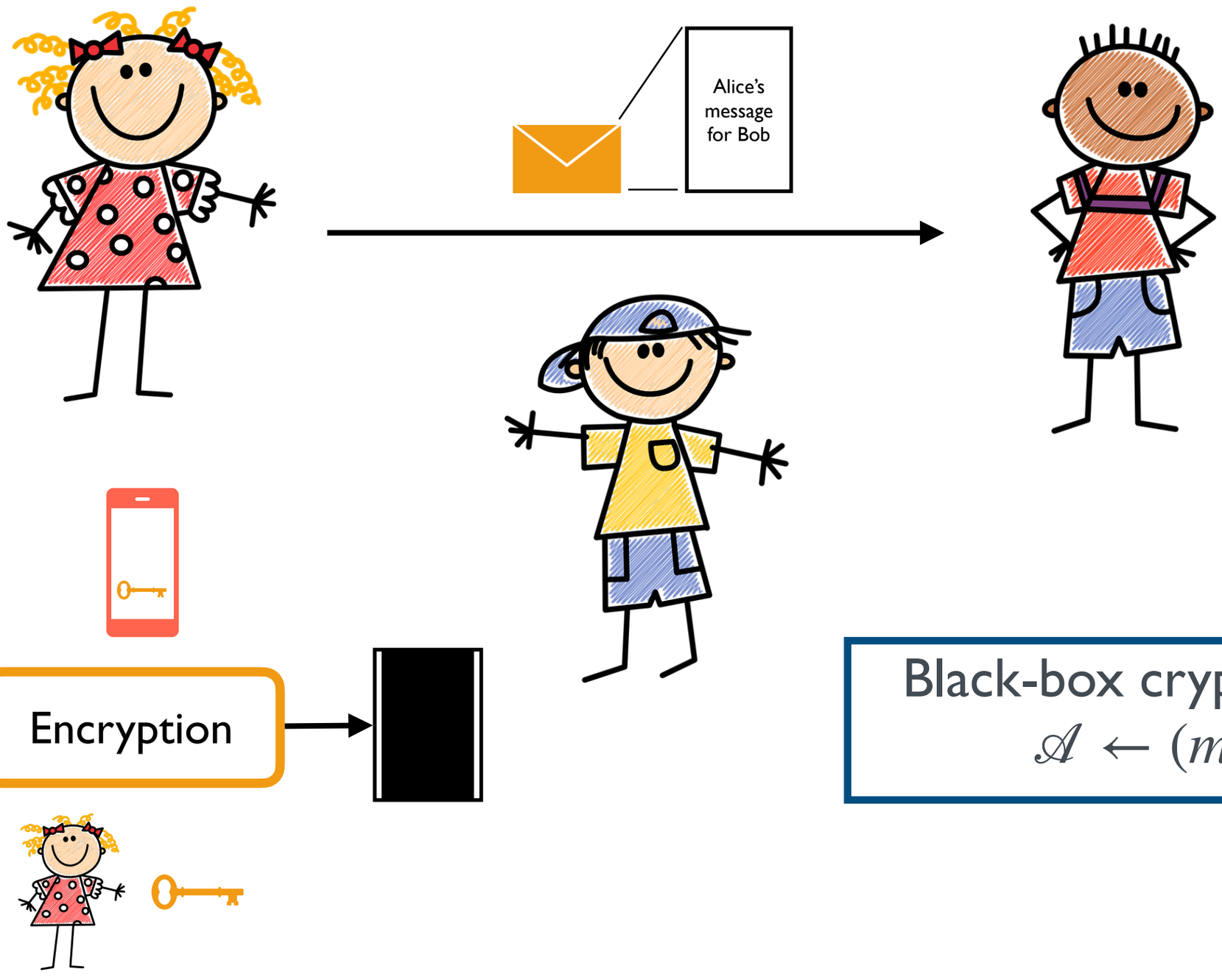




Black-box cryptanalysis:

$$\mathcal{A} \leftarrow (m, c)$$

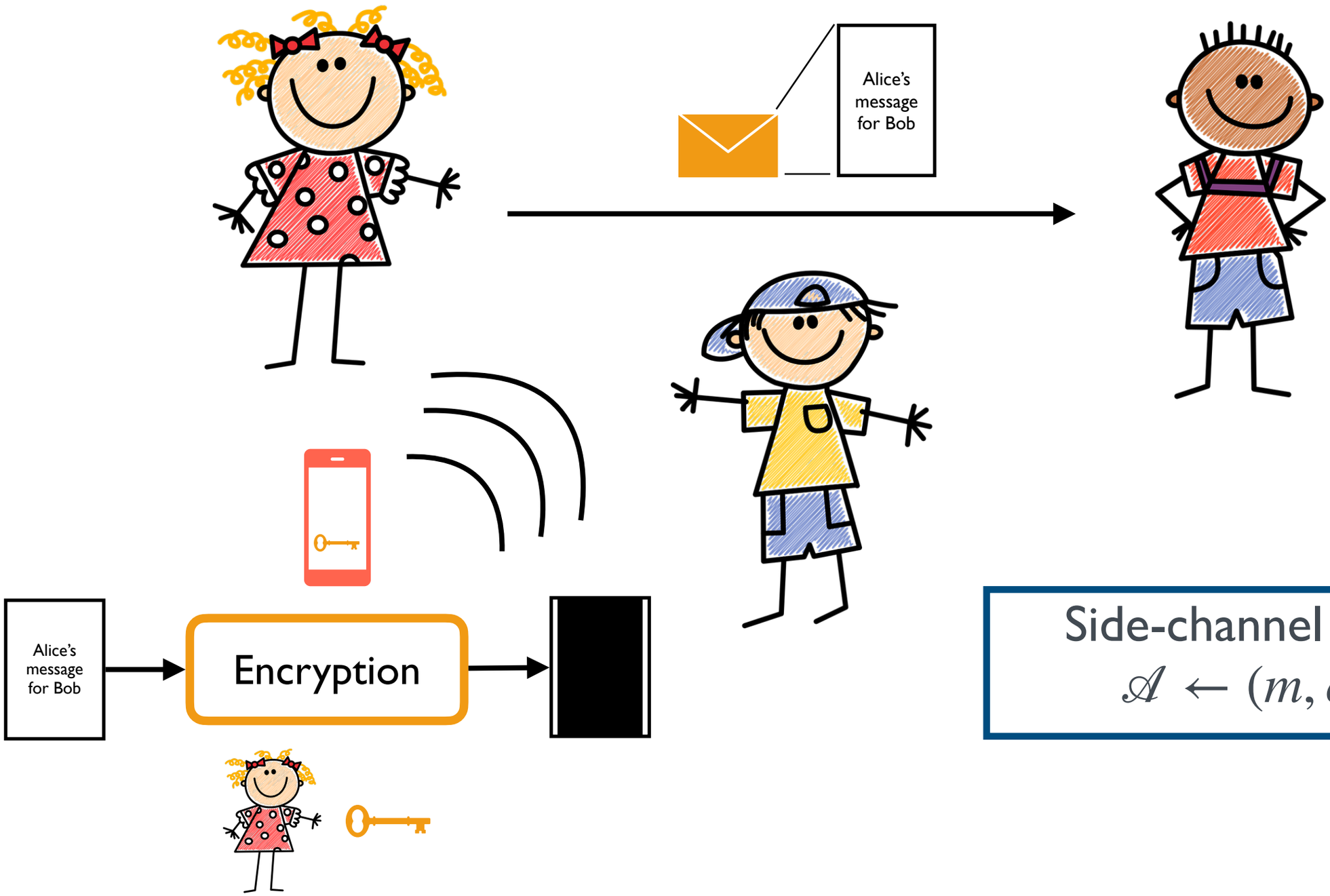
Side-Channel Attacks



Black-box cryptanalysis:

$$\mathcal{A} \leftarrow (m, c)$$

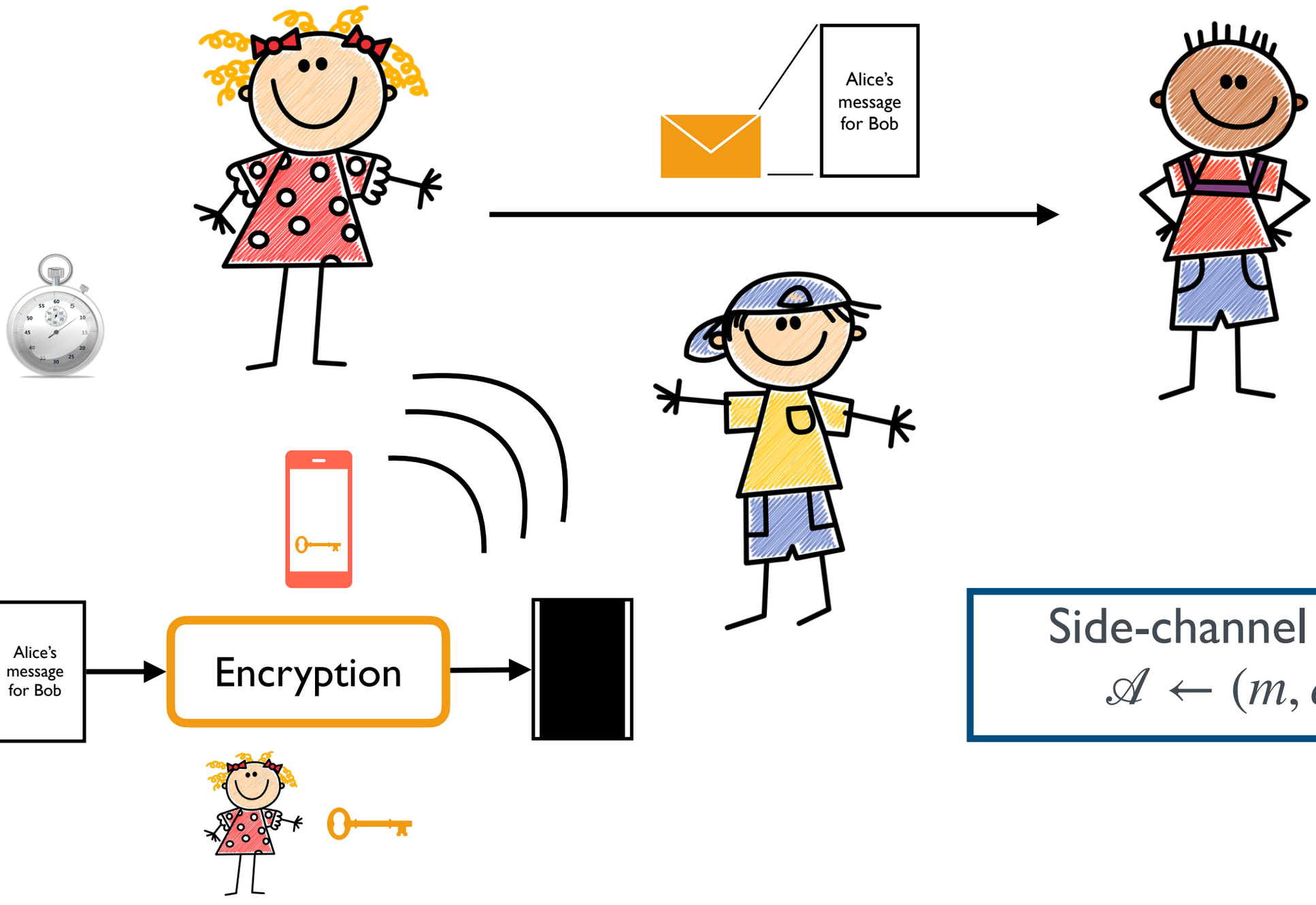
Side-Channel Attacks



Side-channel analysis:

$\mathcal{A} \leftarrow (m, c, \mathcal{L})$

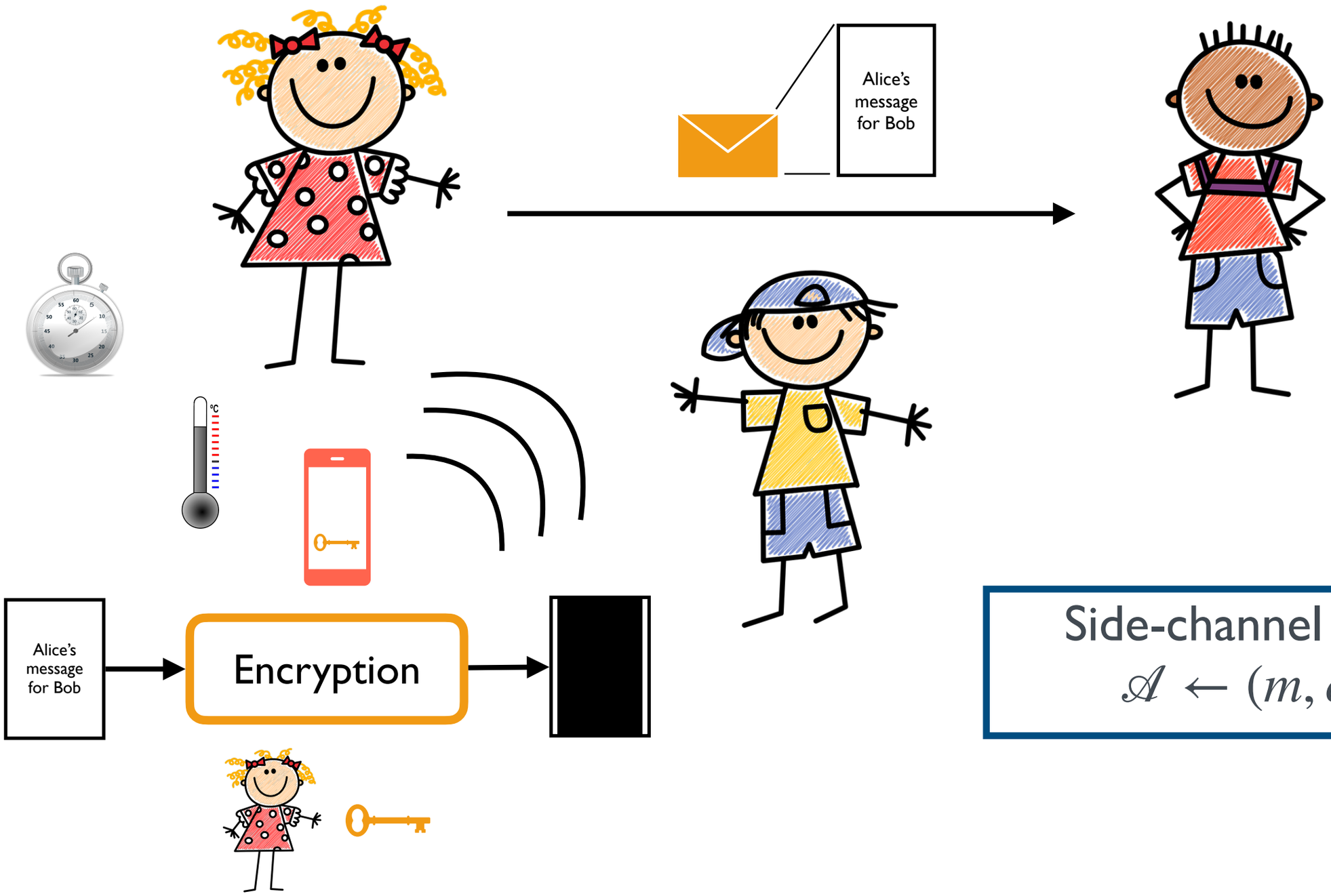
Side-Channel Attacks



Side-channel analysis:

$$\mathscr{A} \leftarrow (m, c, \mathscr{L})$$

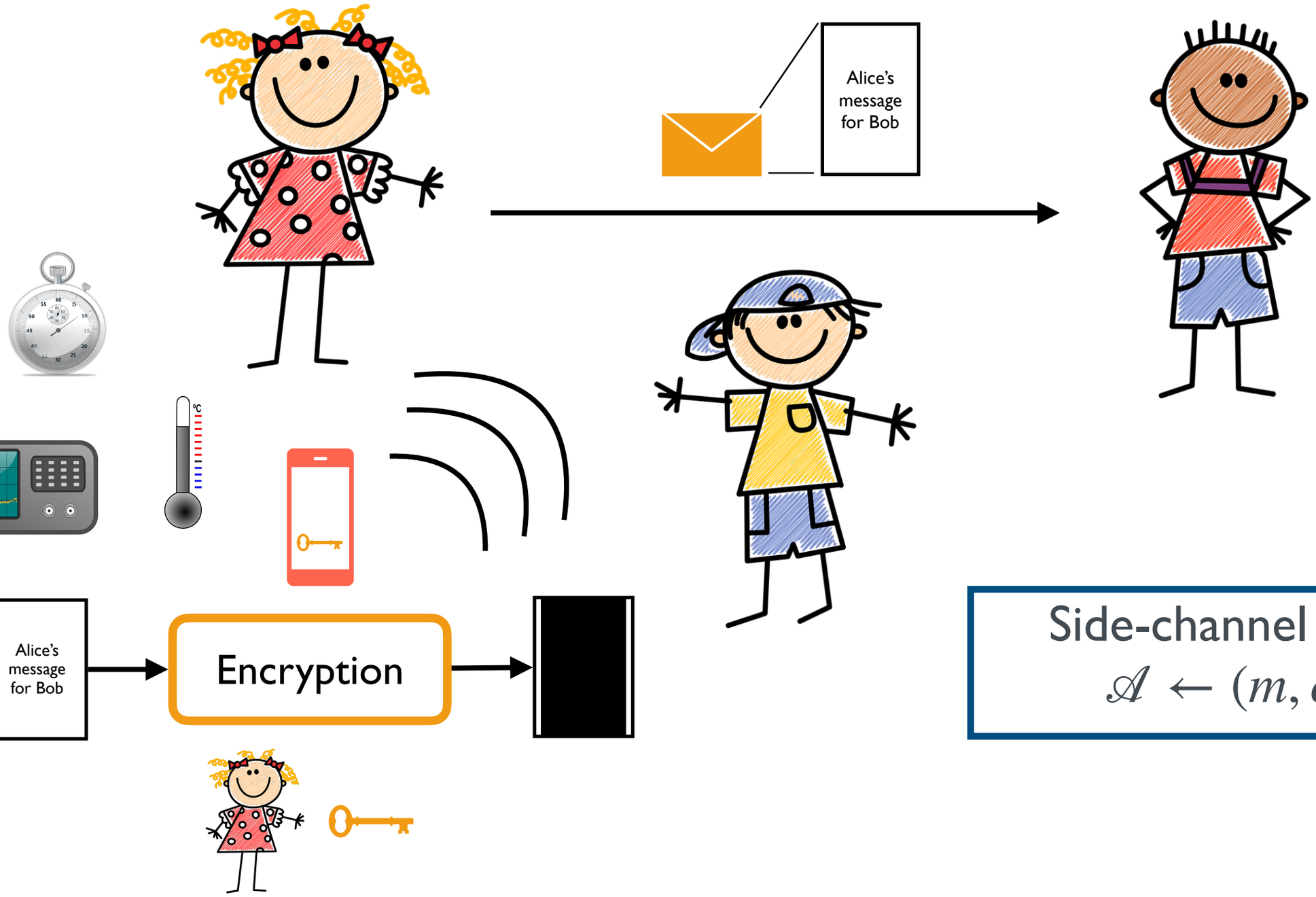
Side-Channel Attacks



Side-channel analysis:

$$\mathscr{A} \leftarrow (m, c, \mathscr{L})$$

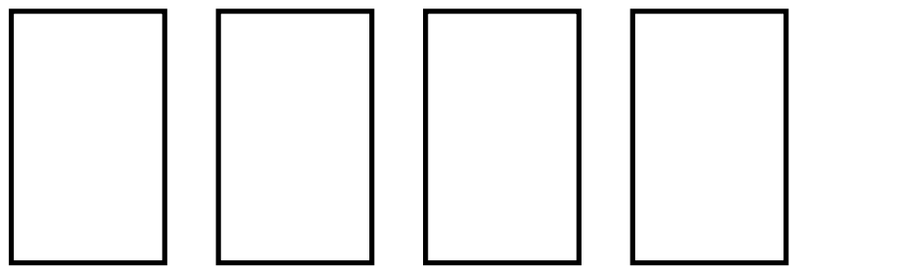
Side-Channel Attacks



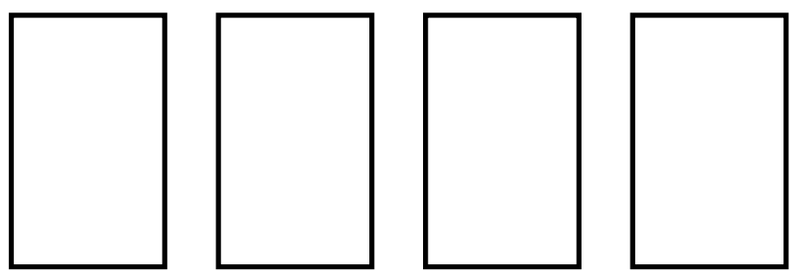
Side-channel analysis:

$$\mathscr{A} \leftarrow (m, c, \mathscr{L})$$

Side-Channel Attacks



Side-Channel Attacks



10000 possibilities

Side-Channel Attacks

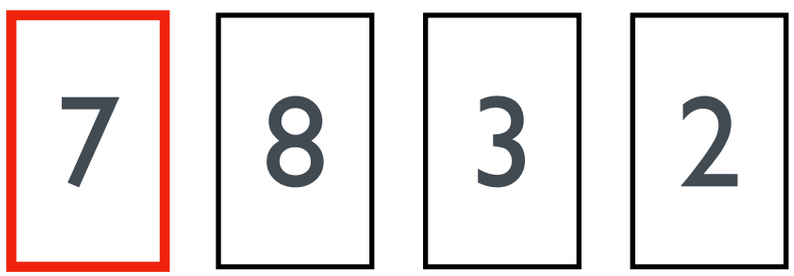
Correct PIN: 9401

7	8	3	2

10000 possibilities

Side-Channel Attacks

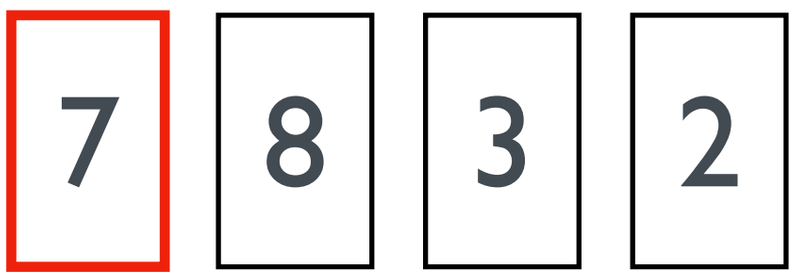
Correct PIN: 9401



10000 possibilities

Side-Channel Attacks

Correct PIN: 9401



Execution time: α ms

10000 possibilities

Side-Channel Attacks

Correct PIN: 9401

9	8	3	2

10000 possibilities

Side-Channel Attacks

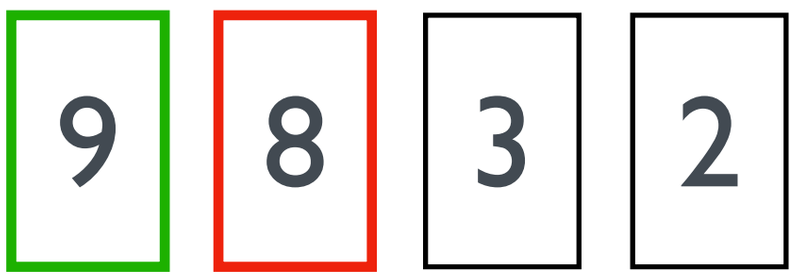
Correct PIN: 9401

9	8	3	2

10000 possibilities

Side-Channel Attacks

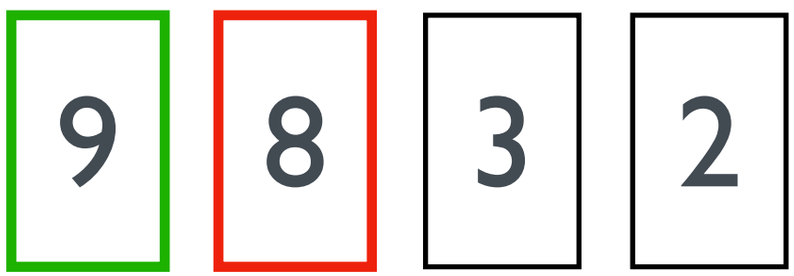
Correct PIN: 9401



10000 possibilities

Side-Channel Attacks

Correct PIN: 9401

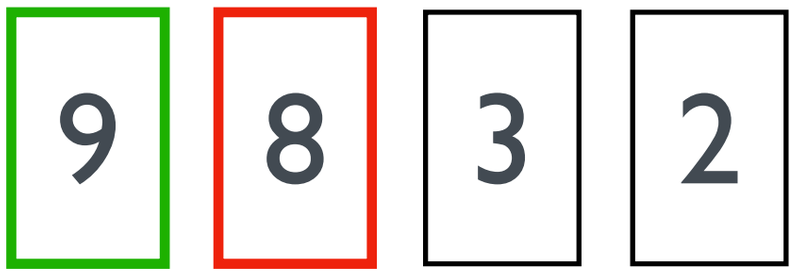


Execution time: 2α ms

10000 possibilities

Side-Channel Attacks

Correct PIN: 9401

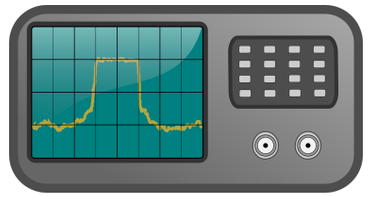


Execution time: 2α ms

$$10 + 10 + 10 + 10 = 40$$

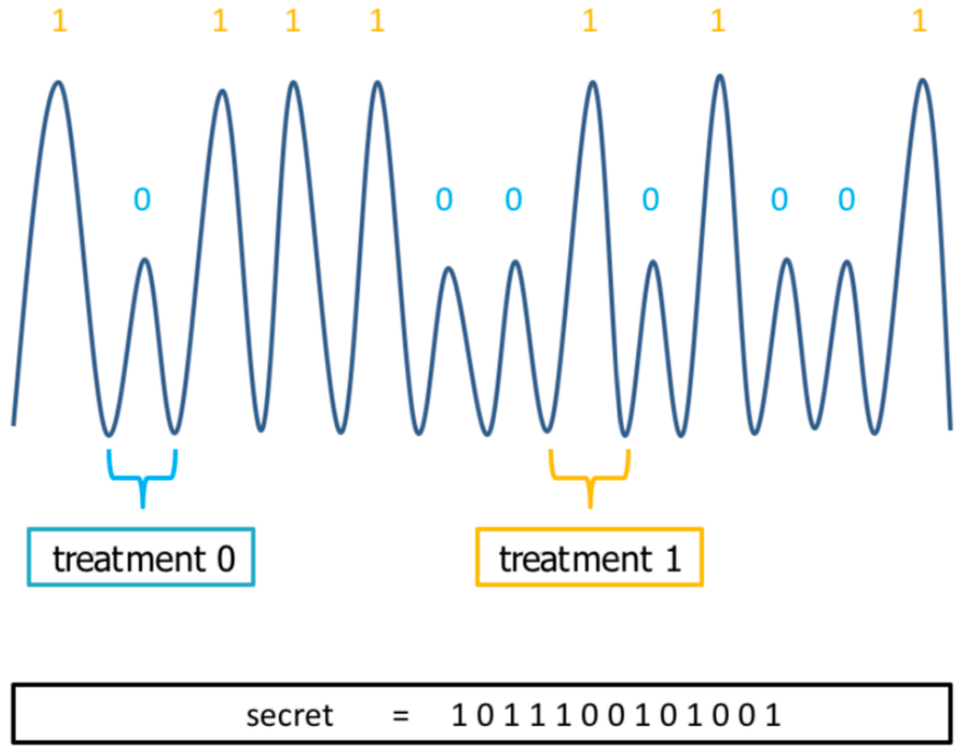
possibilities

Example of SPA



Algorithm 1 Example

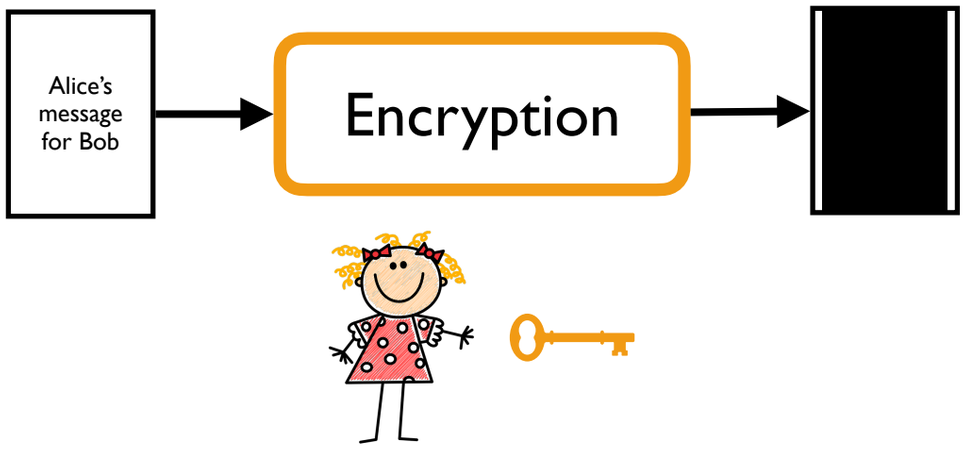
```
for i = 1 to n do
    if key[i] = 0 then
        do treatment 0
    else
        do treatment 1
    end if
end for
```

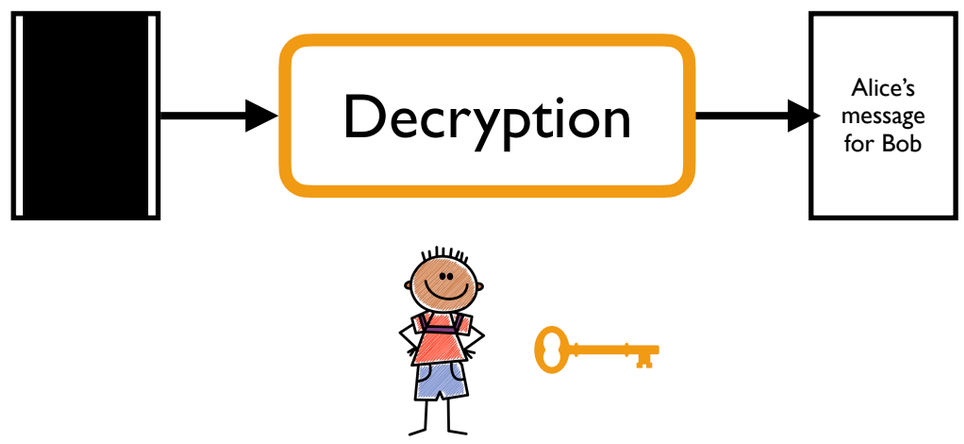


SPA: one single trace to recover the secret key

Example of DPA

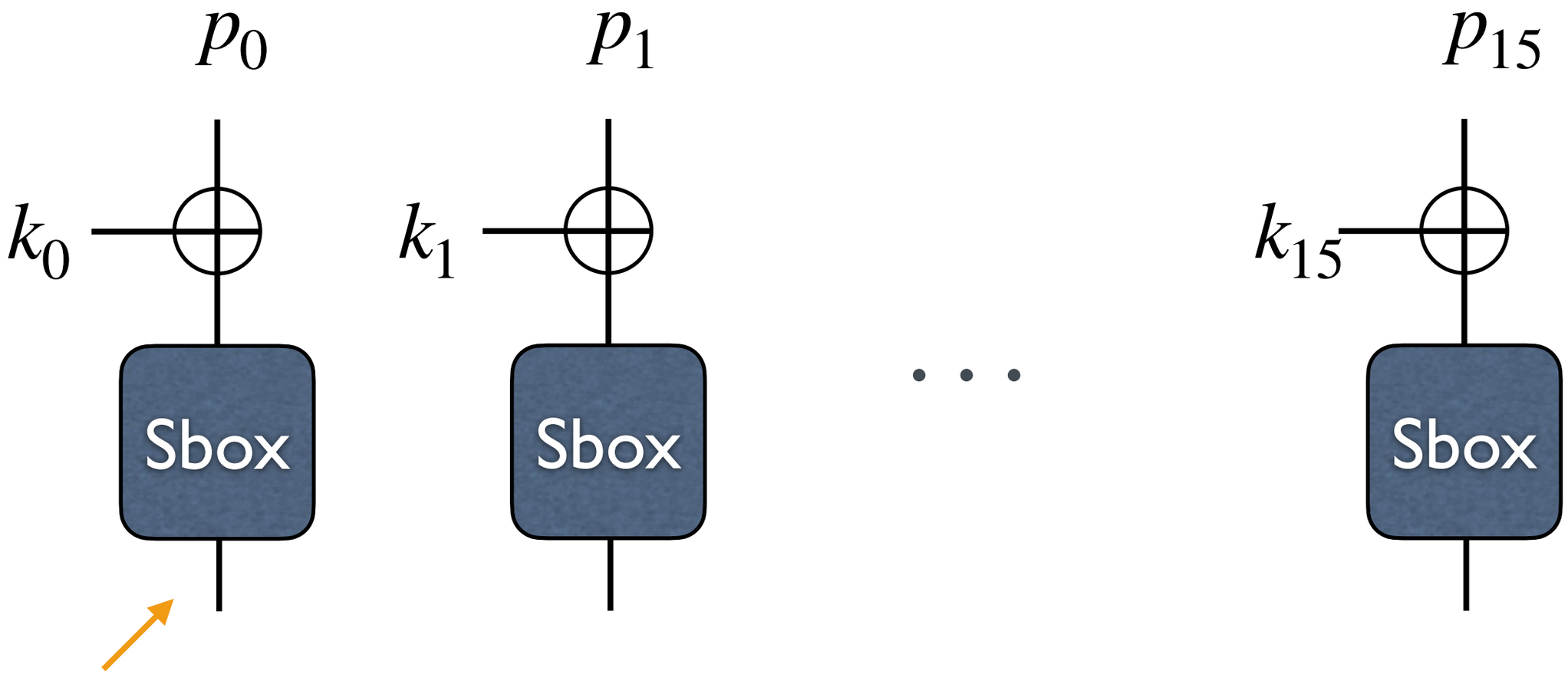
- AES: Advanced Encryption Standard
  - Message $(p_0, p_1, \ldots, p_{15})$ and key $(k_0, k_1, \ldots, k_{15})$ on 16 bytes
  - First round: 16 S-boxes



