THALES

On the Use of Masking to Defeat Power-Analysis Attacks

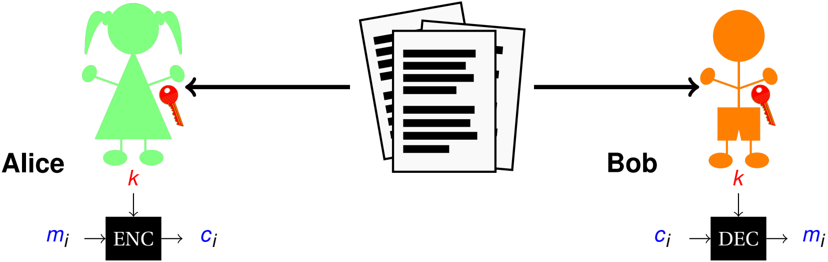
ENS Paris Crypto Day

February 16, 2016

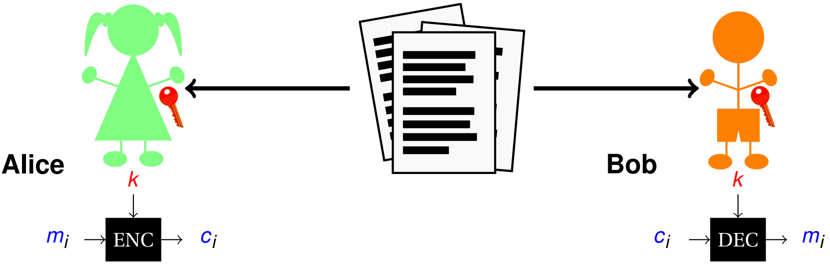
Presented by Sonia Belaïd

Outline

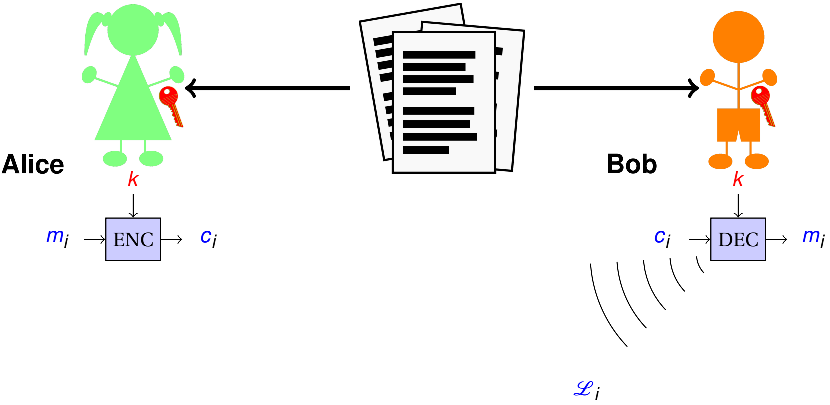
- Black-box cryptanalysis
- Side-channel analysis



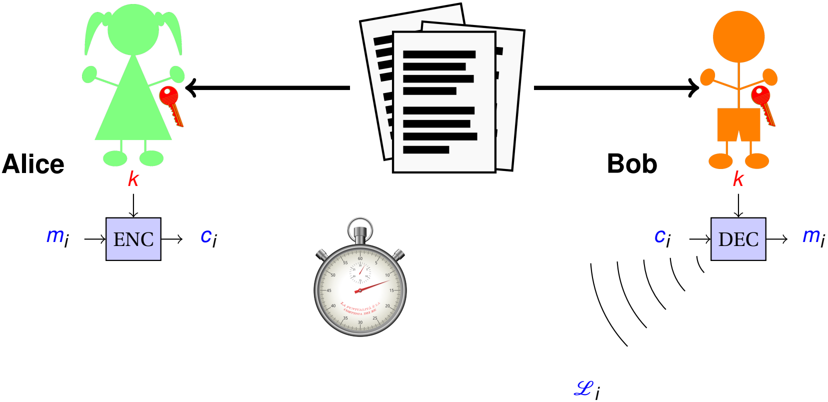
- Black-box cryptanalysis: $\mathscr{A} \leftarrow (m_i, c_i)$
- Side-Channel Analysis



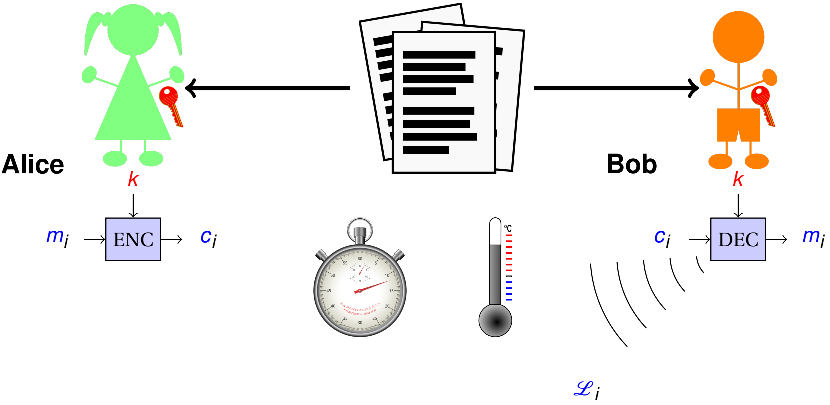
- Black-box cryptanalysis
- Side-Channel Analysis: $\mathscr{A} \leftarrow (m_i, c_i, \mathscr{L}_i)$



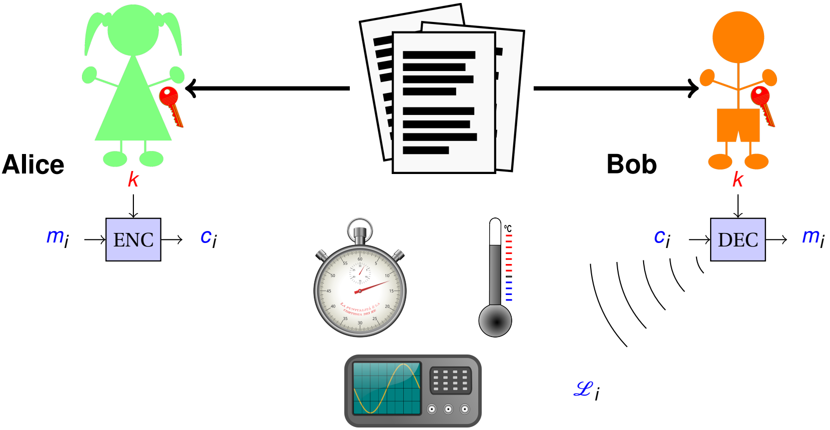
- ➜ Black-box cryptanalysis
- ➜ Side-Channel Analysis: $\mathscr{A} \leftarrow (m_i, c_i, \mathscr{L}_i)$



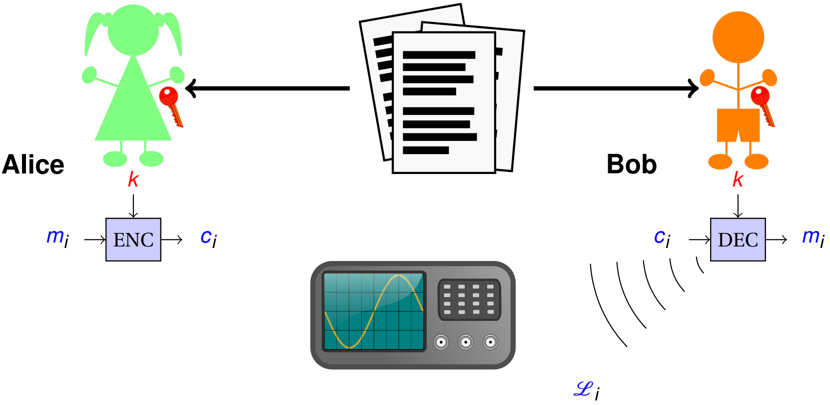
- ➜ Black-box cryptanalysis
- ➜ Side-Channel Analysis: $\mathscr{A} \leftarrow (m_i, c_i, \mathscr{L}_i)$



- Black-box cryptanalysis
- Side-Channel Analysis: $\mathscr{A} \leftarrow (m_i, c_i, \mathscr{L}_i)$



- Black-box cryptanalysis
- Side-Channel Analysis: $\mathscr{A} \leftarrow (m_i, c_i, \mathscr{L}_i)$



