Secure Masked Implementations with the Least Refreshing

Sonia Belaïd

March, 18th 2019

CryptoExperts

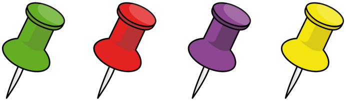
Power Analysis Attacks

Masking

- split every sensitive variable x into $t+1$ shares $(x_i)_{0 \leq i \leq t}$ such that
  - for every $1 \leq i \leq t$, x_i is picking uniformly at random
  - $x_0 \leftarrow x \oplus x_1 \oplus \cdots \oplus x_t$
- any strict subvector of at most t shares is independent from x
- t is called *masking order* or *security order*

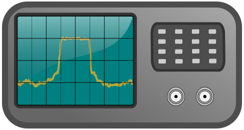
Leakage Models

- **Probing model**
  - any set of t intermediate variables independent from secrets



Leakage Models

- **Probing model**
  - any set of t intermediate variables independent from secrets
- **Noisy leakage model**
  - all noisy functions of intermediate variables are jointly independent from secrets



Leakage Models

- **Probing model**
  - any set of t intermediate variables independent from secrets
- **Noisy leakage model**
  - all noisy functions of intermediate variables are jointly independent from secrets
- **Reduction**

Probing Model

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

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$$
\begin{aligned}
r_0 &\leftarrow \$ \\
r_1 &\leftarrow \$ \\
y_0 &\leftarrow x_0 + r_0 \\
y_1 &\leftarrow x_3 + r_1 \\
t_1 &\leftarrow x_1 + r_0 \\
t_2 &\leftarrow (x_1 + r_0) + x_2 \\
y_2 &\leftarrow (x_1 + r_0 + x_2) + r_1 \\
y_3 &\leftarrow c + r_1
\end{aligned}
$$

return(y_0, y_1, y_2, y_3)

Probing Model

- 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)

independent from the secret?

Probing Model

- 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)

independent from the secret?

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

function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

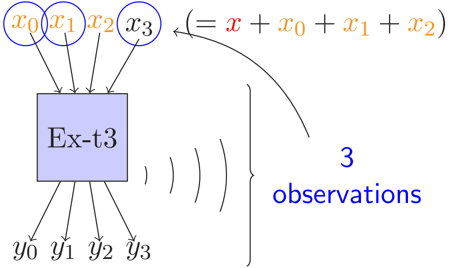
$r_0 \leftarrow \$$
$r_1 \leftarrow \$$
$y_0 \leftarrow x_0 + r_0$
$y_1 \leftarrow x_3 + r_1$
$t_1 \leftarrow x_1 + r_0$
$t_2 \leftarrow (x_1 + r_0) + x_2$
$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
$y_3 \leftarrow c + r_1$

return(y_0, y_1, y_2, y_3)

can be simulated with x_0 and x_1

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

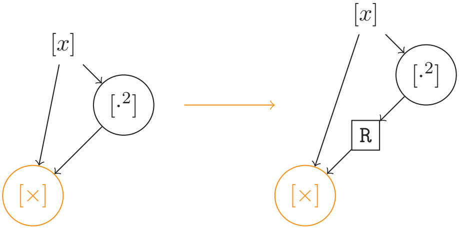


Until Recently

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

First Proposal

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

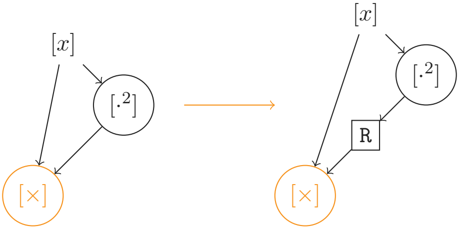


Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$

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) on AES S-box on $\mathrm{GF}(2^8)$



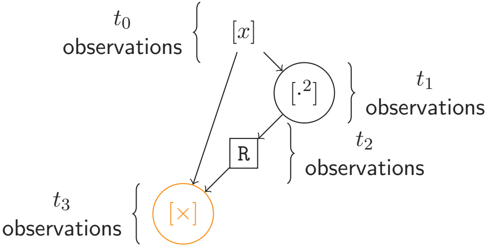
Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$
  for $i = 1$ to t **do**
    $r \leftarrow \$$
    $x_0 \leftarrow x_0 + r$
    $x_i \leftarrow x_i + r$
  end for
  return $[x]$

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

Why This Flaw?

