A process algebraic analysis of privacy-type properties in cryptographic protocols

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Saturday, September 6th, 2014
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.

→ studied in [Arapinis et al., 10]
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 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 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 \((K_E, K_M)\)

get\_challenge

\(N_P, K_P\)

\(N_P\)


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

Passport

\((K_E, K_M)\)

\(N_P, K_P\)

Reader

\((K_E, K_M)\)

\(N_R, K_R\)

get\_challenge

\(N_P\)


\(N_P, N_R, K_P\)_{K_E}, \ MAC_{K_M}(\{N_P, N_R, K_P\}_{K_E})\)
Basic Access Control (BAC) protocol

Passport
$(K_E, K_M)$

Reader
$(K_E, K_M)$

get_challenge

$N_P, K_P$

$N_P$


${N_P, N_R, K_P}_K_E, \ MAC_{K_M}({N_P, N_R, K_P}_K_E)$

$K_{seed} = f(K_P, K_R)$

$N_R, K_R$

$K_{seed} = f(K_P, K_R)$
**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,

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

↑

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

→ the passport must reply to all received messages.

Passport \((K_E, K_M)\)

\[\text{get\_challenge}\]

\[N_P, K_P\]

\[N_P\]


Reader \((K_E, K_M)\)
the passport must reply to all received messages.
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 eavesdrops on Alice using her passport and records message $M$.

**Alice’s Passport**

$(K_E, K_M)$

**Reader**

$(K_E, K_M)$

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.