A Power-Analysis Attack against AES-128

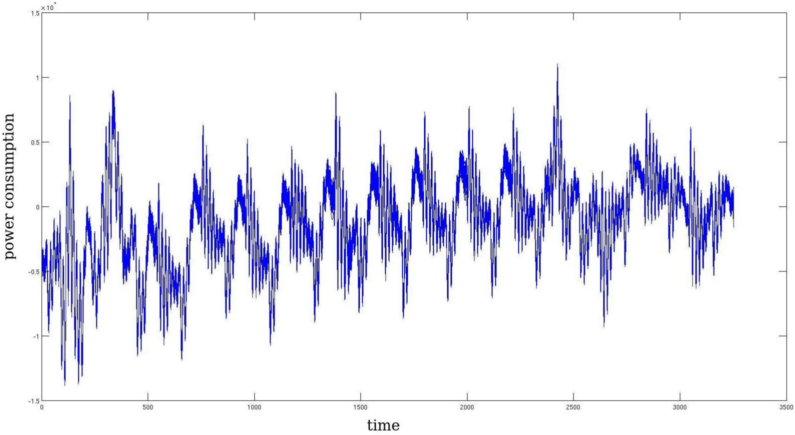


Figure : Consumption trace of a full AES-128 from the DPA Contest v2

A Power-Analysis Attack against AES-128

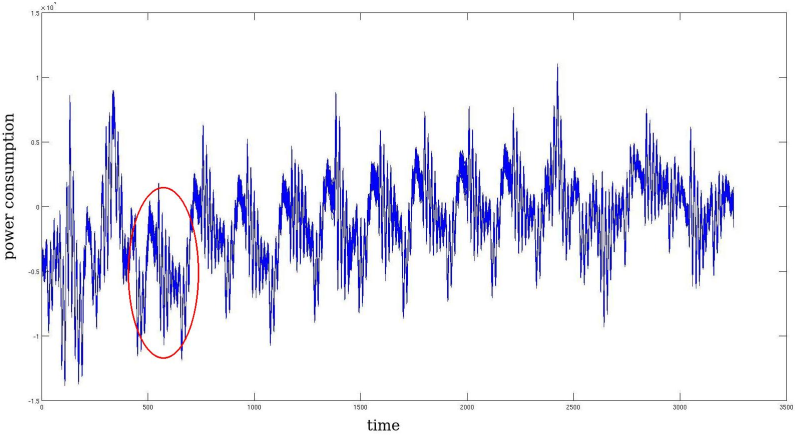
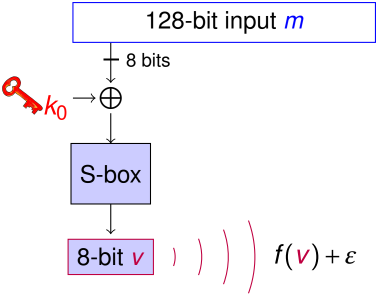
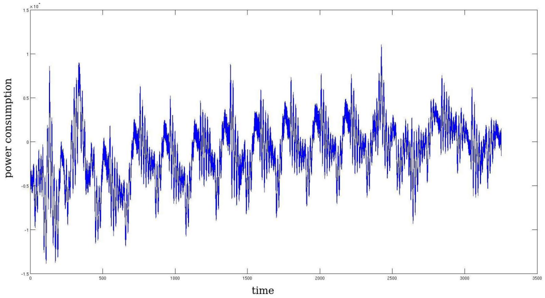


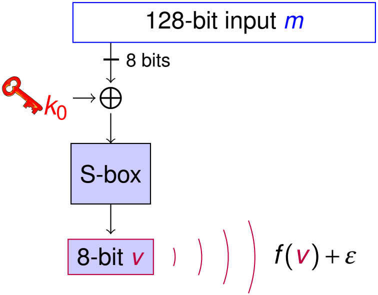
Figure : Consumption trace of a full AES-128 from the DPA Contest v2

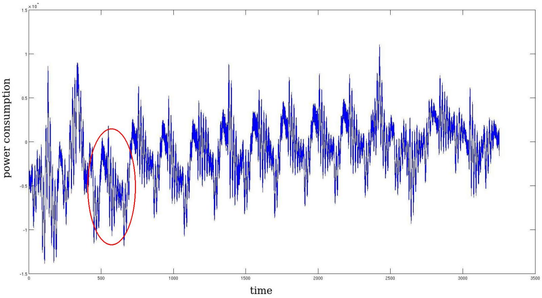
A Power-Analysis Attack against AES-128



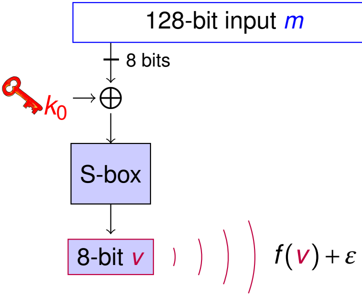


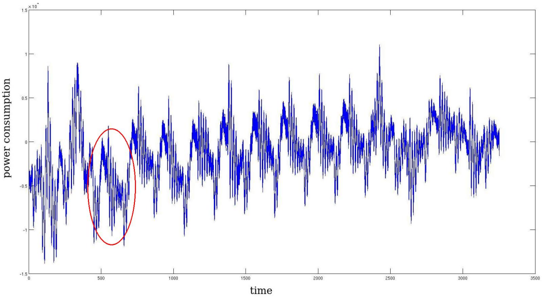
A Power-Analysis Attack against AES-128





A Power-Analysis Attack against AES-128





Attack on 8 bits

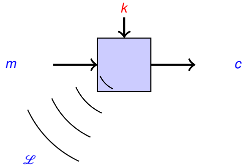
- prediction of the outputs for the 256 possible 8-bit secret
- correlation between predictions and leakage
- selection of the best correlation to find the correct 8-bit secret

Attack on 128 bits

- repetition of the attack on 8 bits on each S-box

Algorithmic Countermeasures

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

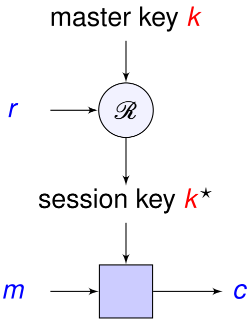


Two main algorithmic solutions:

- **Fresh Re-keying**: regularly change k
- **Masking**: make leakage $\mathscr{L}$ random

Fresh Re-keying

Idea: regularly change k



Masking

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

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

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

➜ each t-uple of $(v_i)_i$ is independent from v

Outline

Power-Analysis Attacks

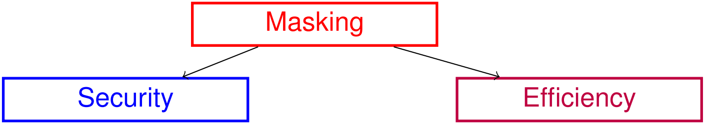
Masking Countermeasure

- Leakage Models
- Security in the probing model
- Construction of Secure Masking Schemes - Composition

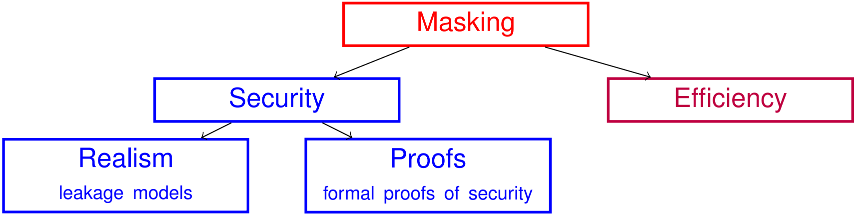
Current Research on Masking

Masking

Current Research on Masking



Current Research on Masking



Current Research on Masking

Masking

Security

Efficiency

Realism

leakage models

Proofs

formal proofs of security

[EC:PR13] Masking against Side-Channel Attacks: A Formal Security Proof

[EC:DDF14] Unifying Leakage Models: From Probing Attacks to Noisy Leakage

[EC:DFS15] Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device

...

Current Research on Masking

Masking

Security

Efficiency

Realism

leakage models

Proofs

formal proofs of security

[EC:PR13] Masking against Side-Channel Attacks: A Formal Security Proof

[EC:DDF14] Unifying Leakage Models: From Probing Attacks to Noisy Leakage

[EC:DFS15] Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device

...

[C:ISW03] Private Circuits: Securing Hardware against Probing Attacks

[CHES:RP10] Provably Secure Higher-Order Masking of AES

[FSE:CPRR13] Higher-Order Side Channel Security and Mask Refreshing

Current Research on Masking

- **Masking**
  - **Security**
    - **Realism** — leakage models
    - **Proofs** — formal proofs of security
  - **Efficiency**

Realism

[EC:PR13] Masking against Side-Channel Attacks: A Formal Security Proof

[EC:DDF14] Unifying Leakage Models: From Probing Attacks to Noisy Leakage

[EC:DFS15] Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device

...

Proofs

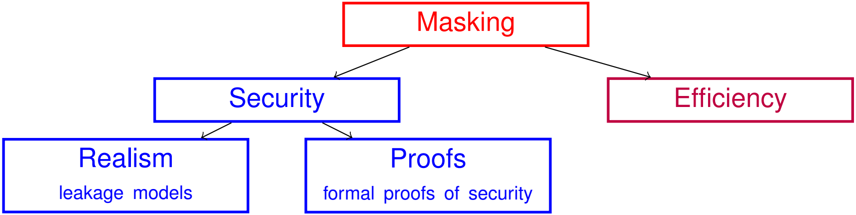
[C:ISW03] Private Circuits: Securing Hardware against Probing Attacks

[CHES:RP10] Provably Secure Higher-Order Masking of AES

[FSE:CPRR13] Higher-Order Side Channel Security and Mask Refreshing

[EC:B**B**DFGS15] formal proofs of masking schemes

Current Research on Masking



[EC:PR13] Masking against Side-Channel Attacks: A Formal Security Proof

[EC:DDF14] Unifying Leakage Models: From Probing Attacks to Noisy Leakage

[EC:DFS15] Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device

...

