A method for verifying privacy-type properties: the unbounded case

Stéphanie Delaune

Équipe EMSEC (IRISA), CNRS, France

Friday, March 17th, 2017

Cryptographic protocols everywhere !

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

Cryptographic protocols everywhere !

- 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.

Electronic passport

An e-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;
- ...

Electronic passport

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

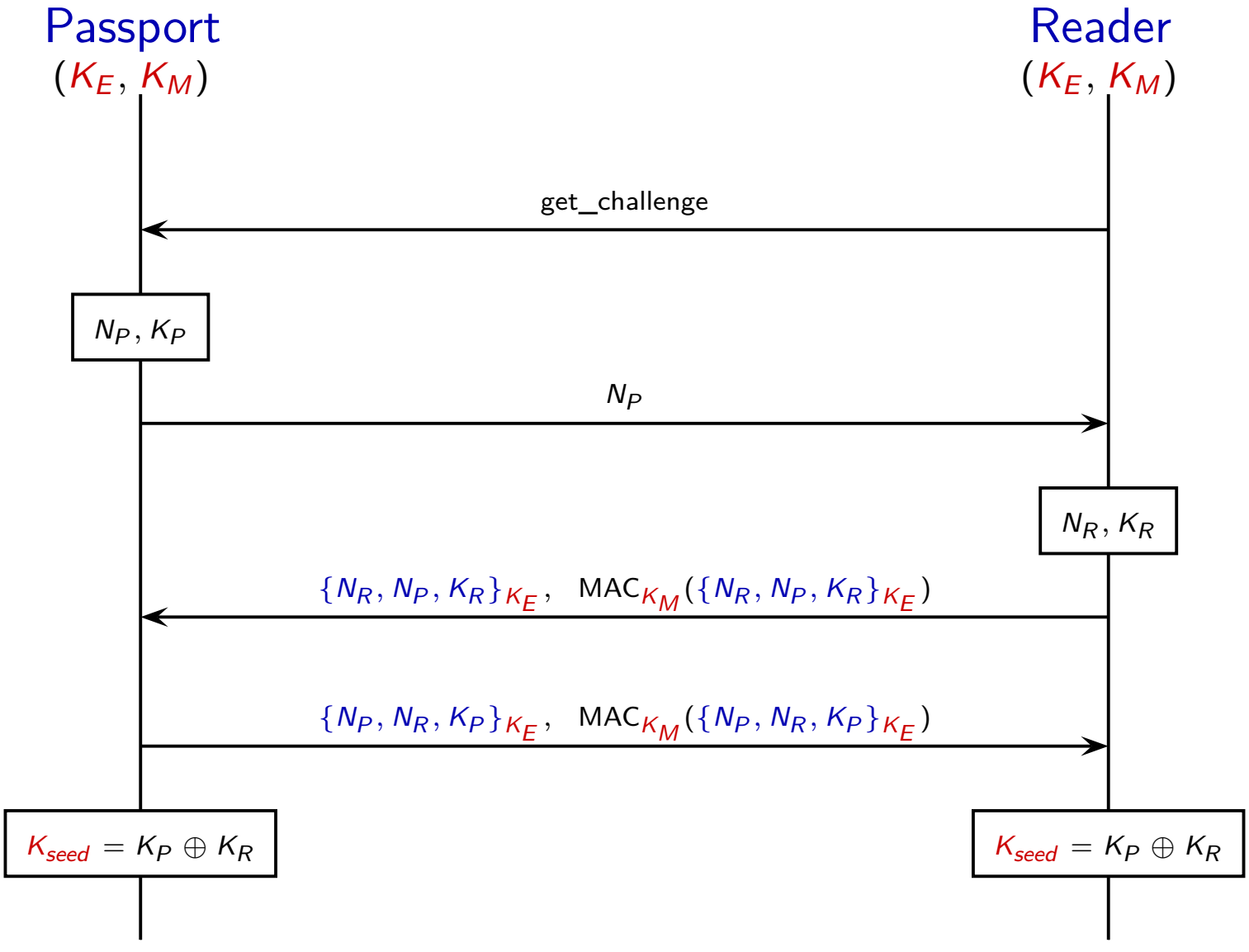
The RFID tag stores:

- 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 protect our personnal data, and to ensure unlinkability.

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*. [ISO/IEC standard 15408]

BAC protocol



How cryptographic protocols can be attacked?

How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  $\hookrightarrow$ replay attack, man-in-the middle attack, ...
- subtle and hard to detect by "eyeballing" the protocol

How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  $\hookrightarrow$ replay attack, man-in-the middle attack, ...
- subtle and hard to detect by "eyeballing" the protocol

$\longrightarrow$ A traceability attack on the BAC protocol (2010)

Security

Defects in e-passports allow real-time tracking

This threat brought to you by RFID

The register - Jan. 2010

Why is it so difficult to verify security protocols ?

- testing their resilience against well-known attacks is not sufficient;
- manual security analysis is error-prone;

A sucessful approach: formal symbolic verification

TAMARIN
Tamarin prover interactive mode

ProVerif

AVANTSSAR

A sucessful approach: formal symbolic verification

$\longrightarrow$ provides a rigorous framework and automatic tools to analyse security protocols and find their logical flaws.

Example: Authentication flaw in the Single Sign-On protocol used *e.g.* in GMail

[Armando *et al.* (2011)] using SATMC (Avantssar verification platform)

What about privacy-type properties?

