ProVerif, restrictions, equivalence... what could go wrong?

Alexandre Debant

Université de Lorraine, Inria, CNRS, Nancy, France

Pesto seminar
April 12th, 2024 - Nancy, France

Laboratoire lorrain de recherche en informatique et ses applications

FRANCE 2030

Opening remarks

- this talk does not necessarily follow ProVerif notations
- what is written is not necessarily formally correct
- this talk is about ProVerif v2.05 (unless specific comment)

Modelling protocols

$$
\begin{aligned}
P, Q := {} & 0 \\
& \mid \mathsf{new}\ n;\ \ P \\
& \mid \mathsf{in}(c, x);\ \ P \\
& \mid \mathsf{out}(c, u);\ \ P \\
& \mid \mathsf{let}\ u = v\ \mathsf{in}\ P\ \mathsf{else}\ Q \\
& \mid \mathsf{insert}\ tbl(u);\ \ P \\
& \mid \mathsf{get}\ tbl(x)\ \mathsf{suchthat}\ \phi\ \mathsf{in}\ P\ \mathsf{else}\ Q \\
& \mid (P \mid Q) \\
& \mid\ !P \\
& \mid \mathsf{event}\ e(u_1, \ldots, u_n);\ \ P
\end{aligned}
$$

ProVerif before v2.02

Modelling protocols

$$
\begin{aligned}
P, Q := 0 & \\
& \mid \text{new } n;\ P \\
& \mid \text{in}(c, x);\ P \\
& \mid \text{out}(c, u);\ P \\
& \mid \text{let } u = v \text{ in } P \text{ else } Q \\
& \mid \text{insert } tbl(u);\ P \\
& \mid \text{get } tbl(x) \text{ suchthat } \phi \text{ in } P \text{ else } Q \\
& \mid (P \mid Q) \\
& \mid\ !P \\
& \mid \text{event } e(u_1, \ldots, u_n);\ P
\end{aligned}
$$

ProVerif before v2.02

$+$

Restrictions:

$\rho := F_1 \ \&\& \ \ldots F_n \ \&\& \ \Rightarrow H$

💡 "Consider only traces that satisfy ρ, i.e. $tr \vdash \rho$"

ProVerif since v2.02

Example

Evoting: ballot weeding

$$
\begin{array}{l}
Server = \\
\quad !\ (\\
\qquad \mathsf{in}(c, x); \\
\qquad \mathsf{in}(cell, x_{token}); \\
\qquad \mathsf{get}\ BB(y)\ \mathsf{suchthat}\ x = y\ \mathsf{in} \\
\qquad\quad \mathsf{out}(cell, x_{token}) \quad \textit{(* ballot already accepted *)} \\
\qquad \mathsf{else} \\
\qquad\quad \mathsf{insert}\ BB(x); \\
\qquad\quad \mathsf{out}(cell, x_{token}); \\
\qquad\quad \ldots \\
\quad)
\end{array}
$$

Example

Evoting: ballot weeding

$$
\begin{array}{l}
Server = \\
\quad !\ (\\
\quad\quad \mathsf{in}(c, x); \\
\quad\quad \mathsf{in}(cell, x_{token}); \\
\quad\quad \mathsf{get}\ BB(y)\ \mathsf{suchthat}\ x = y\ \mathsf{in} \\
\quad\quad\quad \mathsf{out}(cell, x_{token}) \quad \textit{(* ballot already accepted *)} \\
\quad\quad \mathsf{else} \\
\quad\quad\quad \mathsf{insert}\ BB(x); \\
\quad\quad\quad \mathsf{out}(cell, x_{token}); \\
\quad\quad\quad \ldots \\
\quad)
\end{array}
$$

You may have troubles with else branches and cells …

Example

Evoting: ballot weeding

$Server =$
$\quad ! \, ($
$\quad\quad$ in(c, x);
$\quad\quad$ in$(cell, x_{token})$;
$\quad\quad$ get $BB(y)$ suchthat $x = y$ in
$\quad\quad\quad$ out$(cell, x_{token})$ *(* ballot already accepted *)*
$\quad\quad$ else
$\quad\quad\quad$ insert $BB(x)$;
$\quad\quad\quad$ out$(cell, x_{token})$;
$\quad\quad\quad \ldots$
$\quad)$

You may have troubles with **else branches** and **cells** …

$Server =$
$\quad ! \, ($
$\quad\quad$ in(c, x);
$\quad\quad$ new st; event $Inserted(st, x)$;
$\quad\quad$ insert $BB(x)$;
$\quad\quad \ldots$
$\quad)$

$+$

Restriction:

$$\text{event}(Inserted(st_1, x)) \;\&\&\; \text{event}(Inserted(st_2, x)) \Rightarrow st_1 = st_2 \,.$$

Example

Evoting: ballot weeding

$Server =$
  $!\,($
    in(c, x);
    in$(cell, x_{token})$;
    get $BB(y)$ suchthat $x = y$ in
      out$(cell, x_{token})$ *(* ballot already accepted *)*
    else
      insert $BB(x)$;
      out$(cell, x_{token})$;
      $\ldots$
  $)$

You may have troubles with **else branches** and **cells** …

$Server =$
  $!\,($
    in(c, x);
    new st; event $Inserted(st, x)$;
    insert $BB(x)$;
    $\ldots$
  $)$

$+$

Restriction:
  event$(Inserted(st_1, x))$
   && event$(Inserted(st_2, x)) \Rightarrow st_1 = st_2$.

No cell, no else branch

Other examples

- Ballot weeding in evoting protocols — event$(Inserted(st_1, x))$ && event$(Inserted(st_2, x)) \Rightarrow st_1 = st_2$
- Key updates / key revocations — event$(Use(k_1))$ && event$(Inserted(k_2))$ && $subterm(k_1, k_2) \Rightarrow false$
- Model protocol assumptions (e.g., audits) — event$(PublishedOnBB(b)) \Rightarrow \phi(b)$
- Easily bound the number of executions — event$(Iteration(n)) \Rightarrow n < 2$
- Abstract e.g. arithmetic properties — See [Cortier et. al. - CCS'21]
- …

How does it work?

(simplified)

$$\frac{\mathbb{C} \cup \{R = H \to C\} \qquad (\wedge_{i=1}^{n} F_i \Rightarrow \psi) \in \mathscr{R} \qquad \text{For all } i,\ F_i\sigma \in H}{\mathbb{C} \cup \{R = H \wedge \psi\sigma \to C\}}$$

How does it work?
(simplified)

$$\frac{\mathbb{C} \cup \{R = H \rightarrow C\} \qquad (\wedge_{i=1}^{n} F_i \Rightarrow \psi) \in \mathscr{R} \qquad \text{For all } i,\ F_i\sigma \in H}{\mathbb{C} \cup \{R = H \wedge \psi\sigma \rightarrow C\}}$$

⚠ It is just a matching!

If the clause is not instantiated enough (e.g. noselect) the restriction will not be applied!

