Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

LSV, CNRS & ENS Cachan, Université Paris Saclay, France

Monday, June 27th, 2016

Research at LSV

Verification of critical software and systems

Goal: develop the algorithmic foundations for proving correctness and detecting flaws in various types of programs

Applications: computerized systems, databases, security protocols

LSV in figures

- founded in 1997
- around 25 permanents + 15 PhD students
- 6 research teams

SECSI team

Security of Information Systems

- 4 permanents: David Baelde, H. Comon-Lundh, S. Delaune, et J. Goubault-Larrecq.




- 1 engineer + 1 postdoc
- 3 PhD students

Cryptographic protocols everywhere !

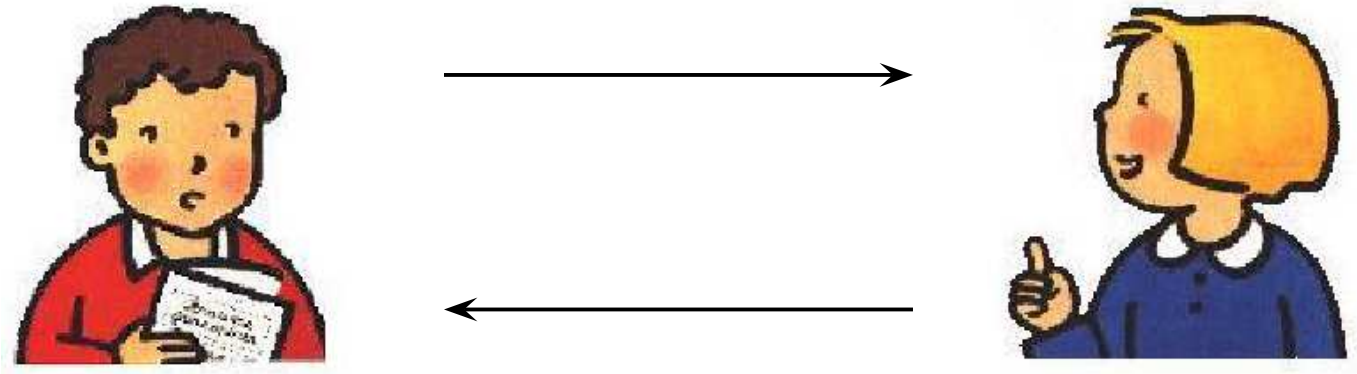
Goal: they aim at securing communications over public/insecure networks

A variety of security properties

- **Secrecy**: May an intruder learn some secret message exchanged 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.
- ...

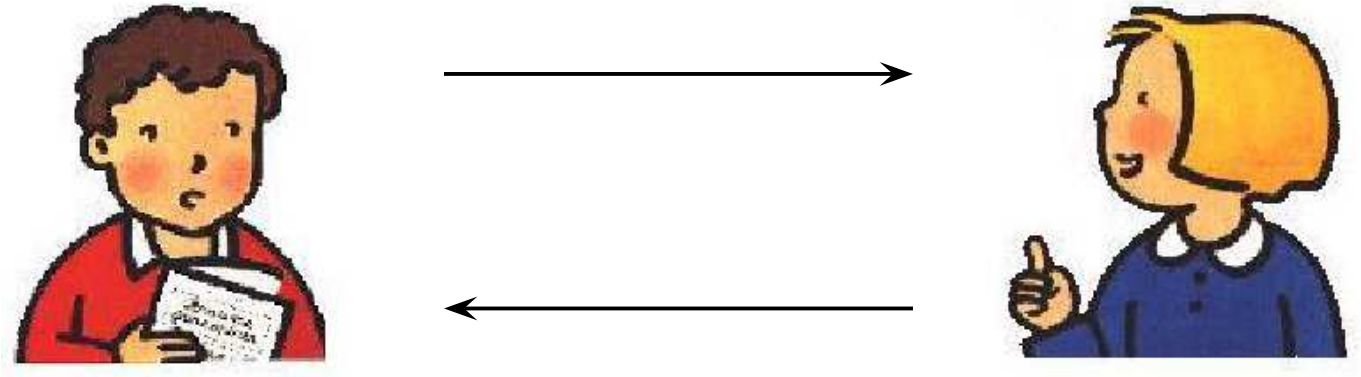
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



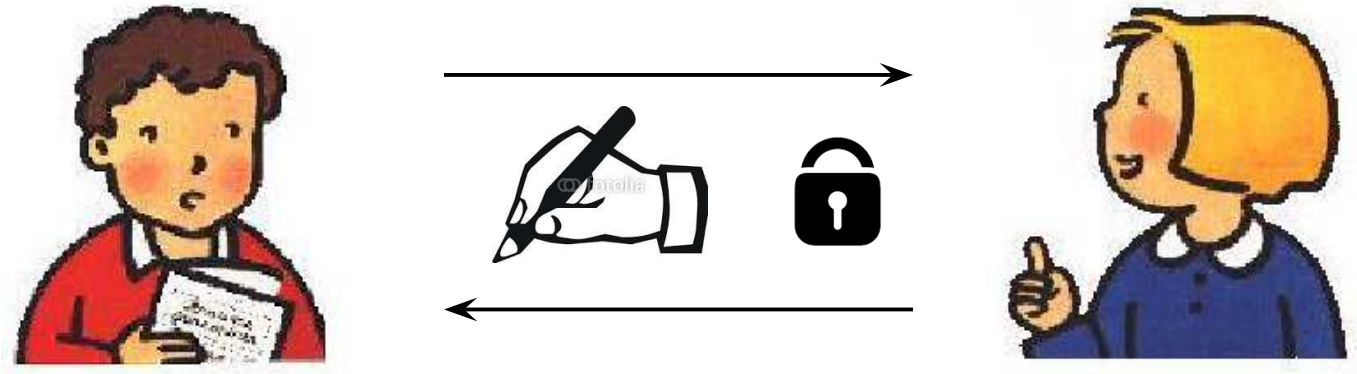
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages

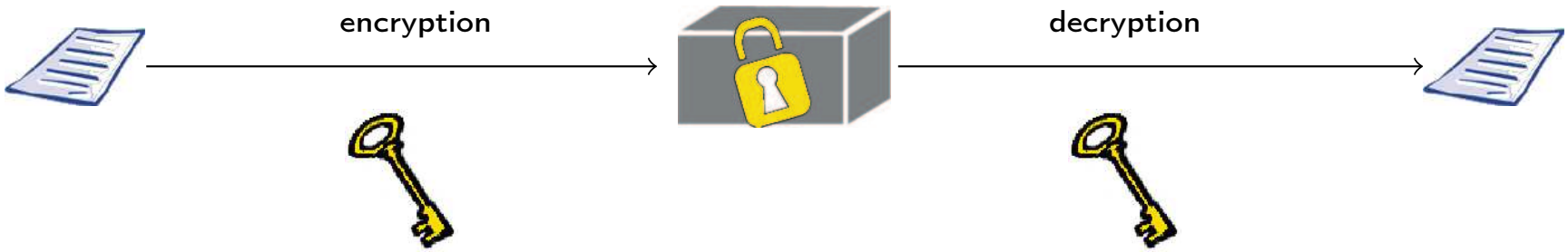


Cryptographic: make use of cryptographic primitives

Examples: symmetric encryption, asymmetric encryption, signature, hashes, ...

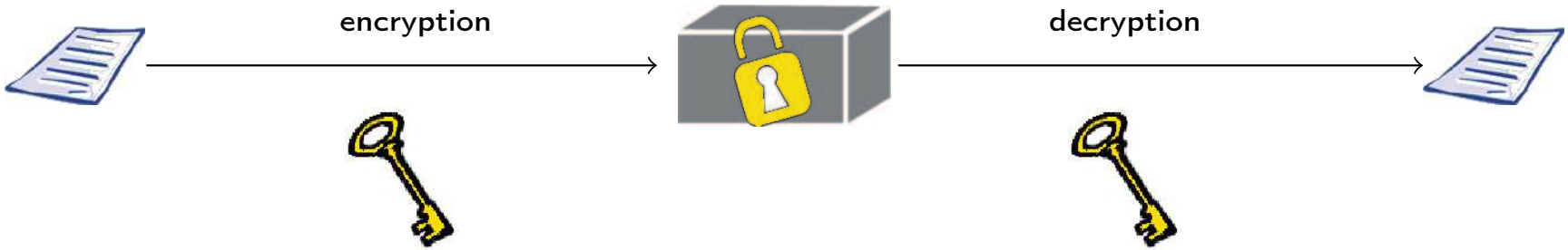
What is a symmetric encryption scheme?

Symmetric encryption

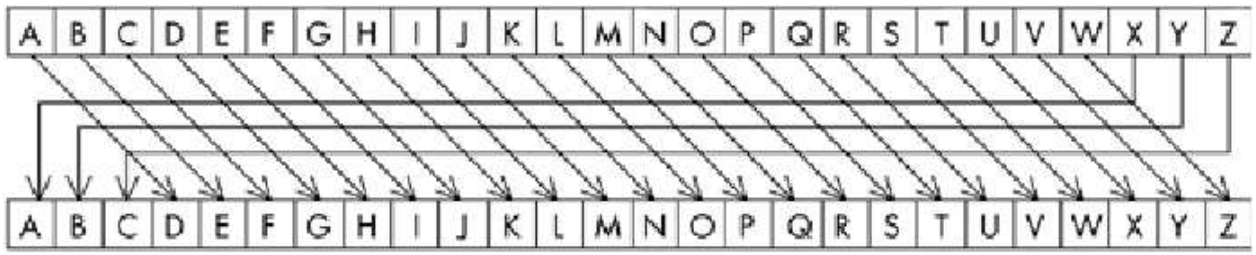


What is a symmetric encryption scheme?

Symmetric encryption



Example: This might be as simple as shifting each letter by a number of places in the alphabet (e.g. Caesar cipher)



Today: DES (1977), AES (2000)

A famous example

Enigma machine (1918-1945)

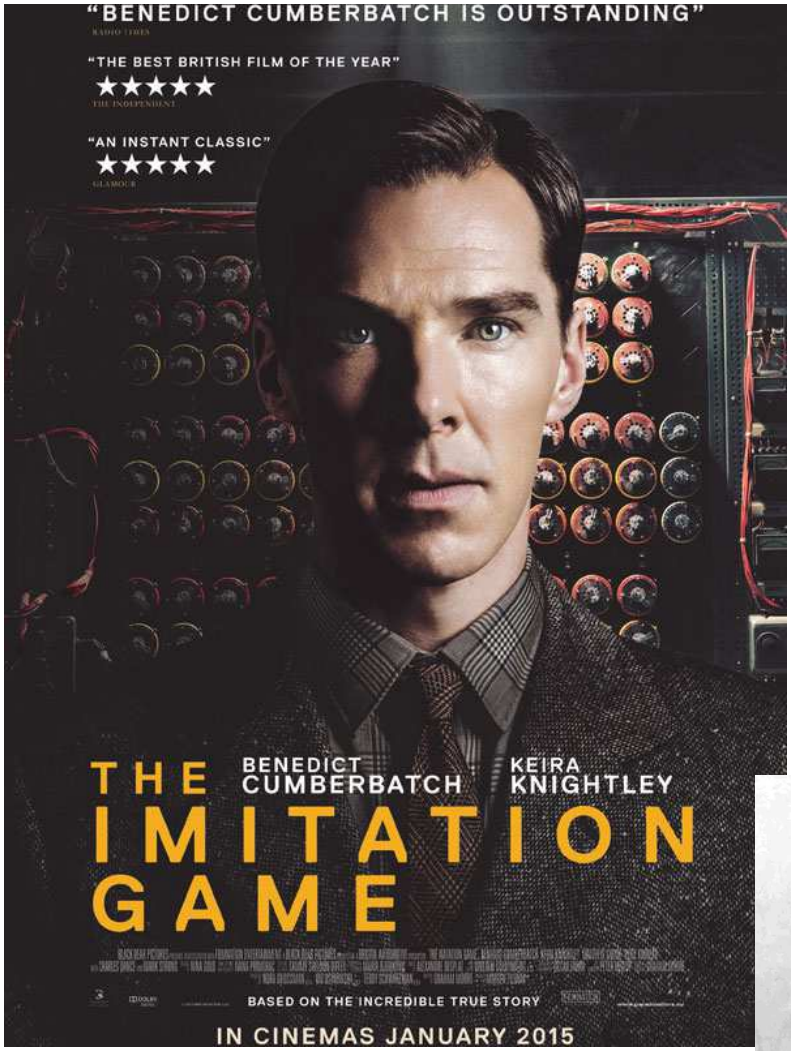
- electro-mechanical rotor cipher machines used by the German to encrypt during Wold War II
- permutations and substitutions

A bit of history

- 1918: invention of the Enigma machine
- 1940: Battle of the Atlantic during which Alan Turing's Bombe was used to test Enigma settings.

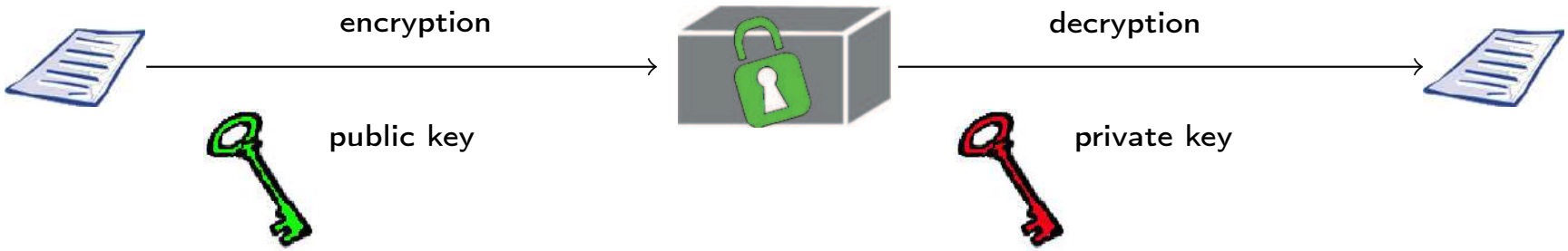
$\longrightarrow$ Everything about the breaking of the Enigma cipher systems remained secret until the mid-1970s.

Advertisement



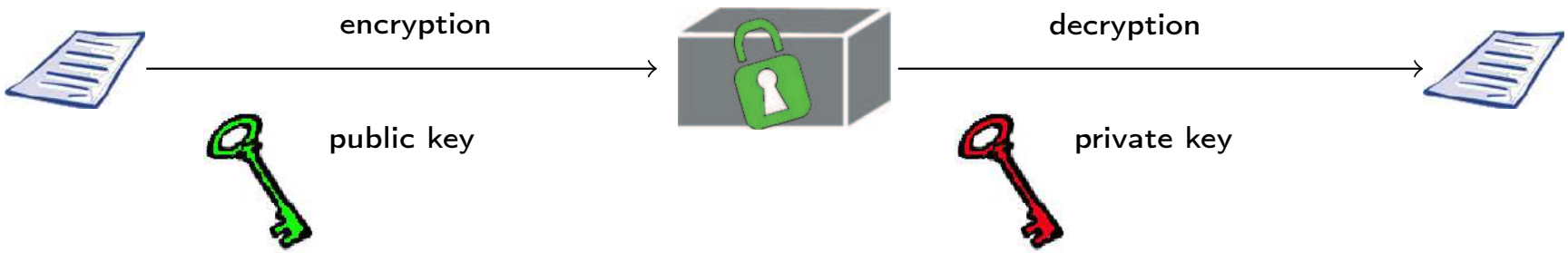
What is an asymmetric encryption scheme?

Asymmetric encryption



What is an asymmetric encryption scheme?