[C:ISW03] Private Circuits: Securing Hardware against Probing Attacks

[CHES:RP10] Provably Secure Higher-Order Masking of AES

[FSE:CPRR13] Higher-Order Side Channel Security and Mask Refreshing

[EC:B**B**DFGS15] formal proofs of masking schemes

[ePrint:B**B**DFG15] generation of formally proven masking schemes at any order

Current Research on Masking

Masking

Security

Efficiency

Realism
leakage models

Proofs
formal proofs of security

[EC:PR13] Masking against Side-Channel Attacks: A Formal Security Proof

[EC:DDF14] Unifying Leakage Models: From Probing Attacks to Noisy Leakage

[EC:DFS15] Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device

...

[C:ISW03] Private Circuits: Securing Hardware against Probing Attacks

[CHES:RP10] Provably Secure Higher-Order Masking of AES

[FSE:CPRR13] Higher-Order Side Channel Security and Mask Refreshing

[EC:B**B**DFGS15] formal proofs of masking schemes

[ePrint:B**B**DFG15] generation of formally proven masking schemes at any order

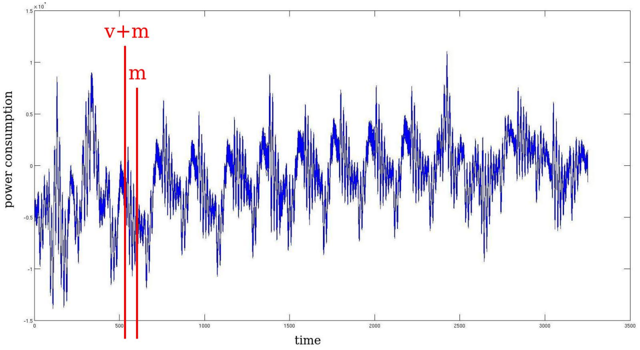
[FSE:CPRR13] Higher-Order Side Channel Security and Mask Refreshing

[EC:**B**BPPTV16] improvement of the randomness complexity for some multiplications

Outline

Power-Analysis Attacks on Masking Schemes

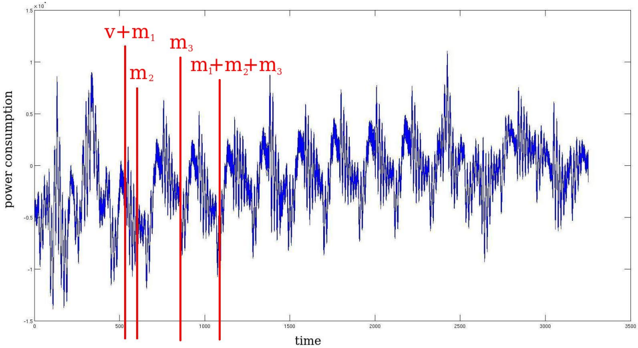
First-order masking



➜ compare $\mathscr{C}(\mathscr{L}(v+m), \mathscr{L}(m))$ to the predictions on v

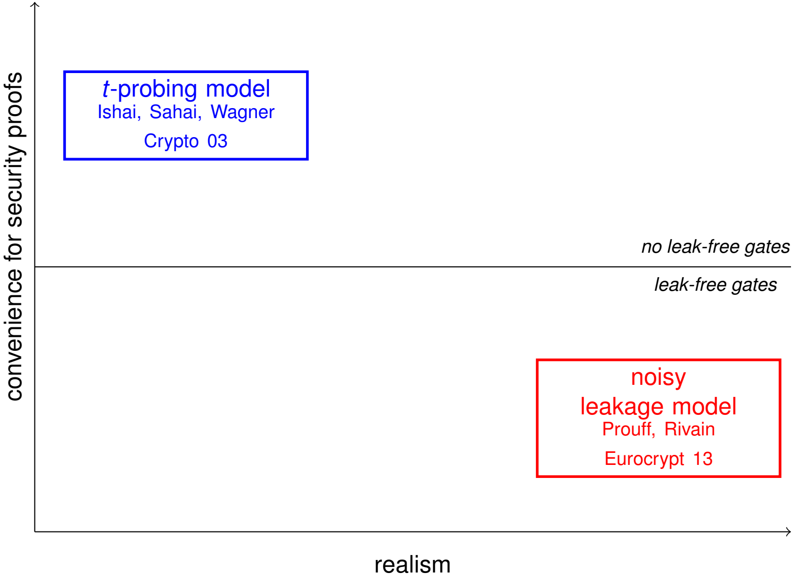
Power-Analysis Attacks on Masking Schemes

3^{rd}-order masking

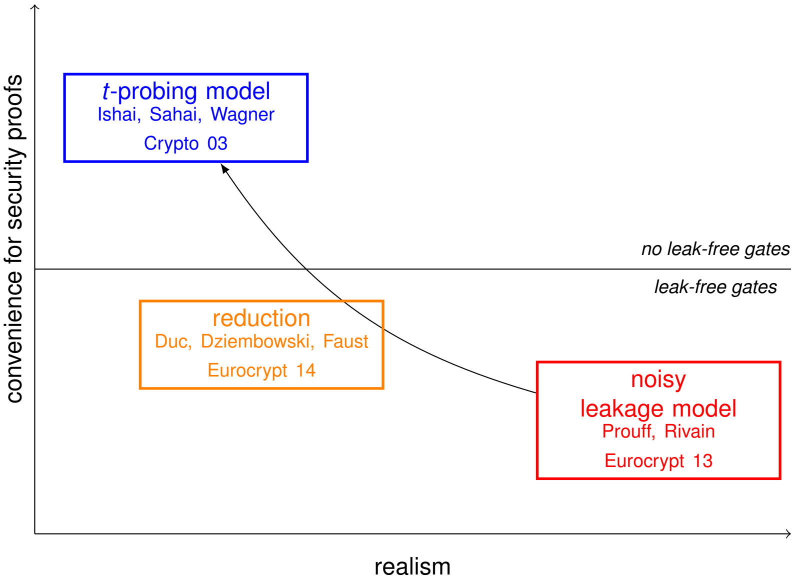


➜ compare $\mathscr{C}(\mathscr{L}(v+m_1),\mathscr{L}(m_2),\mathscr{L}(m_3),\mathscr{L}(m_1+m_2+m_3))$ to the predictions on v

Security of Masked Programs: Leakage Model



Security of Masked Programs: Leakage Model



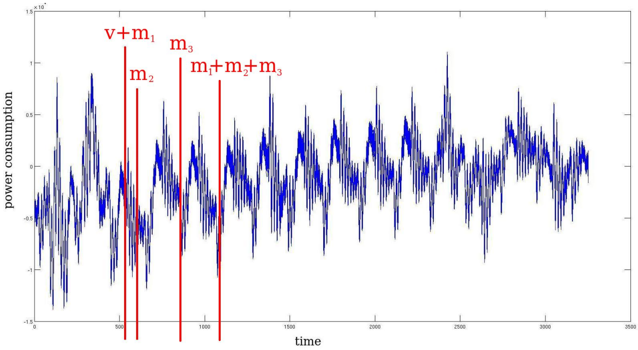
Outline

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

➜ show that all the t-uples are independent from the secret



Security in the t-probing model

v: randomly generated variable

c: known constant

x: secret variable

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

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

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

$\mathrm{return}(y_1, y_2, y_3, y_4)$

Security in the t-probing model

v: randomly generated variable

c: known constant

x: secret variable

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

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

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$t_1 \leftarrow x_2 + r_1$

$t_2 \leftarrow (x_2 + r_1) + x_3$

$y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

return(y_1, y_2, y_3, y_4)

1. independent from the secret?

Security in the t-probing model

v: randomly generated variable

c: known constant

x: secret variable

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

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

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

return(y_1, y_2, y_3, y_4)

1. independent from the secret?

✘

Security in the t-probing model

v: randomly generated variable

c: known constant

x: secret variable

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

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

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$t_1 \leftarrow x_2 + r_1$

$t_2 \leftarrow (x_2 + r_1) + x_3$

$y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

$\mathrm{return}(y_1, y_2, y_3, y_4)$

1. independent from the secret?

✘ many mistakes

Security in the t-probing model

v: randomly generated variable

c: known constant

x: secret variable

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

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

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

$\text{return}(y_1, y_2, y_3, y_4)$

1. independent from the secret?

✘ many mistakes

2. test 286 3-uples

✘ missing cases

✘ inefficient

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

Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub.
Verified proofs of higher-order masking. EUROCRYPT 2015.

