Analysing privacy-type properties in cryptographic protocols

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Friday, March 20th, 2015

Cryptographic protocols everywhere !

Cryptographic protocols

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

The network is unsecure!

Communications take place over a public network like the Internet.

Cryptographic protocols everywhere !

Cryptographic protocols

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

It becomes more and more important to protect our privacy.

Electronic passport

⟶ studied in [Arapinis *et al.*, 10]

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

The RFID tag stores:

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

Electronic passport

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

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

The RFID tag stores:

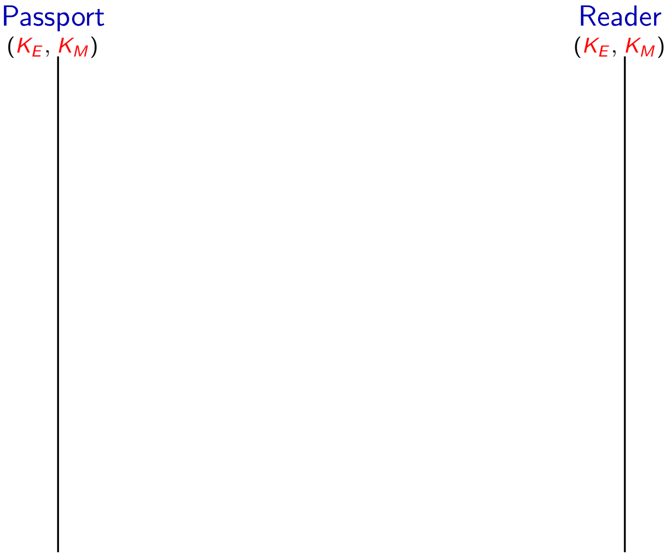
- the information printed on your passport,
- a JPEG copy of your picture.