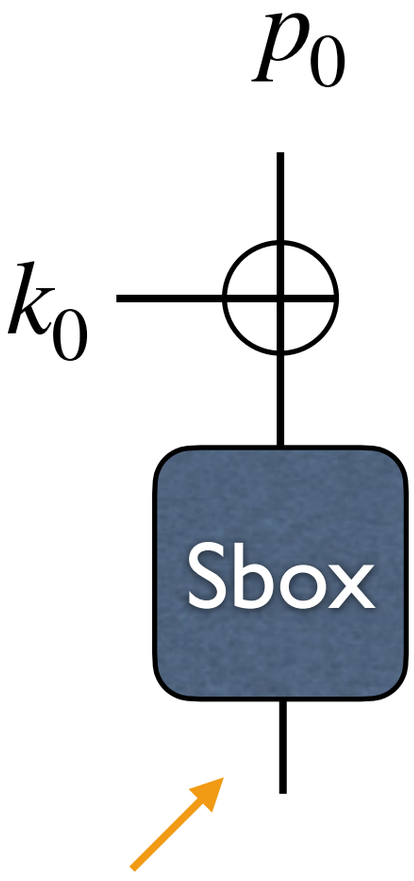


Example of DPA

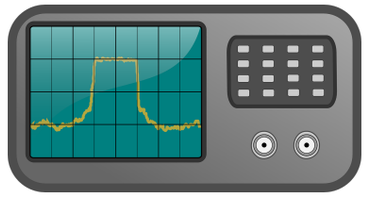
- AES: Advanced Encryption Standard
  - Message $(p_0, p_1, \ldots, p_{15})$ and key $(k_0, k_1, \ldots, k_{15})$ on 16 bytes
  - First round: 16 S-boxes

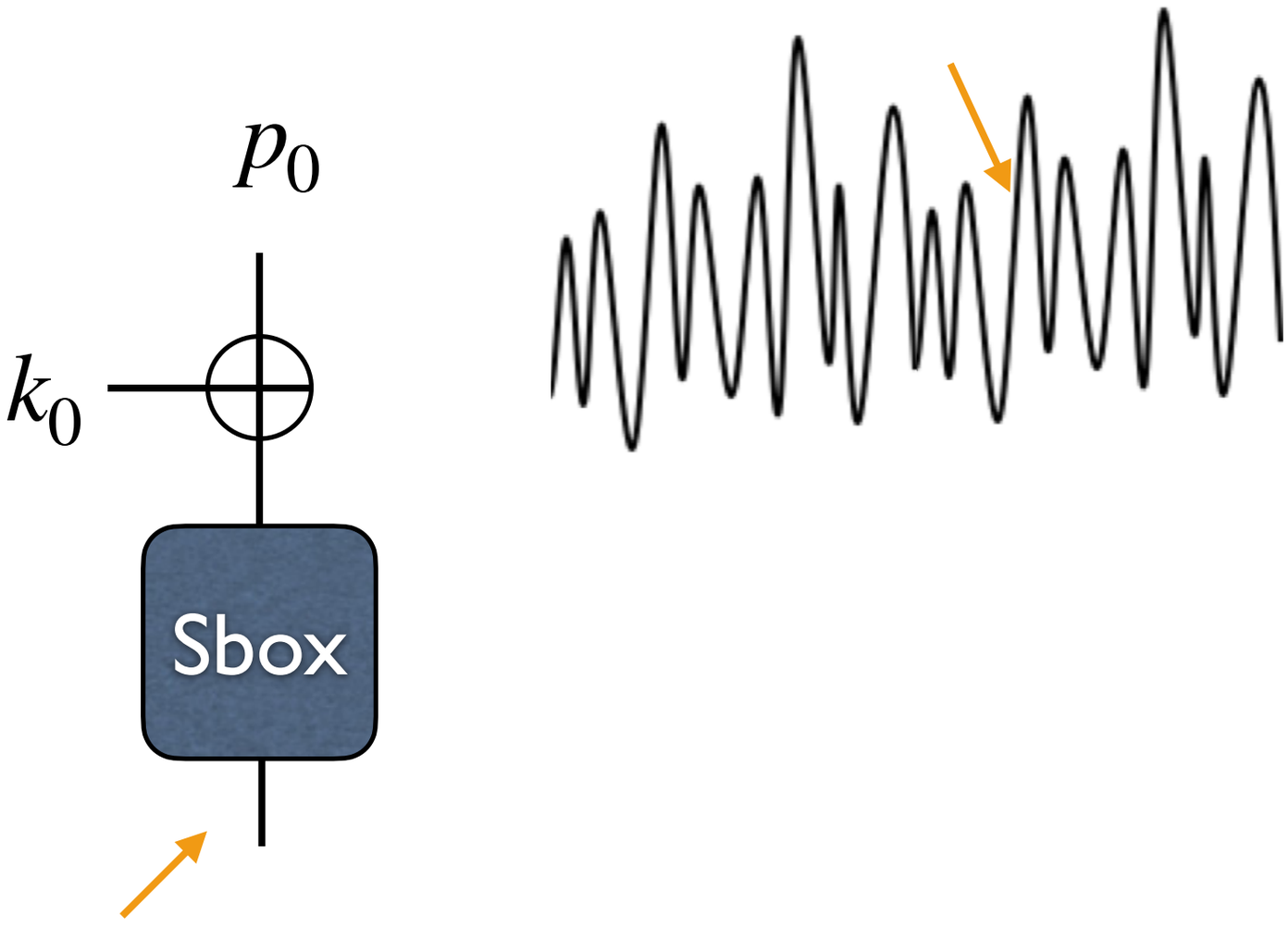


Example of DPA

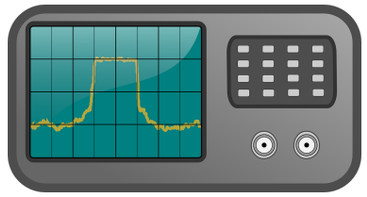


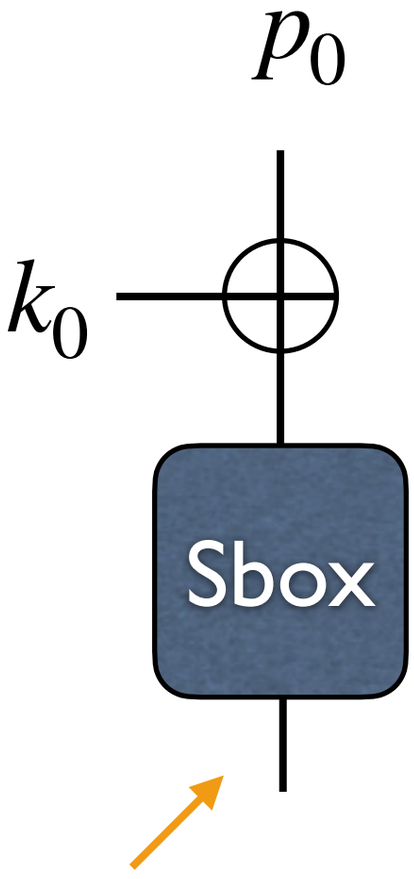
Example of DPA

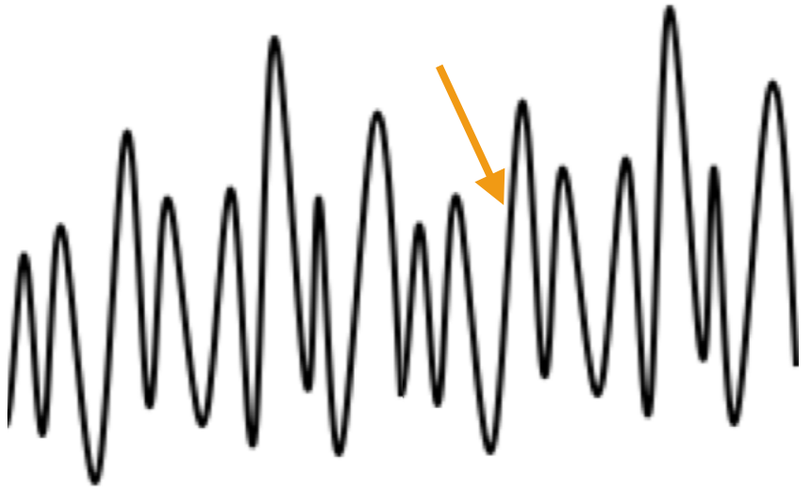




Example of DPA

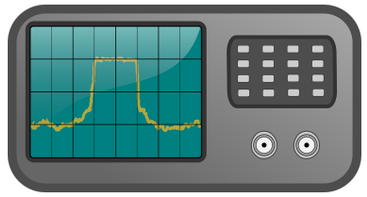


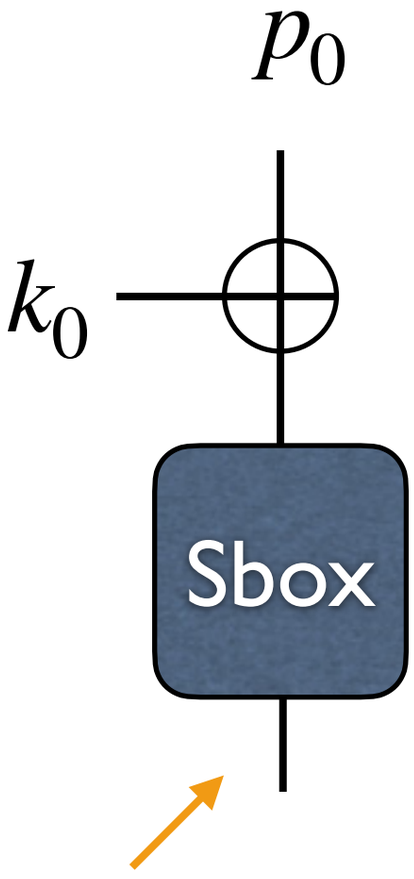


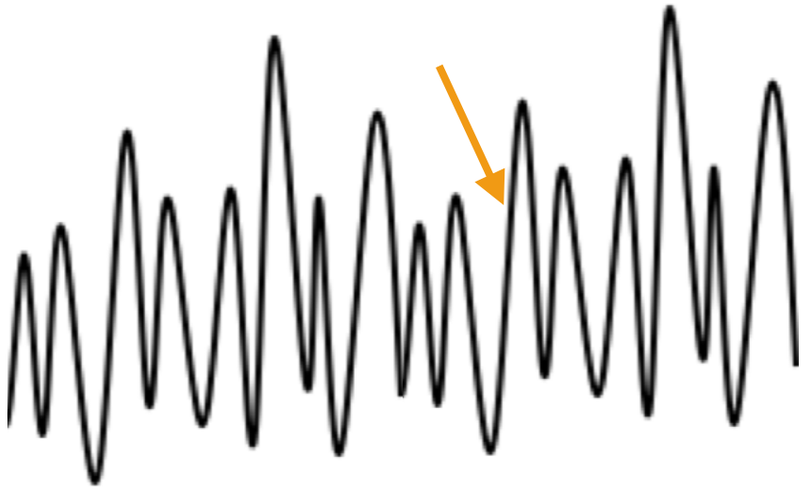


$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

Example of DPA



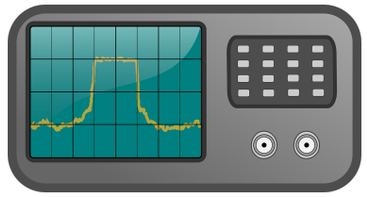


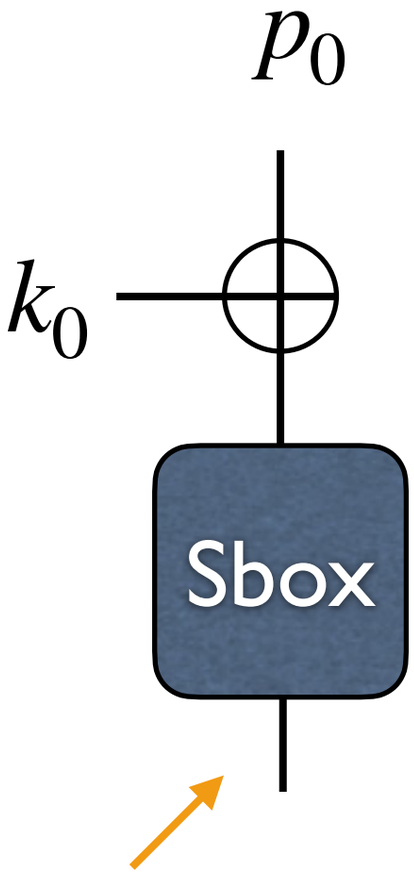


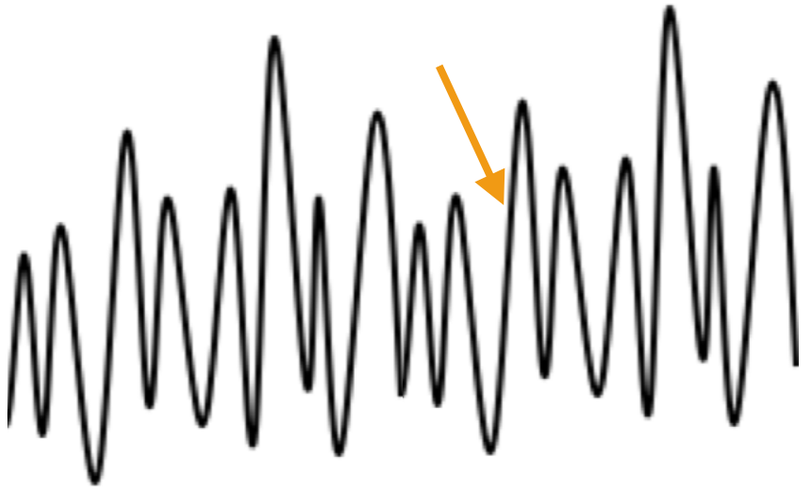
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

$$\begin{pmatrix} \mathsf{Sbox}(p_0^0 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^1 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^2 \oplus \mathbf{0}) \\ \cdots \\ \mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{0}) \end{pmatrix}$$

Example of DPA





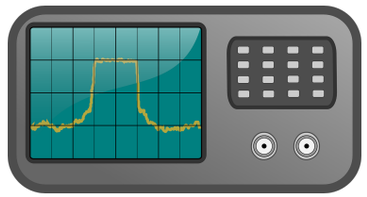


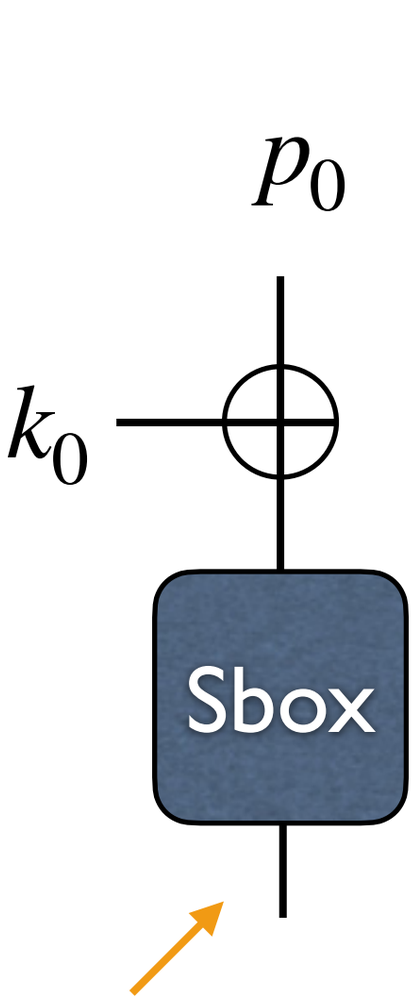
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

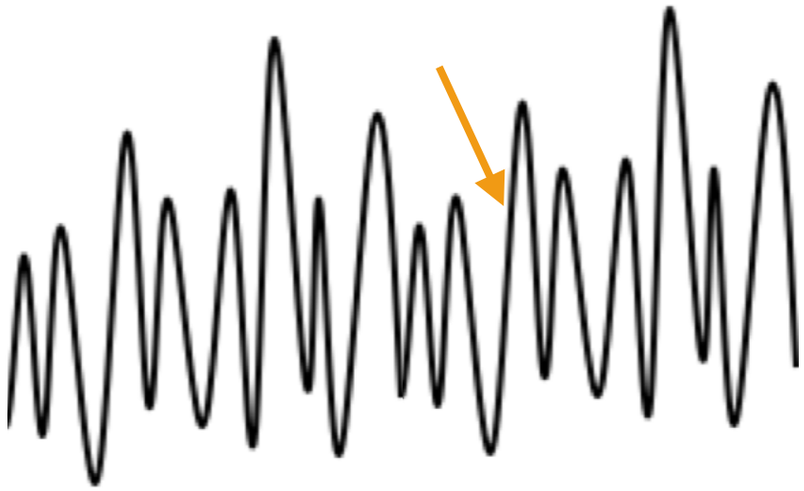
$$\begin{pmatrix} \mathsf{Sbox}(p_0^0 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^1 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^2 \oplus \mathbf{0}) \\ \cdots \\ \mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{0}) \end{pmatrix}$$

$$\begin{pmatrix} \mathsf{HW}(\mathsf{Sbox}(p_0^0 \oplus \mathbf{0})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^1 \oplus \mathbf{0})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^2 \oplus \mathbf{0})) \\ \cdots \\ \mathsf{HW}(\mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{0})) \end{pmatrix}$$

Example of DPA







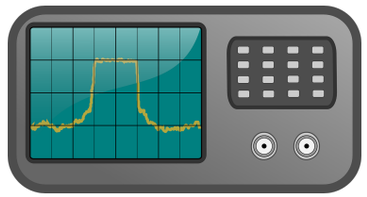
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

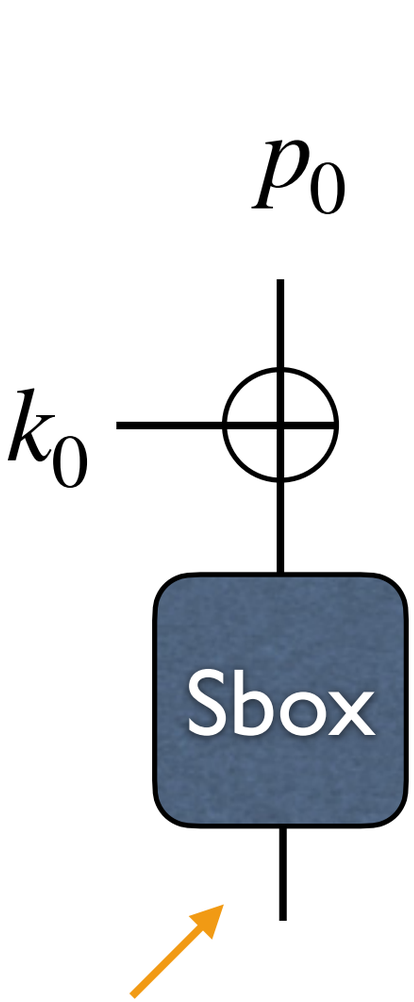
$\mathscr{C}_0$

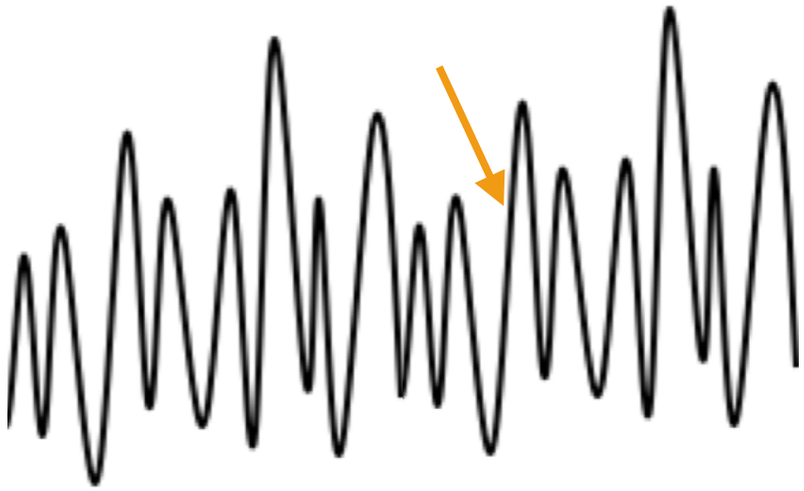
$$\begin{pmatrix} \mathsf{Sbox}(p_0^0 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^1 \oplus \mathbf{0}) \\ \mathsf{Sbox}(p_0^2 \oplus \mathbf{0}) \\ \cdots \\ \mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{0}) \end{pmatrix}$$

$$\begin{pmatrix} \mathsf{HW}(\mathsf{Sbox}(p_0^0 \oplus \mathbf{0})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^1 \oplus \mathbf{0})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^2 \oplus \mathbf{0})) \\ \cdots \\ \mathsf{HW}(\mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{0})) \end{pmatrix}$$

Example of DPA





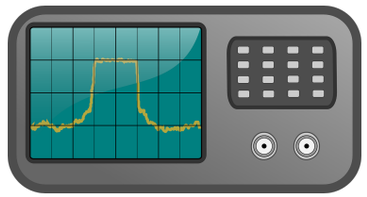


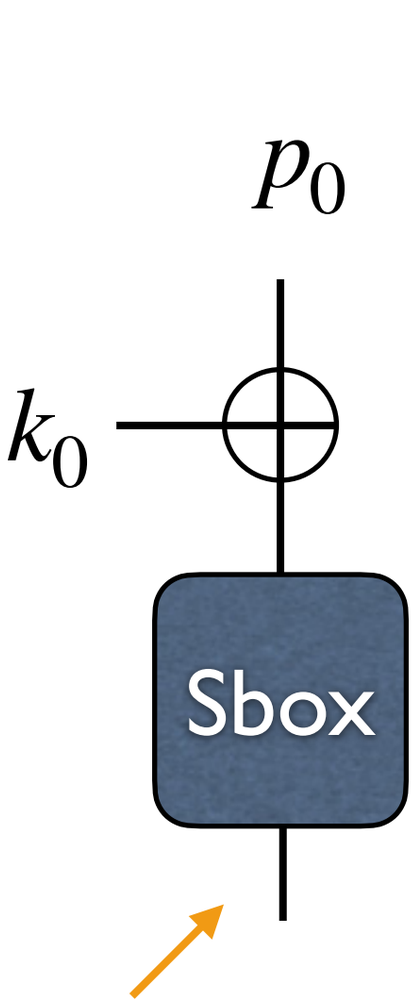
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

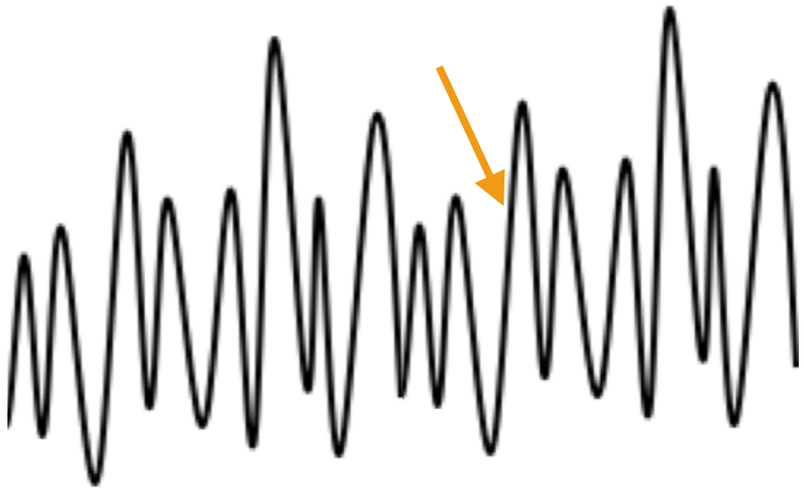
$$(\mathscr{C}_0, \mathscr{C}_1)$$

$$\begin{pmatrix} \text{Sbox}(p_0^0 \oplus \mathbf{1}) \\ \text{Sbox}(p_0^1 \oplus \mathbf{1}) \\ \text{Sbox}(p_0^2 \oplus \mathbf{1}) \\ \cdots \\ \text{Sbox}(p_0^{n-1} \oplus \mathbf{1}) \end{pmatrix} \qquad \begin{pmatrix} \text{HW}(\text{Sbox}(p_0^0 \oplus \mathbf{1})) \\ \text{HW}(\text{Sbox}(p_0^1 \oplus \mathbf{1})) \\ \text{HW}(\text{Sbox}(p_0^2 \oplus \mathbf{1})) \\ \cdots \\ \text{HW}(\text{Sbox}(p_0^{n-1} \oplus \mathbf{1})) \end{pmatrix}$$

Example of DPA





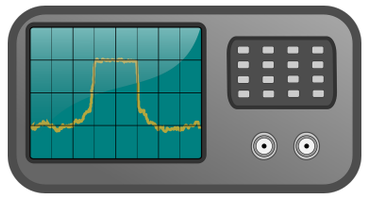


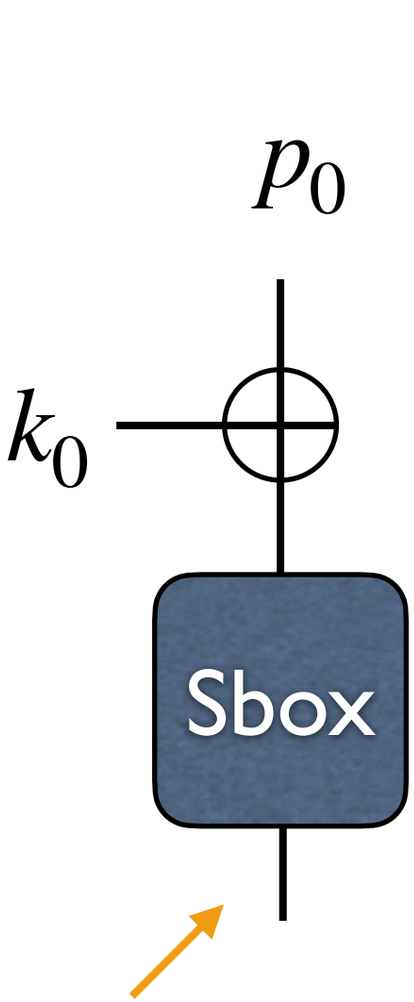
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

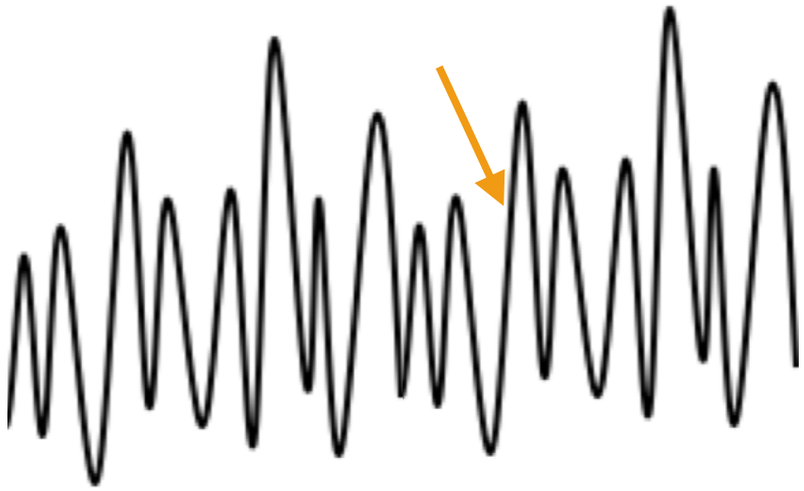
$$(\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2)$$

$$\begin{pmatrix} \mathsf{Sbox}(p_0^0 \oplus \mathbf{2}) \\ \mathsf{Sbox}(p_0^1 \oplus \mathbf{2}) \\ \mathsf{Sbox}(p_0^2 \oplus \mathbf{2}) \\ \cdots \\ \mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{2}) \end{pmatrix} \quad \begin{pmatrix} \mathsf{HW}(\mathsf{Sbox}(p_0^0 \oplus \mathbf{2})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^1 \oplus \mathbf{2})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^2 \oplus \mathbf{2})) \\ \cdots \\ \mathsf{HW}(\mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{2})) \end{pmatrix}$$

Example of DPA







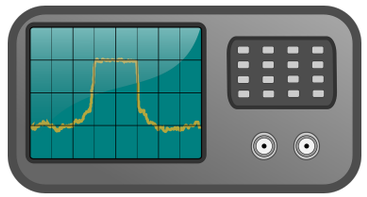
$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

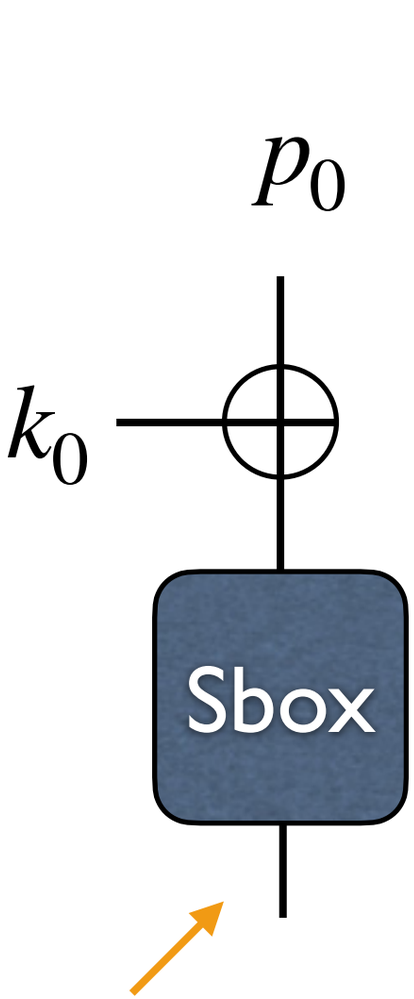
$$(\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_{255})$$

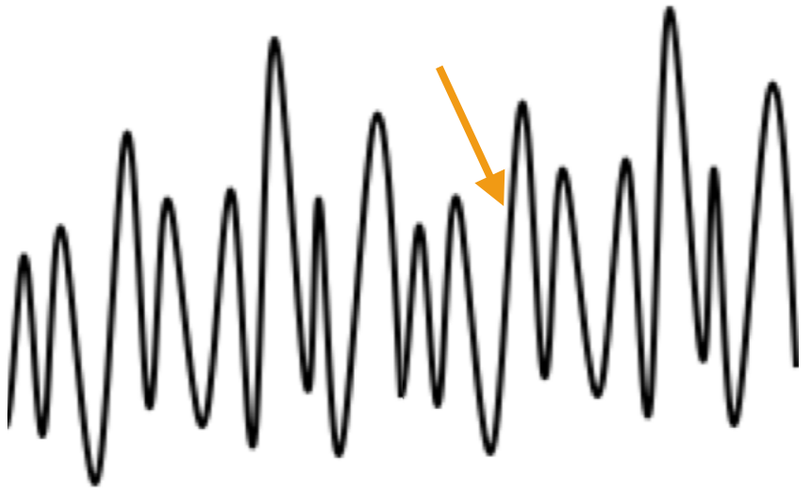
$$\begin{pmatrix} \mathsf{Sbox}(p_0^0 \oplus \mathbf{255}) \\ \mathsf{Sbox}(p_0^1 \oplus \mathbf{255}) \\ \mathsf{Sbox}(p_0^2 \oplus \mathbf{255}) \\ \cdots \\ \mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{255}) \end{pmatrix}$$

$$\begin{pmatrix} \mathsf{HW}(\mathsf{Sbox}(p_0^0 \oplus \mathbf{255})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^1 \oplus \mathbf{255})) \\ \mathsf{HW}(\mathsf{Sbox}(p_0^2 \oplus \mathbf{255})) \\ \cdots \\ \mathsf{HW}(\mathsf{Sbox}(p_0^{n-1} \oplus \mathbf{255})) \end{pmatrix}$$

Example of DPA







$$\begin{pmatrix} v_0 = \mathscr{L}(p_0^0) \\ v_1 = \mathscr{L}(p_0^1) \\ v_2 = \mathscr{L}(p_0^2) \\ \cdots \\ v_{n-1} = \mathscr{L}(p_0^{n-1}) \end{pmatrix}$$

$$\text{argmax } (\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, \ldots, \mathscr{C}_{255})$$

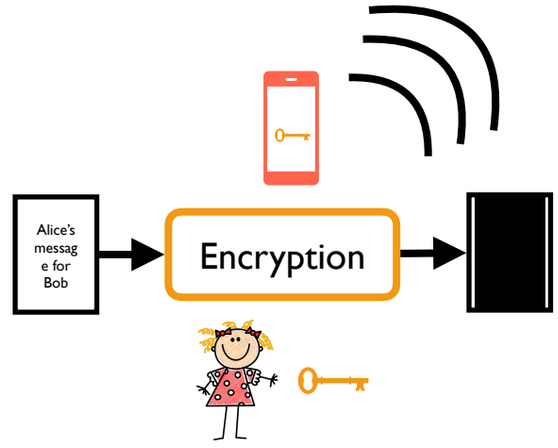
$$\begin{pmatrix} \text{Sbox}(p_0^0 \oplus \mathbf{255}) \\ \text{Sbox}(p_0^1 \oplus \mathbf{255}) \\ \text{Sbox}(p_0^2 \oplus \mathbf{255}) \\ \cdots \\ \text{Sbox}(p_0^{n-1} \oplus \mathbf{255}) \end{pmatrix} \qquad \begin{pmatrix} \text{HW}(\text{Sbox}(p_0^0 \oplus \mathbf{255})) \\ \text{HW}(\text{Sbox}(p_0^1 \oplus \mathbf{255})) \\ \text{HW}(\text{Sbox}(p_0^2 \oplus \mathbf{255})) \\ \cdots \\ \text{HW}(\text{Sbox}(p_0^{n-1} \oplus \mathbf{255})) \end{pmatrix}$$

Side-Channel Attacks

- Cheap equipment
  - Basic oscilloscope is enough
- Few traces
  - Less than a hundred traces to recover secrets in software
  - A few hundreds/thousands traces in hardware
- Fast
  - A few minutes to get the traces
  - A few seconds to mount the attack

Masking Countermeasure

How to thwart SCA?