1. Show that a t-uple is independent 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 $\mathrm{Ex\text{-}t3}(x_1, x_2, x_3, x_4, c)$:

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

$\mathrm{return}(y_1, y_2, y_3, y_4)$

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

1. Show that a t-uple is independent 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 $\mathrm{Ex\text{-}t3}(x_1, x_2, x_3, x_4, c)$:

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

$\mathrm{return}(y_1, y_2, y_3, y_4)$

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

1. Show that a t-uple is independent 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 $\mathrm{Ex\text{-}t3}(x_1, x_2, x_3, x_4, c)$:

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$t_1 \leftarrow x_2 + r_1$

$t_2 \leftarrow (x_2 + r_1) + x_3$

$y_3 \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

$\mathrm{return}(y_1, y_2, y_3, y_4)$

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

1. Show that a t-uple is independent 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 $\text{Ex-t3}(x_1, x_2, x_3, x_4, c)$:

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

$\text{return}(y_1, y_2, y_3, y_4)$

✔ ➜ distribution independent from the secret

✘ ➜ might be used for an attack

2. Extension to All Possible Sets

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

2. Extension to All Possible Sets

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

New Idea: proofs for sets of more than t variables

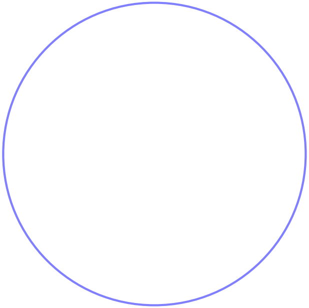
- find larger sets which cover all the intermediate variables is a hard problem
- two algorithms efficient in practice

2. 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



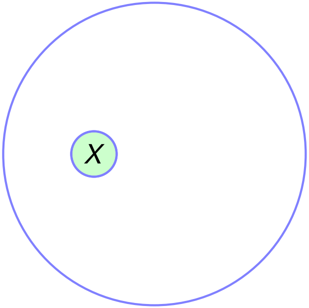
Algorithm 1:

2. 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:

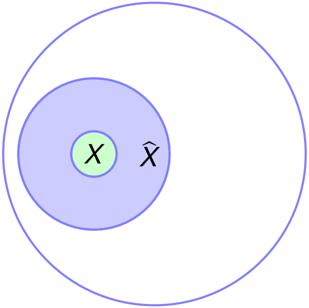
1. select $X = (t \text{ variables})$ and prove its independence

2. 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:

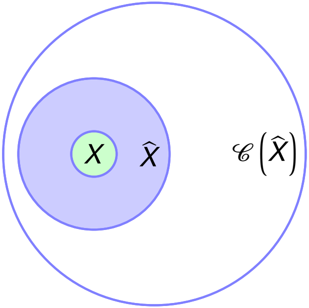
1. select $X = (t \text{ variables})$ and prove its independence
2. extend X to $\widehat{X}$ with more observations but still independence

2. 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:

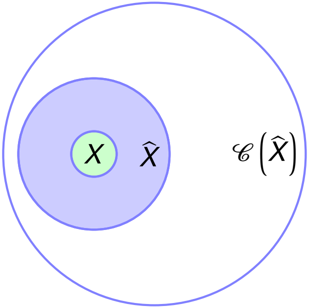
1. select $X = (t$ variables) and prove its independence
2. extend X to $\widehat{X}$ with more observations but still independence
3. recursively descend in set $\mathscr{C}\left(\widehat{X}\right)$

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

2. Extension to All Possible Sets: Example

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)

2. Extension to All Possible Sets: Example

X: ✔

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)

2. Extension to All Possible Sets: Example

$\hat{X}$: ✔

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)

2. Extension to All Possible Sets: Example

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

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$\hat{X}$: ✔ $\quad y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$\mathscr{C}(\hat{X})$: ✔ $\quad t_1 \leftarrow x_2 + r_1$

$\boxed{t_2} \leftarrow (x_2 + r_1) + x_3$

$\boxed{y_3} \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

return(y_1, y_2, y_3, y_4)

2. Extension to All Possible Sets: Example

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

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$\widehat{X}$: ✔ $\quad y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$\mathscr{C}(\widehat{X})$: ✔ $\quad t_1 \leftarrow x_2 + r_1$

$\boxed{t_2} \leftarrow (x_2 + r_1) + x_3$

merge $\widehat{X}$ and $\mathscr{C}(\widehat{X})$: ✘ $\quad \boxed{y_3} \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

return(y_1, y_2, y_3, y_4)

2. Extension to All Possible Sets: Example

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

$r_1 \leftarrow \$$

$r_2 \leftarrow \$$

$\widehat{X}$: ✔ $\quad y_1 \leftarrow x_1 + r_1$

$y_2 \leftarrow (x + x_1 + x_2 + x_3) + r_2$

$\mathscr{C}(\widehat{X})$: ✔ $\quad t_1 \leftarrow x_2 + r_1$

$\boxed{t_2} \leftarrow (x_2 + r_1) + x_3$

merge $\widehat{X}$ and $\mathscr{C}(\widehat{X})$: ✘ $\quad \boxed{y_3} \leftarrow (x_2 + r_1 + x_3) + r_2$

$y_4 \leftarrow c + r_2$

return(y_1, y_2, y_3, y_4)

➜ 207 proofs instead of 286

Application to the Sbox [CPRR13, Algorithm 4]

Method	# tuples	Security	Complexity # sets	Complexity time*
First-Order Masking				
naive	63	✔	63	0.001s
Alg. 1			17	0.001s
Alg. 2			17	0.001s
Second-Order Masking				
naive	12,561	✔	12,561	0.180s
Alg. 1			851	0.046s
Alg. 2			619	0.029s
Third-Order Masking				
naive	4,499,950	✔	4,499,950	140.642s
Alg. 1			68,492	9.923s
Alg. 2			33,075	3.894s
Fourth-Order Masking				
naive	2,277,036,685	✔	-	unpractical
Alg. 1			8,852,144	2959.770s
Alg. 2			3,343,587	879.235s

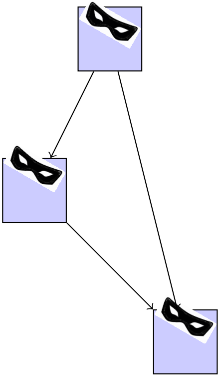
*run on a headless VM with a dual core (only one core is used in the computation) 64-bit processor clocked at 2GHz

Benchmarks

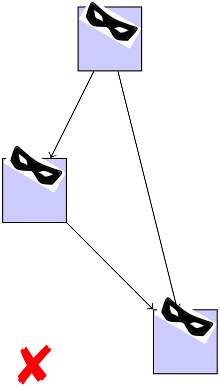
Reference	Target	# tuples	Security	Complexity # sets	time (s)
First-Order Masking					
FSE13	full AES	17,206	✔	3,342	128
MAC-SHA3	full Keccak-f	13,466	✔	5,421	405
Second-Order Masking					
RSA06	Sbox	1,188,111	✔	4,104	1.649
CHES10	Sbox	7,140	1^{st}-order flaws (2)	866	0.045
CHES10	AES KS	23,041,866	✔	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

Outline

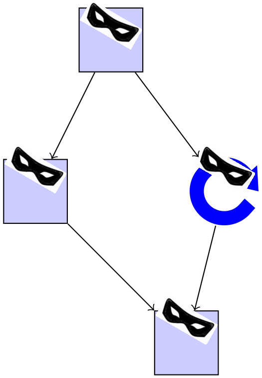
Current Issues in Composition



Current Issues in Composition

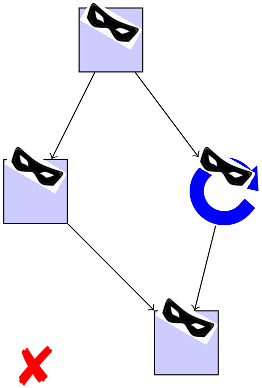


Current Issues in Composition



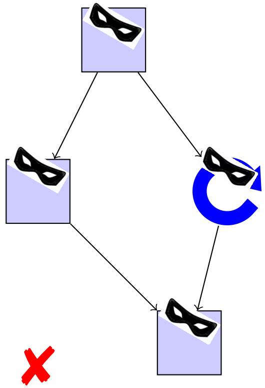
A refresh algorithm takes as input a sharing $(x_i)_{i\geq 0}$ of x and returns a new sharing $(x'_i)_{i\geq 0}$ of x such that $(x_i)_{i\geq 1}$ and $(x'_i)_{i\geq 1}$ are mutually independent.

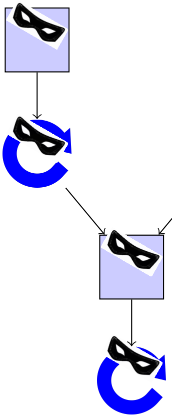
Current Issues in Composition



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 $(x_i)_{i \geq 1}$ and $(x'_i)_{i \geq 1}$ 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 $(x_i)_{i\geq 1}$ 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 scheme from any C implementation
   - efficient
   - any order

Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire.
Compositional Verification of Higher-Order Masking Application to a Verifying Masking Compiler. ePrint 2015.