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

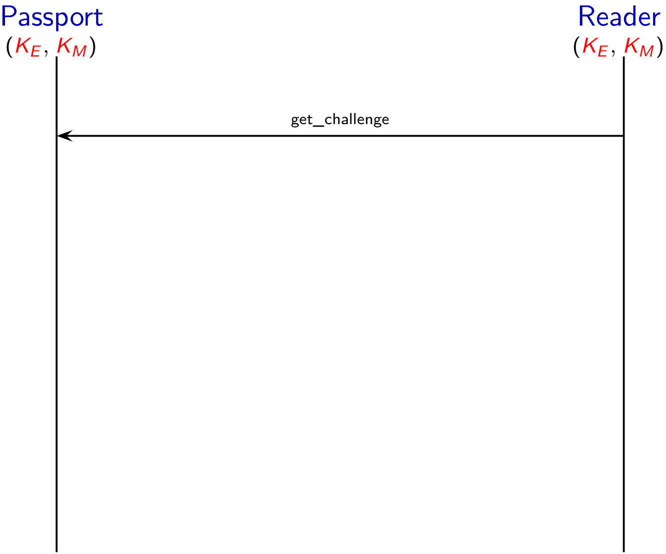
ISO/IEC standard 15408

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.*

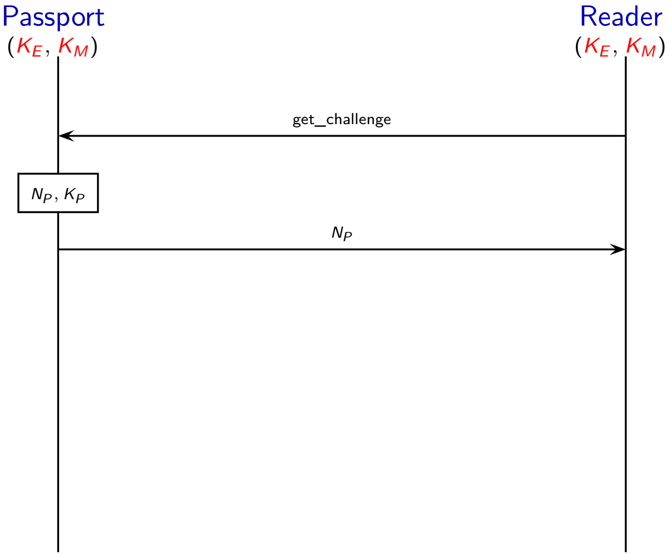
Basic Acccess Control (BAC) protocol



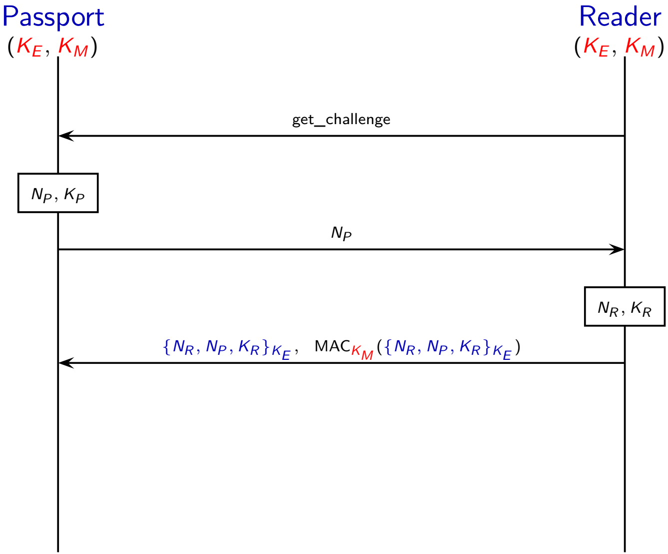
Basic Acccess Control (BAC) protocol



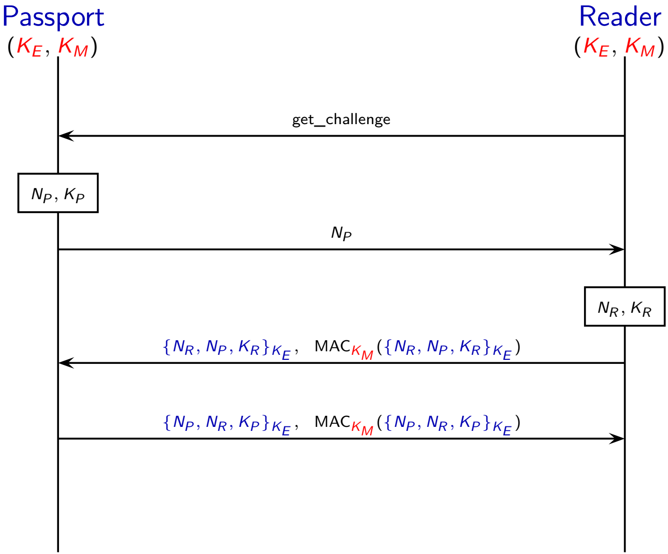
Basic Acccess Control (BAC) protocol



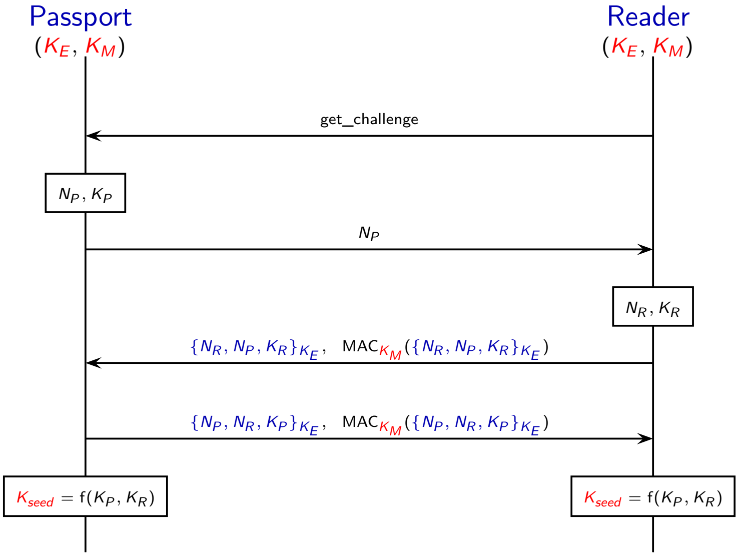
Basic Acccess Control (BAC) protocol



Basic Acccess Control (BAC) protocol



Basic Acccess Control (BAC) protocol



What does unlinkability mean?

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

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

What does unlinkability mean?

