Formal Verification of Side-Channel Countermeasures

Sonia Belaïd

June 5th 2018

1. Side-Channel Attacks
2. Masking
3. Formal Tools
   - Verification of Masked Implementations at Fixed Order
   - Verification of Masked Implementations for Generic t
   - Composition
4. Conclusion

1. Side-Channel Attacks

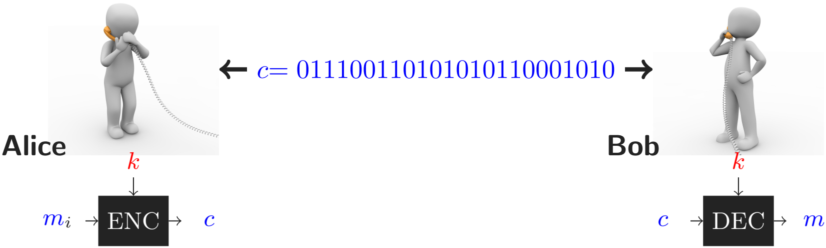
2. Masking

3. Formal Tools
   - Verification of Masked Implementations at Fixed Order
   - Verification of Masked Implementations for Generic t
   - Composition

4. Conclusion

Cryptanalysis

- Black-box cryptanalysis
- Side-channel analysis



Alice

k

$m_i \rightarrow \boxed{\text{ENC}} \rightarrow c$

$\leftarrow c=$ 011100110101010110001010 $\rightarrow$

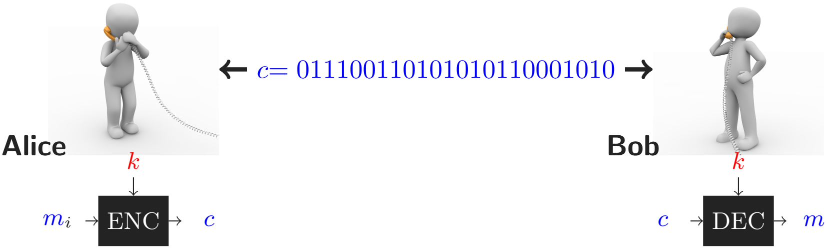
Bob

k

$c \rightarrow \boxed{\text{DEC}} \rightarrow m$

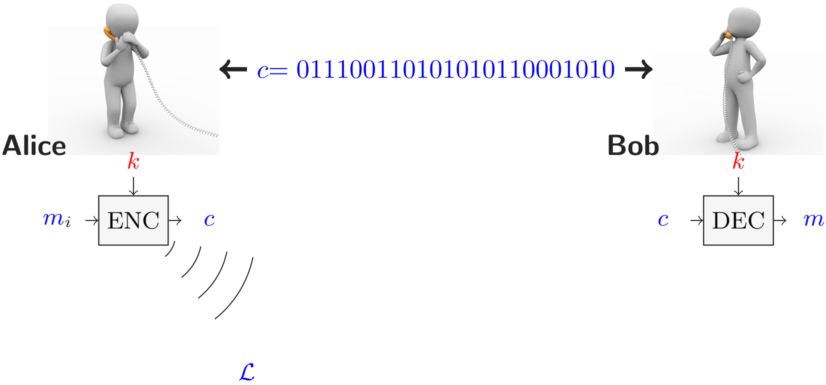
Cryptanalysis

- ➜ Black-box cryptanalysis: $\mathcal{A} \leftarrow (m, c)$
- ➜ Side-Channel Analysis



Cryptanalysis

- ➜ Black-box cryptanalysis
- ➜ Side-Channel Analysis: $\mathcal{A} \leftarrow (m, c, \mathcal{L})$



Alice

Bob

← $c=$ 0111001101010110001010 →

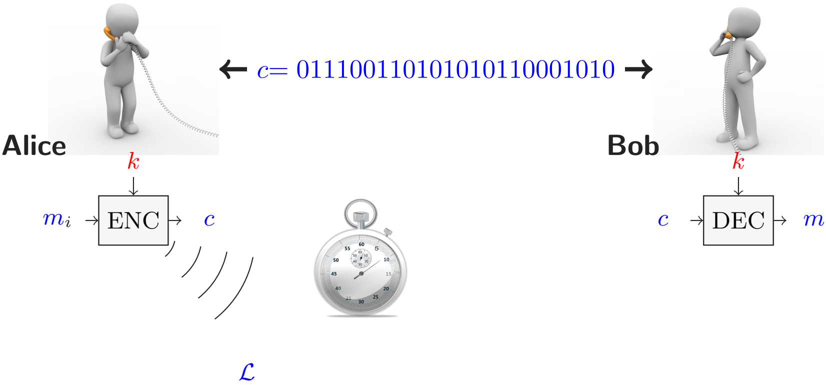
$$m_i \rightarrow \boxed{\text{ENC}} \rightarrow c \quad (k \downarrow)$$

$$c \rightarrow \boxed{\text{DEC}} \rightarrow m \quad (k \downarrow)$$

$\mathcal{L}$

Cryptanalysis

- Black-box cryptanalysis
- Side-Channel Analysis: $\mathcal{A} \leftarrow (m, c, \mathcal{L})$