Part I

Modelling: protocols, security properties, and the attacker

Protocols as processes

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{new}\ n.P$	fresh name generation
		$\mathsf{let}\ \vec{x} = \vec{v}\ \mathsf{then}\ P\ \mathsf{else}\ Q$	term evaluation conditional
		$P \mid Q$	parallel composition
		$!P$	replication

Protocols as processes

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{new}\ n.P$	fresh name generation
		$\mathsf{let}\ \vec{x} = \vec{v}\ \mathsf{then}\ P\ \mathsf{else}\ Q$	term evaluation conditional
		$P \mid Q$	parallel composition
		$!P$	replication

$\mathsf{i}P$	repetition
$P; Q$	sequence

Protocols as processes

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{new}\ n.P$	fresh name generation
		$\mathsf{let}\ \vec{x} = \vec{v}\ \mathsf{then}\ P\ \mathsf{else}\ Q$	term evaluation conditional
		$P \mid Q$	parallel composition
		$!P$	replication

$\mathsf{i}P$	repetition
$P; Q$	sequence

Messages that are exchanged are not necessarily atomic !

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_{\mathsf{E}}$:

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_{\mathsf{E}}$:

Example:

$\Sigma_c = \{\mathsf{senc},\ \mathsf{mac},\ \langle\,\rangle,\ \oplus,\ 0,\ \mathsf{ok}\}$ and $\Sigma_d = \{\mathsf{sdec},\ \mathsf{proj}_1,\ \mathsf{proj}_2,\ \mathsf{eq}\}$.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & \to & x & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_1 \\ \mathsf{eq}(x,x) & \to & \mathsf{ok} & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_2 \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & x \oplus 0 & = & x \\ x \oplus y & = & y \oplus x & x \oplus x & = & 0 \end{array}$$

Messages as terms

They are built from names $\mathcal{N}$, and elements in $\Sigma = \Sigma_c \uplus \Sigma_d$.

To reflect the properties of the cryptographic primitives:

- destructor symbols are subject to a computation relation $\Downarrow$:
- constructor terms are subject to an equational theory $=_{\mathsf{E}}$:

Example:

$\Sigma_c = \{\mathsf{senc},\ \mathsf{mac},\ \langle\,\rangle,\ \oplus,\ 0,\ \mathsf{ok}\}$ and $\Sigma_d = \{\mathsf{sdec},\ \mathsf{proj}_1,\ \mathsf{proj}_2,\ \mathsf{eq}\}$.

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{sdec}(\mathsf{senc}(x,y),y) & \to & x & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_1 \\ \mathsf{eq}(x,x) & \to & \mathsf{ok} & \mathsf{proj}_1(\langle x_1,x_2\rangle) & \to & x_2 \end{array}$$

$$\begin{array}{rcl@{\qquad}rcl} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & x \oplus 0 & = & x \\ x \oplus y & = & y \oplus x & x \oplus x & = & 0 \end{array}$$

$\mathsf{sdec}(\mathsf{senc}(s, k_1 \oplus k_2), k_2 \oplus k_1) \Downarrow s$ $\qquad$ $\mathsf{eq}(k_1 \oplus k_2, k_2 \oplus k_1) \Downarrow \mathsf{ok}$

Back to the BAC protocol

$$
\begin{array}{lll}
P \to R: & N_p & \\
R \to P: & \{N_R, N_P, K_R\}_{K_E}, & \mathrm{MAC}_{K_M}(\{N_R, N_P, K_R\}_{K_E}) \\
P \to R: & \{N_P, N_R, K_P\}_{K_E}, & \mathrm{MAC}_{K_M}(\{N_P, N_R, K_P\}_{K_E})
\end{array}
$$

Nonces and keys are modelled using names: n_r, n_p, k_r, k_p, k_e, k_m.

Back to the BAC protocol

$$
\begin{array}{lll}
P \to R: & N_p & \\
R \to P: & \{N_R, N_P, K_R\}_{K_E}, & \mathrm{MAC}_{K_M}(\{N_R, N_P, K_R\}_{K_E}) \\
P \to R: & \{N_P, N_R, K_P\}_{K_E}, & \mathrm{MAC}_{K_M}(\{N_P, N_R, K_P\}_{K_E})
\end{array}
$$

Nonces and keys are modelled using names: n_r, n_p, k_r, k_p, k_e, k_m.

Modelling Passport's role

$$
\begin{array}{l}
P_{\mathrm{BAC}}(k_E, k_M) = \\
\quad \text{new } n_P.\text{new } k_P.\mathrm{out}(c, n_P).\mathrm{in}(c, x). \\
\quad \text{let } z_1,\ z_2 = \mathrm{eq}(\mathrm{mac}(\mathrm{proj}_1(x), k_M), \mathrm{proj}_2(x)), \\
\qquad\qquad \mathrm{eq}(n_P, \mathrm{proj}_1(\mathrm{proj}_2(\mathrm{sdec}(\mathrm{proj}_1(x), k_E)))) \\
\qquad\qquad\qquad\qquad \texttt{then}\ \mathrm{out}(c, \langle m, \mathrm{mac}(m, k_M)\rangle) \\
\qquad\qquad\qquad\qquad \texttt{else}\ \mathrm{out}(c, \mathrm{error})
\end{array}
$$

where $m = \mathrm{senc}(\langle n_P, \langle \mathrm{proj}_1(\mathrm{sdec}(\mathrm{proj}_1(x), k_E))), k_P\rangle\rangle, k_E)$.

