Trace equivalence via constraint solving

Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune

LSV, CNRS & ENS Cachan & INRIA Saclay Île-de-France, France

Dagstuhl - August 2011

Cryptographic protocols

PayPal™

Cryptographic protocols

- small programs designed to secure communication (*e.g.* secrecy, authentication, anonymity, ...)
- use cryptographic primitives (*e.g.* encryption, signature, ...)

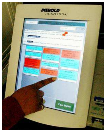
The network is unsecure!

Communications take place over a public network like the Internet.

PayPal™

Cryptographic protocols

- small programs designed to secure communication (*e.g.* secrecy, authentication, anonymity, . . .)
- use cryptographic primitives (*e.g.* encryption, signature, ...)

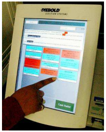


Cryptographic protocols

Cryptographic protocols

- small programs designed to secure communication (*e.g.* secrecy, authentication, anonymity, ...)
- use cryptographic primitives (*e.g.* encryption, signature, ...)

It becomes more and more important to protect our privacy.



Example: electronic passport

$\longrightarrow$ studied in [Arapinis, Chothia, Ritter & Ryan,10]

An electronic passport is a passport with an RFID tag embedded in it.

The RFID tag stores:

- the information printed on your passport,
- a JPEG copy of your picture.

Example: electronic passport

$\longrightarrow$ studied in [Arapinis, Chothia, Ritter & Ryan,10]

An electronic passport is a passport with an RFID tag embedded in it.

The RFID tag stores:

- the information printed on your passport,
- a JPEG copy of your picture.

The Basic Access Control (BAC) protocol is a key establishment protocol that has been designed to also ensure unlinkability.

ISO/IEC standard 15408

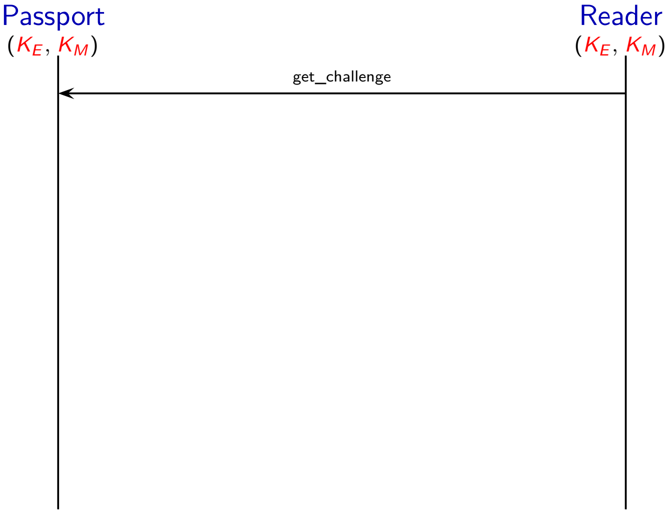
Unlinkability aims to ensure *that a user may make multiple uses of a service or resource without others being able to link these uses together.*

The electronic passport protocol

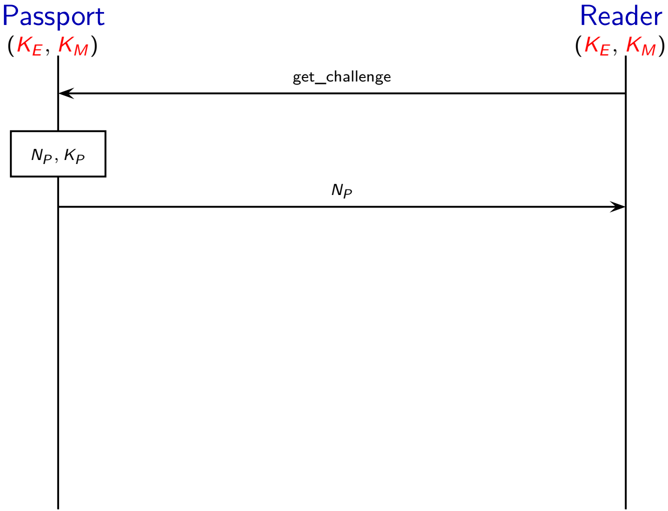
Passport
(K_E, K_M)

Reader
(K_E, K_M)

