Secure Masked Implementations with the Least Refreshing

Sonia Belaïd

March, 18th 2019
1. Introduction

2. Composition of Masked Circuits

3. Improved Composition of Masked Circuits

4. Conclusion
1. Introduction

2. Composition of Masked Circuits

3. Improved Composition of Masked Circuits

4. Conclusion
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 = $ *)

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

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

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 ← $

r_1 ← $

y_0 ← x_0 + r_0
y_1 ← 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)
```
variables: secret, shares, constant
masking order $t = 3$

\begin{center}
\textbf{function} $\text{Ex-t3}(x_0, x_1, x_2, x_3, c)$:
\end{center}

\begin{align*}
(* & x_0, x_1, x_2 = \$ *) \\
(* & x_3 = x + x_0 + x_1 + x_2 *)
\end{align*}

\begin{align*}
& 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{align*}

\text{return}$(y_0, y_1, y_2, y_3)$
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)
```

The function 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

\[ x_0 x_1 x_2 x_3 (= x + x_0 + x_1 + x_2) \]

\[ \text{Ex-t3} \]

\[ y_0, y_1, y_2, y_3 \] 3 observations
1. Introduction

2. Composition of Masked Circuits

3. Improved Composition of Masked Circuits

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

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

\[
\text{for } i = 1 \text{ to } t \text{ do } \\
\quad r \leftarrow \$ \\
\quad x_0 \leftarrow x_0 + r \\
\quad x_i \leftarrow x_i + r \\
\text{end for} \\
\text{return } [x]
\]
First Proposal

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

\[
\begin{align*}
\text{Require:} & \quad \text{Encoding } [x] \\
\text{Ensure:} & \quad \text{Fresh encoding } [x] \\
\text{for } i = 1 \text{ to } t \text{ do} & \\
& \quad r \left\leftarrow \$ \\
& \quad x_0 \left\leftarrow x_0 + r \\
& \quad x_i \left\leftarrow x_i + r \\
\text{end for} & \\
& \quad \text{return } [x]
\end{align*}
\]

$\Rightarrow$ 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 $\mathbb{GF}(2^8)$

Constraint:

$$t_0 + t_1 + t_2 + t_3 \leq t$$
Why This Flaw?

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

Constraint:
\[
t_0 + t_1 + t_2 + t_3 \leq t
\]
Why This Flaw?

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

Constraint:

$$t_0 + t_1 + t_2 + t_3 \leq t$$
Why This Flaw?

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

\[
\begin{align*}
\text{Constraint:} & \quad t_0 + t_1 + t_2 + t_3 \leq t \\
\text{Observations:} & \quad t_0 + t_3 \quad t_1 \quad t_2 + t_3
\end{align*}
\]
Why This Flaw?

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

Constraint:

$t_0 + t_1 + t_2 + t_3 \leq t$
Why This Flaw?

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

Constraint:

\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Why This Flaw?

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

Constraint:

$$t_0 + t_1 + t_2 + t_3 \leq t$$
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
    $r \leftarrow S$
    $x_i \leftarrow x_i + r$
    $x_j \leftarrow x_j + r$
  end for
end for

return $[x]$

⇒ Formal security proof for any order $t$
Second Proposal

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

\[
\begin{align*}
\text{Require:} & \quad \text{Encoding } [x] \\
\text{Ensure:} & \quad \text{Fresh encoding } [x] \\
\text{for } i = 0 \text{ to } t \text{ do} & \\
\quad & \text{for } j = i + 1 \text{ to } t \text{ do} \\
\quad & \quad r \leftarrow $ \\
\quad & \quad x_i \leftarrow x_i + r \\
\quad & \quad x_j \leftarrow x_j + r \\
\quad & \text{end for} \\
\text{end for} \\
\text{return } [x]
\end{align*}
\]

⇒ 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

```
function Ex-t3(x₀, x₁, x₂, x₃, c):
    (* x₀, x₁, x₂ = $ *)
    (* x₃ = x + x₀ + x₁ + x₂ *)
    r₀ ← $
    r₁ ← $
    y₀ ← x₀ + r₀
    y₁ ← x₃ + r₁
    t₁ ← x₁ + r₀
    t₂ ← (x₁ + r₀) + x₂
    y₂ ← (x₁ + r₀ + x₂) + r₁
    y₃ ← c + r₁
    return(y₀, y₁, y₂, y₃)
```

require $x₀$ and $x₁$ to be perfectly simulated $\Rightarrow$ not 3-SNI since $y₀$ is an output variable
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

\[ x_0, x_1, x_2, x_3 \]
\[ y_0, y_1, y_2, y_3 \]

2 internal observations
+ 1 output observation
Why Does It Works?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Why Does It Work?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Why Does It Work?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Why Does It Works?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]

Diagram:
- \( t_0 + t_3 \) observations
- \( [x] \)
- \( [\cdot^2] \)
- \( R \)
- \( t_1 \) observations
- \( t_2 \) internal observations
- \( t_3 \) output observations
Barthe, B., Dupressoir, Fouque, Grégoire, Strub, Zucchini (CCS 2016): add **stronger** refresh gadgets (SNI)

Constraint:

\[
t_0 + t_1 + t_2 + t_3 \leq t
\]
Why Does It Works?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Why Does It Works?