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

$$\textsc{Out} \qquad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\})$$

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

$$\textsc{Out} \quad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\})$$

$$\textsc{In} \quad (\mathsf{in}(c, x).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{x \mapsto u\} \cup \mathcal{P}; \phi)$$

where R is a recipe such that $R\phi \Downarrow u$

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

$$\textsc{Out} \quad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\})$$

$$\textsc{In} \quad (\mathsf{in}(c, x).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{x \mapsto u\} \cup \mathcal{P}; \phi)$$

where R is a recipe such that $R\phi \Downarrow u$

$$\textsc{Then} \quad (\mathsf{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P\ \mathtt{else}\ Q \cup \mathcal{P}; \phi) \xrightarrow{\tau} (P\{\vec{x} \mapsto \vec{u}\} \cup \mathcal{P}; \phi)$$

when $\vec{v} \Downarrow \vec{u}$ for some $\vec{u}$

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

OUT $\quad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\})$

IN $\quad (\mathsf{in}(c, x).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{x \mapsto u\} \cup \mathcal{P}; \phi)$
where R is a recipe such that $R\phi \Downarrow u$

THEN $\quad (\mathsf{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P\ \mathtt{else}\ Q \cup \mathcal{P}; \phi) \xrightarrow{\tau} (P\{\vec{x} \mapsto \vec{u}\} \cup \mathcal{P}; \phi)$
when $\vec{v} \Downarrow \vec{u}$ for some $\vec{u}$

ELSE $\quad \ldots$

Semantics – How processes can evolve?

$\longrightarrow$ a labelled transition system over configurations $K = (\mathcal{P}; \phi)$:

- $\mathcal{P}$ is a multiset of processes;
- $\phi = \{w_1 \mapsto u_1, \ldots, w_n \mapsto u_n\}$ is a frame.

$$\textsc{Out} \quad (\mathsf{out}(c, u).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{out}(c,w)} (P \cup \mathcal{P}; \phi \cup \{w \mapsto u\})$$

$$\textsc{In} \quad (\mathsf{in}(c, x).P \cup \mathcal{P}; \phi) \xrightarrow{\mathsf{in}(c,R)} (P\{x \mapsto u\} \cup \mathcal{P}; \phi)$$

where R is a recipe such that $R\phi \Downarrow u$

$$\textsc{Then} \quad (\mathsf{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P\ \mathtt{else}\ Q \cup \mathcal{P}; \phi) \xrightarrow{\tau} (P\{\vec{x} \mapsto \vec{u}\} \cup \mathcal{P}; \phi)$$

when $\vec{v} \Downarrow \vec{u}$ for some $\vec{u}$

$$\textsc{Else} \quad \ldots$$

$$\mathsf{trace}(K) = \{(\mathsf{tr}, \phi') \mid K \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \phi') \text{ for some configuration } (\mathcal{P}'; \phi')\}.$$

Privacy-type security properties

$\longrightarrow$ they are modelled as equivalence-based properties

Privacy-type security properties

$\longrightarrow$ they are modelled as equivalence-based properties

Trace equivalence $K \approx K'$

For any $(\mathsf{tr}, \phi) \in \mathsf{trace}(K)$, there exists $(\mathsf{tr}', \phi') \in \mathsf{trace}(K')$ such that $\mathsf{tr} = \mathsf{tr}'$, and ϕ and ϕ' are indistinguishable (and conversely).

Privacy-type security properties

$\longrightarrow$ they are modelled as equivalence-based properties

Trace equivalence $K \approx K'$

For any $(\mathrm{tr}, \phi) \in \mathrm{trace}(K)$, there exists $(\mathrm{tr}', \phi') \in \mathrm{trace}(K')$ such that $\mathrm{tr} = \mathrm{tr}'$, and ϕ and ϕ' are indistinguishable (and conversely).

Indistinguishability between frames $\phi \sim \phi'$

- for any recipe R, we have $R\phi \Downarrow u$ (for some u) implies $R\phi' \Downarrow u'$ (for some u');
- for any recipes R_1, R_2, we have $R_1\phi \Downarrow =_E R_2\phi \Downarrow$ implies $R_1\phi' \Downarrow =_E R_2\phi' \Downarrow$.

(and conversely)

Privacy-type security properties

$\longrightarrow$ they are modelled as equivalence-based properties

Trace equivalence $K \approx K'$

For any $(\mathrm{tr}, \phi) \in \mathrm{trace}(K)$, there exists $(\mathrm{tr}', \phi') \in \mathrm{trace}(K')$ such that $\mathrm{tr} = \mathrm{tr}'$, and ϕ and ϕ' are indistinguishable (and conversely).

Indistinguishability between frames $\phi \sim \phi'$

- for any recipe R, we have $R\phi \Downarrow u$ (for some u) implies $R\phi' \Downarrow u'$ (for some u');
- for any recipes R_1, R_2, we have $R_1\phi \Downarrow =_E R_2\phi \Downarrow$ implies $R_1\phi' \Downarrow =_E R_2\phi' \Downarrow$.

(and conversely)

Example (k, k' private names, ok public constant)

$$\{w_1 \mapsto \mathrm{senc}(\mathrm{ok}, k), w_2 \mapsto k\} \not\sim \{w_1 \mapsto \mathrm{senc}(n, k'), w_2 \mapsto k'\}$$

$\longrightarrow$ with $R_1 = \mathrm{sdec}(w_1, w_2)$ and $R_2 = \mathrm{ok}$.

