On the use of formal tools to improve the security of masked implementations
Symposium European Cyber Week

November 23, 2016

Sonia Belaïd
Cryptanalysis

→ Black-box cryptanalysis
→ Side-channel analysis

Alice $m_i \rightarrow k \rightarrow ENC \rightarrow c_i \rightarrow Bob$ $c_i \rightarrow k \rightarrow DEC \rightarrow m_i$
Cryptanalysis

- Black-box cryptanalysis: $\mathcal{A} \leftarrow (m_i, c_i)$
- Side-Channel Analysis
Cryptanalysis

→ Black-box cryptanalysis

→ Side-Channel Analysis: $A \leftarrow (m_i, c_i, \mathcal{L}_i)$
Cryptanalysis

→ Black-box cryptanalysis

→ Side-Channel Analysis: $\mathcal{A} \leftarrow (m_i, c_i, \mathcal{L}_i)$
Cryptanalysis

→ Black-box cryptanalysis

→ Side-Channel Analysis: $A \leftarrow (m_i, c_i, L_i)$
Cryptanalysis

- Black-box cryptanalysis
- Side-Channel Analysis: $A \leftarrow (m_i, c_i, L_i)$
Cryptanalysis

- Black-box cryptanalysis
- Side-Channel Analysis: $\mathcal{A} \leftarrow (m_i, c_i, L_i)$
A power-analysis attack against AES-128
A power-analysis attack against AES-128
A power-analysis attack against AES-128
Algorithmic Countermeasures

Problem: leakage $L$ is key-dependent

Fresh Re-keying

Idea: regularly change $k$

- master key $k$
- session key $k^*$

Masking

Idea: make leakage $L$ random

- sensitive value: $v = f(m, k)$
- $v_0 \leftarrow v \oplus (\bigoplus_{1 \leq i \leq t} v_i)$
- $v_1 \leftarrow \$
- $\ldots$
- $v_t \leftarrow \$

$\rightarrow$ each $t$-uple of $v_i$ is independent from $v$
Algorithmic Countermeasures

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

**Masking**

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

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

- $v_0 \leftarrow v \oplus \left( \bigoplus_{1 \leq i \leq t} v_i \right)$
- $v_1 \leftarrow \$
- $\ldots$
- $v_t \leftarrow \$

$\rightarrow$ each $t$-uple of $v_i$ is independent from $v$
Security of Masked Programs: Leakage Model

- **t-probing model**
  - Ishai, Sahai, Wagner
  - Crypto 03

- **no leak-free gates**

- **reduction**
  - Duc, Dziembowski, Faust
  - Eurocrypt 14

- **noisy leakage model**
  - Prouff, Rivain
  - Eurocrypt 13
Security in the $t$-probing model

$t$-probing model assumptions:
- only one variable is leaking at a time
- the attacker can get the exact value of at most $t$ variables

Secure if all the $t$-uples are independent from the secret.
Security in the $t$-probing model

- $\nu$: randomly generated variable
- $c$: known constant
- $x$: secret variable

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

\[
\begin{align*}
(* x_1, x_2, x_3 &= \$ *) \\
(* x_4 &= x + x_1 + x_2 + x_3 *) \\
 r_1 &\leftarrow \$
\end{align*}
\]

\[
\begin{align*}
 r_2 &\leftarrow \$
 y_1 &\leftarrow x_1 + r_1 \\
 y_2 &\leftarrow (x + x_1 + x_2 + x_3) + r_2 \\
 t_1 &\leftarrow x_2 + r_1 \\
 t_2 &\leftarrow (x_2 + r_1) + x_3 \\
 y_3 &\leftarrow (x_2 + r_1 + x_3) + r_2 \\
 y_4 &\leftarrow c + r_2 \\
 \text{return}(y_1, y_2, y_3, y_4)
\end{align*}
\]
Security in the \(t\)-probing model