$\mathbf{???'s Passport}$

$(K'_E, K'_M)$

Attacker

$\mathbf{get\_challenge}$

$N'_P, K'_P$

$N'_P$


$\mathbf{mac\_error}$

$\Rightarrow$ MAC check failed $\Rightarrow$ $K'_M \neq K_M$ $\Rightarrow$ $\mathbf{???' is not Alice}$
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.

```plaintext

????'s Passport

$(K'_E, K'_M)$

Attacker

get_challenge

$N'_P, K'_P$

$N'_P$


nonce_error

⇒ MAC check succeeded ⇒ $K'_M = K_M$ ⇒ ???? 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
Part I

Modelling cryptographic protocols and their security properties
Protocols as processes

Applied pi calculus

basic programming language with constructs for concurrency and communication

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

\[
P, Q := 0 \\
in(c, x).P \\
out(c, u).P \\
if u = v \text{ then } P \text{ else } Q \\
P | Q \\
!P \\
new n.P
\]

null process
input
output
conditional
parallel composition
replication
fresh name generation
Protocols as processes

Applied pi calculus \[\text{[Abadi \\& Fournet, 01]}\]

Basic programming language with constructs for concurrency and communication

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

\[
P, Q \ ::= \ 0 \quad \text{null process}
\]

\[
in(c, x).P
\quad \text{input}
\]

\[
out(c, u).P
\quad \text{output}
\]

\[
\text{if } u = v \text{ then } P \text{ else } Q
\quad \text{conditional}
\]

\[
P \parallel Q
\quad \text{parallel composition}
\]

\[
!P
\quad \text{replication}
\]

\[
\text{new } n.P
\quad \text{fresh name generation}
\]

... 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 $\mathcal{F}$.

$$t ::= n \quad \text{name } n$$

$$| f(t_1, \ldots, t_k) \quad \text{application of symbol } f \in \mathcal{F}$$
Messages as terms

Messages are abstracted by (ground) terms

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

$$t ::= n \quad \text{name } n$$

$$\mid f(t_1, \ldots, t_k) \quad \text{application of symbol } f \in \mathcal{F}$$

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 $\mathcal{F}$.

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

The term algebra is equipped with an equational theory $\mathcal{E}$.

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

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

$$\rightarrow \text{sdec}(\text{senc}(x, y), y) = x, \quad \text{proj}_1(\text{pair}(x, y)) = x, \quad \text{proj}_2(\text{pair}(x, y)) = y.$$
Going back to the e-passport

Cryptographic primitives are modelled using function symbols

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

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

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: \(s\text{enc}/2, s\text{dec}/2\)
- concatenation/projections: \(\langle , \rangle/2, \text{proj}_1/1, \text{proj}_2/1\)
- mac construction: \(\text{mac}/2\)

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

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

Modelling Passport’s role

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

if \(z_M = \text{mac}(z_E, k_M)\) then if \(n_P = \text{proj}_1(\text{proj}_2(\text{sdec}(z_E, k_E)))\)

then \(\text{out}(\langle m, \text{mac}(m, k_M) \rangle)\)

else \(\text{out}(\text{nonce\_error})\)

else \(\text{out}(\text{mac\_error})\)

where \(m = \text{senc}(\langle n_P, \langle \text{proj}_1(z_E), k_P \rangle \rangle, k_E)\).
Semantics →:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMM</strong></td>
<td>`out(c, u).P</td>
</tr>
<tr>
<td><strong>THEN</strong></td>
<td><code>if u = v then P else Q → P</code> when <code>u =_E v</code></td>
</tr>
<tr>
<td><strong>ELSE</strong></td>
<td><code>if u = v then P else Q → Q</code> when <code>u ≠_E v</code></td>
</tr>
</tbody>
</table>
Semantics →:

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

**THEN** if \( u = v \) then \( P \) else \( Q \rightarrow P \) when \( u \equiv_E v \)

**ELSE** if \( u = v \) then \( P \) else \( Q \rightarrow Q \) when \( u \not\equiv_E v \)

closed by

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

- **application of evaluation contexts**:
  \[
  \frac{P \rightarrow P'}{	ext{new } n.\ P \rightarrow \text{new } n.\ P'} \quad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}
  \]
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:

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

where $P \Downarrow_c$ means that $P$ can evolve and emits on public channel $c$. 
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: $\text{out}(a, s) \approx_t \text{out}(a, s')$
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: $\text{out}(a, s) \not\approx_t \text{out}(a, s')$

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

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

$$\not\approx_t$$

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

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

**Question:** Are the two following processes in testing equivalence?

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

Unlinkability

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

many sessions for each passport

only one session for each passport

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

Unlinkability

\[ \text{!new ke.new km.}(\text{!P}_{\text{BAC}} | \text{!R}_{\text{BAC}}) \approx_t \text{!new ke.new km.}(\text{P}_{\text{BAC}} | \text{!R}_{\text{BAC}}) \]

\[ \uparrow \]

many sessions for each passport

only one session for each passport

[Arapinis et al, 2010]

Vote privacy