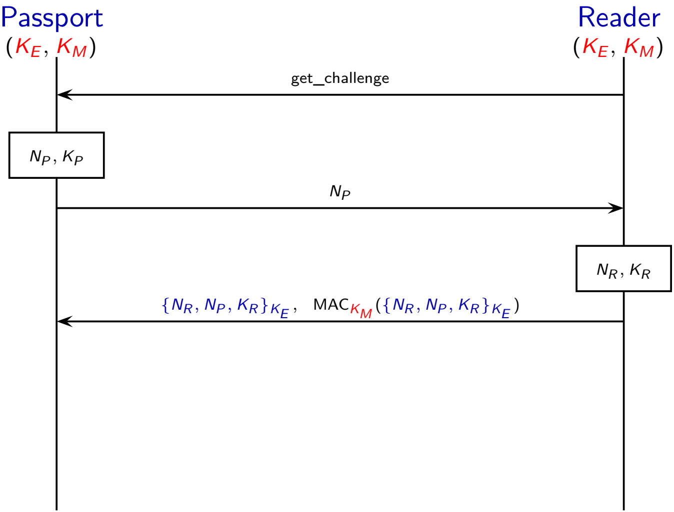
The electronic passport protocol



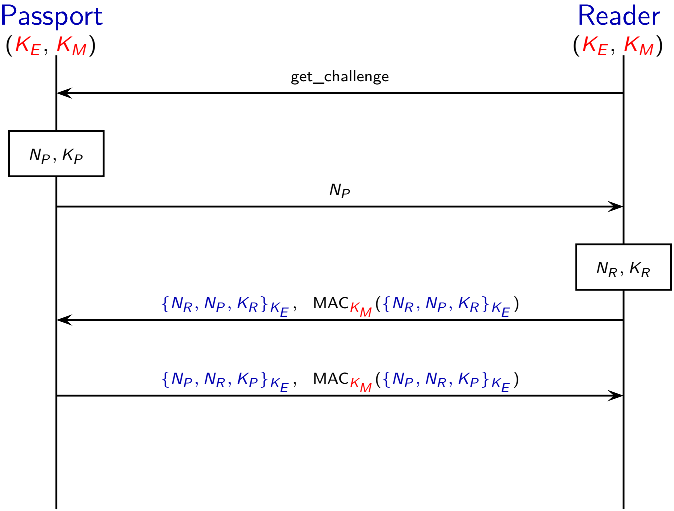
The electronic passport protocol



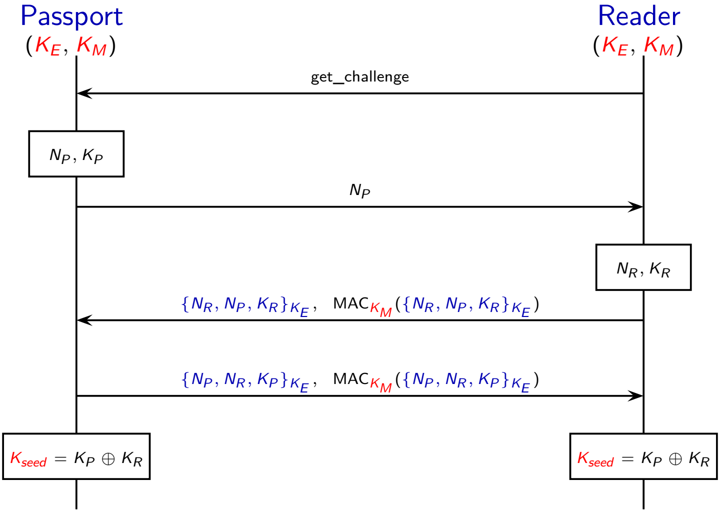
The electronic passport protocol



The electronic passport protocol



The electronic passport protocol



Privacy-type security properties

Equivalence based properties

"An observer cannot observe any difference between P and Q"

$\longrightarrow$ unlinkability, anonymity, privacy related properties in e-voting, strong secrecy, ...

Privacy-type security properties

Equivalence based properties

"An observer cannot observe any difference between P and Q"

$\longrightarrow$ unlinkability, anonymity, privacy related properties in e-voting, strong secrecy, ...

Example: E-passport protocol (unlinkability)

$$P \mid P \mid !Reader \overset{?}{\approx} P \mid P' \mid !Reader$$

Privacy-type security properties

Equivalence based properties

"An observer cannot observe any difference between P and Q"

$\longrightarrow$ unlinkability, anonymity, privacy related properties in e-voting, strong secrecy, ...

Example: E-passport protocol (unlinkability)

$$P \mid P \mid !Reader \stackrel{?}{\approx} P \mid P' \mid !Reader$$

What is the attacker able to distinguish?

- $\mathtt{new}\ k;\ \mathsf{out}(\{0\}_k) \approx \mathtt{new}\ k; \mathsf{out}(\{1\}_k)$

  $\longrightarrow$ *We assume a Dolev-Yao attacker and perfect cryptography*

Privacy-type security properties

Equivalence based properties

"An observer cannot observe any difference between P and Q"

$\longrightarrow$ unlinkability, anonymity, privacy related properties in e-voting, strong secrecy, ...

Example: E-passport protocol (unlinkability)

$$P \mid P \mid !Reader \stackrel{?}{\approx} P \mid P' \mid !Reader$$

What is the attacker able to distinguish?

- $\texttt{if } \phi \texttt{ then } P \texttt{ else } Q \approx \texttt{if } \neg\phi \texttt{ then } Q \texttt{ else } P$

  $\longrightarrow$ *He can not observe the result of a test*

Privacy-type security properties

Equivalence based properties

"An observer cannot observe any difference between P and Q"

$\longrightarrow$ unlinkability, anonymity, privacy related properties in e-voting, strong secrecy, ...

Example: E-passport protocol (unlinkability)

$$P \mid P \mid !Reader \overset{?}{\approx} P \mid P' \mid !Reader$$

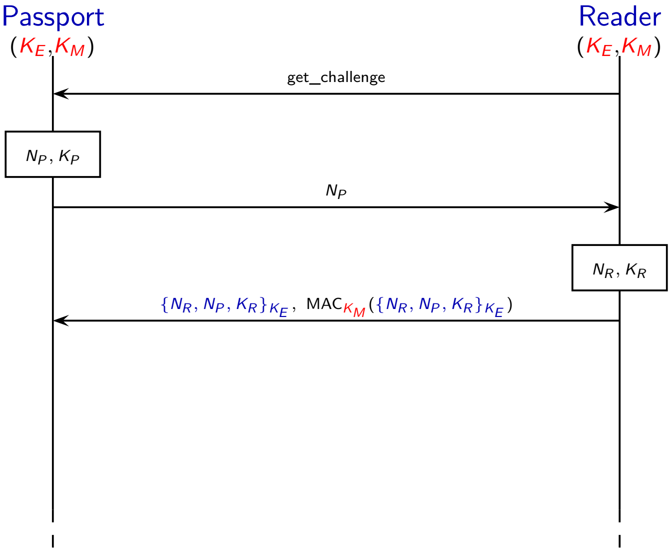
What is the attacker able to distinguish?