Some privacy-type properties [Arapinis et al, 2010]

Unlinkability

$$!\text{new } ke.\text{new } km.(!P_{\text{BAC}} \mid !R_{\text{BAC}}) \approx\ !\text{new } ke.\text{new } km.(\ P_{\text{BAC}} \mid \ R_{\text{BAC}})$$

many sessions for each passport

only one session for each passport

Some privacy-type properties [Arapinis et al, 2010]

Unlinkability

$$!\text{new } ke.\text{new } km.(!P_{\text{BAC}} \mid !R_{\text{BAC}}) \approx !\text{new } ke.\text{new } km.(P_{\text{BAC}} \mid R_{\text{BAC}})$$

many sessions for each passport

only one session for each passport

Anonymity (w.r.t. id)

$$\begin{array}{rl} & !\text{new } k.\text{new } id.(!P \mid !Q) \\ \approx & !\text{new } k.\text{new } id.(!P \mid !Q) \mid \text{new } k.(!P_0 \mid !Q_0) \end{array}$$

where $P_0 = P\{id \mapsto \text{id}_0\}$ and $Q_0 = Q\{id \mapsto \text{id}_0\}$

Part II

How can we check trace equivalence?

for an unbounded number of sessions

Some recent theoretical results

$\longrightarrow$ [Chrétien PhD thesis, 2016]

- undecidable in general (and even under quite severe restriction)
- a first decidability result through a characterization of equivalence of protocols in terms of equality of languages of deterministic pushdown automata. [Icalp'13, TOCL'15]

- decidable for an interesting subclass of tagged protocols [CSF'15]

Some recent theoretical results

$\longrightarrow$ [Chrétien PhD thesis, 2016]

- undecidable in general (and even under quite severe restriction)
- a first decidability result through a characterization of equivalence of protocols in terms of equality of languages of deterministic pushdown automata. [Icalp'13, TOCL'15]

- decidable for an interesting subclass of tagged protocols [CSF'15]

Main limitations:

- a restricted set of primitives: symmetric encryption, and concatenation only;
- not really practical (no verification tool).

A more pragmatic approach

$\longrightarrow$ [Blanchet *et al.*, LICS'05]

ProVerif tool: `http://www.proverif.ens.fr`

- various cryptographic primitives modeled using equations;
- various security properties: secrecy, authentication, and equivalence-based properties (namely diff-equivalence);

The tool may not terminate or give false attacks.

Works very well in many situations, e.g. strong secrecy

A more pragmatic approach

$\longrightarrow$ [Blanchet *et al.*, LICS'05]

ProVerif tool: `http://www.proverif.ens.fr`

- various cryptographic primitives modeled using equations;
- various security properties: secrecy, authentication, and equivalence-based properties (namely diff-equivalence);

The tool may not terminate or give false attacks.

Works very well in many situations, e.g. strong secrecy

Main issue: diff-equivalence is too strong in many situations.
$\longrightarrow$ ProVerif is not suitable to analyse unlinkability properties.

The Tamarin and Maude-NPA tools are also based on diff-equivalence and suffers from the same problem.

Our approach is pragmatic too

Provide a method to analyse unlinkability for a large class of 2 party protocols, and tool support for that.

Our approach is pragmatic too

Provide a method to analyse unlinkability for a large class of 2 party protocols, and tool support for that.

On the theoretical side

2 reasonable conditions implying anonymity and unlinkability for a large class of 2 party protocols

On the practical side

- our conditions can be checked automatically using existing tools, and we provide tool support for that.
- new proofs and attacks on several RFID protocols.

$\longrightarrow$ first results published at Security & Privacy in 2016 extended since to deal with a larger class of processes

Our class of 2-party protocols

A protocol Π is a tuple $(\vec{k}, \vec{n_I}, \vec{n_R}, \dagger_I, \dagger_R, \mathcal{I}, \mathcal{R})$ such that:

Initiator $\mathcal{I}$	P_I	$:=$	$0 \mid \mathsf{out}(c, u).P_R$
Responder $\mathcal{R}$	P_R	$:=$	$0 \mid \mathsf{in}(c, y).\mathsf{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ 0$
		$\mid$	$\mathsf{in}(c, y).\mathsf{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ \mathsf{out}(c', u')$

Our class of 2-party protocols

A protocol Π is a tuple $(\vec{k}, \vec{n_I}, \vec{n_R}, \dagger_I, \dagger_R, \mathcal{I}, \mathcal{R})$ such that:

$$
\begin{array}{lrcl}
\textit{Initiator } \mathcal{I} & P_I & := & 0 \mid \mathrm{out}(c, u).P_R \\
\textit{Responder } \mathcal{R} & P_R & := & 0 \mid \mathrm{in}(c, y).\mathrm{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ 0 \\
 & & \mid & \mathrm{in}(c, y).\mathrm{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ \mathrm{out}(c', u')
\end{array}
$$

- $\vec{k}$ are identity names whereas $\vec{n_I}$ (resp. $\vec{n_R}$) are session names of role $\mathcal{I}$ (resp. $\mathcal{R}$);
- $\dagger_I$ (resp. $\dagger_R$) is either ! or i (sessions in // or in seq.).

Our class of 2-party protocols

A protocol Π is a tuple $(\vec{k}, \vec{n_I}, \vec{n_R}, \dagger_I, \dagger_R, \mathcal{I}, \mathcal{R})$ such that:

$$\begin{array}{lrcl} \textit{Initiator } \mathcal{I} & P_I & := & 0 \mid \mathrm{out}(c, u).P_R \\ \textit{Responder } \mathcal{R} & P_R & := & 0 \mid \mathrm{in}(c, y).\mathrm{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ 0 \\ & & \mid & \mathrm{in}(c, y).\mathrm{let}\ \vec{x} = \vec{v}\ \mathtt{then}\ P_I\ \mathtt{else}\ \mathrm{out}(c', u') \end{array}$$

- $\vec{k}$ are identity names whereas $\vec{n_I}$ (resp. $\vec{n_R}$) are session names of role $\mathcal{I}$ (resp. $\mathcal{R}$);
- $\dagger_I$ (resp. $\dagger_R$) is either ! or i (sessions in // or in seq.).