Usual issues

Given the process $P :=$ event$(E1)$; event$(E2)$; event$(E3)$

and the restriction $\rho :=$ event$(E1) \Rightarrow$ event$(E2)$, is event$(E3)$ reachable?

Usual issues

Given the process $P := \text{event}(E1); \text{event}(E2); \text{event}(E3)$

and the restriction $\rho := \text{event}(E1) \Rightarrow \text{event}(E2)$, is $\text{event}(E3)$ reachable?

No!
Restrictions have the same semantics as queries

Usual issues

Given the process $P :=$ event$(E1)$; event$(E2)$; event$(E3)$

and the restriction $\rho :=$ event$(E1) \Rightarrow$ event$(E2)$, is event$(E3)$ reachable?

No!
Restrictions have the same semantics as queries

Given the process $P :=$ (event$(E1)$; event$(E2)$) $|$ event$(E3)$

and the restriction $\rho :=$ event$(E3) \Rightarrow$ event$(E2)$,

is ProVerif able to prove $\rho' :=$ event$(E3) \Rightarrow$ event$(E1)$?

```
adebant@macbook-pro-de-alexandre-2 proverif-examples % proverif example4.pv
Process 0 (that is, the initial process):
(
    {1}event E1;
    {2}event E2
) | (
    {3}event E3
)

-- Restriction event(E3) ==> event(E2) in process 0.
-- Query event(E3) ==> event(E1) in process 0.
Translating the process into Horn clauses...
Completing...
Starting query event(E3) ==> event(E1)
goal reachable: b-event(E2) -> event(E3)

Derivation:

1. Event E3 may be executed at {3}.
event(E3).

2. By 1, event(E3).
The goal is reached, represented in the following fact:
event(E3).


A more detailed output of the traces is available with
  set traceDisplay = long.

event E3 at {3} (goal)

The event E3 is executed at {3}.
A trace has been found.

The attack trace does not satisfy the following restriction, declared at File "example4.pv", line 16, characters 13-35:
event(E3) ==> event(E2)
RESULT event(E3) ==> event(E1) cannot be proved.

--------------------------------------------------------------
```

Usual issues

Given the process $P := \text{event}(E1); \text{event}(E2); \text{event}(E3)$

and the restriction $\rho := \text{event}(E1) \Rightarrow \text{event}(E2)$, is $\text{event}(E3)$ reachable?

No!
Restrictions have the same semantics as queries

Given the process $P := (\text{event}(E1); \text{event}(E2)) \mid \text{event}(E3)$

and the restriction $\rho := \text{event}(E3) \Rightarrow \text{event}(E2)$,

is ProVerif able to prove $\rho' := \text{event}(E3) \Rightarrow \text{event}(E1)$?

No...

$$\Rightarrow \text{event}(E3) \xrightarrow{\text{apply } \rho} \text{event}(E2) \Rightarrow \text{event}(E3)$$

Not enough to conclude… 😢

Usual issues

Given the process $P := \text{event}(E1); \text{event}(E2); \text{event}(E3)$

and the restriction $\rho := \text{event}(E1) \Rightarrow \text{event}(E2)$, is $\text{event}(E3)$ reachable?

No!
Restrictions have the same semantics as queries

Given the process $P := (\text{event}(E1); \text{event}(E2)) \mid \text{event}(E3)$

and the restriction $\rho := \text{event}(E3) \Rightarrow \text{event}(E2)$,

is ProVerif able to prove $\rho' := \text{event}(E3) \Rightarrow \text{event}(E1)$?

No…

$\Rightarrow \text{event}(E3) \xrightarrow{\text{apply } \rho} \text{event}(E2) \Rightarrow \text{event}(E3)$

Not enough to conclude… 😢

💡 **You can use the development branch** `improve-scope-lemma` **to make it prove**

What about equivalence properties?

🤔

Reminder

- ProVerif proves equivalence of processes that differ only by terms
- ProVerif internally proves diff-equivalence

$$P[a_1, \ldots, a_n] \approx P[b_1, \ldots, b_n]$$

$$\downarrow$$

$$P[\mathit{diff}[a_1, b_1], \ldots, \mathit{diff}[a_n, b_n]] \upharpoonleft\!\downharpoonright$$

Definition - "A biprocess P is in diff-equivalence if $traces(P) \downharpoonleft\!\upharpoonright$ i.e., for all traces of P, the first and the second projections progress in the same way."

Reminder

- ProVerif proves equivalence of processes that differ only by terms

$$P[a_1, \ldots, a_n] \approx P[b_1, \ldots, b_n]$$

$$\downarrow$$

$$P[\mathit{diff}[a_1, b_1], \ldots, \mathit{diff}[a_n, b_n]] \upharpoonleft\!\downharpoonright$$

- ProVerif internally proves diff-equivalence

Definition - "A biprocess P is in diff-equivalence if $\mathit{traces}(P) \downharpoonleft\!\upharpoonright$ i.e., for all traces of P, the first and the second projections progress in the same way."

$$(\mathsf{let}\ x = v\ \mathsf{in}\ P\ \mathsf{else}\ Q) \mid \mathscr{P} \longrightarrow P\{x \mapsto \mathit{diff}[M^{\mathsf{L}}, M^{\mathsf{R}}]\} \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(v)\Downarrow = M^{\mathsf{L}} \text{ and } \mathsf{snd}(v)\Downarrow = M^{\mathsf{R}}$$

Reminder

- ProVerif proves equivalence of processes that differ only by terms

$$P[a_1, \dots, a_n] \approx P[b_1, \dots, b_n]$$

$$\Big\downarrow$$

$$P[\mathit{diff}[a_1, b_1], \dots, \mathit{diff}[a_n, b_n]] \upharpoonleft\downharpoonright$$

- ProVerif internally proves diff-equivalence

> **Definition -** "A biprocess P is in diff-equivalence if $\mathit{traces}(P) \downharpoonleft\upharpoonright$ i.e., for all traces of P, the first and the second projections progress in the same way."

$$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathscr{P} \longrightarrow P\{x \mapsto \mathit{diff}[M^{\mathsf{L}}, M^{\mathsf{R}}]\} \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(v)\Downarrow = M^{\mathsf{L}} \text{ and } \mathsf{snd}(v)\Downarrow = M^{\mathsf{R}}$$

$$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathscr{P} \longrightarrow Q \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(v)\Downarrow = \mathit{fail} \text{ and } \mathsf{snd}(v)\Downarrow = \mathit{fail}$$

Reminder

- ProVerif proves equivalence of processes that differ only by terms

- ProVerif internally proves diff-equivalence

$$P[a_1, \dots, a_n] \approx P[b_1, \dots, b_n]$$

$$\downarrow$$

$$P[\mathit{diff}[a_1, b_1], \dots, \mathit{diff}[a_n, b_n]] \upharpoonleft\!\downharpoonright$$