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

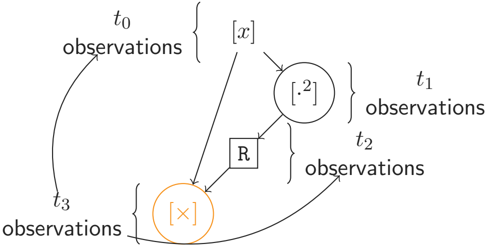
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why This Flaw?

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

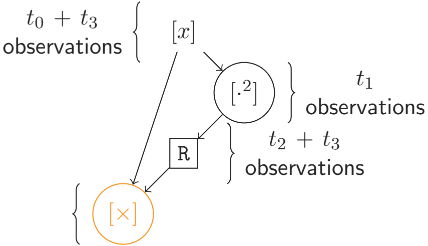
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why This Flaw?

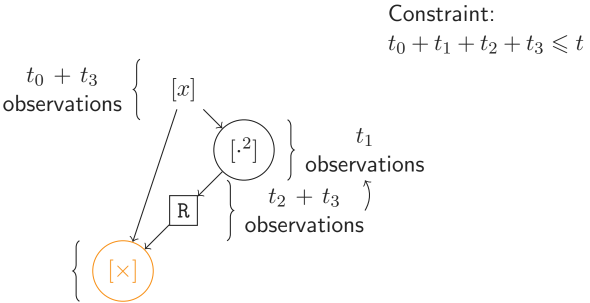
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI) on AES S-box on $\mathrm{GF}(2^8)$

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



Why This Flaw?

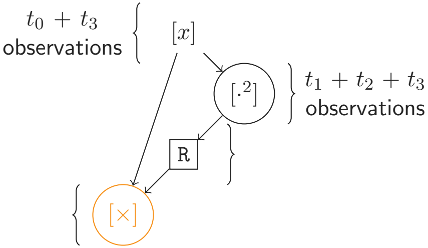
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI) on AES S-box on $\mathrm{GF}(2^8)$



Why This Flaw?

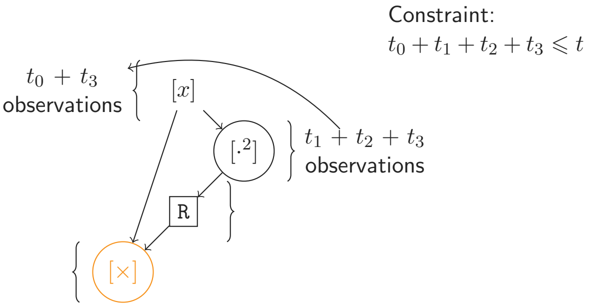
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI) on AES S-box on $\mathrm{GF}(2^8)$

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



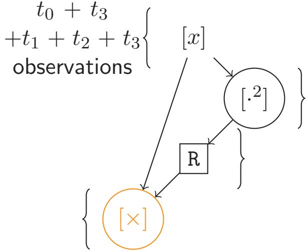
Why This Flaw?

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



Why This Flaw?