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

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

More formally,

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

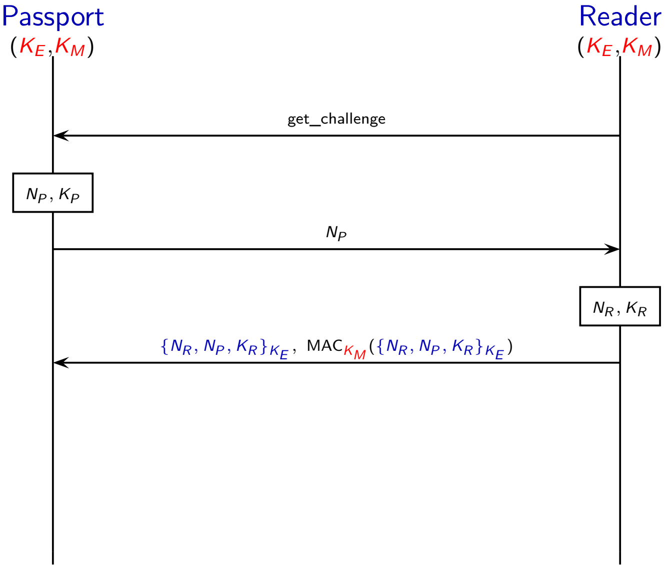
↑ many sessions for each passport

↑ only one session for each passport

(we still have to formalize the processes and the notion of equivalence)

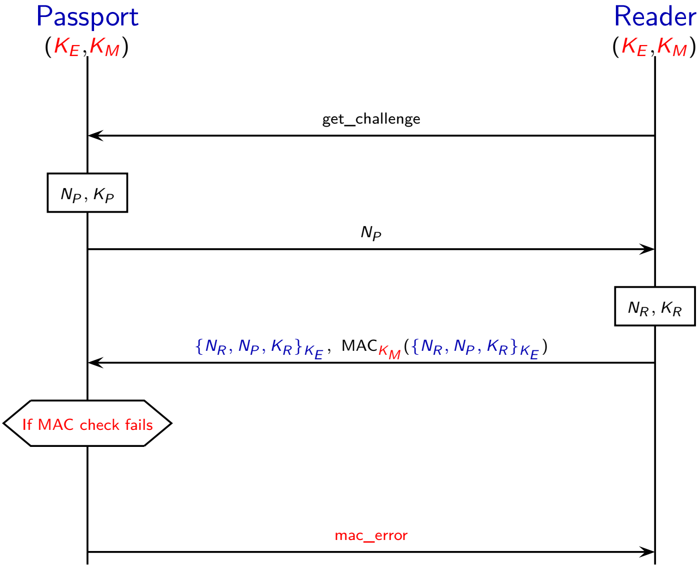
French electronic passport

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



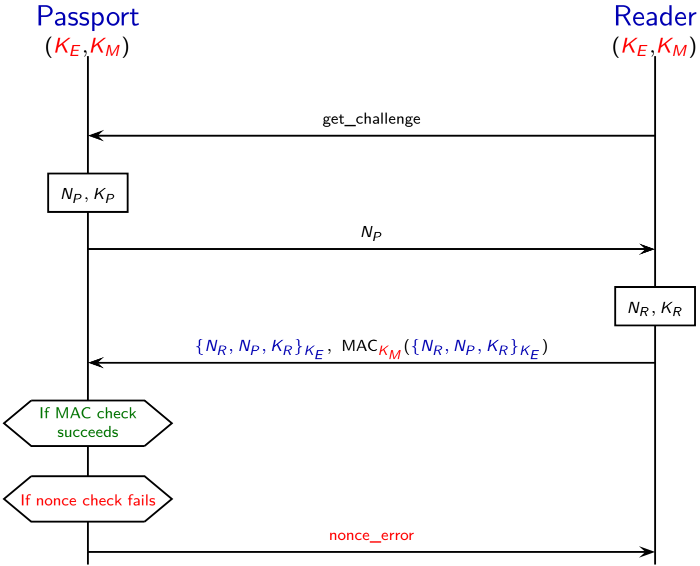
French electronic passport

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



French electronic passport

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



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

Attack against unlinkability

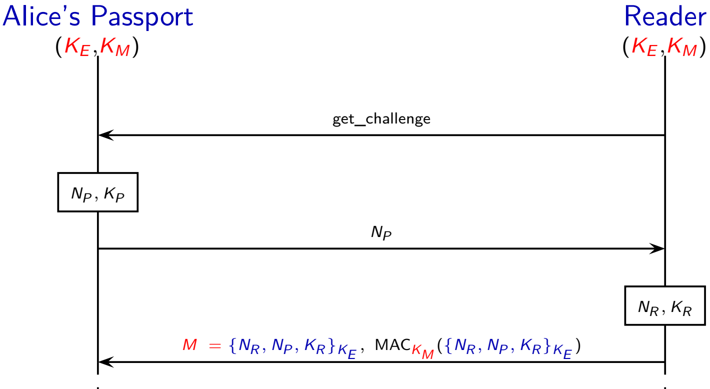
An attacker can track a French passport, provided he has once witnessed a successful authentication.