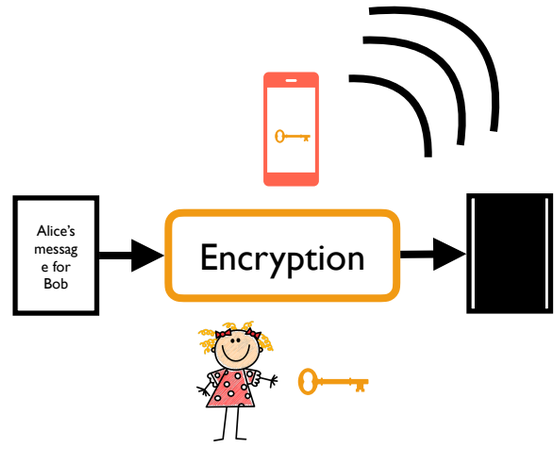


Problem: the leakage is key-dependent

A solution: masking ≈ randomizing the leakage

- replace each sensitive variable v into $(v_1, v_2, \ldots, v_n)$
- such that any tuple of at most $n - 1$ shares is independent from v

How to thwart SCA?



Problem: the leakage is key-dependent

A solution: masking ≈ randomizing the leakage

- replace each sensitive variable v into $(v_1, v_2, \ldots, v_n)$
- such that any tuple of at most $n-1$ shares is independent from v

Example of linear masking:

$$v_1 \leftarrow \$$$
$$v_2 \leftarrow \$$$
$$\ldots$$
$$v_{n-1} \leftarrow \$$$
$$v_n \leftarrow v - v_1 - v_2 - \ldots - v_{n-1}$$

Masking in Practice

- Masking linear operations: $z \leftarrow x + y$
  - Sharing(x) = $(x_1, \ldots, x_n)$
  - Sharing(y) = $(y_1, \ldots, y_n)$

Masking in Practice

- Masking linear operations: $z \leftarrow x + y$
  - Sharing(x) = $(x_1, \ldots, x_n)$
  - Sharing(y) = $(y_1, \ldots, y_n)$

$$\Rightarrow (x_1 + y_1, \ldots, x_n + y_n)$$

Masking in Practice

- Masking linear operations: $z \leftarrow x + y$
  - Sharing$(x) = (x_1, \ldots, x_n)$
  - Sharing$(y) = (y_1, \ldots, y_n)$

$$\Rightarrow (x_1 + y_1, \ldots, x_n + y_n)$$

- Masking non linear operations: $z \leftarrow x \cdot y$
  - Cannot be done share by share
  - Example of multiplication with $n = 2$
    - Sharing$(x) = (x_1, x_2)$
    - Sharing$(y) = (y_1, y_2)$

Masking in Practice

- Masking linear operations: $z \leftarrow x + y$
  - Sharing$(x) = (x_1, \ldots, x_n)$
  - Sharing$(y) = (y_1, \ldots, y_n)$

$$\Rightarrow (x_1 + y_1, \ldots, x_n + y_n)$$

- Masking non linear operations: $z \leftarrow x \cdot y$
  - Cannot be done share by share
  - Example of multiplication with $n = 2$
    - Sharing$(x) = (x_1, x_2)$
    - Sharing$(y) = (y_1, y_2)$

$$\Rightarrow z_1 \leftarrow x_1 \cdot y_1 + x_2 \cdot y_1$$
$$\Rightarrow z_2 \leftarrow x_1 \cdot y_2 + x_2 \cdot y_2$$

Masking in Practice

- Masking linear operations: $z \leftarrow x + y$
  - Sharing(x) = $(x_1, \ldots, x_n)$
  - Sharing(y) = $(y_1, \ldots, y_n)$

$$\Rightarrow (x_1 + y_1, \ldots, x_n + y_n)$$

- Masking non linear operations: $z \leftarrow x \cdot y$
  - Cannot be done share by share
  - Example of multiplication with $n = 2$
    - Sharing(x) = (x_1, x_2)
    - Sharing(y) = (y_1, y_2)

$$\Rightarrow z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$\Rightarrow z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

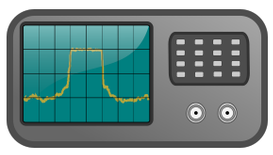
$$r \leftarrow \$$$

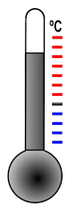
Security of an implementation

- How to evaluate the security of an implementation?

Security of an implementation

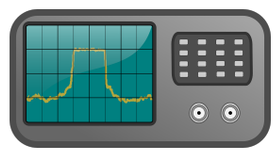
- How to evaluate the security of an implementation?
  - Integrate it on a device and try to attack it
    - Not always possible

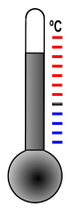




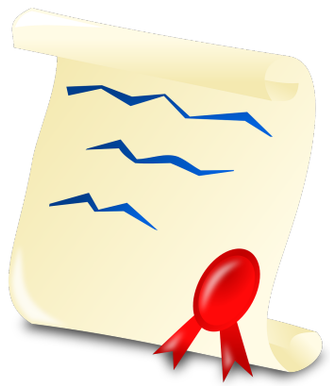
Security of an implementation

- How to evaluate the security of an implementation?
  - Integrate it on a device and try to attack it
    - Not always possible



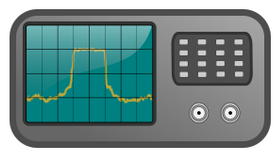


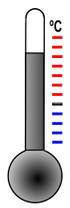
  - Model the leakage and prove its security or exhibit an attack



Security of an implementation

- How to evaluate the security of an implementation?
  - Integrate it on a device and try to attack it
    - Not always possible

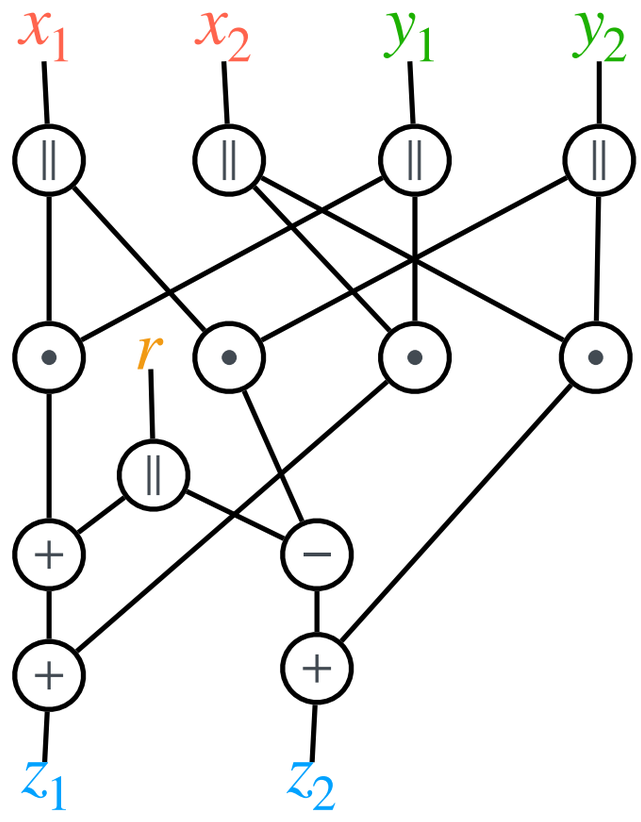




- Model the leakage and prove its security or exhibit an attack

Leakage Models

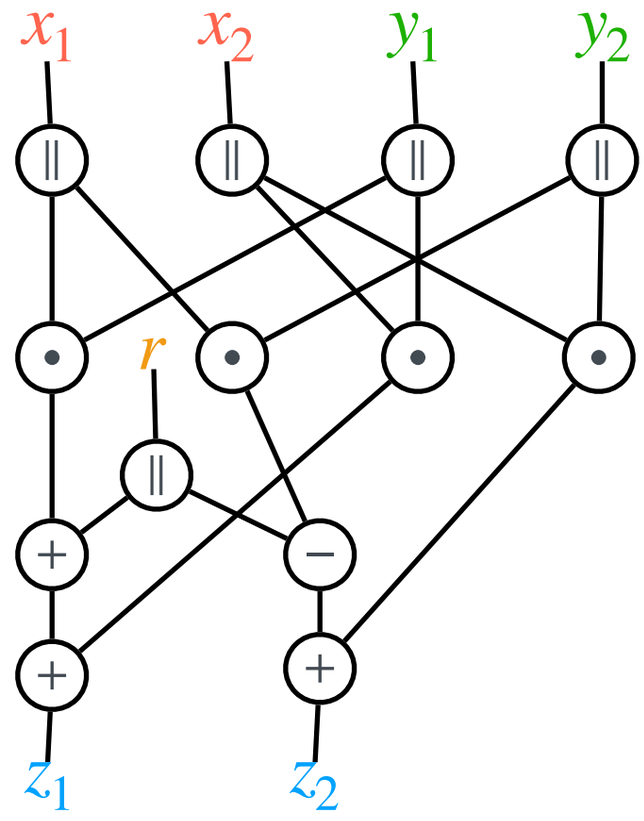
$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$



Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value



Security in the t-probing model: any set of t intermediate variables is independent from the secret

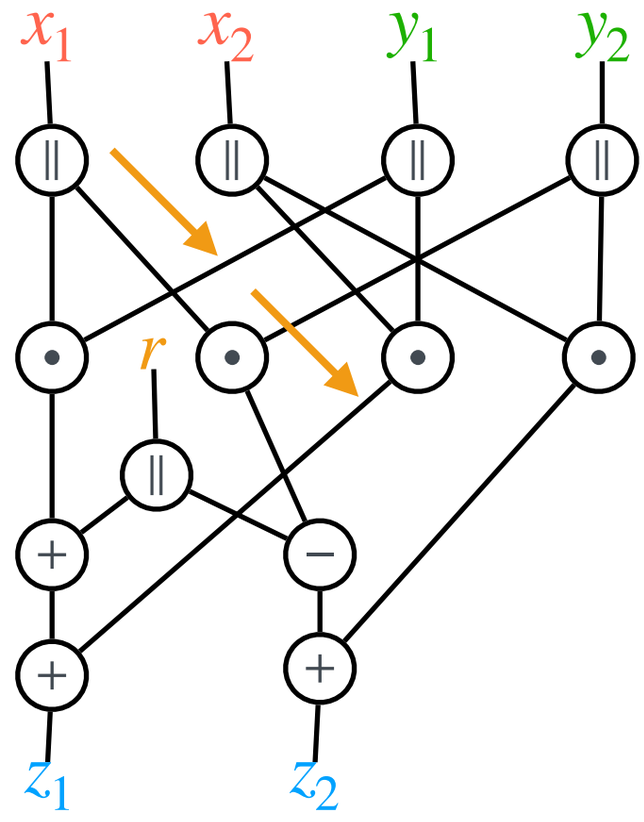
Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value

Example with $t = 2$

↘ Probe 1: y_1
↘ Probe 2: $x_2 \cdot y_1$



Security in the t-probing model: any set of t intermediate variables is independent from the secret

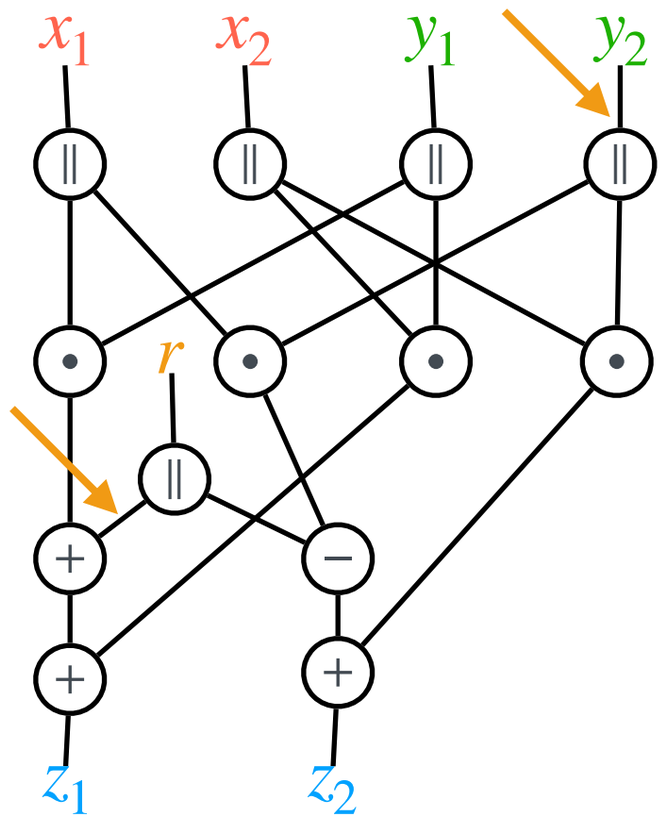
Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value

Example with $t = 2$

↘ Probe 1: y_2
↘ Probe 2: r



Security in the t-probing model: any set of t intermediate variables is independent from the secret

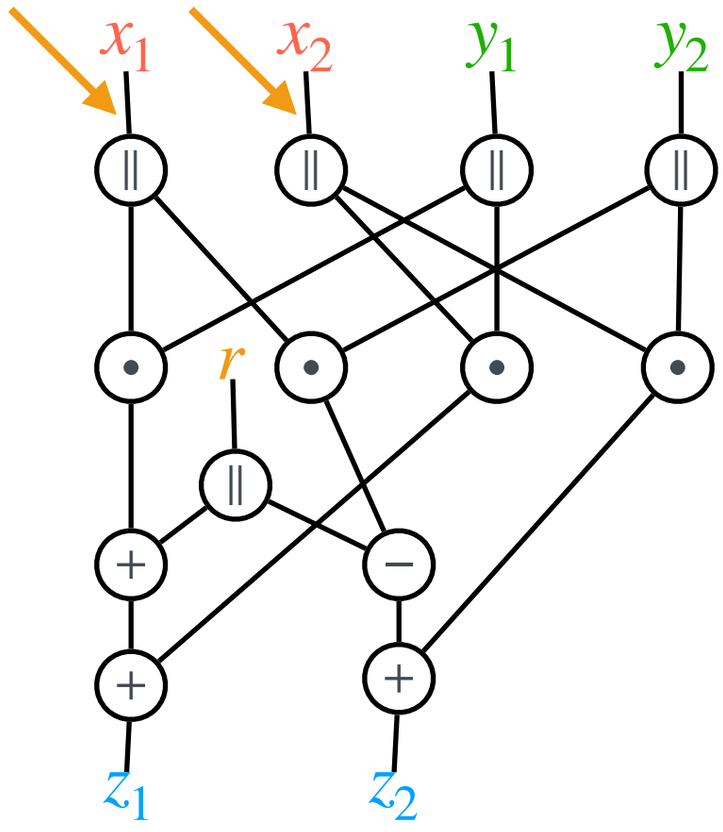
Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value

Example with $t = 2$

↘ Probe 1: x_1
↘ Probe 2: x_2

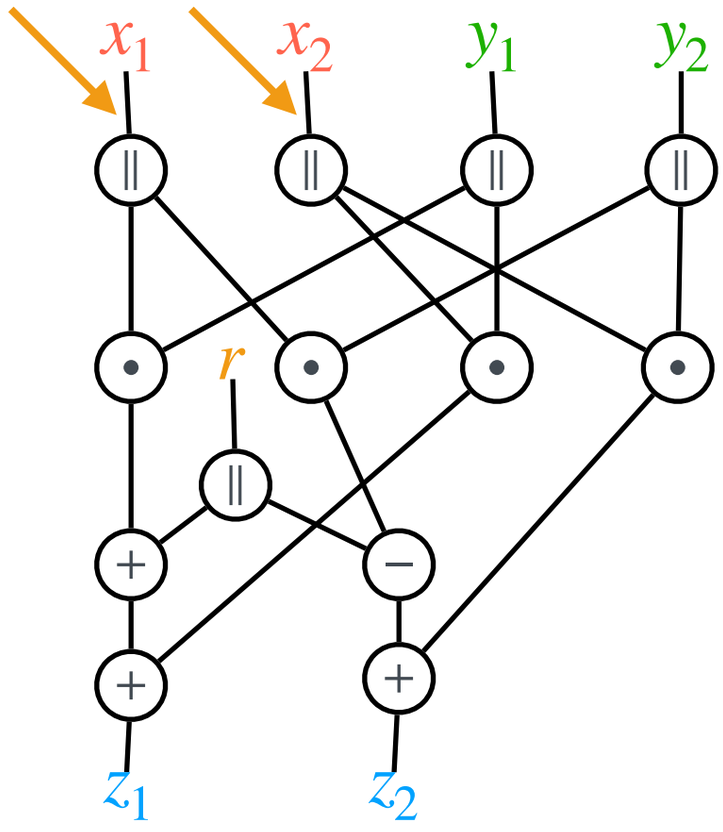


Security in the t-probing model: any set of t intermediate variables is independent from the secret

Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value



Example with $t = 2$

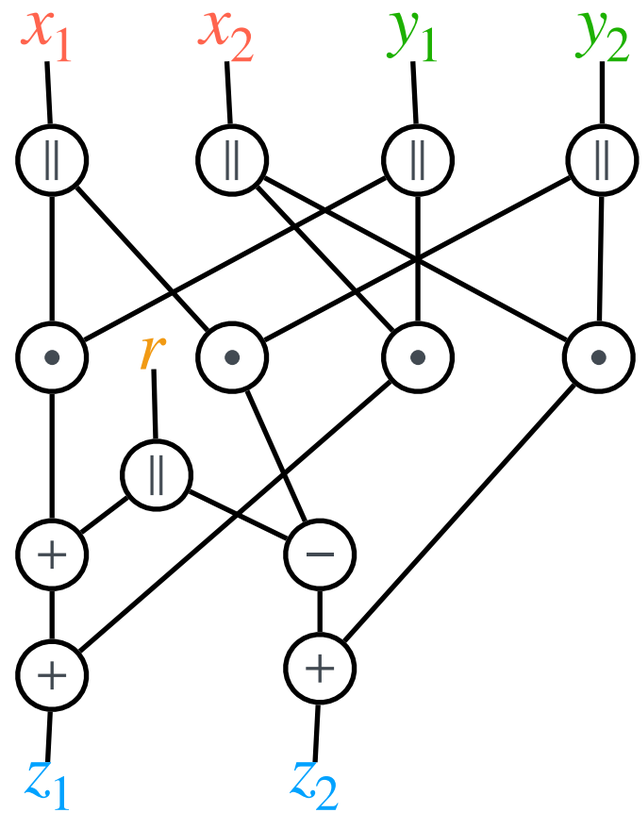
↘ Probe 1: x_1
↘ Probe 2: x_2

Secret $x = x_1 + x_2$

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$



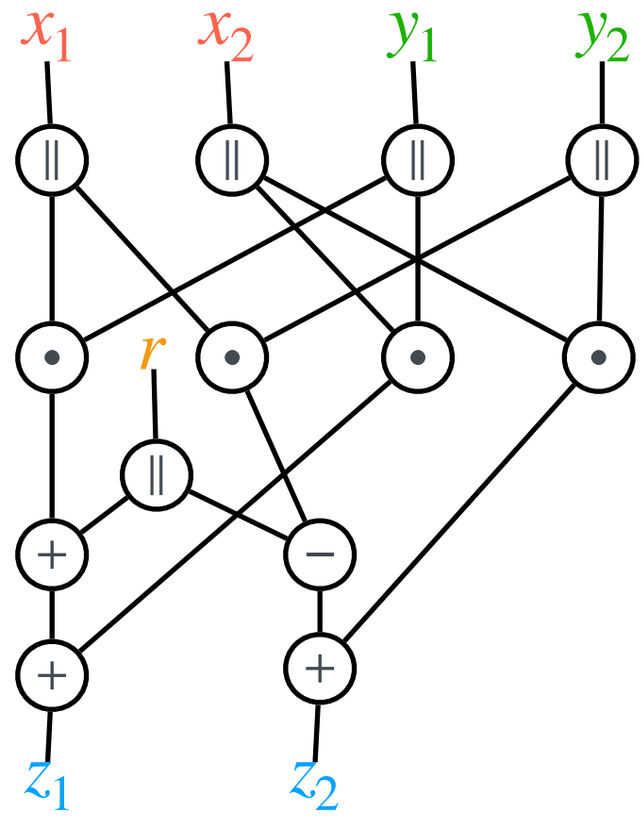
- Leakage
  - Only t variables leak in the implementation
  - Leakage = exact value
- Pros and Cons
  - Easy to make security proofs
  - Not that close to the reality…

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Random Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks with probability p
  - Leakage = exact value

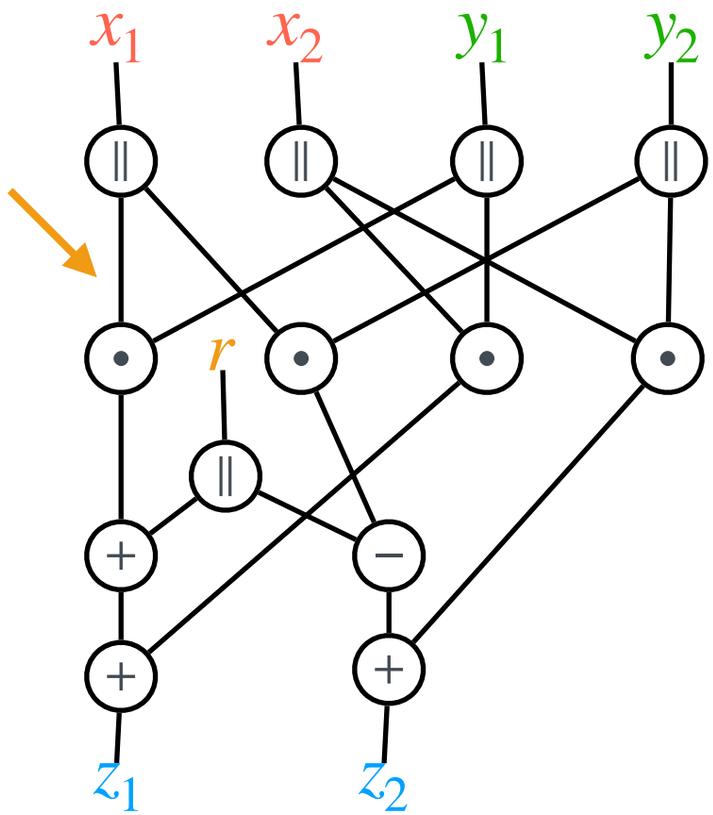


Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Random Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks with probability p
  - Leakage = exact value



Example

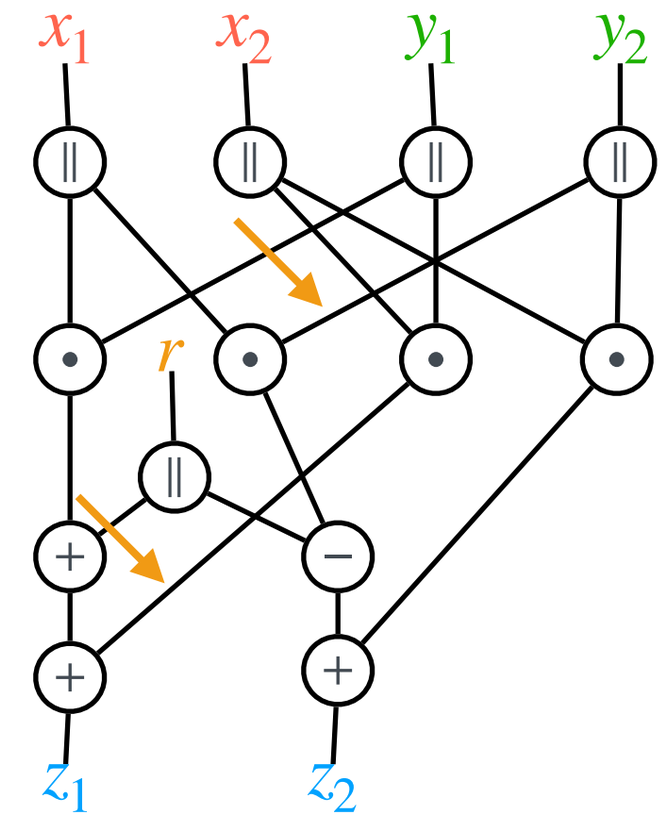
↘ 1 probe with probability $p(1-p)^{s-1}$

Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Random Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks with probability p
  - Leakage = exact value



Example

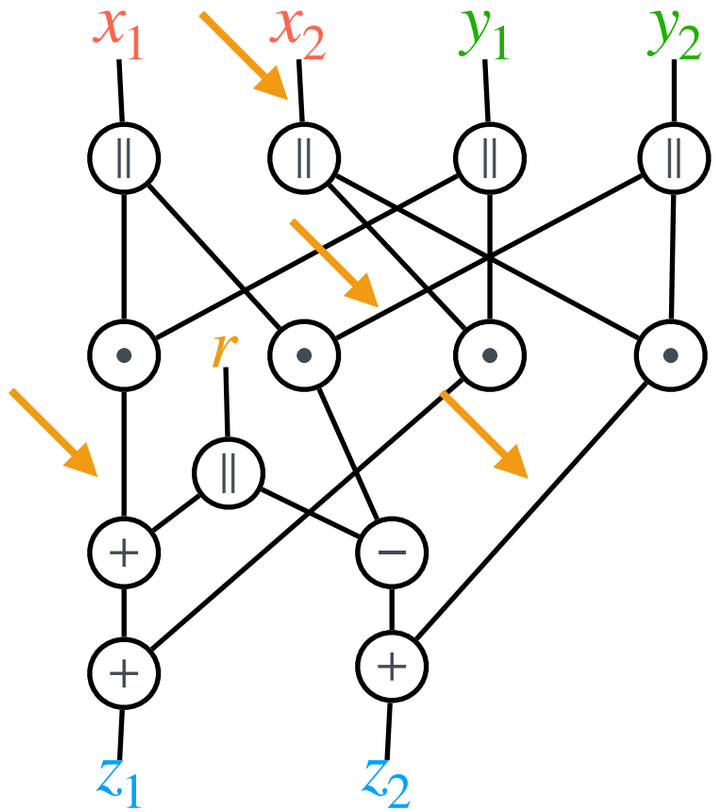
↘ 1 probe with probability $p(1-p)^{s-1}$
↘ 2 probes with probability $p^2(1-p)^{s-2}$

Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Random Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks with probability p
  - Leakage = exact value



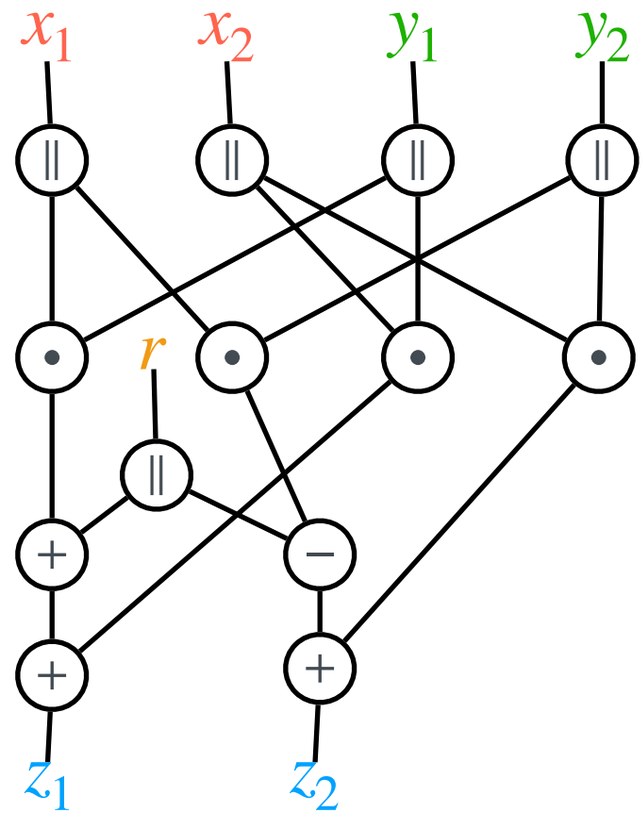
Example

- 1 probe with probability $p(1-p)^{s-1}$
- 2 probes with probability $p^2(1-p)^{s-2}$
- i probes with probability $p^i(1-p)^{s-i}$

Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Random Probing Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$



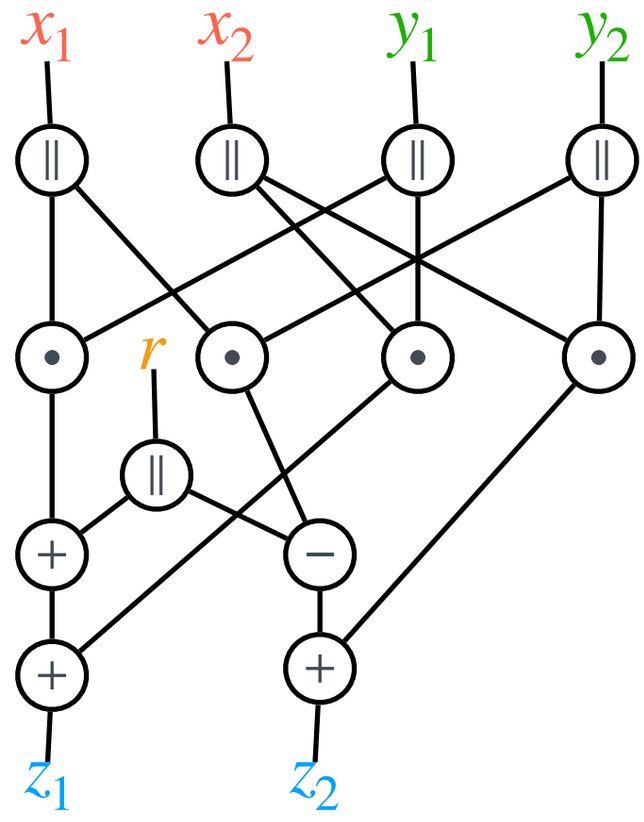
- Leakage
  - Every variable leaks with probability p
  - Leakage = exact value
- Pros and Cons
  - A bit more complicated to make security proofs
  - Closer to the reality

Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Noisy Leakage Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks
  - Leakage = noisy function of the value



Security in the (σ, ε)-noisy leakage model: given the noise standard deviation σ, the probability to recover information on the secret is bounded by ε.