Definition - "A biprocess P is in diff-equivalence if $\mathit{traces}(P) \downharpoonleft\!\upharpoonright$ i.e., for all traces of P, the first and the second projections progress in the same way."

$$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathscr{P} \longrightarrow P\{x \mapsto \mathit{diff}[M^{\mathsf{L}}, M^{\mathsf{R}}]\} \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(v)\Downarrow = M^{\mathsf{L}} \text{ and } \mathsf{snd}(v)\Downarrow = M^{\mathsf{R}}$$

$$(\text{let } x = v \text{ in } P \text{ else } Q) \mid \mathscr{P} \longrightarrow Q \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(v)\Downarrow = \mathit{fail} \text{ and } \mathsf{snd}(v)\Downarrow = \mathit{fail}$$

$$(\mathsf{in}(c, x);\ P) \mid (\mathsf{out}(c', u);\ Q) \mid \mathscr{P} \longrightarrow P\{x \mapsto u\} \mid Q \mid \mathscr{P} \qquad \text{if } \mathsf{fst}(c) = \mathsf{fst}(c') \text{ and } \mathsf{snd}(c) = \mathsf{snd}(c')$$

$$\dots$$

Reminder

Theorem [Blanchet et. al. 2006]

Given a biprocess P, $\quad \mathit{traces}(P) \downarrow\!\uparrow \Rightarrow \mathsf{fst}(P) \approx \mathsf{snd}(P)$

where $\approx$ denotes the observational equivalence relation.

Reminder

Theorem [Blanchet et. al. 2006]

Given a biprocess P, $\quad traces(P) \downarrow\uparrow \Rightarrow \mathsf{fst}(P) \approx \mathsf{snd}(P)$

where $\approx$ denotes the observational equivalence relation.

```
adebant@macbook-pro-de-alexandre-2 proverif-examples % proverif example1.pv
Biprocess 0 (that is, the initial process):
(
    {1}new n: bitstring;
    {2}new m: bitstring;
    {3}out(cpriv, choice[n,m])
) | (
    {4}in(cpriv, x: bitstring);
    {5}out(cpub, x)
)

-- Observational equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Observational equivalence is true.

--------------------------------------------------------------
Verification summary:

Observational equivalence is true.

--------------------------------------------------------------
```

Equivalence with restrictions

- We can write restrictions, e.g. $\rho := \mathsf{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}} \ \&\& \ x^{\mathsf{R}} = y^{\mathsf{R}}$

Equivalence with restrictions

- We can write restrictions, e.g. $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}} \ \&\&\ x^{\mathsf{R}} = y^{\mathsf{R}}$

$$\rho' := \text{event}(E(x, y)) \Rightarrow x = y \not\equiv \rho$$

$$\rho' := \text{event}(E(x, y)) \Rightarrow x = y \ \equiv\ \text{event}(E(\mathit{diff}[x, x], \mathit{diff}[y, y])) \Rightarrow x = y$$

Equivalence with restrictions

- We can write restrictions, e.g. $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}} \;\&\&\; x^{\mathsf{R}} = y^{\mathsf{R}}$

$\rho' := \text{event}(E(x, y$ …

$\rho' := \text{event}(E(x, y$ … $\Rightarrow x = y$

Always define restrictions with explicit $\mathit{diff}[\,\cdot\,,\,\cdot\,]$ operators!

Equivalence with restrictions

- We can write restrictions, e.g. $\rho := \mathsf{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}} \;\&\&\; x^{\mathsf{R}} = y^{\mathsf{R}}$

$\rho' := \mathsf{event}(E(x, y$

Always define restrictions with explicit $\mathit{diff}[\,\cdot\,,\cdot\,]$ operators!

$\rho' := \mathsf{event}(E(x, y)) \Rightarrow x = y \;\equiv\; \mathsf{event}(E(\mathit{diff}[x, x], \mathit{diff}[y, y])) \Rightarrow x = y$

Definition - A biprocess P is in diff-equivalence for the restrictions $\mathscr{R}$, if $\mathit{traces}_{|\mathscr{R}}(P) \downarrow\uparrow$ i.e., for all traces tr of P that satisfy $\mathscr{R}$, $\forall \rho \in \mathscr{R}, tr \vdash \rho$ the first and the second projections progress in the same way.

Relation with observational equivalence

Definition - Let P^{L}, P^{R} be two processes and $\mathscr{R}^{\mathsf{L}}$, $\mathscr{R}^{\mathsf{R}}$ be two sets of restrictions.
Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😇) and denoted $(P^{\mathsf{L}}, \mathscr{R}^{\mathsf{L}}) \approx (P^{\mathsf{R}}, \mathscr{R}^{\mathsf{R}})$

Relation with observational equivalence

Definition - Let P^{L}, P^{R} be two processes and $\mathscr{R}^{\mathsf{L}}$, $\mathscr{R}^{\mathsf{R}}$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😇) and denoted $(P^{\mathsf{L}}, \mathscr{R}^{\mathsf{L}}) \approx (P^{\mathsf{R}}, \mathscr{R}^{\mathsf{R}})$

New-theorem?

Given a biprocess P, and a set of restrictions $\mathscr{R}$,

$$traces_{|\mathscr{R}}(P) \downarrow\uparrow \Rightarrow (\mathsf{fst}(P), \mathsf{fst}(\mathscr{R})) \approx (\mathsf{snd}(P), \mathsf{snd}(\mathscr{R})).$$

Relation with observational equivalence

Definition - Let P^{L}, P^{R} be two processes and $\mathscr{R}^{\mathsf{L}}$, $\mathscr{R}^{\mathsf{R}}$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😇) and denoted $(P^{\mathsf{L}}, \mathscr{R}^{\mathsf{L}}) \approx (P^{\mathsf{R}}, \mathscr{R}^{\mathsf{R}})$

New-theorem?

Given a biprocess P, and a set of restrictions $\mathscr{R}$,

$$traces_{|\mathscr{R}}(P) \downharpoonleft\!\upharpoonright \Rightarrow (\mathsf{fst}(P), \mathsf{fst}(\mathscr{R})) \approx (\mathsf{snd}(P), \mathsf{snd}(\mathscr{R})).$$

Relation with observational equivalence

Definition - Let $P^{\mathsf{L}}, P^{\mathsf{R}}$ be two processes and $\mathscr{R}^{\mathsf{L}}, \mathscr{R}^{\mathsf{R}}$ be two sets of restrictions. Observational equivalence is extended with restrictions as expected (i.e. considering only traces that satisfy restrictions 😇) and denoted $(P^{\mathsf{L}}, \mathscr{R}^{\mathsf{L}}) \approx (P^{\mathsf{R}}, \mathscr{R}^{\mathsf{R}})$

```
adebant@macbook-pro-de-alexandre-2 proverif-examples % proverif example1.pv
Biprocess 0 (that is, the initial process):
(
    {1}new n: bitstring;
    {2}new m: bitstring;
    {3}out(cpriv, choice[n,m])
) | (
    {4}in(cpriv, x: bitstring);
    {5}out(cpub, x)
)

-- Restriction not event(E(x_1)) encoded as not event2(E(x_1),E(x_1)) in biprocess 0.
-- Diff-equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Diff-equivalence is true.

--------------------------------------------------------------
Verification summary:

Query(ies):
 - Diff-equivalence is true.
Associated restriction(s):
 - Restriction not event(E(x_1)) encoded as not event2(E(x_1),E(x_1)) in biprocess 0.

--------------------------------------------------------------
```

New-theorem?