Asymmetric encryption



Examples:

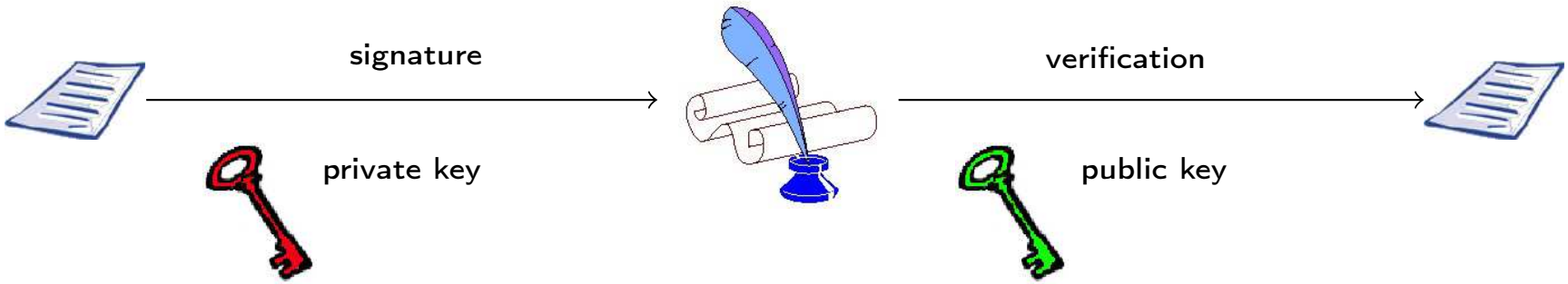
- 1976: first system published by W. Diffie, and M. Hellman,
- 1977: RSA system published by R. Rivest, A. Shamir, and L. Adleman.

$\longrightarrow$ their security relies on well-known mathematical problems (*e.g.* factorizing large numbers, computing discrete logarithms)

Today: those systems are still in use **Prix Turing 2016**

What is a signature scheme?

Signature



Example:

The RSA cryptosystem (in fact, most public key cryptosystems) can be used as a signature scheme.

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\quad \mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{priv}(A)), \mathsf{pub}(B)) \quad}$$

Is the Denning Sacco protocol a good key exchange protocol?

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\quad \mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{priv}(A)), \mathsf{pub}(B)) \quad}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Example: Denning Sacco protocol (1981)

$$\xrightarrow{\quad \mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{priv}(A)), \mathsf{pub}(B)) \quad}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:

$$\xrightarrow{\quad \mathsf{aenc}(\mathsf{sign}(k_{AC}, \mathsf{priv}(A)), \mathsf{pub}(C)) \quad}$$

Example: Denning Sacco protocol (1981)

$$\text{aenc}(\text{sign}(k_{AB}, \text{priv}(A)), \text{pub}(B))$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:

$$\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(C))$$

$$\text{sign}(k_{AC}, \text{priv}(A))$$

$$k_{AC}$$

$$\text{aenc}(\text{sign}(k_{AC}, \text{priv}(A)), \text{pub}(B))$$

Exercise

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

Version 1

$$A \to B \quad : \quad \mathsf{aenc}(\langle A, B, \mathsf{sign}(k, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Version 2

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(\langle A, B, k\rangle, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Which version would you prefer to use?

Exercise

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

Version 1

$$A \to B \quad : \quad \mathsf{aenc}(\langle A, B, \mathsf{sign}(k, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

Version 2

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(\langle A, B, k\rangle, \mathsf{priv}(A))\rangle, \mathsf{pub}(B))$$

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 ?

Credit Card payment protocol

Serge Humpich case - “ Yescard “ (1997)

Credit Card payment protocol

Serge Humpich case - “ Yescard “ (1997)

Step 1: A logical flaw in the protocol allows one to copy a card and to use it without knowing the PIN code.

$\longrightarrow$ not a real problem, there is still a bank account to withdraw

Credit Card payment protocol

Serge Humpich case - “ Yescard “ (1997)

Step 1: A logical flaw in the protocol allows one to copy a card and to use it without knowing the PIN code.

$\longrightarrow$ not a real problem, there is still a bank account to withdraw

Step 2: breaking encryption via factorisation of the following (96 digits) number: 213598703592091008239502270499962879705109534182641740644252416500858395774644508840500943086599 9

$\longrightarrow$ now, the number that is used is made of 232 digits

HTTPS connections

Lots of bugs and attacks, with fixes every month

FREAK attack discovered by Baraghavan et al (Feb. 2015)

1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

$\longrightarrow$ 'export-grade' were introduced under the pressure of US governments agencies to ensure that they would be able to decrypt all foreign encrypted communication.

Electronic passport

$\longrightarrow$ studied in [Arapinis *et al.*, 10]

This 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

$\longrightarrow$ studied in [Arapinis *et al.*, 10]

This is a passport with an RFID tag embedded in it.

The RFID tag stores:

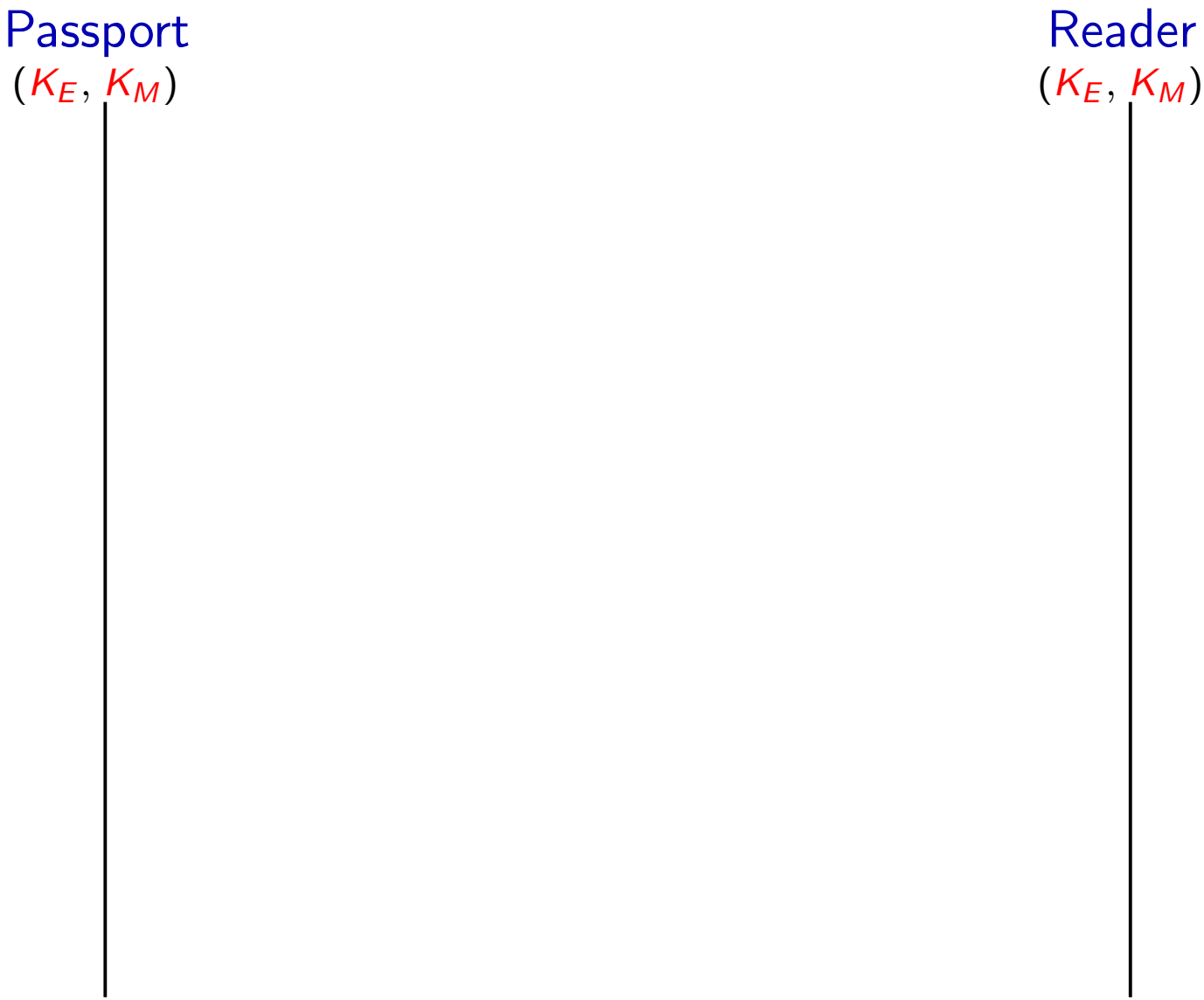
- the information printed on your passport,
- a JPEG copy of your picture.

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

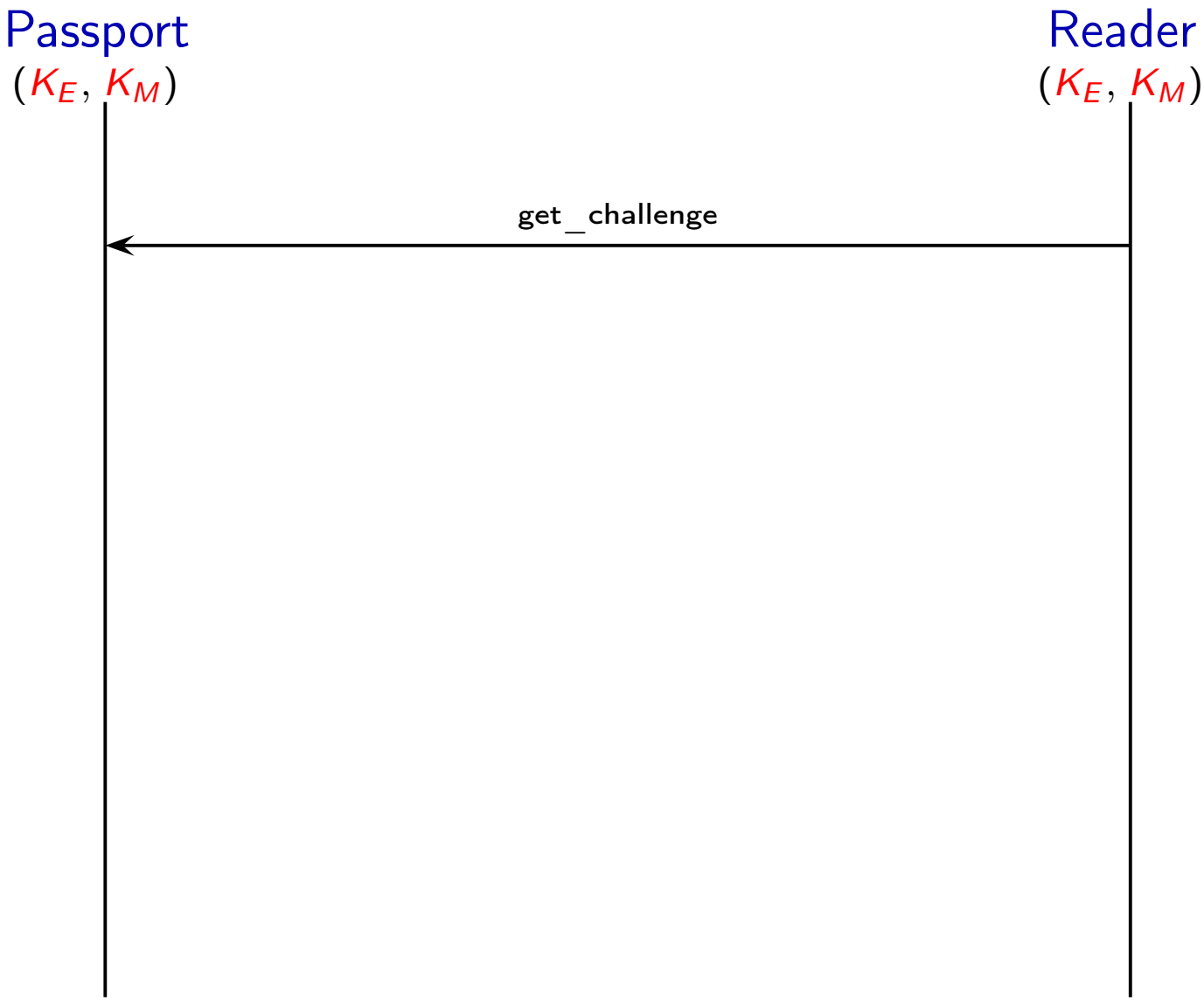
ISO/IEC standard 15408

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*.