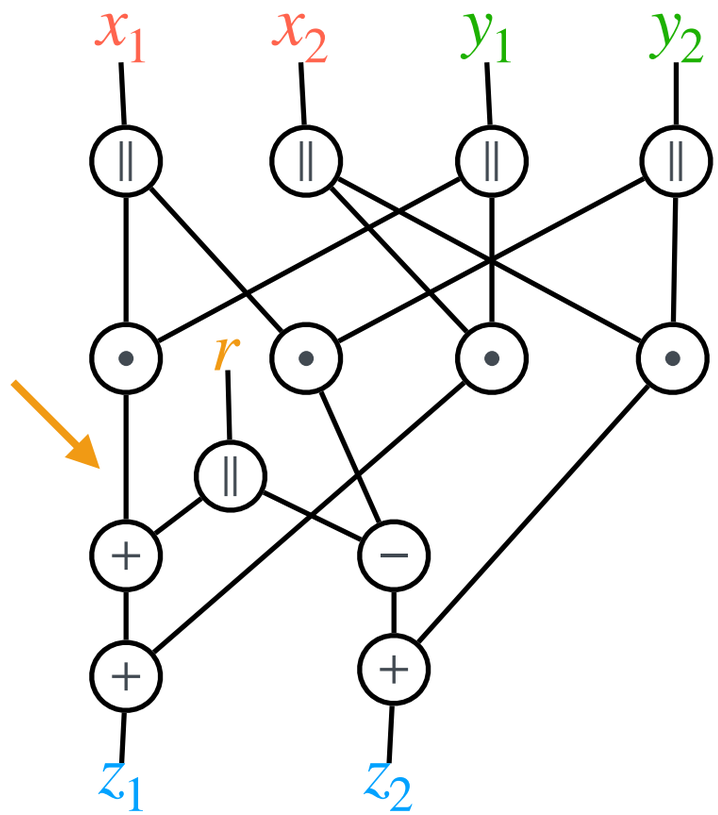
Noisy Leakage Model

$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$

- Leakage
  - Every variable leaks
  - Leakage = noisy function of the value

Example

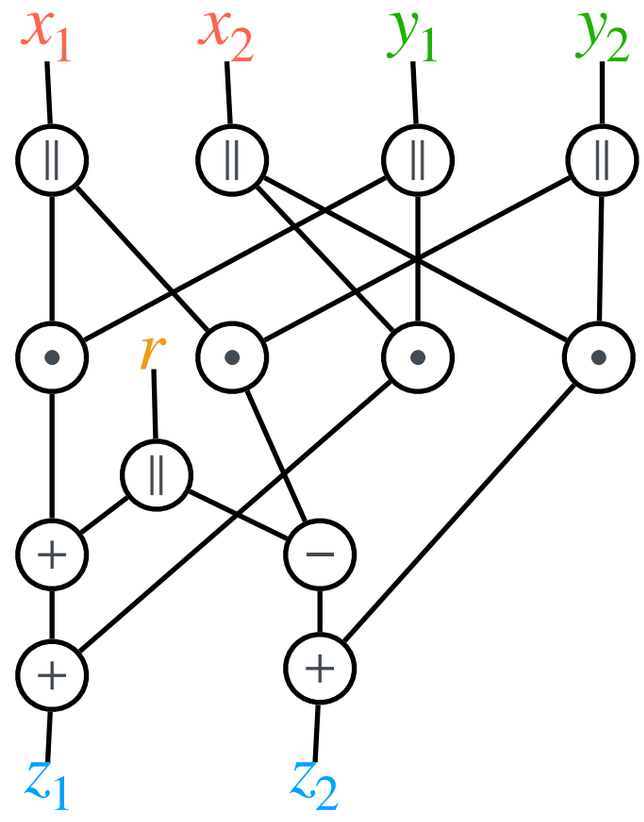
↘ The adversary gets $f(x_1 \cdot y_1) + \eta$



Security in the (σ, ε)-noisy leakage model: given the noise standard deviation σ, the probability to recover information on the secret is bounded by ε.

Noisy Leakage Model

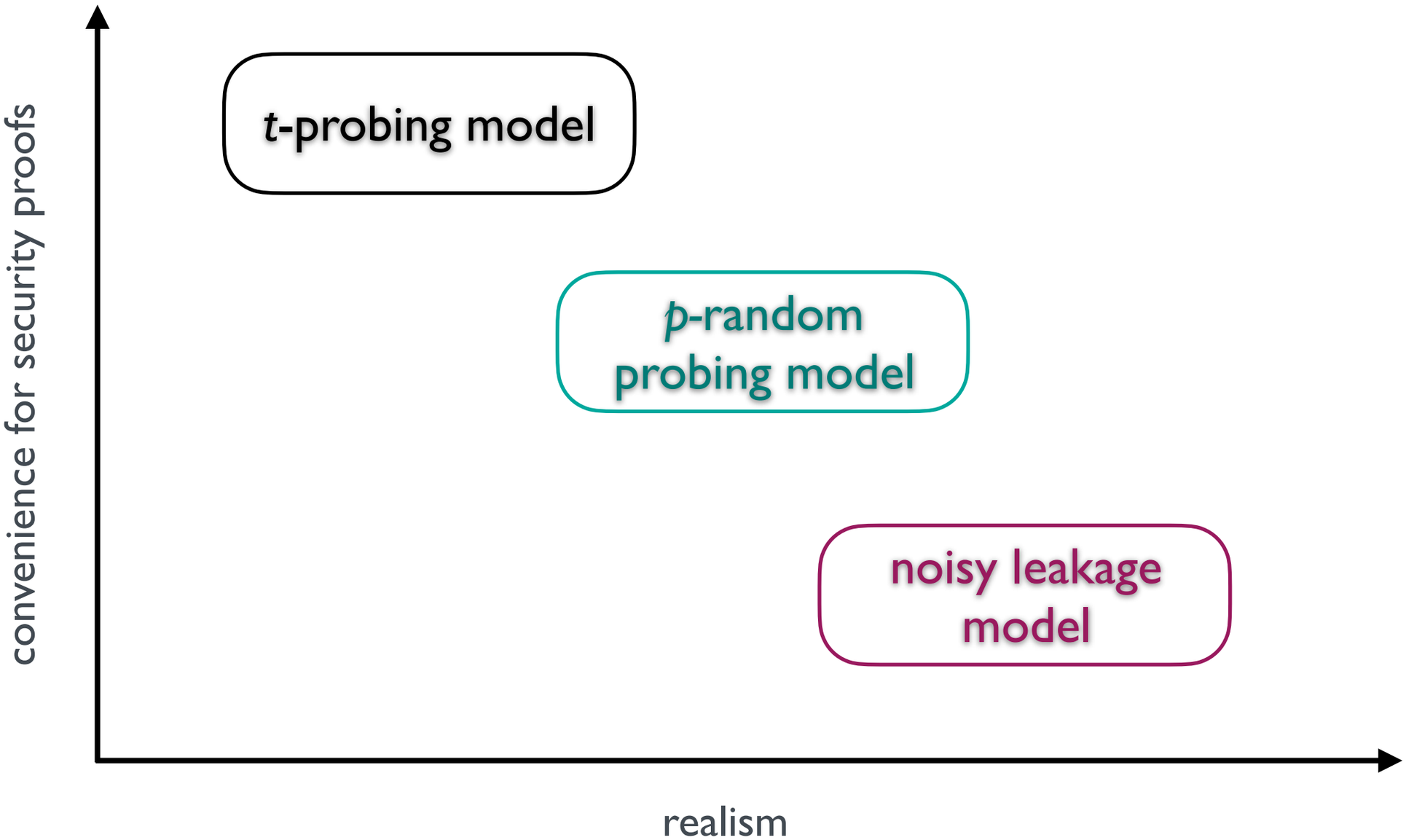
$$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$$
$$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$$



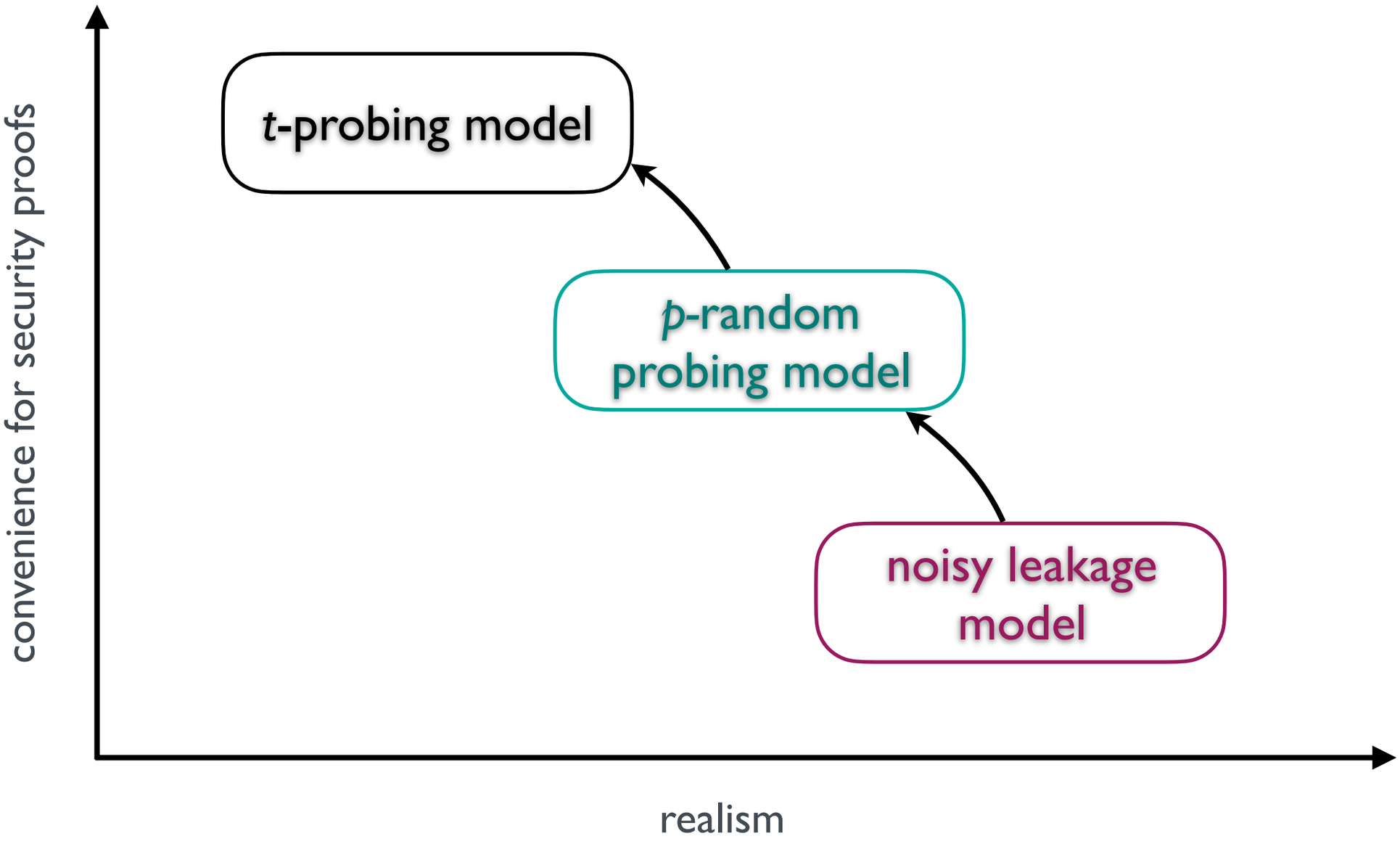
- Leakage
  - Every variable leaks
  - Leakage = noisy function of the value
- Pros and Cons
  - Much more complicated to make security proofs
  - The closest to the reality

Security in the (σ, ε)-noisy leakage model: given the noise standard deviation σ, the probability to recover information on the secret is bounded by ε.

Reductions


convenience for security proofs
realism
t-probing model
p-random probing model
noisy leakage model

Reductions


convenience for security proofs
realism
t-probing model
p-random probing model
noisy leakage model

Security Proofs

Security Proofs

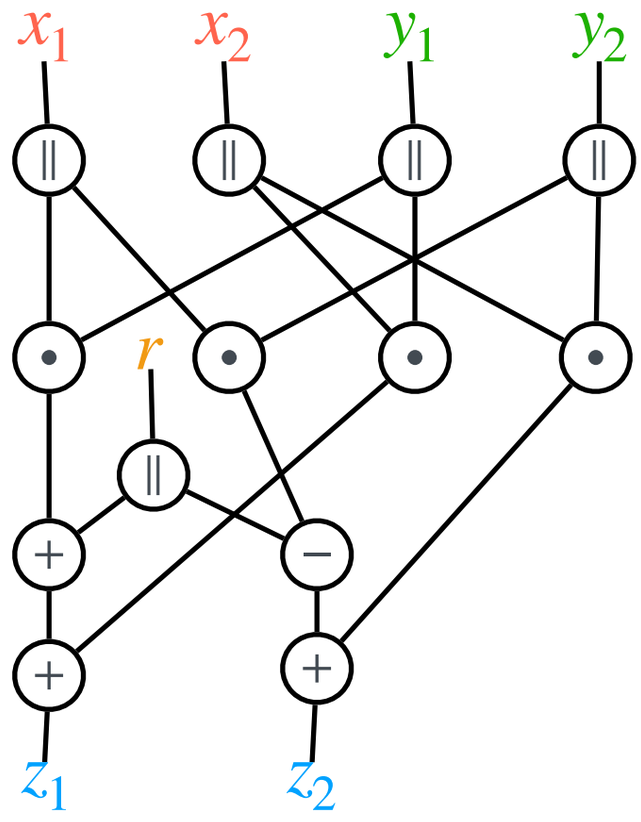
- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
  - Probing Security
  - Random Probing Security

Security Proofs

- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
    - Probing Security
    - Random Probing Security

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

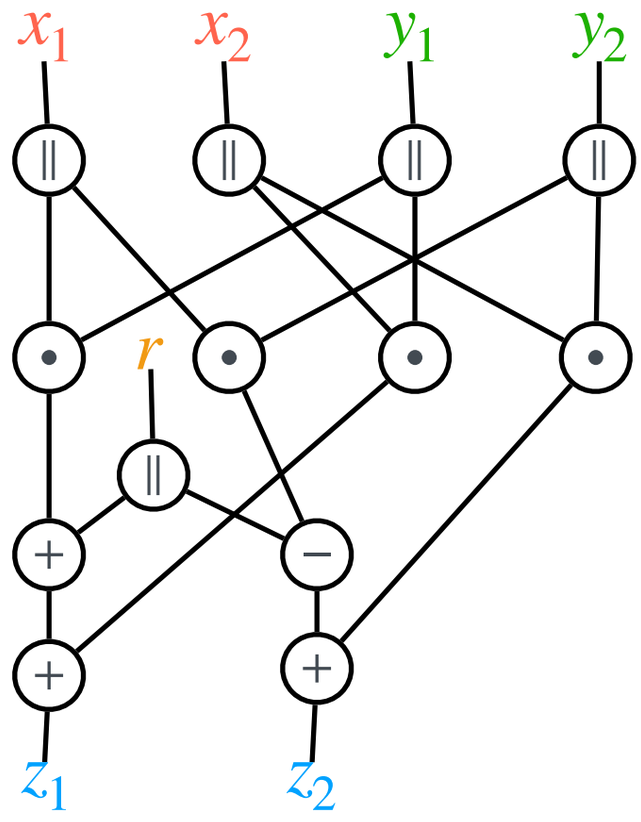


Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

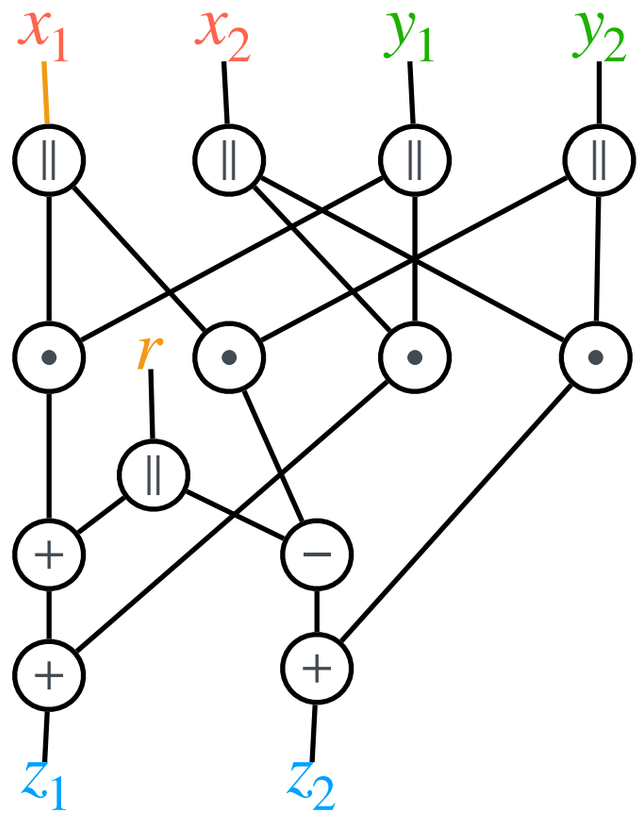
1-probing secure?



Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

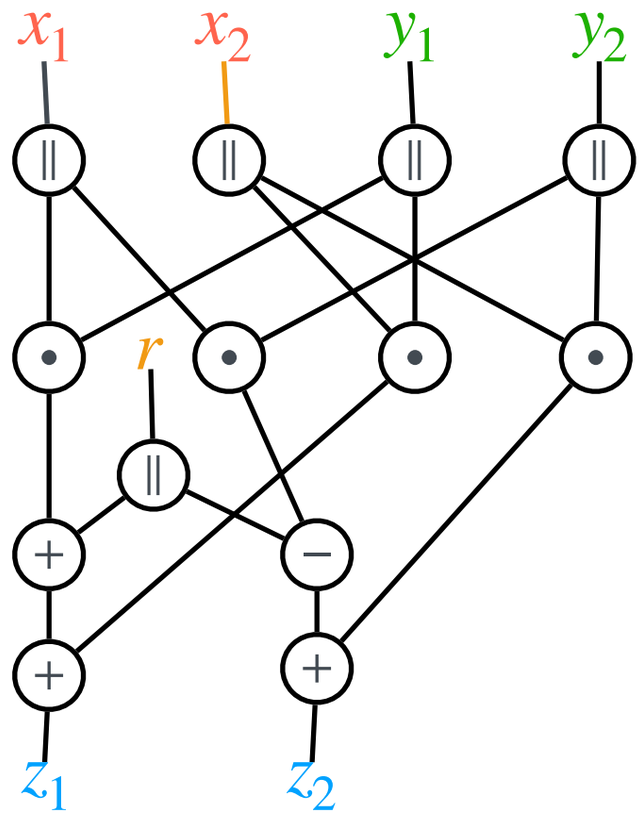
Independent from secrets?

$x_1 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

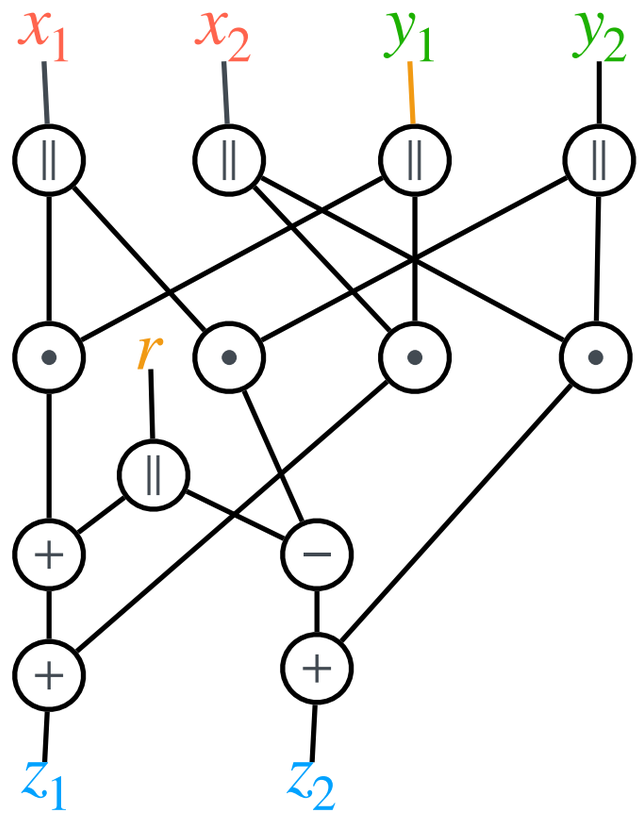
$x_2 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

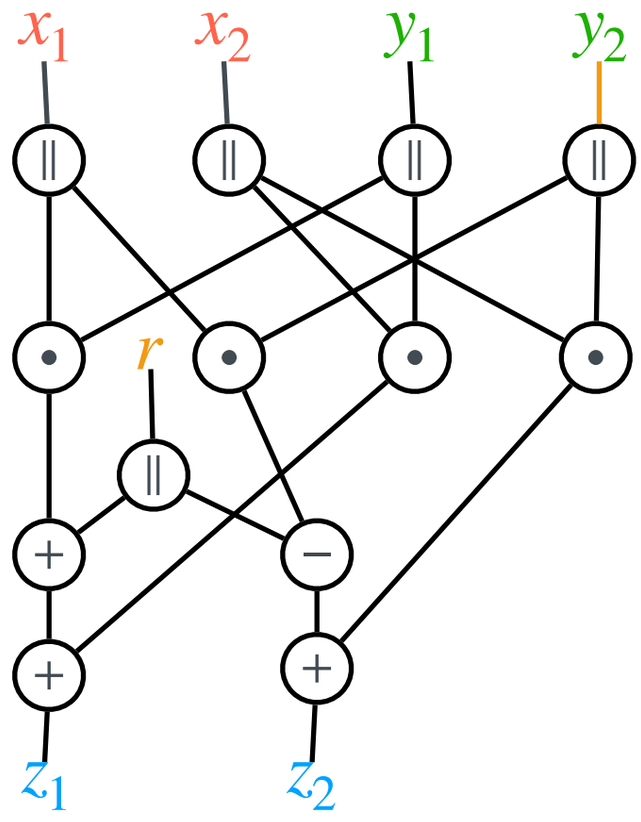
$y_1 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

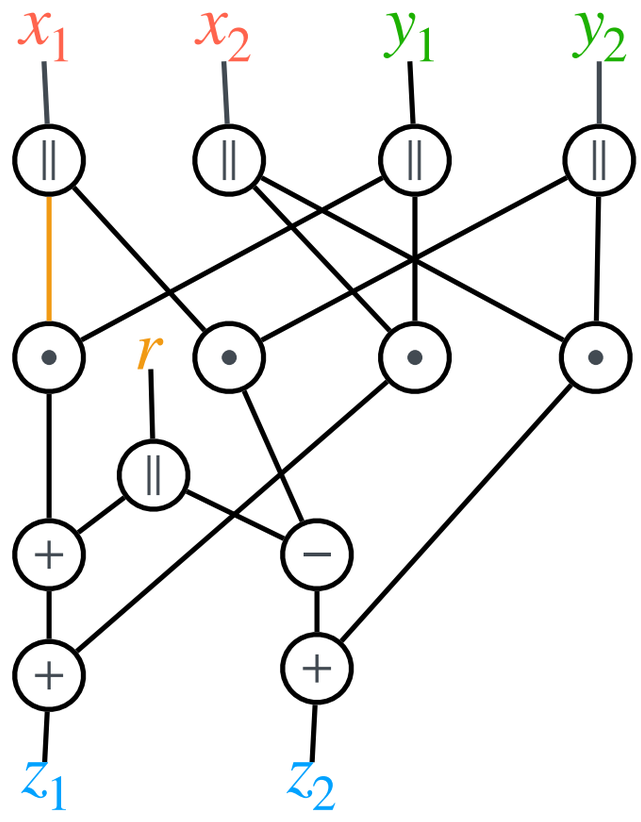
$y_2 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

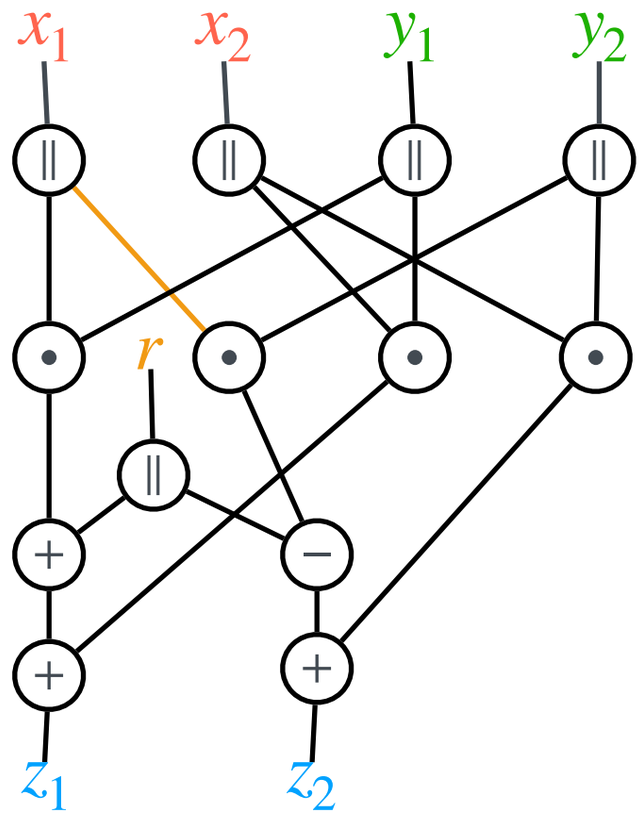
Independent from secrets?

$x_1 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

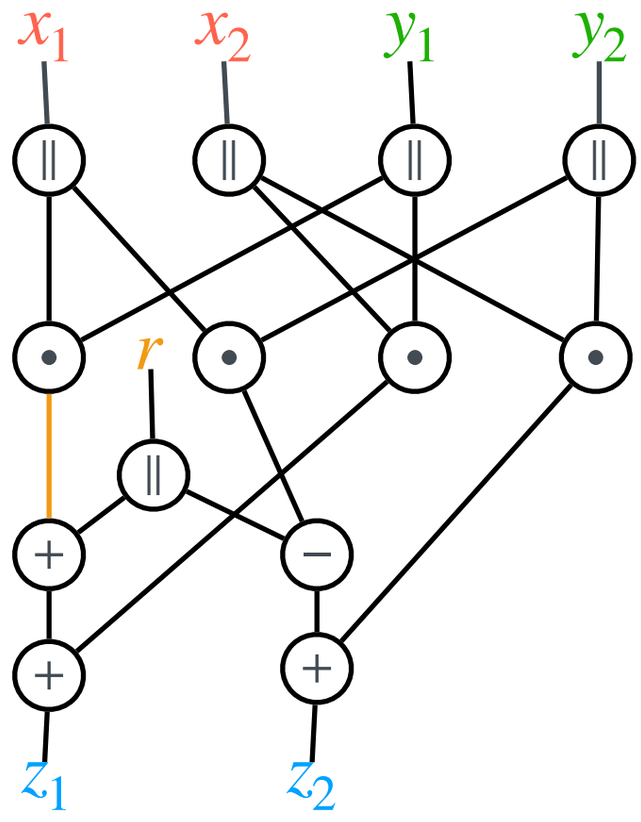
$x_1 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

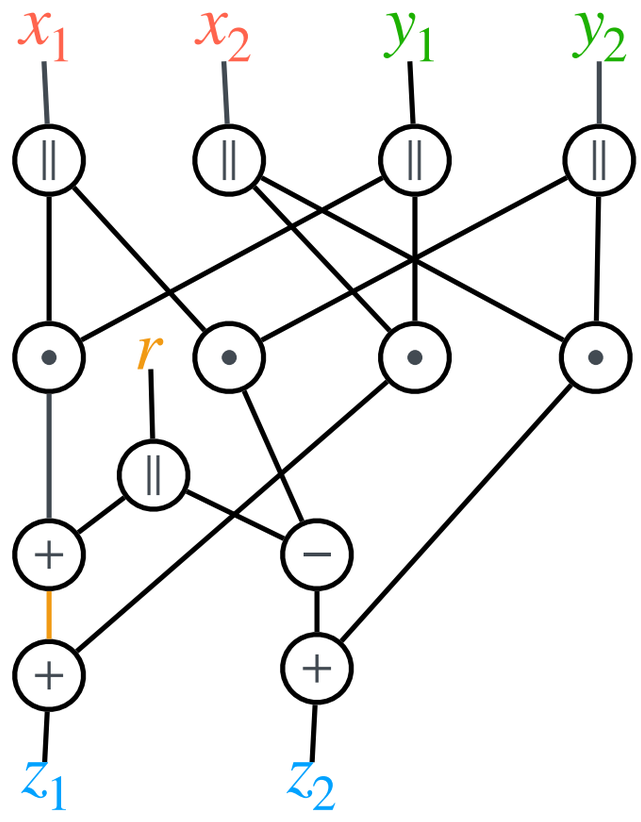
$x_1 \cdot y_1 \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

Independent from secrets?

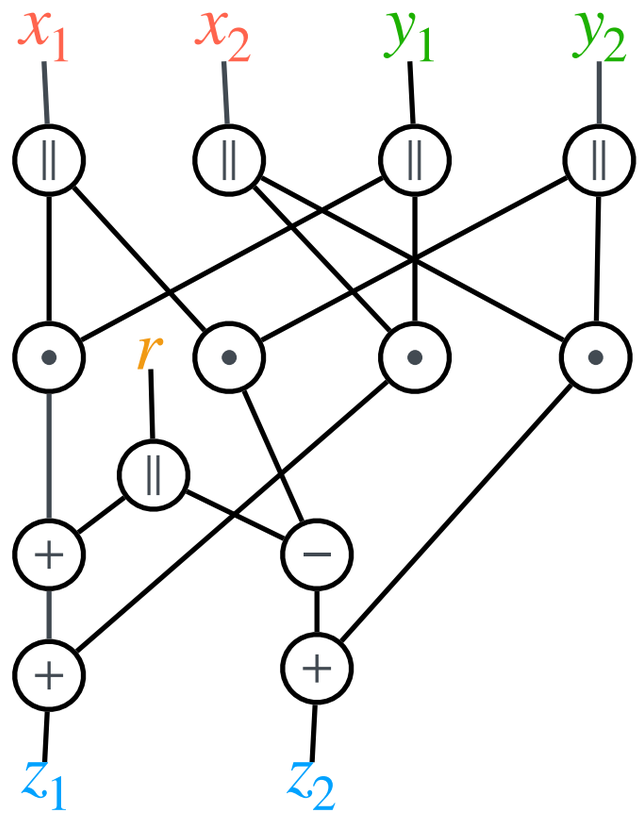
$x_1 \cdot y_1 + r \rightarrow$ ✔

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$

$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1-probing secure?

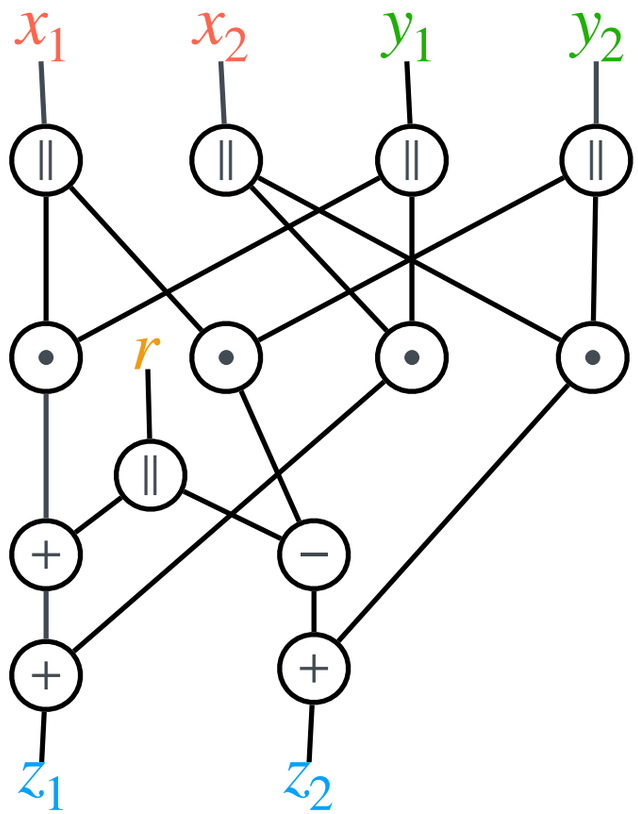
Independent from secrets?

23 wires → 23 variables to check

Security in the t-probing model: any set of t intermediate variables is independent from the secret

Proof in the Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



2-probing secure?

Independent from secrets?