- $\mathsf{out}(a); (\mathsf{out}(b) + \mathsf{out}(c)) \approx \mathsf{out}(a); \mathsf{out}(b) + \mathsf{out}(b); \mathsf{out}(c)$

  $\longrightarrow$ *We consider trace equivalence (also called may-testing)*

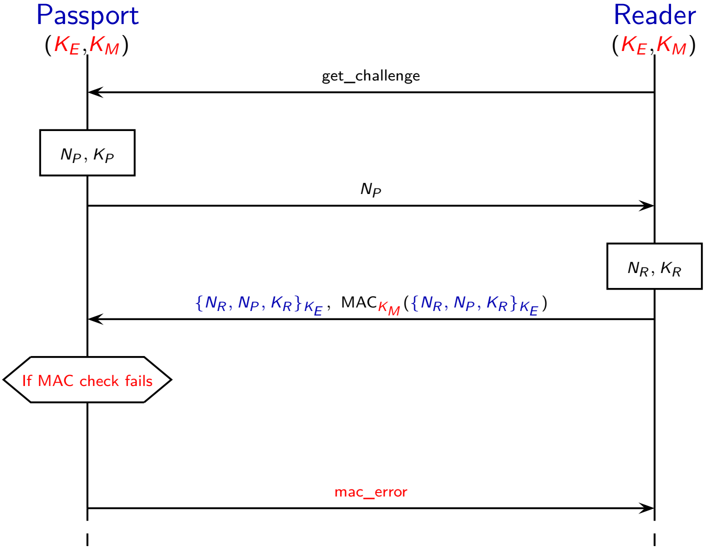
French electronic passport

$\longrightarrow$ the passport must reply to all received messages.



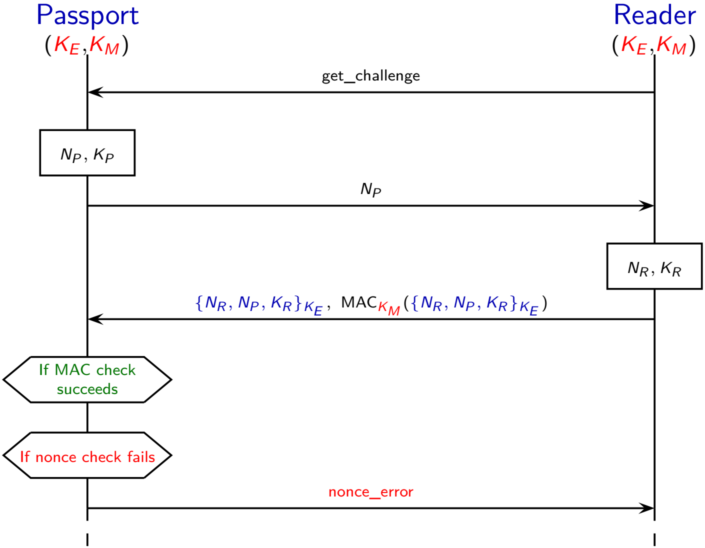
French electronic passport

$\longrightarrow$ the passport must reply to all received messages.



French electronic passport

$\longrightarrow$ the passport must reply to all received messages.



An attack on the French passport [Chothia & Smirnov, 10]

Attack against unlinkability

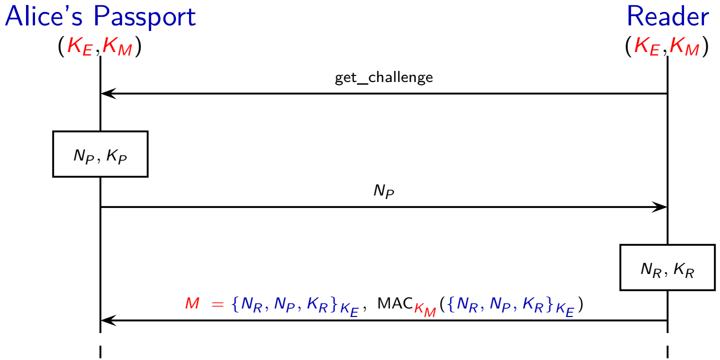
An attacker can track a French passport, provided he has eavesdropped a successful authentication.

An attack on the French passport [Chothia & Smirnov, 10]

Attack against unlinkability

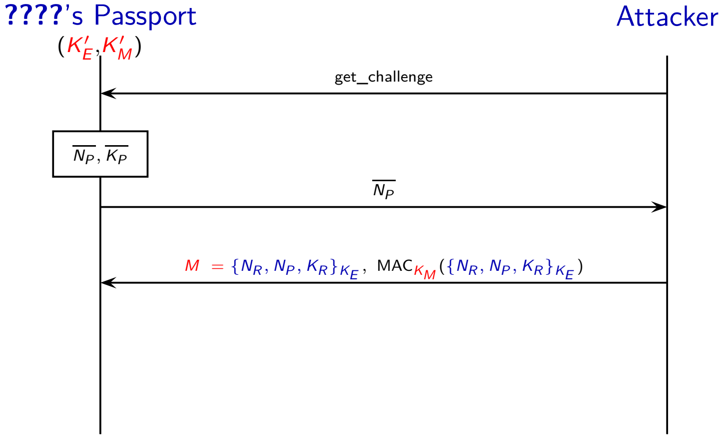
An attacker can track a French passport, provided he has eavesdropped a successful authentication.

Part 1 of the attack. The attacker eavesdropes on Alice using her passport and records message M.



Part 2 of the attack.

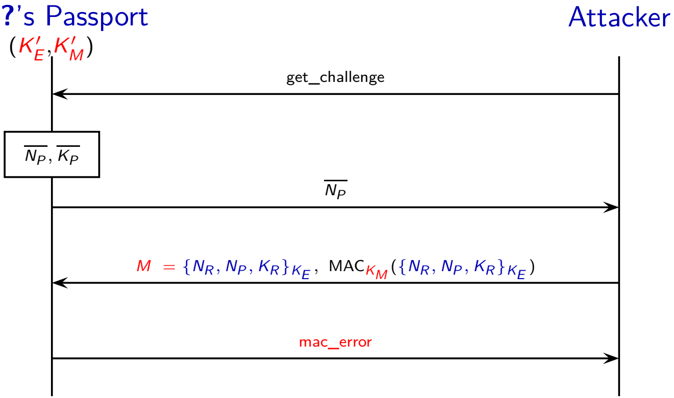
The attacker replays the message M and checks the error code he receives.



An attack on the French passport [Chothia & Smirnov, 10]

Part 2 of the attack.