$\leftarrow c=$ 011100110101010110001010 $\rightarrow$

Alice

Bob

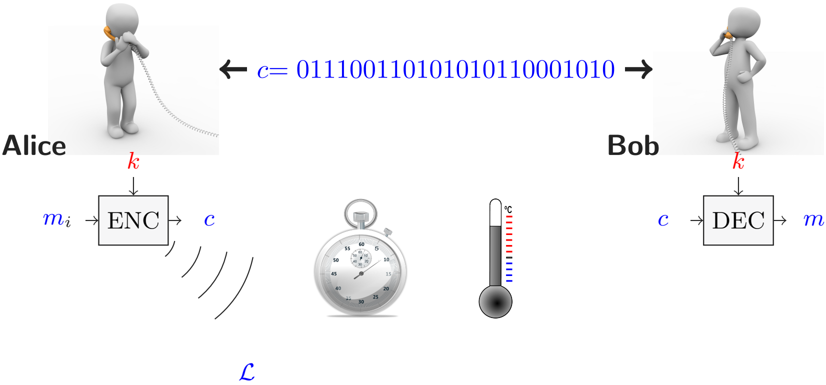
$$m_i \rightarrow \boxed{\text{ENC}} \xleftarrow{} \!\!\overset{k\,\downarrow}{} \rightarrow c$$

$$c \rightarrow \overset{k\,\downarrow}{\boxed{\text{DEC}}} \rightarrow m$$

$\mathcal{L}$

Cryptanalysis

- ➜ Black-box cryptanalysis
- ➜ Side-Channel Analysis: $\mathcal{A} \leftarrow (m, c, \mathcal{L})$



Alice

Bob

$\leftarrow c = 0111001101010110001010 \rightarrow$

k

$m_i \rightarrow \boxed{\text{ENC}} \rightarrow c$

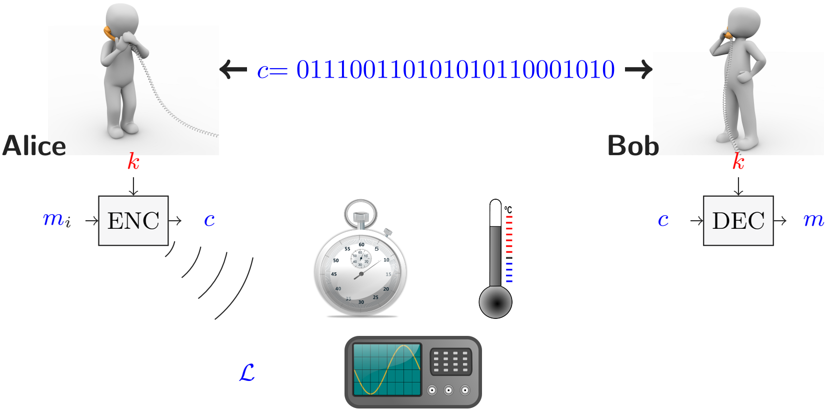
k

$c \rightarrow \boxed{\text{DEC}} \rightarrow m$

$\mathcal{L}$

Cryptanalysis

- ➜ Black-box cryptanalysis
- ➜ Side-Channel Analysis: $\mathcal{A} \leftarrow (m, c, \mathcal{L})$



$\leftarrow c = 01110011010101011000101 0 \rightarrow$

Alice

Bob

k

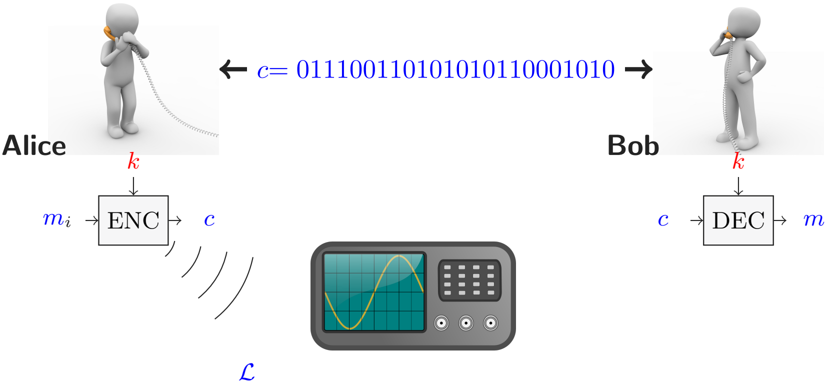
$m_i \rightarrow \boxed{\text{ENC}} \rightarrow c$

$c \rightarrow \boxed{\text{DEC}} \rightarrow m$

$\mathcal{L}$

Cryptanalysis

- Black-box cryptanalysis
- Side-Channel Analysis: $\mathcal{A} \leftarrow (m, c, \mathcal{L})$