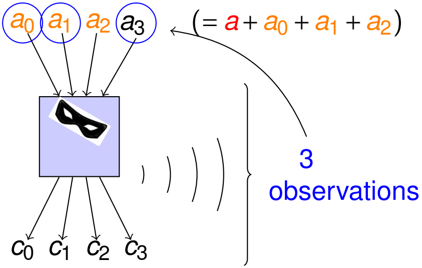
Security properties in the t-probing model

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

Security properties in the t-probing model

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

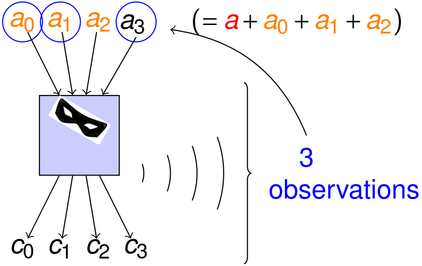
if t is not fixed: show that any set of t intermediate variables can be simulated with at most t shares of each input



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($a_0, \ldots, a_i, \ldots a_t$):

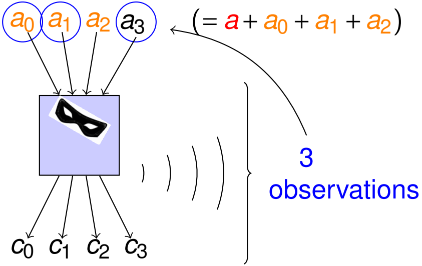
    for $i = 0$ to t
        $c_i \leftarrow f(a_i)$
    return $(c_0, \ldots, c_i, \ldots, c_t)$

➜ straightforward for linear functions

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

$c_i \leftarrow f(a_i)$

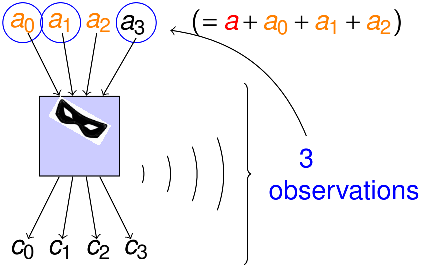
return $(c_0, \ldots, c_i, \ldots, c_t)$

➜ straightforward for linear functions

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

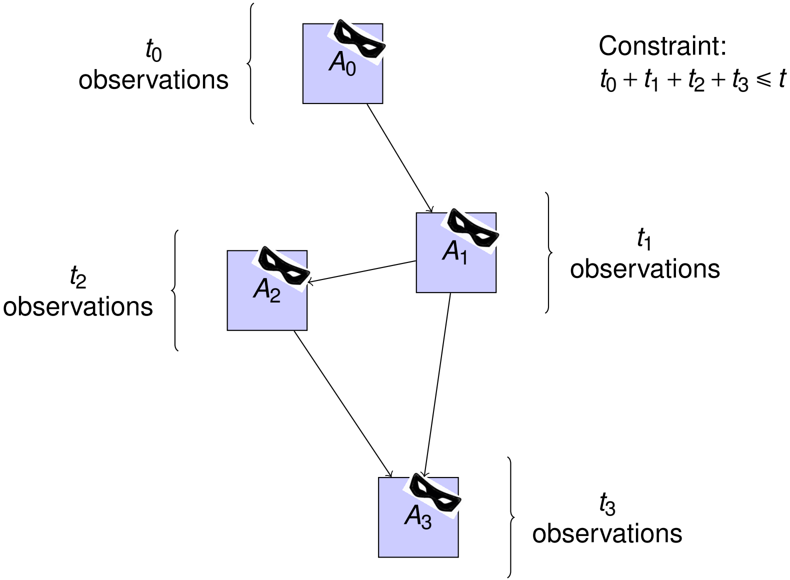
$c_i \leftarrow f(a_i)$

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

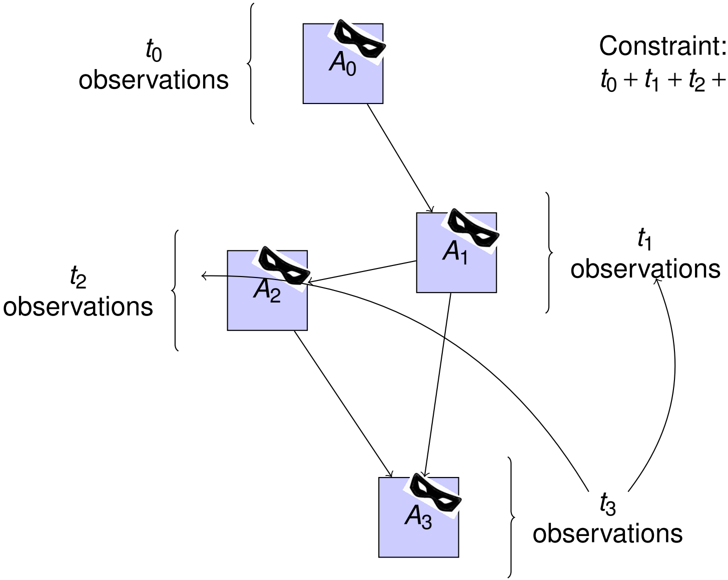
➜ straightforward for linear functions

➜ formal proofs with EasyCrypt and pen-and paper proofs for small non-linear functions

Current Issues

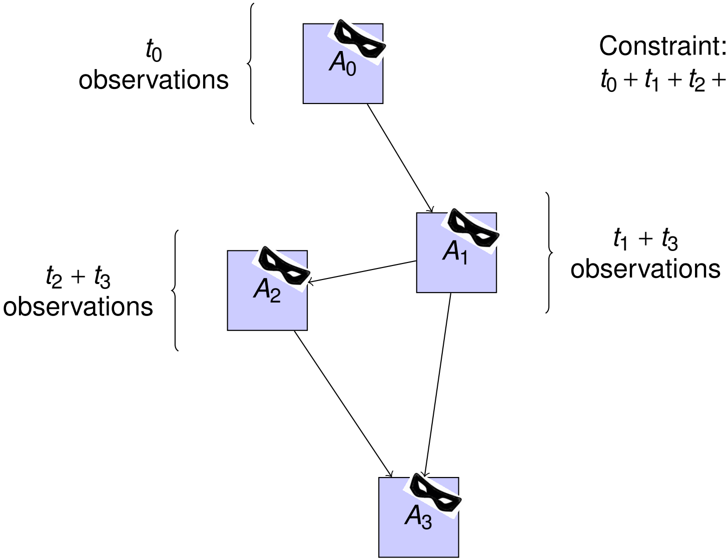


Current Issues



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

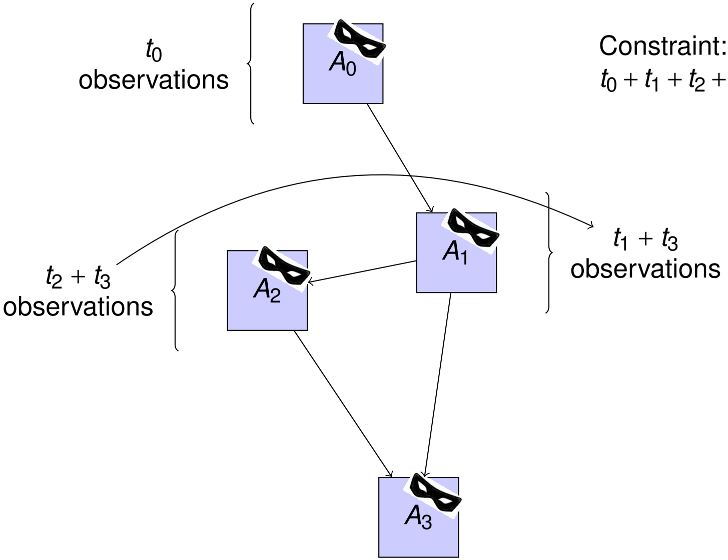
Current Issues



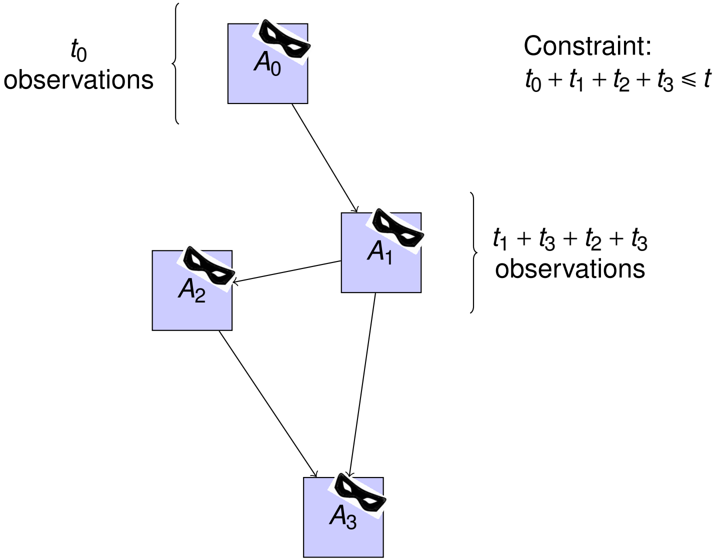
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$