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.

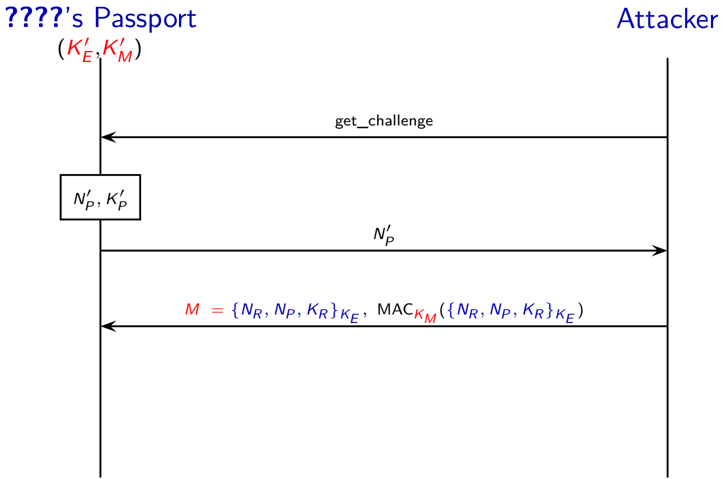
Part 1 of the attack. The attacker eavesdropes on Alice using her passport and records message M.



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

Part 2 of the attack.

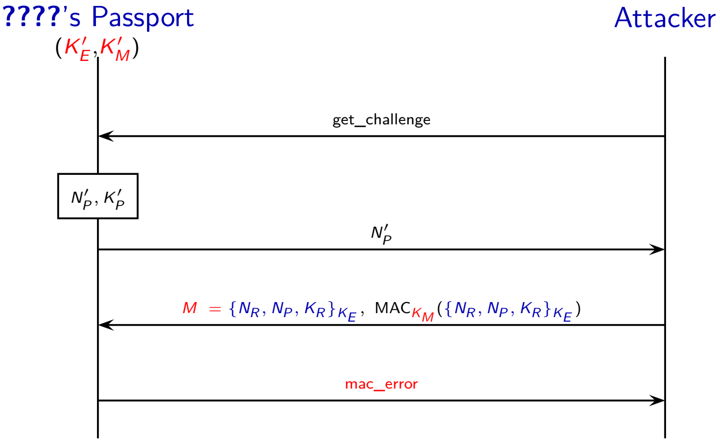
The attacker replays the message M and checks the error code he receives.



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.

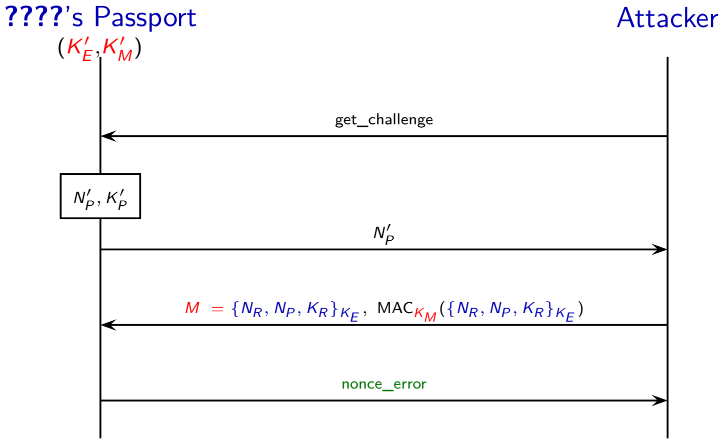


$\Longrightarrow$ MAC check failed $\Longrightarrow$ $K'_M \neq K_M$ $\Longrightarrow$ **????** is not Alice

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.

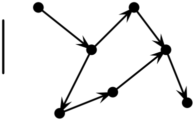


$\Longrightarrow$ MAC check succeeded $\Longrightarrow$ $K'_M = K_M$ $\Longrightarrow$ **????** is Alice

Outline

Does the protocol *satisfy* a security property?

