Analysing privacy-type properties in cryptographic protocols

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Wednesday, January 14th, 2015
Cryptographic protocols everywhere!

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.
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

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 un-linkability.

ISO/IEC standard 15408

Un-linkability 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 Access Control (BAC) protocol

Passport \((K_E, K_M)\)

Reader \((K_E, K_M)\)
Basic Access Control (BAC) protocol

Passport $(K_E, K_M)$

Reader $(K_E, K_M)$

get_challenge
Basic Access Control (BAC) protocol

Passport\( (K_E, K_M)\)

Reader\( (K_E, K_M)\)

get\_challenge

\(N_P, K_P\)

\(N_P\)
Basic Access Control (BAC) protocol

Passport \((K_E, K_M)\)

Reader \((K_E, K_M)\)

\[\text{get\_challenge}\]

\[N_P, K_P\]

\[N_P\]


\[N_R, K_R\]
Basic Access Control (BAC) protocol

Passport
$(K_E, K_M)$

Reader
$(K_E, K_M)$

$N_P, K_P$

$N_R, K_R$

$\text{get\_challenge}$


Basic Access Control (BAC) protocol

Passport
\((\mathcal{K}_E, \mathcal{K}_M)\)

Reader
\((\mathcal{K}_E, \mathcal{K}_M)\)

\(\text{get\_challenge}\)

\(\mathcal{N}_P, \mathcal{K}_P\)

\(\mathcal{N}_P\)

\(\{\mathcal{N}_R, \mathcal{N}_P, \mathcal{K}_R\}_{\mathcal{K}_E}, \text{MAC}_{\mathcal{K}_M}(\{\mathcal{N}_R, \mathcal{N}_P, \mathcal{K}_R\}_{\mathcal{K}_E})\)

\(\{\mathcal{N}_P, \mathcal{N}_R, \mathcal{K}_P\}_{\mathcal{K}_E}, \text{MAC}_{\mathcal{K}_M}(\{\mathcal{N}_P, \mathcal{N}_R, \mathcal{K}_P\}_{\mathcal{K}_E})\)

\(\mathcal{K}_{\text{seed}} = f(\mathcal{K}_P, \mathcal{K}_R)\)

\(\mathcal{K}_{\text{seed}} = f(\mathcal{K}_P, \mathcal{K}_R)\)

\(\mathcal{N}_R, \mathcal{K}_R\)
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 cannot 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**, we have:

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

\[\uparrow\]

\[\text{many sessions for each passport}\]

\[\text{only one session for each passport}\]

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

→ the passport must reply to all received messages.

Passport \((K_E, K_M)\)

Reader \((K_E, K_M)\)

\[
\text{get\_challenge}
\]

\[
N_P, K_P
\]

\[
N_P
\]

\[
\]

\[
N_R, K_R
\]
French electronic passport

→ the passport must reply to all received messages.

Passport

$(K_E, K_M)$

Reader

$(K_E, K_M)$

get_challenge

$N_P, K_P$

$N_P$


If MAC check fails

mac_error

$N_R, K_R$
French electronic passport

→ the passport must reply to all received messages.

Passport \((K_E,K_M)\)

Reader \((K_E,K_M)\)

\(N_P, K_P\)

\(N_P\)


If MAC check succeeds

If nonce check fails

nonce_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 eavesdrops on Alice using her passport and records message $M$.

Alice’s Passport

$(K_E, K_M)$

$N_P, K_P$

Reader

$(K_E, K_M)$

$N_R, K_R$


get\_challenge

$N_P$
Part 2 of the attack.
The attacker replays the message $M$ and checks the error code he receives.

**????’s Passport**

$\left( K_E', K_M' \right)$

---

**Attacker**

get\_challenge

$N_P'$


$N_P'$
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.

???'s Passport

$(K'_E, K'_M)$

Attacker

get_challenge

$N'_P, K'_P$


$N'_P$

mac_error

$\Rightarrow$ MAC check failed $\Rightarrow K'_M \neq K_M$ $\Rightarrow$ ?? 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.

$$\text{nonce\_error} \Rightarrow \text{MAC check succeeded} \Rightarrow K'_M = K_M \Rightarrow \text{????'s Passport is Alice}$$
Outline

Does the protocol satisfy a security property?