$\leftarrow c = 011100110101010110001010 \rightarrow$

Alice

Bob

k

$m_i \rightarrow \boxed{\text{ENC}} \rightarrow c$

$\mathcal{L}$

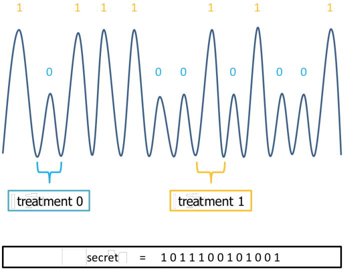
k

$c \rightarrow \boxed{\text{DEC}} \rightarrow m$

Example of SPA

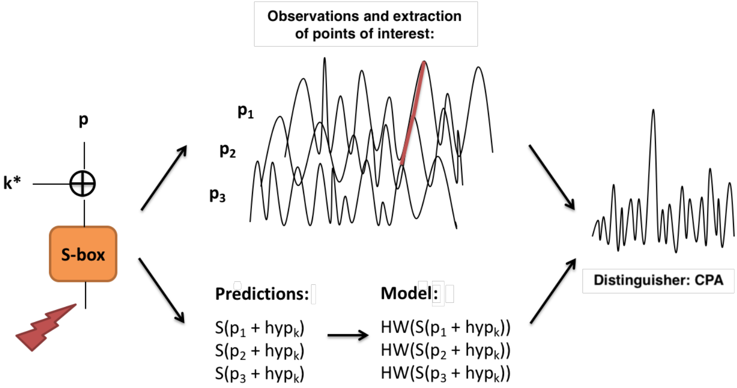
Algorithm 1 Example

```
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



Observations and extraction of points of interest:

Predictions:

$S(p_1 + hyp_k)$
$S(p_2 + hyp_k)$
$S(p_3 + hyp_k)$

Model:

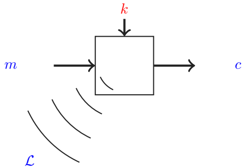
$HW(S(p_1 + hyp_k))$
$HW(S(p_2 + hyp_k))$
$HW(S(p_3 + hyp_k))$

Distinguisher: CPA

DPA: several traces to recover the secret key

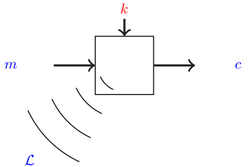
1. Side-Channel Attacks
2. Masking
3. Formal Tools
    - Verification of Masked Implementations at Fixed Order
    - Verification of Masked Implementations for Generic t
    - Composition
4. Conclusion

How to thwart SCA?



Issue: leakage $\mathcal{L}$ is key-dependent

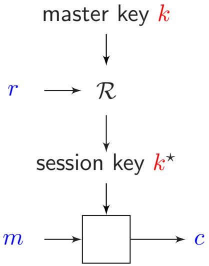
How to thwart SCA?



Issue: leakage $\mathcal{L}$ is key-dependent

Fresh Re-keying

Idea: regularly change k



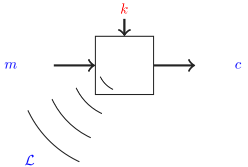
master key k

r → $\mathcal{R}$

session key $k^\star$

m → c

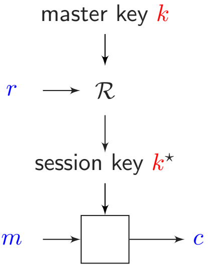
How to thwart SCA?



Issue: leakage $\mathcal{L}$ is key-dependent

Fresh Re-keying

Idea: regularly change k



master key k

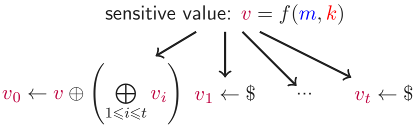
r $\longrightarrow$ $\mathcal{R}$

session key $k^\star$

m $\longrightarrow$ c

Masking

Idea: make leakage $\mathcal{L}$ random

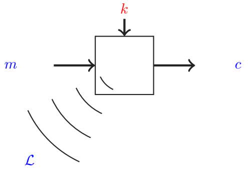


sensitive value: $v = f(m, k)$

$v_0 \leftarrow v \oplus \left(\bigoplus_{1 \leqslant i \leqslant t} v_i \right)$ $v_1 \leftarrow \$$ $\cdots$ $v_t \leftarrow \$$

➜ any t-uple of v_i is independent from v

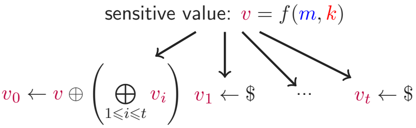
How to thwart SCA?



Issue: leakage $\mathcal{L}$ is key-dependent

Masking

Idea: make leakage $\mathcal{L}$ random


sensitive value: $v = f(m, k)$

$v_0 \leftarrow v \oplus \left(\bigoplus_{1 \leqslant i \leqslant t} v_i \right)$ $v_1 \leftarrow \$$ $\cdots$ $v_t \leftarrow \$$

➜ any t-uple of v_i is independent from v

Masked Implementations

- Linear functions: apply the function to each share

$$v \oplus w \rightarrow (v_0 \oplus w_0, v_1 \oplus w_1, \ldots, v_t \oplus w_t)$$

Masked Implementations

- Linear functions: apply the function to each share

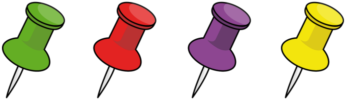
$$v \oplus w \rightarrow (v_0 \oplus w_0, v_1 \oplus w_1, \ldots, v_t \oplus w_t)$$

- Non-linear functions: much more complex

$$\begin{aligned} \forall\ 0 \leq i < j \leq t-1, &\qquad r_{i,j} \leftarrow \$ \\ \forall\ 0 \leq i < j \leq t-1, &\qquad r_{j,i} \leftarrow (r_{i,j} \oplus v_i w_j) \oplus v_j w_i \\ \forall\ 0 \leq i \leq d-1, &\qquad c_i \leftarrow v_i w_i \oplus \sum_{j \neq i} r_{i,j} \\ vw &\quad \rightarrow \quad (c_0, c_1, \ldots, c_t) \end{aligned}$$

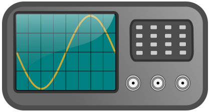
Leakage Models

- **Probing model** by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is t-probing secure iff any set composed of the exact values of at most t intermediate variables is independent from the secret



Leakage Models

- **Probing model** by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is t-probing secure iff any set composed of the exact values of at most t intermediate variables is independent from the secret
- **Noisy leakage model** by Chari, Jutla, Rao, and Rohatgi (Crypto 1999) then Rivain and Prouff (EC 2013)
  - a circuit is secure in the noisy leakage model iff the adversary cannot recover information on the secret from the noisy values of all the intermediate variables



Leakage Models

- **Probing model** by Ishai, Sahai, and Wagner (Crypto 2003)
  - a circuit is t-probing secure iff any set composed of the exact values of at most t intermediate variables is independent from the secret
- **Noisy leakage model** by Chari, Jutla, Rao, and Rohatgi (Crypto 1999) then Rivain and Prouff (EC 2013)
  - a circuit is secure in the noisy leakage model iff the adversary cannot recover information on the secret from the noisy values of all the intermediate variables
- **Reduction** by Duc, Dziembowski, and Faust (EC 2014)
  - t-probing security $\Rightarrow$ security in the noisy leakage model for some level of noise

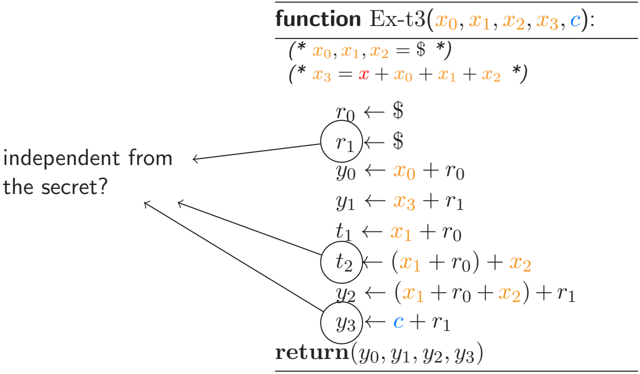
How to Verify Probing Security?

- variables: secret, shares, constant
- masking order $t = 3$