Modelling



$\models \varphi$

Outline of the remaining of this talk

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

$\longrightarrow$ we focus here on privacy-type security properties

Modelling cryptographic protocols and their security properties

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

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

Messages as terms

Messages are abstracted by (ground) terms

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

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

Messages as terms

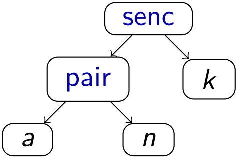
Messages are abstracted by (ground) terms

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

t	::=	n	name n
	|	$f(t_1, \ldots, t_k)$	application of symbol $f \in \mathcal{F}$

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

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



Messages as terms

Messages are abstracted by (ground) terms

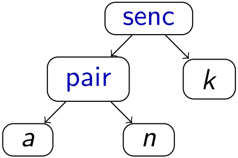
Ground terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

t	::=	n	name n
	|	$f(t_1, \ldots, t_k)$	application of symbol $f \in \mathcal{F}$

$\longrightarrow$ The term algebra is equipped with an equational theory E.

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

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



$\longrightarrow$ $\mathsf{sdec}(\mathsf{senc}(x,y),y) = x, \quad \mathsf{proj}_1(\mathsf{pair}(x,y)) = x, \quad \mathsf{proj}_2(\mathsf{pair}(x,y)) = y.$

Going back to the e-passport

Cryptographic primitives are modelled using function symbols

- encryption/decryption: senc/2, sdec/2
- concatenation/projections: $\langle\ ,\ \rangle/2$, $\mathsf{proj}_1/1$, $\mathsf{proj}_2/1$
- mac construction: mac/2

$$\longrightarrow \quad \mathsf{sdec}(\mathsf{senc}(x,y),y) = x, \quad \mathsf{proj}_1(\langle x,y\rangle) = x, \quad \mathsf{proj}_2(\langle x,y\rangle) = y.$$

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

Going back to the e-passport

Cryptographic primitives are modelled using function symbols

- encryption/decryption: senc/2, sdec/2
- concatenation/projections: $\langle\ ,\ \rangle/2$, $\mathsf{proj}_1/1$, $\mathsf{proj}_2/1$
- mac construction: mac/2

$\longrightarrow \quad \mathsf{sdec}(\mathsf{senc}(x,y),y) = x, \quad \mathsf{proj}_1(\langle x,y\rangle) = x, \quad \mathsf{proj}_2(\langle x,y\rangle) = y.$

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

Modelling Passport's role

$$
\begin{array}{l}
P_{\mathrm{BAC}}(k_E, k_M) \;=\; \mathsf{new}\ n_P.\mathsf{new}\ k_P.\mathsf{out}(n_P).\mathsf{in}(\langle z_E, z_M\rangle). \\
\quad \mathtt{if}\ z_M = \mathsf{mac}(z_E, k_M)\ \mathtt{then}\ \mathtt{if}\ n_P = \mathsf{proj}_1(\mathsf{proj}_2(\mathsf{sdec}(z_E, k_E))) \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathtt{then}\ \mathsf{out}(\langle m, \mathsf{mac}(m, k_M)\rangle) \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathtt{else}\ \mathsf{out}(nonce\_error) \\
\qquad\qquad\qquad\qquad \mathtt{else}\ \mathsf{out}(mac\_error)
\end{array}
$$

where $m = \mathsf{senc}(\langle n_P, \langle \mathsf{proj}_1(z_E), k_P\rangle\rangle, k_E)$.

Semantics

Semantics $\to$:

COMM	$\mathsf{out}(c, u).P \mid \mathsf{in}(c, x).Q \to P \mid Q\{u/x\}$
THEN	if $u = v$ then P else $Q \to P$ when $u =_{\mathsf{E}} v$
ELSE	if $u = v$ then P else $Q \to Q$ 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	if $u = v$ then P else $Q \rightarrow P$ when $u =_{\mathsf{E}} v$
ELSE	if $u = v$ then P 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:

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

Example 1: $\mathsf{out}(a, s) \overset{?}{\approx}_t \mathsf{out}(a, s')$

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

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

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

Example 2:

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

Example 2:

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

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

Security properties - privacy

Privacy-type properties are modelled as equivalence-based properties

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

for all processes A, we have that:

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

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

Question: Are the two following processes in testing equivalence?