Given a biprocess

tra

Why is it false?

Why is it false?

Strange restrictions $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$

Why is it false?

Strange restrictions $\rho := \mathsf{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$ ✕ $\mathsf{fst}(\rho)$ is not properly defined!

Why is it false?

Strange restrictions $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$ ✗ fst(ρ) is not properly defined!

A bi-restriction impact both sides of the equivalence

Why is it false?

Strange restrictions $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$

✗ $\text{fst}(\rho)$ is not properly defined!

A bi-restriction impact both sides of the equivalence

$$
\begin{array}{l}
P = (\\
\quad \text{new } n;\ \text{new } m; \\
\quad \text{out}(cpriv1, \mathit{diff}[n, n]); \\
\quad \text{out}(cpriv2, \mathit{diff}[n, m]); \\
\quad) \mid (\\
\quad \text{in}(cpriv1, x); \\
\quad \text{in}(cpriv, y); \\
\quad \text{event } E(x, y); \\
\quad \text{out}(cpub, ok) \\
\quad)
\end{array}
$$

Restriction: $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{R}} = y^{\mathsf{R}}$

Why is it false?

Strange restrictions $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$ ✗ $\mathsf{fst}(\rho)$ is not properly defined!

A bi-restriction impact both sides of the equivalence

$$
\begin{array}{l}
P = (\\
\quad \text{new } n;\ \text{new } m; \\
\quad \text{out}(\mathit{cpriv}1, \mathit{diff}[n,n]); \\
\quad \text{out}(\mathit{cpriv}2, \mathit{diff}[n,m]); \\
\quad)\ |\ (\\
\quad \text{in}(\mathit{cpriv}1, x); \\
\quad \text{in}(\mathit{cpriv}, y); \\
\quad \text{event } E(x,y); \\
\quad \text{out}(\mathit{cpub}, \mathit{ok}) \\
\quad)
\end{array}
$$

$$
\begin{array}{rl}
T := & \text{out}(\mathit{cpriv}1, n) \,.\, \text{in}(\mathit{cpriv}1, n)\,. \\
& \text{out}(\mathit{cpriv}2, n) \,.\, \text{in}(\mathit{cpriv}2, n)\,. \\
& \text{event}(E(n,n)) \,.\, \text{out}(\mathit{cpub}, \mathit{ok})
\end{array}
$$

$T \in \mathit{traces}(\mathsf{fst}(P))$ and $T \vdash \mathit{true} = \mathsf{fst}(\rho)$

But $\mathit{event}(E(n,m))$ cannot be executed in $\mathsf{snd}(P)$ while satisfying $\mathsf{snd}(\rho)$

Restriction: $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{R}} = y^{\mathsf{R}}$

Why is it false?

Strange restrictions $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = x^{\mathsf{R}}$ ✗ $\mathsf{fst}(\rho)$ is not properly defined!

A bi-restriction impact both sides of the equivalence

$P = ($
 new n; new m;
 out$(\mathit{cpriv}1, \mathit{diff}[n, n])$;
 out$(\mathit{cpriv}2, \mathit{diff}[n, m])$;
 $) \mid ($
 in$(\mathit{cpriv}1, x)$;
 in(cpriv, y);
 event $E(x, y)$;
 out$(\mathit{cpub}, \mathit{ok})$
 $)$

$T :=$ out$(\mathit{cpriv}1, n)$. in$(\mathit{cpriv}1, n)$.
 out$(\mathit{cpriv}2, n)$. in$(\mathit{cpriv}2, n)$.
 event$(E(n, n))$. out$(\mathit{cpub}, \mathit{ok})$

$T \in t$

$(\mathsf{fst}(P), \emptyset) \not\approx (\mathsf{snd}(P), \mathsf{snd}(\rho))$

But $\mathit{event}(E(n, m))$ cannot be executed in $\mathsf{snd}(P)$ while satisfying $\mathsf{snd}(\rho)$

Restriction: $\rho := \text{event}(E(\mathit{diff}[x^{\mathsf{L}}, x^{\mathsf{R}}], \mathit{diff}[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{R}} = y^{\mathsf{R}}$

Why is it false?

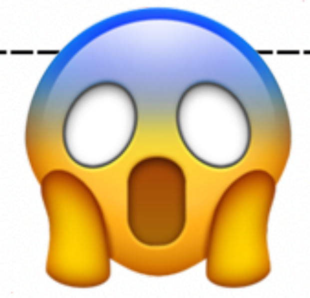
```
adebant@macbook-pro-de-alexandre-2 proverif-examples % proverif example3.pv
Biprocess 0 (that is, the initial process):
(
    {1}new n: bitstring;
    {2}new m: bitstring;
    {3}out(cpriv1, n);
    {4}out(cpriv2, choice[n,m])
) | (
    {5}in(cpriv1, x: bitstring);
    {6}in(cpriv2, y: bitstring);
    {7}event E(x,y);
    {8}out(cpub, choice[ok,ko])
)

-- Restriction event(E(choice[xL,xR],choice[yL,yR])) ==> xR = yR encoded as event2(E(xL,yL),E(xR,yR)) ==> xR = yR in biprocess 0.
-- Diff-equivalence in biprocess 0.
Translating the process into Horn clauses...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
Completing...
Termination warning: v ≠ v_1 && attacker2(v_2,v) && attacker2(v_2,v_1) -> bad
Selecting 0
Termination warning: v ≠ v_1 && attacker2(v,v_2) && attacker2(v_1,v_2) -> bad
Selecting 0
RESULT Diff-equivalence is true.

--------------------------------------------------------------
Verification summary:

Query(ies):
 - Diff-equivalence is true.
Associated restriction(s):
 - Restriction event(E(choice[xL,xR],choice[yL,yR])) ==> xR = yR encoded as event2(E(xL,yL),E(xR,yR)) ==> xR = yR in biprocess 0.

--------------------------------------------------------------
```



Strang … erly defined!

A bi-re

$P = ($
new
out(
out(
) | (
in(
in(
even
out(
)

d(ρ))

cuted in

Restriction: $\rho :=$ event$(E(diff[x^{\mathsf{L}}, x^{\mathsf{R}}], diff[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{R}} = y^{\mathsf{R}}$

What can I do now…?

