Verification of security protocols
from confidentiality to privacy

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Thursday, July 10th, 2014

Cryptographic protocols everywhere !

Goal: they aim at securing communications over public/insecure networks

Some security properties

- **Secrecy**: May an intruder learn some secret message between two honest participants?
- Authentication: Is the agent Alice really talking to Bob?
- Anonymity: Is an attacker able to learn something about the identity of the participants who are communicating?
- Non-repudiation: Alice sends a message to Bob. Alice cannot later deny having sent this message. Bob cannot deny having received the message.
- ...

Example: E-voting application

Eligibility: only legitimate voters can vote, and only once

No early results: no early results can be obtained which could influence the remaining voters

Vote-privacy/Receipt-freeness/Coercion-resistance: the fact that a particular voted in a particular way is not revealed to anyone

Individual/Universal verifiability:
a voter can verify that her vote was really counted, and that the published outcome is the sum of all the votes

How does a cryptographic protocol work (or not)? (1/2)

cryptographic primitives = basic building blocks

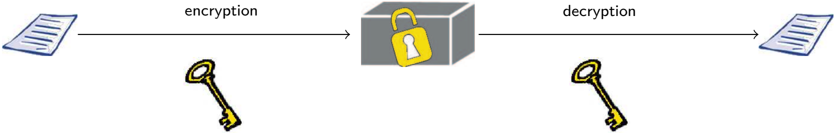
$\longrightarrow$ symmetric/ asymmetric encryption, signature, hash function, ...

How does a cryptographic protocol work (or not)? (1/2)

cryptographic primitives = basic building blocks

$\longrightarrow$ symmetric/ asymmetric encryption, signature, hash function, ...

Symmetric encryption



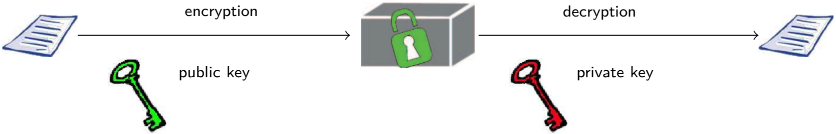
Examples: Caesar cipher, Enigma machine ($\sim$ 1918-1945), or more recently DES (1973) and AES (1997)

How does a cryptographic protocol work (or not)? (1/2)

cryptographic primitives = basic building blocks

$\longrightarrow$ symmetric/ asymmetric encryption, signature, hash function, ...

Asymmetric encryption



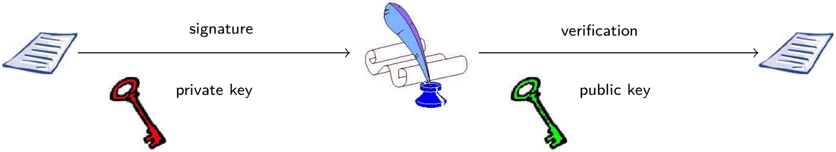
Examples: First system published by Diffie-Hellman (1976), RSA (1977)

How does a cryptographic protocol work (or not)? (1/2)

cryptographic primitives = basic building blocks

$\longrightarrow$ symmetric/ asymmetric encryption, signature, hash function, ...

Signature



How does a cryptographic protocol work (or not)? (2/2)

protocol = small programs explaining how to exchange messages

$\longrightarrow$ key-exchange protocols, authentication protocols, e-voting protocols, Bitcoin protocol, ...

How does a cryptographic protocol work (or not)? (2/2)

protocol = small programs explaining how to exchange messages

$\longrightarrow$ key-exchange protocols, authentication protocols, e-voting protocols, Bitcoin protocol, ...

Example: A simplified version of the Denning-Sacco protocol (1981)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What about secrecy of s ?

How does a cryptographic protocol work (or not)? (2/2)

protocol = small programs explaining how to exchange messages

$\longrightarrow$ key-exchange protocols, authentication protocols, e-voting protocols, Bitcoin protocol, ...

Example: A simplified version of the Denning-Sacco protocol (1981)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What about secrecy of s ?
Consider a scenario where A starts a session with C who is dishonest.

$$1.\ A \rightarrow C :\ \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C))$$

How does a cryptographic protocol work (or not)? (2/2)

protocol = small programs explaining how to exchange messages

$\longrightarrow$ key-exchange protocols, authentication protocols, e-voting protocols, Bitcoin protocol, ...

Example: A simplified version of the Denning-Sacco protocol (1981)

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

What about secrecy of s ?
Consider a scenario where A starts a session with C who is dishonest.

1. $A \to C :\ \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(C))$

2. $C(A) \to B : \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B))$
3. $\quad B \to A : \mathrm{senc}(s, k)$ Attack !

Exercise

We propose to fix the Denning-Sacco protocol as follows:

Version 1

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\langle A, B, \mathrm{sign}(k, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Version 2

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(\langle A, B, k\rangle, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Which version would you prefer to use?

Exercise

We propose to fix the Denning-Sacco protocol as follows:

Version 1

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\langle A, B, \mathrm{sign}(k, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Version 2

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(\langle A, B, k\rangle, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Which version would you prefer to use? Version 2

$\longrightarrow$ Version 1 is still vulnerable to the aforementioned attack.

What about protocols used in real life ?

E-passport

Electronic passport

$\longrightarrow$ studied in [Arapinis *et al.*, 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.

Electronic passport

⟶ studied in [Arapinis *et al.*, 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.

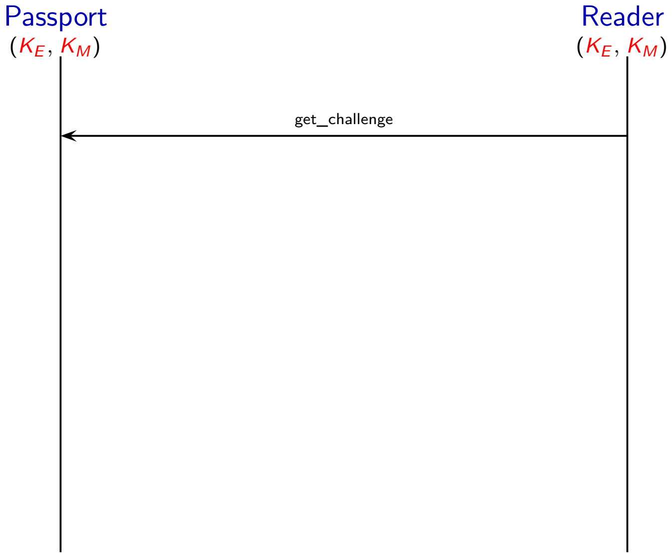
ISO/IEC standard 15408

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.*

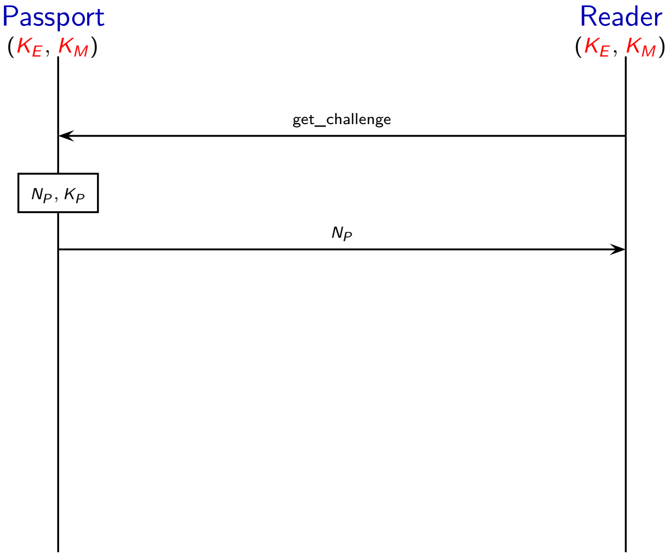
BAC protocol

Passport	Reader
(K_E, K_M)	(K_E, K_M)