Modelling

Outline of the remaining of this talk

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

→ we focus here on privacy-type security properties
Modelling cryptographic protocols and their security properties
### Applied pi calculus

Basic programming language with constructs for concurrency and communication

→ based on the \( \pi \)-calculus [Milner et al., 92] ...

\[
\begin{align*}
P, Q & := 0 & \text{null process} \\
in(c, x).P & & \text{input} \\
out(c, u).P & & \text{output} \\
\text{if } u = v \text{ then } P \text{ else } Q & & \text{conditional} \\
P | Q & & \text{parallel composition} \\
!P & & \text{replication} \\
\text{new } n.P & & \text{fresh name generation}
\end{align*}
\]
Protocols as processes

Applied pi calculus

basic programming language with constructs for concurrency and communication

\[ \rightarrow \] based on the $\pi$-calculus [Milner et al., 92] ...

\[ P, Q := \begin{align*}
0 & \quad \text{null process} \\
in(c, x).P & \quad \text{input} \\
out(c, u).P & \quad \text{output} \\
\text{if } u = v \text{ then } P \text{ else } Q & \quad \text{conditional} \\
P \mid Q & \quad \text{parallel composition} \\
!P & \quad \text{replication} \\
n\text{ew } n.P & \quad \text{fresh name generation}
\end{align*} \]

... 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 $\mathcal{F}$.

$t ::= \begin{aligned} n & \quad \text{name } n \\ \mid f(t_1, \ldots, t_k) & \quad \text{application of symbol } f \in \mathcal{F} \end{aligned}$
Messages as terms

Messages are abstracted by (ground) terms

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

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

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 $\mathcal{F}$.

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

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

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

- Names: $n$, $k$, $a$
- Constructors: $\text{senc}$, $\text{pair}$,
- Destructors: $\text{sdec}$, $\text{proj}_1$, $\text{proj}_2$.

$\rightarrow \text{sdec}(\text{senc}(x, y), y) = x$, $\text{proj}_1(\text{pair}(x, y)) = x$, $\text{proj}_2(\text{pair}(x, y)) = y$. 
Cryptographic primitives are modelled using function symbols

- encryption/decryption: $senc/2$, $sdec/2$
- concatenation/projections: $\langle, \rangle/2$, $proj_1/1$, $proj_2/1$
- mac construction: $mac/2$

$$\rightarrow \quad sdec(senc(x, y), y) = x, \quad proj_1(\langle x, y \rangle) = x, \quad proj_2(\langle x, y \rangle) = y.$$ 

Nonces $n_r$, $n_p$, and keys $k_r$, $k_p$, $k_e$, $k_m$ are modelled using names
Cryptographic primitives are modelled using function symbols

- encryption/decryption: senc/2, sdec/2
- concatenation/projections: ⟨, ⟩/2, proj₁/1, proj₂/1
- mac construction: mac/2

\[ \rightarrow \quad \text{sdec}(\text{senc}(x, y), y) = x, \quad \text{proj}_1(⟨x, y⟩) = x, \quad \text{proj}_2(⟨x, y⟩) = y. \]

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

Modelling Passport’s role

\[
P_{\text{BAC}}(k_E, k_M) = \text{new } n_P.\text{new } k_P.\text{in}(⟨z_E, z_M⟩).
\]

\[
\quad \text{if } z_M = \text{mac}(z_E, k_M) \text{ then if } n_P = \text{proj}_1(\text{proj}_2(\text{sdec}(z_E, k_E))) \text{ then out}(⟨m, \text{mac}(m, k_M)⟩) \text{ else out}(\text{nonce\_error}) \text{ else out}(\text{mac\_error})
\]

where \( m = \text{senc}(⟨n_P, \langle \text{proj}_1(z_E), k_P⟩⟩, k_E). \)
Semantics $\rightarrow$:

**Comm** \[\text{out}(c, u).P \mid \text{in}(c, x).Q \rightarrow P \mid Q\{u/x\}\]

**Then** \[\text{if } u = v \text{ then } P \text{ else } Q \rightarrow P \text{ when } u \equiv_v v\]

**Else** \[\text{if } u = v \text{ then } P \text{ else } Q \rightarrow Q \text{ when } u \not\equiv_v v\]
Semantics →:

**COMM** \( \text{out}(c, u).P \mid \text{in}(c, x).Q \rightarrow P \mid Q\{u/x\} \)

**THEN** \( \text{if } u = v \text{ then } P \text{ else } Q \rightarrow P \text{ when } u =_E v \)

**ELSE** \( \text{if } u = v \text{ then } P \text{ else } Q \rightarrow Q \text{ when } u \neq_E v \)

closed by

- **structural equivalence** \( (\equiv) \):
  
  \[ P \mid Q \equiv Q \mid P, \quad P \mid 0 \equiv P, \quad \ldots \]

- **application of evaluation contexts**:

\[
\frac{P \rightarrow P'}{\text{newn. } P \rightarrow \text{newn. } P'} \quad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}
\]
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:

$$(A \parallel P) \Downarrow_c \mbox{ if, and only if, } (A \parallel Q) \Downarrow_c$$