Current Issues

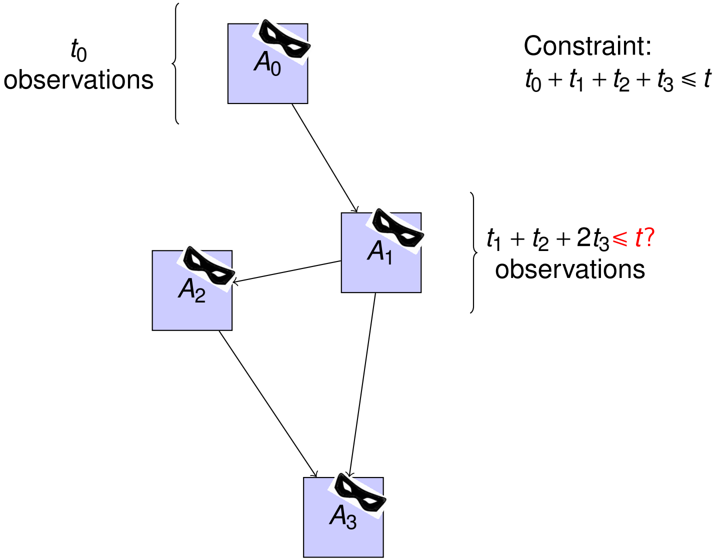
Constraint:
$t_0 + t_1 + t_2 + t_3 \leqslant t$



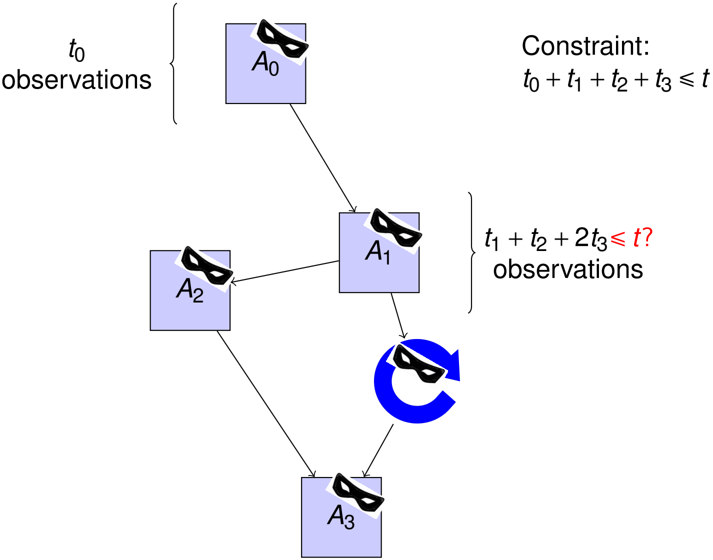
Current Issues



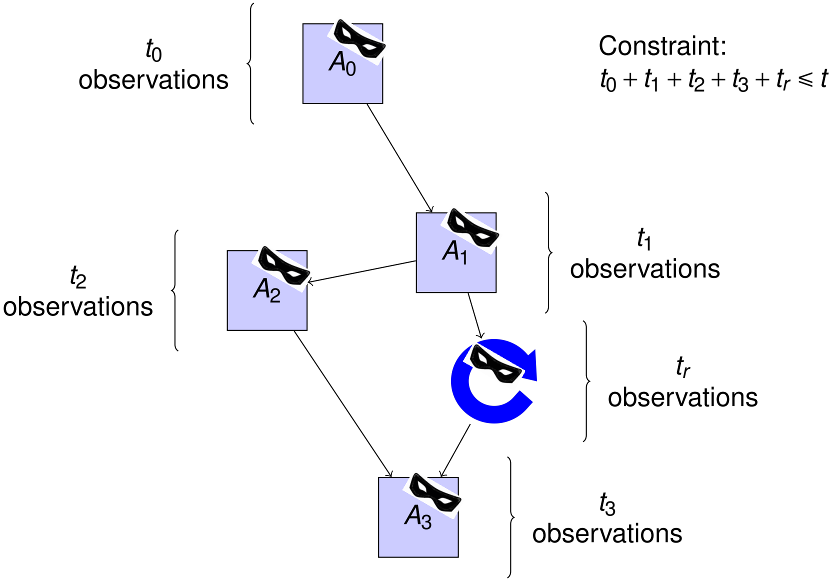
Current Issues



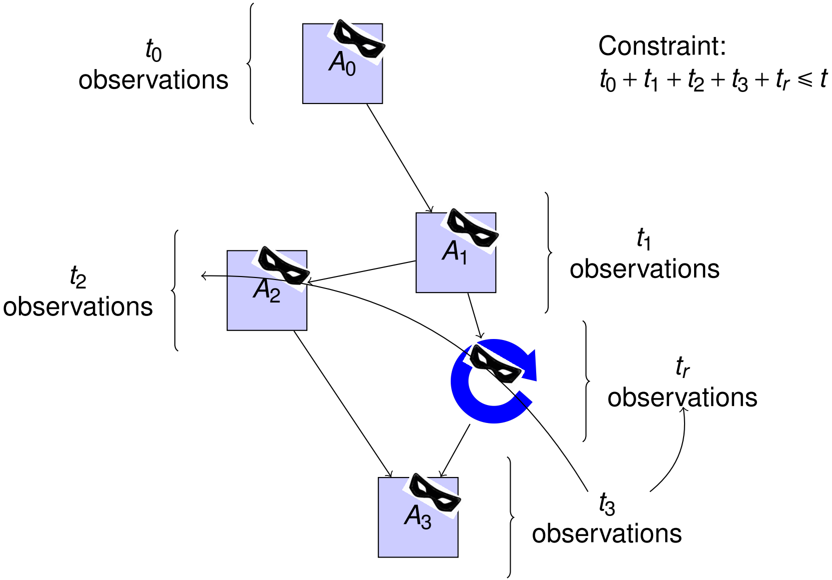
Current Issues



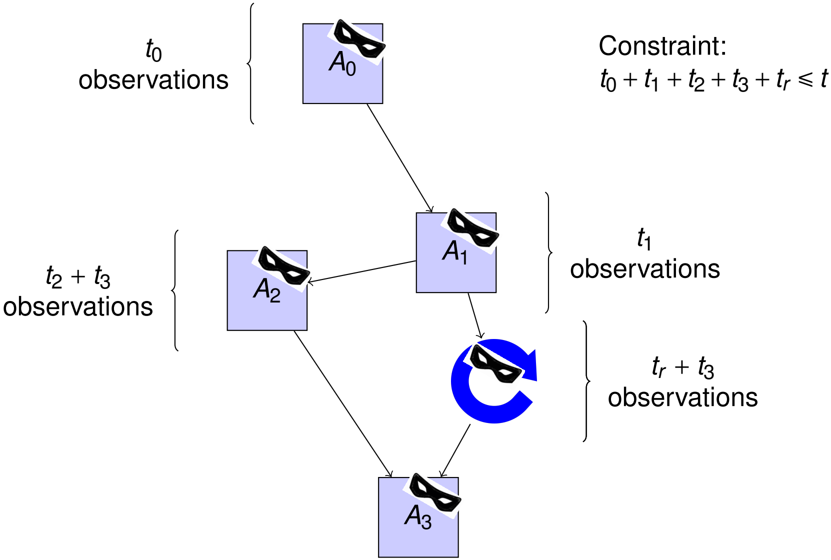
Current Issues



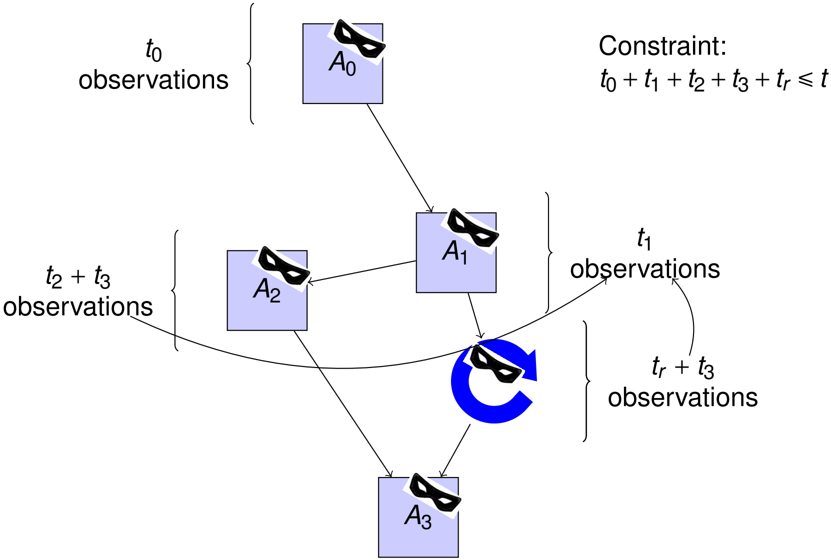
Current Issues



Current Issues

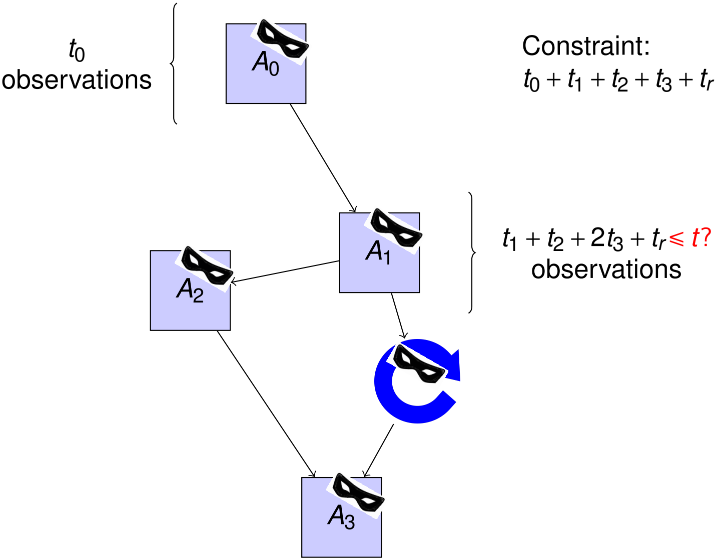


Current Issues



Current Issues

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



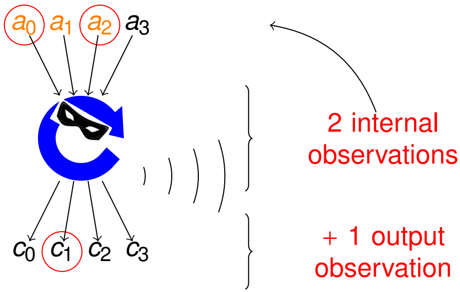
Stronger security property for Refresh

Strong Non-Interference in the t-probing model:

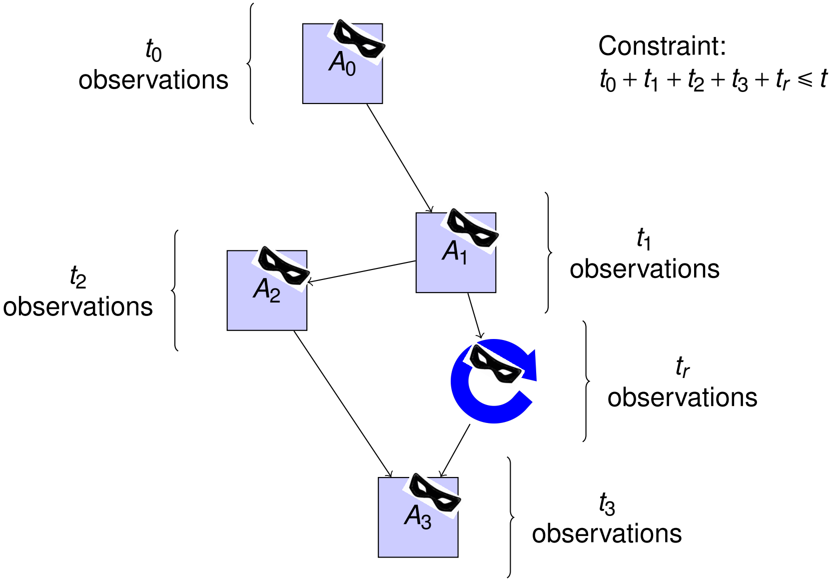
if t is not fixed: show that any set of t intermediate variables with

- t_1 on internal variables
- $t_2 = t - t_1$ on the outputs

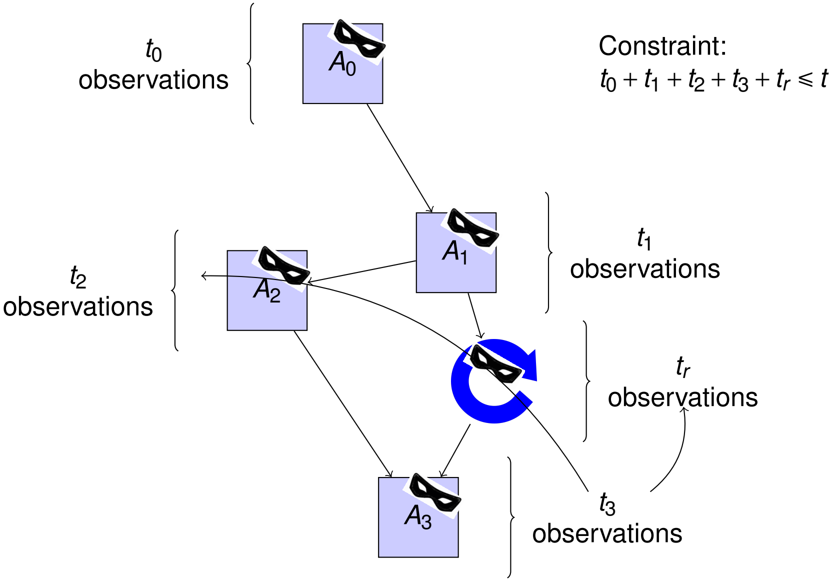
can be simulated with at most t_1 shares of each input



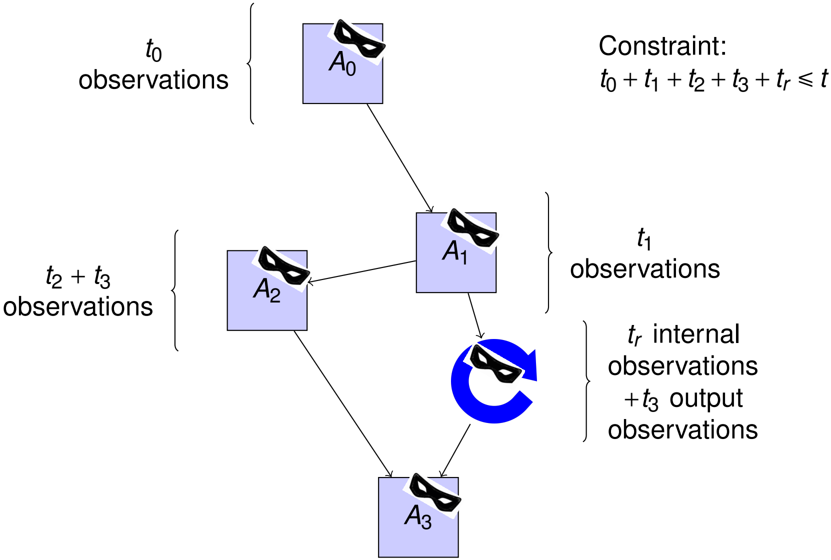
Secure Composition



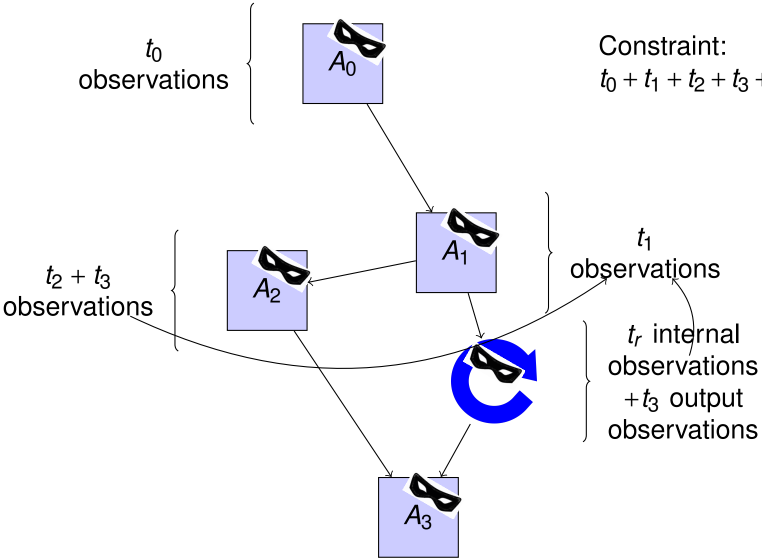
Secure Composition



Secure Composition

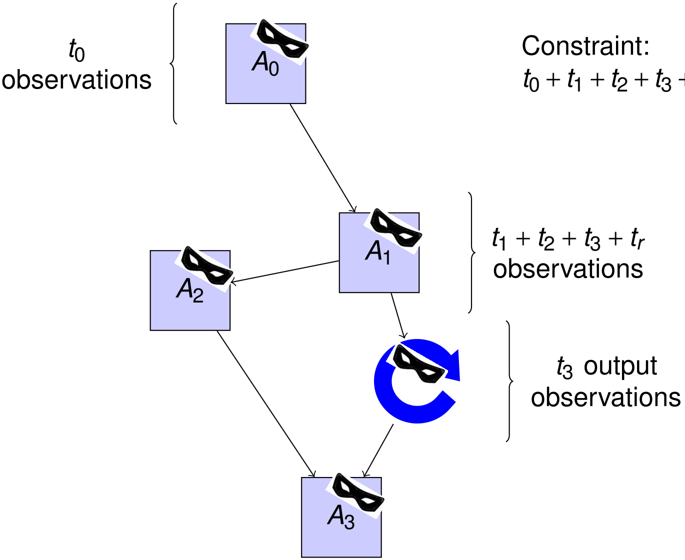


Secure Composition



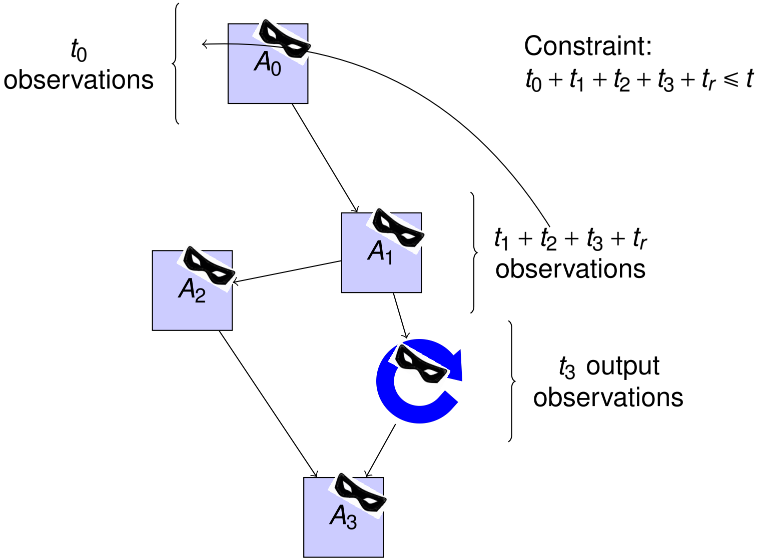
Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leqslant t$