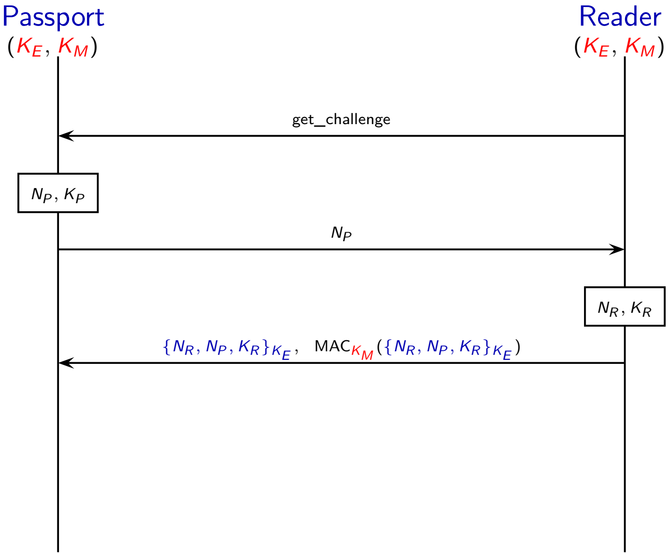
BAC protocol



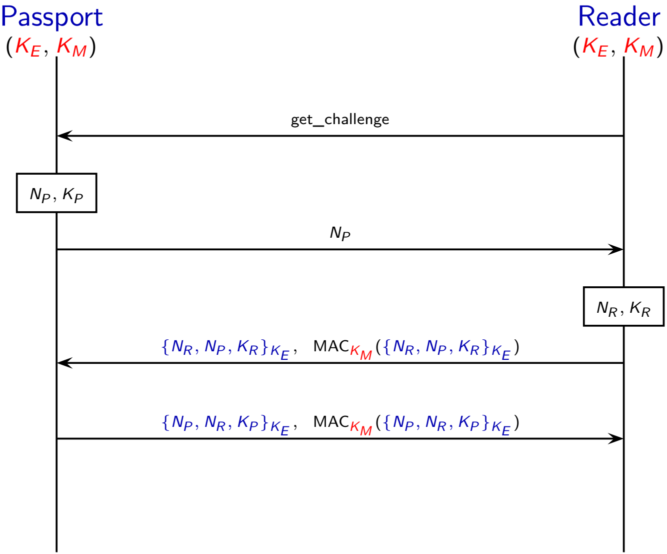
BAC protocol



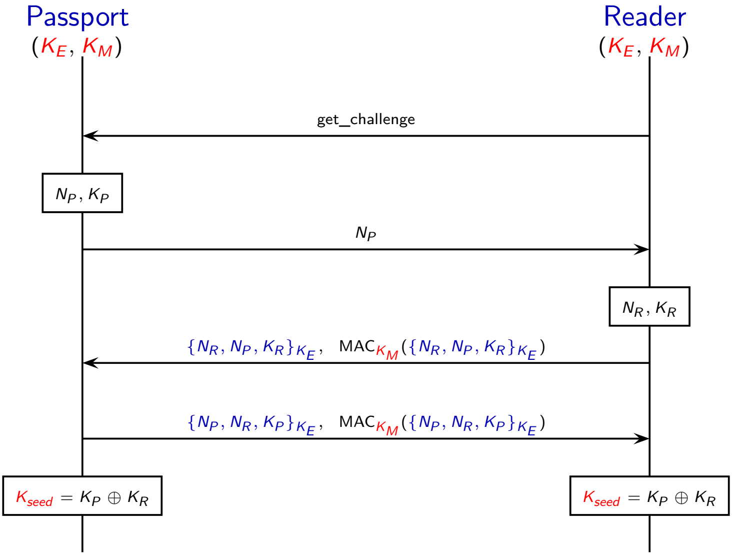
BAC protocol



BAC protocol



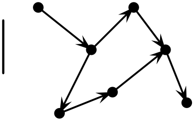
BAC protocol



This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling

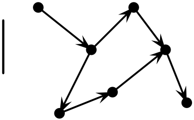


$\models \quad \varphi$

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling



$\models \varphi$

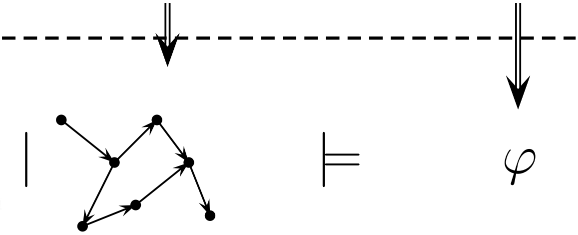
E-passport application

What about unlinkability of the ePassport holders ?

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling



Outline of the this talk

1. Modelling cryptographic protocols and their security properties
2. Designing verification algorithms

Modelling cryptographic protocols and their security properties

Messages as terms

Terms built over a signature $\mathcal{F}$, and an infinite set of names $\mathcal{N}$.

$$
\begin{array}{lcll}
t & ::= & & \\
 & \mid & n & \text{name } n \\
 & \mid & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

Example:

- $\mathcal{F} = \{\mathsf{senc}(\cdot,\cdot),\ \mathsf{sdec}(\cdot,\cdot),\ \langle \cdot,\cdot \rangle,\ \mathsf{proj}_1(\cdot),\ \mathsf{proj}_2(\cdot)\}$;
- $\mathcal{N} = \{n, a, k, s, \ldots\}$.

$\longrightarrow$ some terms: $\mathsf{senc}(\langle s, a \rangle, k)$, $\mathsf{sdec}(\mathsf{enc}(s, k), k)$, and s.

Messages as terms

Terms built over a signature $\mathcal{F}$, and an infinite set of names $\mathcal{N}$.

$$
\begin{array}{lcll}
\mathrm{t} & ::= & & \\
 & | & n & \text{name } n \\
 & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

Example:

- $\mathcal{F} = \{\mathrm{senc}(\cdot,\cdot),\ \mathrm{sdec}(\cdot,\cdot),\ \langle\cdot,\cdot\rangle,\ \mathrm{proj}_1(\cdot),\ \mathrm{proj}_2(\cdot)\}$;
- $\mathcal{N} = \{n, a, k, s, \ldots\}$.

$\longrightarrow$ some terms: $\mathrm{senc}(\langle s, a\rangle, k)$, $\mathrm{sdec}(\mathrm{enc}(s, k), k)$, and s.

Term algebra is equipped with an equational theory E.

