From Security Protocols to Pushdown Automata

Rémy Chrétien[1], Véronique Cortier[2] and Stéphanie Delaune[1]

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

[2]LORIA - CNRS

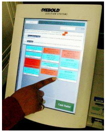
5th September 2013

Cryptographic protocols everywhere

PayPal™

Cryptographic protocols

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

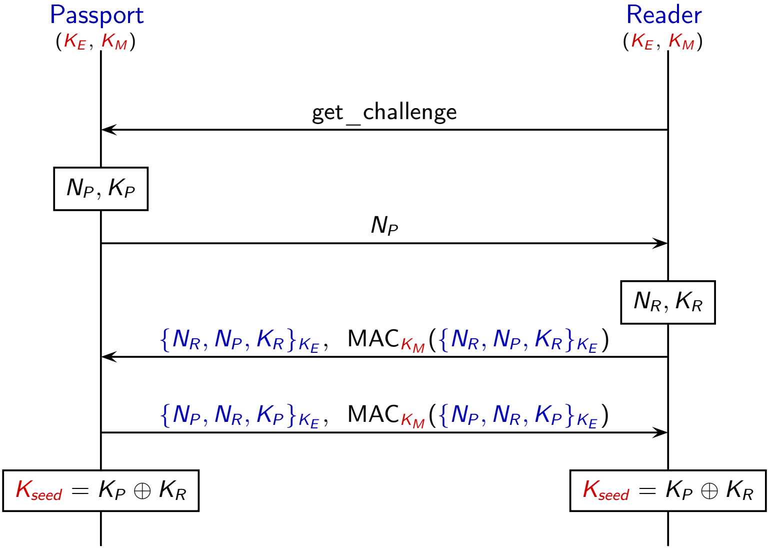


Protocols and Security

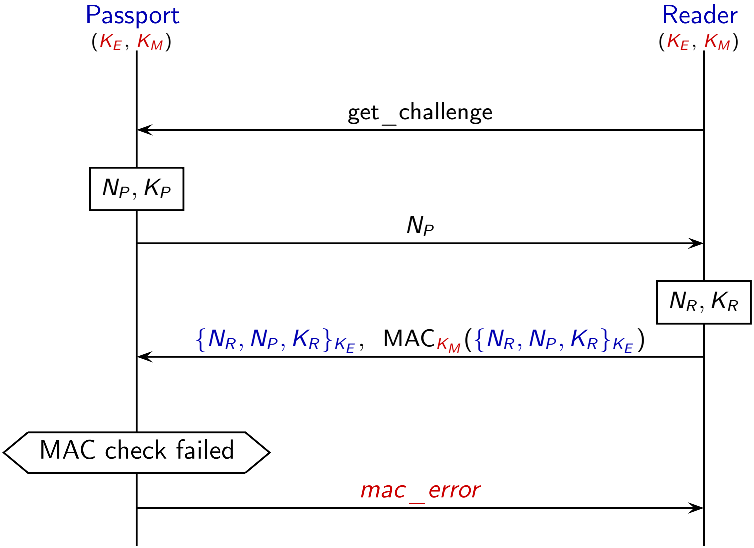
A difficult design:

- Needham-Shroeder protocol (1978), correction and attack by Lowe (1995): an attacker could pretend to be an honest agent.
- Google Single-Sign-On protocol (2008): an attacker can log in to the Google services of a user.
- French e-passport (2010): an attacker can trace a particular user.

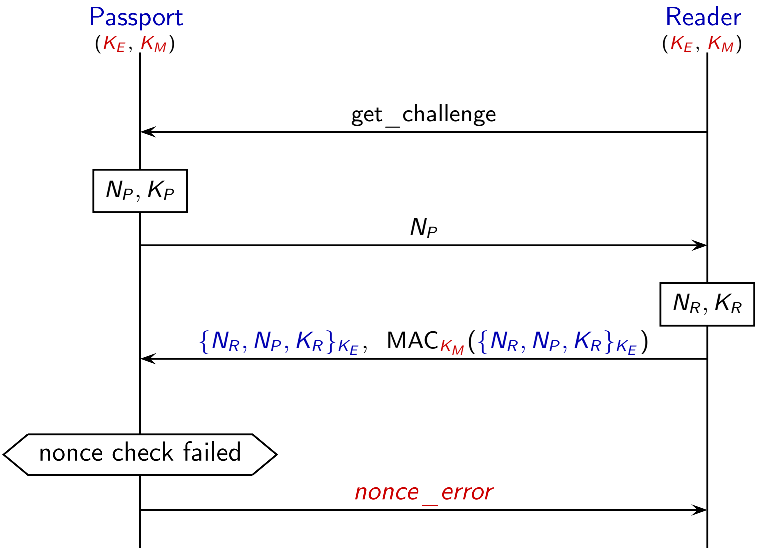
French e-passport



French e-passport



French e-passport



Formal methods

- Messages abstracted through terms. Example: $\{s\}_k$
- Enough to express various properties : secrecy, authentication, anonymity...
- Two main notions:
  - **reachability:** can the attacker reach a certain term?
  - **equivalence:** are two protocols distinguishable by the attacker?
- Efficient tools: ProVerif, AVISPA, Scyther find the aforementioned attacks.