We define $P_\Pi := \mathrm{new}\ \vec{k}.(\mathrm{new}\ \vec{n_I}.\mathcal{I} \mid \mathrm{new}\ \vec{n_R}.\mathcal{R})$ and assume the existence of an honest run, i.e. $P_\Pi \xrightarrow{\mathrm{tr}_h} (\emptyset; \phi_h)$ for some frame ϕ_h and some trace tr_h (without relying on the ELSE rule) of the form:

$$\mathrm{out}(c_I, w_1).\mathrm{in}(c_R, w_1).\mathrm{out}(c_R, w_2).\mathrm{in}(c_I, w_2).\ldots.$$

A flavour of our first condition – frame opacity

$$
\begin{aligned}
R \to T : &\quad n_R \\
T \to R : &\quad \{n_R\}_k
\end{aligned}
$$

A flavour of our first condition – frame opacity

$$\begin{aligned} R \to T : &\quad n_R \\ T \to R : &\quad \{n_R\}_k \end{aligned}$$

An attacker who simply listens to the communication will see ...

$$n_R^1, \{n_R^1\}_k,\ n_R^2, \{n_R^2\}_k,\ n_R^3, \{n_R^3\}_{k'}, \ldots$$

A flavour of our first condition – frame opacity

$$\begin{aligned} R \to T : &\quad n_R \\ T \to R : &\quad \{n_R\}_k \end{aligned}$$

An attacker who simply listens to the communication will see ...

$$n_R^1, \{n_R^1\}_k,\ n_R^2, \{n_R^2\}_k,\ n_R^3, \{n_R^3\}_{k'}, \ldots$$

.. but a more active one will be able to observe:

$$n_R^i, \underline{\{n_R^i\}_k},\ n_R^i, \underline{\{n_R^i\}_k},\ n_R^i, \{n_R^i\}_{k'}, \ldots$$

A flavour of our first condition – frame opacity

$$\begin{aligned} R \to T : &\quad n_R \\ T \to R : &\quad \{n_R\}_k \end{aligned}$$

An attacker who simply listens to the communication will see ...

$$n_R^1, \{n_R^1\}_k,\ n_R^2, \{n_R^2\}_k,\ n_R^3, \{n_R^3\}_{k'}, \ldots$$

.. but a more active one will be able to observe:

$$n_R^i, \underline{\{n_R^i\}_k},\ n_R^i, \underline{\{n_R^i\}_k},\ n_R^i, \{n_R^i\}_{k'}, \ldots$$

$\longrightarrow$ Roughly, frame opacity will impose that the sequences of messages that can be observed by an attacker (in any possible execution of the protocol) are indistinguishable from fresh nonces, or to be more precise from a fixed pattern filled with fresh nonces

Frame opacity

To each output action occurring in the specification of the protocol, we associate a pattern.

Examples: $\Box$, error, $\langle \text{req}, \Box \rangle$.

The idealized version of an outputted term with pattern p is p in which the holes have been replaced by fresh nonces.

Frame opacity

To each output action occurring in the specification of the protocol, we associate a pattern.

Examples: $\Box$, error, $\langle \mathsf{req}, \Box \rangle$.

The idealized version of an outputted term with pattern p is p in which the holes have been replaced by fresh nonces.

Frame opacity:

A protocol Π ensures frame opacity if, for any execution

$$! \,\mathsf{new}\, \vec{k}.(\dagger_I \mathsf{new}\, \vec{n_I}.\mathcal{I} \mid \dagger_R \mathsf{new} \vec{n_R}.\mathcal{R}) \xrightarrow{\mathsf{tr}} (\mathcal{P}; \phi)$$

we have that $\phi \sim \Phi_{\mathsf{ideal}}(\mathsf{tr})$.

Frame opacity

To each output action occurring in the specification of the protocol, we associate a pattern.

Examples: $\Box$, error, $\langle \mathsf{req}, \Box \rangle$.

The idealized version of an outputted term with pattern p is p in which the holes have been replaced by fresh nonces.

Frame opacity:
A protocol Π ensures frame opacity if, for any execution

$$! \,\mathsf{new}\, \vec{k}.(\dagger_I \mathsf{new}\, \vec{n_I}.\mathcal{I} \mid \dagger_R \mathsf{new} \vec{n_R}.\mathcal{R}) \xrightarrow{\mathsf{tr}} (\mathcal{P}; \phi)$$

we have that $\phi \sim \Phi_{\mathsf{ideal}}(\mathsf{tr})$.

Going back to our previous example ($out_1 \mapsto \Box$, and $out_2 \mapsto \Box$)

$$\phi = n_R^i, \{n_R^i\}_k, n_R^i, \{n_R^i\}_k, \ldots \qquad \phi_{\mathsf{ideal}} = n_1, n_2, n_3, n_4, \ldots$$

Frame opacity

To each output action occurring in the specification of the protocol, we associate a pattern.

Examples: $\Box$, error, $\langle \text{req}, \Box \rangle$.

The idealized version of an outputted term with pattern p is p in which the holes have been replaced by fresh nonces.

Frame opacity:
A protocol Π ensures frame opacity if, for any execution

$$! \text{ new } \vec{k}.(\dagger_I \text{new } \vec{n_I}.\mathcal{I} \mid \dagger_R \text{new} \vec{n_R}.\mathcal{R}) \xrightarrow{\text{tr}} (\mathcal{P}; \phi)$$

we have that $\phi \sim \Phi_{\text{ideal}}(\text{tr})$.

Going back to our previous example ($out_1 \mapsto \Box$, and $out_2 \mapsto \Box$)

$$\phi = n_R^i, \{n_R^i\}_k, n_R^i, \{n_R^i\}_k, \ldots \quad \not\sim \quad \phi_{\text{ideal}} = n_1, n_2, n_3, n_4, \ldots$$

$\Rightarrow$ Frame opacity does not hold.

A flavour of our second condition – well-authentication

$$
\begin{aligned}
R \to T : &\quad n_R \\
T \to R : &\quad \{n_R, n_T\}_k \\
R \to T : &\quad \{n_T\}_k
\end{aligned}
$$

Assume that T only checks whether the last message is encrypted with k (but does not verify whether the nonce is n_T).