Example: $\mathrm{dec}(\mathrm{enc}(x, y), y) = x$, $\mathrm{proj}_1(\langle x, y\rangle) = x$, $\mathrm{proj}_2(\langle x, y\rangle) = y$.

We have that $\mathrm{sdec}(\mathrm{senc}(s, k), k) =_{\mathsf{E}} s$.

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What function symbols and equations do we need to model this protocol?

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What function symbols and equations do we need to model this protocol?

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

Going back to the Denning Sacco protocol

$$\begin{array}{rcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

What function symbols and equations do we need to model this protocol?

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

2. asymmetric encryption: $\mathsf{aenc}(\cdot,\cdot)$, $\mathsf{adec}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$

   $\longrightarrow \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What function symbols and equations do we need to model this protocol?

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

2. asymmetric encryption: $\mathsf{aenc}(\cdot,\cdot)$, $\mathsf{adec}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$

   $\longrightarrow \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$

3. signature: $\mathsf{sign}(\cdot,\cdot)$, $\mathsf{check}(\cdot,\cdot)$

   $\longrightarrow \mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = x$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

What function symbols and equations do we need to model this protocol?

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $$\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: $\mathsf{aenc}(\cdot,\cdot)$, $\mathsf{adec}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$

   $$\longrightarrow \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

3. signature: $\mathsf{sign}(\cdot,\cdot)$, $\mathsf{check}(\cdot,\cdot)$

   $$\longrightarrow \mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = x$$

The two terms involved in a normal execution are:

$$\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skb)), \text{ and } \mathsf{senc}(s, k)$$

Protocols as processes

Applied pi calculus [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

$\longrightarrow$ based on the π-calculus [Milner *et al.*, 92], and in some ways similar to the spi-calculus [Abadi & Gordon, 98]

Protocols as processes

Applied pi calculus [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

$\longrightarrow$ based on the π-calculus [Milner *et al.*, 92], and in some ways similar to the spi-calculus [Abadi & Gordon, 98]

Some advantages:

- allows us to model cryptographic primitives
- both reachability and equivalence-based specification of properties
- powerful proof techniques for hand proofs
- some tool support, e.g. ProVerif (2001) [Blanchet]

  $\longrightarrow$ `http://proverif.rocq.inria.fr/`

Protocols as processes - syntax and semantics

$$
\text{Syntax :}\quad P, Q \quad := \quad
\begin{array}{ll}
0 & \text{null process} \\
\mathsf{in}(c, x).P & \text{input} \\
\mathsf{out}(c, u).P & \text{output} \\
\text{if } u = v \text{ then } P \text{ else } Q & \text{conditional} \\
P \mid Q & \text{parallel composition} \\
!P & \text{replication} \\
\text{new } n.P & \text{fresh name generation}
\end{array}
$$

Protocols as processes - syntax and semantics

Syntax : $P, Q \quad := \quad 0$ — null process

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

Semantics $\rightarrow$:

Comm	$\mathsf{out}(c, M).P \mid \mathsf{in}(c, x).Q \rightarrow P \mid Q\{M/x\}$
Then	if $M = N$ then P else $Q \rightarrow P$ when $M =_{\mathsf{E}} N$
Else	if $M = N$ then P else $Q \rightarrow Q$ when $M \neq_{\mathsf{E}} N$

closed by structural equivalence ($\equiv$) and application of evaluation contexts.

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lll} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lll} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\ \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \qquad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in}\ldots \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \quad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in} \ldots \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \mathsf{let}\ y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a)\ \mathsf{in} \\ & & \quad \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{rcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{rcl} P_A(sk_a, pk_b) & = & \text{new } k.\,\mathrm{out}(c, \mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b)). \\ & & \quad \mathrm{in}(c, x_a).\ \text{let } y_a = \mathrm{sdec}(x_a, k) \text{ in} \ldots \end{array}$$

$$\begin{array}{rcl} P_B(sk_b, pk_a) & = & \mathrm{in}(c, x_b).\ \text{let } y_b = \mathrm{check}(\mathrm{adec}(x_b, sk_b), pk_a) \text{ in} \\ & & \quad \text{new } s.\mathrm{out}(c, \mathrm{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$P_{\mathrm{DS}} \;=\; \text{new } sk_a, sk_b.(P_A(sk_a, \mathrm{pk}(sk_b)) \mid P_B(sk_b, \mathrm{pk}(sk_a)))$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{rcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\ \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \qquad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in} \ldots \end{array}$$

$$\begin{array}{rcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \mathsf{let}\ y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a)\ \mathsf{in} \\ & & \qquad \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \mathsf{new}\ sk_a, sk_b.(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))) \\ & \to & \mathsf{new}\ sk_a, sk_b, k.(\mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in}\ \ldots \\ & & \qquad \mid\ \mathsf{let}\ y_b = k\ \mathsf{in}\ \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b))) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \quad \mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in...} \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \text{let } y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) \text{ in} \\ & & \quad \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \text{new } sk_a, sk_b.(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))) \\ & \rightarrow & \text{new } sk_a, sk_b, k.(\mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in } \ldots \\ & & \quad \mid \text{ let } y_b = k \text{ in new } s.\mathsf{out}(c, \mathsf{senc}(s, y_b))) \\ & \rightarrow & \text{new } sk_a, sk_b, k, s.(\text{ let } y_a = \mathsf{sdec}(\mathsf{senc}(s, k), k) \text{ in } \ldots \mid 0) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathrm{new}\ k.\,\mathrm{out}(c, \mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b)). \\ & & \quad \mathrm{in}(c, x_a).\ \mathrm{let}\ y_a = \mathrm{sdec}(x_a, k)\ \mathrm{in} \ldots \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathrm{in}(c, x_b).\ \mathrm{let}\ y_b = \mathrm{check}(\mathrm{adec}(x_b, sk_b), pk_a)\ \mathrm{in} \\ & & \quad \mathrm{new}\ s.\mathrm{out}(c, \mathrm{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$\begin{array}{lcl} P_{\mathrm{DS}} & = & \mathrm{new}\ sk_a, sk_b.(P_A(sk_a, \mathrm{pk}(sk_b)) \mid P_B(sk_b, \mathrm{pk}(sk_a))) \\ & \to & \mathrm{new}\ sk_a, sk_b, k.(\mathrm{in}(c, x_a).\ \mathrm{let}\ y_a = \mathrm{sdec}(x_a, k)\ \mathrm{in}\ \ldots \\ & & \qquad \mid\ \mathrm{let}\ y_b = k\ \mathrm{in}\ \mathrm{new}\ s.\mathrm{out}(c, \mathrm{senc}(s, y_b))) \\ & \to & \mathrm{new}\ sk_a, sk_b, k, s.(\ \mathrm{let}\ y_a = \mathrm{sdec}(\mathrm{senc}(s, k), k)\ \mathrm{in}\ \ldots \mid 0) \end{array}$$

$\longrightarrow$ this simply models a normal execution between two honest participants

Security properties - confidentiality

Confidentiality for process P w.r.t. secret s

For all processes A such that $A \mid P \rightarrow^* Q$, we have that Q is not of the form $C[\mathrm{out}(c, s).Q']$ with c public.