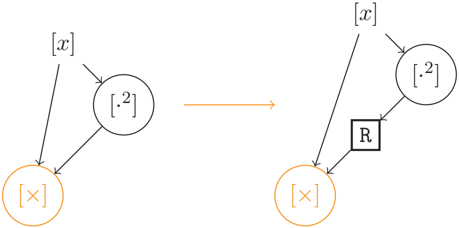
- Rivain and Prouff (CHES 2010): add refresh gadgets (NI) on AES S-box on $\mathrm{GF}(2^8)$



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)

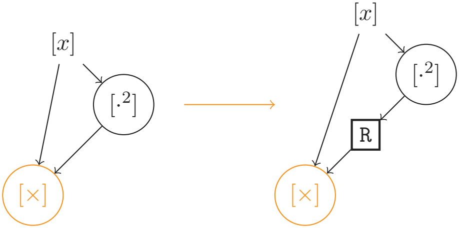


Require: Encoding $[x]$
Ensure: Fresh encoding $[x]$

  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)



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 $\Rightarrow$ t-NI $\Rightarrow$ 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 $\Rightarrow$ not 3-SNI since y_0 is an output variable

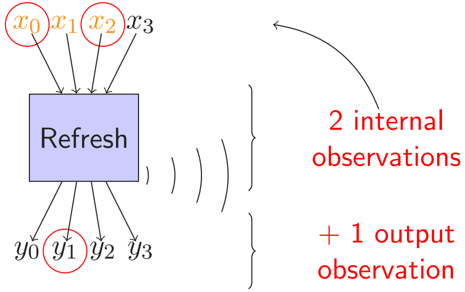
function Ex-t3(x_0, x_1, x_2, x_3, c):

(* $x_0, x_1, x_2 = \$$ *)
(* $x_3 = x + x_0 + x_1 + x_2$ *)

$r_0 \leftarrow \$$
$r_1 \leftarrow \$$
$y_0 \leftarrow x_0 + r_0$
$y_1 \leftarrow x_3 + r_1$
$t_1 \leftarrow x_1 + r_0$
$t_2 \leftarrow (x_1 + r_0) + x_2$
$y_2 \leftarrow (x_1 + r_0 + x_2) + r_1$
$y_3 \leftarrow c + r_1$
return(y_0, y_1, y_2, y_3)

Strong Non-Interference (SNI)

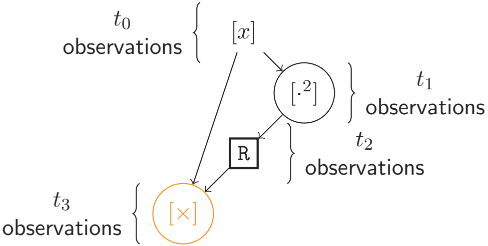
- t-SNI $\Rightarrow$ t-NI $\Rightarrow$ t-probing secure
- a circuit is t-SNI iff any set of t intermediate variables, whose t_1 on the internal variables and t_2 and the outputs, can be perfectly simulated with at most t_1 shares of each input



Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)

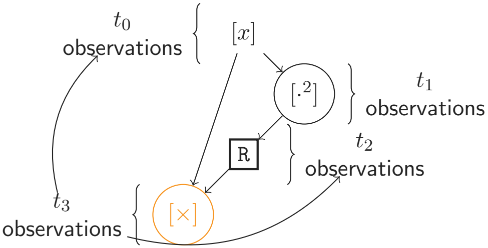
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)

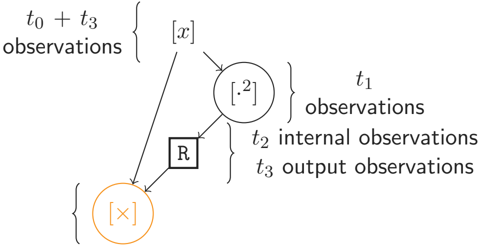
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



Why Does It Works?

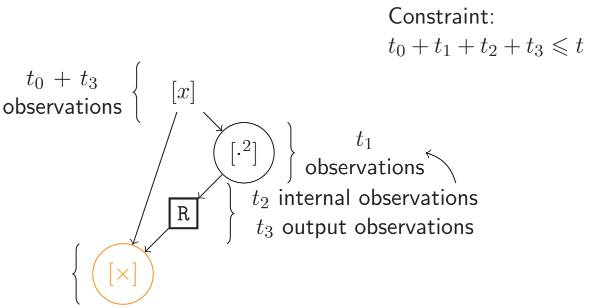
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)