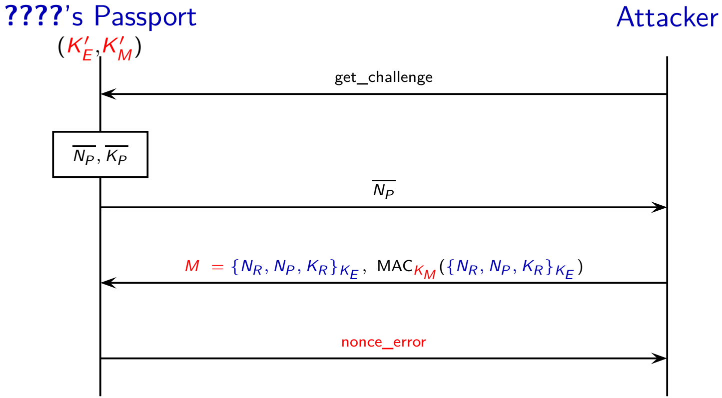
The attacker replays the message M and checks the error code he receives.



$\Longrightarrow$ MAC check failed $\Longrightarrow$ $K'_M \neq K_M$ $\Longrightarrow$ **????** is not Alice

Part 2 of the attack.

The attacker replays the message M and checks the error code he receives.



$\Longrightarrow$ MAC check succeeded $\Longrightarrow$ $K'_M = K_M$ $\Longrightarrow$ **????** is Alice

State of the art in a nutshell

for analysing equivalence-based security properties

State of the art in a nutshell

for analysing equivalence-based security properties

Bounded number of sessions
e.g. [Baudet, 05], [Dawson & Tiu, 10], [Chevalier & Rusinowitch, 10], ...

$\longrightarrow$ this allows us to decide trace equivalence between simple processes with trivial else branches. [Cortier & Delaune, 09]

State of the art in a nutshell

for analysing equivalence-based security properties

Bounded number of sessions

e.g. [Baudet, 05], [Dawson & Tiu, 10], [Chevalier & Rusinowitch, 10], ...

$\longrightarrow$ this allows us to decide trace equivalence between simple processes with trivial else branches. [Cortier & Delaune, 09]

Unbounded number of sessions [Blanchet, Abadi & Fournet, 05]

ProVerif tool [Blanchet, 01] `http://www.proverif.ens.fr/`

- + unbounded number of sessions; various cryptographic primitives;
- − termination is not guaranteed; diff-equivalence (too strong)

$\longrightarrow$ ProSwapper extension [Smyth, 10]

State of the art in a nutshell

for analysing equivalence-based security properties

Bounded number of sessions

e.g. [Baudet, 05], [Dawson & Tiu, 10], [Chevalier & Rusinowitch, 10], ...

$\longrightarrow$ this allows us to decide trace equivalence between simple processes with trivial else branches. [Cortier & Delaune, 09]

Unbounded number of sessions [Blanchet, Abadi & Fournet, 05]

ProVerif tool [Blanchet, 01] `http://www.proverif.ens.fr/`

- $+$ unbounded number of sessions; various cryptographic primitives;
- $-$ termination is not guaranteed; diff-equivalence (too strong)

$\longrightarrow$ ProSwapper extension [Smyth, 10]

$\longrightarrow$ None of these results is able to analyse the e-passport protocol.

Our contribution

Main result

A procedure for deciding trace equivalence for a large class of processes.

Our contribution

Main result

A procedure for deciding trace equivalence for a large class of processes.

Our class of processes:

- + non-trivial else branches, private channels, and non-deterministic choice;
- – but no replication, and a fixed set of cryptographic primitives (signature, encryption, hash function, mac).

Our contribution

Main result

A procedure for deciding trace equivalence for a large class of processes.

Our class of processes:

- + non-trivial else branches, private channels, and non-deterministic choice;
- – but no replication, and a fixed set of cryptographic primitives (signature, encryption, hash function, mac).

Some applications:

- unlinkability in RFID protocols (*e.g.* e-passport protocol);
- anonymity/privacy (*e.g.* private authentication protocols [Abadi & Fournet, 04]).

Outline

1. Introduction

2. From trace equivalence to symbolic equivalence

3. Deciding symbolic equivalence using constraint solving techniques

4. Conclusion

Outline

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}\\
\qquad\quad \mathtt{if}\ N_P = x'_P\\
\qquad\qquad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}\\
\qquad\qquad\qquad\quad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})\\
\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})
\end{array}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= get\_challenge); \texttt{new}\ N_P; \texttt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\texttt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \texttt{then let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ in\\
\qquad\quad \texttt{if}\ N_P = x'_P\\
\qquad\qquad \texttt{then let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ in\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \texttt{else}\ \mathrm{out}(nonce\_error)\\
\quad \texttt{else}\ \mathrm{out}(mac\_error)
\end{array}
$$

$$T_0 \overset{?}{\vdash} get\_challenge$$

$$\Phi = T_0;$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= get\_challenge); \mathtt{new}\ N_P; \mathtt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ in\\
\qquad\quad \mathtt{if}\ N_P = x'_P\\
\qquad\qquad \mathtt{then}\,\mathtt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ in\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \mathtt{else}\ \mathrm{out}(nonce\_error)\\
\quad \mathtt{else}\ \mathrm{out}(mac\_error)
\end{array}
$$

$$T_0 \overset{?}{\vdash} get\_challenge$$
$$T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle$$

$$\Phi = T_0;$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}\\
\qquad\quad \mathtt{if}\ N_P = x'_P\\
\qquad\qquad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})\\
\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})
\end{array}
$$

$$
\begin{array}{c}
T_0 \overset{?}{\vdash} \mathit{get\_challenge}\\
T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle\\
z_M \overset{?}{\neq} \mathrm{MAC}_{K_M}(z_E)\\
\hline
\Phi = T_0;
\end{array}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}\\
\qquad\quad \mathtt{if}\ N_P = x'_P\\
\qquad\qquad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})\\
\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})
\end{array}
$$

$$
\frac{\begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{\neq} \mathrm{MAC}_{K_M}(z_E) \end{array}}{\Phi = T_0; \mathit{mac\_error}}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