Security properties - confidentiality

Confidentiality for process P w.r.t. secret s

For all processes A such that $A \mid P \rightarrow^* Q$, we have that Q is not of the form $C[\mathsf{out}(c, s).Q']$ with c public.

Some difficulties:

- we have to consider all the possible executions in presence of an arbitrary adversary (modelled as a process)
- we have to consider realistic initial configurations
  $\longrightarrow$ replications to model an unbounded number of sessions,
  $\longrightarrow$ reveal public keys and private keys to model dishonest agents,
  $\longrightarrow$ P_A/P_B may play with other (and perhaps) dishonest agents, ...

Going back to the Denning Sacco protocol

$$\begin{array}{rcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

$$1.\ A \to C : \ \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C))$$

$$\begin{array}{ll} 2. & C(A) \to B : \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ 3. & \qquad B \to A : \mathsf{senc}(s, k) \end{array}$$

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a.\mathsf{new}\ sk_b.(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a)))$$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

$$1.\ A \to C : \ \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C))$$

$$\begin{array}{ll} 2. & C(A) \to B : \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ 3. & \quad B \to A : \mathsf{senc}(s, k) \end{array}$$

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a.\mathsf{new}\ sk_b.(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a)))$$

Exercise: Exhibit the process A (the behaviour of the attacker) that witnesses the aforementioned attack.

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Example 1: $\mathrm{out}(a, s) \stackrel{?}{\approx} \mathrm{out}(a, s')$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Example 1: $\mathsf{out}(a, s) \not\approx \mathsf{out}(a, s')$

$$\longrightarrow \quad A = \mathsf{in}(a, x).\mathsf{if}\ x = s\ \mathsf{then}\ \mathsf{out}(c, ok)$$

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Example 2:

$$\mathsf{new}\ s.\mathsf{out}(a, \mathsf{senc}(s, k)).\mathsf{out}(a, \mathsf{senc}(s, k'))$$
$$\stackrel{?}{\approx}$$
$$\mathsf{new}\ s, s'.\mathsf{out}(a, \mathsf{senc}(s, k)).\mathsf{out}(a, \mathsf{senc}(s', k'))$$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Example 2:

$$\begin{gathered}\text{new } s.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s, k')) \\ \not\approx \\ \text{new } s, s'.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s', k'))\end{gathered}$$

$$\longrightarrow A = \text{in}(a, x).\text{in}(a, y).\text{if } (\text{sdec}(x, k) = \text{sdec}(y, k')) \text{ then out}(c, ok)$$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

testing equivalence between P and Q, $P \approx Q$

for all processes A, we have that:

$$(A \mid P) \Downarrow_c \text{ if, and only if, } (A \mid Q) \Downarrow_c$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Exercise: Are the two following processes in testing equivalence?

$$\mathsf{new}\, s.\mathsf{out}(a, s) \overset{?}{\approx} \mathsf{new}\, s.\mathsf{new}\, k.\mathsf{out}(a, \mathsf{enc}(s, k))$$

Going back to the e-passport protocol

What does unlinkability mean?

Informally, an observer can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.

Going back to the e-passport protocol

What does unlinkability mean?

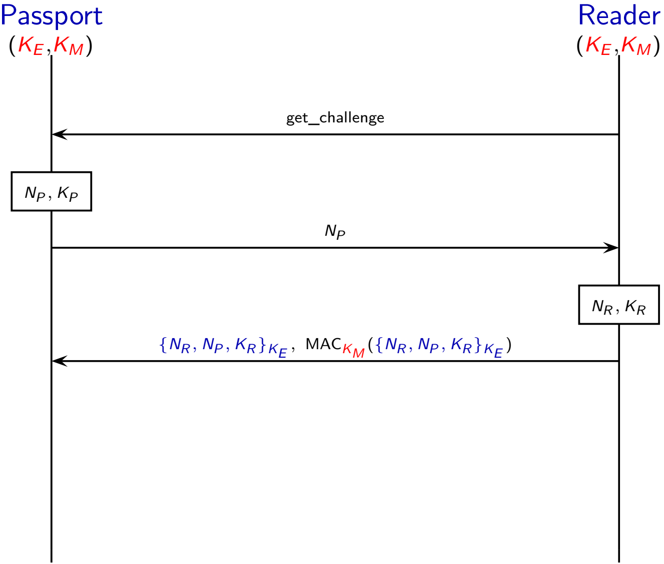
Informally, an observer can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.

More precisely, we have that:

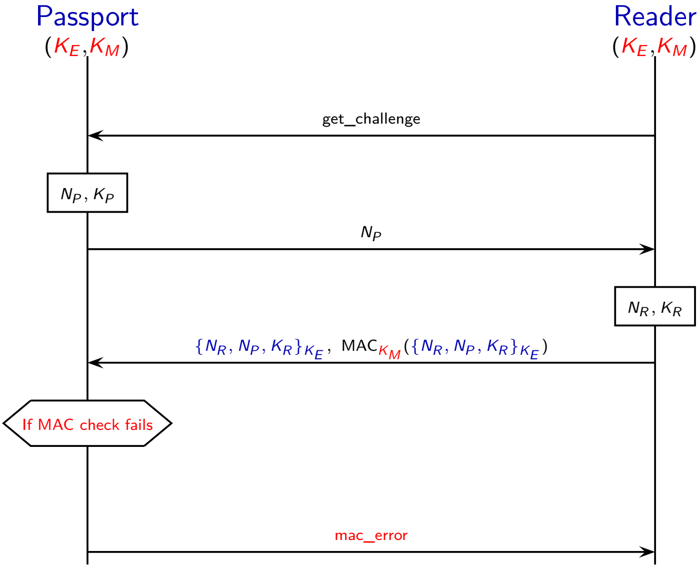
$$!\mathrm{new}\ ke.\mathrm{new}\ km.(!P_{\mathrm{Pass}} \mid !P_{\mathrm{Reader}}) \overset{?}{\approx} !\mathrm{new}\ ke.\mathrm{new}\ km.(\ P_{\mathrm{Pass}} \mid !P_{\mathrm{Reader}})$$

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



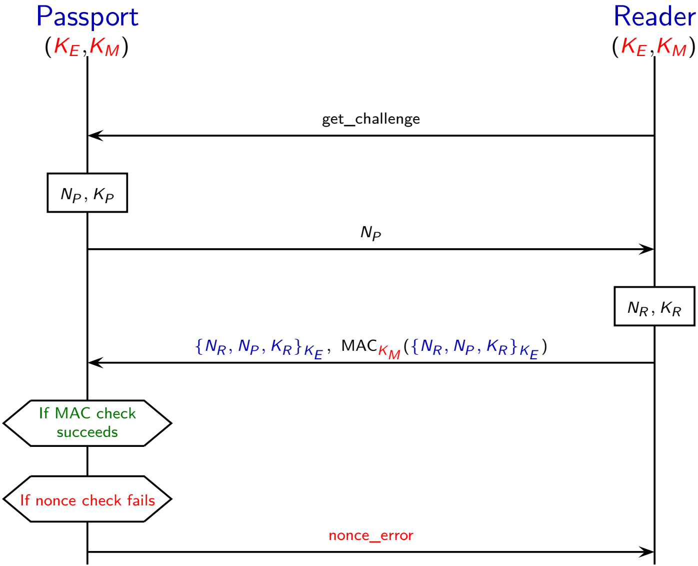
French electronic passport

$\longrightarrow$ 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

Attack against unlinkability [Chothia & Smirnov, 10]