A flavour of our second condition – well-authentication

$$
\begin{array}{lll}
R \to T: & n_R \\
T \to R: & \{n_R, n_T\}_k \\
R \to T: & \{n_T\}_k
\end{array}
$$

Assume that T only checks whether the last message is encrypted with k (but does not verify whether the nonce is n_T).

There is a linkability attack!

$$
\begin{array}{lll}
i.1 & R \to T: & n_R \\
i.2 & T \to R: & \{n_R, n_T\}_k \\
i.3 & R \to T: & \{n_T\}_k
\end{array}
$$

A flavour of our second condition – well-authentication

$$\begin{array}{lll} R \to T: & n_R \\ T \to R: & \{n_R, n_T\}_k \\ R \to T: & \{n_T\}_k \end{array}$$

Assume that T only checks whether the last message is encrypted with k (but does not verify whether the nonce is n_T).

There is a linkability attack!

$$\begin{array}{llll} i.1 & R \to T: & n_R & \\ i.2 & T \to R: & \{n_R, n_T\}_k & \\ i.3 & R \to T: & \underline{\{n_T\}_k} & \end{array} \qquad \begin{array}{llll} ii.1 & R' \to T': & n'_R & \\ ii.2 & T' \to R': & \{n'_R, n'_T\}_{k'} & \\ ii.3 & \to T': & \underline{\{n_T\}_k} & \end{array}$$

T' will accept this last message iff $T = T'$.

A flavour of our second condition – well-authentication

$$\begin{array}{lll} R \to T: & n_R \\ T \to R: & \{n_R, n_T\}_k \\ R \to T: & \{n_T\}_k \end{array}$$

Assume that T only checks whether the last message is encrypted with k (but does not verify whether the nonce is n_T).

There is a linkability attack!

$$\begin{array}{llll} i.1 & R \to T: & n_R & \\ i.2 & T \to R: & \{n_R, n_T\}_k & \\ i.3 & R \to T: & \underline{\{n_T\}_k} & \end{array} \qquad \begin{array}{llll} ii.1 & R' \to T': & n'_R & \\ ii.2 & T' \to R': & \{n'_R, n'_T\}_{k'} & \\ ii.3 & \to T': & \underline{\{n_T\}_k} & \end{array}$$

T' will accept this last message iff $T = T'$.

$\longrightarrow$ Well-authentication basically requires that when an execution deviates from the honest one (i.e. the attacker interfers), the agents that are involved cannot successfully pass a conditional.

Well-authentication

For any execution, a conditional evaluates positively if, and only if, the attacker did not interfer.

Well-authentication:
A protocol Π ensures well-authentication if, for any execution

$$!\,\text{new}\,\vec{k}.(\dagger_I \text{new}\,\vec{n_I}.\mathcal{I} \mid \dagger_R \text{new}\vec{n_R}.\mathcal{R}) \xrightarrow{\text{tr}.\tau_{\text{then}}} (\mathcal{P};\phi)$$

with τ_{then} executed by $\mathcal{R}(\vec{k}, \vec{n_R})$ (resp. $\mathcal{I}$), there exists $\mathcal{I}(\vec{k}, \vec{n_I})$ (resp $\mathcal{R}$) such that both agents follow the honest interaction.

Well-authentication

For any execution, a conditional evaluates positively if, and only if, the attacker did not interfer.

Well-authentication:
A protocol Π ensures well-authentication if, for any execution

$$!\,\text{new}\,\vec{k}.(\dagger_I \text{new}\,\vec{n_I}.\mathcal{I} \mid \dagger_R \text{new}\vec{n_R}.\mathcal{R}) \xrightarrow{\text{tr}.\tau_{\text{then}}} (\mathcal{P};\phi)$$

with τ_{then} executed by $\mathcal{R}(\vec{k},\vec{n_R})$ (resp. $\mathcal{I}$), there exists $\mathcal{I}(\vec{k},\vec{n_I})$ (resp $\mathcal{R}$) such that both agents follow the honest interaction.

Going back to our previous example

$i.1$	$R \to T:$	n_R	$ii.1$	$R' \to T':$	n'_R
$i.2$	$T \to R:$	$\{n_R, n_T\}_k$	$ii.2$	$T' \to R':$	$\{n'_R, n'_T\}_{k'}$
$i.3$	$R \to T:$	$\underline{\{n_T\}_k}$	$ii.3$	$\to T':$	$\underline{\{n_T\}_k}$

$\Rightarrow$ Well-authentication does not hold.

Main result

Theorem

Consider a protocol $\Pi = (\vec{k}, \vec{n}_I, \vec{n}_R, \dagger_I, \dagger_R, \mathcal{I}, \mathcal{R})$ that satisfies both well-authentication and frame opacity. We have that Π ensures unlinkability.

$\longrightarrow$ A similar result holds to establish anonymity.

Main result

Theorem

Consider a protocol $\Pi = (\vec{k}, \vec{n}_I, \vec{n}_R, \dagger_I, \dagger_R, \mathcal{I}, \mathcal{R})$ that satisfies both well-authentication and frame opacity. We have that Π ensures unlinkability.

$\longrightarrow$ A similar result holds to establish anonymity.