- \(v\): randomly generated variable
- \(c\): known constant
- \(x\): secret variable

\[
\text{function } \text{Ex-t3}(x_1, x_2, x_3, x_4, c): \\
(* x_1, x_2, x_3 = *) \\
(* x_4 = x + x_1 + x_2 + x_3 *)
\]

\[
\begin{align*}
  r_1 & \leftarrow \$ \\
  r_2 & \leftarrow \$ \\
  y_1 & \leftarrow x_1 + r_1 \\
  y_2 & \leftarrow (x + x_1 + x_2 + x_3) + r_2 \\
  t_1 & \leftarrow x_2 + r_1 \\
  t_2 & \leftarrow (x_2 + r_1) + x_3 \\
  y_3 & \leftarrow (x_2 + r_1 + x_3) + r_2 \\
  y_4 & \leftarrow c + r_2 \\
\end{align*}
\]

1. independent from the secret?
Security in the $t$-probing model

- $v$: randomly generated variable
- $c$: known constant
- $x$: secret variable

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

\[ (* x_1, x_2, x_3 = $ *) \]
\[ (* x_4 = x + x_1 + x_2 + x_3 * ) \]

\[ r_1 \leftarrow $ \]
\[ r_2 \leftarrow $ \]

1. independent from the secret? 

$y_1 \leftarrow x_1 + r_1$

\[ y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2 \]

\[ t_1 \leftarrow x_2 + r_1 \]

\[ t_2 \leftarrow (x_2 + r_1) + x_3 \]

\[ y_3 \leftarrow (x_2 + r_1 + x_3) + r_2 \]

\[ y_4 \leftarrow c + r_2 \]

return($y_1, y_2, y_3, y_4$)
Security in the $t$-probing model

- $v$: randomly generated variable
- $c$: known constant
- $x$: secret variable

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

\[
\begin{align*}
(* x_1, x_2, x_3 &= \$ *) \\
(* x_4 &= x + x_1 + x_2 + x_3 *) \\

r_1 &\leftarrow \$ \\
r_2 &\leftarrow \$ \\
\end{align*}
\]

1. independent from the secret?

\[
\begin{align*}
y_1 &\leftarrow x_1 + r_1 \\
y_2 &\leftarrow (x + x_1 + x_2 + x_3) + r_2 \\
t_1 &\leftarrow x_2 + r_1 \\
t_2 &\leftarrow (x_2 + r_1) + x_3 \\
y_3 &\leftarrow (x_2 + r_1 + x_3) + r_2 \\
y_4 &\leftarrow c + r_2 \\
\end{align*}
\]

return($y_1, y_2, y_3, y_4$)
Security in the \( t \)-probing model

- \( v \): randomly generated variable
- \( c \): known constant
- \( x \): secret variable

function \( \text{Ex-t3}(x_1, x_2, x_3, x_4, c) \):

\[
\begin{align*}
(* x_1, x_2, x_3 &= \$ *) \\
(* x_4 &= x + x_1 + x_2 + x_3 *) \\
[r_1 &\leftarrow \$] \\
[r_2 &\leftarrow \$] \\
[y_1 &\leftarrow x_1 + r_1] \\
[y_2 &\leftarrow (x + x_1 + x_2 + x_3) + r_2] \\
[t_1 &\leftarrow x_2 + r_1] \\
[t_2 &\leftarrow (x_2 + r_1) + x_3] \\
[y_3 &\leftarrow (x_2 + r_1 + x_3) + r_2] \\
[y_4 &\leftarrow c + r_2] \\
\text{return}(y_1, y_2, y_3, y_4)
\end{align*}
\]

1. independent from the secret?

- \( \times \) many mistakes

many mistakes
Security in the $t$-probing model

- $v$: randomly generated variable
- $c$: known constant
- $x$: secret variable

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

1. independent from the secret?