23 wires $\rightarrow \binom{23}{2} = 253$ pairs of variables to check

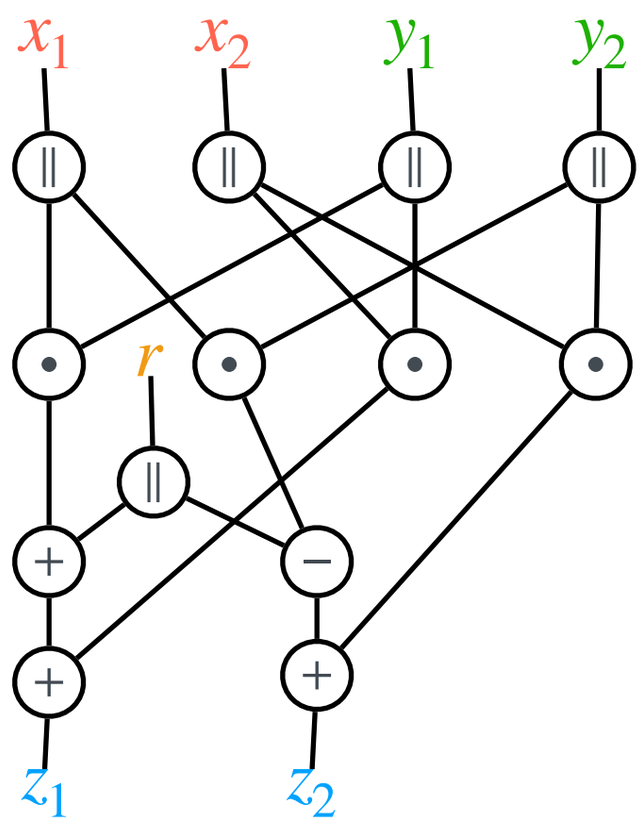
Security in the t-probing model: any set of t intermediate variables is independent from the secret

Security Proofs

- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
  - Probing Security
  - Random Probing Security

Proof in the Random Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

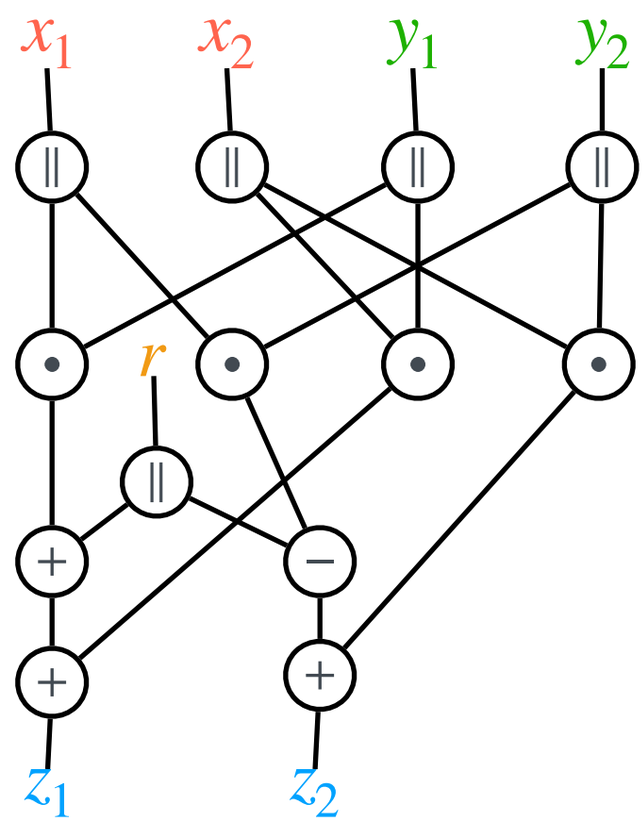


Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Proof in the Random Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

1. Identify all the sets of probes revealing x or y
2. Compute their probability to happen



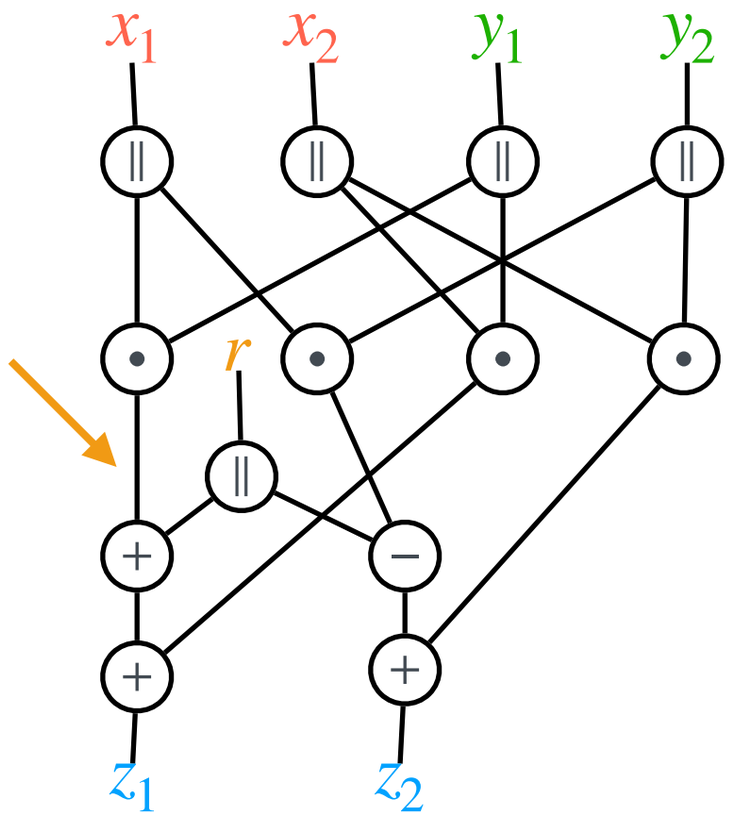
Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Proof in the Random Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

1. Identify all the sets of probes revealing x or y
2. Compute their probability to happen

We count how many probes of size 1 depends on the secrets $\Rightarrow c_1$



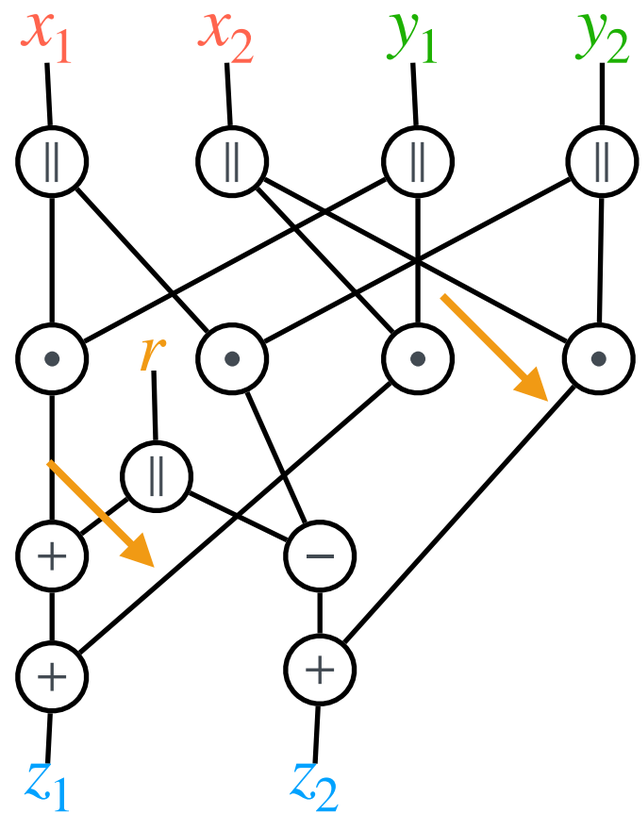
Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Proof in the Random Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$

1. Identify all the sets of probes revealing x or y
2. Compute their probability to happen

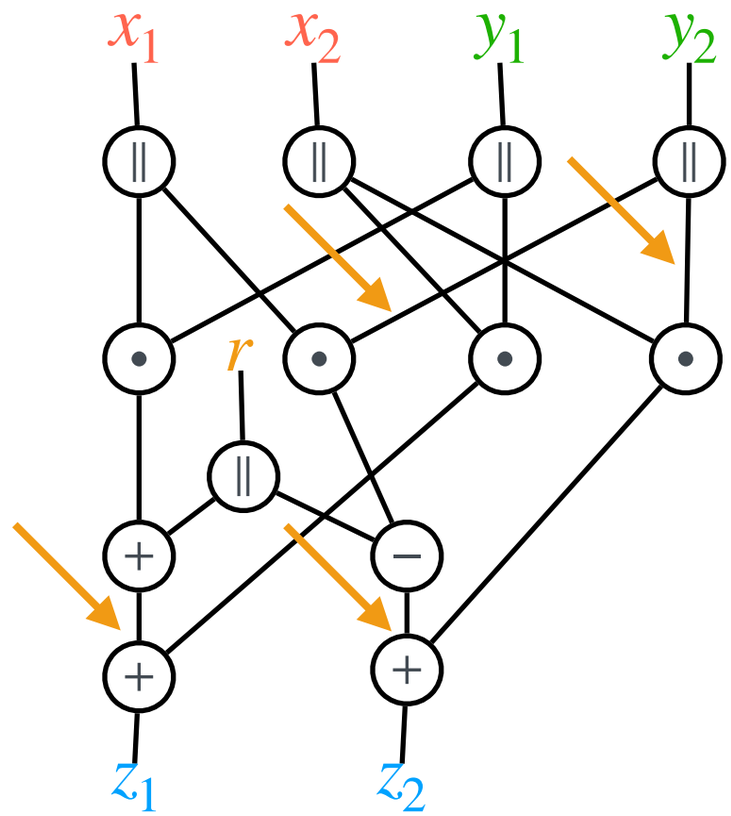
We count how many probes of size 2 depends on the secrets $\Rightarrow c_2$



Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

Proof in the Random Probing Model

$z_1 \leftarrow x_1 \cdot y_1 + r + x_2 \cdot y_1$
$z_2 \leftarrow x_1 \cdot y_2 - r + x_2 \cdot y_2$



1. Identify all the sets of probes revealing x or y
2. Compute their probability to happen

We count how many probes of size l depends on the secrets $\Rightarrow c_i$

$$\varepsilon = \sum_{i=1}^{s} c_i \cdot p^i \cdot (1-p)^{s-i}$$

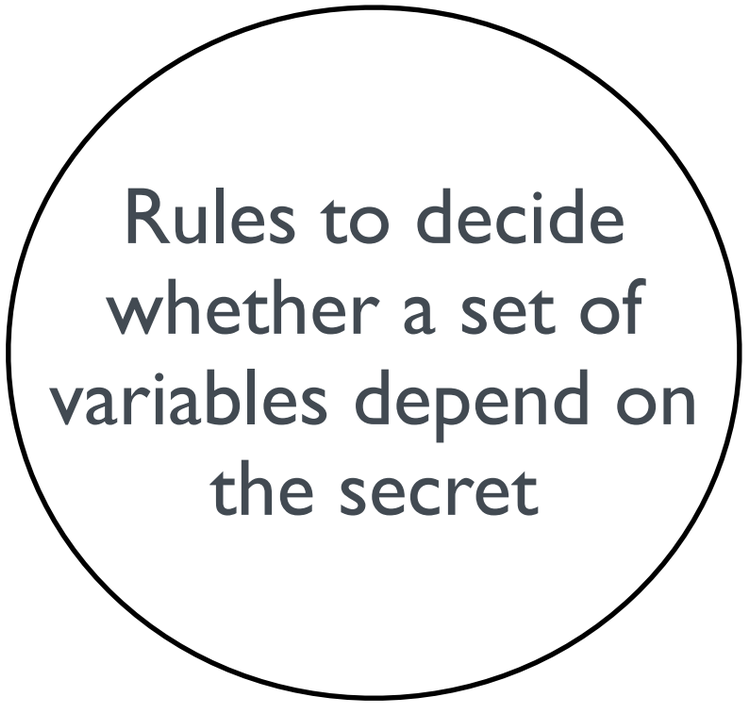
Security in the (p, ε)-random probing model: given p, the probability to recover information on the secret is bounded by ε.

With Formal Methods

Formally verify security in the probing model with a tool

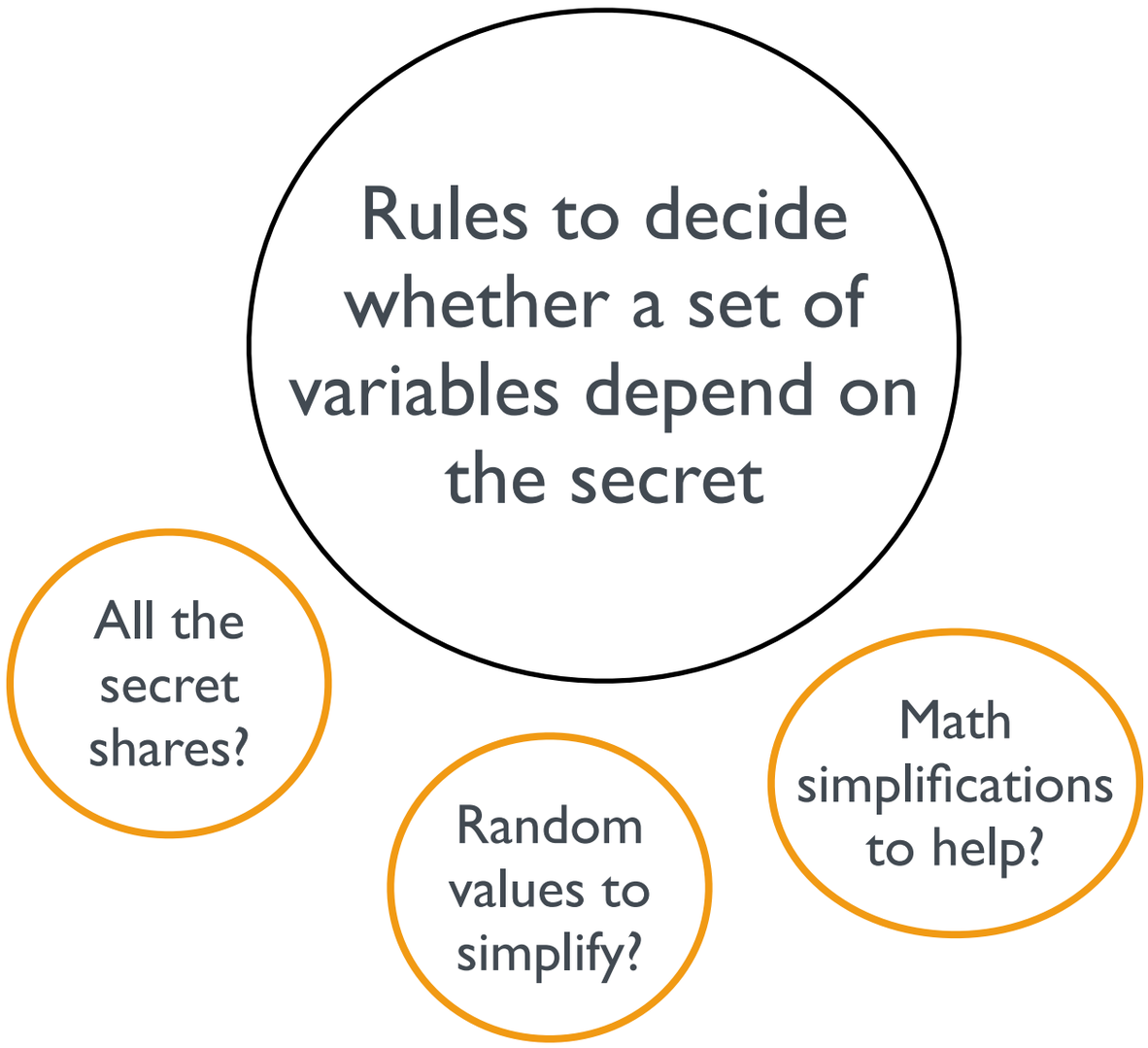
With Formal Methods

Formally verify security in the probing model with a tool



With Formal Methods

Formally verify security in the probing model with a tool



With Formal Methods

Formally verify security in the probing model with a tool

Rules to decide whether a set of variables depend on the secret

All the secret shares?

Random values to simplify?

Math simplifications to help?

Smart enumeration to go through all the sets of variables

Example of Automatic Tools

Verification tool

Example of Automatic Tools

Security property

function example$(a_0, a_1, a_2, b_0, b_1, b_2)$

$r_{00}, r_{01}, r_{02}, r_{12} \leftarrow \$$
$t \leftarrow a_0 \cdot b_0$
$c_0 \leftarrow t \oplus r_{00}$
$t \leftarrow a_0 \cdot b_1$
$t \leftarrow t \oplus r_{01}$
$c_0 \leftarrow c_0 \oplus t$
$t \leftarrow a_0 \cdot b_2$
$t \leftarrow t \oplus r_{02}$
$c_0 \leftarrow c_0 \oplus t$
$t \leftarrow a_1 \cdot b_0$
$c_1 \leftarrow t \oplus r_{01}$
$t \leftarrow a_1 \cdot b_1$
$c_1 \leftarrow c_1 \oplus t$
$\ldots$
return (c_0, c_1, c_2)

Verification tool

Example of Automatic Tools

Security property

function example$(a_0, a_1, a_2, b_0, b_1, b_2)$

$r_{00}, r_{01}, r_{02}, r_{12} \leftarrow \$$
$t \leftarrow a_0 \cdot b_0$
$c_0 \leftarrow t \oplus r_{00}$
$t \leftarrow a_0 \cdot b_1$
$t \leftarrow t \oplus r_{01}$
$c_0 \leftarrow c_0 \oplus t$
$t \leftarrow a_0 \cdot b_2$
$t \leftarrow t \oplus r_{02}$
$c_0 \leftarrow c_0 \oplus t$
$t \leftarrow a_1 \cdot b_0$
$c_1 \leftarrow t \oplus r_{01}$
$t \leftarrow a_1 \cdot b_1$
$c_1 \leftarrow c_1 \oplus t$
$\ldots$
return (c_0, c_1, c_2)

→ **Verification tool** →

Security proof or potential attacks

Security Proofs

- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
  - Probing Security
  - Random Probing Security

Proof in the Probing Model

- Example:
  - Addition between two secrets a and b with the shares $(a_i)_{1 \leq i \leq n}$ and $(b_i)_{1 \leq i \leq n}$

$$\text{for } i = 1 \text{ to } n$$
$$c_i \leftarrow a_i + b_i$$

Proof in the Probing Model

- Example:
  - Addition between two secrets a and b with the shares $(a_i)_{1 \le i \le n}$ and $(b_i)_{1 \le i \le n}$

$$\text{for } i = 1 \text{ to } n$$
$$c_i \leftarrow a_i + b_i$$

- Possible probes
  - $a_1, a_2, \cdots, a_n$
  - $b_1, b_2, \ldots, b_n$
  - $c_1, c_2, \ldots, c_n$

Proof in the Probing Model

- Example:
  - Addition between two secrets a and b with the shares $(a_i)_{1 \le i \le n}$ and $(b_i)_{1 \le i \le n}$

$$\text{for } i = 1 \text{ to } n$$
$$c_i \leftarrow a_i + b_i$$

- Possible probes
  - $a_1, a_2, \cdots, a_n$
  - $b_1, b_2, \ldots, b_n$
  - $c_1, c_2, \ldots, c_n$

With at most $n - 1$ probes
$\Rightarrow$ impossible to recover a or b

Proof in the Probing Model

- Example:
  - Extract of a multiplication between two secrets a and b with the shares $(a_i)_{1 \le i \le n}$ and $(b_i)_{1 \le i \le n}$

for $i = 1$ to n
  for $j = i + 1$ to n
    $r_{i,j} \leftarrow \$$
    $r_{j,i} \leftarrow (r_{i,j} \oplus a_i \cdot b_j) \oplus a_j \cdot b_i$

Proof in the Probing Model

- Example:
  - Extract of a multiplication between two secrets a and b with the shares $(a_i)_{1 \le i \le n}$ and $(b_i)_{1 \le i \le n}$

for $i = 1$ to n
  for $j = i + 1$ to n
    $r_{i,j} \leftarrow \$$
    $r_{j,i} \leftarrow (r_{i,j} \oplus a_i \cdot b_j) \oplus a_j \cdot b_i$

- Possible probes
  - $r_{i,j}\ (i < j), r_{j,i}$
  - $a_1, a_2, \cdots, a_n$
  - $b_1, b_2, \ldots, b_n$
  - $r_{j,i}\ (i < j)$
  - $a_i \cdot b_j$
  - $r_{i,j} \oplus a_i \cdot b_j\ (i < j)$

Proof in the Probing Model

- Example:
  - Extract of a multiplication between two secrets a and b with the shares $(a_i)_{1 \le i \le n}$ and $(b_i)_{1 \le i \le n}$

for $i = 1$ to n
  for $j = i + 1$ to n
    $r_{i,j} \leftarrow \$$
    $r_{j,i} \leftarrow (r_{i,j} \oplus a_i \cdot b_j) \oplus a_j \cdot b_i$

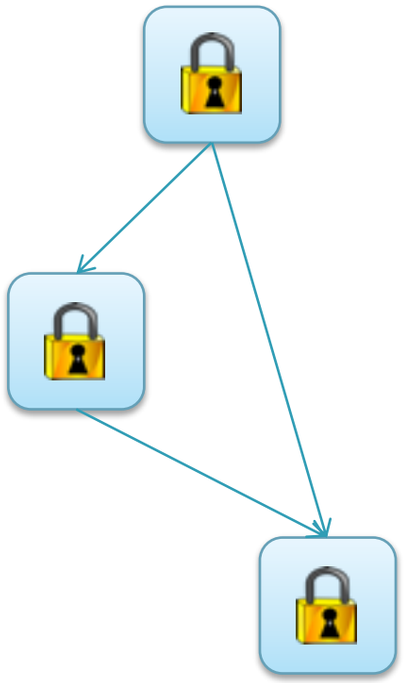
- Possible probes
  - $r_{i,j}\ (i < j)$
  - $a_1, a_2, \cdots, a_n$
  - $b_1, b_2, \ldots, b_n$
  - $r_{j,i}\ (i < j)$
  - $a_i \cdot b_j$
  - $r_{i,j} \oplus a_i \cdot b_j\ (i < j)$

2 shares in a single probe

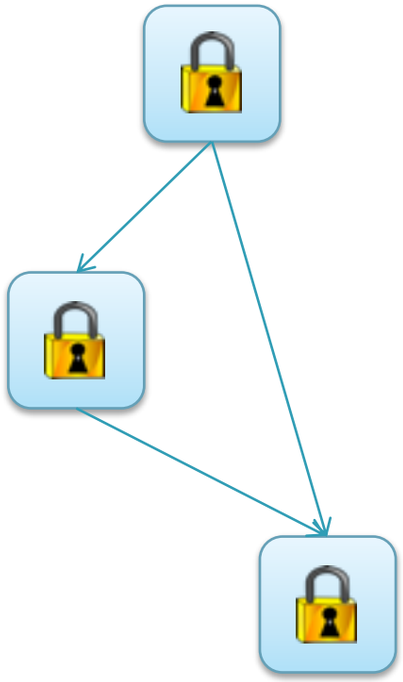
Security Proofs

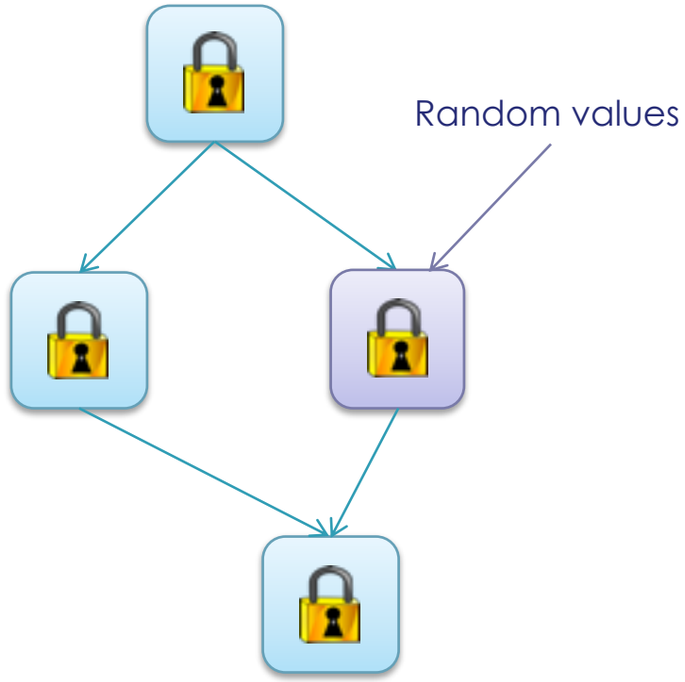
- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
  - Probing Security
  - Random Probing Security

Composition of gadgets



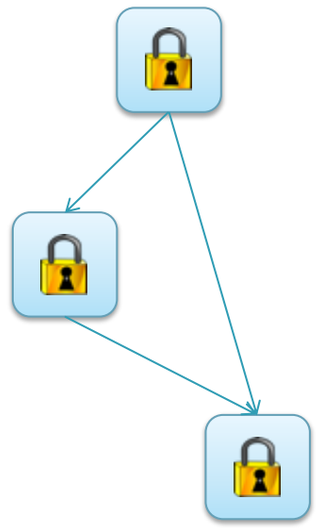
Composition of gadgets





Composition of gadgets

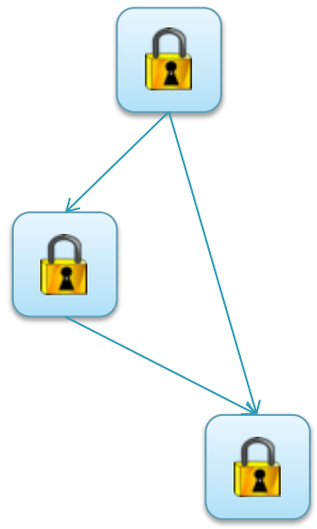
- Reminder: an implementation is *t*-probing secure iff any set of at most *t* variables is independent from the secret
- How to reason on composition?



Composition of gadgets

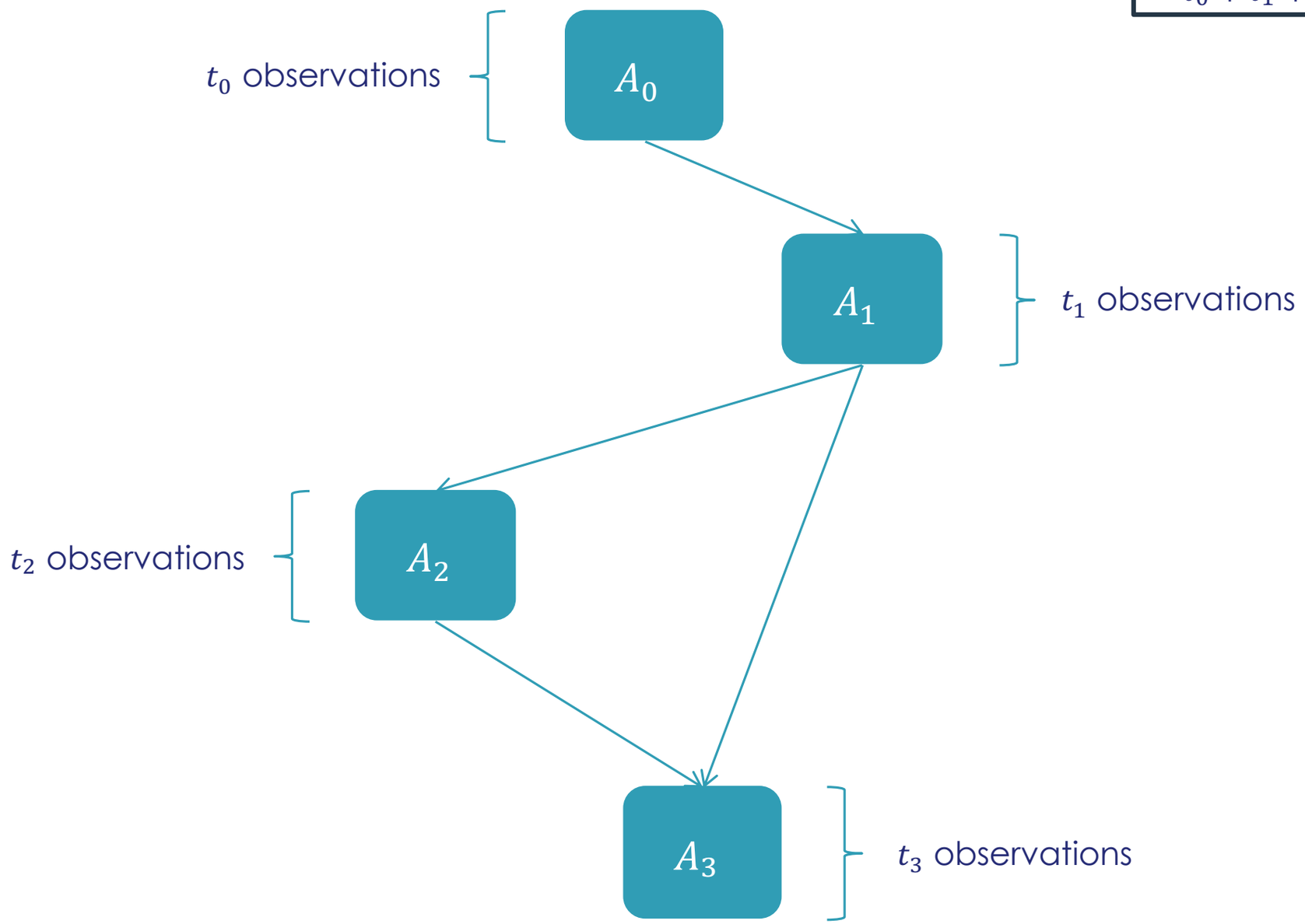
- Reminder: an implementation is *t*-probing secure iff any set of at most *t* variables is independent from the secret
- How to reason on composition?
  - Stronger property: *t*-non-interference

any set of *t* variables can be simulated with at most *t* input shares



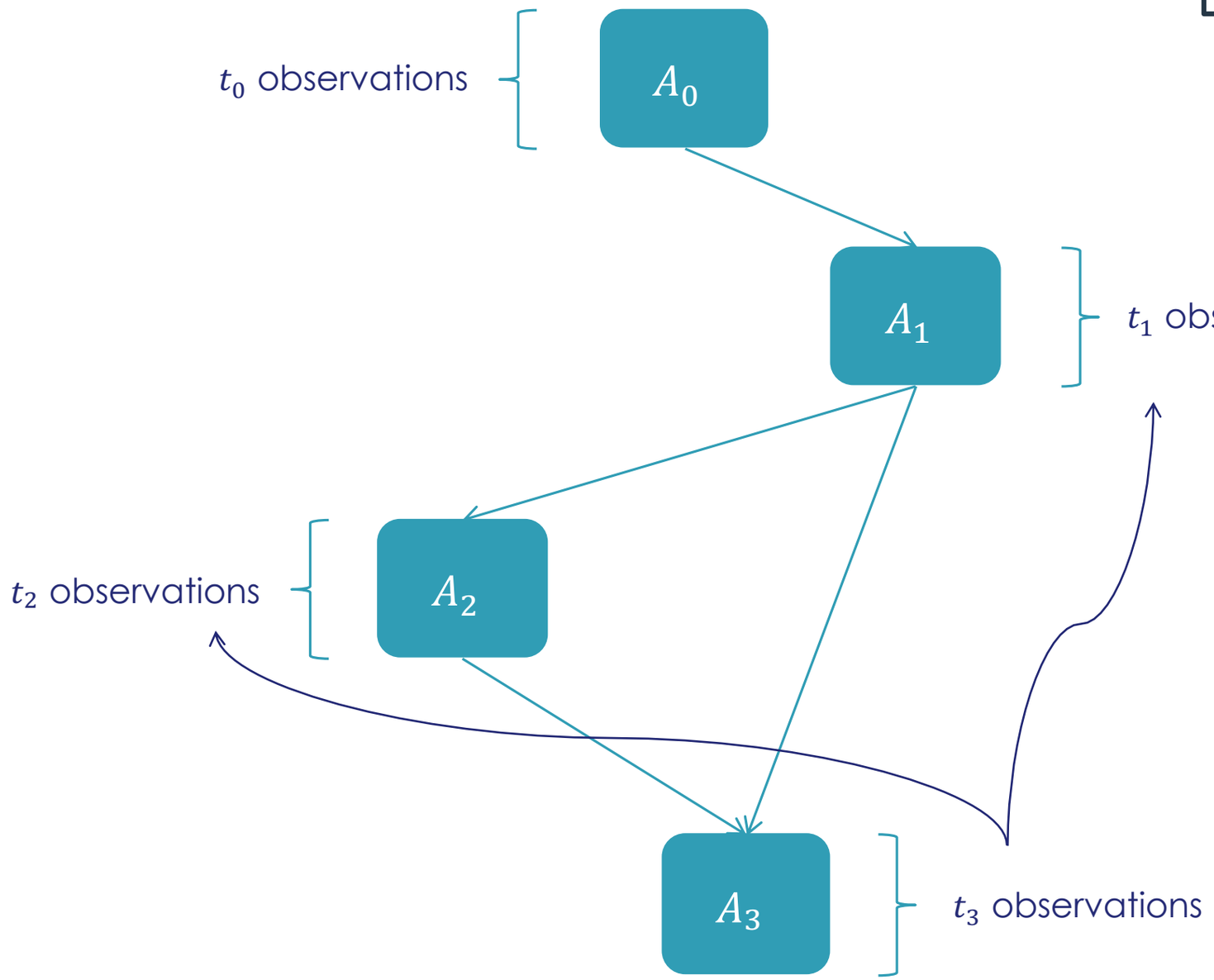
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 \leq t$$