Secure Composition

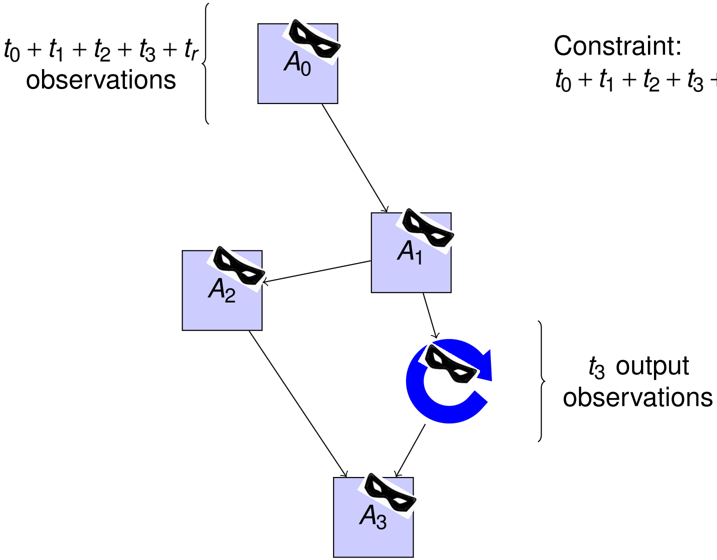


Constraint:
$t_0 + t_1 + t_2 + t_3 + t_r \leqslant t$

Secure Composition

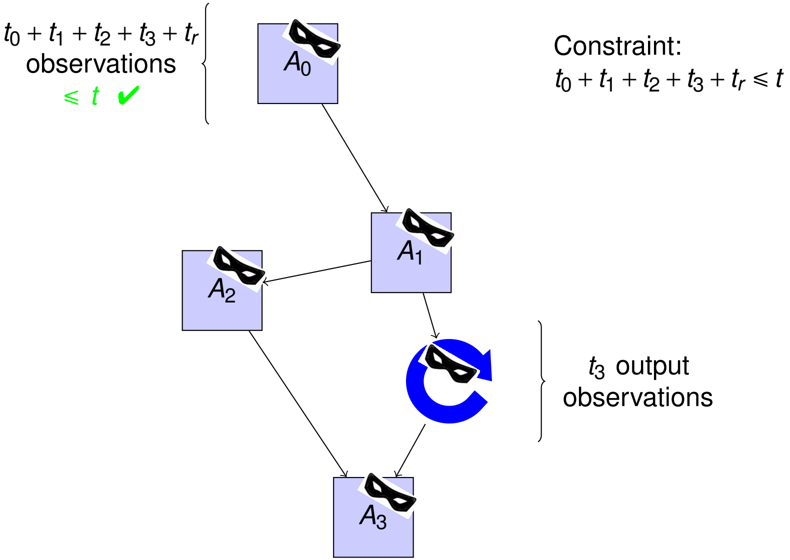


Secure Composition



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

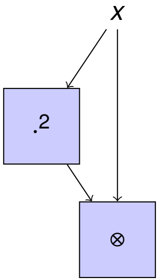
Secure Composition



Secure Composition

Automatic tool for C-based algorithms

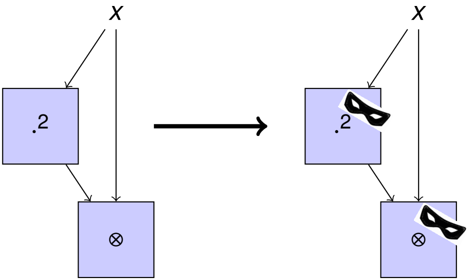
- unprotected algorithm ➜ higher-order masked algorithm
- example for AES S-box



Secure Composition

Automatic tool for C-based algorithms

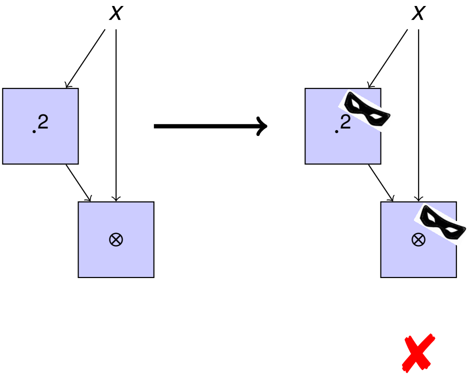
- unprotected algorithm ➜ higher-order masked algorithm
- example for AES S-box



Secure Composition

Automatic tool for C-based algorithms

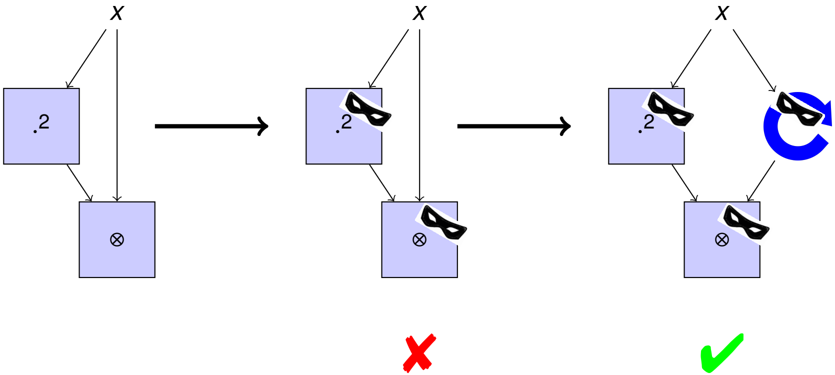
- unprotected algorithm ➜ higher-order masked algorithm
- example for AES S-box



Secure Composition

Automatic tool for C-based algorithms

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



Some Results

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

Scheme	# Refresh	Time	Memory
AES ($\odot$)	2/Sbox	0.09s	4Mo
AES ($x \odot g(x)$)	0	0.05s	4Mo
Keccak with Refresh	0	121.20	456Mo
Keccak	600	2728.00s	22870Mo
Simon	67	0.38s	15Mo
Speck	61	6.22s	38Mo

[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]

Scheme	# Refresh	Time	Memory
AES ($\odot$)	2/Sbox	0.09s	4Mo
AES ($x \odot g(x)$)	0	0.05s	4Mo
Keccak with Refresh	0	121.20s	456Mo
Keccak	600	2728.00s	22870Mo
Simon	67	0.38s	15Mo
Speck	61	6.22s	38Mo

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

Conclusion

Masking

Security

Realism

models close enough to the reality

Proofs

formal proofs of security

[EC15] formal proofs of masking schemes

[ePrint15] generation of formally proven masking schemes at any order

Efficiency

[EC16] improvement of the randomness complexity for some multiplications

Conclusion

Masking

Security

Efficiency

[EC16] improvement of the randomness complexity for some multiplications

Realism
models close enough to the reality

Proofs
formal proofs of security

[EC15] formal proofs of masking schemes

[ePrint15] generation of formally proven masking schemes at any order

➜ extend the verification to higher orders using composition

Conclusion

Masking

- **Security**
  - **Realism**
    models close enough to the reality
  - **Proofs**
    formal proofs of security

    [EC15] formal proofs of masking schemes

    [ePrint15] generation of formally proven masking schemes at any order

    ➜ extend the verification to higher orders using composition

  ➜ integrate transition/glitch-based model
- **Efficiency**

  [EC16] improvement of the randomness complexity for some multiplications

Conclusion

Masking

Security

Efficiency

[EC16] improvement of the randomness complexity for some multiplications

Realism

models close enough to the reality

Proofs

formal proofs of security

[EC15] formal proofs of masking schemes

[ePrint15] generation of formally proven masking schemes at any order

➜ extend the verification to higher orders using composition

➜ integrate transition/glitch-based model

➜ build practical experiments for both attacks and new countermeasures

Conclusion

Masking

Security

Realism
models close enough to the reality

Proofs
formal proofs of security

[EC15] formal proofs of masking schemes

[ePrint15] generation of formally proven masking schemes at any order

➜ extend the verification to higher orders using composition

➜ integrate transition/glitch-based model

➜ build practical experiments for both attacks and new countermeasures

Efficiency

[EC16] improvement of the randomness complexity for some multiplications

➜ still reduce the randomness in multiplications