2. test 286 3-uples
   - $\times$ missing cases
   - $\times$ inefficient

\[
\begin{align*}
\text{r}_1 &\leftarrow \$_{} \\
\text{r}_2 &\leftarrow \$_{} \\
\text{y}_1 &\leftarrow \text{x}_1 + \text{r}_1 \\
\text{y}_2 &\leftarrow (\text{x} + \text{x}_1 + \text{x}_2 + \text{x}_3) + \text{r}_2 \\
\text{t}_1 &\leftarrow \text{x}_2 + \text{r}_1 \\
\text{t}_2 &\leftarrow (\text{x}_2 + \text{r}_1) + \text{x}_3 \\
\text{y}_3 &\leftarrow (\text{x}_2 + \text{r}_1 + \text{x}_3) + \text{r}_2 \\
\text{y}_4 &\leftarrow c + \text{r}_2 \\
\text{return}(\text{y}_1, \text{y}_2, \text{y}_3, \text{y}_4)
\end{align*}
\]
Security in the $t$-probing model

Contributions:

1. new algorithm to decide whether a $t$-uple is independent from the secret
   - no false positive
   - more efficient than existing works

2. new algorithm to enumerate all the $t$-uples
   - more efficient than existing works

1. Show that a $t$-uple is independent from the secret

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

(Rule 1) secret variables?

yes $\rightarrow$ (Rule 2)
no $\rightarrow$ ✓

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

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

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

yes $\rightarrow$ simplify; $b \leftarrow \text{false}$; (Rule 1)
no $\rightarrow$ x

✓ $\rightarrow$ distribution independent from the secret
x $\rightarrow$ might be used for an attack

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

$r_1 \leftarrow$
$r_2 \leftarrow$
$y_1 \leftarrow x_1 + r_1$
$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$
$t_1 \leftarrow x_2 + r_1$
$t_2 \leftarrow (x_2 + r_1) + x_3$
$y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$
$y_4 \leftarrow c + r_2$

return $(y_1, y_2, y_3, y_4)$
1. Show that a $t$-uple is independent from the secret inputs: $t$ intermediate variables, $b \leftarrow \text{true}$

(Rule 1) secret variables?
- yes $\rightarrow$ (Rule 2)
- no $\rightarrow$ ✓

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

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

✓ $\rightarrow$ distribution independent from the secret
x $\rightarrow$ might be used for an attack

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($y_1, y_2, y_3, y_4$)
1. Show that a $t$-uple is independent from the secret

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

(Rule 1) secret variables?
  yes $\rightarrow$ (Rule 2)
  no $\rightarrow$ ✓

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

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

✓ $\rightarrow$ distribution independent from the secret
✗ $\rightarrow$ might be used for an attack

function $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

\[
\begin{align*}
  r_1 &\leftarrow $ \\
  r_2 &\leftarrow $ \\
  y_1 &\leftarrow x_1 + r_1 \\
  y_2 &\leftarrow (x + x_1 + x_2 + x_3) + r_2 \\
  t_1 &\leftarrow x_2 + r_1 \\
  t_2 &\leftarrow (x_2 + r_1) + x_3 \\
  y_3 &\leftarrow (x_2 + r_1 + x_3) + r_2 \\
  y_4 &\leftarrow c + r_2
\end{align*}
\]

return $(y_1, y_2, y_3, y_4)$
1. Show that a \( t \)-uple is independent from the secret

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

(Rule 1) secret variables?
  yes \( \rightarrow \) (Rule 2)
  no \( \rightarrow \checkmark \)

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

(Rule 3) is flag \( b = \text{true} \)?
  yes \( \rightarrow \) simplify; \( b \leftarrow \text{false} \); (Rule 1)
  no \( \rightarrow \times \)

\( \checkmark \) \( \rightarrow \) distribution independent from the secret
\( \times \) \( \rightarrow \) might be used for an attack

function Ex-t3\((x_1, x_2, x_3, x_4, c)\):
  \( r_1 \leftarrow x_2 + r_1 \)
  \( r_2 \leftarrow x_3 \)
  \( y_1 \leftarrow r_1 + r_2 \)
  \( y_2 \leftarrow c + r_2 \)
  \( t_1 \leftarrow r_1 + r_2 \)
  \( t_2 \leftarrow (x_2 + r_1) + x_3 \)
  \( y_3 \leftarrow (x_2 + r_1 + x_3) + r_2 \)
  return\((y_1, y_2, y_3, y_4)\)
2. Extension to All Possible Sets

Problem: $n$ intermediate variables $\rightarrow \binom{n}{t}$ proofs
2. Extension to All Possible Sets