```
function Ex-t3($x_0, x_1, x_2, x_3, c$):
  (* $x_0, x_1, x_2 = \$$ *)
  (* $x_3 = x + x_0 + x_1 + x_2$ *)
      $r_0 \leftarrow \$$
      $r_1 \leftarrow \$$
      $y_0 \leftarrow x_0 + r_0$
      $y_1 \leftarrow x_3 + r_1$
      $t_1 \leftarrow x_1 + r_0$
      $t_2 \leftarrow (x_1 + r_0) + x_2$
      $y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
      $y_3 \leftarrow c + r_1$
return($y_0, y_1, y_2, y_3$)
```

How to Verify Probing Security?

- variables: secret, shares, constant
- masking order $t = 3$

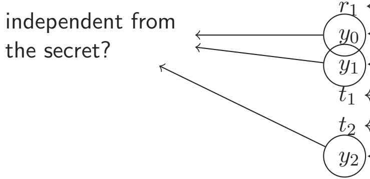


independent from the secret?

```
function Ex-t3($x_0, x_1, x_2, x_3, c$):
  (* $x_0, x_1, x_2 = \$$ *)
  (* $x_3 = x + x_0 + x_1 + x_2$ *)
    $r_0 \leftarrow \$$
    $r_1 \leftarrow \$$
    $y_0 \leftarrow x_0 + r_0$
    $y_1 \leftarrow x_3 + r_1$
    $t_1 \leftarrow x_1 + r_0$
    $t_2 \leftarrow (x_1 + r_0) + x_2$
    $y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
    $y_3 \leftarrow c + r_1$
return($y_0, y_1, y_2, y_3$)
```

How to Verify Probing Security?

- variables: secret, shares, constant
- masking order $t = 3$



independent from the secret?

```
function Ex-t3($x_0, x_1, x_2, x_3, c$):
  (* $x_0, x_1, x_2 = \$$ *)
  (* $x_3 = x + x_0 + x_1 + x_2$ *)
    $r_0 \leftarrow \$$
    $r_1 \leftarrow \$$
    $y_0 \leftarrow x_0 + r_0$
    $y_1 \leftarrow x_3 + r_1$
    $t_1 \leftarrow x_1 + r_0$
    $t_2 \leftarrow (x_1 + r_0) + x_2$
    $y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
    $y_3 \leftarrow c + r_1$
return($y_0, y_1, y_2, y_3$)
```

1 Side-Channel Attacks

2 Masking

3 Formal Tools
  - Verification of Masked Implementations at Fixed Order
  - Verification of Masked Implementations for Generic t
  - Composition

4 Conclusion

1 Side-Channel Attacks

2 Masking

3 Formal Tools
  - Verification of Masked Implementations at Fixed Order
  - Verification of Masked Implementations for Generic t
  - Composition

4 Conclusion

State-Of-The-Art

- several tools were built to formally verify security of first-order implementations $t = 1$
- then a sequence of work tackled higher-order implementations $t \leq 5$
  - `maskVerif` from Barthe et al.: first tool to achieve verification at high orders
  - `CheckMasks` from Coron: improvements in terms of efficiency
  - Bloem et al.'s tool: treatment of glitches attacks

State-Of-The-Art

- several tools were built to formally verify security of first-order implementations $t = 1$
- then a sequence of work tackled higher-order implementations $t \leq 5$
  - `maskVerif` from Barthe et al.: first tool to achieve verification at high orders
  - `CheckMasks` from Coron: improvements in terms of efficiency
  - Bloem et al.'s tool: treatment of glitches attacks

`maskVerif`

- input:
  - pseudo-code of a masked implementation
  - order t
- output:
  - formal proof of t-probing security
  - potential flaws
- language: Easycrypt

Gilles Barthe and Sonia Belaïd and François Dupressoir and Pierre-Alain Fouque and Benjamin Grégoire and Pierre-Yves Strub *Verified Proofs of Higher-Order Masking,* EUROCRYPT 2015, Proceedings, Part I, 457–485.

Independence from the secret

Inputs: t intermediate variables, $b \leftarrow \mathrm{true}$

(Rule 1) secret variables?
- yes ➜ (Rule 2)
- no ➜ ✔

(Rule 2) an expression v is invertible in the only occurrence of a random r?
- yes ➜ $v \leftarrow r$; (Rule 1)
- no ➜ (Rule 3)

(Rule 3) is flag $b = \mathrm{true}$?
- yes ➜ simplify; $b \leftarrow \mathrm{false}$; (Rule 1)
- no ➜ ✘