From processes to constraint systems

Passport P - (K_E, K_M)

$$\begin{array}{l}
\mathrm{in}(= \mathit{get_challenge}); \texttt{new}\ N_P; \texttt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\texttt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \texttt{then let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}\\
\qquad \texttt{if}\ N_P = x'_P\\
\qquad\quad \texttt{then let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}\\
\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\quad \texttt{else}\ \mathrm{out}(\mathit{nonce_error})\\
\quad \texttt{else}\ \mathrm{out}(\mathit{mac_error})
\end{array}$$

$$\frac{\begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{\neq} \mathrm{MAC}_{K_M}(z_E) \end{array}}{\Phi = T_0; \mathit{mac\_error}}$$

$$\longrightarrow \mathcal{C}_{\mathrm{mac}}$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$\mathrm{in}(= get\_challenge); \mathtt{new}\ N_P; \mathtt{new}\ K_P;$
$\mathrm{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)$
$\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ in$
$\qquad \mathtt{if}\ N_P = x'_P$
$\qquad\quad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P \rangle\}_{K_E}\ in$
$\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m) \rangle)$
$\qquad\quad \mathtt{else}\ \mathrm{out}(nonce\_error)$
$\quad \mathtt{else}\ \mathrm{out}(mac\_error)$

$$T_0 \overset{?}{\vdash} get\_challenge$$

$$\Phi = T_0;$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;$
$\mathrm{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)$
$\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}$
$\qquad \mathtt{if}\ N_P = x'_P$
$\qquad\quad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P \rangle\}_{K_E}\ \mathit{in}$
$\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m) \rangle)$
$\qquad\quad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})$
$\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})$

$$T_0 \overset{?}{\vdash} \mathit{get\_challenge}$$
$$T_0 \overset{?}{\vdash} \langle z_E, z_M \rangle$$

$$\Phi = T_0;$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;$
$\mathrm{in}(\langle z_E, z_M\rangle);$
$\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)$
$\quad \mathtt{then\ let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}$
$\qquad \mathtt{if}\ N_P = x'_P$
$\qquad\quad \mathtt{then\ let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}$
$\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)$
$\qquad\quad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})$
$\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})$

$$
\frac{\begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{=} \mathrm{MAC}_{K_M}(z_E) \\ \langle x_R, x'_P, y_R\rangle = \mathrm{dec}(z_E, K_E) \\ N_P \overset{?}{\neq} x'_P \end{array}}{\Phi = T_0;}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= get\_challenge); \texttt{new}\ N_P; \texttt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\texttt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \texttt{then}\ \texttt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ in\\
\qquad\quad \texttt{if}\ N_P = x'_P\\
\qquad\qquad \texttt{then}\ \texttt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ in\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \texttt{else}\ \mathrm{out}(nonce\_error)\\
\quad \texttt{else}\ \mathrm{out}(mac\_error)
\end{array}
$$

$$
\begin{array}{c}
T_0 \overset{?}{\vdash} get\_challenge\\
T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle\\
z_M \overset{?}{=} \mathrm{MAC}_{K_M}(z_E)\\
\langle x_R, x'_P, y_R\rangle = \mathrm{dec}(z_E, K_E)\\
N_P \overset{?}{\neq} x'_P\\
\hline
\Phi = T_0; nonce\_error
\end{array}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;$
$\mathrm{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)$
$\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}$
$\qquad \mathtt{if}\ N_P = x'_P$
$\qquad\quad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P \rangle\}_{K_E}\ \mathit{in}$
$\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m) \rangle)$
$\qquad\quad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})$
$\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})$

$$\frac{\begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M \rangle \\ z_M \overset{?}{=} \mathrm{MAC}_{K_M}(z_E) \\ \langle x_R, x'_P, y_R \rangle = \mathrm{dec}(z_E, K_E) \\ N_P \overset{?}{\neq} x'_P \end{array}}{\Phi = T_0; \mathit{nonce\_error}}$$

$$\longrightarrow \mathcal{C}_{\mathrm{nonce}}$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};\ \mathcal{C}_{\mathrm{nonce}};$$

From processes to constraint systems

Passport P - (K_E, K_M)

$$
\begin{array}{l}
\mathrm{in}(= \mathit{get\_challenge}); \mathtt{new}\ N_P; \mathtt{new}\ K_P;\\
\mathrm{in}(\langle z_E, z_M\rangle);\\
\mathtt{if}\ z_M = \mathrm{MAC}_{K_M}(z_E)\\
\quad \mathtt{then}\ \mathtt{let}\ (x_R, x'_P, y_R) = \mathrm{dec}(z_E, K_E)\ \mathit{in}\\
\qquad\quad \mathtt{if}\ N_P = x'_P\\
\qquad\qquad \mathtt{then}\ \mathtt{let}\ m = \{\langle N_P, x_R, K_P\rangle\}_{K_E}\ \mathit{in}\\
\qquad\qquad\qquad \mathrm{out}(\langle m, \mathrm{MAC}_{K_M}(m)\rangle)\\
\qquad\qquad \mathtt{else}\ \mathrm{out}(\mathit{nonce\_error})\\
\quad \mathtt{else}\ \mathrm{out}(\mathit{mac\_error})
\end{array}
$$

Once an interleaving of symbolic actions has been fixed (*e.g.* in; in; out), we generate the associated constraint systems:

$$\mathcal{C}_{\mathrm{mac}};\ \mathcal{C}_{\mathrm{nonce}};\ \ldots$$

Trace equivalence via constraint solving

To check whether $P \approx P'$, we have to check whether

$\Sigma \approx_s \Sigma'$ for all sequence of symbolic actions (*e.g.* in;in;out).