Problem: \( n \) intermediate variables \( \mapsto \binom{n}{t} \) 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
2. Extension to All Possible Sets

Problem: \( n \) intermediate variables \( \rightarrow \binom{n}{t} \) 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 \( X = (t \text{ variables}) \) and prove its independence
2. extend \( X \) to \( \hat{X} \) with more observations but still independence
3. recursively descend in set \( C(\hat{X}) \)
4. merge \( \hat{X} \) and \( C(\hat{X}) \) once they are processed separately.
2. Extension to All Possible Sets

Problem: \( n \) intermediate variables \( \rightarrow \binom{n}{t} \) 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 \( X = (t \text{ variables}) \) and prove its independence
2. Extension to All Possible Sets

Problem: \( n \) intermediate variables \( \rightarrow \binom{n}{t} \) 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 \( X = (t \text{ variables}) \) and prove its independence
2. extend \( X \) to \( \hat{X} \) with more observations but still independence
2. Extension to All Possible Sets

Problem: $n$ intermediate variables $\Rightarrow \binom{n}{t}$ 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 $\hat{X}$ with more observations but still independence
3. recursively descend in set $\mathcal{C}(\hat{X})$
2. Extension to All Possible Sets

Problem: \( n \) intermediate variables \( \Rightarrow \binom{n}{t} \) 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 \( X = (t \text{ variables}) \) and prove its independence
2. extend \( X \) to \( \hat{X} \) with more observations but still independence
3. recursively descend in set \( \mathcal{C}(\hat{X}) \)
4. merge \( \hat{X} \) and \( \mathcal{C}(\hat{X}) \) once they are processed separately.
Benchmarks

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

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

A refresh algorithm takes as input a sharing \((x_i)_{i\geq 0}\) of \(x\) and returns a new sharing \((x'_i)_{i\geq 0}\) of \(x\) such that \((x_i)_{i\geq 0}\) and \((x'_i)_{i\geq 0}\) are mutually independent.
Current Issues in Composition

A refresh algorithm takes as input a sharing \((x_i)_{i \geq 0}\) of \(x\) and returns a new sharing \((x'_i)_{i \geq 0}\) of \(x\) such that \((x_i)_{i \geq 0}\) and \((x'_i)_{i \geq 0}\) are mutually independent.
A refresh algorithm takes as input a sharing \((x_i)_{i \geq 0}\) of \(x\) and returns a new sharing \((x'_i)_{i \geq 0}\) of \(x\) such that \((x_i)_{i \geq 1}\) and \((x'_i)_{i \geq 1}\) are mutually independent.
A refresh algorithm takes as input a sharing \((x_i)_{i \geq 0}\) of \(x\) and returns a new sharing \((x'_i)_{i \geq 0}\) of \(x\) such that \((x_i)_{i \geq 1}\) and \((x'_i)_{i \geq 1}\) are mutually independent.
A refresh algorithm takes as input a sharing \((x_i)_{i \geq 0}\) of \(x\) and returns a new sharing \((x'_i)_{i \geq 0}\) of \(x\) such that \((x_i)_{i \geq 1}\) and \((x'_i)_{i \geq 1}\) are mutually independent.
Composition in the $t$-probing model

Contributions:

1. new algorithm to verify the security of compositions
   ▶ formal security
   ▶ any order
2. compiler to build a higher-order secure from any C implementation
   ▶ efficient
   ▶ any order

Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rebecca Zucchini.

Strong Non-Interference and Type-Directed Higher-Order Masking. CCS 2016.
Security properties in the $t$-probing model

if $t$ is fixed: show that any set of $t$ intermediate variables is independent from the secret

straightforward for linear functions
formal proofs with EasyCrypt and pen-and-paper proofs for small non-linear functions
Security properties in the $t$-probing model

if $t$ is fixed: show that any set of $t$ intermediate variables is independent from the secret

if $t$ is not fixed: show that any set of $t$ intermediate variables can be simulated with at most $t$ shares of each input
Security properties in the $t$-probing model

if $t$ is fixed: show that any set of $t$ intermediate variables is independent from the secret