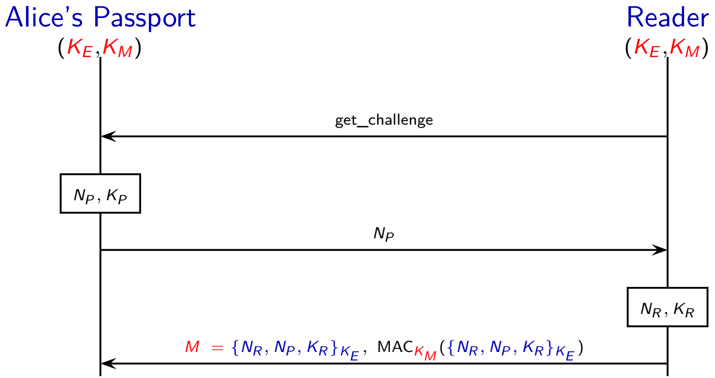
An attacker can track a French passport, provided he has once witnessed a successful authentication.

An attack on the French passport

Attack against unlinkability [Chothia & Smirnov, 10]

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

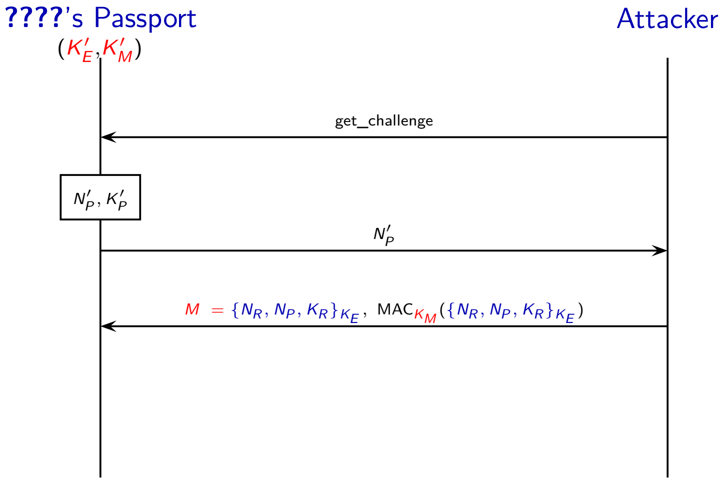
Part 1 of the attack. The attacker eavesdropes on Alice using her passport and records message M.



An attack on the French passport

Part 2 of the attack.

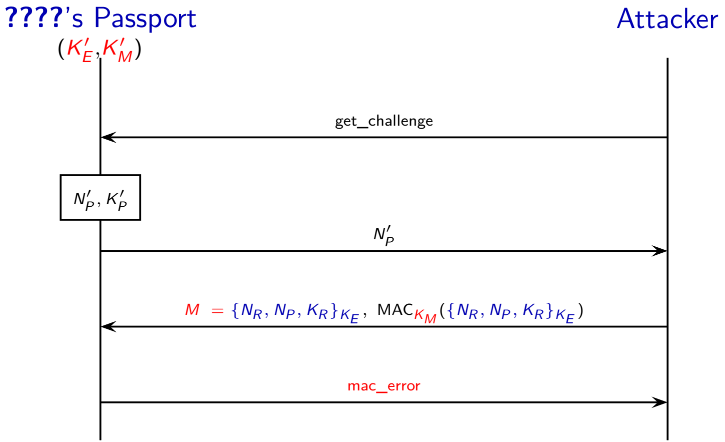
The attacker replays the message M and checks the error code he receives.



An attack on the French passport

Part 2 of the attack.

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

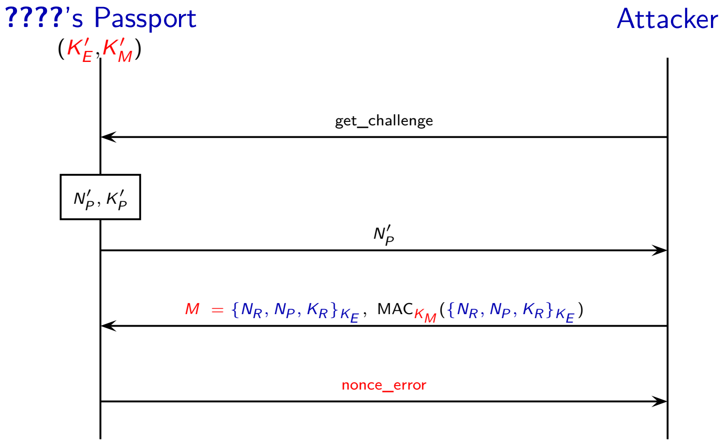


$\Longrightarrow$ MAC check failed $\Longrightarrow$ $K'_M \neq K_M$ $\Longrightarrow$ **????** is not Alice

An attack on the French passport

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

An attack on the French passport

Attack !

The equivalence does not hold: $P_{\text{same}} \not\approx P_{\text{diff}}$.

More formally,

$$P_{\text{same}} \stackrel{\text{def}}{=} !\text{new } ke.\text{new } km.(!P_{\text{Pass}} \mid !P_{\text{Reader}})$$
$$\not\approx$$
$$P_{\text{diff}} \stackrel{\text{def}}{=} !\text{new } ke.\text{new } km.(\ P_{\text{Pass}} \mid !P_{\text{Reader}})$$

An attack on the French passport

Attack !

The equivalence does not hold: $P_{\mathsf{same}} \not\approx P_{\mathsf{diff}}$.

More formally,

$$P_{\mathsf{same}} \stackrel{\mathsf{def}}{=} !\mathsf{new}\ ke.\mathsf{new}\ km.(!P_{\mathsf{Pass}} \mid !P_{\mathsf{Reader}})$$
$$\not\approx$$
$$P_{\mathsf{diff}} \stackrel{\mathsf{def}}{=} !\mathsf{new}\ ke.\mathsf{new}\ km.(\ P_{\mathsf{Pass}} \mid !P_{\mathsf{Reader}})$$

Exercise: Exhibit the process A that witnesses the attack.

Part II

Designing verification algorithms

1. From confidentiality ...

   $\longrightarrow$ *i.e.* trace-based security properties

2. ... to privacy

   $\longrightarrow$ *i.e.* equivalence-based security properties

Designing verification algorithms

1. **From confidentiality ...**

   $\longrightarrow$ ***i.e.* trace-based security properties**

2. ... to privacy

   $\longrightarrow$ *i.e.* equivalence-based security properties

State of the art in a nutshell

for analysing trace-based security properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]

$\longrightarrow$ ProVerif: A tool that does not correspond to any decidability result but works well in practice. [Blanchet, 01]

State of the art in a nutshell

for analysing trace-based security properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]

$\longrightarrow$ ProVerif: A tool that does not correspond to any decidability result but works well in practice. [Blanchet, 01]

Bounded number of sessions

- a decidability result (NP-complete) [Rusinowitch & Turuani, 01; Millen & Shmatikov, 01]
- result extended to deal with various cryptographic primitives.