Trace equivalence via constraint solving

To check whether $P \approx P'$, we have to check whether

$$\Sigma \approx_s \Sigma' \text{ for all sequence of symbolic actions (\textit{e.g.} in;in;out).}$$

Symbolic equivalence $\Sigma \approx_s \Sigma'$

- for all $\mathcal{C} \in \Sigma$ for all $(\sigma, \theta) \in \mathsf{Sol}(\mathcal{C})$, there exists $\mathcal{C}' \in \Sigma'$ such that:
$$(\sigma', \theta) \in \mathsf{Sol}(\mathcal{C}') \quad \text{and} \quad \Phi\sigma \sim \Phi'\sigma' \text{ (static equivalence).}$$
- and conversely

Trace equivalence via constraint solving

To check whether $P \approx P'$, we have to check whether

$$\Sigma \approx_s \Sigma' \text{ for all sequence of symbolic actions (\textit{e.g.} in;in;out).}$$

Symbolic equivalence $\Sigma \approx_s \Sigma'$

- for all $\mathcal{C} \in \Sigma$ for all $(\sigma, \theta) \in \mathsf{Sol}(\mathcal{C})$, there exists $\mathcal{C}' \in \Sigma'$ such that:

$$(\sigma', \theta) \in \mathsf{Sol}(\mathcal{C}') \quad \text{and} \quad \Phi\sigma \sim \Phi'\sigma' \text{ (static equivalence).}$$

- and conversely

Going back to the E-passport example

Among others, we have to check whether

$$\{\mathcal{C}_{\mathsf{mac}};\ \mathcal{C}_{\mathsf{nonce}};\ldots\} \overset{?}{\approx}_s \{\mathcal{C}'_{\mathsf{mac}};\mathcal{C}'_{\mathsf{nonce}};\ldots\}$$

where $\mathcal{C}'_{\mathsf{mac}}$, $\mathcal{C}'_{\mathsf{nonce}}$, ... are the counterparts of $\mathcal{C}_{\mathsf{mac}}$, $\mathcal{C}_{\mathsf{nonce}}$, ... in which K_E and K_M have been replaced by K'_E and K'_M.

French passport (1/2)

$$\{\mathcal{C}_{\mathsf{mac}};\ \mathcal{C}_{\mathsf{nonce}};\ \ldots\} \overset{?}{\approx}_s \{\mathcal{C}'_{\mathsf{mac}};\mathcal{C}'_{\mathsf{nonce}};\ldots\}$$

when T_0 contains $\langle\{\overline{N_R},\overline{N_P},\overline{K_R}\}_{K_E},\ \mathsf{MAC}_{K_M}(\{\overline{N_R},\overline{N_P},\overline{K_R}\}_{K_E})\rangle$

$\longrightarrow$ the answer should be no

French passport (1/2)

$$\{\mathcal{C}_{\mathsf{mac}};\ \mathcal{C}_{\mathsf{nonce}};\ \ldots\} \overset{?}{\approx}_s \{\mathcal{C}'_{\mathsf{mac}}; \mathcal{C}'_{\mathsf{nonce}}; \ldots\}$$

when T_0 contains $\langle \{\overline{N_R}, \overline{N_P}, \overline{K_R}\}_{K_E},\ \mathsf{MAC}_{K_M}(\{\overline{N_R}, \overline{N_P}, \overline{K_R}\}_{K_E})\rangle$

$\longrightarrow$ the answer should be no

$$\mathcal{C}_{\mathsf{nonce}} = \left\{ \begin{array}{c} T_0 \overset{?}{\vdash} get\_challenge \\ T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{=} \mathsf{MAC}_{K_M}(z_E) \\ \langle x_R, x'_P, y_R\rangle = \mathsf{dec}(z_E, K_E) \\ N_P \overset{?}{\neq} x'_P \\ \hline \Phi = T_0; nonce\_error \end{array} \right.$$

$\longrightarrow$ A solution for $\mathcal{C}_{\mathsf{nonce}}$ consists of replaying the message in T_0.

French passport (2/2)

If the attacker performed this replay, what will happen in the other side?

$$\{\mathcal{C}'_{\mathsf{mac}};\ \mathcal{C}'_{\mathsf{nonce}};\ \ldots\}$$

$\longrightarrow$ this computation does not lead to a solution for constraint system that contains $z_M = \mathsf{MAC}_{K'_M}(z_E)$.

French passport (2/2)

If the attacker performed this replay, what will happen in the other side?

$$\{\mathcal{C}'_{\mathsf{mac}};\ \mathcal{C}'_{\mathsf{nonce}};\ \ldots\}$$

$\longrightarrow$ this computation does not lead to a solution for constraint system that contains $z_M = \mathsf{MAC}_{K'_M}(z_E)$.

What about the constraint system $\mathcal{C}'_{\mathsf{mac}}$?

$$\mathcal{C}'_{\mathsf{mac}} = \begin{cases} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M \rangle \\ z_M \overset{?}{\neq} \mathsf{MAC}_{K'_M}(z_E) \\ \hline \Phi' = T_0; \mathit{mac\_error} \end{cases}$$

French passport (2/2)

If the attacker performed this replay, what will happen in the other side?

$$\{\mathcal{C}'_{\mathsf{mac}};\ \mathcal{C}'_{\mathsf{nonce}};\ \ldots\}$$

$\longrightarrow$ this computation does not lead to a solution for constraint system that contains $z_M = \mathsf{MAC}_{K'_M}(z_E)$.

What about the constraint system $\mathcal{C}'_{\mathsf{mac}}$?

$$\mathcal{C}'_{\mathsf{mac}} = \begin{cases} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M \rangle \\ z_M \overset{?}{\neq} \mathsf{MAC}_{K'_M}(z_E) \\ \hline \Phi' = T_0; \mathit{mac\_error} \end{cases}$$

$\longrightarrow$ this computation leads to a solution for $\mathcal{C}'_{\mathsf{mac}}$ but the resulting sequence of messages Φ and Φ' are not in static equivalence.