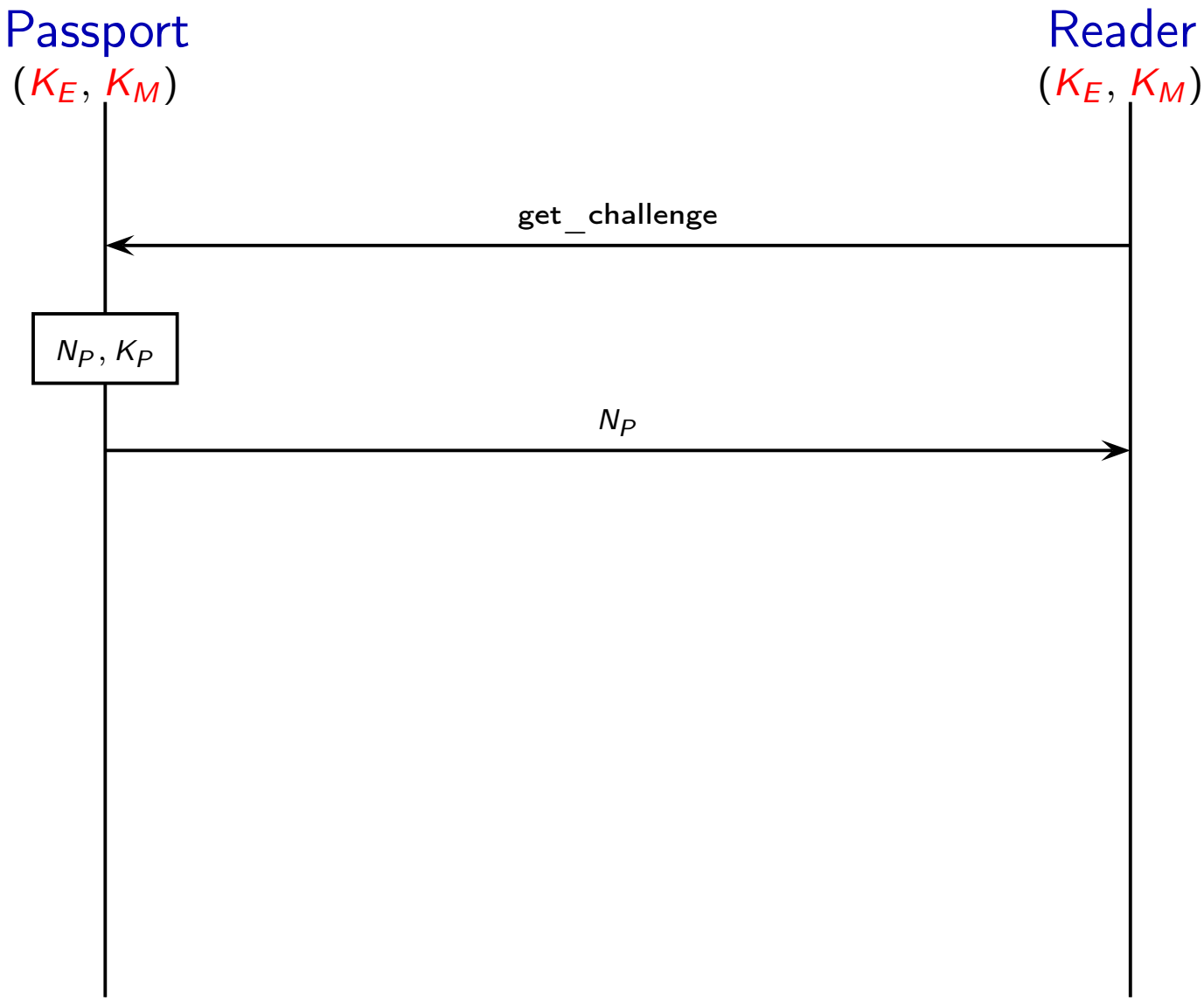
BAC protocol



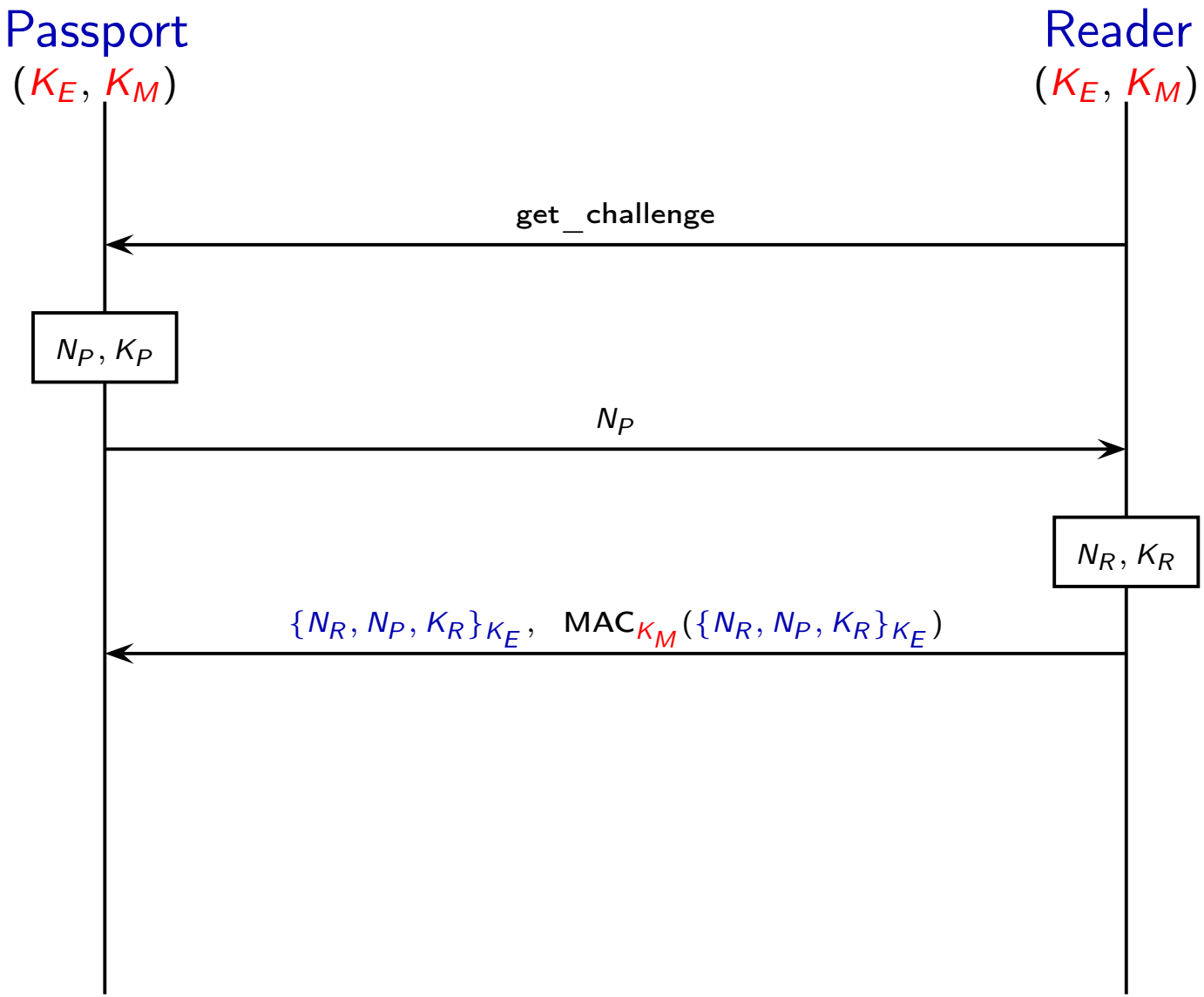
BAC protocol



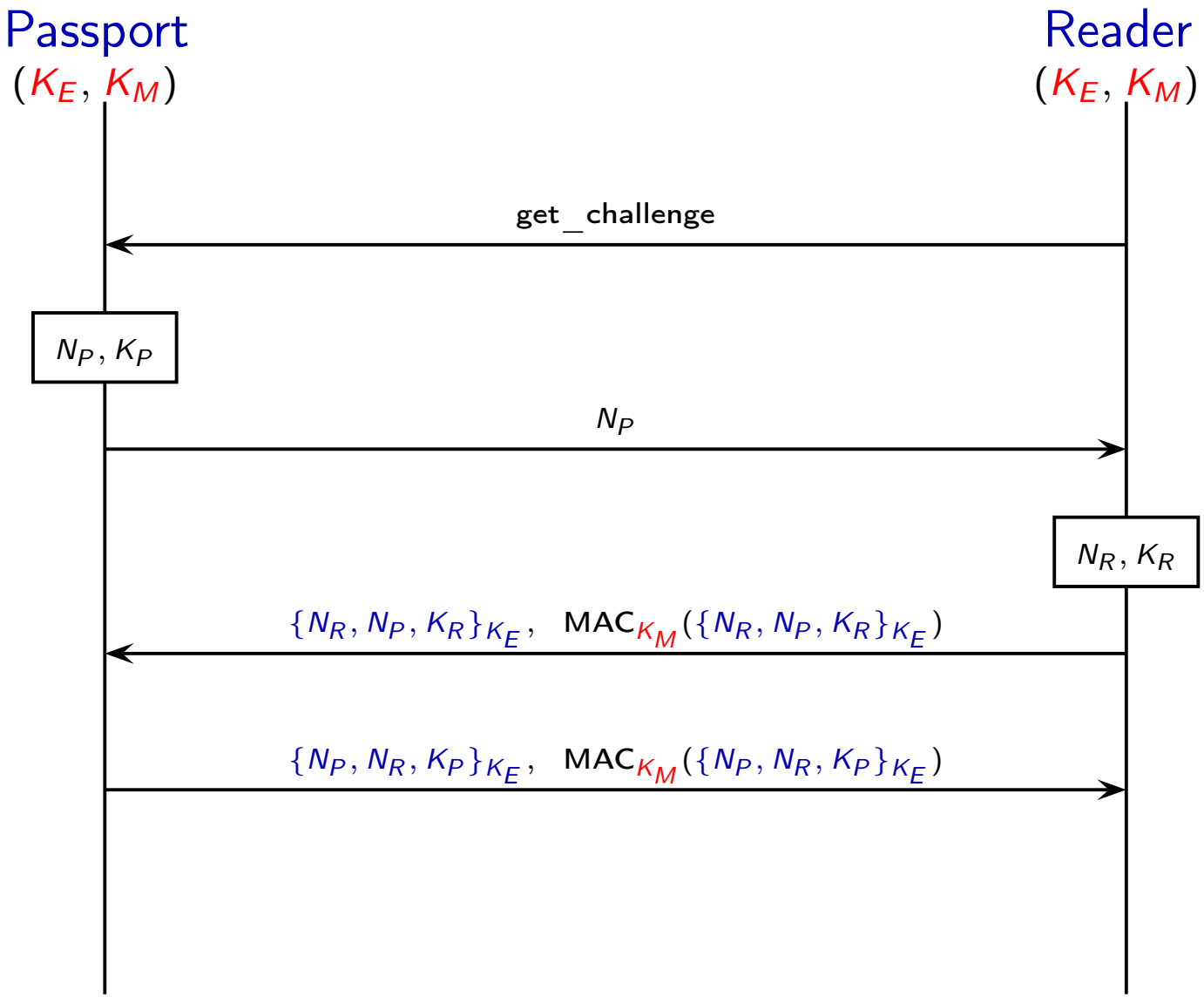
BAC protocol



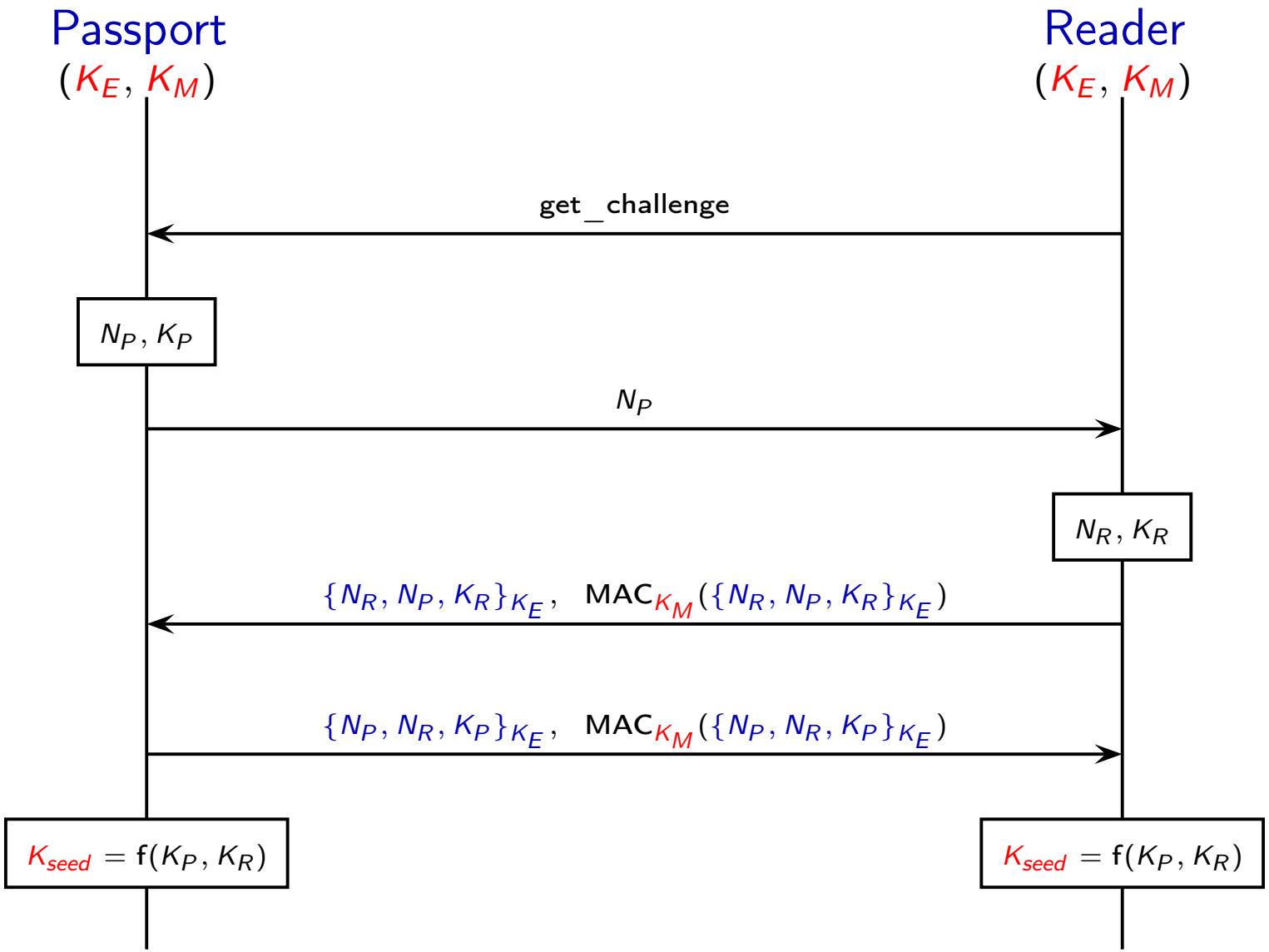
BAC protocol



BAC protocol



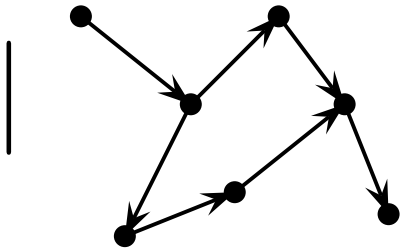
BAC protocol



This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling

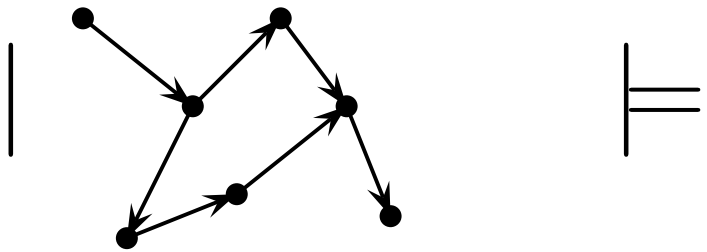


$\models$ φ

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling

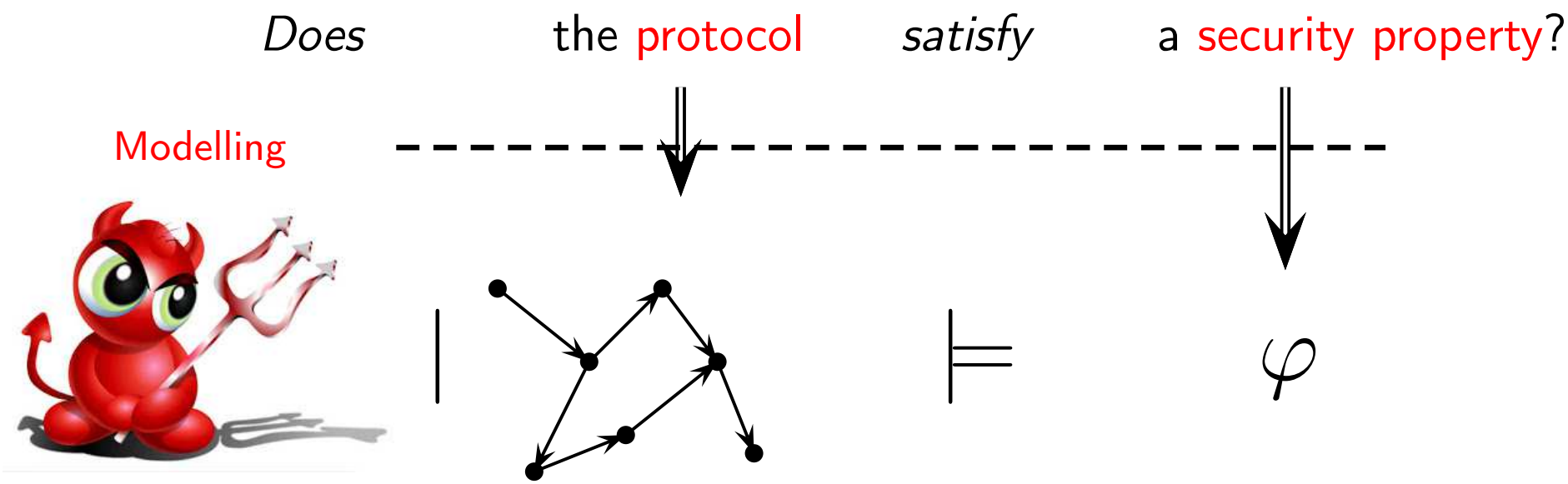


$\models \varphi$

E-passport application

What about unlinkability of the ePassport holders ?

This talk: formal methods for protocol verification



Outline of the this talk

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

Part I

Modelling cryptographic protocols and their security properties

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- + messages are bitstring, a general and powerful adversary
- − manual proofs, tedious and error-prone

Symbolic model

- − abstract model, e.g. messages are terms
- + automatic proofs

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- + messages are bitstring, a general and powerful adversary
- − manual proofs, tedious and error-prone

Symbolic model

- − abstract model, e.g. messages are terms
- + automatic proofs

Some results allowed to make a link between these two very different models.