$\longrightarrow$ various automatic tools, e.g. AVISPA platform [Armando *et al.*, 05]

Skip

The deduction problem: is u deducible from T?

We consider a signature $\mathcal{F}$ and an equational theory E.

The deduction problem

input A sequence ϕ of ground terms (*i.e.* messages) and a term s (the secret) $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$

output Can the attacker learn s from ϕ, *i.e.* does there exist a term (called recipe) R built using public symbols and $w_1, \ldots, w_n$ such that $R\phi =_{\mathsf{E}} s$.

The deduction problem: is u deducible from T?

We consider a signature $\mathcal{F}$ and an equational theory E.

The deduction problem

input A sequence ϕ of ground terms (*i.e.* messages) and a term s (the secret) $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$

output Can the attacker learn s from ϕ, *i.e.* does there exist a term (called recipe) R built using public symbols and $w_1, \ldots, w_n$ such that $R\phi =_{\mathsf{E}} s$.

Example: Let $\phi = \{w_1 \triangleright \mathsf{pk}(ska);\ w_2 \triangleright \mathsf{pk}(skb);\ w_3 \triangleright skc;$
$w_4 \triangleright \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc));\ w_5 \triangleright \mathsf{senc}(s, k)\}$.

We have that:

- k is deducible from ϕ using $R_1 = \mathsf{check}(\mathsf{adec}(w_4, w_3), w_1)$,
- s is deducible from ϕ using $R_2 = \mathsf{sdec}(w_5, R_1)$.

The ground case

Proposition

The deduction problem is decidable in PTIME for the equational theory modelling the DS protocol (and for many others)

Proof (sketch)

1. saturation of ϕ with its deducible subterm ϕ^+
2. does there exist R such that $R\phi^+ = s$ (syntaxic equality)

The ground case

Proposition

The deduction problem is decidable in PTIME for the equational theory modelling the DS protocol (and for many others)

Proof (sketch)

1. saturation of ϕ with its deducible subterm ϕ^+
2. does there exist R such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

- $\phi = \{w_1 \rhd \mathrm{pk}(ska);\ w_2 \rhd \mathrm{pk}(skb);\ w_3 \rhd skc;$
  $w_4 \rhd \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc));\ w_5 \rhd \mathrm{senc}(s, k)\}.$
- $\phi^+ = \phi \uplus \{w_6 \rhd \mathrm{sign}(k, ska);\ w_7 \rhd k;\ w_8 \rhd s\}.$

The ground case

Proposition

The deduction problem is decidable in PTIME for the equational theory modelling the DS protocol (and for many others)

Proof (sketch)

1. saturation of ϕ with its deducible subterm ϕ^+
2. does there exist R such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

- $\phi = \{w_1 \triangleright \mathsf{pk}(ska);\ w_2 \triangleright \mathsf{pk}(skb);\ w_3 \triangleright skc;$
  $w_4 \triangleright \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc));\ w_5 \triangleright \mathsf{senc}(s, k)\}.$
- $\phi^+ = \phi \uplus \{w_6 \triangleright \mathsf{sign}(k, ska);\ w_7 \triangleright k;\ w_8 \triangleright s\}.$

$\longrightarrow$ The deduction problem is actually decidable for many interesting equational theories.

Confidentiality using the constraint solving approach

$\longrightarrow$ for a bounded number of sessions

Two main steps:

1. A symbolic exploration of all the possible traces

   The infinite number of possible traces (*i.e.* experiment) are represented by a finite set of constraint systems
   $\longrightarrow$ this set can be huge (exponential on the number of sessions) ... but some optimizations are used to reduce this number

2. A decision procedure for deciding whether a constraint system has a solution or not.

   $\longrightarrow$ this algorithm works quite well

Confidentiality via constraint solving

Constraint systems are used to specify confidentiality (or more generally any trace-based property) under a particular scenario.

Protocol rules

- a particular interleaving -

$\mathrm{in}(u_1);$

$\mathrm{out}(v_1); \mathrm{in}(u_2);$

$\ldots$

$\mathrm{out}(v_n)$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 \\ \quad ... \\ T_0, v_1, .., v_n \overset{?}{\vdash} s \end{cases}$$

Confidentiality via constraint solving

Constraint systems are used to specify confidentiality (or more generally any trace-based property) under a particular scenario.

Protocol rules

- a particular interleaving -

$\mathrm{in}(u_1);$

$\mathrm{out}(v_1); \mathrm{in}(u_2);$

$\ldots$

$\mathrm{out}(v_n)$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \overset{?}{\vdash} s \end{cases}$$

Solution of a constraint system $\mathcal{C}$

A substitution σ such that

for every $T \overset{?}{\vdash} u \in \mathcal{C}$, $u\sigma$ *is deducible from* $T\sigma$.

for every $u = v \in \mathcal{C}$ *(resp.* $u \neq v$*),* $u\sigma =_{\mathsf{E}} v\sigma$ *(resp.* $u\sigma \neq_{\mathsf{E}} v\sigma$*)*

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

One possible interleaving:

$$\mathsf{out}(\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)))$$
$$\mathsf{in}(\mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(skb))); \mathsf{out}(\mathsf{senc}(s, x))$$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

One possible interleaving:

$$\begin{array}{l} \mathsf{out}(\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc))) \\ \mathsf{in}(\mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(skb))); \mathsf{out}(\mathsf{senc}(s, x)) \end{array}$$

The associated constraint system is:

$$\begin{array}{rcl} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) & \stackrel{?}{\vdash} & \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(skb)) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc));\ \mathsf{senc}(s, x) & \stackrel{?}{\vdash} & s \end{array}$$

with $T_0 = \{\mathsf{pk}(ska),\ \mathsf{pk}(skb);\ skc\}$.

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B)) \\ B \rightarrow A & : & \mathrm{senc}(s, k) \end{array}$$

One possible interleaving:

$$\begin{array}{l} \mathrm{out}(\mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc))) \\ \mathrm{in}(\mathrm{aenc}(\mathrm{sign}(x, ska), \mathrm{pk}(skb))); \mathrm{out}(\mathrm{senc}(s, x)) \end{array}$$

The associated constraint system is:

$$\begin{array}{rcl} T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc)) & \stackrel{?}{\vdash} & \mathrm{aenc}(\mathrm{sign}(x, ska), \mathrm{pk}(skb)) \\ T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc));\ \mathrm{senc}(s, x) & \stackrel{?}{\vdash} & s \end{array}$$

with $T_0 = \{\mathrm{pk}(ska),\ \mathrm{pk}(skb);\ skc\}$.

Question: Does $\mathcal{C}$ admit a solution?

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

One possible interleaving:

$$\begin{array}{l} \mathsf{out}(\mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc))) \\ \mathsf{in}(\mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(skb))); \mathsf{out}(\mathsf{senc}(s, x)) \end{array}$$

The associated constraint system is:

$$\begin{array}{rcl} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) & \overset{?}{\vdash} & \mathsf{aenc}(\mathsf{sign}(x, ska), \mathsf{pk}(skb)) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc));\ \mathsf{senc}(s, x) & \overset{?}{\vdash} & s \end{array}$$

with $T_0 = \{\mathsf{pk}(ska),\ \mathsf{pk}(skb);\ skc\}$.