Example

$$\begin{aligned} R \to T : &\quad n_R \\ T \to R : &\quad \{n_R, n_T\}_k \\ R \to T : &\quad \{n_T\}_k \text{ with a check on } n_T. \end{aligned}$$

This is a 2-party protocol that satisfies our 2 conditions.
Therefore, unlinkability holds.

$$!\text{new}\, \vec{k}.(!\text{new}\, \vec{n_R}.R \mid !\text{new}\, \vec{n_T}.T) \approx !\text{new}\, \vec{k}.(\text{new}\, \vec{n_R}.R \mid \text{new}\, \vec{n_T}.T)$$

Tool support

Our two conditions can be automatically verified using ProVerif:

- well-authentication: this is a pure reachability property
  $\longrightarrow$ ProVerif (and other existing tools) works well
- frame opacity: equivalence between sequences of messages
  $\longrightarrow$ checkable with good precision via diff-equivalence

Tool support

Our two conditions can be automatically verified using ProVerif:

- well-authentication: this is a pure reachability property
  $\longrightarrow$ ProVerif (and other existing tools) works well
- frame opacity: equivalence between sequences of messages
  $\longrightarrow$ checkable with good precision via diff-equivalence

Tool UKANO

A tool built on top of ProVerif that automatically checks our two conditions.

`http://projects.lsv.ens-cachan.fr/ukano/`

Summary of our case studies using UKANO

Protocol	FO	WA	unlinkability
Feldhofer	✓	✓	**safe**
Feldhofer variant (with !)	✓	✗	attack
Hash-Lock	✓	✓	**safe**
LAK (stateless)	–	✗	attack
Fixed LAK	✓	✓	**safe**
BAC	✓	✓	**safe**
BAC/PA/AA	✓	✓	**safe**
PACE (faillible dec)	–	✗	attack
PACE (as in [Bender et al, 09])	–	✗	attack
PACE	–	✗	attack
PACE with tags	✓	✓	**safe**
DAA sign	✓	✓	**safe**
DAA join	✓	✓	**safe**
abcdh (irma)	✓	✓	**safe**

▸ More details about PACE

Conclusion

A method to analyse unlinkability for a large class of 2 party protocols, and tool support (based on ProVerif) for that.

Conclusion

A method to analyse unlinkability for a large class of 2 party protocols, and tool support (based on ProVerif) for that.

Future work

- considering other existing verification tools as a backend, e.g. Tamarin.
  $\longrightarrow$ useful to go beyond some ProVerif limitations (e.g. i)

Conclusion

A method to analyse unlinkability for a large class of 2 party protocols, and tool support (based on ProVerif) for that.

Future work

- considering other existing verification tools as a backend, e.g. Tamarin.
  $\longrightarrow$ useful to go beyond some ProVerif limitations (e.g. i)
- applying the same method to analyse some other privacy-type security properties, e.g. vote-privacy

$$S[V_A(\mathit{yes})|\ V_B(\mathit{no})] \approx S[V_A(\mathit{no})|\ V_B(\mathit{yes})]$$

Conclusion

A method to analyse unlinkability for a large class of 2 party protocols, and tool support (based on ProVerif) for that.

Future work

- considering other existing verification tools as a backend, e.g. Tamarin.
  $\longrightarrow$ useful to go beyond some ProVerif limitations (e.g. i)
- applying the same method to analyse some other privacy-type security properties, e.g. vote-privacy

$$S[V_A(yes)|\ V_B(no)] \approx S[V_A(no)|\ V_B(yes)]$$

- Considering a larger class of protocols, e.g. stateful protocols

PACE protocol

1. T $\rightarrow$ R : $\{s_T\}_k$
2. R $\rightarrow$ T : g^{n_R}
3. T $\rightarrow$ R : g^{n_T}
4. Both parties compute $G = \mathrm{gen}(s_T, g^{n_R n_T})$.
5. R $\rightarrow$ T : $G^{n'_R}$
6. T $\rightarrow$ R : $G^{n'_T}$ $\quad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. R $\rightarrow$ T : $\mathrm{mac}(G^{n'_T}, k')$
9. T $\rightarrow$ R : $\mathrm{mac}(G^{n'_R}, k')$

PACE protocol

1. $\mathsf{T} \rightarrow \mathsf{R}: \quad \{s_T\}_k$
2. $\mathsf{R} \rightarrow \mathsf{T}: \quad g^{n_R}$
3. $\mathsf{T} \rightarrow \mathsf{R}: \quad g^{n_T}$
4. Both parties compute $G = \mathsf{gen}(s_T, g^{n_R n_T})$.
5. $\mathsf{R} \rightarrow \mathsf{T}: \quad G^{n'_R}$
6. $\mathsf{T} \rightarrow \mathsf{R}: \quad G^{n'_T} \qquad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. $\mathsf{R} \rightarrow \mathsf{T}: \quad \mathsf{mac}(G^{n'_T}, k')$
9. $\mathsf{T} \rightarrow \mathsf{R}: \quad \mathsf{mac}(G^{n'_R}, k')$

Model 1 (faillible dec)

$\Sigma_c = \{\mathsf{senc},\ \mathsf{exp},\ \mathsf{mac},\ \mathsf{gen},\ \mathsf{g},\ \mathsf{ok}\}$ and $\Sigma_d = \{\mathsf{sdec},\ \mathsf{neq}\}$.

$$\mathsf{sdec}(\mathsf{senc}(x,y),y) \rightarrow x \qquad \mathsf{eq}(x,x) \rightarrow \mathsf{ok}$$

$$\mathsf{exp}(\mathsf{exp}(\mathsf{g},y),z) = \mathsf{exp}(\mathsf{exp}(\mathsf{g},z),y)$$

$$\mathsf{exp}(\mathsf{exp}(\mathsf{gen}(x_1,x_2),y),z) = \mathsf{exp}(\mathsf{exp}(\mathsf{gen}(x_1,x_2),z),y)$$