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



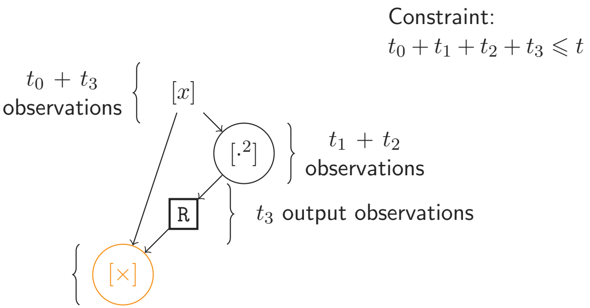
Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)



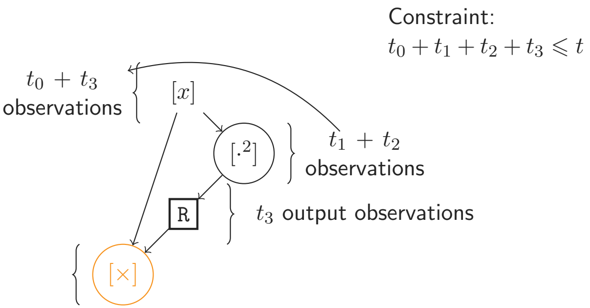
Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)



Why Does It Works?

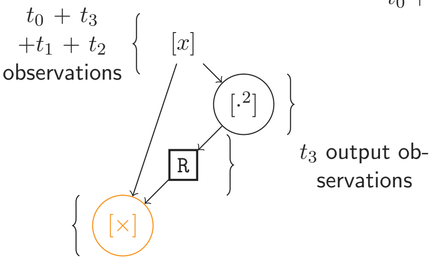
- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)



Why Does It Works?

- Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add stronger refresh gadgets (SNI)

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



Tool `maskComp`

- from t-NI and t-SNI gadgets $\Rightarrow$ build a t-NI circuit by inserting t-SNI regfresh gadgets at carefully chosen locations
- formally proven

Limitations of maskComp

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

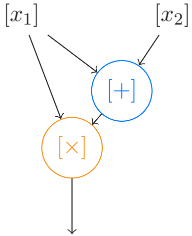


Figure: Circuit 1.

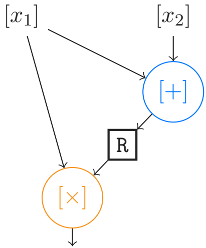


Figure: Circuit 1 after maskComp.

New Proposal

- Joint work with Dahmun Goudarzi and Matthieu Rivain, published at Asiacrypt 2018
- Apply to standard shared circuits:
  - sharewise additions,
  - ISW-multiplications,
  - ISW-refresh gadgets
- Determine exactly whether a standard 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

First Step: Game 0

<u>ExpReal$(\mathcal{A}, C)$:</u>

1: $(\mathcal{P}, x_1, \ldots, x_n) \leftarrow \mathcal{A}()$
2: $[x_1] \leftarrow \mathrm{Enc}(x_1), \ldots, [x_n] \leftarrow \mathrm{Enc}(x_n)$
3: $(v_1, \ldots, v_t) \leftarrow C([x_1], \ldots, [x_n])_{\mathcal{P}}$
4: Return $(v_1, \ldots, v_t)$

<u>ExpSim$(\mathcal{A}, \mathcal{S}, C)$:</u>

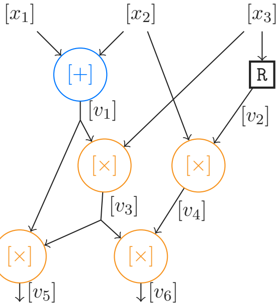
1: $(\mathcal{P}, x_1, \ldots, x_n) \leftarrow \mathcal{A}()$
2: $(v_1, \ldots, v_t) \leftarrow \mathcal{S}(\mathcal{P})$
3: Return $(v_1, \ldots, v_t)$

Figure: t-probing security game.