```
function Ex-t3($x_1, x_2, x_3, x_4, c$):
    $r_1 \leftarrow \$$
    $r_2 \leftarrow \$$
    $y_1 \leftarrow x_1 + r_1$
    $y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$
    $t_1 \leftarrow x_2 + r_1$
    $t_2 \leftarrow (x_2 + r_1) + x_3$
    $y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$
    $y_4 \leftarrow c + r_2$
return($y_1, y_2, y_3, y_4$)
```

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

Independence from the secret

Inputs: t intermediate variables, $b \leftarrow \text{true}$

(Rule 1) secret variables?

- yes ➜ (Rule 2)
- no ➜ ✔

(Rule 2) an expression v is invertible in the only occurrence of a random r?

- yes ➜ $v \leftarrow r$; (Rule 1)
- no ➜ (Rule 3)

(Rule 3) is flag $b = \text{true}$?

- yes ➜ simplify; $b \leftarrow \text{false}$; (Rule 1)
- no ➜ ✘

```
function Ex-t3($x_1, x_2, x_3, x_4, c$):
    $\boxed{r_1} \leftarrow \$$
    $r_2 \leftarrow \$$
    $\boxed{y_1} \leftarrow x_1 + r_1$
    $y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$
    $t_1 \leftarrow x_2 + r_1$
    $t_2 \leftarrow (x_2 + r_1) + x_3$
    $\boxed{y_3} \leftarrow (x_2 + r_1 + x_3) + r_2$
    $y_4 \leftarrow c + r_2$
return($y_1, y_2, y_3, y_4$)
```

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

Independence from the secret

Inputs: t intermediate variables, $b \leftarrow \mathrm{true}$

(Rule 1) secret variables?

- yes ➜ (Rule 2)
- no ➜ ✔

(Rule 2) an expression v is invertible in the only occurrence of a random r?

- yes ➜ $v \leftarrow r$; (Rule 1)
- no ➜ (Rule 3)

(Rule 3) is flag $b = \mathrm{true}$?

- yes ➜ simplify; $b \leftarrow \mathrm{false}$; (Rule 1)
- no ➜ ✘

```
function Ex-t3($x_1, x_2, x_3, x_4, c$):
    $r_1 \leftarrow \$$
    $\enclose{circle}{r_2} \leftarrow \$$
    $\enclose{circle}{y_1} \leftarrow x_1 + r_1$
    $\enclose{circle}{y_2} \leftarrow (x + x_1 + x_2 + x_3) + r_2$
    $t_1 \leftarrow x_2 + r_1$
    $t_2 \leftarrow (x_2 + r_1) + x_3$
    $y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$
    $y_4 \leftarrow c + r_2$
return($y_1, y_2, y_3, y_4$)
```

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

Independence from the secret

Inputs: t intermediate variables, $b \leftarrow \text{true}$

(Rule 1) secret variables?
- yes ➜ (Rule 2)
- no ➜ ✔

(Rule 2) an expression v is invertible in the only occurrence of a random r?
- yes ➜ $v \leftarrow r$; (Rule 1)
- no ➜ (Rule 3)

(Rule 3) is flag $b = \text{true}$?
- yes ➜ simplify; $b \leftarrow \text{false}$; (Rule 1)
- no ➜ ✘

```
function Ex-t3($x_1, x_2, x_3, x_4, c$):
    $r_1 \leftarrow \$$
    $\boxed{r_2} \leftarrow \$$
    $\boxed{y_1} \leftarrow x_1 + r_1$
    $\boxed{y_2} \leftarrow x_3$
    $t_1 \leftarrow x_2 + r_1$
    $t_2 \leftarrow (x_2 + r_1) + x_3$
    $y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$
    $y_4 \leftarrow c + r_2$
return($y_1, y_2, y_3, y_4$)
```

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

Extension to All Possible Sets

Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

Extension to All Possible Sets

Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice

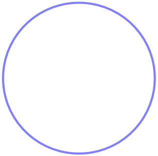
Extension to All Possible Sets

Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice

Algorithm 1:



Extension to All Possible Sets

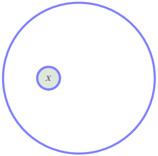
Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice

Algorithm 1:

1. select $X = (t$ variables) and prove its independence

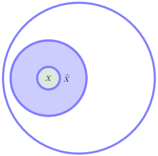


Extension to All Possible Sets

Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice



Algorithm 1:

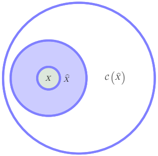
1. select $X = (t$ variables) and prove its independence
2. extend X to $\widehat{X}$ with more observations but still independence

Extension to All Possible Sets

Problen: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice



Algorithm 1:

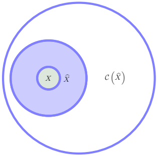
1. select $X = (t$ variables) and prove its independence
2. extend X to $\widehat{X}$ with more observations but still independence
3. recursively descend in set $\mathcal{C}\left(\widehat{X}\right)$

Extension to All Possible Sets

Problem: n intermediate variables ➜ $\binom{n}{t}$ proofs

New Idea: proofs for sets of more than t variables

- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice



Algorithm 1:

1. select $X = (t$ variables$)$ and prove its independence
2. extend X to $\widehat{X}$ with more observations but still independence
3. recursively descend in set $\mathcal{C}\left(\widehat{X}\right)$
4. merge $\widehat{X}$ and $\mathcal{C}\left(\widehat{X}\right)$ once they are processed separately.

Benchmarks

Reference	Target	# tuples	Security	Complexity	
				# sets	time (s)
First-Order Masking					
FSE13	full AES	17,206	✔	3,342	128
MAC-SHA3	full Keccak-f	13,466	✔	5,421	405
Second-Order Masking					
RSA06	Sbox	1,188,111	✔	4,104	1.649
CHES10	Sbox	7,140	1^{st}-order flaws (2)	866	0.045
CHES10	AES KS	23,041,866	✔	771,263	340,745
FSE13	2 rnds AES	25,429,146	✔	511,865	1,295
FSE13	4 rnds AES	109,571,806	✔	2,317,593	40,169
Third-Order Masking					
RSA06	Sbox	2,057,067,320	3^{rd}-order flaws $(98, 176)$	2,013,070	695
FSE13	Sbox(4)	4,499,950	✔	33,075	3.894
FSE13	Sbox(5)	4,499,950	✔	39,613	5.036
Fourth-Order Masking					
FSE13	Sbox (4)	$2,277,036,685$	✔	3,343,587	879
Fifth-Order Masking					
CHES10	$\odot$	216,071,394	✔	856,147	45

*run on a headless VM with a dual core (only one core is used in the computation) 64-bit processor clocked at 2GHz

1 Side-Channel Attacks

2 Masking

3 Formal Tools

- Verification of Masked Implementations at Fixed Order
- Verification of Masked Implementations for Generic t
- Composition

4 Conclusion

Probing Model