Question: Does $\mathcal{C}$ admit a solution? Yes: $x \to k$.

The general case: is the constraint system $\mathcal{C}$ satisfiable?

Main idea: simplify them until reaching $\perp$ or solved forms

Constraint system in solved form

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} x_0 \\ T_0 \cup T_1 \overset{?}{\vdash} x_1 \\ \quad \ldots \\ T_0 \cup T_1 \ldots \cup T_n \overset{?}{\vdash} x_n \end{cases}$$

Question

Is there a solution to such a system ?

The general case: is the constraint system $\mathcal{C}$ satisfiable?

Main idea: simplify them until reaching $\perp$ or solved forms

Constraint system in solved form

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} x_0 \\ T_0 \cup T_1 \overset{?}{\vdash} x_1 \\ \quad \dots \\ T_0 \cup T_1 \ldots \cup T_n \overset{?}{\vdash} x_n \end{cases}$$

Question

Is there a solution to such a system ?

Of course, yes ! Choose $u_0 \in T_0$, and consider the substitution:

$$\sigma = \{x_0 \mapsto u_0, \ldots, x_n \mapsto u_0\}$$

Skip

Simplification rules

$\longrightarrow$ these rules deal with pairs and symmetric encryption only

$$\mathsf{R}_{\mathsf{ax}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow \quad \mathcal{C} \quad \text{if } T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_{\mathsf{unif}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$$

$$\text{if } \sigma = \mathrm{mgu}(t_1, t_2) \text{ where } t_1, t_2 \in st(T) \cup \{u\}$$

$$\mathsf{R}_{\mathsf{fail}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow \quad \perp \quad \text{if } vars(T \cup \{u\}) = \emptyset \text{ and } T \not\vdash u$$

$$\mathsf{R}_{\mathsf{f}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} f(u_1, u_2) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u_1 \wedge T \overset{?}{\vdash} u_2 \quad f \in \{\langle\rangle, \mathrm{senc}\}$$

Simplification rules

$\longrightarrow$ these rules deal with pairs and symmetric encryption only

$$\mathsf{R}_{\mathsf{ax}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow \quad \mathcal{C} \quad \text{if } T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

$$\mathsf{R}_{\mathsf{unif}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$$

$$\text{if } \sigma = \mathrm{mgu}(t_1, t_2) \text{ where } t_1, t_2 \in st(T) \cup \{u\}$$

$$\mathsf{R}_{\mathsf{fail}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} u \quad \rightsquigarrow \quad \perp \quad \text{if } vars(T \cup \{u\}) = \emptyset \text{ and } T \not\vdash u$$

$$\mathsf{R}_{\mathsf{f}}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} f(u_1, u_2) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u_1 \wedge T \overset{?}{\vdash} u_2 \quad f \in \{\langle\rangle, \mathrm{senc}\}$$

Example: $T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc)) \overset{?}{\vdash} \mathrm{aenc}(\mathrm{sign}(x, ska), \mathrm{pk}(skb))$

$$\rightsquigarrow (\text{with } \mathsf{R}_{\mathsf{f}}) \begin{cases} T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc)) \overset{?}{\vdash} \mathrm{sign}(x, ska) \\ T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc)) \overset{?}{\vdash} \mathrm{pk}(skb) \end{cases}$$

Exercice - still about the Denning Sacco protocol

Exercise

Reach a solved form starting with the constraint system:

$$
\begin{array}{rcl}
T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc)) & \overset{?}{\vdash} & \mathrm{aenc}(\mathrm{sign}(x, ska), \mathrm{pk}(skb)) \\
T_0;\ \mathrm{aenc}(\mathrm{sign}(k, ska), \mathrm{pk}(skc));\ \mathrm{senc}(s, x) & \overset{?}{\vdash} & s
\end{array}
$$

You should be able to reach a constraint system in solved form (actually the empty one) in 3 steps.

Hint: $\mathsf{R}_{\mathrm{unif}}$ and R_{ax} twice.

Results on the simplification rules

Given a (well-formed) constraint system $\mathcal{C}$:

Soundness:
If $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ and θ solution of $\mathcal{C}'$ then $\sigma\theta$ is a solution of $\mathcal{C}$.

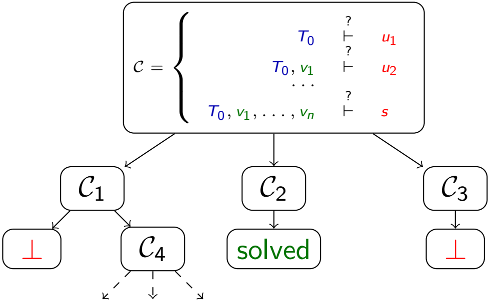
$\longrightarrow$ easy to show

Completeness:
If θ is a solution of $\mathcal{C}$ then there exists $\mathcal{C}'$ and θ' such that $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$, θ' is a solution of $\mathcal{C}'$, and $\theta = \sigma\theta'$.

$\longrightarrow$ more involved to show

Termination: There is no infinite chain $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$.
$\longrightarrow$ using the lexicographic order (number of var, size of rhs)

Main idea of the procedure:

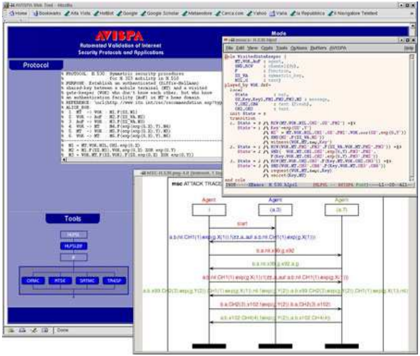


$\longrightarrow$ this gives us a symbolic representation of all the solutions.

Results and tools

Theorem

Deciding confidentiality for a bounded number of sessions is decidable for classical primitives (actually in co-NP).



$\longrightarrow$ This approach has been implemented in the AVISPA Platform

http://www.avispa-project.org/

Part II

Designing verification algorithms

1. From confidentiality ...

   $\longrightarrow$ *i.e.* trace-based security properties

2. **... to privacy**

   $\longrightarrow$ ***i.e.* equivalence-based security properties**

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

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

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

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

$\longrightarrow$ some extensions to go beyond diff-equivalence

Bounded number of sessions

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

$\longrightarrow$ this allows one to decide testing equivalence between "simple" processes without else branch.

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

A recent contribution

$\longrightarrow$ V. Cheval, H. Comon-Lundh, and S. Delaune CCS 2011

Main result

A procedure for deciding testing equivalence for a large class of processes for a bounded number of sessions.

A recent contribution

$\longrightarrow$ V. Cheval, H. Comon-Lundh, and S. Delaune   CCS 2011

Main result

A procedure for deciding testing equivalence for a large class of processes for a bounded number of sessions.

Our class of processes:

- + non-trivial else branches, private channels, and non-deterministic choice;
- – a fixed set of cryptographic primitives (signature, encryption, hash function, mac).