Related work on symbolic equivalence

Several works have already been done:

- for subterm convergent equational theories:

  [Baudet,05]; [Chevalier & Rusinowitch,10]

  $\longrightarrow$ does not lead to a practical algorithm

- for a fixed set of cryptographic primitives:

  [Dawson & Tiu,10]; [Cheval, Comon-Lundh & Delaune,10]

  $\longrightarrow$ those algorithms have been implemented

Related work on symbolic equivalence

Several works have already been done:

- for subterm convergent equational theories:
  [Baudet,05]; [Chevalier & Rusinowitch,10]
  $\longrightarrow$ does not lead to a practical algorithm
- for a fixed set of cryptographic primitives:
  [Dawson & Tiu,10]; [Cheval, Comon-Lundh & Delaune,10]
  $\longrightarrow$ those algorithms have been implemented

Two main limitations

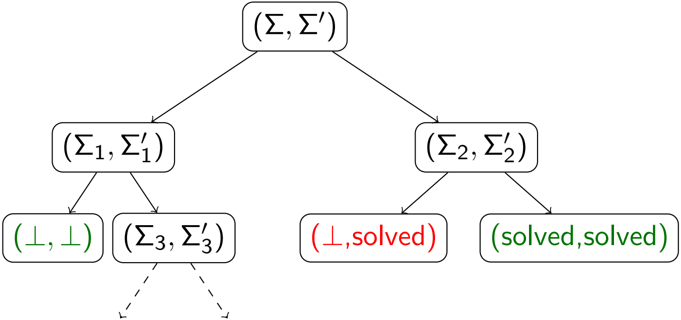
- positive constraint systems only;
- symbolic equivalence between two constraint systems (and not sets of constraint systems)

Outline

1. Introduction
2. From trace equivalence to symbolic equivalence
3. Deciding symbolic equivalence using constraint solving techniques
4. Conclusion

Our procedure in a nutshell

Main idea of our procedure: We rewrite pairs of sets of constraint systems until a trivial failure or a trivial success is found.



Our simplification rules

We propose a finite set of simplification rules that transform a constraint system into two constraint system.

Example: the Cons simplification rule

$$\mathcal{C} = \begin{cases} \dots \\ T \overset{?}{\vdash} \mathsf{enc}(u_1, u_2) \\ \dots \end{cases}$$

$$\mathcal{C}_1 = \begin{cases} \dots \\ T \overset{?}{\vdash} u_1;\ T \overset{?}{\vdash} u_2 \\ \dots \end{cases} \qquad \mathcal{C}_2 = \begin{cases} \dots \\ T \overset{?}{\vdash} \mathsf{enc}(u_1, u_2) \\ \text{using a recipe } \zeta \text{ such} \\ \text{that } \mathsf{Top}(\zeta) \neq \mathsf{enc} \\ \dots \end{cases}$$

$\longrightarrow$ We have also an Axiom rule and a Dest rule.

Step 1: reaching a constraint system in pre-solved form

$\longrightarrow$ the CONS, DEST, and AXIOM rules allow us to reach a pre-solved form, *i.e.* a system of the form

$$\mathcal{C} = \begin{cases} T_1 \overset{?}{\vdash} x_1 \\ T_2 \overset{?}{\vdash} x_2 \\ \ldots \\ T_n \overset{?}{\vdash} x_n \end{cases} + \text{ some disequalities}$$

$\longrightarrow$ this is sufficient to decide satisfiability but

Step 1: reaching a constraint system in pre-solved form

$\longrightarrow$ the $\textsc{Cons}$, $\textsc{Dest}$, and $\textsc{Axiom}$ rules allow us to reach a pre-solved form, *i.e.* a system of the form

$$\mathcal{C} = \begin{cases} T_1 \overset{?}{\vdash} x_1 \\ T_2 \overset{?}{\vdash} x_2 \\ \ldots \\ T_n \overset{?}{\vdash} x_n \end{cases} + \text{ some disequalities}$$

$\longrightarrow$ this is sufficient to decide satisfiability but not sufficient to decide symbolic equivalence.

Step 1: reaching a constraint system in pre-solved form

$\longrightarrow$ the CONS, DEST, and AXIOM rules allow us to reach a pre-solved form, *i.e.* a system of the form

$$\mathcal{C} = \begin{cases} T_1 \overset{?}{\vdash} x_1 \\ T_2 \overset{?}{\vdash} x_2 \\ \ldots \\ T_n \overset{?}{\vdash} x_n \end{cases} + \text{ some disequalities}$$

$\longrightarrow$ this is sufficient to decide satisfiability but not sufficient to decide symbolic equivalence. Why?

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} x \\ T_0 \overset{?}{\vdash} y \end{cases} + \quad y \overset{?}{\neq} \mathsf{enc}(x, x) \qquad\qquad \mathcal{C}' = \begin{cases} T'_0 \overset{?}{\vdash} x' \\ T'_0 \overset{?}{\vdash} y' \end{cases}$$

$\mathcal{C}$ and $\mathcal{C}'$ are in pre-solved form but they are not in symbolic equivalence.

Step 2: dealing with disequations

For these we have some specific rules to:

- simplify the disequations; and
- "match" the disequations of each constraint system.

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \overset{?}{\vdash} x \\ T_0 \overset{?}{\vdash} y \end{array} \right. \quad + \quad y \overset{?}{\neq} \mathsf{enc}(x, x) \qquad\qquad \mathcal{C}' = \left\{ \begin{array}{l} T_0' \overset{?}{\vdash} x' \\ T_0' \overset{?}{\vdash} y' \end{array} \right.$$

Step 2: dealing with disequations

For these we have some specific rules to:

- simplify the disequations; and
- "match" the disequations of each constraint system.