if $t$ is not fixed: show that any set of $t$ intermediate variables can be simulated with at most $t$ shares of each input

function Linear-function-$t(a_0,\ldots,a_i,\ldots a_t)$:

for $i = 0$ to $t$

$c_i \leftarrow f(a_i)$

return $(c_0,\ldots,c_i,\ldots,c_t)$

→ straightforward for linear functions

$\begin{array}{c}
a_0 \quad a_1 \quad a_2 \quad a_3 \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
(= a + a_0 + a_1 + a_2) \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
c_0 \quad c_1 \quad c_2 \quad c_3 \\
\end{array}$

3 observations
Security properties in the $t$-probing model

if $t$ is fixed: show that any set of $t$ intermediate variables is independent from the secret

if $t$ is not fixed: show that any set of $t$ intermediate variables can be simulated with at most $t$ shares of each input

function $\text{Linear-function-}t(a_0, \ldots, a_i, \ldots, a_t)$:

for $i = 0$ to $t$

\[
\begin{align*}
    c_i &= f(a_i) \\
\end{align*}
\]

return $(c_0, \ldots, c_i, \ldots, c_t)$

$\rightarrow$ straightforward for linear functions

\[ (= a + a_0 + a_1 + a_2 ) \]

3 observations
Security properties in the $t$-probing model

if $t$ is fixed: show that any set of $t$ intermediate variables is independent from the secret

if $t$ is not fixed: show that any set of $t$ intermediate variables can be simulated with at most $t$ shares of each input

function $\text{Linear-function-t}(a_0, \ldots, a_i, \ldots, a_t)$:

for $i = 0$ to $t$

\[ c_i = f(a_i) \]

return $(c_0, \ldots, c_i, \ldots, c_t)$

→ straightforward for linear functions

→ formal proofs with EasyCrypt and pen-and-paper proofs for small non-linear functions
Current Issues

Constraint:

\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Current Issues

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

Constraint:

\[ t_0 + t_1 + t_2 + t_3 \leq t \]
Current Issues

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

$t_0$ observations

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

$t_1 + t_3 + t_2 + t_3$ observations
Current Issues

$t_0$ observations

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

$t_1 + t_2 + 2t_3 \leq t$?

observations
Current Issues

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

Constraint:

- \( t_0 \): observations
- \( t_1 + t_2 + 2t_3 \leq t? \) observations
Current Issues

\[ t_0 + t_1 + t_2 + t_3 + t_r \leq t \]
Current Issues

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

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

- \( t_0 \) observations
- \( t_2 + t_3 \) observations
- \( t_1 \) observations
- \( t_r + t_3 \) observations
Current Issues

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

\( A_0 \) observations

\( A_1 \) observations

\( A_2 \) observations

\( A_3 \) observations

Flow from \( t_0 \) to \( t_1 \) to \( t_r \) to \( t_3 \) to \( t_0 \)

\[ t_0 + t_2 + t_3 \] observations

\[ t_1 + t_r + t_3 \] observations
Current Issues

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

\[ t_0 \text{ observations} \]
\[ t_1 + t_2 + 2t_3 + t_r \leq t? \]

\[ t_1 + t_2 + 2t_3 + t_r \leq t? \text{ observations} \]
Stronger security property for Refresh

**Strong** Non-Interference in the $t$-probing model:

if $t$ is not fixed: show that any set of $t$ intermediate variables with
- $t_1$ on internal variables
- $t_2 = t - t_1$ on the outputs
can be simulated with at most $t_1$ shares of each input
Secure Composition

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

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

Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leq t$

$t_0$ observations

$t_2 + t_3$ observations

$t_1$ observations

$t_r$ internal observations
$+ t_3$ output observations
Secure Composition

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

$t_0$ observations

Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leq t$

$t_1 + t_2 + t_3 + t_r$ observations

$t_3$ output observations
Secure Composition

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

- \( A_0 \) observations:
  - \( t_0 \) observations

- \( A_1 \) observations:
  - \( t_1 + t_2 + t_3 + t_r \) internal observations

- \( A_2 \) observations

- \( A_3 \) observations:
  - \( t_3 \) output observations
Secure Composition

$t_0 + t_1 + t_2 + t_3 + t_r$

Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leq t$

t_3 output observations

\[ A_0 \]
\[ A_1 \]
\[ A_2 \]
\[ A_3 \]
Secure Composition

$\sum_{i=0}^{3} t_i + t_r \leq t$

Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leq t$

$t_3$ output observations
Secure Composition

Automatic tool for C-based algorithms

- unprotected algorithm → higher-order masked algorithm
- example for AES S-box
Secure Composition