$\longrightarrow$ Abadi & Rogaway 2000

© Can Stock Photo

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] ...

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

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] ...

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

... but messages that are exchanged are not necessarily atomic !

Messages as terms

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

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

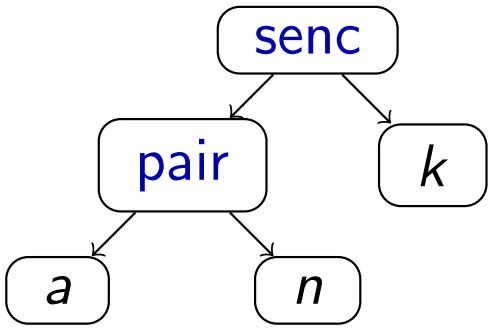
Messages as terms

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

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

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,



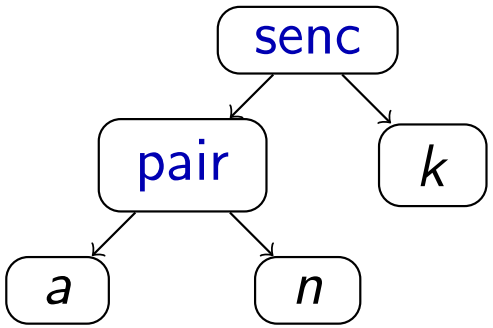
Messages as terms

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

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

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,
- destructors: sdec, proj_1, proj_2.



The term algebra is equipped with an equational theory E.

$$\mathrm{sdec}(\mathrm{senc}(x, y), y) = x \qquad \begin{array}{rcl} \mathrm{proj}_1(\mathrm{pair}(x, y)) & = & x \\ \mathrm{proj}_2(\mathrm{pair}(x, y)) & = & y \end{array}$$

Example: $\mathrm{sdec}(\mathrm{senc}(s, k), k) =_{\mathsf{E}} s$.

Semantics

Semantics $\rightarrow$:

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

Semantics

Semantics $\rightarrow$:

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

closed by

- structural equivalence ($\equiv$):

$$P \mid Q \equiv Q \mid P, \qquad P \mid 0 \equiv P, \qquad \ldots$$

- application of evaluation contexts:

$$\frac{P \rightarrow P'}{\mathsf{new} n.\, P \rightarrow \mathsf{new} n.\, P'} \qquad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$$

Going back to the Denning Sacco protocol (1/2)

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

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

Going back to the Denning Sacco protocol (1/2)

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

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

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

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

Going back to the Denning Sacco protocol (1/2)

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

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: $\mathsf{aenc}(\cdot,\cdot)$, $\mathsf{adec}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$

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

Going back to the Denning Sacco protocol (1/2)

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

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: ok, $\mathsf{sign}(\cdot,\cdot)$, $\mathsf{check}(\cdot,\cdot)$, $\mathsf{getmsg}(\cdot)$

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

Going back to the Denning Sacco protocol (1/2)

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

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: ok, $\mathsf{sign}(\cdot,\cdot)$, $\mathsf{check}(\cdot,\cdot)$, $\mathsf{getmsg}(\cdot)$

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

The two terms involved in a normal execution are:

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

Going back to the Denning Sacco protocol (2/2)

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

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) \quad = \quad \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) = \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

$$P_B(sk_b, pk_a) = \mathsf{in}(c, x_b).\ \text{if } \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$$

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) = \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

$$P_B(sk_b, pk_a) = \mathsf{in}(c, x_b).\ \text{if } \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$$

One possible scenario:

$$P_{\mathsf{DS}} = \text{new } sk_a, sk_b.\left(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\right.$$

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) = \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

$$P_B(sk_b, pk_a) = \mathsf{in}(c, x_b).\ \text{if } \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \text{new } sk_a, sk_b.\left(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\right) \\ & \to & \text{new } sk_a, sk_b, k.\left(\ \mathsf{in}(c, x_a).\ \ldots\right. \\ & & \mid\ \text{if } \mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok} \text{ then} \\ & & \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))))) \end{array}$$

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) = \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

$$P_B(sk_b, pk_a) = \mathsf{in}(c, x_b).\ \text{if } \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \text{new } sk_a, sk_b.\left(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\right) \\ & \to & \text{new } sk_a, sk_b, k.\left(\ \mathsf{in}(c, x_a).\ \ldots\right. \\ & & \left.\mid\ \text{if } \mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))))\right) \\ & \to & \text{new } sk_a, sk_b, k.\left(\ \mathsf{in}(c, x_a).\ \ldots\right. \\ & & \left.\text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \underline{\mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))}))\right) \end{array}$$

Going back to the Denning Sacco protocol (2/2)

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

Alice and Bob as processes:

$$P_A(sk_a, pk_b) = \text{new } k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\mathsf{in}(c, x_a).\ \ldots$$

$$P_B(sk_b, pk_a) = \mathsf{in}(c, x_b).\ \text{if } \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok} \text{ then } \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \text{new } sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big) \\ & \to & \text{new } sk_a, sk_b, k.\big(\ \mathsf{in}(c, x_a).\ \ldots \\ & & \quad \mid\ \text{if } \mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok} \text{ then} \\ & & \quad \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))))\big) \\ & \to & \text{new } sk_a, sk_b, k.\big(\ \mathsf{in}(c, x_a).\ \ldots \\ & & \quad \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, \underline{\mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b))}))\big) \end{array}$$

$\longrightarrow$ 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 $C[\mathsf{out}(c, s).Q']$ 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 $C[\mathrm{out}(c, s).Q']$ 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
  - $\longrightarrow$ an unbounded number of agents,
  - $\longrightarrow$ replications to model an unbounded number of sessions,
  - $\longrightarrow$ reveal public keys and private keys to model dishonest agents,
  - $\longrightarrow$ honest agents may initiate a session with a dishonest agent, ...

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

$$\begin{array}{lll} 1.\ A \to C : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C)) & \\ & 2.\ C(A) \to B : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ & 3. \quad B \to A : & \mathsf{senc}(s, k) \end{array}$$

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

$$P_{\mathsf{DS}} = \mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a) \mid \mathsf{out}(c, \mathsf{pk}(skb)))\big)$$

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

$$\begin{array}{lll} 1.\ A \to C : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(C)) & \\ & 2.\ C(A) \to B : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ & 3.\ \quad B \to A : & \mathsf{senc}(s, k) \end{array}$$

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

$$P_{\mathsf{DS}} = \mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a) \mid \mathsf{out}(c, \mathsf{pk}(skb)))\big)$$

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

$$A \mid P_{\mathsf{DS}} \ \to^* \ C[\mathsf{out}(c, s).Q']$$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

Testing equivalence between P and Q, denoted $P \approx Q$

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, denoted $P \approx Q$

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 1: $\quad \text{out}(a, \mathit{yes}) \overset{?}{\approx} \text{out}(a, \mathit{no})$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

Testing equivalence between P and Q, denoted $P \approx Q$

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 1: $\quad \mathsf{out}(a, \mathit{yes}) \not\approx \mathsf{out}(a, \mathit{no})$

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

Testing equivalence between P and Q, denoted $P \approx Q$

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 2: k and k' are known to the attacker

$$\begin{array}{c} \text{new } s.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s, k')) \\ \stackrel{?}{\approx} \\ \text{new } s, s'.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s', k')) \end{array}$$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

Testing equivalence between P and Q, denoted $P \approx Q$

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 2: k and k' are known to the attacker

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

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

Testing equivalence between P and Q, denoted $P \approx Q$

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 3: Are the two following processes in testing equivalence?

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

Some privacy-type properties

Unlinkability

[Arapinis et al, 2010]

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

many sessions for each passport

only one session for each passport

Some privacy-type properties

Unlinkability

[Arapinis et al, 2010]

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

many sessions for each passport

only one session for each passport

Vote privacy

[Kremer and Ryan, 2005]

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

A votes yes
B votes no

A votes no
B votes yes

Part II

Designing verification algorithms

(from **confidentiality** to privacy)

State of the art in a nutshell

for analysing confidentiality properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99; Rammanujam & Suresh, 03]

$\longrightarrow$ 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 confidentiality properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99; Rammanujam & Suresh, 03]

$\longrightarrow$ 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]

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 $R\phi =_{\mathsf{E}} s$.

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

1. Is k deducible from ϕ?
2. What about 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 $R\phi =_{\mathsf{E}} s$.

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

1. Is k deducible from ϕ? Yes, using $R_1 = \mathsf{getmsg}(\mathsf{adec}(w_4, w_3))$
2. What about 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 $R\phi =_{\mathsf{E}} s$.

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

1. Is k deducible from ϕ? Yes, using $R_1 = \mathsf{getmsg}(\mathsf{adec}(w_4, w_3))$
2. What about s? Yes, using $R_2 = \mathsf{sdec}(w_5, R_1)$.

The deduction problem

Proposition

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

Algorithm

1. saturation of ϕ with its deducible subterms in one-step: ϕ^+
2. does there exist R such that $R\phi^+ = s$ (syntaxic equality)

The deduction problem

Proposition

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

Algorithm

1. saturation of ϕ with its deducible subterms in one-step: ϕ^+
2. does there exist R such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

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

Soundness, completeness, and termination

Soundness If the algorithm returns Yes then u is indeed deducible from ϕ. $\longrightarrow$ easy to prove

Soundness, completeness, and termination

Soundness If the algorithm returns Yes then u is indeed deducible from ϕ.
$\longrightarrow$ easy to prove

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.
$\longrightarrow$ easy to prove for the deduction rules under study

Soundness, completeness, and termination

Soundness If the algorithm returns Yes then u is indeed deducible from ϕ.
$\longrightarrow$ easy to prove

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.
$\longrightarrow$ easy to prove for the deduction rules under study

Completeness If u is deducible from ϕ, then the algorithm returns Yes.

Soundness, completeness, and termination

Soundness If the algorithm returns Yes then u is indeed deducible from ϕ.
$\longrightarrow$ easy to prove

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.
$\longrightarrow$ easy to prove for the deduction rules under study

Completeness If u is deducible from ϕ, then the algorithm returns Yes.
$\longrightarrow$ this relies on a locality property

Locality lemma

Let ϕ be a frame and u be a deducible subterm of ϕ. There exists a recipe R witnessing this fact which satisfies the locality property:

for any R' subterm of R, we have that $R'\phi\downarrow$ is a subterm of ϕ.

Caution !

One should never underestimate the attacker !

The attacker can listen to the communication but also:

- intercept the messages that are sent by the participants,
- build new messages according to his deduction capabilities, and
- send messages on the communication network.

$\longrightarrow$ this is the co-called active attacker

Confidentiality using the constraint solving approach

$\longrightarrow$ active attacker, only for a bounded number of sessions

Confidentiality using the constraint solving approach

$\longrightarrow$ active attacker, only 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

Step 1: confidentiality via constraint solving

We consider a finite sequence of actions:

$$\mathsf{in}(u_1); \mathsf{out}(v_1); \mathsf{in}(u_2); \ldots \mathsf{out}(v_n)$$

We build the following constraint system: $\longrightarrow$ u_i and v_i may contain variables

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

Step 1: confidentiality via constraint solving

We consider a finite sequence of actions:

$$\mathsf{in}(u_1); \mathsf{out}(v_1); \mathsf{in}(u_2); \ldots \mathsf{out}(v_n)$$

We build the following constraint system: $\longrightarrow$ u_i and v_i may contain variables

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \stackrel{?}{\vdash} u_1 \\ T_0, v_1 \stackrel{?}{\vdash} u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \stackrel{?}{\vdash} s \end{array} \right.$$

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

A substitution σ such that: for every $T \stackrel{?}{\vdash} u \in \mathcal{C}$, $u\sigma$ is deducible from $T\sigma$.

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:

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

The associated constraint system is:

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

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

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:

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

The associated constraint system is:

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

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

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:

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

The associated constraint system is:

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

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 $\bot$ or solved forms

Constraint system in solved form

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

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 $\bot$ or solved forms

Constraint system in solved form

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

Question

Is there a solution to such a system ?

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

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

Step 2: simplification rules

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

R_{ax} : $\quad \mathcal{C} \wedge T \stackrel{?}{\vdash} u \quad \rightsquigarrow \quad \mathcal{C}$ if u is deducible from $T \cup \{x \mid T' \stackrel{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\}$

$\mathsf{R}_{\mathsf{unif}}$: $\quad \mathcal{C} \wedge T \stackrel{?}{\vdash} u \quad \rightsquigarrow_{\sigma} \quad \mathcal{C}\sigma \wedge T\sigma \stackrel{?}{\vdash} u\sigma$ if $\sigma = \mathsf{mgu}(t_1, t_2)$ where $t_1, t_2 \in st(T) \cup \{u\}$

$\mathsf{R}_{\mathsf{fail}}$: $\quad \mathcal{C} \wedge T \stackrel{?}{\vdash} u \quad \rightsquigarrow \quad \bot$ if $vars(T \cup \{u\}) = \emptyset$ and $T \nvdash u$

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

Applying rule $\mathsf{R_f}$

$$\mathsf{R_f}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} \mathsf{f}(u_1, u_2) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u_1 \wedge T \overset{?}{\vdash} u_2$$

Example:

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

Applying rule $\mathsf{R_f}$

$$\mathsf{R_f}: \quad \mathcal{C} \wedge T \overset{?}{\vdash} \mathsf{f}(u_1, u_2) \rightsquigarrow \mathcal{C} \wedge T \overset{?}{\vdash} u_1 \wedge T \overset{?}{\vdash} u_2$$

Example:

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

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

Applying rule $\mathsf{R}_{\mathsf{unif}}$

$$\mathsf{R}_{\mathsf{unif}}: \quad \mathcal{C} \wedge T \stackrel{?}{\vdash} u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \stackrel{?}{\vdash} u\sigma$$
$$\text{if } \sigma = \mathsf{mgu}(t_1, t_2) \text{ where } t_1, t_2 \in st(T) \cup \{u\}$$

Example:

$$\begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(x, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

Applying rule $\mathsf{R}_{\mathsf{unif}}$

$$\mathsf{R}_{\mathsf{unif}}: \quad \mathcal{C} \wedge T \stackrel{?}{\vdash} u \quad \rightsquigarrow_\sigma \quad \mathcal{C}\sigma \wedge T\sigma \stackrel{?}{\vdash} u\sigma$$
$$\text{if } \sigma = \mathsf{mgu}(t_1, t_2) \text{ where } t_1, t_2 \in st(T) \cup \{u\}$$

Example:

$$\begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(x, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

$$\rightsquigarrow \begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(k, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

Applying rule R_{ax}

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

Example: (assuming that skc and $\mathsf{pk}(skb)$ are in T_0)

$$\begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(k, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

Applying rule R_{ax}

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

Example: (assuming that skc and $\mathsf{pk}(skb)$ are in T_0)

$$\begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(k, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

$$\rightsquigarrow \begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \stackrel{?}{\vdash} \mathsf{sign}(k, ska) \end{cases}$$

Applying rule R_{ax}

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

Example: (assuming that skc and $\mathsf{pk}(skb)$ are in T_0)

$$\begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \overset{?}{\vdash} \mathsf{sign}(k, ska) \\ T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \overset{?}{\vdash} \mathsf{pk}(skb) \end{cases}$$

$$\rightsquigarrow \begin{cases} T_0;\ \mathsf{aenc}(\mathsf{sign}(k, ska), \mathsf{pk}(skc)) \overset{?}{\vdash} \mathsf{sign}(k, ska) \end{cases}$$

$$\rightsquigarrow \quad \emptyset \quad \text{(empty constraint system)}$$

Results on the simplification rules

R_{ax} : $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \mathcal{C}$ if u is deducible from $T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\}$

$\mathsf{R}_{\mathsf{unif}}$: $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$ if $\sigma = \mathrm{mgu}(t_1, t_2)$ where $t_1, t_2 \in st(T) \cup \{u\}$

$\mathsf{R}_{\mathsf{fail}}$: $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \bot$ if $vars(T \cup \{u\}) = \emptyset$ and $T \nvdash u$

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

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

Results on the simplification rules

R_{ax} : $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \mathcal{C}$ if u is deducible from $T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\}$

$\mathsf{R}_{\mathsf{unif}}$: $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$ if $\sigma = \mathrm{mgu}(t_1, t_2)$ where $t_1, t_2 \in st(T) \cup \{u\}$

$\mathsf{R}_{\mathsf{fail}}$: $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \bot$ if $vars(T \cup \{u\}) = \emptyset$ and $T \nvdash u$

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

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

Exercise: Termination

There is no infinite chain $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \ldots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$.