I don’t know what I’m proving…

😕

Solution 1

Trust yourself 🤞

It's the most often used technique… 🙈

Solution 2

Do a paper proof to
justify each restriction…

Solution 3

Let ProVerif do the proof for you

Solution 3

Let ProVerif do the proof for you

Methodology - Given a biprocess P, and a restriction $\rho := F_1 \;\&\&\; \dots \;\&\&\; F_n \Rightarrow H^{\mathsf{L}} \;\&\&\; H^{\mathsf{R}}$ such that:

- $vars(H^{\mathsf{L}}) \subseteq vars(\mathsf{fst}(\rho))$ and $vars(H^{\mathsf{R}}) \subseteq vars(\mathsf{snd}(\rho))$
- $vars(\mathsf{fst}(\rho)) \cap vars(\mathsf{snd}(\rho)) = \varnothing$

Let ProVerif prove that: for all $tr \in traces(P)$, $tr \vdash \overline{\mathsf{fst}(\rho)}$ implies $tr \vdash \overline{\mathsf{snd}(\rho)}$ and conversely.

Solution 3

Let ProVerif do the proof for you

Methodology - Given a biprocess P, and a restriction $\rho := F_1 \ \&\& \ \dots \ \&\& \ F_n \Rightarrow H^{\mathsf{L}} \ \&\& \ H^{\mathsf{R}}$ such that:

- $vars(H^{\mathsf{L}}) \subseteq vars(\mathsf{fst}(\rho))$ and $vars(H^{\mathsf{R}}) \subseteq vars(\mathsf{snd}(\rho))$
- $vars(\mathsf{fst}(\rho)) \cap vars(\mathsf{snd}(\rho)) = \emptyset$

Let ProVerif prove that: for all $tr \in traces(P)$, $tr \vdash \overline{\mathsf{fst}(\rho)}$ implies $tr \vdash \overline{\mathsf{snd}(\rho)}$ and conversely.

Add $diff[\,\cdot\,,\cdot\,]$ each time it is necessary with fresh variables on the right side

Solution 3

Let ProVerif do the proof for you

Methodology - Given a biprocess P, and a restriction $\rho := F_1 \ \&\& \ \dots \ \&\& \ F_n \Rightarrow H^{\mathsf{L}} \ \&\& \ H^{\mathsf{R}}$ such that:

- $vars(H^{\mathsf{L}}) \subseteq vars(\mathsf{fst}(\rho))$ and $vars(H^{\mathsf{R}}) \subseteq vars(\mathsf{snd}(\rho))$
- $vars(\mathsf{fst}(\rho)) \cap vars(\mathsf{snd}(\rho)) = \varnothing$

Let ProVerif prove that: for all $tr \in traces(P)$, $tr \vdash \overline{\mathsf{fst}(\rho)}$ implies $tr \vdash \overline{\mathsf{snd}(\rho)}$ and conversely.

Add $diff[\,\cdot\,,\cdot\,]$ each time it is necessary with fresh variables on the right side

Example: $\rho := \mathsf{event}(E(diff[x^{\mathsf{L}}, x^{\mathsf{R}}], diff[y^{\mathsf{L}}, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}} \ \&\& \ x^{\mathsf{R}} = y^{\mathsf{R}}$

$\overline{\mathsf{fst}(\rho)} := \mathsf{event}(E(diff[x^{\mathsf{L}}, x_1], diff[y^{\mathsf{L}}, x_2])) \Rightarrow x^{\mathsf{L}} = y^{\mathsf{L}}$

$\overline{\mathsf{snd}(\rho)} := \mathsf{event}(E(diff[x_1, x^{\mathsf{R}}], diff[x_2, y^{\mathsf{R}}])) \Rightarrow x^{\mathsf{R}} = y^{\mathsf{R}}$

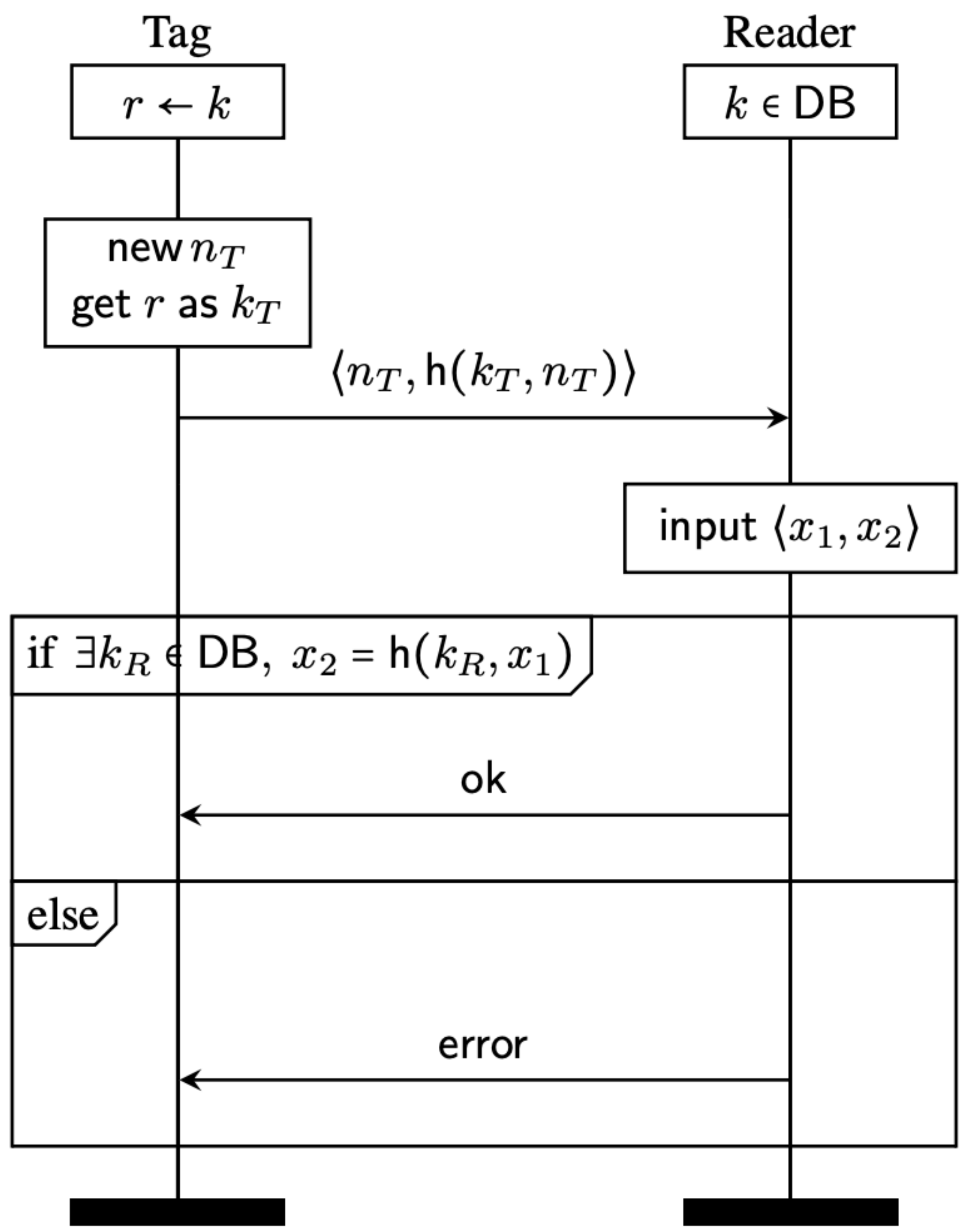
Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

$$P := !\mathit{Reader} \mid !\mathsf{new}\ k;\ \ !\mathsf{new}\ kk;\ \ \mathsf{insert}\ \mathit{DB}(\mathit{diff}[k, kk]);\ \ \mathit{Tag}(\mathit{diff}[k, kk])$$