Automatic tool for C-based algorithms

- unprotected algorithm $\rightarrow$ higher-order masked algorithm
- example for AES S-box

\[ x \cdot 2 \otimes x \cdot 2 \otimes x \otimes \]
Secure Composition

Automatic tool for C-based algorithms

- unprotected algorithm → higher-order masked algorithm
- example for AES S-box

\[ x \cdot 2 \otimes x \cdot 2 \otimes x \]
Secure Composition

Automatic tool for C-based algorithms

- unprotected algorithm ➔ higher-order masked algorithm
- example for AES S-box

\[ x \cdot 2 \otimes x \cdot 2 \otimes x \]

\[ X \]
Some Results

Resource usage statistics for generating masked algorithms (at any order) from some unmasked implementations\(^1\)

<table>
<thead>
<tr>
<th>Scheme</th>
<th># Refresh</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES ($\circ$)</td>
<td>2</td>
<td>0.09s</td>
<td>4Mo</td>
</tr>
<tr>
<td>AES ($x \circ g(x)$)</td>
<td>0</td>
<td>0.05s</td>
<td>4Mo</td>
</tr>
<tr>
<td>Keccak with Refresh</td>
<td>0</td>
<td>121.20s</td>
<td>456Mo</td>
</tr>
<tr>
<td>Keccak</td>
<td>600</td>
<td>2728.00s</td>
<td>22870Mo</td>
</tr>
<tr>
<td>Simon</td>
<td>67</td>
<td>0.38s</td>
<td>15Mo</td>
</tr>
<tr>
<td>Speck</td>
<td>61</td>
<td>6.22s</td>
<td>38Mo</td>
</tr>
</tbody>
</table>

\(^1\)On a Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz with 64Go of memory running Linux (Fedora)
Some Results

Resource usage statistics for generating masked algorithms (at any order) from some unmasked implementations\(^1\)

<table>
<thead>
<tr>
<th>Scheme</th>
<th># Refresh</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES ((\odot))</td>
<td>2 per S-box</td>
<td>0.09s</td>
<td>4Mo</td>
</tr>
<tr>
<td>AES ((x \odot g(x)))</td>
<td>0</td>
<td>0.05s</td>
<td>4Mo</td>
</tr>
<tr>
<td>Keccak with Refresh</td>
<td>0</td>
<td>121.20s</td>
<td>456Mo</td>
</tr>
<tr>
<td>Keccak</td>
<td>600</td>
<td>2728.00s</td>
<td>22870Mo</td>
</tr>
<tr>
<td>Simon</td>
<td>67</td>
<td>0.38s</td>
<td>15Mo</td>
</tr>
<tr>
<td>Speck</td>
<td>61</td>
<td>6.22s</td>
<td>38Mo</td>
</tr>
</tbody>
</table>

---

\(^1\) On a Intel(R) Xeon(R) CPU E5-2667 0 @ 2.90GHz with 64Go of memory running Linux (Fedora)
Conclusion

Summary

✔ verification of higher-order masking schemes
✔ efficient and proven composition
✔ two automatic tools

Further Work

→ extend the verification to higher orders using composition
→ integrate transition/glitch-based model
→ build practical experiments for both attacks and new countermeasures
Conclusion

Cryptanalysis: Power-Analysis Attacks

- investigate the LPN algorithms in the context of power-analysis attacks
- analyze the operation modes

Cryptography: countermeasures against Power-Analysis Attacks

- implement and evaluate our countermeasures on real devices (software and hardware)
- make verifications and compositions as practical as possible
- use the characterization of a device as a leakage model