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

$$(A | P) \downarrow_c \text{ if, and only if, } (A | Q) \downarrow_c$$

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

Example 1: $$\text{out}(a, s) \approx_t \text{out}(a, s')$$
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:

$$\text{out}(a, s) \not\approx_t \text{out}(a, s')$$

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

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

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

Example 2:

$$\text{new } s.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s, k')) \approx_t \text{new } s, s'.\text{out}(a, \text{senc}(s, k)).\text{out}(a, \text{senc}(s', k'))$$
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:

$$\text{new } s.\text{out}(a, s\text{enc}(s, k)).\text{out}(a, s\text{enc}(s, k')) \not\approx_t$$

$$\text{new } s, s'.\text{out}(a, s\text{enc}(s, k)).\text{out}(a, s\text{enc}(s', k'))$$

$$\rightarrow A = \text{in}(a, x).\text{in}(a, y).\text{if } (s\text{dec}(x, k) = s\text{dec}(y, k')) \text{ then out}(c, ok)$$
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?

$$\text{new } s.\text{out}(a, s) \approx_t \text{new } s.\text{new } k.\text{out}(a, \text{senc}(s, k))$$
Some privacy-type properties

Unlinkability

\[ !\text{new } ke.\text{new } km.(!P_{BAC} \mid !R_{BAC}) \sim_t !\text{new } ke.\text{new } km. (P_{BAC} \mid !R_{BAC}) \]

↑

many sessions for each passport

only one session for each passport

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

Unlinkability

\[
\!\text{new } ke.\text{new } km. (!P_{BAC} \mid !R_{BAC}) \not\approx_t \!\text{new } ke.\text{new } km. (P_{BAC} \mid !R_{BAC}) \\
\uparrow
\]

\begin{itemize}
  \item many sessions for each passport
  \item only one session for each passport
\end{itemize}

Vote privacy

\[ V_A(\text{yes}) \approx_t V_A(\text{no}) \]

[Arapinis et al, 2010]

[Kremer and Ryan, 2005]
Some privacy-type properties

Unlinkability

![new ke.new km.(!P_{BAC} \mid !R_{BAC}) \approx_t !new ke.new km. (P_{BAC} \mid !R_{BAC})]

- many sessions for each passport
- only one session for each passport

Vote privacy

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

- A votes yes
- B votes no
- A votes no
- B votes yes
Some privacy-type properties

Unlinkability

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

\[ \uparrow \]

many sessions for each passport

\[ \uparrow \]

only one session for each passport

[Arapinis et al, 2010]

Vote privacy

\[ S[ V_A(\text{yes}) \mid V_B(\text{no}) ] \approx_t S[ V_A(\text{no}) \mid V_B(\text{yes}) ] \]

\[ \uparrow \]

A votes yes
B votes no

\[ \uparrow \]

A votes no
B votes yes

\[ \rightarrow \text{ often requires some assumptions } S[\_\_] \]

[Kremer and Ryan, 2005]
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]

  - 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]

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>+ various cryptographic primitives</td>
<td></td>
</tr>
<tr>
<td>- termination is not guaranteed; diff-equivalence (<strong>too strong</strong>)</td>
<td></td>
</tr>
</tbody>
</table>

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

- We still have to consider any possible behavior for the attacker (for all quantification over processes).
  - \(\rightarrow\) no hope to test each possible behavior of the attacker in turn

- Once the behavior of the attacker is fixed, we still have to decide whether the two sequences of messages that are outputted are indistinguishable or not.
  - \(\rightarrow\) the so-called static equivalence problem.
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 traces (i.e. experiment) are represented by a finite set of symbolic traces.
   
   \[\rightarrow\] this set is still huge (exponential)!

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

   \[\rightarrow\] 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 traces (i.e. experiment) are represented by a finite set of symbolic traces.
   \[ \rightarrow \text{this set is still huge (exponential)} ! \]

2. A decision procedure for deciding (symbolic) equivalence between sets of symbolic traces.
   \[ \rightarrow \text{this algorithm works quite well} \]

Some applications

- unlinkability in RFID protocols (e.g. e-passport protocol)
- anonymity (e.g. private authentication protocol)
Our procedure in a nutshell

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 symbolic traces.
   → this set is still huge (exponential)!

2. A decision procedure for deciding (symbolic) equivalence between sets of symbolic traces.
   → this algorithm works quite well

Main limitations

- e-voting protocols are still out of reach
- we can only handle very few sessions (state space explosion problem)
APTE- Algorithm for Proving Trace Equivalence

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

→ developed by Vincent Cheval

→ written in Ocaml, around 12 KLocs
Conclusion - What remains to do?

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

- formal definitions of some subtle security properties (receipt-freeness, coercion-resistance, . . .)
- algorithms (and tools!) for checking (automatically or not) testing equivalence for various cryptographic primitives;
- result to allow a modular analysis

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

→ a postdoc position is available on this project.