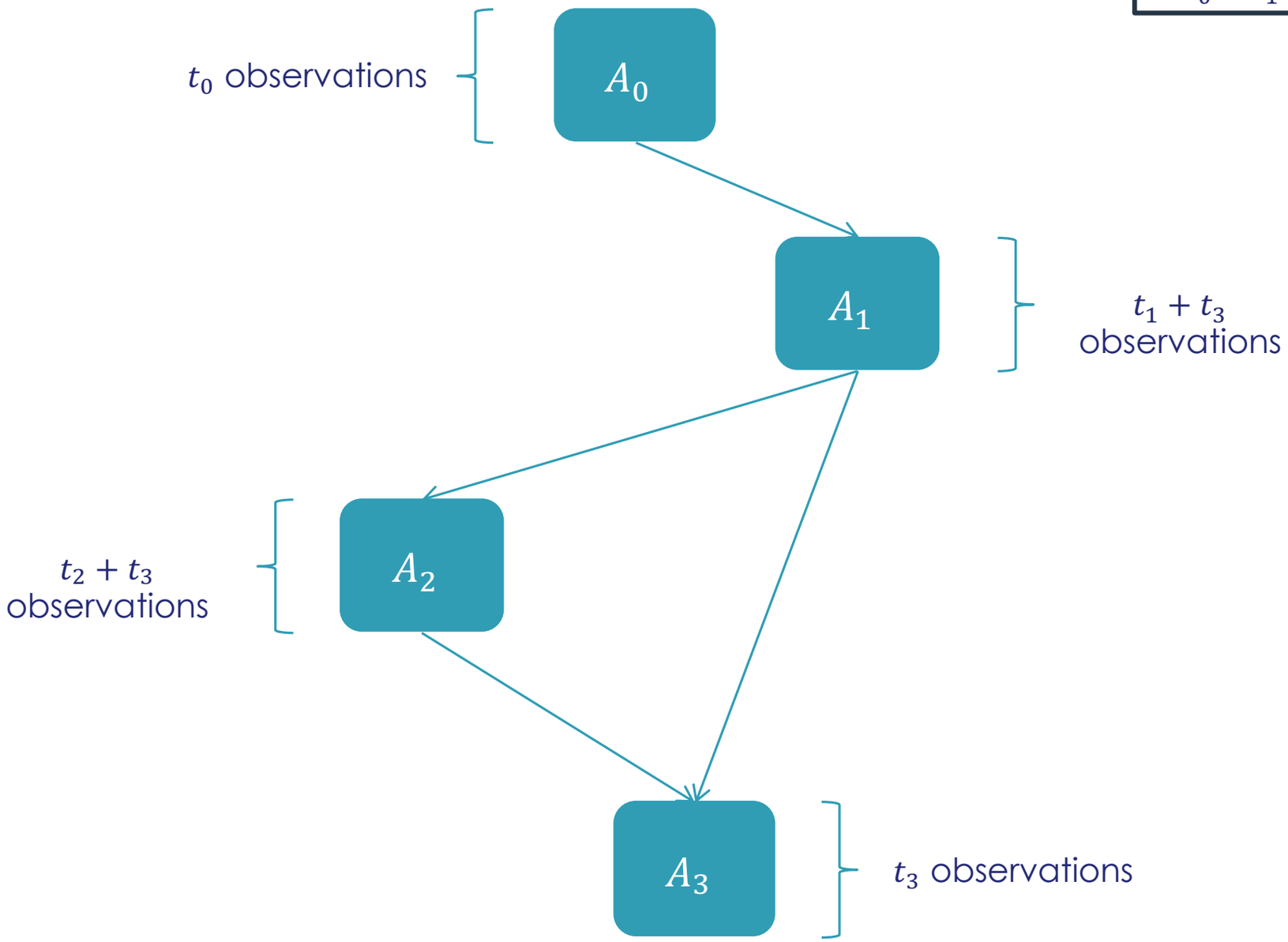
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 \leq t$$



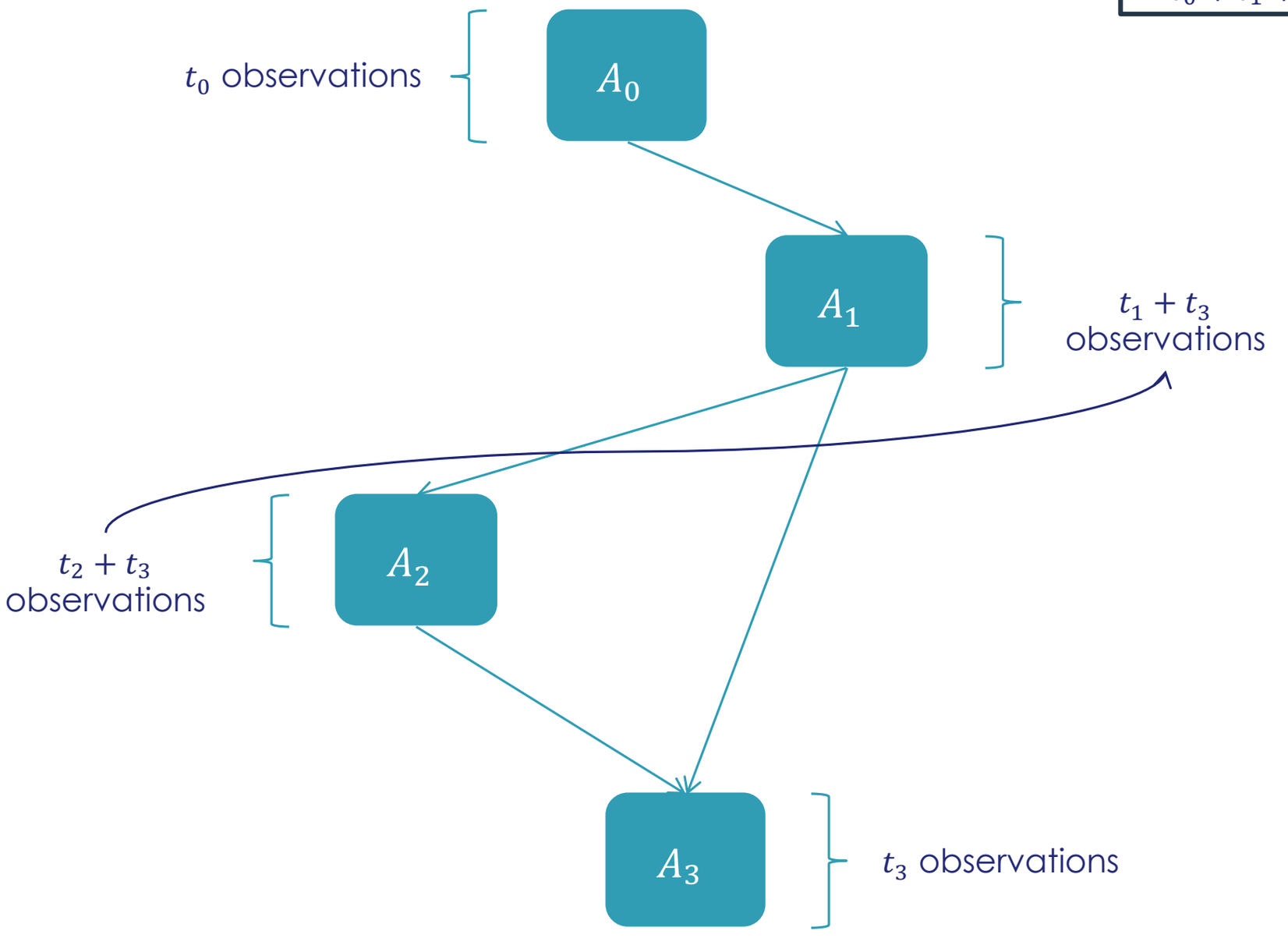
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 \leq t$$



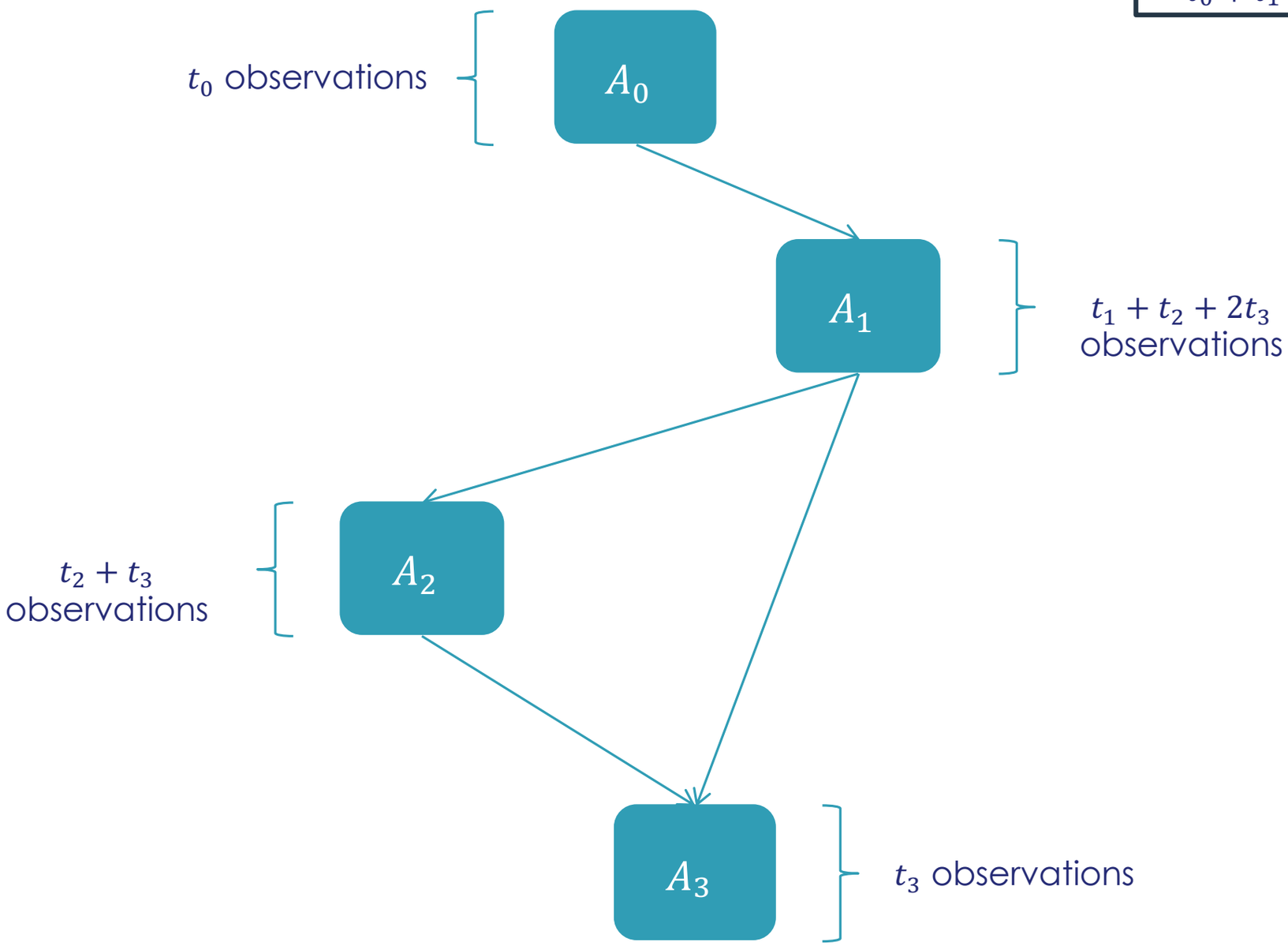
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 \leq t$$



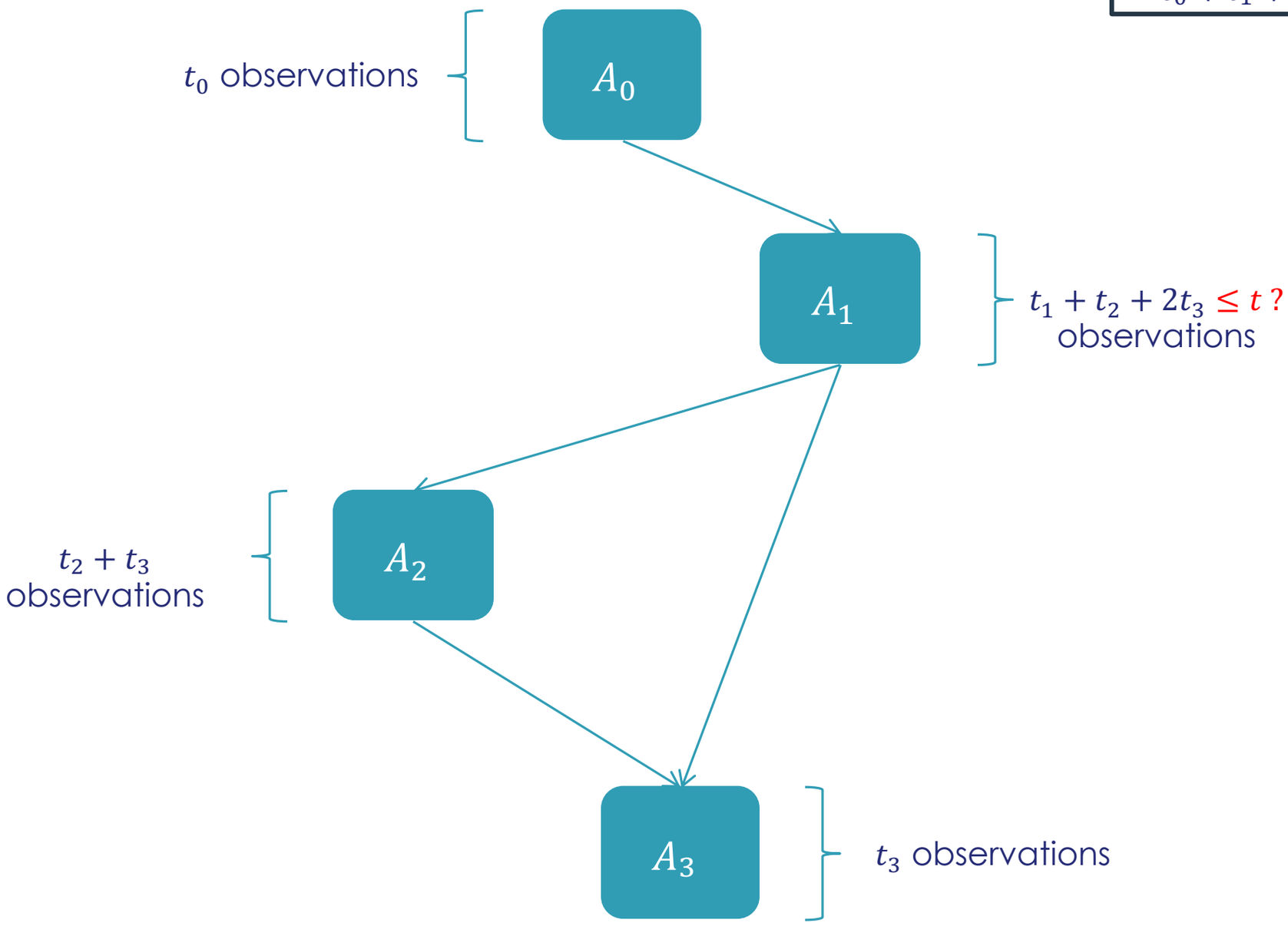
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 \leq t$$



Composition of gadgets

$t_0 + t_1 + t_2 + t_3 \leq t$



Composition of gadgets

- Reminder: an implementation is *t*-probing secure iff any set of at most *t* variables is independent from the secret
- How to reason on composition?
  - Stronger property: *t*-non-interference

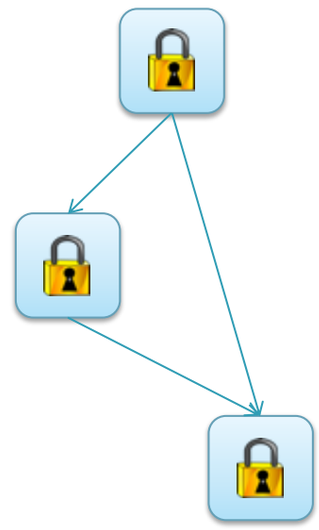
any set of t variables can be simulated with at most t input shares

  - **Stronger** property: **strong** non-interference

any set of

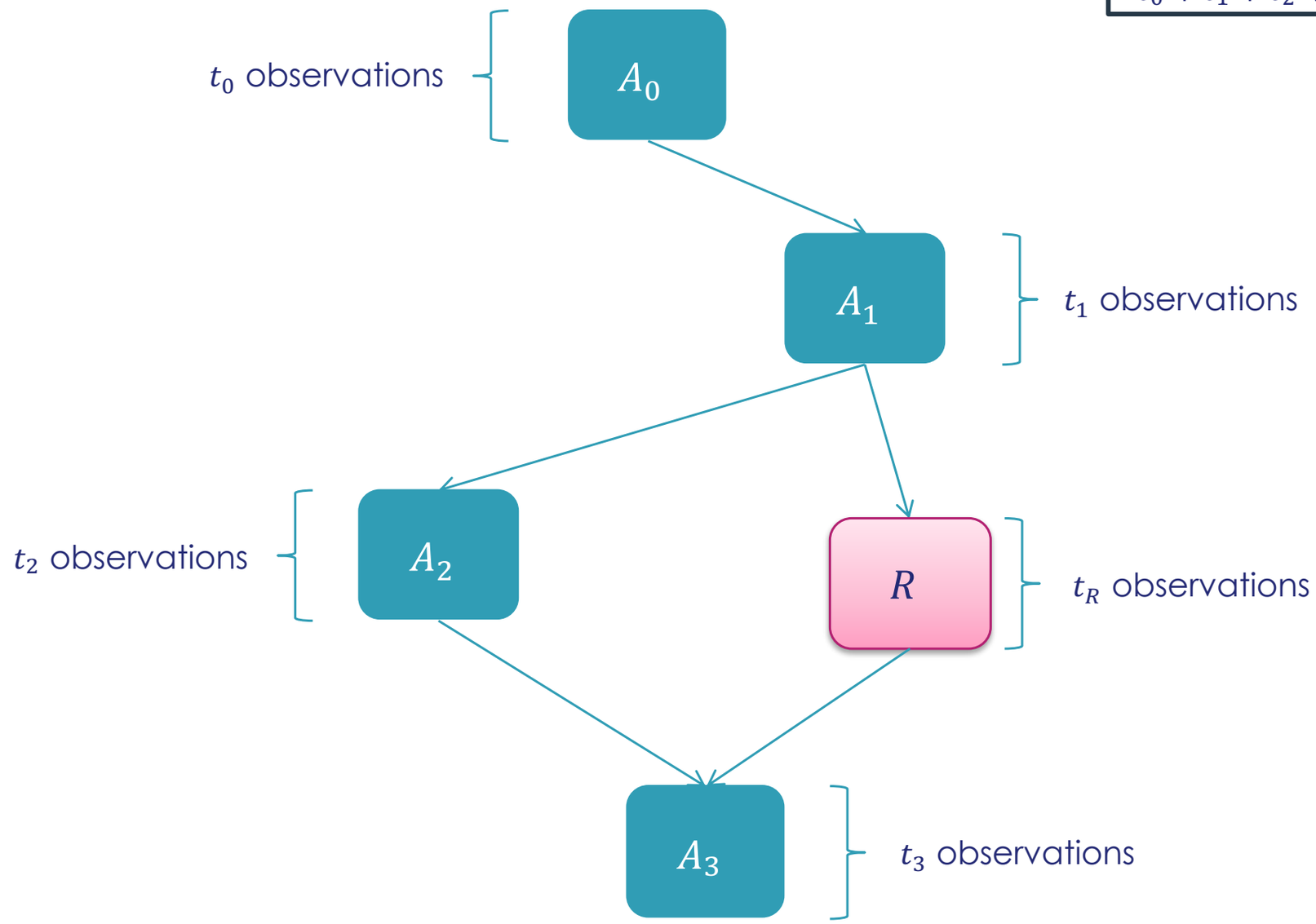
- t_1 internal variables
- t_2 output variables

can be simulated with at most t_1 input shares



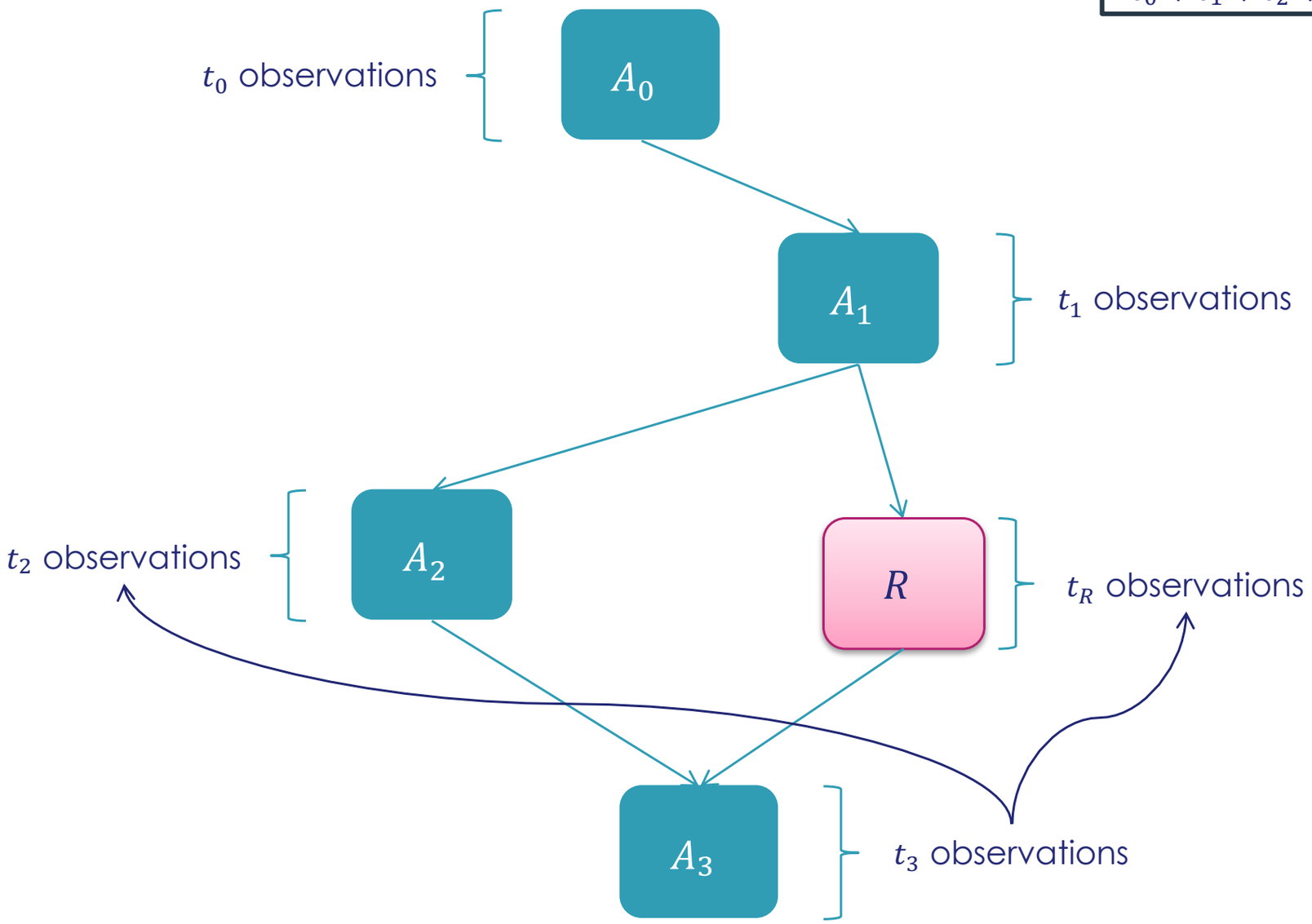
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 + t_R \leq t$$



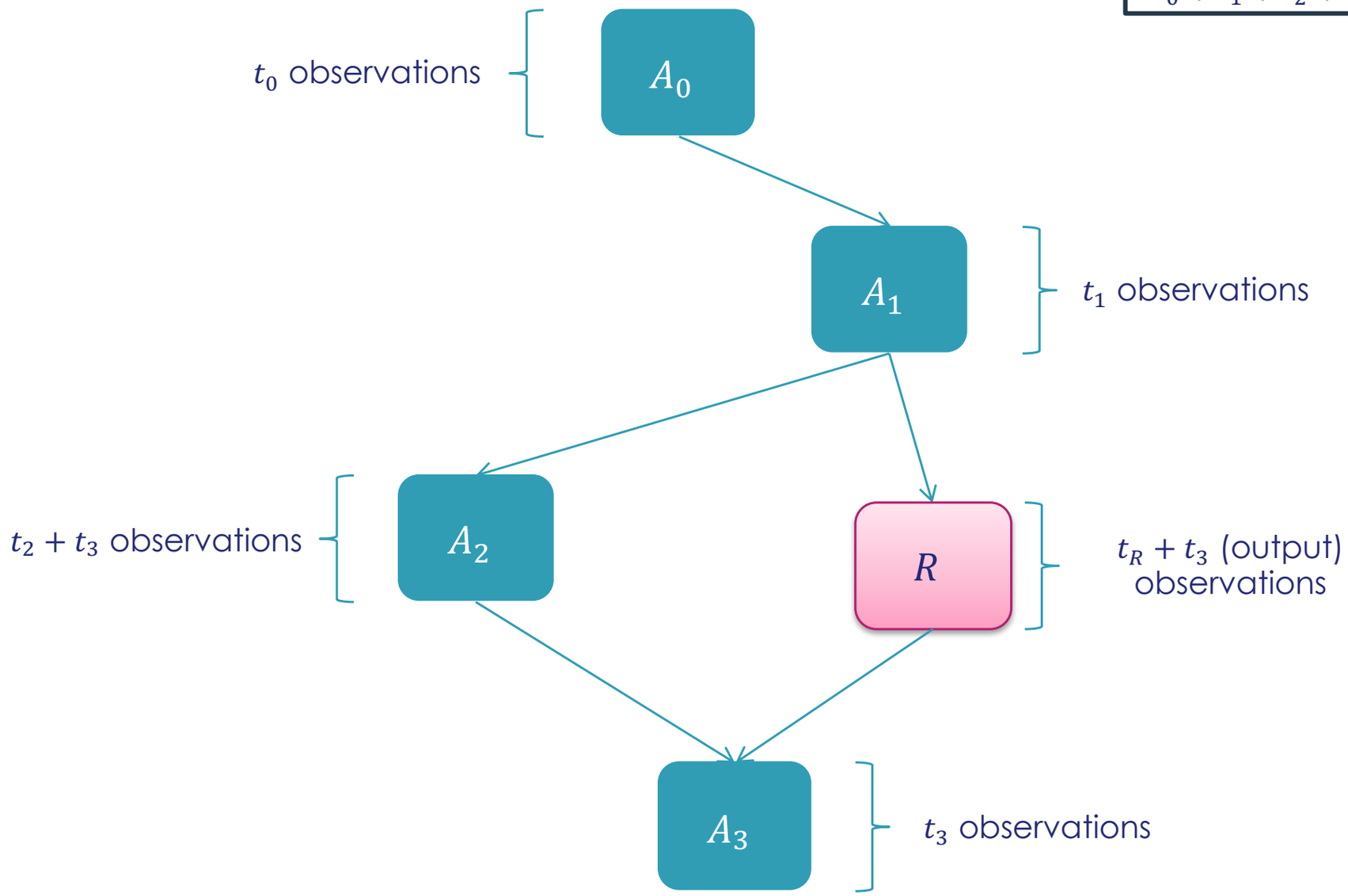
Composition of gadgets

$$t_0 + t_1 + t_2 + t_3 + t_R \leq t$$



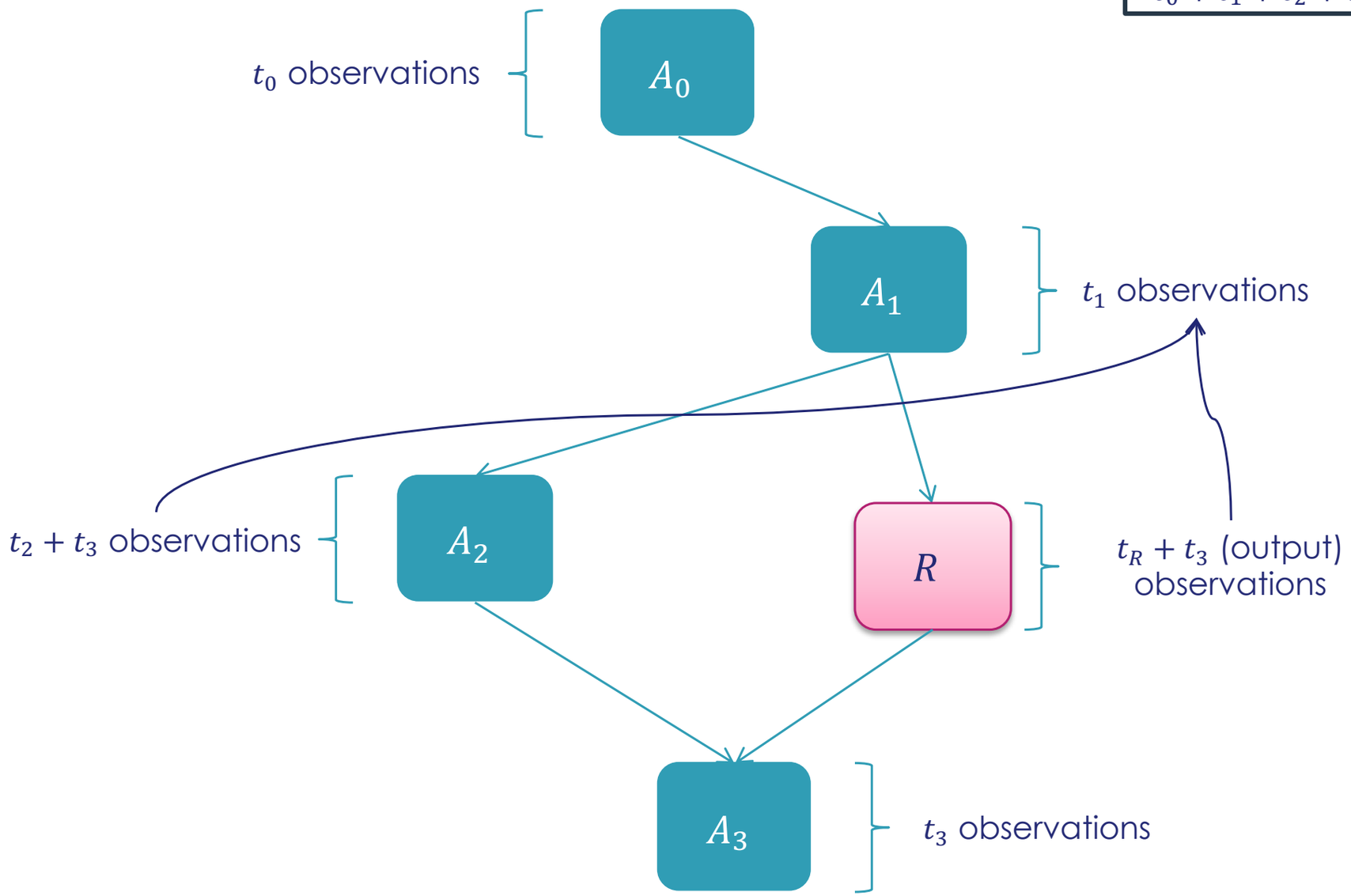
Composition of gadgets

$t_0 + t_1 + t_2 + t_3 + t_R \leq t$



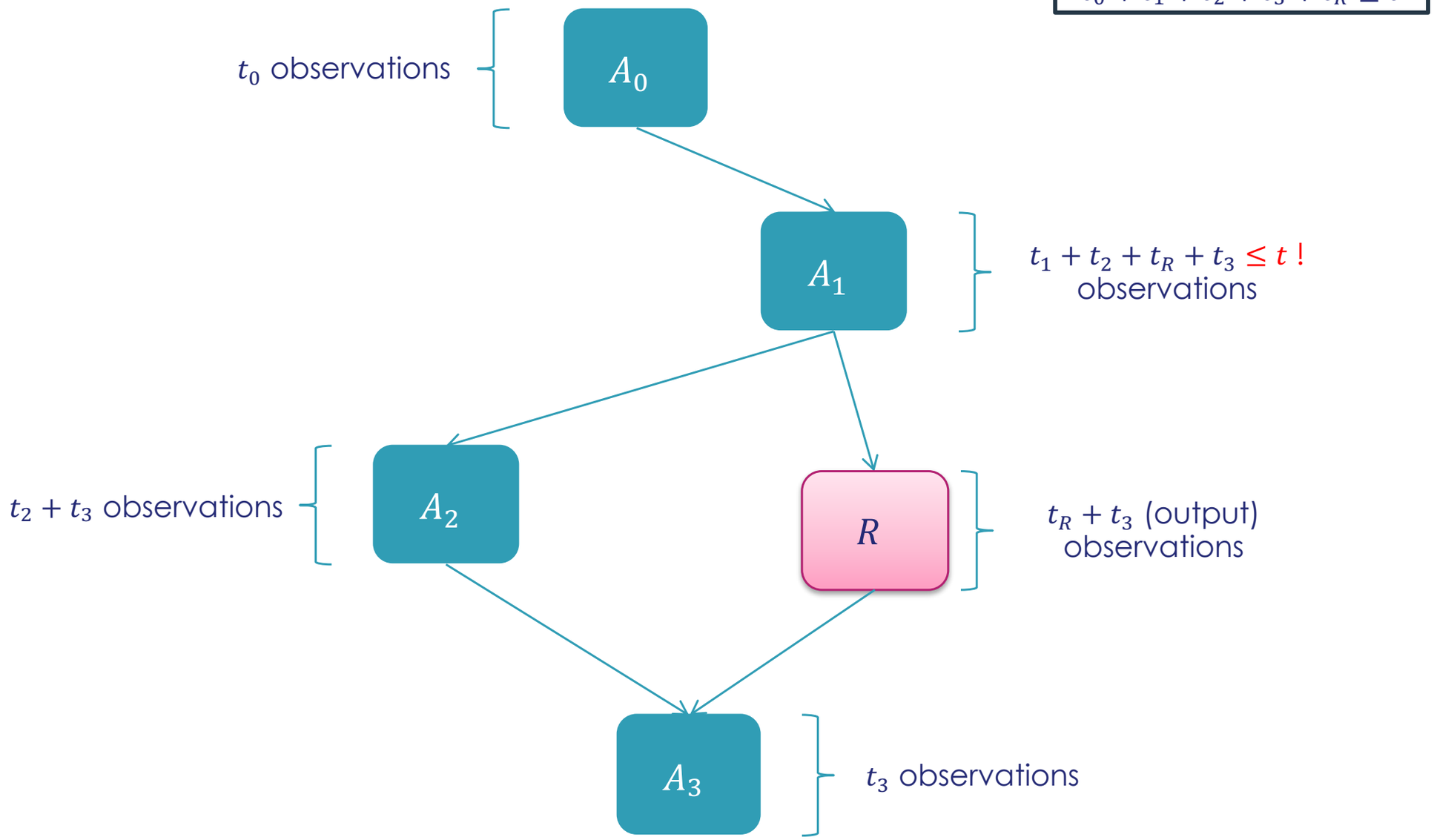
Composition of gadgets

$t_0 + t_1 + t_2 + t_3 + t_R \leq t$



Composition of gadgets

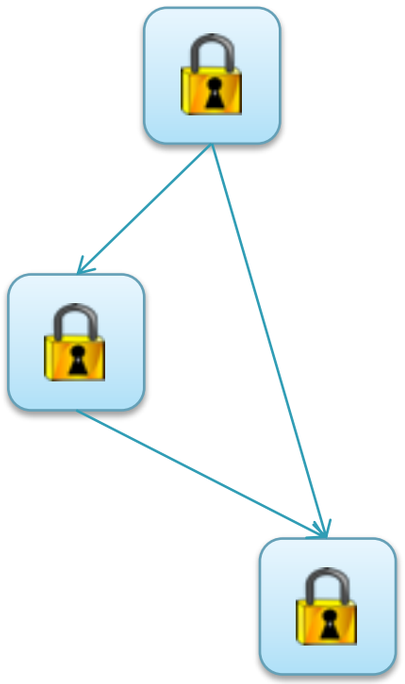
$t_0 + t_1 + t_2 + t_3 + t_R \leq t$



Security Proofs

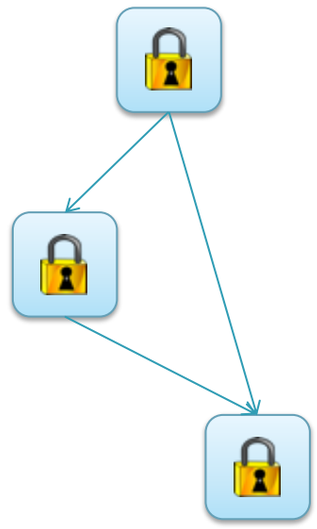
- Small gadgets (small circuit and small masking order)
  - Check the security by hand or using automatic tools
    - Probing Security
    - Random Probing Security
- Bigger gadgets (bigger circuits and/or higher masking order)
  - Build theoretical proofs
    - Probing Security
- Composition of gadgets
  - Probing Security
  - Random Probing Security

Composition of gadgets



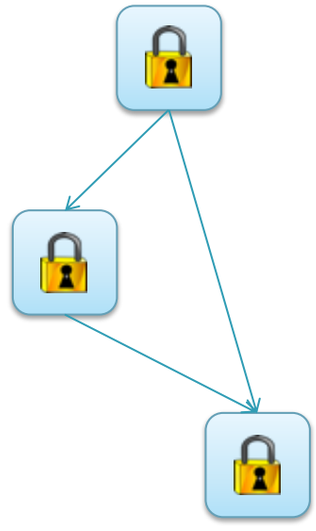
Composition of gadgets

- Reminder: an implementation is (p, ε)-random probing secure iff the probability to recover information on the secret is bounded by ε.
- How to reason on composition?