Anonymity as an equivalence property

A protocol P preserves the anonymity of an identity id if:

$$P[id \mapsto id_1] \approx P[id \mapsto id_2]$$

Decidable classes of protocols

Aim: find classes of protocols for which equivalence is decidable with an unbounded number of sessions.

State of the art:

- reachability is undecidable in general...
- ...but decidability can be achieved through, *e.g.* a bounded number of sessions, a limited number of variables, tagging
- equivalence is decidable for a bounded number of sessions

Our proposition: equivalence result for protocols with one variable [Comon-Cortier, 2003].

The class $\mathcal{C}_{PP}$

$\mathcal{C}_{PP}$ is the class of protocols

$$P = \Big|_{i=1}^{n} \Big|_{j=1}^{p_i} !\mathrm{in}(c_i, u^i_j).\mathrm{new}\ \tilde{n}.\mathrm{out}(c_i, u'^i_j)$$

such that:

- input and output terms contain only one variable except for the random seeds
- the protocol is deterministic

$$\begin{array}{rll} P = & & !\mathrm{in}(c_A, \mathrm{start}).\mathrm{new}\ n.\mathrm{out}(c_A, \{v\}^n_k) \\ & | & !\mathrm{in}(c_S, \{x\}^-_k).\mathrm{new}\ n.\mathrm{out}(c_S, x) \end{array}$$

Semantics

$$P = \quad !\mathsf{in}(c_A, \mathsf{start}).\mathsf{new}\ n.\mathsf{out}(c_A, \{v\}_k^n) \\ \mid \quad !\mathsf{in}(c_S, \{x\}_{\overline{k}}).\mathsf{new}\ n.\mathsf{out}(c_S, x)$$

An example of execution

$$\sigma = \{x_1 \triangleright k; x_2 \triangleright \{s\}_k^n\}$$

$$(\mathsf{in}(c_S, \{x\}_k).\mathsf{out}(c_S, x); \sigma) \xrightarrow{\mathsf{in}(c_S, \{v'\}_{x_1}^r)} (\mathsf{out}(c_S, v'); \sigma)$$

$$(\mathsf{out}(c_S, v'); \sigma) \xrightarrow{\mathsf{out}(c_S, x_3)} (0; \sigma \cup \{x_3 \triangleright v'\})$$

Semantics

$$P = \quad !\mathrm{in}(c_A, \mathrm{start}).\mathrm{new}\ n.\mathrm{out}(c_A, \{v\}_k^n) \;|\; !\mathrm{in}(c_S, \{x\}_{\bar{k}}).\mathrm{new}\ n.\mathrm{out}(c_S, x)$$

An example of execution

$$\sigma = \{x_1 \triangleright k; x_2 \triangleright \{s\}_k^n\}$$

$$(\mathrm{in}(c_S, \{x\}_k).\mathrm{out}(c_S, x); \sigma) \xrightarrow{\mathrm{in}(c_S, \{v'\}_{x_1}^r)} (\mathrm{out}(c_S, v'); \sigma)$$

$$(\mathrm{out}(c_S, v'); \sigma) \xrightarrow{\mathrm{out}(c_S, x_3)} (0; \sigma \cup \{x_3 \triangleright v'\})$$

Semantics

$$P = \quad !\mathsf{in}(c_A, \mathsf{start}).\mathsf{new}\ n.\mathsf{out}(c_A, \{v\}_k^n)$$
$$\mid \quad !\mathsf{in}(c_S, \{x\}_{\overline{k}}).\mathsf{new}\ n.\mathsf{out}(c_S, x)$$

An example of execution

$$\sigma = \{x_1 \triangleright k; x_2 \triangleright \{s\}_k^n\}$$
$$(\mathsf{in}(c_S, \{x\}_k).\mathsf{out}(c_S, x); \sigma) \xrightarrow{\mathsf{in}(c_S, \{v'\}_{x_1}^r)} (\mathsf{out}(c_S, v'); \sigma)$$
$$(\mathsf{out}(c_S, v'); \sigma) \xrightarrow{\mathsf{out}(c_S, x_3)} (0; \sigma \cup \{x_3 \triangleright v'\})$$

Definition (Traces)

The traces of a protocol P is the set $\mathsf{trace}(P)$ *of all sequences of executions and their respective substitutions.*

Trace equivalence

Definition (Static equivalence)

Two sequences of messages are statically equivalent, denoted $\phi \sim_s \phi'$*, if any test that holds true in* ϕ *is true in* ϕ' *(and conversely).*

Definition (Trace equivalence)

$P_A \sqsubseteq_t P_B$ *if for every* $(\mathrm{tr}, \phi) \in \mathrm{trace}(P_A)$*, there exists* $(\mathrm{tr}, \phi') \in \mathrm{trace}(P_B)$ *such that* $\phi \sim_s \phi'$*.*

$P_A \approx_t P_B$ *if* $P_A \sqsubseteq_t P_B$ *and* $P_B \sqsubseteq_t P_A$*.*

The goal

The question

Given two protocols of $\mathcal{C}_{PP}$, can we decide whether $P \approx_t Q$?