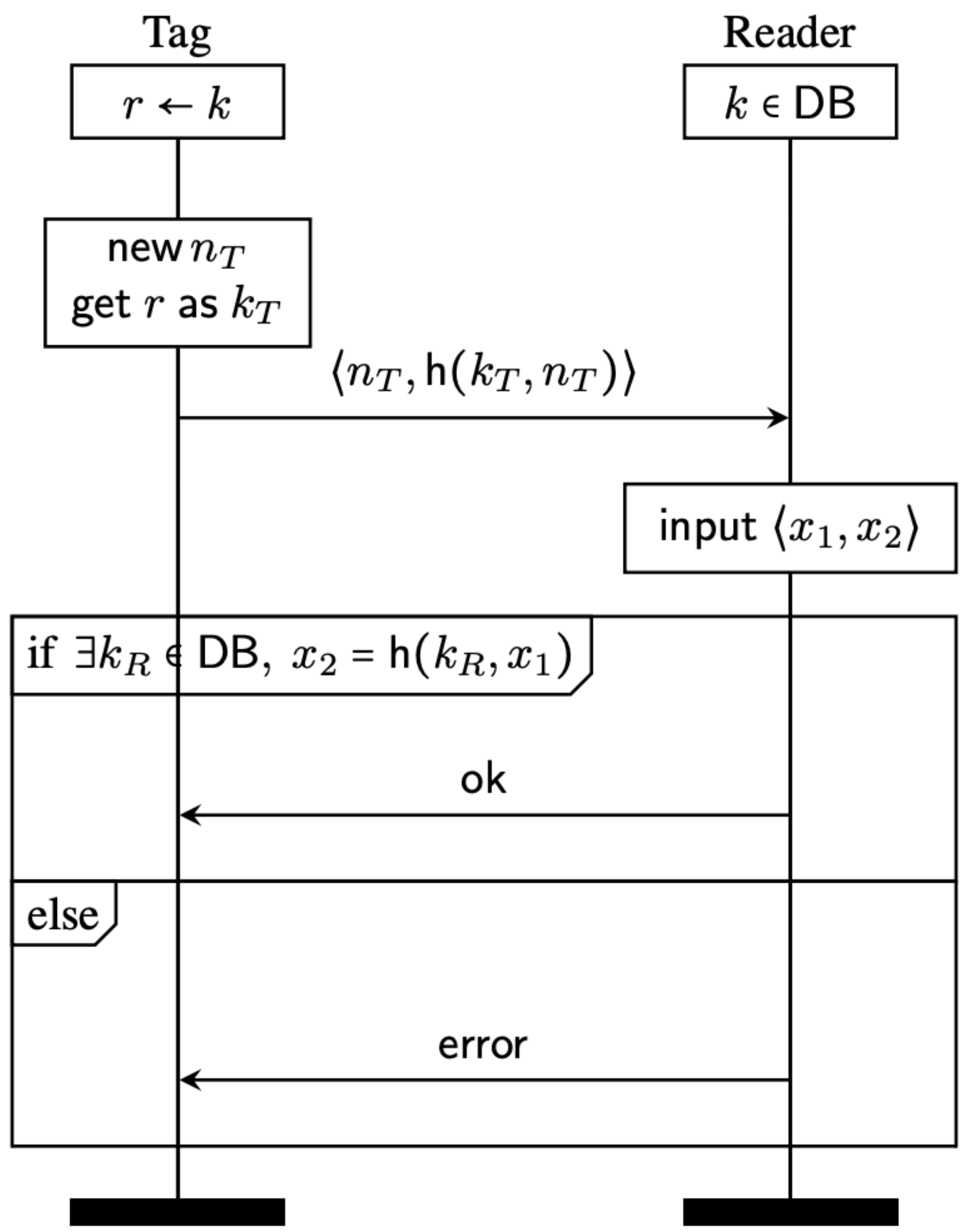
Basic Hash protocol

Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

$$P := !Reader \mid !\mathsf{new}\ k;\ \ !\mathsf{new}\ kk;\ \ \mathsf{insert}\ DB(diff[k, kk]);\ \ Tag(diff[k, kk])$$

Problem: the key k appears in many entries in $DB(\cdot)$,
$\Rightarrow$ diff-equivalence does not hold…