```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 1$ to $t$ do
    $r \leftarrow \$$
    $x_0 \leftarrow x_0 + r$
    $x_i \leftarrow x_i + r$
  end for
  return $[x]$
```

Probing Model

```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 1$ to $t$ do
    $r \leftarrow \$$
    $x_0 \leftarrow x_0 + r$
    $x_i \leftarrow x_i + r$
  end for
  return $[x]$
```

Simulation-based proof:

- show that any set of t variables can be simulated with at most t input shares x_i
- any set of t shares x_i is independent from x

Probing Model

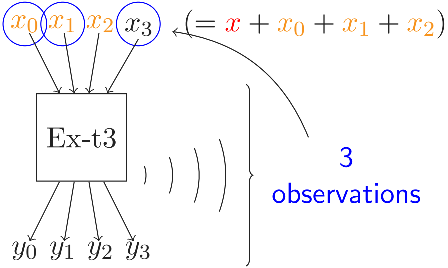
```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 1$ to $t$ do
    $r \leftarrow \$$
    $x_0 \leftarrow x_0 + r$
    $x_i \leftarrow x_i + r$
  end for
  return $[x]$
```

Simulation-based proof:

- show that any set of t variables can be simulated with at most t input shares x_i
- any set of t shares x_i is independent from x

Non-Interference (NI)

- t-NI $\Rightarrow$ t-probing secure
- a circuit is t-NI iff any set of t intermediate variables can be perfectly simulated with at most t shares of each input



And then?

once done for small gadgets, how to extend it?

1. Side-Channel Attacks

2. Masking

3. Formal Tools
   - Verification of Masked Implementations at Fixed Order
   - Verification of Masked Implementations for Generic t
   - Composition

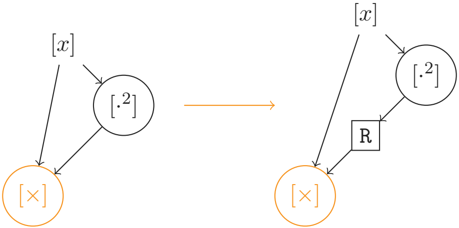
4. Conclusion

Until Recently

- composition probing secure for $2t + 1$ shares
- no solution for $t + 1$ shares

First Proposal

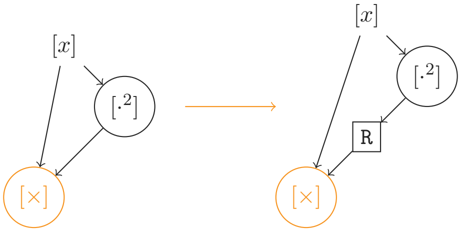
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
    for $i = 1$ to $t$ do
        $r \leftarrow \$$
        $x_0 \leftarrow x_0 + r$
        $x_i \leftarrow x_i + r$
    end for
    return $[x]$
```

First Proposal

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

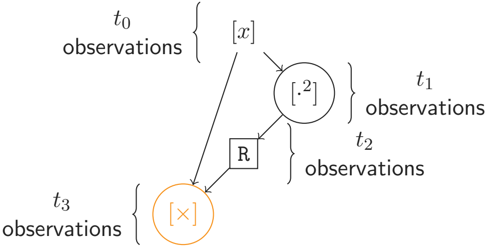


```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
    for $i = 1$ to $t$ do
        $r \leftarrow \$$
        $x_0 \leftarrow x_0 + r$
        $x_i \leftarrow x_i + r$
    end for
    return $[x]$
```

$\Rightarrow$ Flaw from $t = 2$ (FSE 2013: Coron, Prouff, Rivain, and Roche)

Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



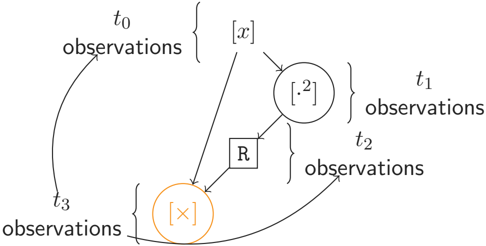
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:

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



t_0 observations

$[x]$

$[\cdot^2]$

t_1 observations

R

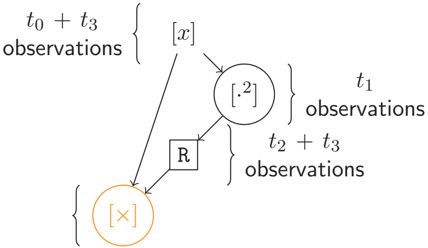
t_2 observations

$[\times]$

t_3 observations

Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

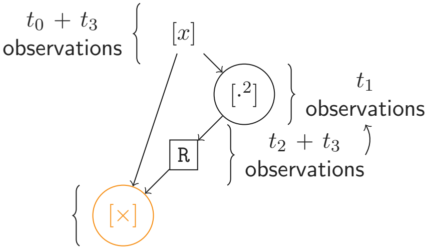


Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



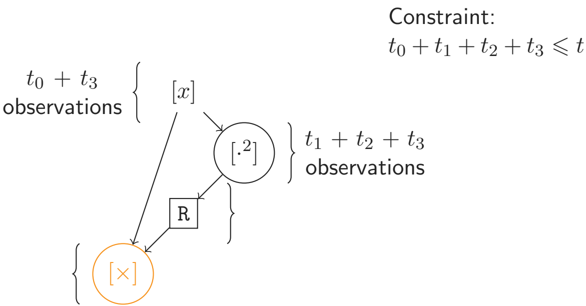
$t_0 + t_3$ observations

t_1 observations

$t_2 + t_3$ observations

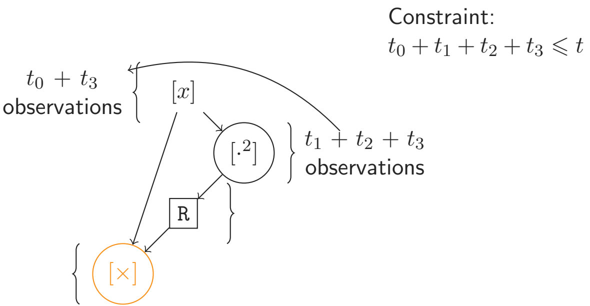
Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Why This Flaw?

- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:

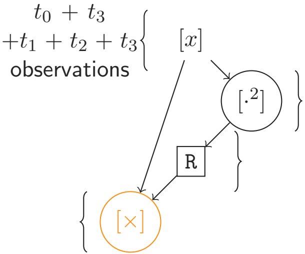
$t_0 + t_1 + t_2 + t_3 \leqslant t$

$t_0 + t_3$ observations

$t_1 + t_2 + t_3$ observations

Why This Flaw?

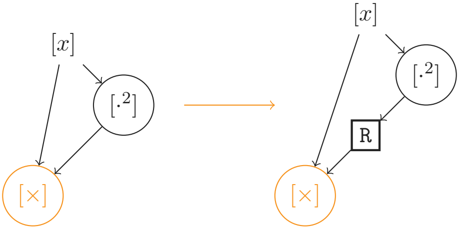
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Second Proposal

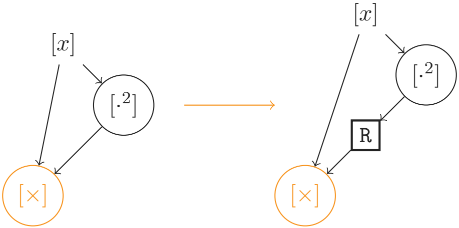
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 0$ to $t$ do
    for $j = i + 1$ to $t$ do
      $r \leftarrow \$$
      $x_i \leftarrow x_i + r$
      $x_j \leftarrow x_j + r$
    end for
  end for
  return $[x]$
```

Second Proposal

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

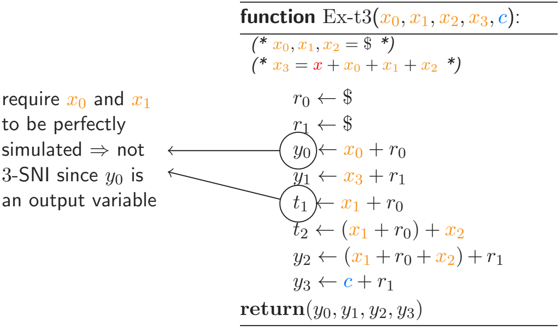


```
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 0$ to $t$ do
    for $j = i + 1$ to $t$ do
      $r \leftarrow \$$
      $x_i \leftarrow x_i + r$
      $x_j \leftarrow x_j + r$
    end for
  end for
  return $[x]$
```

$\Rightarrow$ Formal security proof for any order t

Strong Non-Interference (SNI)

- t-SNI ⇒ t-NI ⇒ t-probing secure
- a circuit is t-SNI iff any set of t intermediate variables, whose t_1 on the internal variables and t_2 and the outputs, can be perfectly simulated with at most t_1 shares of each input

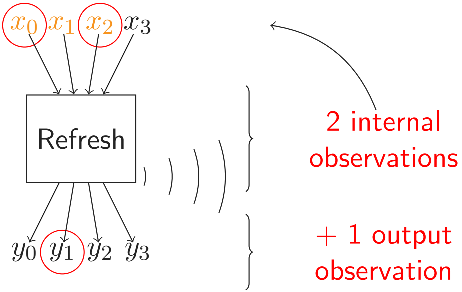


require x_0 and x_1 to be perfectly simulated ⇒ not 3-SNI since y_0 is an output variable