A shared circuit C is t-probing secure iff $\forall$ $\mathcal{A}$, $\exists$ $\mathcal{S}$ that wins the t-probing security game defined in Figure 3, i.e., the random experiments ExpReal$(\mathcal{A}, C)$ and ExpSim$(\mathcal{A}, \mathcal{S}, C)$ output identical distributions.

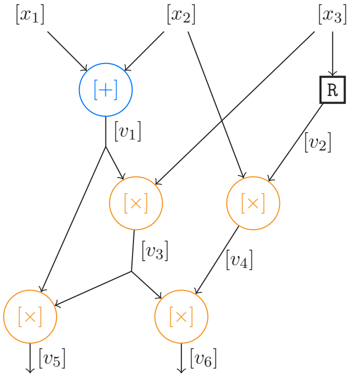
First Step: Game 1

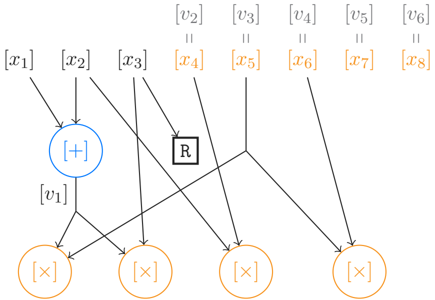
- Probes on multiplication gadgets are replaced by probes on their inputs
- Probes on refresh gadgets are replaced by probes on their input
- Probes on addition gadgets are replaced by probes on their inputs or their output



First Step: Game 2

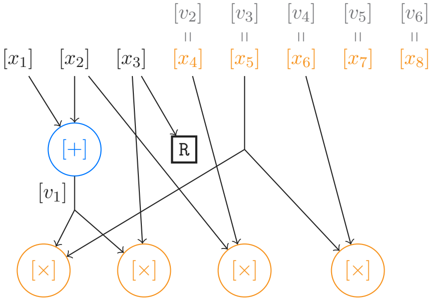
- The tight shared circuit can be replaced by a tight shared circuit of multiplicative depth one with an extended input.





First Step: Game 3

- The attacker is restricted to probes on pairs of multiplication inputs.



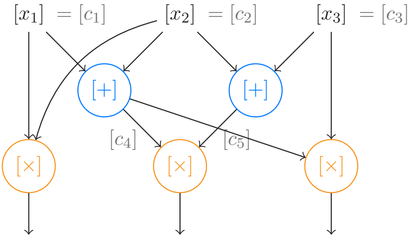
Second Step: Resolution Method

- for each linear combination $[c]$ that is an operand of a multiplication, draw a list of multiplications
  - $\mathcal{G}_1 = \{([c], b_i^1);\ 1 \leq i \leq m_1\}$, let $\mathcal{U}_1 =< b_i^1 >$
  - $\mathcal{G}_2 = \mathcal{G}_1 \cup \{([c] + \mathcal{U}_1, b_i^2);\ 1 \leq i \leq m_2\}$, let $\mathcal{U}_2 = \mathcal{U}_1 \cup < b_i^2 >$
  - $\mathcal{G}_3 = \mathcal{G}_2 \cup \{([c] + \mathcal{U}_2, b_i^3);\ 1 \leq i \leq m_3\}$, let $\mathcal{U}_3 = \mathcal{U}_2 \cup < b_i^3 >$
  - $\dots$
- at each step i,
  - if $[c] \in \mathcal{U}_i$, then stop there is a probing attack on $[c]$
  - if $\mathcal{G}_i = \mathcal{G}_{i-1}$, then stop and consider another combination

Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1], [c_2])$, $([c_4], [c_5])$, and $([c_3], [c_4])$.

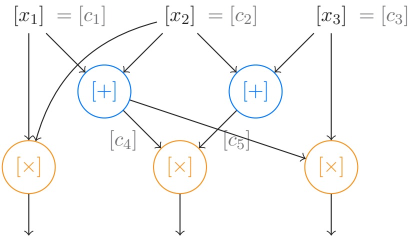
1. Consider $[c_1]$.
   - $\mathcal{G}_1 = ([c_1], [c_2])$ and $\mathcal{U}_1 = [c_2]$



Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1],[c_2])$, $([c_4],[c_5])$, and $([c_3],[c_4])$.