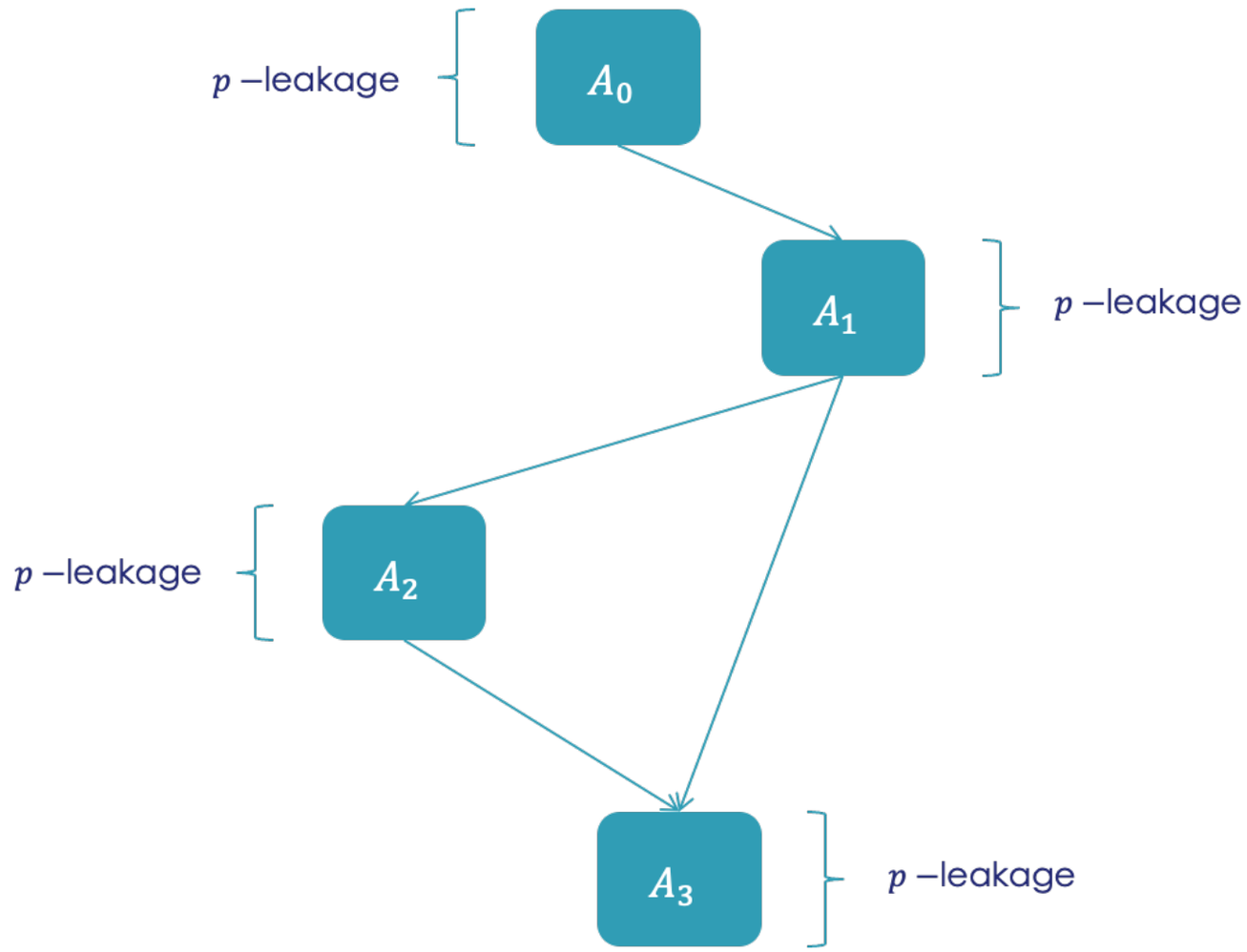


Composition of gadgets

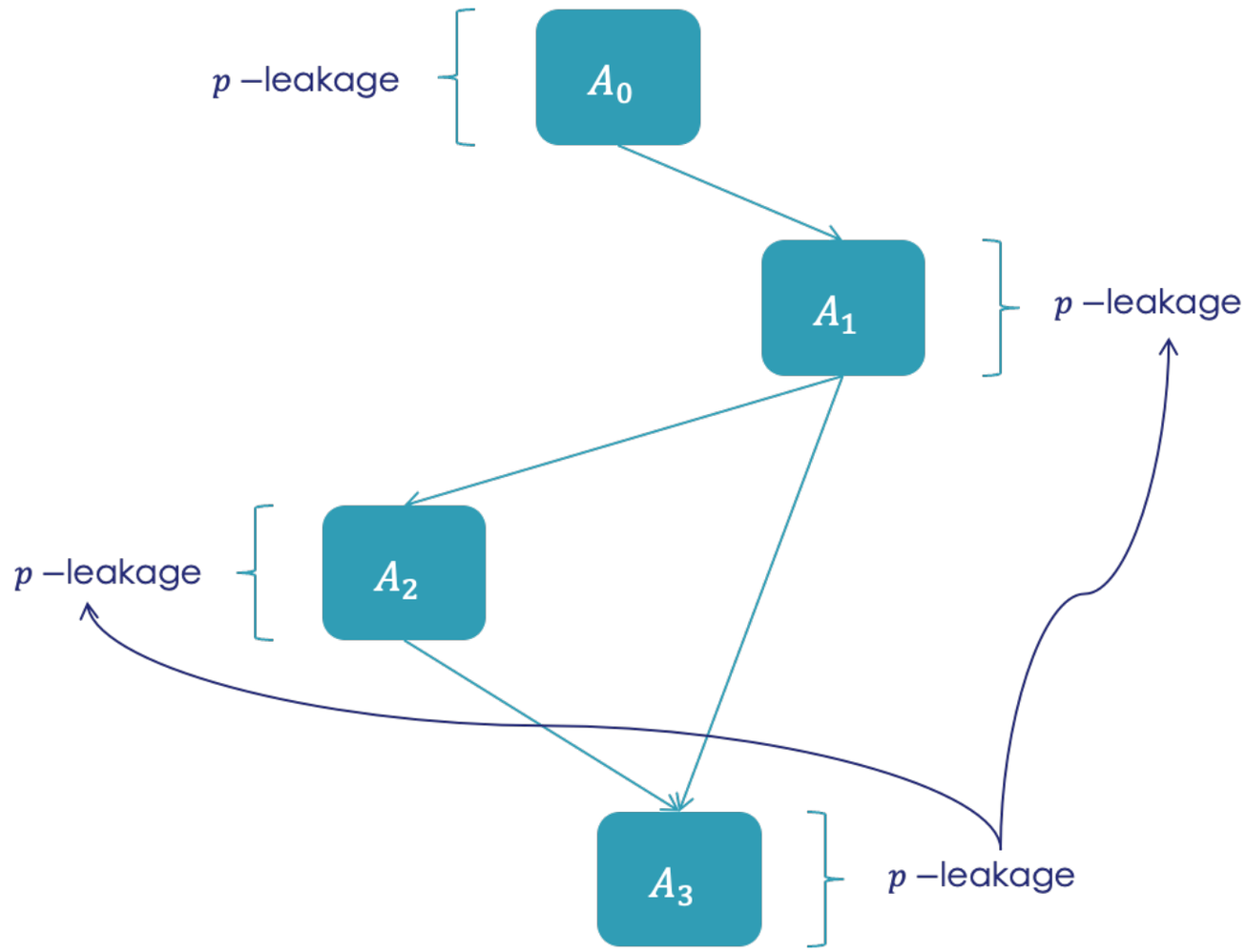
- Reminder: an implementation is (p, ε)-random probing secure iff the probability to recover information on the secret is bounded by ε.
- How to reason on composition?
  - Stronger property: (t, p, ε)-RPC
    - Any t output shares + the leakage can be simulated with at most t input shares with probability $\geq 1 - \varepsilon$



Composition of gadgets

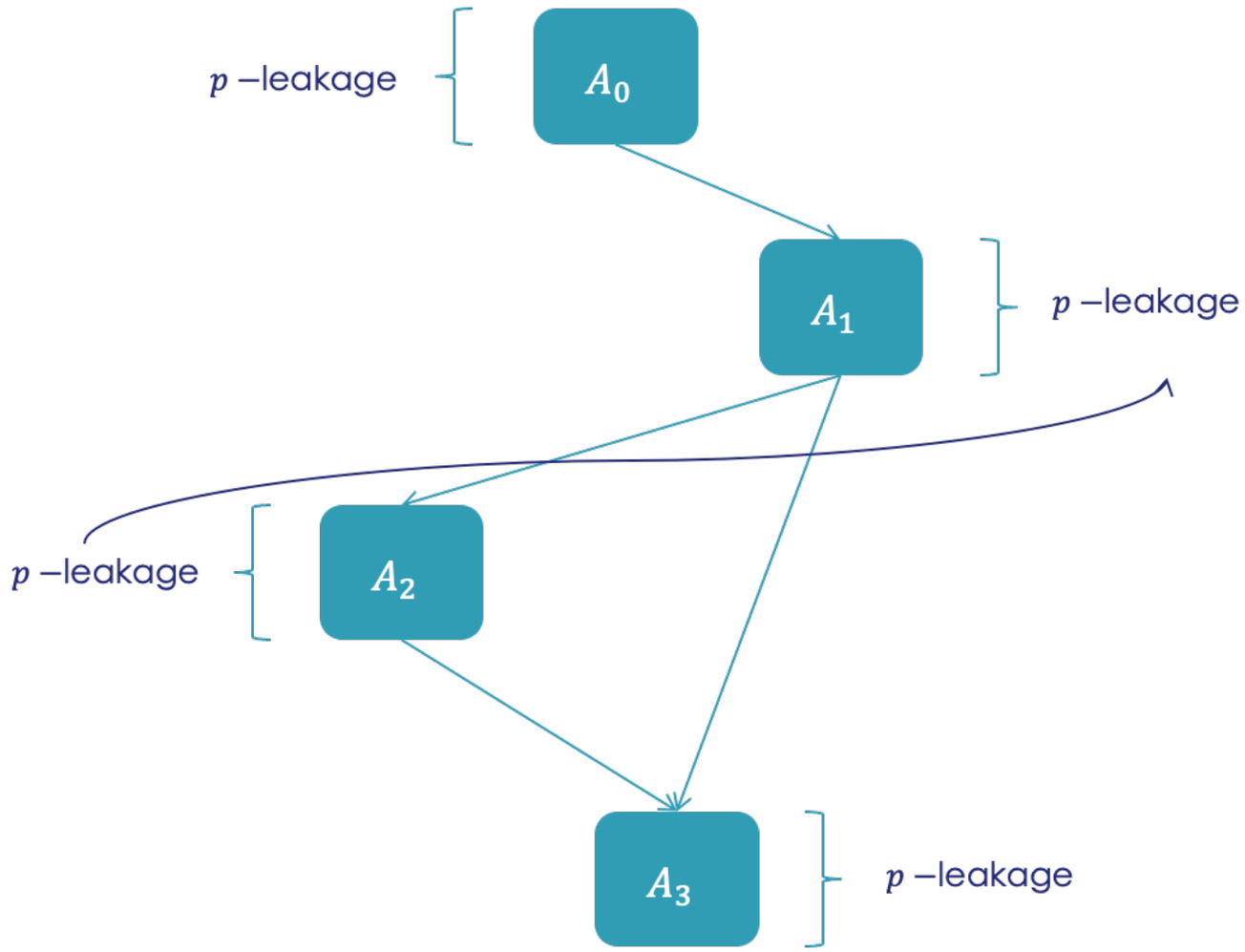


Composition of gadgets



A_3 is (t, p, ε_3)-RPC
→ its leakage can be simulated with t input shares with probability $1 - \varepsilon_3$

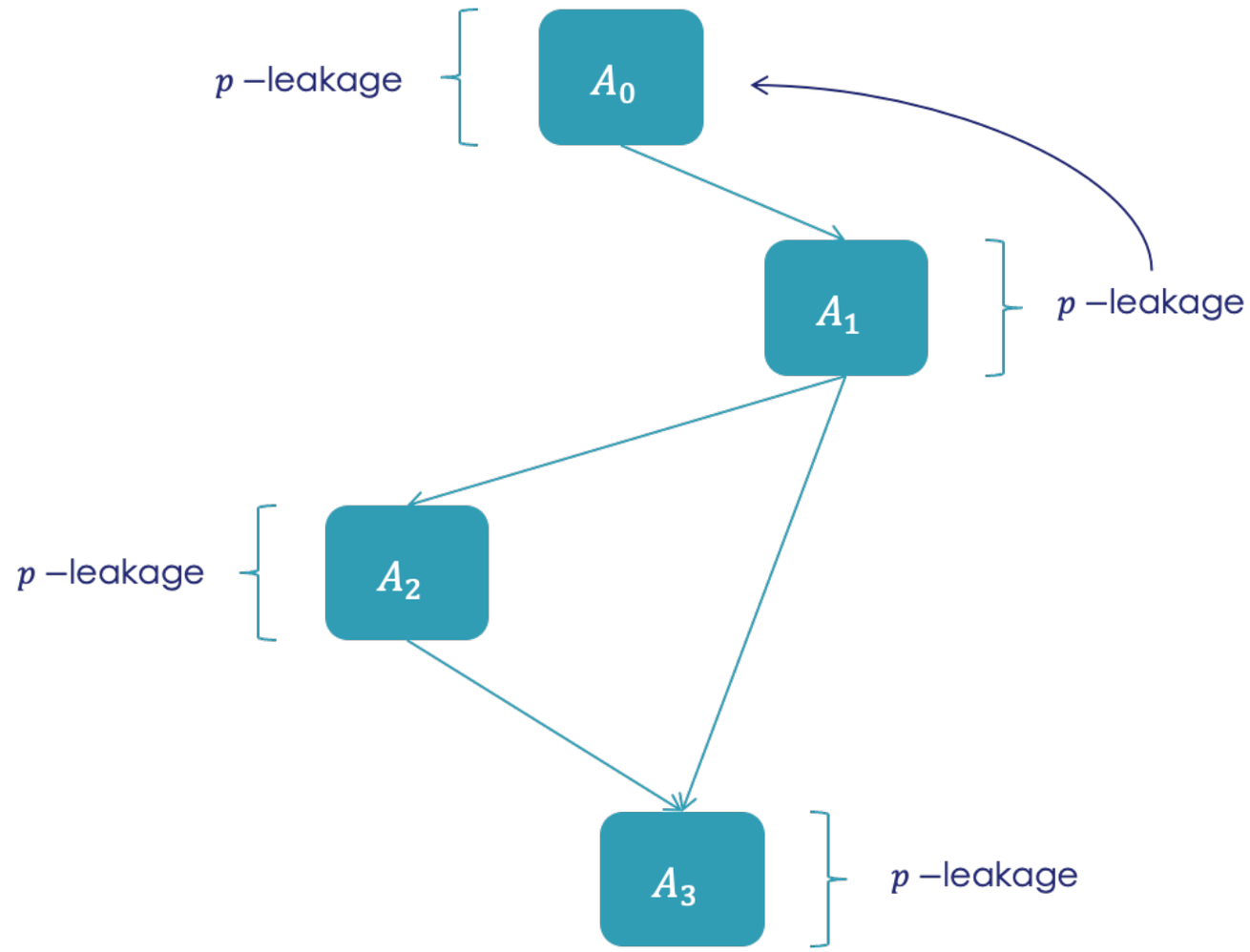
Composition of gadgets



A_2 is (t, p, ε_2)-RPC

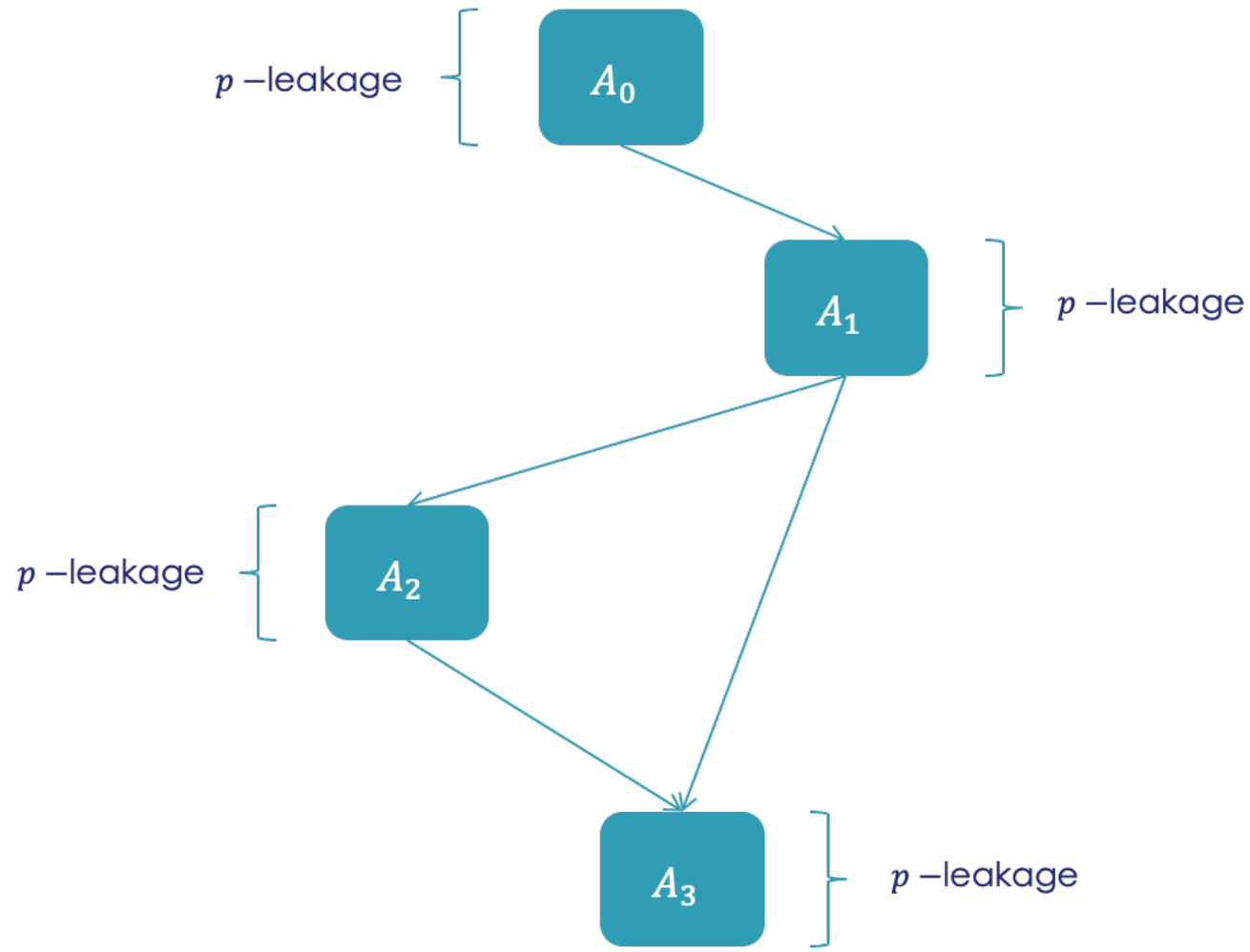
→ its leakage and A_3's inputs can be simulated with t input shares with probability $1 - \varepsilon_2$

Composition of gadgets



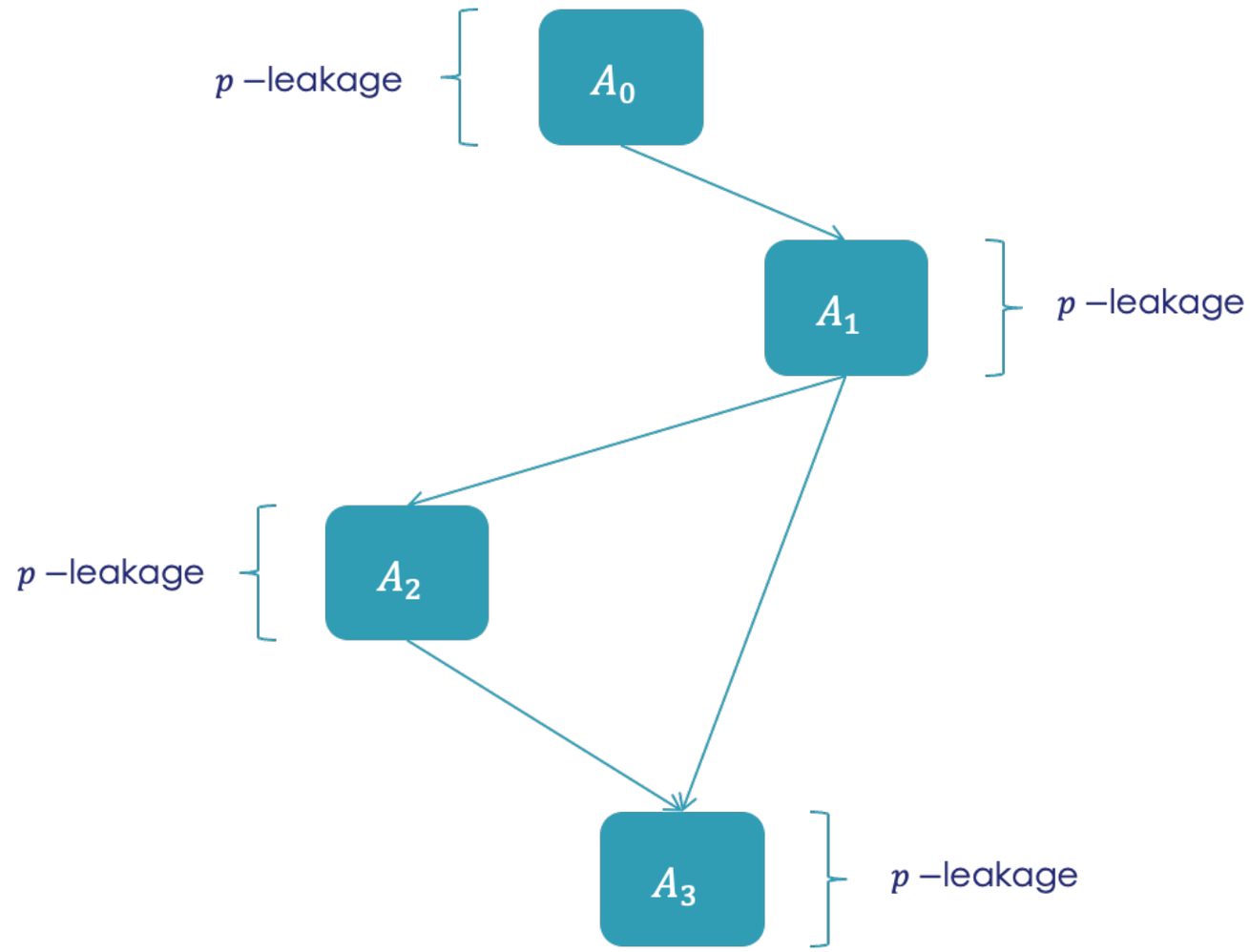
A_1 is (t, p, ε_1)-RPC
→ its leakage and A_2's and A_3's inputs can be simulated with t input shares with probability $1 - \varepsilon_1$

Composition of gadgets



A_0 is (t, p, ε_0)-RPC
→ its leakage and A_1's inputs can be simulated with t input shares with probability $1 - \varepsilon_0$

Composition of gadgets



Probability of failure: A_0 fails or A_1 fails or A_2 fails or A_3 fails $\leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon_4$

Conclusion

Summary

- Side-channel attacks are very powerful
  - Few seconds to recover the key on some software devices
  - Cheap equipments

Summary

- Side-channel attacks are very powerful
  - Few seconds to recover the key on some software devices
  - Cheap equipments

- Countermeasures are mandatory for sensitive devices
  - Hardware and low cost countermeasures
  - Fresh re-keying
  - Masking

Summary

- Side-channel attacks are very powerful
  - Few seconds to recover the key on some software devices
  - Cheap equipments
- Countermeasures are mandatory for sensitive devices
  - Hardware and low cost countermeasures
  - Fresh re-keying
  - Masking
- Practical security
  - Security proofs in relevant leakage models
  - Automatic tools

Challenges

- Efficiency
  - The least possible randomness
  - The least possible operations

Challenges

- Efficiency
  - The least possible randomness
  - The least possible operations
- Security
  - Theoretical proofs of existing schemes
  - Automatic tools to verify the security of implementations

Challenges

- Efficiency
  - The least possible randomness
  - The least possible operations
- Security
  - Theoretical proofs of existing schemes
  - Automatic tools to verify the security of implementations
- Practicality
  - Security of implementations under leakage models as close as possible to the reality

ERC Project AMAskZONE

European Research Council

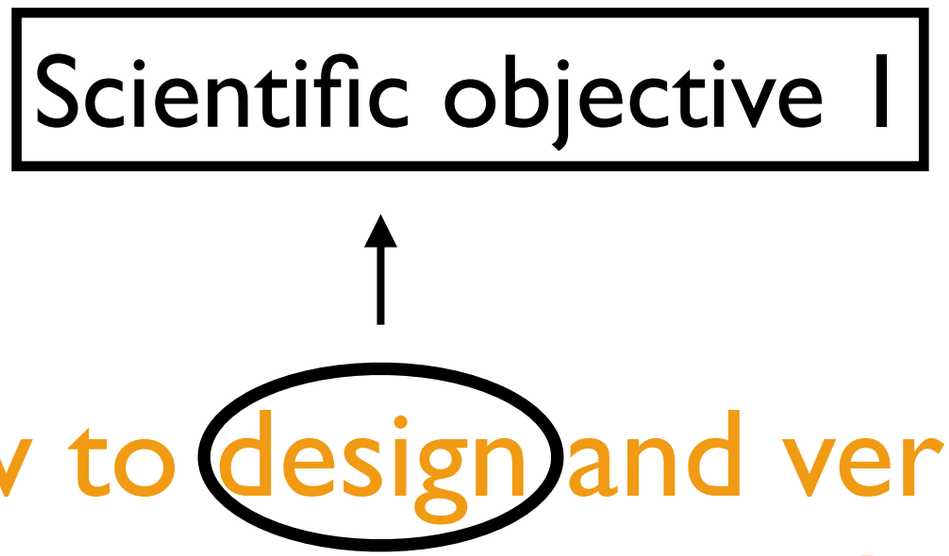
Established by the European Commission

AMAskZONE

How to design and verify cryptographic implementations so that they achieve measurable practical security?

CryptoExperts

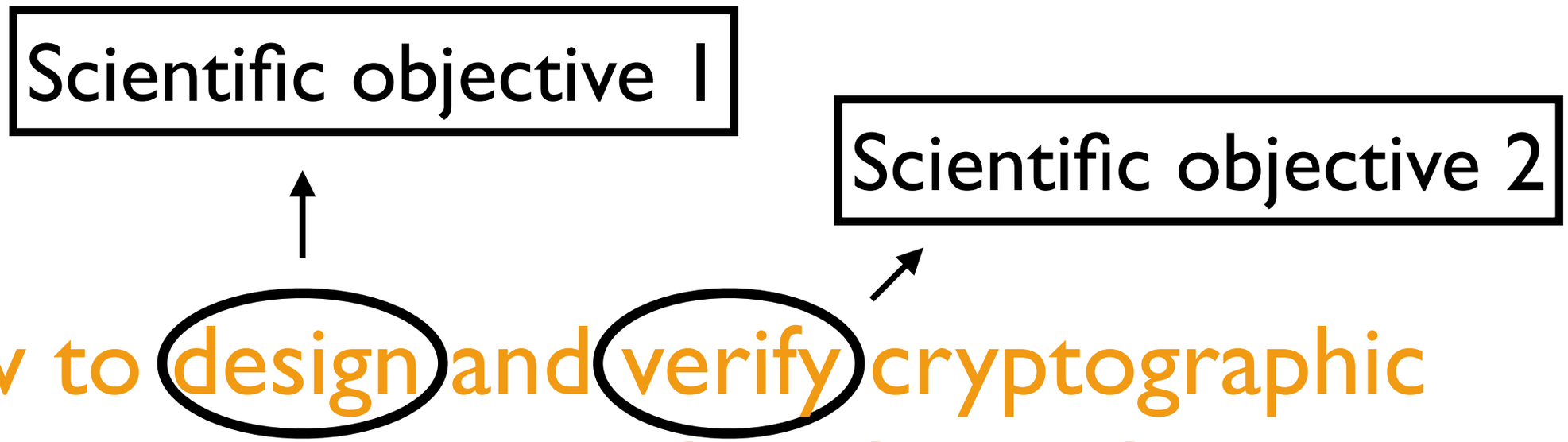
AMAskZONE



How to design and verify cryptographic implementations so that they achieve measurable practical security?