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

Constraint:
\[ t_0 + t_1 + t_2 + t_3 \leq t \]

![Diagram showing the constraint and observations](image)

- Observations: \( t_0 + t_3 + t_1 + t_2 \)
- Output observations: \( t_3 \)
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

Implementation in C language with no countermeasure $\xrightarrow{\text{maskComp}}$ $t$-NI secure implementation in C language
1. Introduction

2. Composition of Masked Circuits

3. Improved Composition of Masked Circuits

4. Conclusion
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

ExpReal($A, C$):

1. $(P, x_1, \ldots, x_n) \leftarrow A()$
2. $[x_1] \leftarrow \text{Enc}(x_1), \ldots, [x_n] \leftarrow \text{Enc}(x_n)$
3. $(v_1, \ldots, v_t) \leftarrow C([x_1], \ldots, [x_n])_P$
4. Return $(v_1, \ldots, v_t)$

ExpSim($A, S, C$):

1. $(P, x_1, \ldots, x_n) \leftarrow A()$
2. $(v_1, \ldots, v_t) \leftarrow S(P)$
3. Return $(v_1, \ldots, v_t)$

**Figure:** $t$-probing security game.

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

\[ x_1 [+] x_2 [+] x_3 [+] R x_1 [+] x_2 [+] x_3 \]

\[ v_1 [+] v_2 [+] v_3 [+] v_4 [+] v_5 [+] v_6 [+] v_7 [+] v_8 \]
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
  - \(G_1 = \{(\lceil c \rceil, b_1^i); 1 \leq i \leq m_1\}\), let \(U_1 = \langle b_1^1 \rangle\)
  - \(G_2 = G_1 \cup \{(\lceil c \rceil + U_1, b_2^i); 1 \leq i \leq m_2\}\), let \(U_2 = U_1 \cup \langle b_2^2 \rangle\)
  - \(G_3 = G_2 \cup \{(\lceil c \rceil + U_2, b_3^i); 1 \leq i \leq m_3\}\), let \(U_3 = U_2 \cup \langle b_3^3 \rangle\)
  - ... 
- at each step \(i\),
  - if \(\lceil c \rceil \in U_i\), then stop there is a probing attack on \(\lceil c \rceil\)
  - if \(G_i = G_{i-1}\), then stop and consider another combination
Second Step: Example

- Operands are: \([c_1], [c_2], [c_3], [c_4], \text{ and } [c_5]\).
- Multiplications are \([(c_1), (c_2)], [(c_4), (c_5)], \text{ and } [(c_3), (c_4)]\).

1. Consider \([c_1]\).
   - \(G_1 = ([c_1], [c_2])\) and \(U_1 = [c_2]\)
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 \([c_1]\).
   - \(G_1 = ([c_1], [c_2])\) and \(U_1 = [c_2]\)
   - \(G_2 = G_1 \cup \{([c_4], [c_5]), ([c_4], [c_3])\}\) since \([c_4] = [c_1] + [c_2]\) and \(U_2 = <[c_2], [c_3], [c_5]>\).
Second Step: Example

- Operands are: \([c_1], [c_2], [c_3], [c_4], \text{ and } [c_5]\).
- Multiplications are \(([c_1], [c_2]), ([c_4], [c_5]), \text{ and } ([c_3], [c_4])\).

1. Consider \([c_1]\).
   - \(G_1 = ([c_1], [c_2])\) and \(U_1 = [c_2]\)
   - \(G_2 = G_1 \cup \{([c_4], [c_5]), ([c_4], [c_3])\}\) since \([c_4] = [c_1] + [c_2]\) and \(U_2 = < [c_2], [c_3], [c_5] >\).
   - \(G_3 = G_2\), there is no attack on \([c_1]\).

\[
\begin{align*}
[x_1] &= [c_1] \\
[x_2] &= [c_2] \\
[x_3] &= [c_3]
\end{align*}
\]
Second Step: Example

- Operands are: \([c_1], \,[c_2], \,[c_3], \,[c_4], \text{ and } [c_5]\).
- Multiplications are \(([c_1], [c_2]), ([c_4], [c_5]), \text{ and } ([c_3], [c_4])\).

2. Consider \([c_2]\).
   
   $G_1 = ([c_2], [c_1]) \text{ and } U_1 = [c_1]$
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 \([c_2]\).
   - \(G_1 = ([c_2], [c_1])\) and \(U_1 = [c_1]\)
   - \(G_2 = G_1 \cup \{([c_4], [c_5]), ([c_4], [c_3])\}\) since \([c_4] = [c_2] + [c_1]\) and \(U_2 = < [c_1], [c_3], [c_5] >\).

\[
\begin{align*}
[x_1] &= [c_1] \\
[x_2] &= [c_2] \\
[x_3] &= [c_3]
\end{align*}
\]
Second Step: Example

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

2. Consider $[c_2]$.
   - $G_1 = ([c_2], [c_1])$ and $U_1 = [c_1]$
   - $G_2 = G_1 \cup \{([c_4], [c_5]), ([c_4], [c_3])\}$ since $[c_4] = [c_2] + [c_1]$ and $U_2 = \langle [c_1], [c_3], [c_5] \rangle$.
   - $[c_2] \in U_2(=\langle [c_1], [c_3], [c_5] \rangle)$ 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.

![Diagram of circuits]

\( t \)-probing secure gadgets

\( t \)-SNI gadgets

\( t \)-private circuit
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.
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.
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.
1. Introduction

2. Composition of Masked Circuits

3. Improved Composition of Masked Circuits

4. Conclusion
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