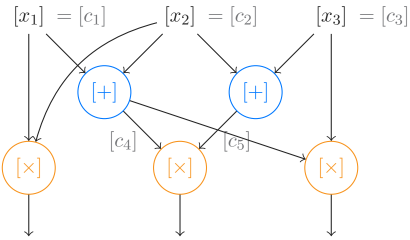
1. Consider $[c_1]$.
   - $\mathcal{G}_1 = ([c_1],[c_2])$ and $\mathcal{U}_1 = [c_2]$
   - $\mathcal{G}_2 = \mathcal{G}_1 \cup \{([c_4],[c_5]),([c_4],[c_3])\}$ since $[c_4] = [c_1] + [c_2]$ and $\mathcal{U}_2 =< [c_2],[c_3],[c_5] >$.



Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1], [c_2])$, $([c_4], [c_5])$, and $([c_3], [c_4])$.

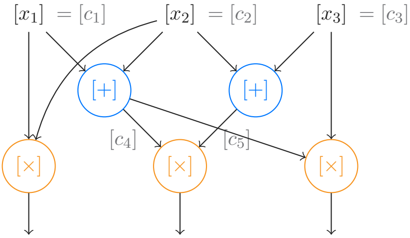
1. Consider $[c_1]$.
   - $\mathcal{G}_1 = ([c_1], [c_2])$ and $\mathcal{U}_1 = [c_2]$
   - $\mathcal{G}_2 = \mathcal{G}_1 \cup \{([c_4], [c_5]), ([c_4], [c_3])\}$ since $[c_4] = [c_1] + [c_2]$ and $\mathcal{U}_2 =< [c_2], [c_3], [c_5] >$.
   - $\mathcal{G}_3 = \mathcal{G}_2$, there is no attack on $[c_1]$.



Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1], [c_2])$, $([c_4], [c_5])$, and $([c_3], [c_4])$.

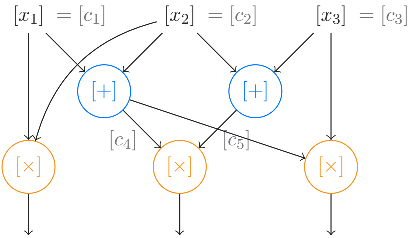
2. Consider $[c_2]$.
   - $\mathcal{G}_1 = ([c_2], [c_1])$ and $\mathcal{U}_1 = [c_1]$



Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1],[c_2])$, $([c_4],[c_5])$, and $([c_3],[c_4])$.

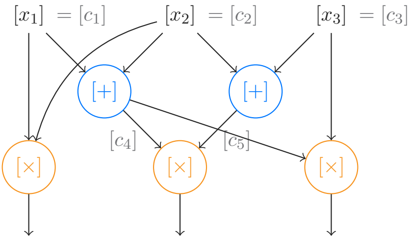
2. Consider $[c_2]$.
   - $\mathcal{G}_1 = ([c_2],[c_1])$ and $\mathcal{U}_1 = [c_1]$
   - $\mathcal{G}_2 = \mathcal{G}_1 \cup \{([c_4],[c_5]),([c_4],[c_3])\}$ since $[c_4] = [c_2] + [c_1]$ and $\mathcal{U}_2 =< [c_1],[c_3],[c_5] >$.



Second Step: Example

- Operands are: $[c_1]$, $[c_2]$, $[c_3]$, $[c_4]$, and $[c_5]$.
- Multiplications are $([c_1],[c_2])$, $([c_4],[c_5])$, and $([c_3],[c_4])$.