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} x \\ T_0 \overset{?}{\vdash} y \end{cases} + \quad y \overset{?}{\neq} \mathsf{enc}(x, x) \qquad\qquad \mathcal{C}' = \begin{cases} T_0' \overset{?}{\vdash} x' \\ T_0' \overset{?}{\vdash} y' \end{cases}$$

Apply a rule to split each constraint system into two constraint systems:

$$\begin{array}{ccc} & \boxed{(\mathcal{C}; \mathcal{C}')} & \\ \swarrow & & \searrow \\ \boxed{(\perp;\ \mathcal{C}' + y' = \mathsf{enc}(x', x'))} & & \boxed{(\mathcal{C};\ \mathcal{C}' + y' \neq \mathsf{enc}(x', x'))} \end{array}$$

Step 3: dealing with static equivalence

$\longrightarrow$ The two resulting sequences of messages have to be indistinguishable.

$$\mathcal{C} = \begin{cases} a;\ \mathrm{pub}(b) \overset{?}{\vdash} x \\ \Phi = a;\ \mathrm{pub}(b);\ \{x\}_{\mathrm{pub}(b)} \end{cases} \quad \mathcal{C}' = \begin{cases} a';\ \mathrm{pub}(b'); \overset{?}{\vdash} x' \\ \Phi' = a';\ \mathrm{pub}(b');\ \{x'\}_{\mathrm{pub}(c')} \end{cases}$$

Step 3: dealing with static equivalence

$\longrightarrow$ The two resulting sequences of messages have to be indistinguishable.

$$\mathcal{C} = \left\{ \begin{array}{l} a;\ \mathrm{pub}(b) \overset{?}{\vdash} x \\ \Phi = a;\ \mathrm{pub}(b);\ \{x\}_{\mathrm{pub}(b)} \end{array} \right. \quad \mathcal{C}' = \left\{ \begin{array}{l} a';\ \mathrm{pub}(b'); \overset{?}{\vdash} x' \\ \Phi' = a';\ \mathrm{pub}(b');\ \{x'\}_{\mathrm{pub}(c')} \end{array} \right.$$

$\longrightarrow$ Rules $\textsc{Eq}$ and $\textsc{Ded-subterm}$.

Step 3: dealing with static equivalence

$\longrightarrow$ The two resulting sequences of messages have to be indistinguishable.

$$\mathcal{C} = \begin{cases} a;\ \mathrm{pub}(b) \overset{?}{\vdash} x \\ \Phi = a;\ \mathrm{pub}(b);\ \{x\}_{\mathrm{pub}(b)} \end{cases} \quad \mathcal{C}' = \begin{cases} a';\ \mathrm{pub}(b'); \overset{?}{\vdash} x' \\ \Phi' = a';\ \mathrm{pub}(b');\ \{x'\}_{\mathrm{pub}(c')} \end{cases}$$

$\longrightarrow$ Rules $\textsc{Eq}$ and $\textsc{Ded-subterm}$.

Applying $\textsc{Ded-subterm}$ on $(\mathcal{C}, \mathcal{C}')$ will generate $(\mathcal{C}_1; \mathcal{C}'_1)$ (on one branch):

$$\begin{cases} \dots \\ a; \mathrm{pub}(b), \{x\}_{\mathrm{pub}(b)} \overset{?}{\vdash} x \\ a; \mathrm{pub}(b); \{x\}_{\mathrm{pub}(b)} \overset{?}{\vdash} \mathrm{pub}(b) \end{cases} \qquad \begin{cases} \dots \\ a'; \mathrm{pub}(b'), \{x'\}_{\mathrm{pub}(c')} \overset{?}{\vdash} x' \\ a'; \mathrm{pub}(b'); \{x'\}_{\mathrm{pub}(c')} \overset{?}{\vdash} \mathrm{pub}(c') \end{cases}$$

Main results on our algorithm

Termination

Applying blindly the simplification rules does not terminate but there is a particular strategy $\mathcal{S}$ that allows us to ensure termination.

Main results on our algorithm

Termination

Applying blindly the simplification rules does not terminate but there is a particular strategy $\mathcal{S}$ that allows us to ensure termination.

Soundness/Completeness

Let (Σ_0, Σ_0') be pair of sets of constraint systems, and consider a binary tree obtained by applying our simplification rule following a strategy $\mathcal{S}$.

1. soundness: If all leaves of the tree are labeled with $(\perp, \perp)$ or $(\mathit{solved}, \mathit{solved})$, then $\Sigma_0 \approx_s \Sigma_0'$.
2. completeness: if $\Sigma_0 \approx_s \Sigma_0'$, then all leaves of the tree are labeled with $(\perp, \perp)$ or $(\mathit{solved}, \mathit{solved})$,.

Main results on our algorithm

Termination

Applying blindly the simplification rules does not terminate but there is a particular strategy $\mathcal{S}$ that allows us to ensure termination.

Soundness/Completeness

Let (Σ_0, Σ_0') be pair of sets of constraint systems, and consider a binary tree obtained by applying our simplification rule following a strategy $\mathcal{S}$.

1. soundness: If all leaves of the tree are labeled with $(\perp, \perp)$ or $(\mathit{solved}, \mathit{solved})$, then $\Sigma_0 \approx_s \Sigma_0'$.
2. completeness: if $\Sigma_0 \approx_s \Sigma_0'$, then all leaves of the tree are labeled with $(\perp, \perp)$ or $(\mathit{solved}, \mathit{solved})$,.

Theorem

Given two sets Σ_0, Σ_0' of constraint systems, it is decidable whether

$$\Sigma_0 \approx_s \Sigma_0'$$

Outline

Conclusion

Main result

The problem whether A and B are trace equivalent is decidable.

$\longrightarrow$ useful to decide privacy-type security properties

Conclusion

Main result

The problem whether A and B are trace equivalent is decidable.

$\longrightarrow$ useful to decide privacy-type security properties

To go further

1. an efficient implementation
   $\longrightarrow$ it seems necessary to come with some optimisations to reduce the search space (*e.g.* the number of interleavings)
2. more primitives
   $\longrightarrow$ this will allow us to analyse a larger class of protocols (*e.g* e-voting protocols, ...)
3. modularity issues (combination/composition)

Conclusion

Main result

The problem whether A and B are trace equivalent is decidable.

$\longrightarrow$ useful to decide privacy-type security properties

To go further

1. an efficient implementation
   $\longrightarrow$ it seems necessary to come with some optimisations to reduce the search space (*e.g.* the number of interleavings)
2. more primitives
   $\longrightarrow$ this will allow us to analyse a larger class of protocols (*e.g* e-voting protocols, ...)
3. modularity issues (combination/composition)

VIP project Jan. 2012 - Dec 2015.

$\longrightarrow$ A postdoc position and a PhD position are available.