Results on the simplification rules

R_{ax} : $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow \mathcal{C}$ if u is deducible from $T \cup \{x \mid T' \overset{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\}$

$\mathsf{R}_{\mathsf{unif}}$: $\mathcal{C} \wedge T \overset{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \overset{?}{\vdash} u\sigma$ if $\sigma = \mathrm{mgu}(t_1, t_2)$ where $t_1, t_2 \in st(T) \cup \{u\}$

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

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

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

Exercise: 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)

Results on the simplification rules

R_{ax} : $\mathcal{C} \wedge T \stackrel{?}{\vdash} u \rightsquigarrow \mathcal{C}$ if u is deducible from $T \cup \{x \mid T' \stackrel{?}{\vdash} x \in \mathcal{C}, T' \subsetneq T\}$

$\mathsf{R}_{\mathsf{unif}}$: $\mathcal{C} \wedge T \stackrel{?}{\vdash} u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \stackrel{?}{\vdash} u\sigma$ if $\sigma = \mathsf{mgu}(t_1, t_2)$ where $t_1, t_2 \in st(T) \cup \{u\}$

$\mathsf{R}_{\mathsf{fail}}$: $\mathcal{C} \wedge T \stackrel{?}{\vdash} u \rightsquigarrow \bot$ if $vars(T \cup \{u\}) = \emptyset$ and $T \nvdash u$

R_{f} : $\mathcal{C} \wedge T \stackrel{?}{\vdash} f(u_1, u_2) \rightsquigarrow \mathcal{C} \wedge T \stackrel{?}{\vdash} u_1 \wedge T \stackrel{?}{\vdash} u_2$ $f \in \{\langle\rangle, \mathsf{senc}\}$

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

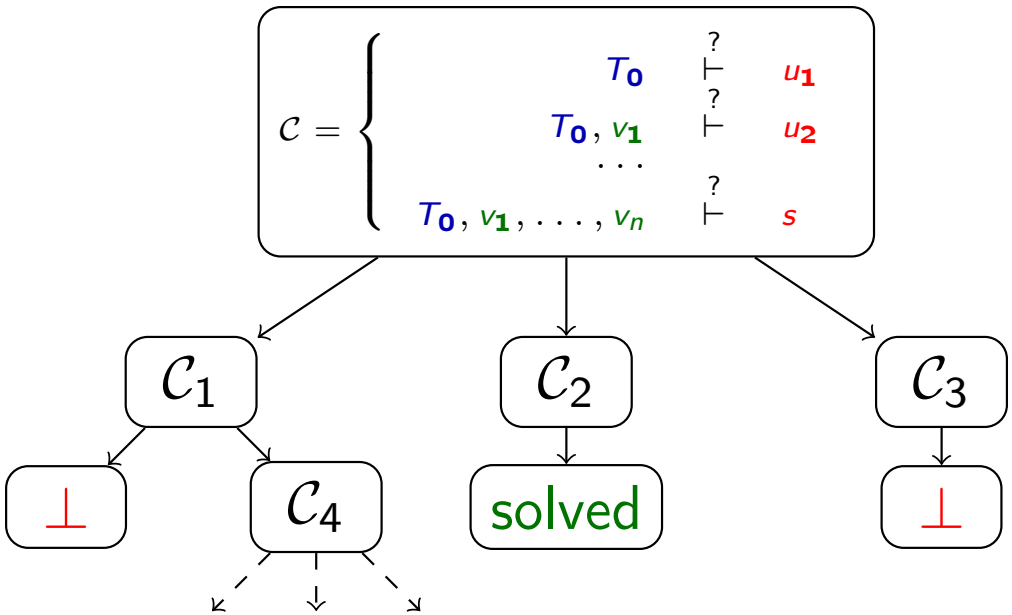
Completeness

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

Step 2: procedure for solving a constraint system

Main idea of the procedure:



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

Main result

Theorem

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

Exercise: NP-hardness can be shown by encoding 3-SAT

Main result

Theorem

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

Exercise: NP-hardness can be shown by encoding 3-SAT

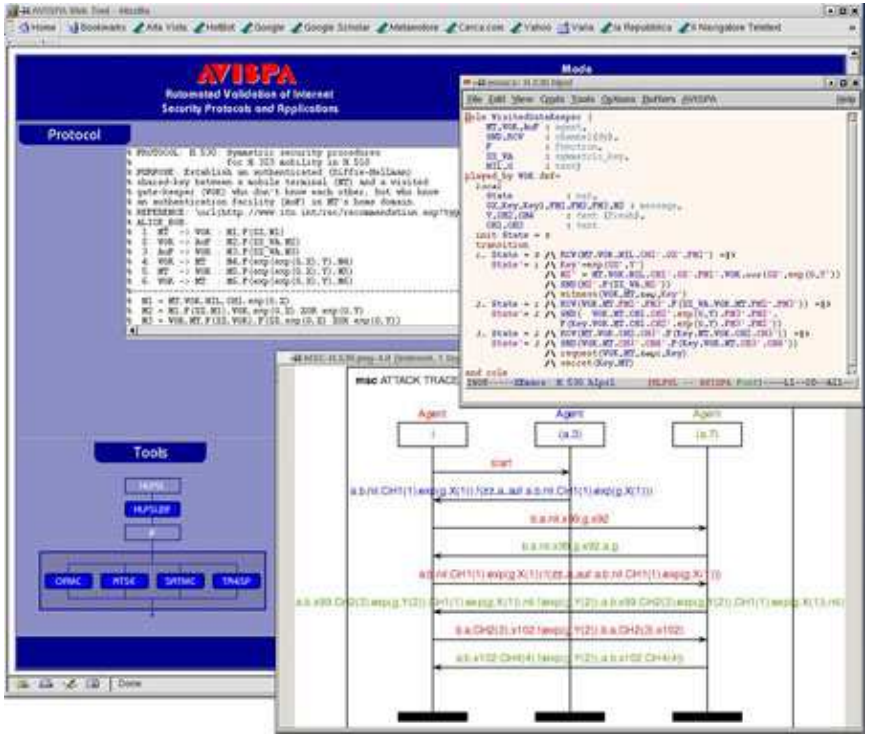
Some extensions that already exist:

1. disequality tests (protocol with else branches)
2. more primitives: asymmetric encryption, blind signature, exclusive-or, ...

Avantssar platform

This approach has been implemented in the Avantssar Platform.

`http://www.avantssar.eu`



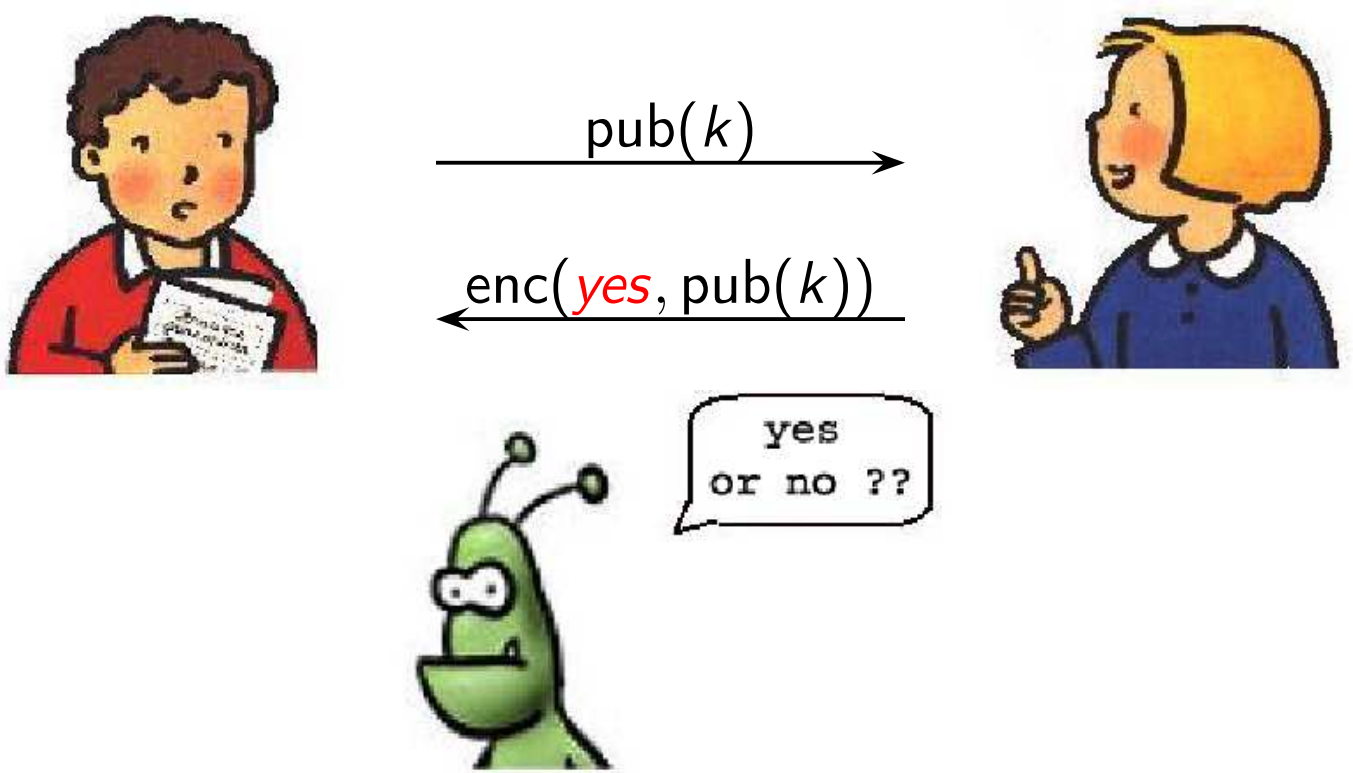
$\longrightarrow$ Typically concludes within few seconds over the flawed protocols of the Clark/Jacob library .

Part II

Designing verification algorithms

(from confidentiality to **privacy**)

Deduction is not always sufficient



→ The intruder knows the values yes and no !

The real question

Is the intruder able to tell whether Alice sends yes or no?

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

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

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

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

Example: Consider the frames:

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

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

Exercise

Consider the equational theories:

- $\mathsf{E}_{\mathsf{senc}}$ defined by $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$, and
- $\mathsf{E}_{\mathsf{cipher}}$ which extends $\mathsf{E}_{\mathsf{senc}}$ by the equation $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent ? Whenever applicable give the distinguishing test.

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\}$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

k, k', and n are a priori unknown to the attacker

Exercise

Consider the equational theories:

- $\mathsf{E}_{\mathsf{senc}}$ defined by $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$, and
- $\mathsf{E}_{\mathsf{cipher}}$ which extends $\mathsf{E}_{\mathsf{senc}}$ by the equation $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent ? Whenever applicable give the distinguishing test.

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

k, k', and n are a priori unknown to the attacker

Exercise

Consider the equational theories:

- $\mathsf{E}_{\mathsf{senc}}$ defined by $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$, and
- $\mathsf{E}_{\mathsf{cipher}}$ which extends $\mathsf{E}_{\mathsf{senc}}$ by the equation $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent ? Whenever applicable give the distinguishing test.

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \quad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \quad \text{✓}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

k, k', and n are a priori unknown to the attacker

Exercise

Consider the equational theories:

- $\mathsf{E}_{\mathsf{senc}}$ defined by $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$, and
- $\mathsf{E}_{\mathsf{cipher}}$ which extends $\mathsf{E}_{\mathsf{senc}}$ by the equation $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent ? Whenever applicable give the distinguishing test.

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \quad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \quad \text{✓}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \quad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

k, k', and n are a priori unknown to the attacker

Exercise

Consider the equational theories:

- $\mathsf{E}_{\mathsf{senc}}$ defined by $\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$, and
- $\mathsf{E}_{\mathsf{cipher}}$ which extends $\mathsf{E}_{\mathsf{senc}}$ by the equation $\mathsf{senc}(\mathsf{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent ? Whenever applicable give the distinguishing test.

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \quad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \quad \text{✓}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \quad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \quad \text{✓}$$

k, k', and n are a priori unknown to the attacker

The static equivalence problem

Proposition

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

The static equivalence problem

Proposition

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

Algorithm

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 our previous example

Example

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

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

Going back to our previous example

Example

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

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

Applying the algorithm

- $\phi^+ = \phi \uplus \{$, and
- $\psi^+ = \psi \uplus \{$.

Going back to our previous example

Example

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

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

Applying the algorithm

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

Going back to our previous example

Example

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

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

Applying the algorithm

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

Going back to our previous example

Example

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

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

Applying the algorithm

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

Going back to our previous example

Example

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

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

Applying the algorithm

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

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

State of the art in a nutshell (active attacker)

for analysing privacy properties

Unbounded number of sessions

- undecidable in general (and even under quite severe restriction)
- decidable for restricted classes [Chrétien PhD thesis, 16]

$\longrightarrow$ ProVerif checks diff-equivalence (too strong) [Blanchet et al, 05]

State of the art in a nutshell (active attacker)

for analysing privacy properties

Unbounded number of sessions

- undecidable in general (and even under quite severe restriction)
- decidable for restricted classes [Chrétien PhD thesis, 16]

$\longrightarrow$ ProVerif checks diff-equivalence (too strong) [Blanchet et al, 05]

Bounded number of sessions

- several decision procedures under various restrictions
  e.g. [Baudet, 05], [Dawson & Tiu, 10], [Chevalier & Rusinowitch, 10], [Chadha et al., 12], **[Cheval PhD thesis, 12]**.