Basic Hash protocol

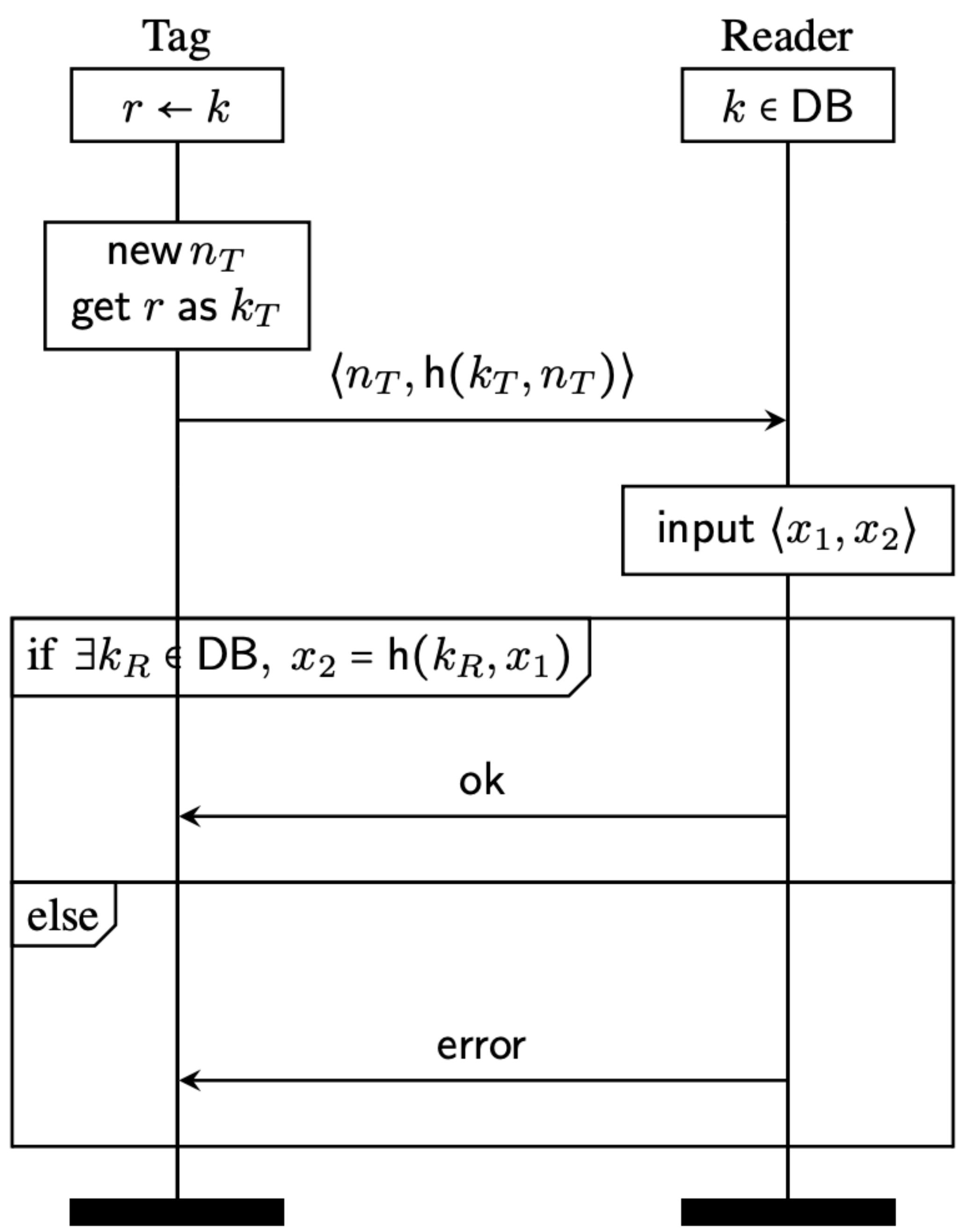
Solution 3…
is not always possible…

The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

$$P := !Reader \mid !\mathsf{new}\ k;\ \ !\mathsf{new}\ kk;\ \ \mathsf{insert}\ DB(diff[k, kk]);\ \ Tag(diff[k, kk])$$

Problem: the key k appears in many entries in $DB(\,\cdot\,)$,
$\Rightarrow$ diff-equivalence does not hold…

Solution: add a restriction to read the "good" entry when it exists



Basic Hash protocol

Solution 3…
is not always possible…

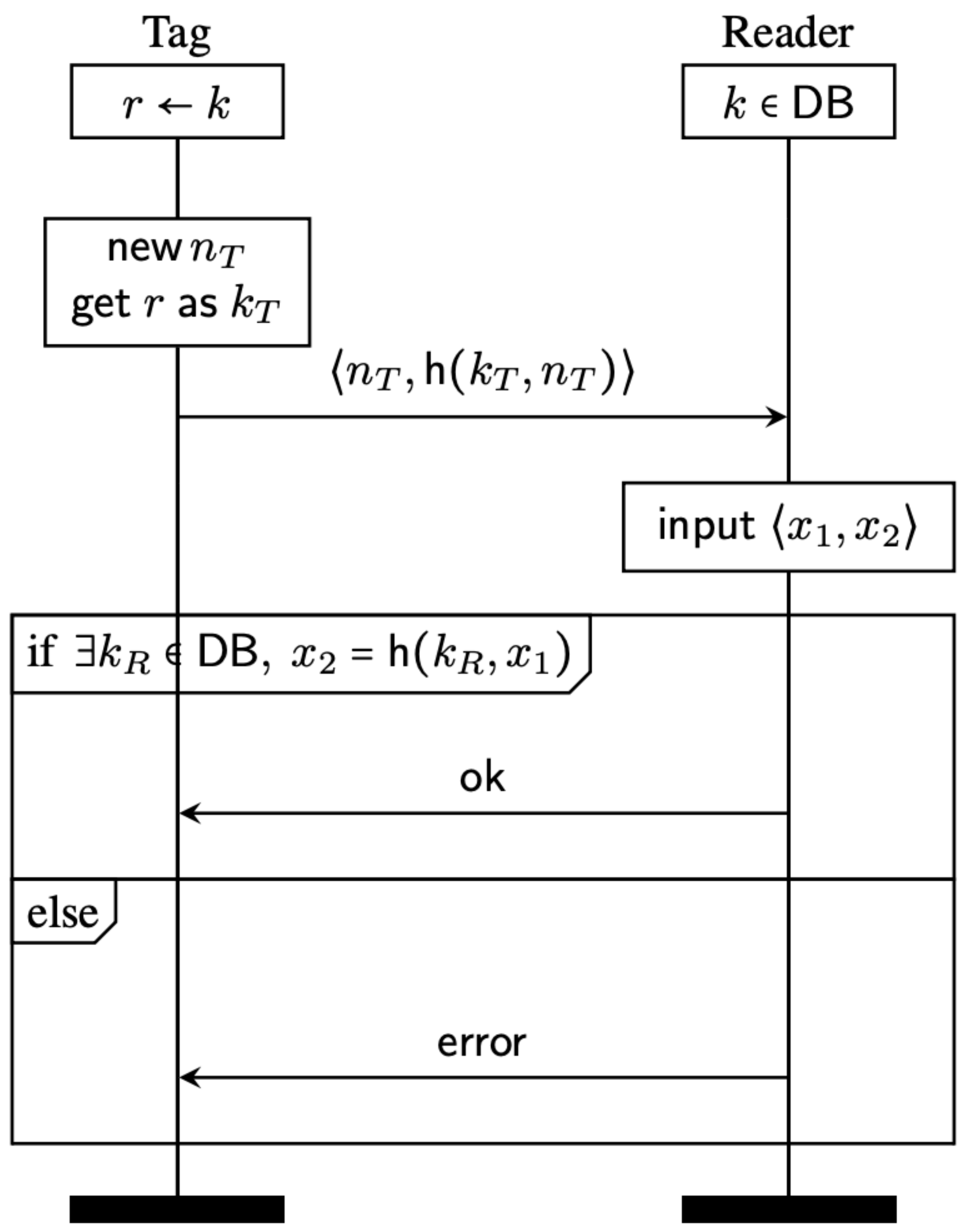
The lemma talks about a unique trace…. in many cases you want to match the first side of a trace with the second side of another trace

$$P := !Reader \mid !\text{new } k;\ \ !\text{new } kk;\ \ \text{insert } DB(diff[k, kk]);\ \ Tag(diff[k, kk])$$

Problem: the key k appears in many entries in $DB(\cdot)$,
$\Rightarrow$ diff-equivalence does not hold…

Solution: add a restriction to read the "good" entry when it exists

The previous lemma does not hold for traces using the "bad" entries



Basic Hash protocol

Solution 4

(ongoing work with Vincent and Itsaka)