2. Consider $[c_2]$.
   - $\mathcal{G}_1 = ([c_2],[c_1])$ and $\mathcal{U}_1 = [c_1]$
   - $\mathcal{G}_2 = \mathcal{G}_1 \cup \{([c_4],[c_5]),([c_4],[c_3])\}$ since $[c_4] = [c_2] + [c_1]$ and $\mathcal{U}_2 =< [c_1],[c_3],[c_5] >$.
   - $[c_2] \in \mathcal{U}_2(=< [c_1],[c_3],[c_5] >)$ since $[c_2] = [c_3] + [c_5]$ so there is an attack!



Second Step: Bitslice AES S-box

- Bitslice implementation from Goudarzi and Rivain
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 32 ISW-refresh gadgets

Second Step: Bitslice AES S-box

- Bitslice implementation from Goudarzi and Rivain
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 32 ISW-refresh gadgets
- `maskComp`
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 32 ISW-refresh gadgets

Second Step: Bitslice AES S-box

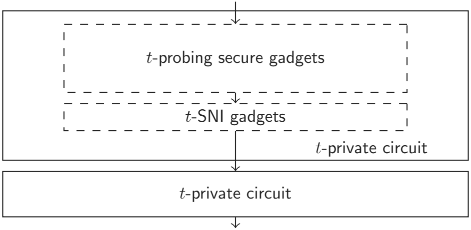
- Bitslice implementation from Goudarzi and Rivain
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 32 ISW-refresh gadgets
- `maskComp`
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 32 ISW-refresh gadgets
- New tool: `tightPROVE`
  - sharewise additions
  - 32 ISW-multiplication gadgets
  - 0 ISW-refresh gadget

Third Step: Extension to Larger Circuits

Proposition. A tight shared circuit $C = C_2 \circ C_1$ composed of two sequential circuits:

- a t-probing secure circuit C_1 whose outputs are all outputs of t-SNI gadgets,
- a t-probing secure circuit C_2 whose inputs are C_1's outputs.

is t-probing secure.

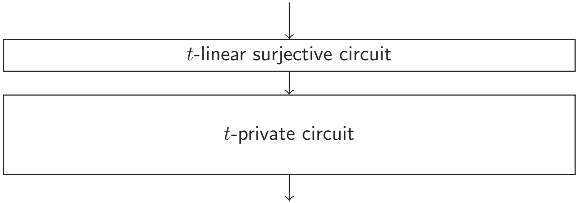


Third Step: Extension to Larger Circuits

Proposition. A tight shared circuit $C = C_2 \circ C_1$ composed of two sequential circuits:

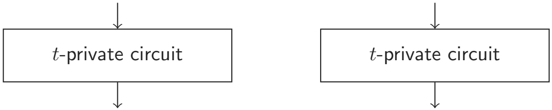
- a t-linear surjective circuit C_1, exclusively composed of sharewise additions,
- a t-probing secure circuit C_2 whose inputs are C_1's outputs.

is t-probing secure.



Third Step: Extension to Larger Circuits

Proposition. A tight shared circuit $C = C_1 \| C_2$ composed of two parallel t-probing secure circuits which operate on independent input sharings is t-probing secure.

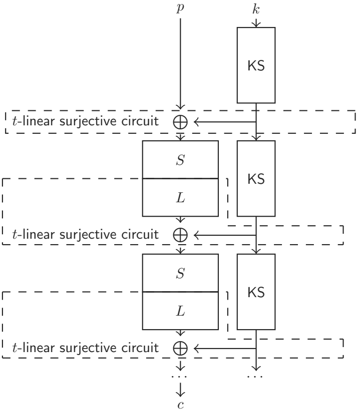


Third Step: SPN Block Ciphers

Proposition. Let C be SPN-block cipher defined as a tight shared circuit. If both conditions

1. S's and KS's outputs are t-SNI gadgets' outputs
2. S and KS are t-probing secure

are fulfilled, then C is t-probing secure.



Conclusion

In a nutshell...

- Method to exactly determine whether or not a tight shared circuit is probing secure for any t
- Significant gain in practice

To continue...

- Extend these results to more general circuits
- Apply this method to reduce randomness on existing applications