$\longrightarrow$ this allows us in particular to deal with the e-passport example

▸ Skip

The ground case: are ϕ and ψ in static equivalence?

The static equivalence problem

input Two frames ϕ and ψ

$$\phi = \{w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell\} \qquad \psi = \{w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell\}$$

ouput Can the attacker distinguish the two frames, *i.e.* does there exist a test $R_1 \stackrel{?}{=} R_2$ such that:

$$R_1\phi =_E R_2\phi \text{ but } R_1\psi \neq_E R_2\psi \text{ (or the converse).}$$

The ground case: are ϕ and ψ in static equivalence?

The static equivalence problem

input Two frames ϕ and ψ

$$\phi = \{w_1 \rhd u_1, \ldots, w_\ell \rhd u_\ell\} \qquad \psi = \{w_1 \rhd v_1, \ldots, w_\ell \rhd v_\ell\}$$

ouput Can the attacker distinguish the two frames, *i.e.* does there exist a test $R_1 \stackrel{?}{=} R_2$ such that:

$$R_1\phi =_E R_2\phi \text{ but } R_1\psi \neq_E R_2\psi \text{ (or the converse).}$$

Example: Consider the frames:

- $\phi = \{w_1 \rhd \mathrm{aenc}(\langle yes, r_1\rangle, \mathrm{pk}(sks));\ w_2 \rhd sks\}$; and
- $\psi = \{w_1 \rhd \mathrm{aenc}(\langle no, r_2\rangle, \mathrm{pk}(sks));\ w_2 \rhd sks\}$.

They are not in static equivalence: $\mathrm{proj}_1(adec(w_1, w_2)) \stackrel{?}{=} yes$.

The ground case

Proposition

The static equivalence problem is decidable in PTIME for the theory modelling the DS protocol (and for many others)

The ground case

Proposition

The static equivalence problem is decidable in PTIME for the theory modelling the DS protocol (and for many others)

Proof (sketch)

1. saturation of ϕ/ψ with their deducible subterms ϕ^+/ψ^+
2. does there exist a test $R_1 \stackrel{?}{=} R_2$ such that $R_1\phi^+ = R_2\phi^+$ whereas $R_1\psi^+ \neq R_2\psi^+$ (again syntaxic equality) ?
   $\longrightarrow$ Actually, we only need to consider small tests

The ground case

Proposition

The static equivalence problem is decidable in PTIME for the theory modelling the DS protocol (and for many others)

Proof (sketch)

1. saturation of ϕ/ψ with their deducible subterms ϕ^+/ψ^+
2. does there exist a test $R_1 \stackrel{?}{=} R_2$ such that $R_1\phi^+ = R_2\phi^+$ whereas $R_1\psi^+ \neq R_2\psi^+$ (again syntaxic equality) ?
   $\longrightarrow$ Actually, we only need to consider small tests

Going back to the previous example:

- $\phi^+ = \phi \uplus \{w_3 \triangleright \langle yes, r_1\rangle;\ w_4 \triangleright yes;\ w_5 \triangleright r_1\}$, and
- $\psi^+ = \psi \uplus \{w_3 \triangleright \langle no, r_2\rangle;\ w_4 \triangleright no;\ w_5 \triangleright r_2\}$.

$\longrightarrow$ ϕ^+ and ψ^+ are not in static equivalence: $w_4 \stackrel{?}{=} yes$.

Privacy using the constraint solving approach

Two main steps:

1. A symbolic exploration of all the possible traces

   The infinite number of possible traces (*i.e.* experiment) are represented by a finite set of constraint systems

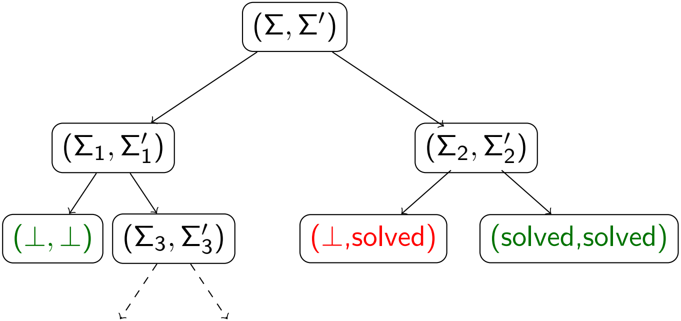
   $\longrightarrow$ this set can be huge (exponential on the number of sessions) !

2. A decision procedure for deciding (symbolic) equivalence between sets of constraint systems

   $\longrightarrow$ this algorithm works quite well

Deciding symbolic equivalence

Main idea: We rewrite pairs (Σ, Σ') of sets of constraint systems (extended to keep track of some information) until a trivial failure or a trivial success is found.



Results on the simplification rules

Termination

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

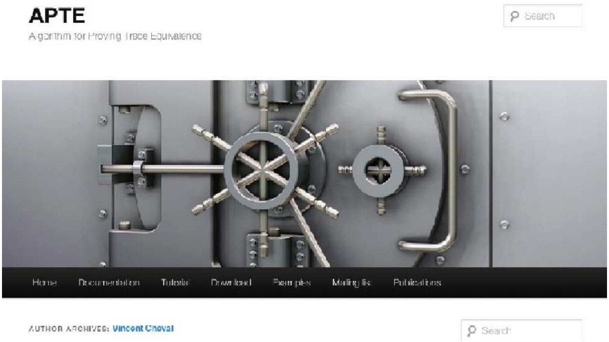
Soundness/Completeness

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

Deciding testing equivalence between processes without replication for classical primitives is decidable.



$\longrightarrow$ This approach has been implemented in APTE by Vincent CHEVAL

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

Limitations of these approaches

1. the algebraic properties of the primitives are abstracted away
$\longrightarrow$ no guarantee if the protocol relies on an encryption that satisfies some additional properties (*e.g.* RSA, ElGamal)

2. only the specification is analysed and not the implementation
$\longrightarrow$ most of the passports are actually linkable by a carefull analysis of time or message length.

   http://www.loria.fr/ glondu/epassport/attaque-tailles.html

3. not all scenario are checked
$\longrightarrow$ no guarantee if the protocol is used one more time !

Conclusion

A need of formal methods in verification of security protocols.
Regarding confidentiality (or authentication), powerful tool support that are nowdays used by industrials and security agencies.

It remains a lot to do for analysing privacy-type properties:

- formal definitions of some sublte security properties
- algorithms (and tools!) for checking automatically trace equivalence for various cryptographic primitives;
- more composition results.

Conclusion

A need of formal methods in verification of security protocols.
Regarding confidentiality (or authentication), powerful tool support that are nowdays used by industrials and security agencies.

It remains a lot to do for analysing privacy-type properties:

- formal definitions of some sublte security properties
- algorithms (and tools!) for checking automatically trace equivalence for various cryptographic primitives;
- more composition results.

VIP - Verification of Indistinguishability Properties

Main topics of the ANR JCJC - VIP project
(Jan. 2012 - Dec 2015)
`http://www.lsv.ens-cachan.fr/Projects/anr-vip/`

$\longrightarrow$ A post-doc position (for 1 year) is available on this project.