One "recent" contribution

$\longrightarrow$ PhD thesis of V. Cheval, 2012

Main result

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

One "recent" contribution

$\longrightarrow$ PhD thesis of V. Cheval, 2012

Main result

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

Class of processes:

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

Privacy using the constraint solving approach

What about unlinkability of the ePassport holders ?

$$P_{\mathsf{BAC}}(K_E, K_M) \stackrel{?}{\approx} P_{\mathsf{BAC}}(K'_E, K'_M)$$

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

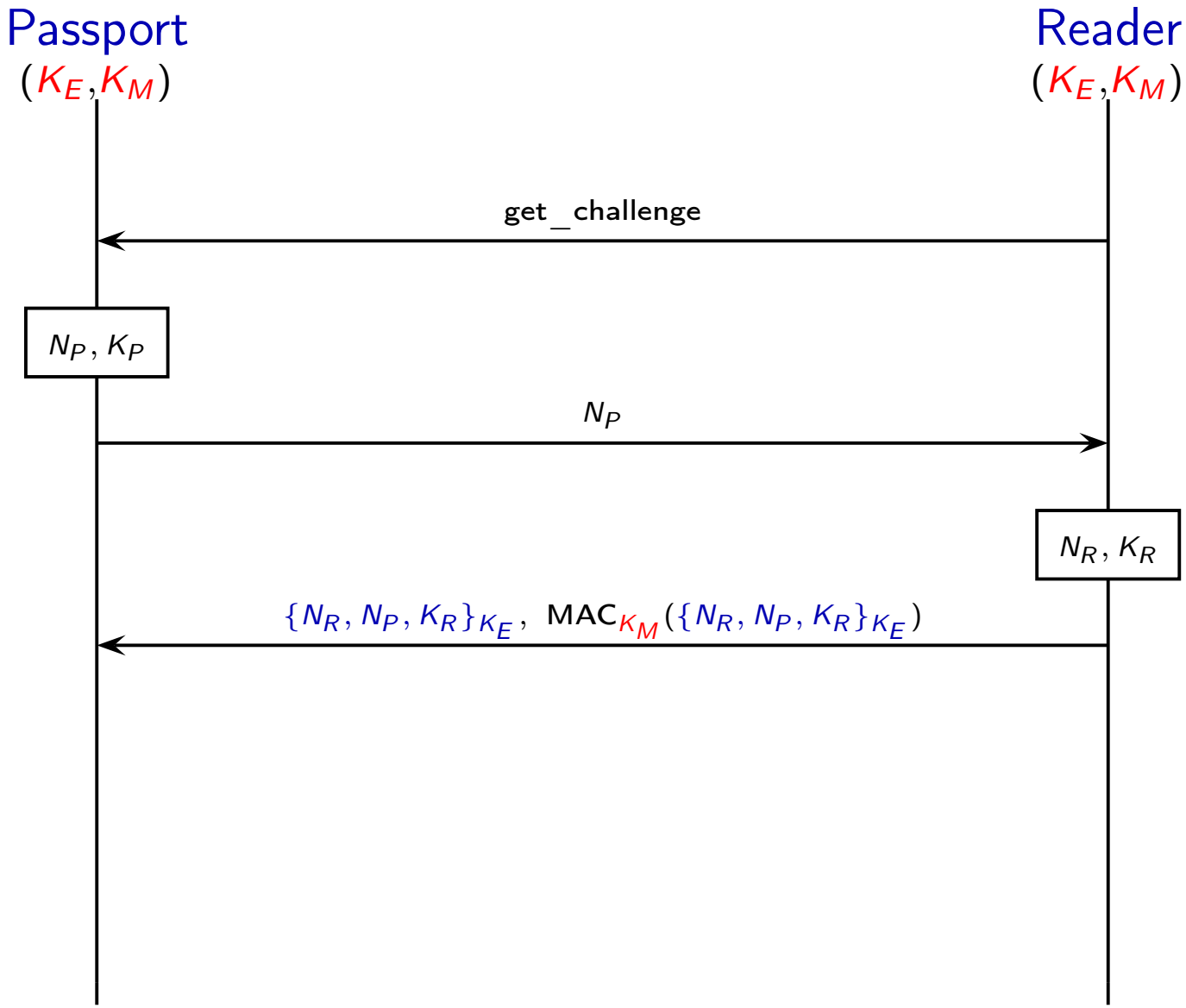
   $\longrightarrow$ this set can be huge (exponential on the number of sessions) !

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

   $\longrightarrow$ this algorithm works quite well

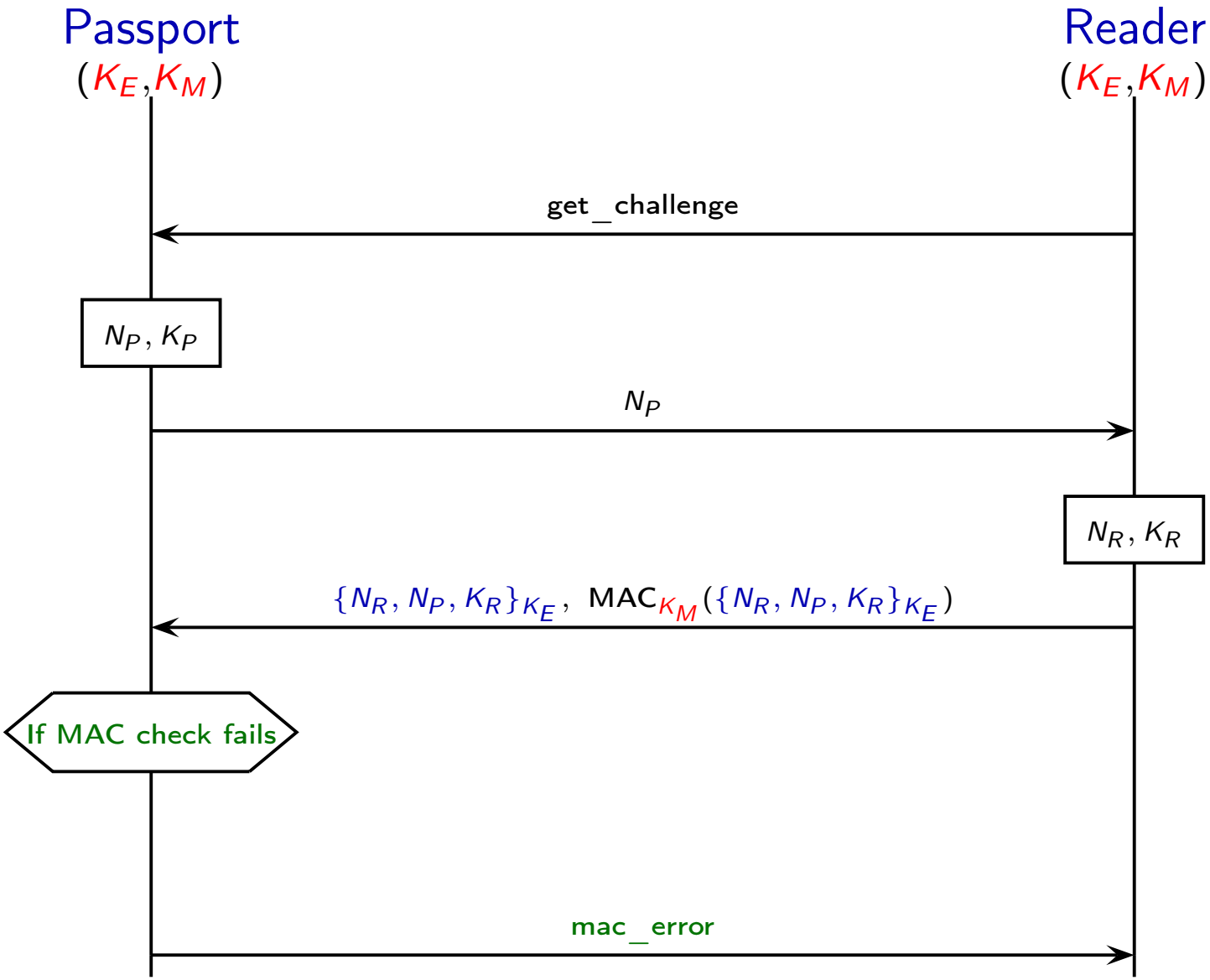
French electronic passport

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



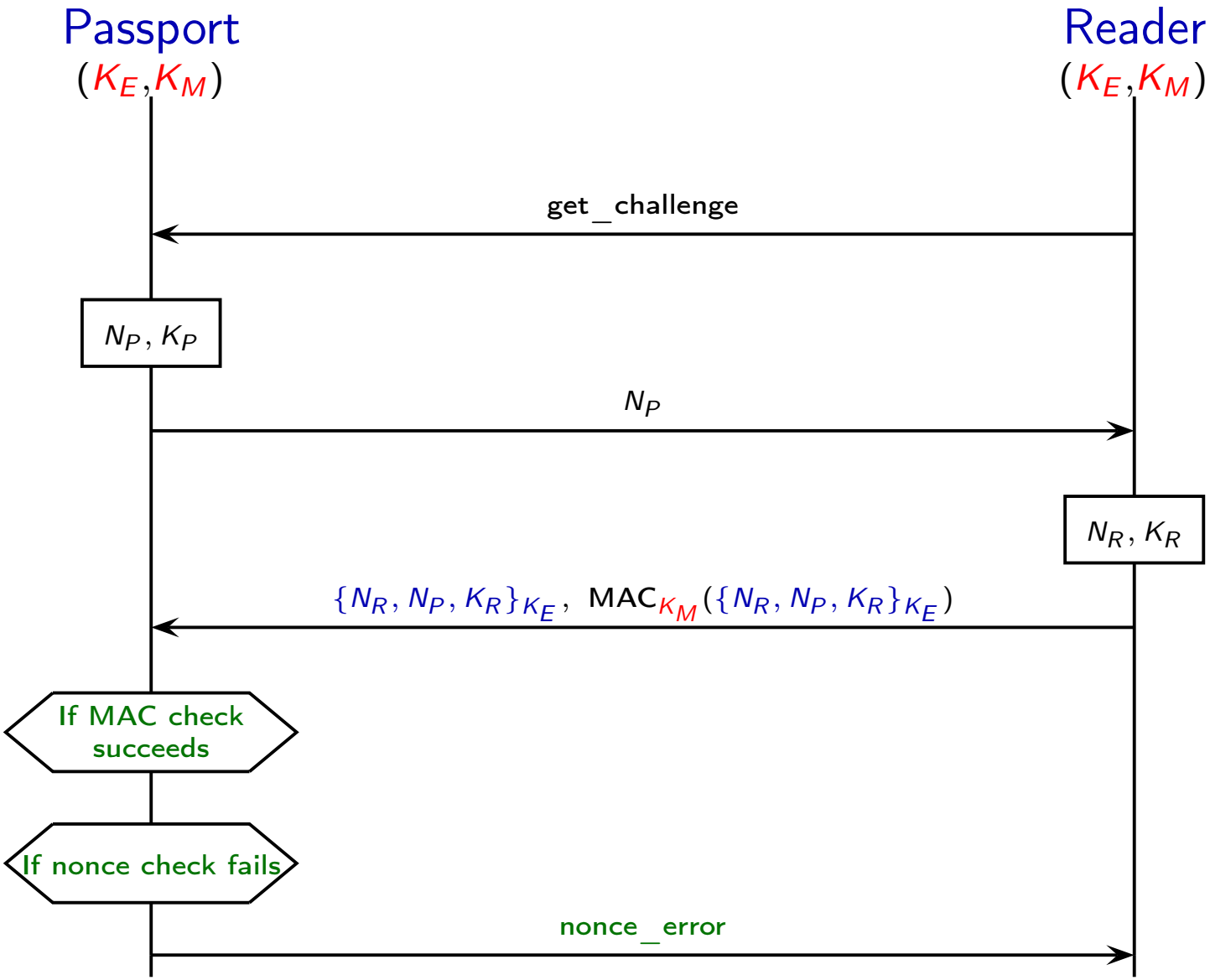
French electronic passport

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



French electronic passport

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



Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0;$$

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0; N_P;$$

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0; N_P;$$

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

$$\frac{T_0 \overset{?}{\vdash} get\_challenge \quad T_0, N_P \overset{?}{\vdash} \langle z_E, z_M \rangle \quad z_M \overset{?}{\neq} \mathsf{mac}_{K_M}(z_E)}{\Phi = T_0; N_P; mac\_error}$$

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

$$\frac{T_0 \overset{?}{\vdash} get\_challenge \quad T_0, N_P \overset{?}{\vdash} \langle z_E, z_M \rangle \quad z_M \overset{?}{\neq} \mathsf{mac}_{K_M}(z_E)}{\Phi = T_0; N_P; mac\_error}$$

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

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0;$$

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0; N_P;$$

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

$$\Phi = T_0; N_P;$$

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
  $\mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
    $\mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
  $\mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

$$\frac{T_0 \overset{?}{\vdash} get\_challenge \quad T_0, N_P \overset{?}{\vdash} \langle z_E, z_M \rangle \quad z_M \overset{?}{=} \mathsf{mac}_{K_M}(z_E) \quad N_P \overset{?}{\neq} \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))}{\Phi = T_0; N_P; nonce\_error}$$

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

$$\frac{T_0 \overset{?}{\vdash} get\_challenge \quad T_0, N_P \overset{?}{\vdash} \langle z_E, z_M \rangle \quad z_M \overset{?}{=} \mathsf{mac}_{K_M}(z_E) \quad N_P \overset{?}{\neq} \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))}{\Phi = T_0; N_P; nonce\_error}$$

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

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

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

Step 1: from processes to constraint systems

Passport $P(K_E, K_M)$

$\mathsf{in}(= get\_challenge); \mathsf{new}\ N_P; \mathsf{new}\ K_P;$
$\mathsf{out}(N_P); \mathsf{in}(\langle z_E, z_M \rangle);$
$\mathtt{if}\ z_M = \mathsf{mac}_{K_M}(z_E)\ \mathtt{then}$
$\quad \mathtt{if}\ N_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E)))\ \mathtt{then}$
$\qquad \mathsf{out}(\langle m, \mathsf{mac}_{K_M}(m) \rangle)$
$\quad \mathtt{else}\ \mathsf{out}(nonce\_error)$
$\mathtt{else}\ \mathsf{out}(mac\_error)$

where
$m = \{\langle N_P, \mathsf{proj}_1(\mathsf{sdec}(z_E, K_E)), K_P \rangle\}_{K_E}$

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

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

Step 2: symbolic equivalence

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

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

Step 2: symbolic equivalence

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

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

Symbolic equivalence $\Sigma \approx_s \Sigma'$

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

Step 2: symbolic equivalence

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

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

Symbolic equivalence $\Sigma \approx_s \Sigma'$

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

Going back to the E-passport example

$$P_{\mathsf{BAC}}(K_E, K_M) \stackrel{?}{\approx} P_{\mathsf{BAC}}(K'_E.K'_M)$$

Among others, we have to check: $\{\mathcal{C}_{\mathsf{mac}};\ \mathcal{C}_{\mathsf{nonce}}; \ldots\} \stackrel{?}{\approx}_s \{\mathcal{C}'_{\mathsf{mac}}; \mathcal{C}'_{\mathsf{nonce}}; \ldots\}$ where $\mathcal{C}'_{\mathsf{mac}}$, $\mathcal{C}'_{\mathsf{nonce}}$, ... are the counterparts of $\mathcal{C}_{\mathsf{mac}}$, $\mathcal{C}_{\mathsf{nonce}}$, ... in which K_E/K_M are replaced by K'_E/K'_M.

French passport (1/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

French passport (1/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

$$\mathcal{C}_{\mathsf{nonce}} = \begin{cases} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0 \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{=} \mathsf{mac}_{K_M}(z_E) \\ N_P \overset{?}{\neq} \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K_E))) \\ \hline \Phi = T_0; \mathit{nonce\_error} \end{cases}$$

$\longrightarrow$ A solution for $\mathcal{C}_{\mathsf{nonce}}$ is:

$$\sigma = \left\{z_E \mapsto \{N_R^0, N_P^0, K_R^0\}_{K_E},\ z_M \mapsto \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\right\}$$

$$\text{with } \theta = \left\{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\right\}.$$

French passport (2/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

Is $\theta = \{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\}$ also a solution on the other side?

French passport (2/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

Is $\theta = \{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\}$ also a solution on the other side?

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

$$\mathcal{C}'_{\mathsf{nonce}} = \left\{ \begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0, N_P \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{=} \mathsf{mac}_{K'_M}(z_E) \\ N_P \overset{?}{\neq} \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K'_E))) \\ \hline \Phi = T_0; N_P; \mathit{nonce\_error} \end{array} \right.$$

French passport (2/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

Is $\theta = \{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\}$ also a solution on the other side?

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

$$\mathcal{C}'_{\mathsf{nonce}} = \left\{ \begin{array}{c} T_0 \overset{?}{\vdash} \mathit{get\_challenge} \\ T_0, N_P \overset{?}{\vdash} \langle z_E, z_M\rangle \\ z_M \overset{?}{=} \mathsf{mac}_{K'_M}(z_E) \\ N_P \overset{?}{\neq} \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, K'_E))) \\ \hline \Phi = T_0; N_P; \mathit{nonce\_error} \end{array} \right.$$

$\longrightarrow$ θ is not a solution !