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

Some privacy-type properties

Unlinkability [Arapinis et al, 2010]

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

↑ many sessions for each passport

↑ only one session for each passport

Some privacy-type properties

Unlinkability [Arapinis et al, 2010]

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

↑ many sessions for each passport

↑ only one session for each passport

Vote privacy [Kremer and Ryan, 2005]

$$V_A(yes) \approx_t V_A(no)$$

Some privacy-type properties

Unlinkability [Arapinis et al, 2010]

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

↑ many sessions for each passport

↑ only one session for each passport

Vote privacy [Kremer and Ryan, 2005]

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

↑ A votes yes, B votes no

↑ A votes no, B votes yes

Designing verification algorithms for privacy-type properties

How can we check testing equivalence?

testing equivalence is undecidable in general

How can we check testing equivalence?

testing equivalence is undecidable in general

Some decidability results [Chrétien, Cortier & D., ICALP'13 & CONCUR'14]

- - restricted set of cryptographic primitives
- - some syntaxic restrictions on the shape of the processes

How can we check testing equivalence?

testing equivalence is undecidable in general

Some decidability results [Chrétien, Cortier & D., ICALP'13 & CONCUR'14]

- - restricted set of cryptographic primitives
- - some syntaxic restrictions on the shape of the processes

A more pragmatic approach [Blanchet *et al.*, 2005]

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

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

How can we check testing equivalence?

testing equivalence is undecidable in general

Some decidability results [Chrétien, Cortier & D., ICALP'13 & CONCUR'14]

- \- restricted set of cryptographic primitives
- \- some syntaxic restrictions on the shape of the processes

A more pragmatic approach [Blanchet *et al.*, 2005]

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

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

⟶ These results are not suitable to analyse vote-privacy, or unlinkability of the BAC protocol.

Testing equivalence (for processes without replication)

For processes without replication
testing equivalence is decidable
(under some extra assumptions)

Testing equivalence (for processes without replication)

For processes without replication
testing equivalence is decidable
(under some extra assumptions)

Some difficulties

1. We have to consider an infinite number of possible behaviours for the attacker (for all quantification over processes).
2. Once the behavior of the attacker is fixed, we have to decide whether the two sequences of messages that are outputted are indistinguishable or not.

A recent result

Cheval, Comon-Lundh & D. — **CCS 2011**

A procedure for deciding testing equivalence for a large class of processes implemented in a tool called APTE

Our class of processes:

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

A recent result

Cheval, Comon-Lundh & D. — **CCS 2011**

A procedure for deciding testing equivalence for a large class of processes implemented in a tool called APTE

Our class of processes:

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

Similar results for restricted class of processes have been obtained in [Baudet, 05], [Dawson & Tiu, 10], [Chevalier & Rusinowitch, 10], [Chadha *et al.*, 12], . . .

Our procedure in a nutshell

Two main steps:

1. A symbolic exploration of all the possible traces

   The infinite number of possible concrete traces are represented by a finite set of constraint systems.

   $\longrightarrow$ this set is huge (exponential) !

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

   $\longrightarrow$ this algorithm works quite well

Our procedure in a nutshell

Two main steps:

1. A symbolic exploration of all the possible traces

   The infinite number of possible concrete traces are represented by a finite set of constraint systems.

   $\longrightarrow$ this set is huge (exponential) !

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

   $\longrightarrow$ this algorithm works quite well

Some applications

- unlinkability in RFID protocols (*e.g.* e-passport protocol)
- anonymity (*e.g.* private authentication protocol)

Symbolic representation using constraints

Constraint systems are used to represent all the possible behaviours of the attacker (under a particular interleaving)

Protocol rules

- a particular interleaving -

$\mathsf{in}(u_1);$

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

$\ldots$

$\mathsf{out}(v_n)$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 + \text{some tests} \\ \quad ... \\ T_0, v_1, .., v_n (\overset{?}{\vdash} s) \end{cases}$$

Symbolic representation using constraints

Constraint systems are used to represent all the possible behaviours of the attacker (under a particular interleaving)

Protocol rules

- a particular interleaving -

in(u_1);

out(v_1); in(u_2);

...

out(v_n)

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 + \text{some tests} \\ \quad \ldots \\ T_0, v_1, .., v_n (\overset{?}{\vdash} s) \end{cases}$$

A solution of $\mathcal{C}$ is a substitution σ such that

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

for every $u = v \in \mathcal{C}$ (resp. $u \neq v$), $u\sigma =_E v\sigma$ (resp. $u\sigma \neq_E v\sigma$)