CryptoExperts

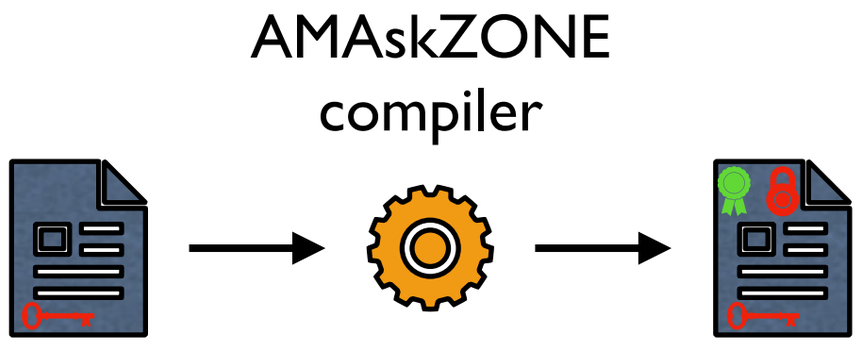
AMAskZONE

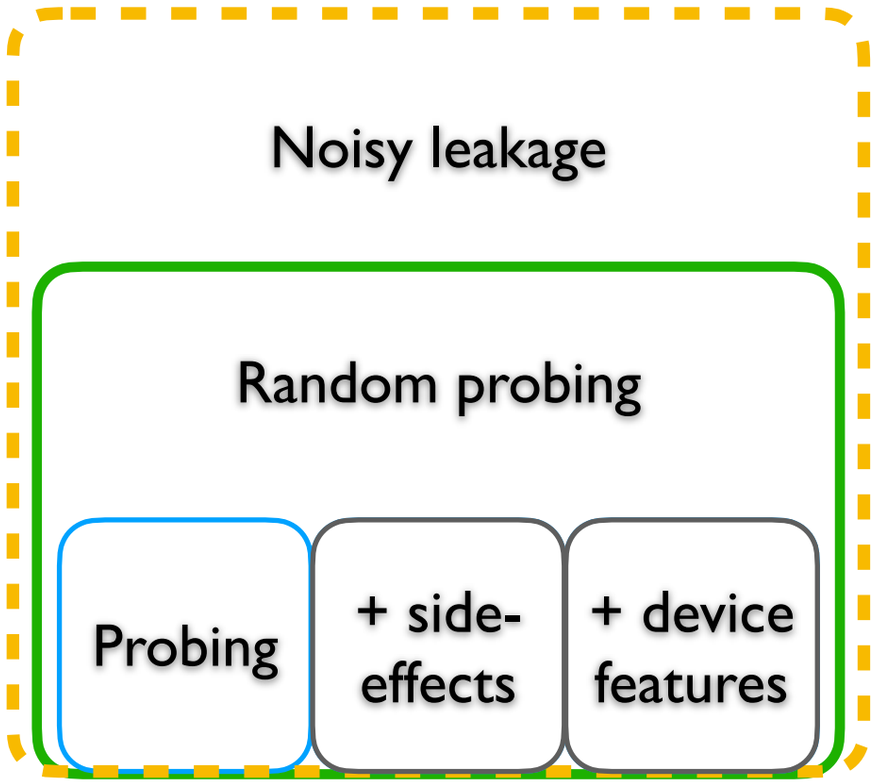


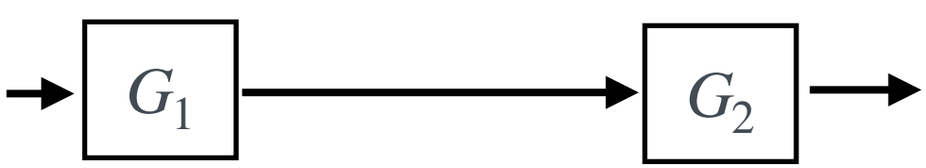
How to design and verify cryptographic implementations so that they achieve measurable practical security?

CryptoExperts

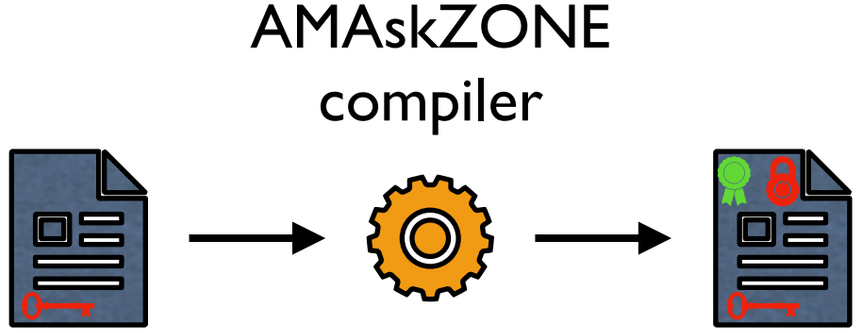
Compilers in the Random Leakage Model

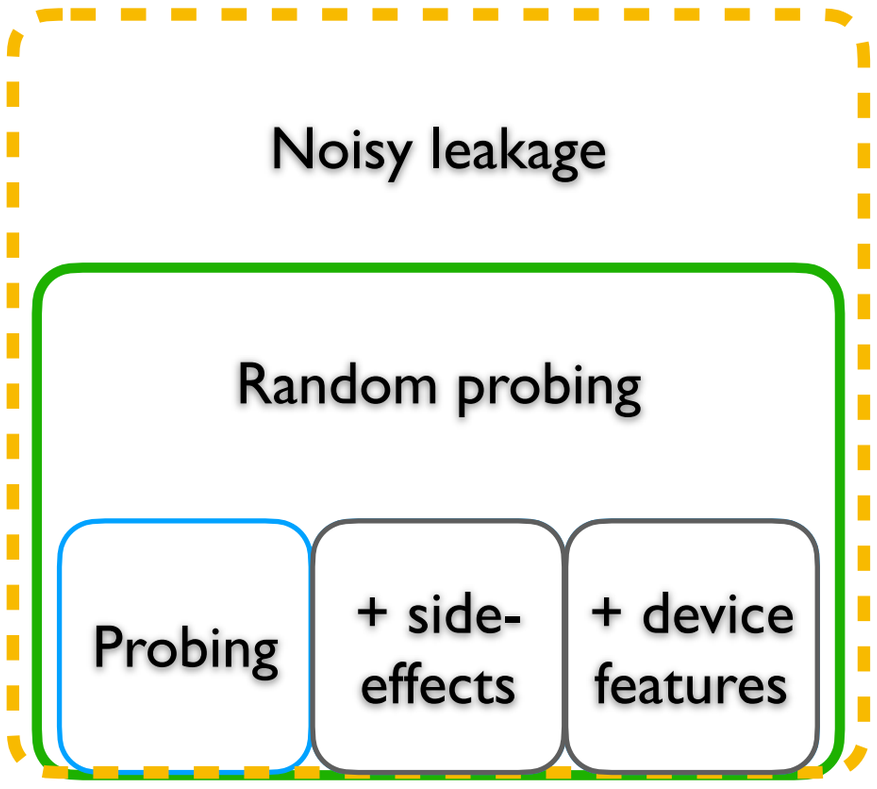






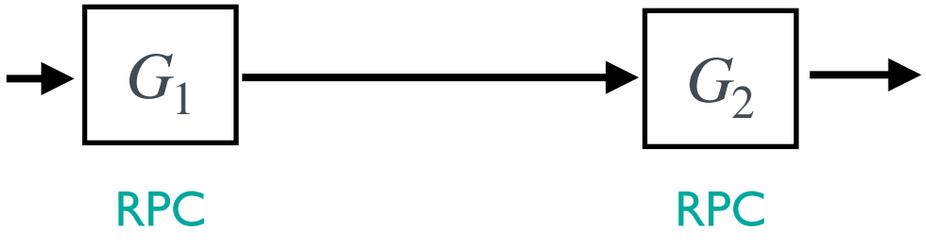
Compilers in the Random Leakage Model



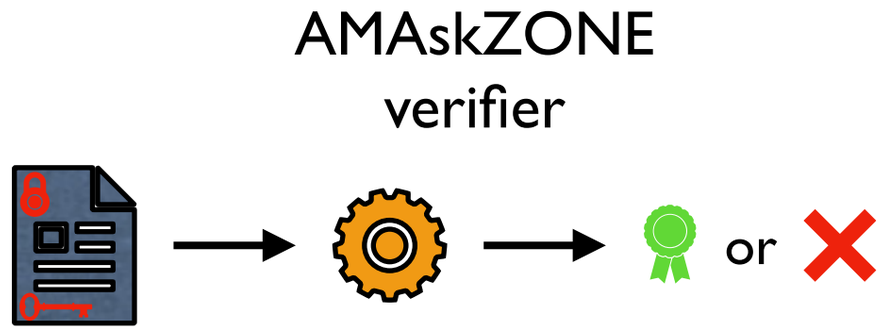


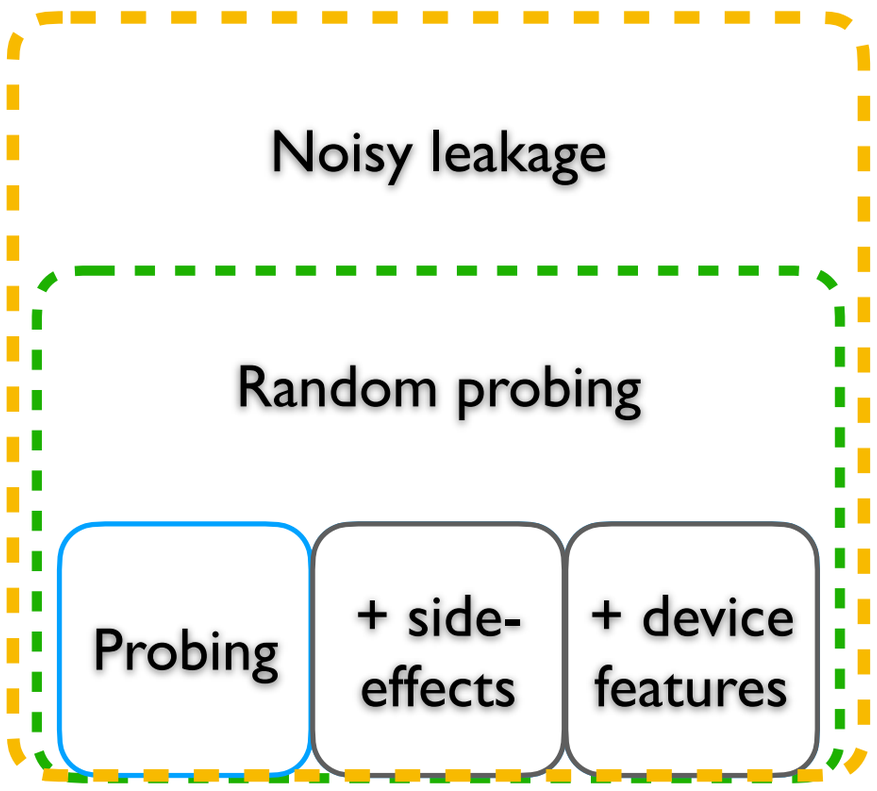
Two steps

- Identify composition rules to assemble gadgets with some security properties
  - Example: RPC security
- Build basic (then advanced) gadgets with these security guarantees
  - Example: RPC secure multiplication, addition, etc



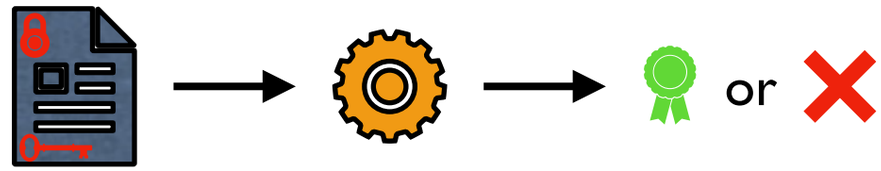
Verification with Polynomial Complexity





Verification with Polynomial Complexity

AMAskZONE verifier



Noisy leakage

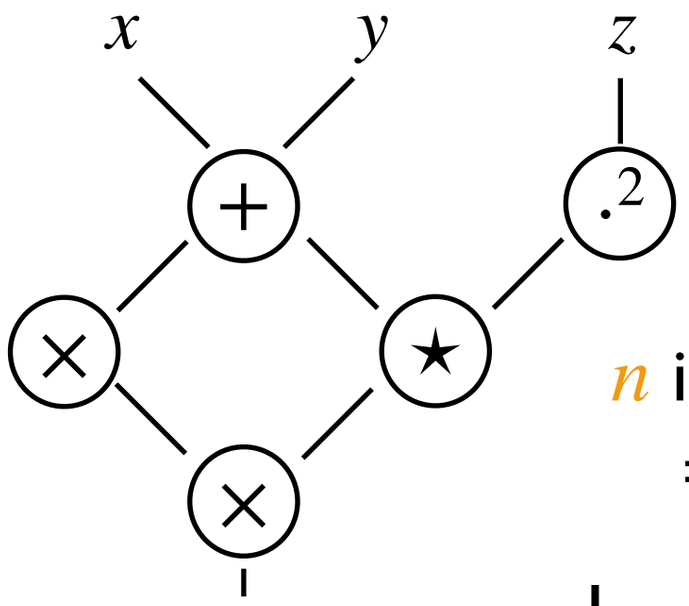
Random probing

Probing

+ side-effects

+ device features

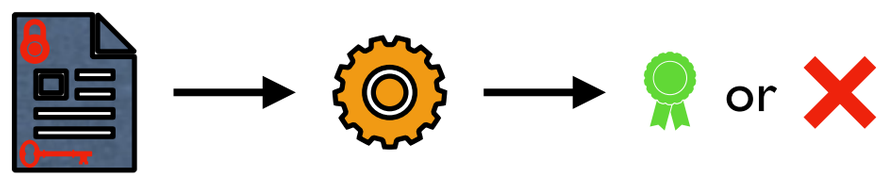
Verification is exponential in the size of the circuit

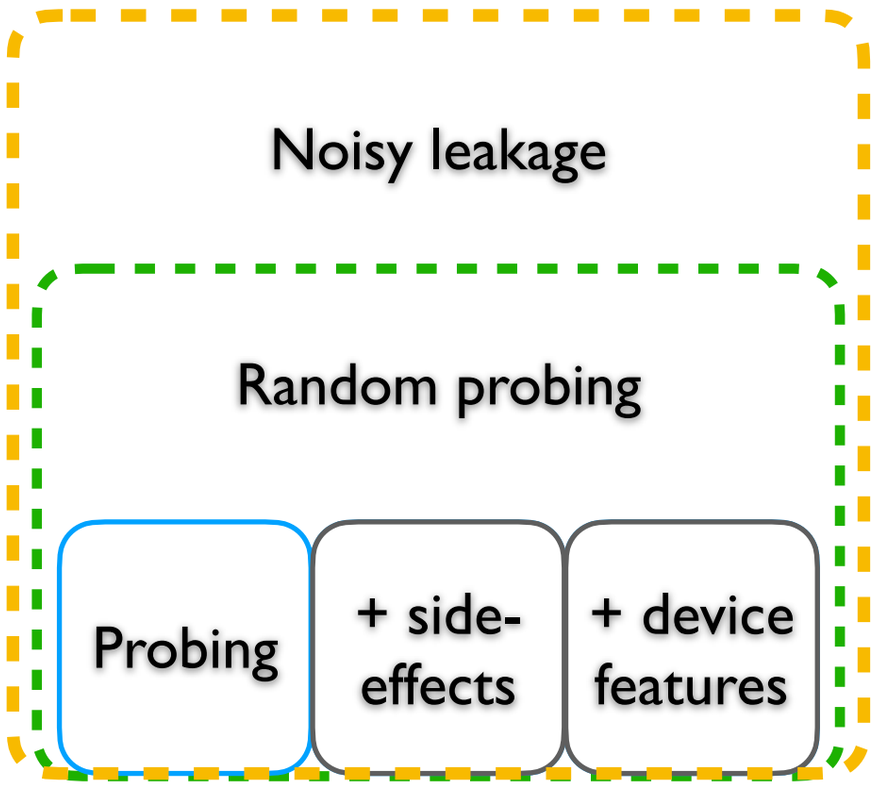


n intermediate variables
$\Rightarrow$ complexity in 2^n

We will make it polynomial

Verification with Polynomial Complexity

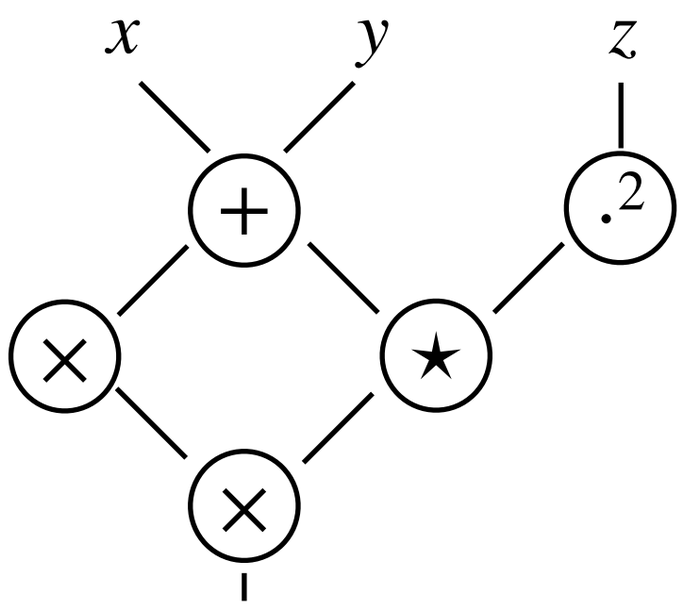




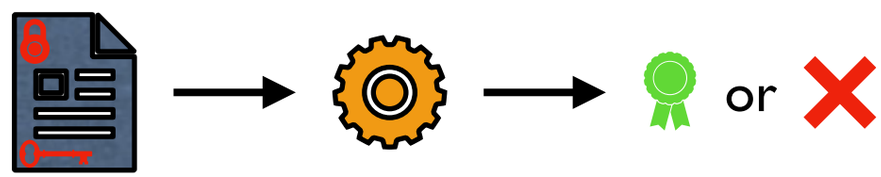
1) verification of advanced properties (linear algebra)

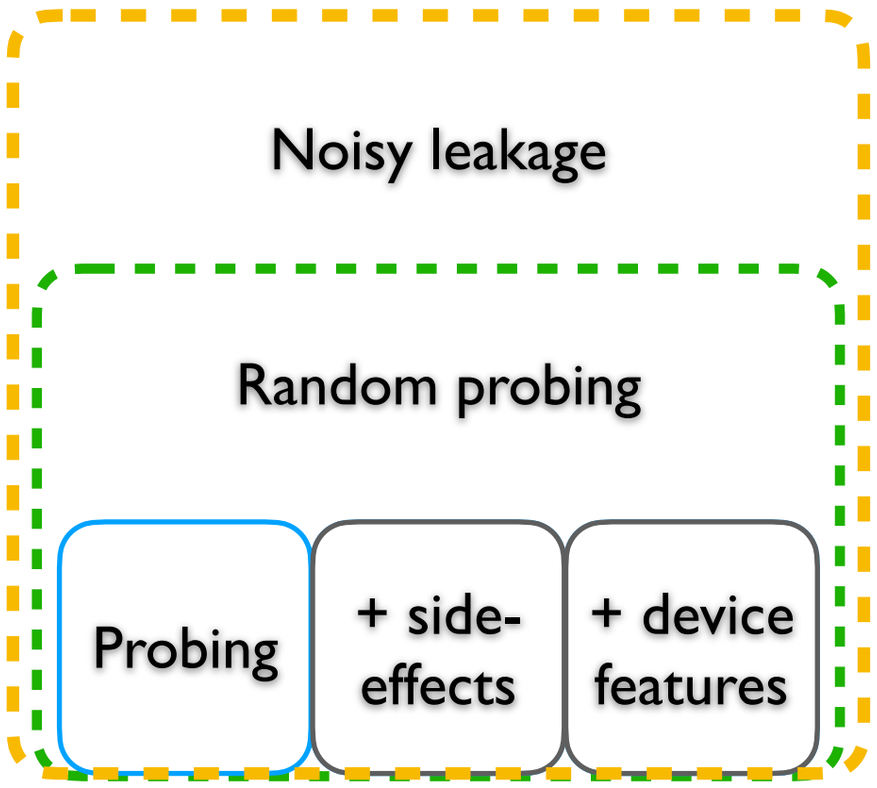
2) composition rules (linear algebra)

3) toolbox (computer science)



Verification with Polynomial Complexity

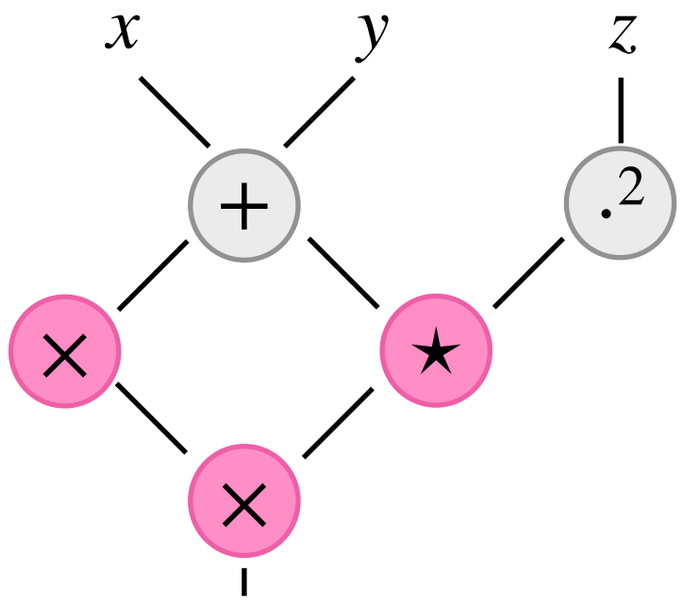




1) verification of advanced properties (linear algebra) ←

2) composition rules (linear algebra)

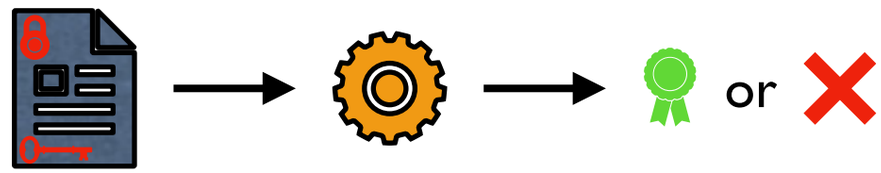
3) toolbox (computer science)

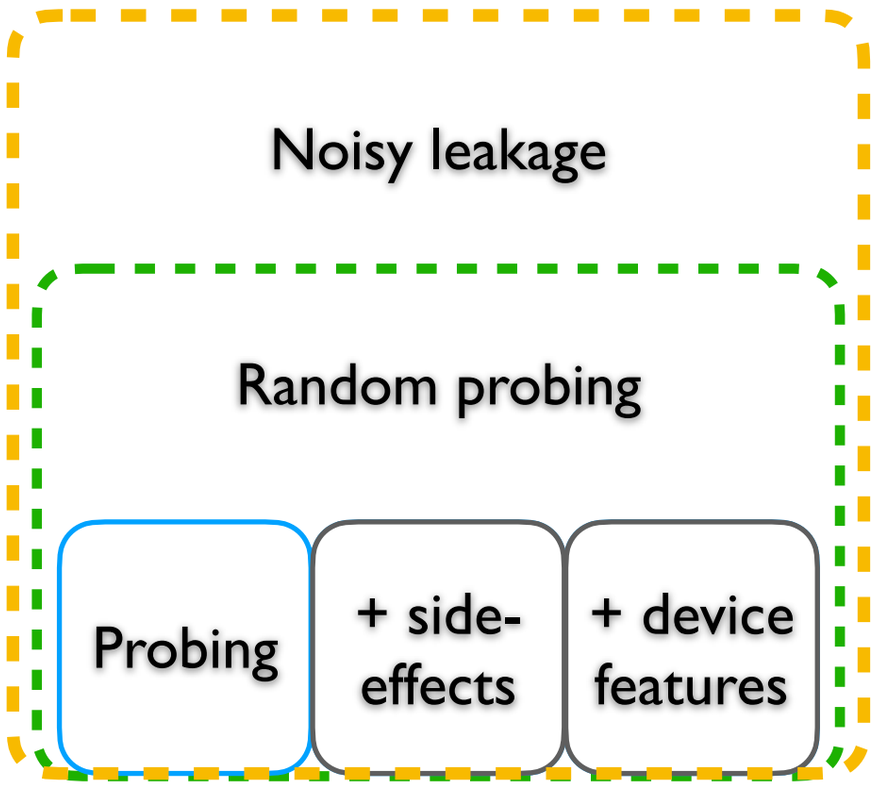


CRYPTOEXPERTS

Verification with Polynomial Complexity

AMAskZONE verifier

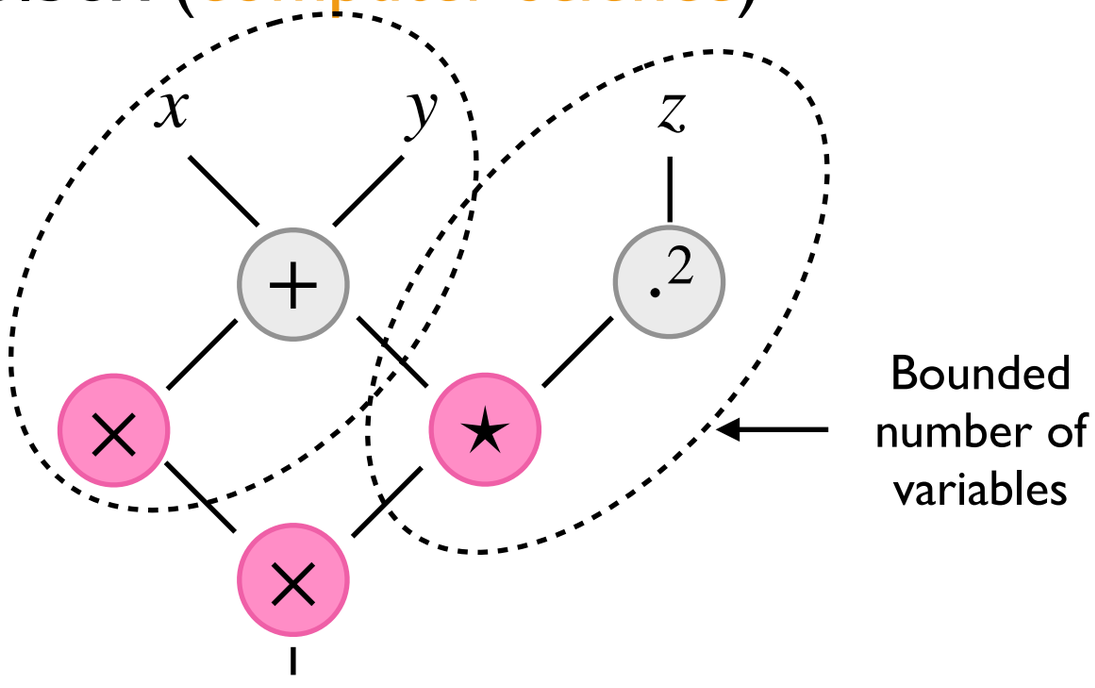




1) verification of advanced properties (linear algebra)

2) composition rules (linear algebra) ←

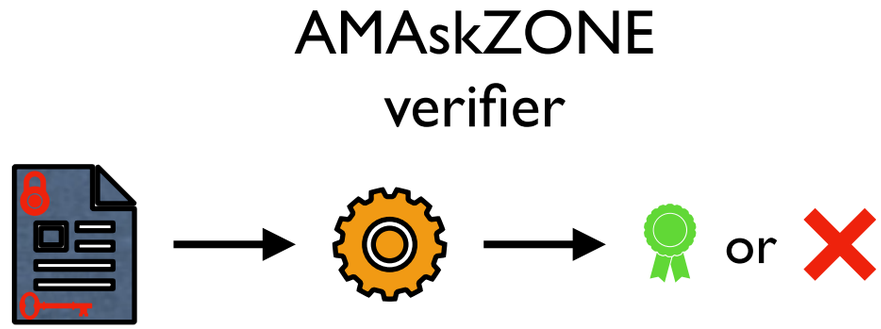
3) toolbox (computer science)

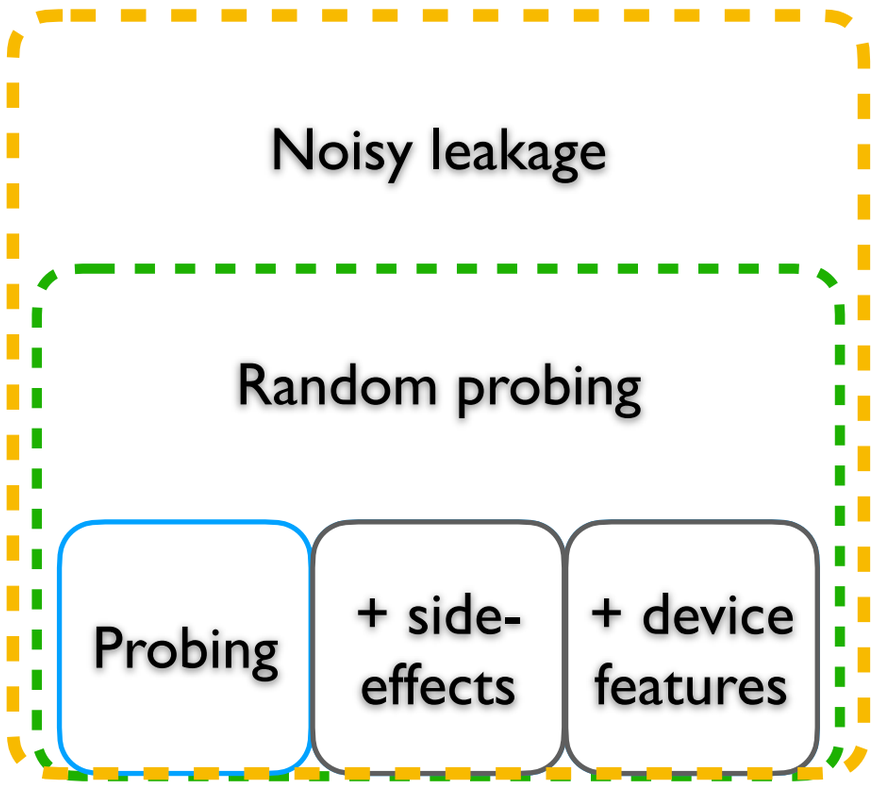


n intermediate variables ⇒ complexity in $m \cdot b$

CryptoExperts

Verification with Polynomial Complexity





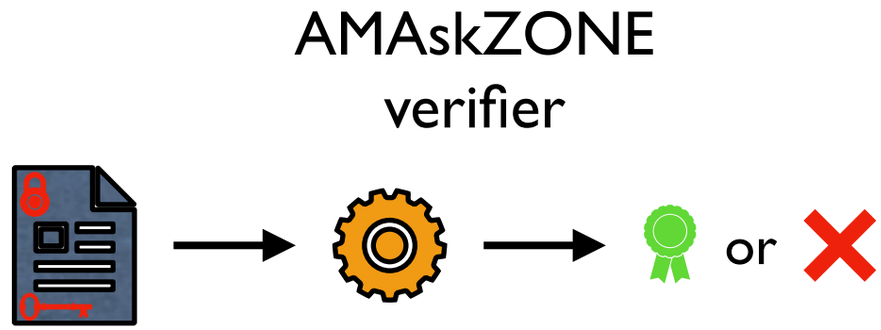
1) verification of advanced properties (linear algebra)

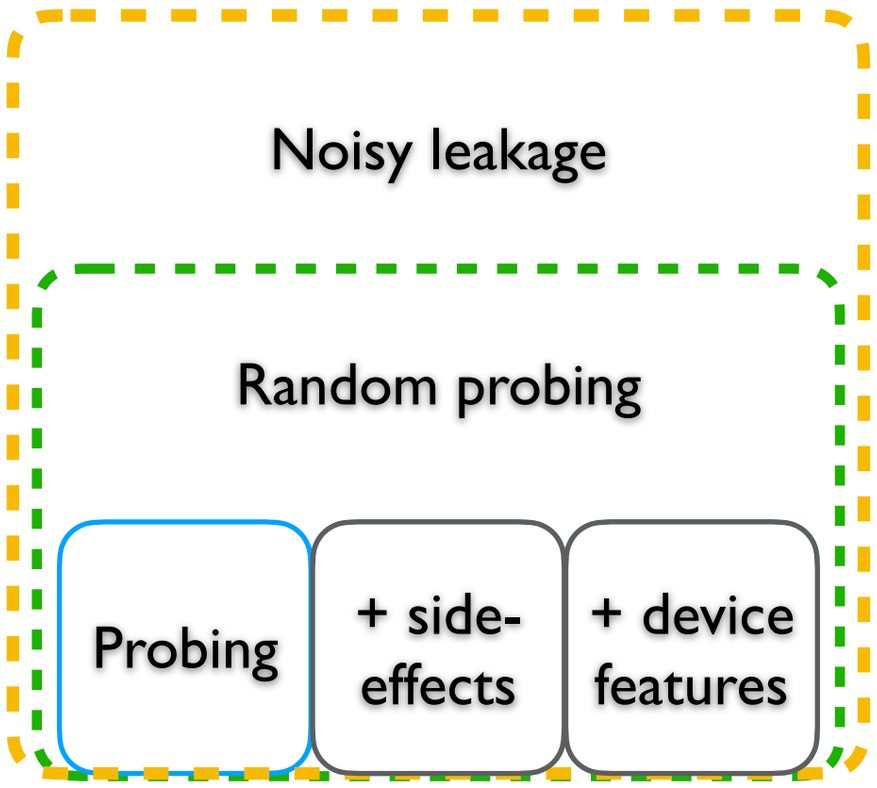
2) composition rules (linear algebra)

3) toolbox (computer science) ←

royalty-free open-source toolbox

Verification with Polynomial Complexity

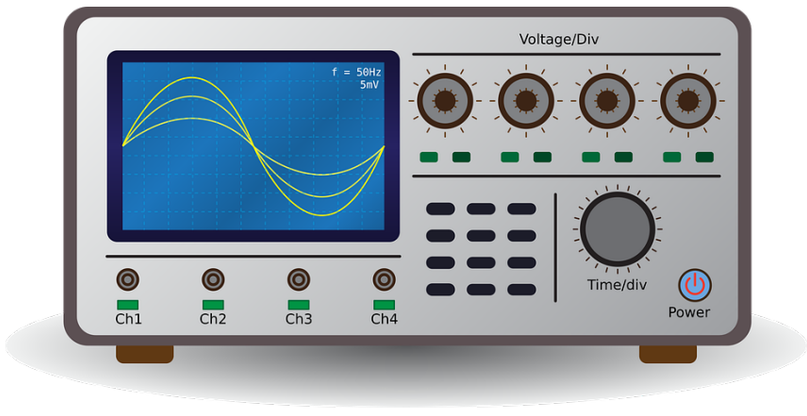




1) verification of advanced properties (linear algebra)

2) composition rules (linear algebra)

3) toolbox (computer science)



CryptoExperts

Thank you