French passport (2/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

Is $\theta = \{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\}$ also a solution on the other side?

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

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

French passport (2/2)

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

when T_0 contains $w_0 \triangleright \langle \{N_R^0, N_P^0, K_R^0\}_{K_E},\ \mathsf{mac}_{K_M}(\{N_R^0, N_P^0, K_R^0\}_{K_E})\rangle$

Is $\theta = \{X_1 \mapsto \mathit{get\_challenge},\ X_2 \mapsto w_0\}$ also a solution on the other side?

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

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

$\longrightarrow$ θ is a solution ...

but the resulting sequence of messages are not in static equivalence.

$$T_0;\ N_P;\ \mathit{nonce\_error} \not\sim T_0;\ N_P,\ \mathit{mac\_error}$$

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

Attack against unlinkability

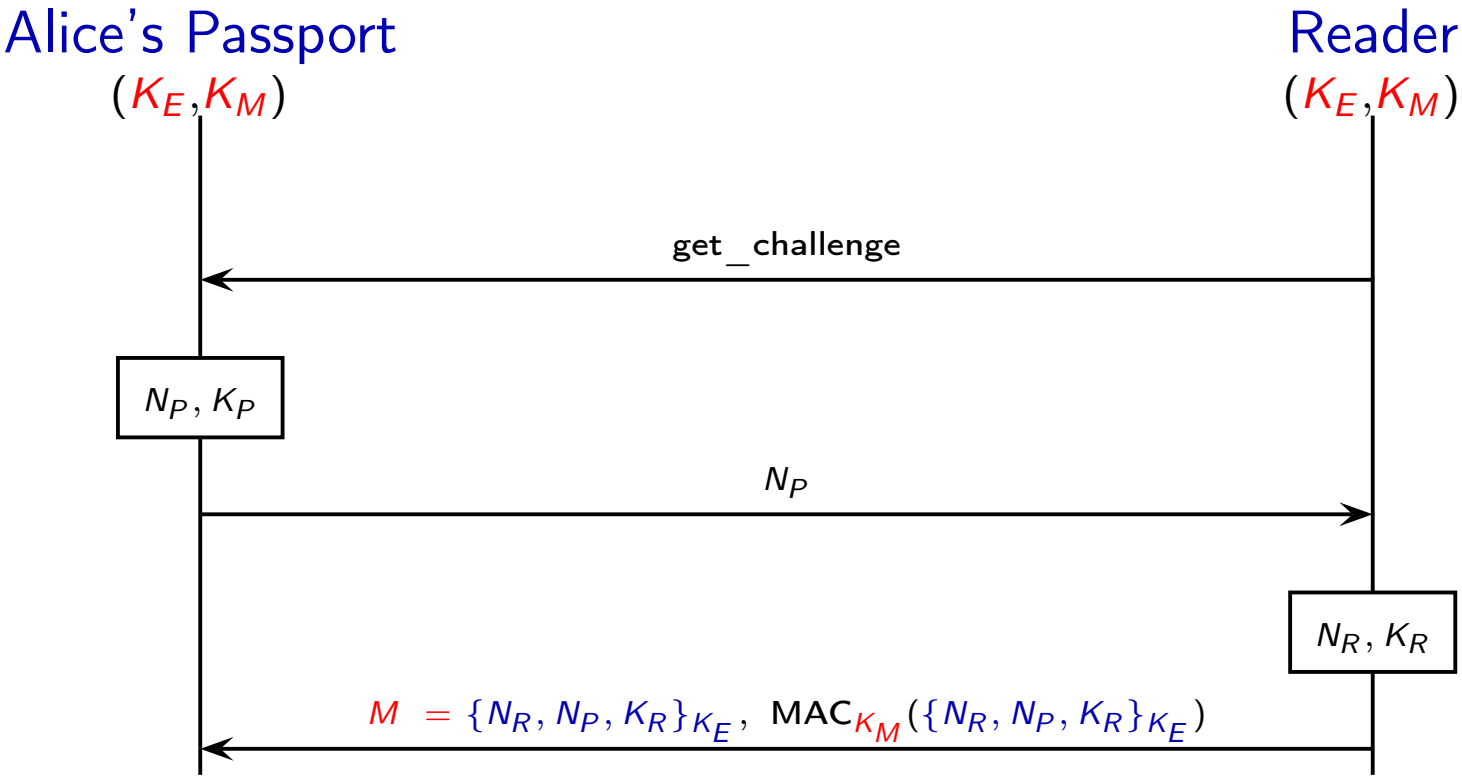
An attacker can track a French passport, provided he has once witnessed a successful authentication.

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

Attack against unlinkability

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

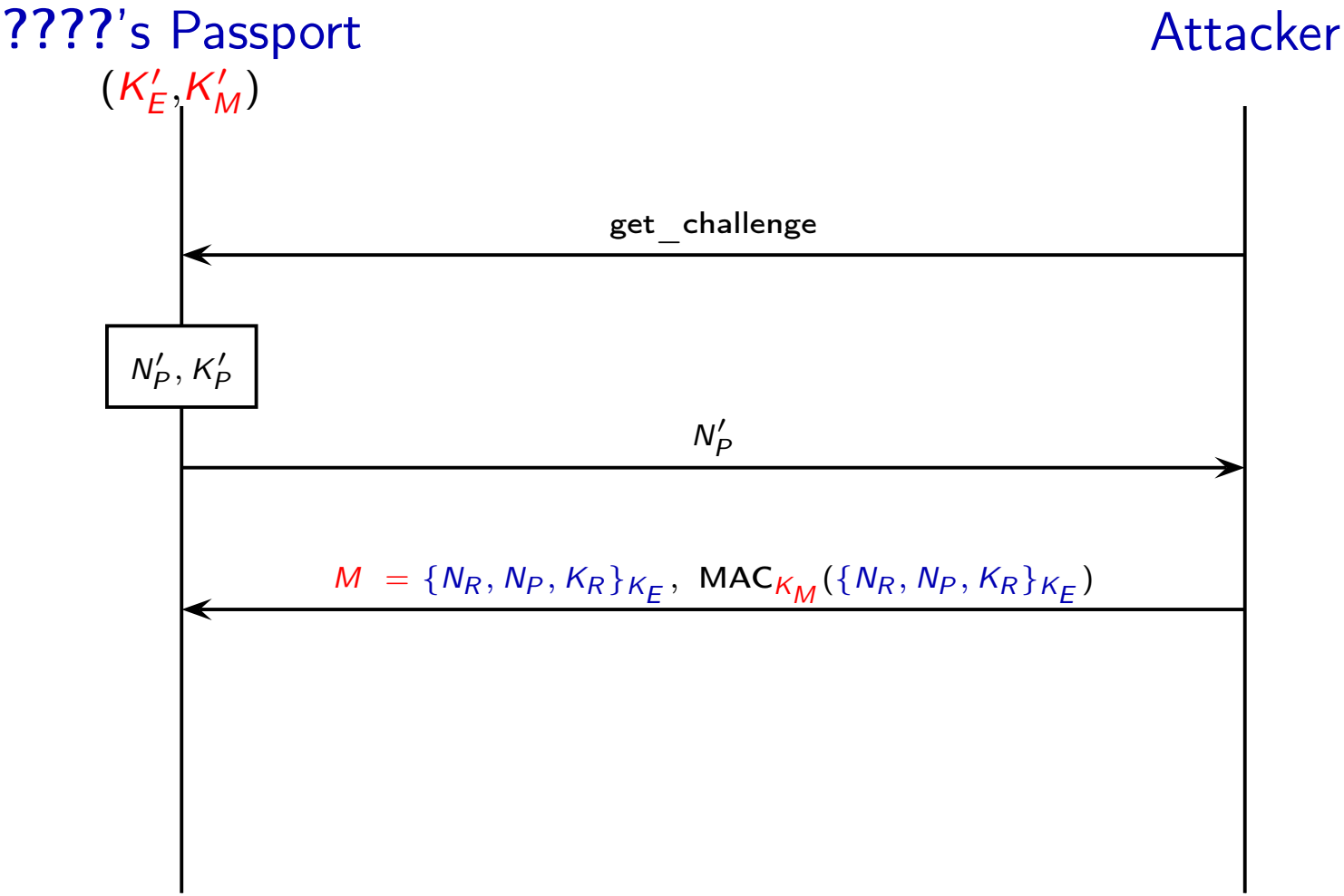
Part 1 of the attack. The attacker eavesdropes on Alice using her passport and records message M.



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

Part 2 of the attack.

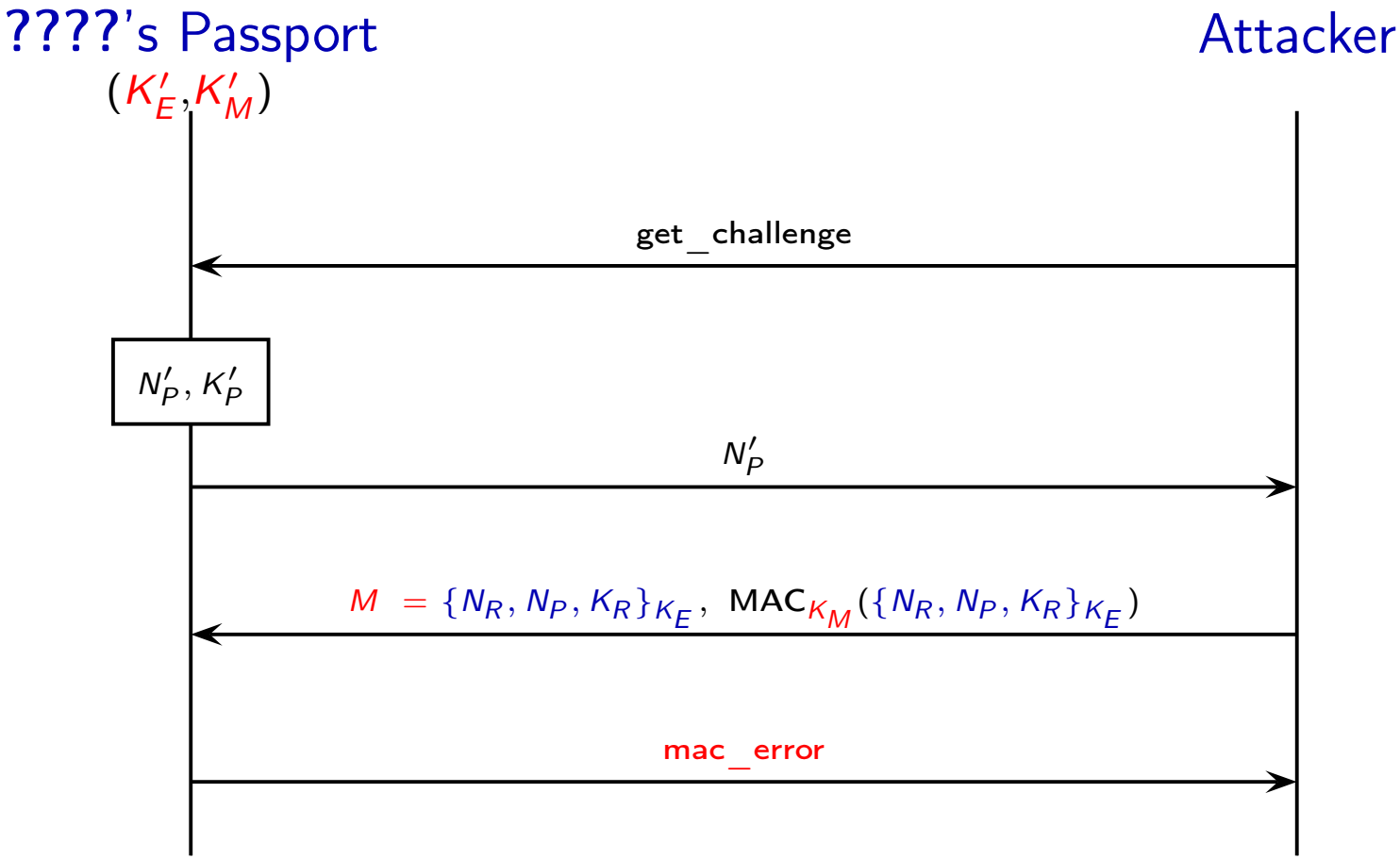
The attacker replays the message M and checks the error code he receives.



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

Part 2 of the attack.

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

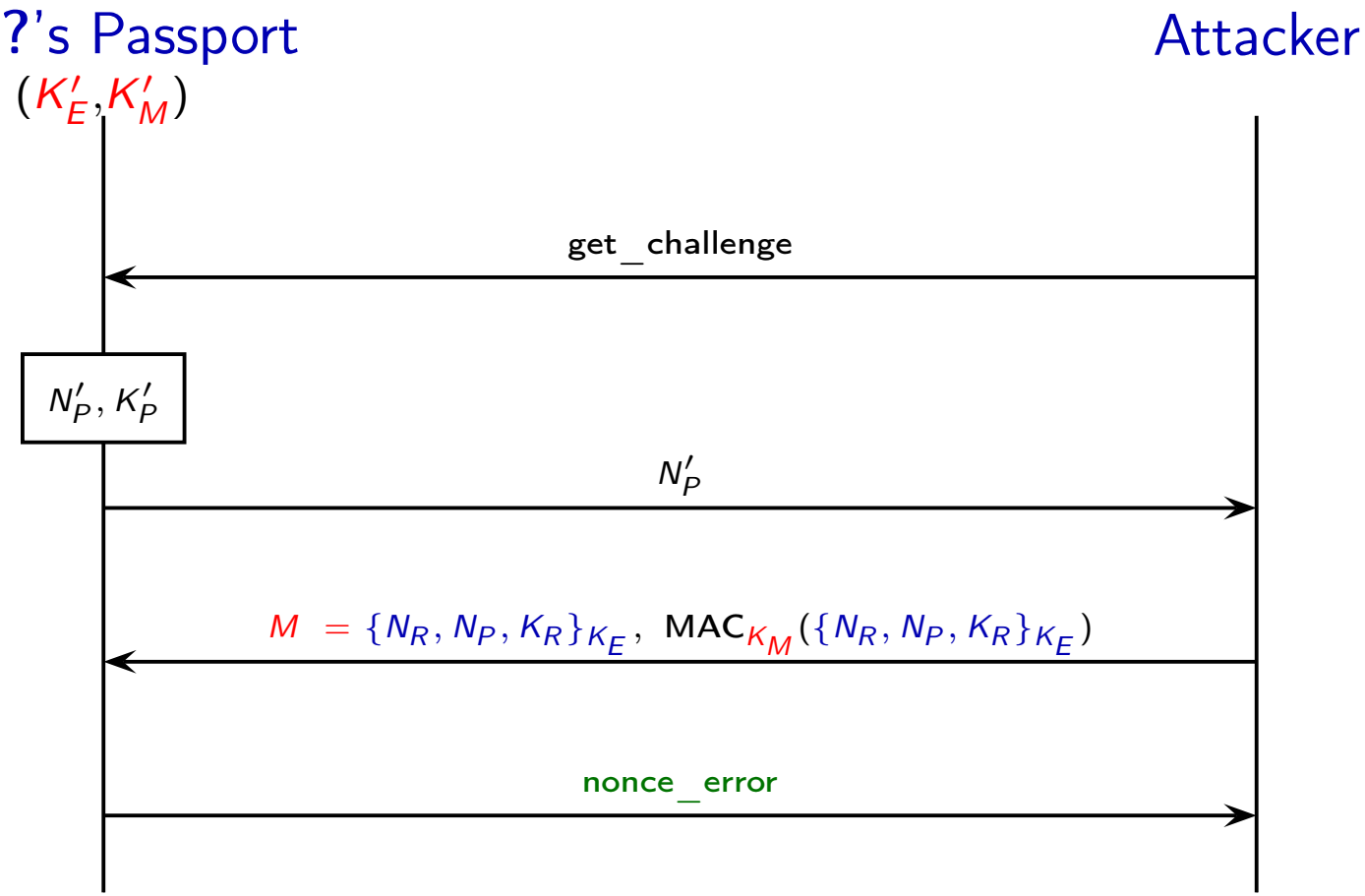


$\Longrightarrow$ MAC check failed $\Longrightarrow$ $K'_M \neq K_M$ $\Longrightarrow$???? is not Alice

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

Part 2 of the attack.