Deciding satisfiability of a constraint system

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

$$T_0 \overset{?}{\vdash} x_0, \quad T_0 \cup T_1 \overset{?}{\vdash} x_1, \quad \ldots, \quad T_0 \cup \ldots \cup T_n \overset{?}{\vdash} x_n$$

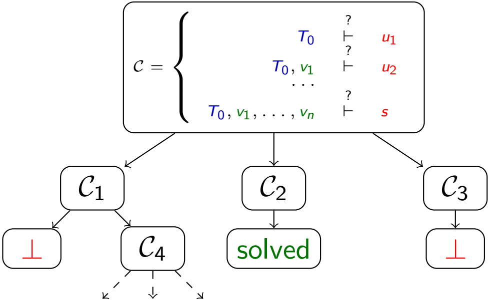
$\longrightarrow$ a system in solved form always has a solution !

Deciding satisfiability of a constraint system

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

$$T_0 \overset{?}{\vdash} x_0, \quad T_0 \cup T_1 \overset{?}{\vdash} x_1, \quad \ldots, \quad T_0 \cup \ldots \cup T_n \overset{?}{\vdash} x_n$$

$\longrightarrow$ a system in solved form always has a solution !



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

Deciding testing equivalence $P \approx_t Q$

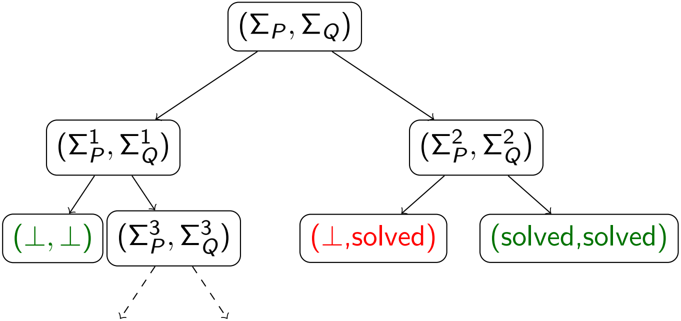
Main idea:

1. for each sequence (in + out)*, we compute a pair (Σ_P, Σ_Q) of sets constraint systems (extended to keep track of some information)

Deciding testing equivalence $P \approx_t Q$

Main idea:

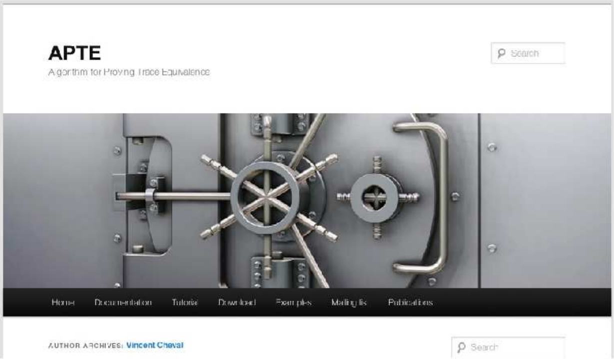
1. for each sequence (in + out)*, we compute a pair (Σ_P, Σ_Q) of sets constraint systems (extended to keep track of some information)
2. we rewrite such a pair until a trivial failure or a trivial success is found.



APTE- Algorithm for Proving Trace Equivalence

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

⟶ developed by Vincent Cheval



⟶ written in Ocaml, around 12 KLocs

Limitation of this approach

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

Limitation of this approach

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

2. the algebraic properties of the primitives are abstracted away
   $\longrightarrow$ *no guarantee if the protocol relies on an encryption that satisfies some additional properties*

   Example: homomorphism property of RSA encryption

$$\{m_1\}_{\mathsf{pub}(S)} \times \{m_2\}_{\mathsf{pub}(S)} = \{m_1 \times m_2\}_{\mathsf{pub}(S)}$$

Limitation of this approach

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

2. the algebraic properties of the primitives are abstracted away
   $\longrightarrow$ *no guarantee if the protocol relies on an encryption that satisfies some additional properties*

   Example: homomorphism property of RSA encryption

$$\{m_1\}_{\mathsf{pub}(S)} \times \{m_2\}_{\mathsf{pub}(S)} = \{m_1 \times m_2\}_{\mathsf{pub}(S)}$$

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

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.

Conclusion

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

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

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

VIP - Verification of Indistinguishability Properties

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