PACE protocol

1. T $\rightarrow$ R : $\{s_T\}_k$
2. R $\rightarrow$ T : g^{n_R}
3. T $\rightarrow$ R : g^{n_T}
4. Both parties compute $G = \mathrm{gen}(s_T, g^{n_R n_T})$.
5. R $\rightarrow$ T : $G^{n'_R}$
6. T $\rightarrow$ R : $G^{n'_T}$ $\quad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. R $\rightarrow$ T : $\mathrm{mac}(G^{n'_T}, k')$
9. T $\rightarrow$ R : $\mathrm{mac}(G^{n'_R}, k')$

Model 1 (faillible dec)

$\Sigma_c = \{\mathrm{senc},\ \mathrm{exp},\ \mathrm{mac},\ \mathrm{gen},\ \mathrm{g},\ \mathrm{ok}\}$ and $\Sigma_d = \{\mathrm{sdec},\ \mathrm{neq}\}$.

$$\mathrm{sdec}(\mathrm{senc}(x,y),y) \rightarrow x \qquad \mathrm{eq}(x,x) \rightarrow \mathrm{ok}$$
$$\mathrm{exp}(\mathrm{exp}(\mathrm{g},y),z) = \mathrm{exp}(\mathrm{exp}(\mathrm{g},z),y)$$
$$\mathrm{exp}(\mathrm{exp}(\mathrm{gen}(x_1,x_2),y),z) = \mathrm{exp}(\mathrm{exp}(\mathrm{gen}(x_1,x_2),z),y)$$

$\longrightarrow$ Well-Auth. does not hold ... and indeed there is an attack

▸ Back to the conclusion

PACE protocol

1. $T \rightarrow R : \{s_T\}_k$
2. $R \rightarrow T : g^{n_R}$
3. $T \rightarrow R : g^{n_T}$
4. Both parties compute $G = \text{gen}(s_T, g^{n_R n_T})$.
5. $R \rightarrow T : G^{n'_R}$
6. $T \rightarrow R : G^{n'_T} \quad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. $R \rightarrow T : \text{mac}(G^{n'_T}, k')$
9. $T \rightarrow R : \text{mac}(G^{n'_R}, k')$

Model 2 (as in [Bender et al, 09]) R does not check " $\neq G^{n'_R}$ "

$\Sigma_c = \{\text{senc}, \text{sdec}, \text{exp}, \text{mac}, \text{gen}, \text{g}, \text{ok}\}$ and $\Sigma_d = \{\text{neq}\}$.

$$\begin{aligned} \text{sdec}(\text{senc}(x, y), y) &= x & \text{eq}(x, x) &\rightarrow \text{ok} \\ \text{exp}(\text{exp}(\text{g}, y), z) &= & \text{exp}(\text{exp}(\text{g}, z), y) \\ \text{exp}(\text{exp}(\text{gen}(x_1, x_2), y), z) &= & \text{exp}(\text{exp}(\text{gen}(x_1, x_2), z), y) \end{aligned}$$

$\longrightarrow$ Well-Authentication does not hold (reflexion attack)

Back to the conclusion

PACE protocol

1. $T \rightarrow R : \{s_T\}_k$
2. $R \rightarrow T : g^{n_R}$
3. $T \rightarrow R : g^{n_T}$
4. Both parties compute $G = \text{gen}(s_T, g^{n_R n_T})$.
5. $R \rightarrow T : G^{n'_R}$
6. $T \rightarrow R : G^{n'_T} \quad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. $R \rightarrow T : \text{mac}(G^{n'_T}, k')$
9. $T \rightarrow R : \text{mac}(G^{n'_R}, k')$

Model 3

$\Sigma_c = \{\text{senc}, \text{sdec}, \text{exp}, \text{mac}, \text{gen}, \text{g}, \text{ok}\}$ and $\Sigma_d = \{\text{neq}\}$.

$$\text{sdec}(\text{senc}(x, y), y) = x \qquad \text{eq}(x, x) \rightarrow \text{ok}$$
$$\text{exp}(\text{exp}(\text{g}, y), z) = \text{exp}(\text{exp}(\text{g}, z), y)$$
$$\text{exp}(\text{exp}(\text{gen}(x_1, x_2), y), z) = \text{exp}(\text{exp}(\text{gen}(x_1, x_2), z), y)$$

$\longrightarrow$ Well-Authentication does not hold (attack ... but not a realistic one in the context of the e-passport application)

Back to the conclusion

PACE protocol

1. $T \rightarrow R : \{s_T\}_k$
2. $R \rightarrow T : g^{n_R}$
3. $T \rightarrow R : g^{n_T}$
4. Both parties compute $G = \text{gen}(s_T, g^{n_R n_T})$.
5. $R \rightarrow T : G^{n'_R}$
6. $T \rightarrow R : G^{n'_T} \quad (\neq G^{n'_R})$
7. Both parties compute $k' = G^{n'_R n'_T}$
8. $R \rightarrow T : \text{mac}((\text{req}, G^{n'_T}), k')$
9. $T \rightarrow R : \text{mac}((\text{rep}, G^{n'_R}), k')$

Model 4 (adding tags in messages 8 and 9)

$\Sigma_c = \{\text{senc}, \text{sdec}, \text{exp}, \text{mac}, \text{gen}, \text{g}, \text{ok}\}$ and $\Sigma_d = \{\text{neq}\}$.

$$\begin{aligned} \text{sdec}(\text{senc}(x, y), y) &= x & \text{eq}(x, x) &\rightarrow \text{ok} \\ \text{exp}(\text{exp}(\text{g}, y), z) &= \text{exp}(\text{exp}(\text{g}, z), y) \\ \text{exp}(\text{exp}(\text{gen}(x_1, x_2), y), z) &= \text{exp}(\text{exp}(\text{gen}(x_1, x_2), z), y) \end{aligned}$$

$\longrightarrow$ Frame Opacity and Well-Authentication hold. Therefore, thanks to our result, unlinkability holds.

Back to the conclusion