Methodology

1. reinforce diff-equivalence to make it even stronger
2. adapt ProVerif procedure to make it sound w.r.t. this new definition
3. build upon Vincent and Itsaka’s approach [CSF’23] to discard false attacks

Solution 4

(ongoing work with Vincent and Itsaka)

Methodology

1. reinforce diff-equivalence to make it even stronger

2. adapt ProVerif procedure to make it sound w.r.t. this new definition

3. build upon Vincent and Itsaka's approach [CSF'23] to discard false attacks

1. Reinforce diff-equivalence

Given a trace T and a well-formed restriction ρ, $T{\downarrow\uparrow}_\rho$ if $T{\downarrow\uparrow}$ and for all $T \to P$ we have:

$$(T \to P) \vdash \overline{\mathsf{fst}(\rho)} \text{ if and only if } (T \to P) \vdash \overline{\mathsf{snd}(\rho)}$$

Solution 4

(ongoing work with Vincent and Itsaka)

2. Adapt ProVerif procedure - translation in "Horn" clauses

Given a process P, we note $\mathscr{C}(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \;\&\&\; \ldots \;\&\&\; F_n \Rightarrow H^{\mathsf{L}} \;\&\&\; H^{\mathsf{R}}$, we define:

- $C_\rho^{\mathsf{L}} = F_1 \;\&\&\; \ldots \;\&\&\; F_n \;\&\&\; H^{\mathsf{L}} \;\&\&\; \neg H^{\mathsf{R}} \Rightarrow \mathsf{bad}$
- $C_\rho^{\mathsf{R}} = F_1 \;\&\&\; \ldots \;\&\&\; F_n \;\&\&\; H^{\mathsf{R}} \;\&\&\; \neg H^{\mathsf{L}} \Rightarrow \mathsf{bad}$

We define $\mathscr{C}_{\mathscr{R}} = \{C_\rho^{\mathsf{X}} \mid \rho \in \mathscr{R}, \mathsf{X} \in \{\mathsf{L}, \mathsf{R}\}\}$

Solution 4

(ongoing work with Vincent and Itsaka)

2. Adapt ProVerif procedure - translation in "Horn" clauses

Given a process P, we note $\mathscr{C}(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \ \&\& \ \ldots \ \&\& \ F_n \Rightarrow H^{\mathsf{L}} \ \&\& \ H^{\mathsf{R}}$, we define:

- $C_\rho^{\mathsf{L}} = F_1 \ \&\& \ \ldots \ \&\& \ F_n \ \&\& \ H^{\mathsf{L}} \ \&\& \ \neg H^{\mathsf{R}} \Rightarrow \mathsf{bad}$
- $C_\rho^{\mathsf{R}} = F_1 \ \&\& \ \ldots \ \&\& \ F_n \ \&\& \ H^{\mathsf{R}} \ \&\& \ \neg H^{\mathsf{L}} \Rightarrow \mathsf{bad}$

We define $\mathscr{C}_{\mathscr{R}} = \{C_\rho^{\mathsf{X}} \mid \rho \in \mathscr{R}, \mathsf{X} \in \{\mathsf{L}, \mathsf{R}\}\}$

Lemma [soundness of the set of initial clauses]

Given a process P and a set of well-formed restrictions $\mathscr{R}$, if $\neg P \downarrow\uparrow_{\mathscr{R}}$ then bad is derivable from $\mathscr{C}(P) \cup \mathscr{C}_{\mathscr{R}}$.

Solution 4

(ongoing work with Vincent and Itsaka)

2. Adapt ProVerif procedure - translation in "Horn" clauses

Given a process P, we note $\mathscr{C}(P)$ the initial set of clauses generated by ProVerif.

Given a well-formed restriction $\rho := F_1 \ \&\& \ \dots \ \&\& \ F_n \Rightarrow H^{\mathsf{L}} \ \&\& \ H^{\mathsf{R}}$, we define:

- $C_\rho^{\mathsf{L}} = F_1 \ \&\& \ \dots \ \&\& \ F_n \ \ \&\& \ H^{\mathsf{L}} \ \&\& \ \neg H^{\mathsf{R}} \Rightarrow \mathsf{bad}$
- $C_\rho^{\mathsf{R}} = F_1 \ \&\& \ \dots \ \&\& \ F_n \ \ \&\& \ H^{\mathsf{R}} \ \&\& \ \neg H^{\mathsf{L}} \Rightarrow \mathsf{bad}$

We define $\mathscr{C}_{\mathscr{R}} = \{C_\rho^{\mathsf{X}} \mid \rho \in \mathscr{R}, \mathsf{X} \in \{\mathsf{L}, \mathsf{R}\}\}$

Lemma [soundness of the set of initial clauses]

Given a process P and a set of well-formed restrictions $\mathscr{R}$, if $\neg P \downarrow\uparrow_{\mathscr{R}}$ then bad is derivable from $\mathscr{C}(P) \cup \mathscr{C}_{\mathscr{R}}$.

Once this lemma is proved, the saturation is (almost) let unchanged, and thus its soundness proof too 😉

Solution 4

(ongoing work with Vincent and Itsaka)

3. Build upon Vincent and Itsaka's approach [CSF'23] to discard false attacks

[Cheval & Rakotonirina - CSF'23] ==> ProVerif extension to (almost) prove session equivalence

Intuition:

- either the restriction is defined to discard some matchings (e.g. Basic Hash) and they are unnecessary to prove session equivalence
  - ➡ Vincent&Itsaka extension will remove the newly reachable bad
- they are safe and bad should not be reachable

Solution 4

(ongoing work with Vincent and Itsaka)

3. Build upon Vincent and Itsaka's approach [CSF'23] to discard false attacks

[Cheval & Rakotonirina - CSF'23] ==> ProVerif extension to (almost) prove session equivalence

Intuition:

- either the restriction is defined to discard some matchings (e.g. Basic Hash) and they are unnecessary to prove session equivalence
  - ➡ Vincent&Itsaka extension will remove the newly reachable bad
- they are safe and bad should not be reachable

TODO

- adapt Vincent&Itsaka extension (i.e. adapt all the proofs…)
- extend ProVerif (or find tricks) to support $\neg H^{\mathsf{X}}$ in premise of a clause for any fact H^{X}

Conclusion

😈 **Be careful when you are using restrictions with equivalence queries…**

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side

Conclusion

😈 **Be careful when you are using restrictions with equivalence queries…**

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side

😇 **The manual of ProVerif and the long version of S&P'21 paper describe all the theory**

Everything is well-documented. Do not hesitate to open them when you're not sure about what you're proving.

Conclusion

Be careful when you are using restrictions with equivalence queries…

It is not possible to think a bi-restriction as a restriction on the left side and a restriction on the right side

The manual of ProVerif and the long version of S&P'21 paper describe all the theory

Everything is well-documented. Do not hesitate to open them when you're not sure about what you're proving.

The `improve-scope-lemma` branch brings many new features

But part of them are under-documented…