Existing tools

- for a unbounded number of sessions: ProVerif [Blanchet, 2001] (no decision procedure)
- for a bounded number of sessions: Apte [Cheval *et al.*, 2011], AKiss [Chadha *et al.*, 2011], SPEC [Tiu, 2010].

Our work

Trace equivalence in $\mathcal{C}_{PP}$ is decidable for an unbounded number of sessions.

Decidability of $\mathcal{C}_{PP}$

Theorem (Decidability of $\mathcal{C}_{PP}$)

Trace equivalence in $\mathcal{C}_{PP}$ is decidable.

Idea

We reduce the problem of trace equivalence of protocols to the equivalence of deterministic pushdown automata (Sénizergues, 2001)

From $\mathcal{C}_{PP}$ to $\mathcal{C}^r_{PP}$

Getting rid of the Dolev-Yao attacker

$P, Q \in \mathcal{C}_{PP}$, $P \approx_t Q$ if, and only if, $\bar{P} \approx'_t \bar{Q}$ where $\bar{P}$, $\bar{Q} \in \mathcal{C}^r_{PP}$.

How?

- push the abilities of the attacker into the procotol (*i.e.* new branches for encryption, decryption...): $P \to \bar{P}$, and $Q \to \bar{Q}$
- but the attacker can only *forward* messages on the network: $\approx'_t$ instead of $\approx_t$.

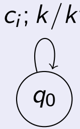
Real-time Generalized Deterministic Pushdown Automata

Applying a branch of the protocol

To a branch

$$!\mathrm{in}(c_i, \{x\}_k^y).\mathrm{new}\ n.\mathrm{out}(c_i, \{x\}_{k'}^n)$$

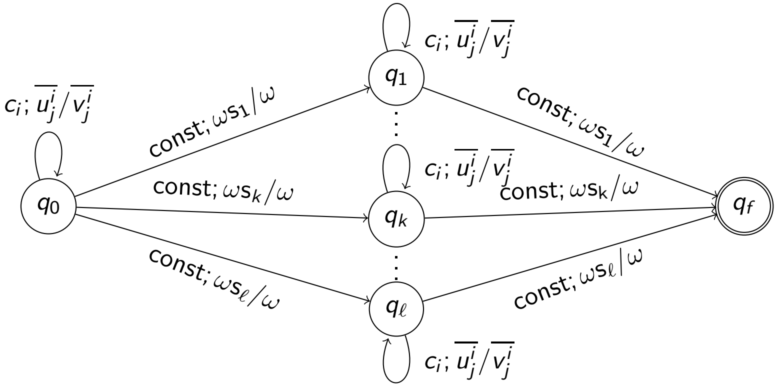
we associate the transition



Trace equivalence is language equivalence

1. $P \approx_t Q$ if, and only if, $L(\mathcal{A}_P) = L(\mathcal{A}_Q)$
2. $L(\mathcal{A}) = L(\mathcal{B})$ if, and only if, $P_\mathcal{A} \approx_t P_\mathcal{B}$

A glimpse at $\mathcal{A}_P$



Undecidability results

Trace inclusion

Suprisingly (or not) trace inclusion is undecidable...

Trace equivalence

Extending $\mathcal{C}_{PP}$ with a single choice operator makes trace equivalence undecidable too.

$\rightarrow$ The frontier between decidable and undecidable classes for equivalence is thin.

Conclusion

Three main points:

- **Decidability of trace equivalence in $\mathcal{C}_{PP}$:**
  First step: weakening the notion trace equivalence.
  Second step: reducing it to language equivalence for RGDPA.
- **Converse encoding of RGDPA into protocols**
- **Undecidability of trace equivalence for slight extensions of $\mathcal{C}_{PP}$**

Extending the class

Deterministic encryption

- Similar result for deterministic encryption...
- ...without the encoding from automata to protocols.

Pairs

- Pairs are difficult to deal with only the automaton's stack
- but still could be emulated and hence added to the class.

Implementing the procedure (WIP)

Combination with the tool lAlBlC of G.Sénizergues and P.Henry to automatically prove trace equivalence of protocols in $\mathcal{C}_{PP}$.

Thank you for your attention.