```
function Ex-t3($x_0, x_1, x_2, x_3, c$):
  (* $x_0, x_1, x_2 = \$$ *)
  (* $x_3 = x + x_0 + x_1 + x_2$ *)
      $r_0 \leftarrow \$$
      $r_1 \leftarrow \$$
      $y_0 \leftarrow x_0 + r_0$
      $y_1 \leftarrow x_3 + r_1$
      $t_1 \leftarrow x_1 + r_0$
      $t_2 \leftarrow (x_1 + r_0) + x_2$
      $y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
      $y_3 \leftarrow c + r_1$
return($y_0, y_1, y_2, y_3$)
```

Strong Non-Interference (SNI)

- t-SNI ⇒ t-NI ⇒ t-probing secure
- a circuit is t-SNI iff any set of t intermediate variables, whose t_1 on the internal variables and t_2 and the outputs, can be perfectly simulated with at most t_1 shares of each input

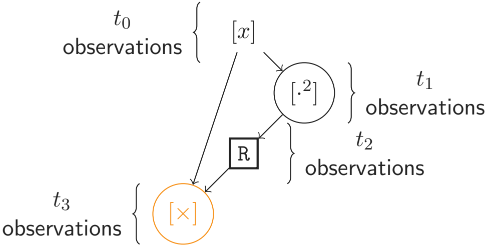


Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:

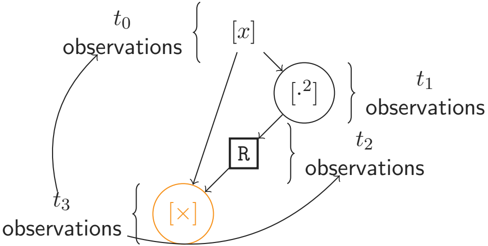
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why Does It Works?

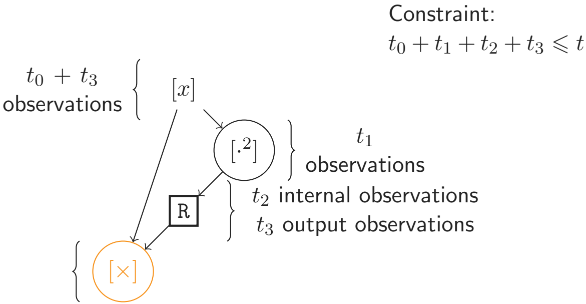
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

$t_0 + t_3$
observations

$[x]$

$[\cdot^2]$

t_1
observations

R

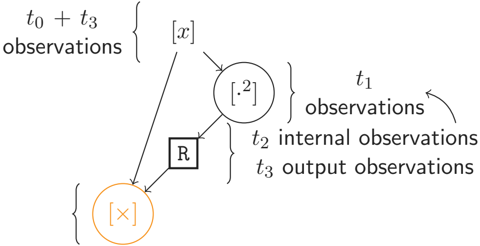
t_2 internal observations
t_3 output observations

$[\times]$

Why Does It Works?

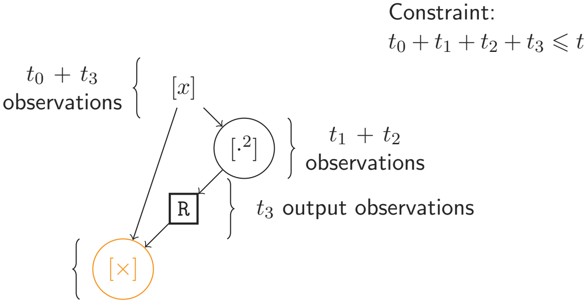
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$

Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

$t_0 + t_3$ observations

$[x]$

$[\cdot^2]$

$t_1 + t_2$ observations

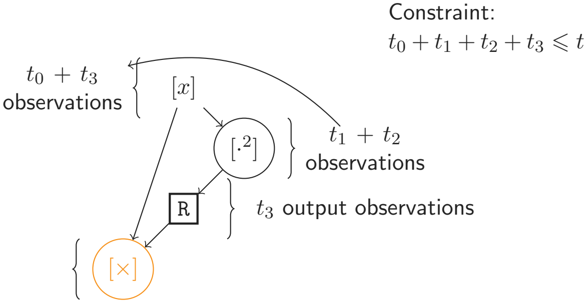
R

t_3 output observations

$[\times]$

Why Does It Works?

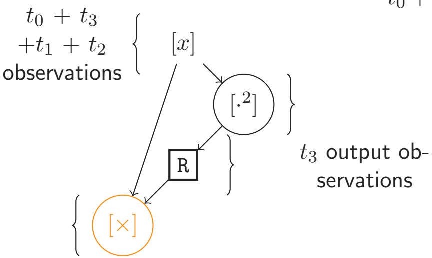
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)
- Example: AES S-box on $\mathrm{GF}(2^8)$



Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Tool `maskComp`

- from t-NI and t-SNI gadgets ⇒ build a t-NI circuit by inserting t-SNI refresh gadgets at carefully chosen locations
- formally proven

Implementation in
C language with
no countermeasure

`maskComp`

t-NI secure
implementation
in C language

Gilles Barthe and Sonia Belaïd and François Dupressoir and Pierre-Alain Fouque and Benjamin Grégoire and Pierre-Yves Strub *Strong Non-Interference and Type-Directed Higher-Order Masking and Rebecca Zucchini,* ACM CCS 2016, Proceedings, 116–129.

Limitations of `maskComp`

- `maskComp` adds a refresh gadget to Circuit 1
- but Circuit 1 was already t-probing secure

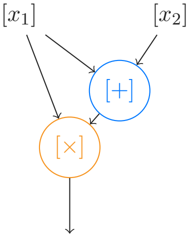


Figure: Circuit 1.

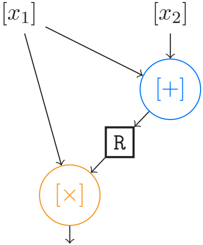


Figure: Circuit 1 after `maskComp`.

New Proposal

- Joint work with Dahmun Goudarzi and Matthieu Rivain
- Apply to tight shared circuits:
  - sharewise additions,
  - ISW-multiplications,
  - ISW-refresh gadgets
- Determine exactly whether a tight shared circuit is probing secure for any order t
  1. Reduction to a simplified problem
  2. Resolution of the simplified problem
  3. Extension to larger circuits

1 Side-Channel Attacks

2 Masking

3 Formal Tools

- Verification of Masked Implementations at Fixed Order
- Verification of Masked Implementations for Generic t
- Composition

4 Conclusion

Conclusion

In a nutshell...

- Formal tools to verify security of masked implementations
- Trade-off between security and performances

To continue...

- Achieve better performances
- Apply such formal verifications to every circuit