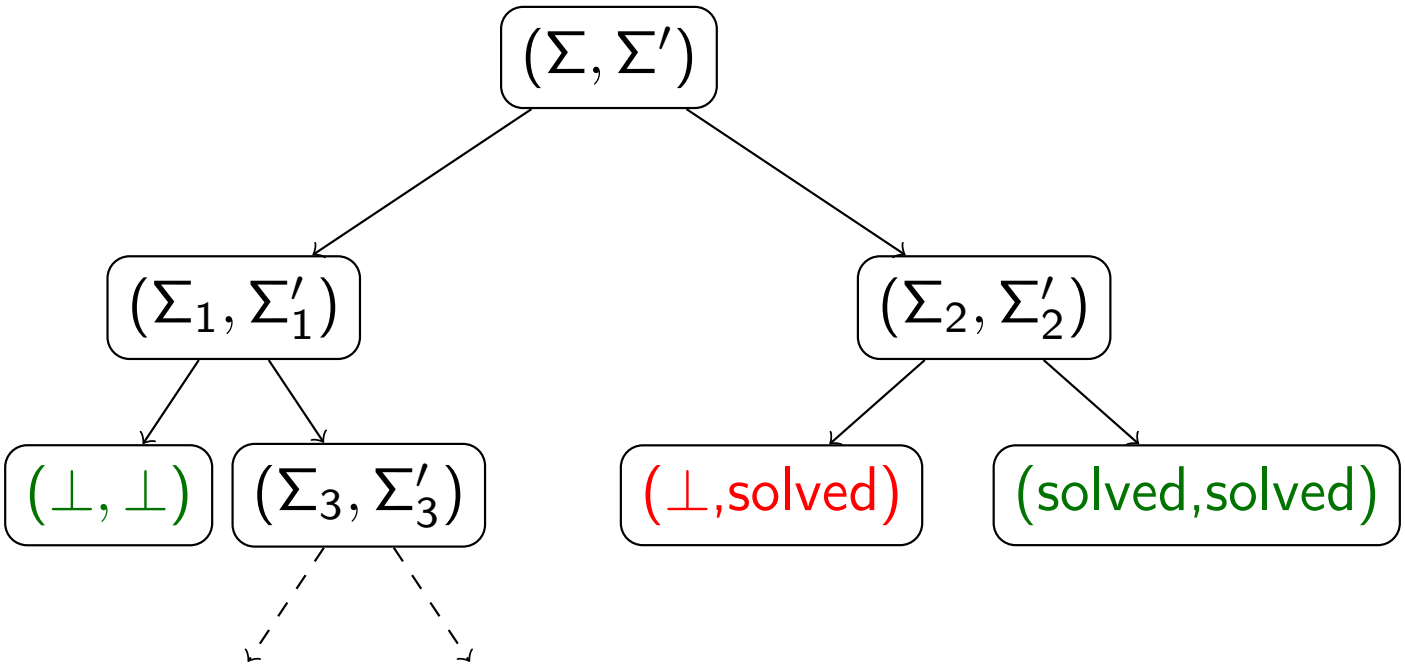
The attacker replays the message M and checks the error code he receives.



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

Step 2: 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

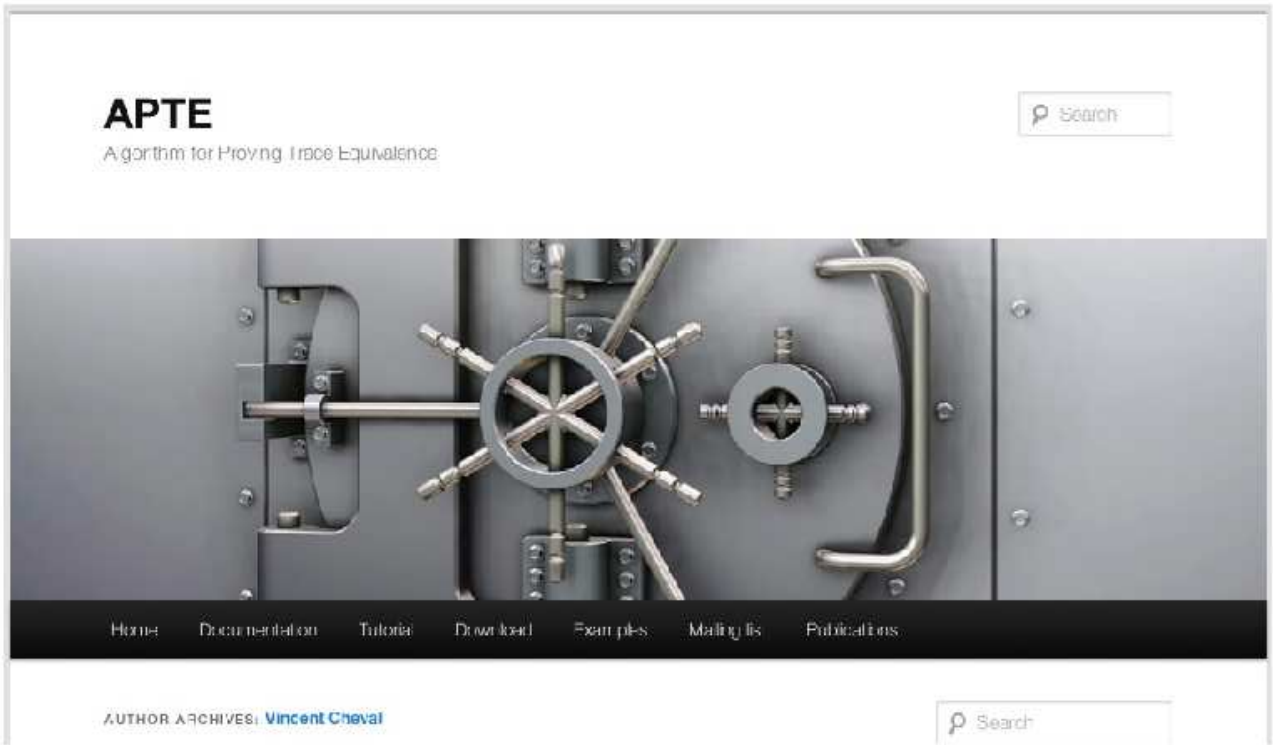
Let (Σ_0, Σ_0') be pair of sets of constraint systems, and consider a binary tree obtained by applying our simplification rule following a strategy $\mathcal{S}$.

1. soundness: If all leaves of the tree are labeled with $(\bot, \bot)$ 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 $(\bot, \bot)$ or $(\mathit{solved}, \mathit{solved})$.

APTE- Algorithm for Proving Trace Equivalence

`http://projects.lsv.ens-cachan.fr/APTE` (Ocaml - 12 KLocs)

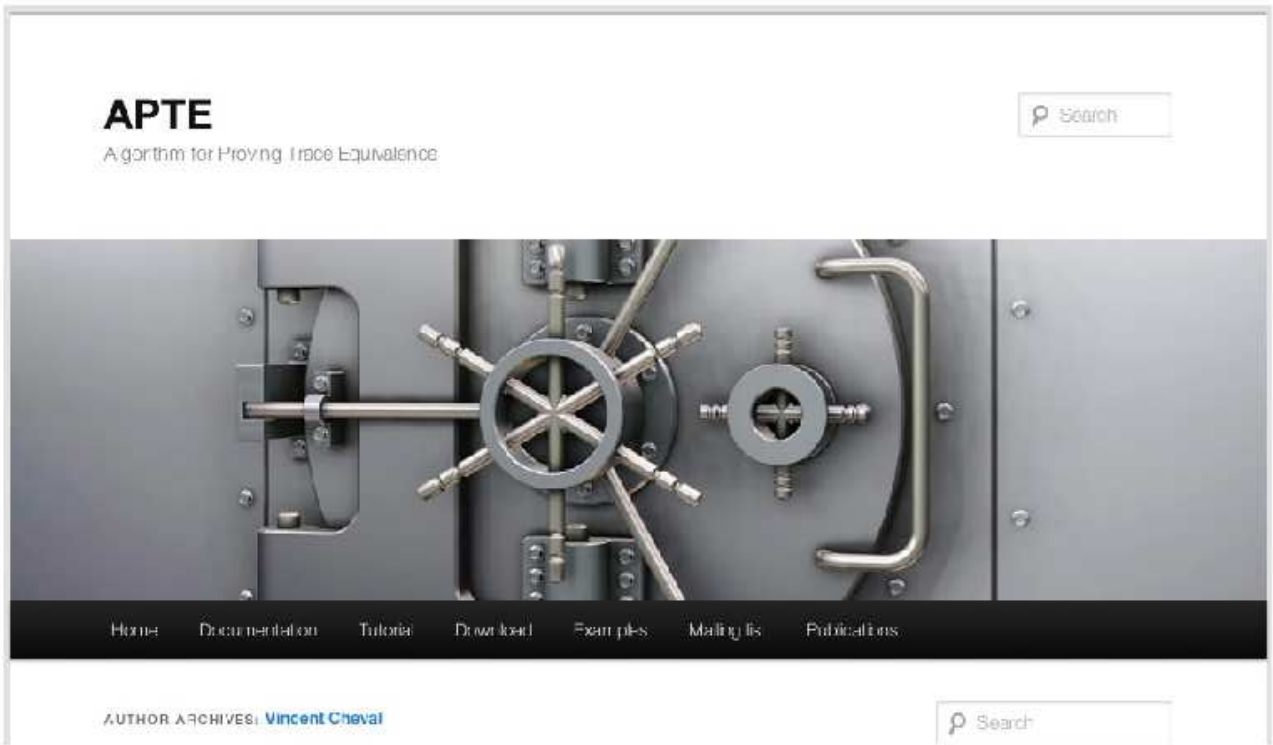
$\longrightarrow$ developed by Vincent Cheval [Cheval, TACAS'14]



APTE- Algorithm for Proving Trace Equivalence

`http://projects.lsv.ens-cachan.fr/APTE` (Ocaml - 12 KLocs)

$\longrightarrow$ developed by Vincent Cheval [Cheval, TACAS'14]



$\longrightarrow$ but a limited practical impact because it scales badly

Partial order reduction for security protocols

part of the PhD thesis of L. Hirschi

Main objective

to develop POR techniques that are suitable for analysing security protocols (especially testing equivalence)

Partial order reduction for security protocols

part of the PhD thesis of L. Hirschi

Main objective

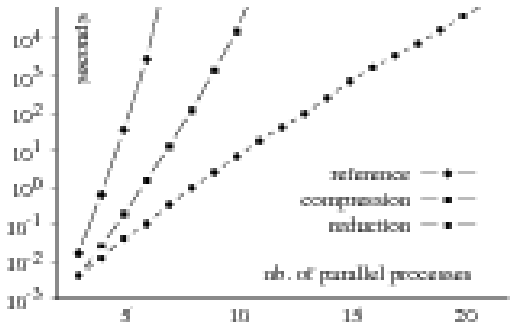
to develop POR techniques that are suitable for analysing security protocols (especially testing equivalence)

Example: $\mathsf{in}(c_1, x_1).\mathsf{out}(c_1, \mathsf{ok}) \mid \mathsf{in}(c_2, x_2).\mathsf{out}(c_2, \mathsf{ok})$

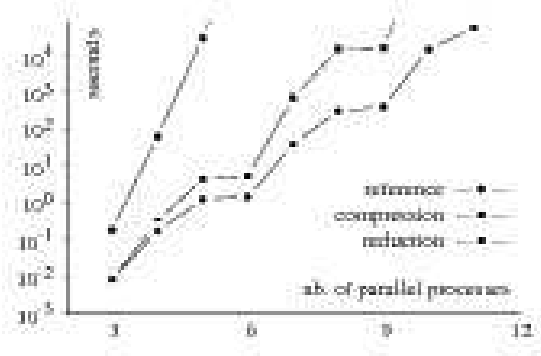
We propose two optimizations:

1. compression: we impose a simple strategy on the exploration of the available actions (roughly outputs are performed first and using a fixed arbitrary order)
2. reduction: we avoid exploring some redundant traces taking into account the data that are exchanged

Practical impact of our optimizations (in APTE)



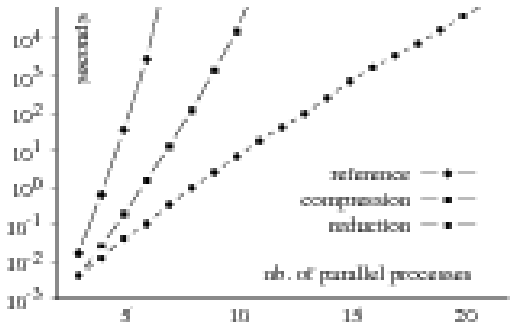
Toy example



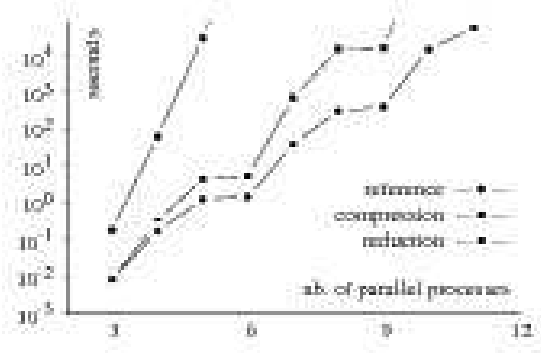
Denning Sacco protocol

$\longrightarrow$ Each optimisation brings an exponential speedup.

Practical impact of our optimizations (in APTE)



Toy example



Denning Sacco protocol

$\longrightarrow$ Each optimisation brings an exponential speedup.

Protocol	reference	with POR
Yahalom (3-party)	4	5
Needham Schroeder (3-party)	4	7
Private Authentication (2-party)	4	7
E-Passport PA (2-party)	4	9
Denning-Sacco (3-party)	5	10
Wide Mouthed Frog (3-party)	6	13

Maximum number of parallel processes verifiable in 20 hours.

$\longrightarrow$ Our optimisations make Apte much more useful in practice for investigating interesting scenarios.

Limitations of this approach

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/g̃londu/epassport/attaque-tailles.html

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

To sum up

Cryptographic protocols are:

- difficult to design and analyse;
- particularly vulnerable to logical attacks.

Strong primitives are necessary . . .

. . . **but this is not sufficient !**

To sum up

Cryptographic protocols are:

- difficult to design and analyse;
- particularly vulnerable to logical attacks.

It is important to ensure that
the protocols we are using every day work properly.

We now have automatic and powerful verification tools to analyse:

- classical security goals, *e.g.* secrecy and authentication;
- relatively small protocols;
- protocols that rely on standard cryptographic primitives.

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
  $\longrightarrow$ receipt-freeness, coercion-resistance in e-voting
- algorithms (and tools!) for checking automatically trace equivalence for various cryptographic primitives;
  $\longrightarrow$ homomorphic encryption used in e-voting, exclusive-or used in RFID protocols
- more composition results
  $\longrightarrow$ Could we derive some security guarantees of the whole e-passport application from the analysis performed on each subprotocol?