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

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

Unlinkability

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

\[ \uparrow \]

many sessions for each passport

\[ \uparrow \]

only one session for each passport

Vote privacy

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

\[ \uparrow \]

A votes yes
B votes no

\[ \uparrow \]

A votes no
B votes yes

[Reprinted with permission from [Arapinis et al, 2010]]

[Reprinted with permission from [Kremer and Ryan, 2005]]
Some privacy-type properties

Unlinkability

Unlinkability of key-pairs:

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

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

Vote privacy

Vote privacy in a voting protocol:

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

- A votes yes
- B votes no
- A votes no
- B votes yes

→ often requires some assumptions \(S[\_\_]\)
Designing verification algorithms for privacy-type properties
Difficulties when checking testing equivalence $P \approx_t Q$

testing equivalence is undecidable in general
Difficulties when checking testing equivalence $P \approx_t Q$

**testing equivalence is undecidable in general**

Processes without replication

- We still have to consider any possible behavior for the attacker (for all quantification over processes).
  - → 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.
  - → the so-called static equivalence problem.
### Static equivalence $\sigma \sim \sigma'$ (modulo $E$)

Two sequences of messages $\sigma = \{w_1 \rightarrow u_1, \ldots, w_n \rightarrow u_n\}$ and $\sigma' = \{w_1 \rightarrow u'_1, \ldots, w_n \rightarrow u'_n\}$ are in static equivalence when:

$$R_1 \sigma =_E R_2 \sigma \iff R_1 \sigma' =_E R_2 \sigma'$$

for any recipes $R_1, R_2$. 

---

S. Delaune (LSV)  
Verification of cryptographic protocols  
6th September 2014
Static equivalence

Two sequences of messages $\sigma = \{w_1 \rightarrow u_1, \ldots, w_n \rightarrow u_n\}$ and $\sigma' = \{w_1 \rightarrow u'_1, \ldots, w_n \rightarrow u'_n\}$ are in static equivalence when:

$$R_1 \sigma =_E R_2 \sigma \iff R_1 \sigma' =_E R_2 \sigma'$$

for any recipes $R_1, R_2$.

Example: a simple voting protocol with 3 voters $a, b,$ and $c$.

$$V \rightarrow S : \langle V, \{vote\}_k \rangle$$
Static equivalence

**Static equivalence $\sigma \sim \sigma'$ (modulo E)**

Two sequences of messages $\sigma = \{w_1 \rightarrow u_1, \ldots, w_n \rightarrow u_n\}$ and $\sigma' = \{w_1 \rightarrow u'_1, \ldots, w_n \rightarrow u'_n\}$ are in static equivalence when:

$$R_1 \sigma =_E R_2 \sigma \iff R_1 \sigma' =_E R_2 \sigma'$$

for any recipes $R_1, R_2$.

**Example:** a simple voting protocol with 3 voters $a$, $b$, and $c$.

$$V \rightarrow S : \langle V, \{vote\}_k \rangle$$

- $\sigma = \{w_1 \rightarrow \langle a, \{yes\}_k \rangle; \ w_2 \rightarrow \langle b, \{no\}_k \rangle; \ w_3 \rightarrow \langle c, \{yes\}_k \rangle\}$, and
- $\sigma' = \{w_1 \rightarrow \langle a, \{no\}_k \rangle; \ w_2 \rightarrow \langle b, \{yes\}_k \rangle; \ w_3 \rightarrow \langle c, \{yes\}_k \rangle\}$.

where $a$, $b$, $c$, $yes$, and $no$ are public constants.
Static equivalence

Static equivalence $\sigma \sim \sigma'$ (modulo $E$)

Two sequences of messages $\sigma = \{w_1 \rightarrow u_1, \ldots, w_n \rightarrow u_n\}$ and $\sigma' = \{w_1 \rightarrow u'_1, \ldots, w_n \rightarrow u'_n\}$ are in static equivalence when:

$$R_1\sigma =_E R_2\sigma \iff R_1\sigma' =_E R_2\sigma'$$

for any recipes $R_1, R_2$.

Example: a simple voting protocol with 3 voters $a$, $b$, and $c$.

$V \rightarrow S : \langle V, \{\text{vote}\}_k \rangle$

- $\sigma = \{w_1 \rightarrow \langle a, \{\text{yes}\}_k \rangle; w_2 \rightarrow \langle b, \{\text{no}\}_k \rangle; w_3 \rightarrow \langle c, \{\text{yes}\}_k \rangle\}$, and
- $\sigma' = \{w_1 \rightarrow \langle a, \{\text{no}\}_k \rangle; w_2 \rightarrow \langle b, \{\text{yes}\}_k \rangle; w_3 \rightarrow \langle c, \{\text{yes}\}_k \rangle\}$.

where $a$, $b$, $c$, yes, and no are public constants.

$\rightarrow \sigma$ and $\sigma'$ are not in static equivalence

$R_1 = \text{proj}_2(w_1)$ and $R_2 = \text{proj}_2(w_3)$.
Static equivalence

Static equivalence $\sigma \sim \sigma'$ (modulo $E$)

Two sequences of messages $\sigma = \{ w_1 \to u_1, \ldots, w_n \to u_n \}$ and $\sigma' = \{ w_1 \to u'_1, \ldots, w_n \to u'_n \}$ are in static equivalence when:

$$R_1 \sigma =_E R_2 \sigma \iff R_1 \sigma' =_E R_2 \sigma'$$

for any recipes $R_1, R_2$

The static equivalence problem is decidable (even in PTIME) for many interesting equational theories useful to model cryptography primitives.

[Abadi & Cortier, TCS 2006], [Cortier & Delaune, JAR 2012]

→ Some automatic tools are available, e.g.

- **KISS**: [http://www.lsv.ens-cachan.fr/~ciobaca/kiss/](http://www.lsv.ens-cachan.fr/~ciobaca/kiss/)
- **FAST**: [http://www.infsec.ethz.ch/people/brunoco](http://www.infsec.ethz.ch/people/brunoco)
Testing equivalence (for processes with replication)

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)

Cheval, Comon-Lundh & D.
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).
Testing equivalence (for processes without replication)

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.
   → this set is still huge (exponential)!

2. A decision procedure for deciding (symbolic) equivalence between sets of symbolic traces.
   → 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.
   ----> this set is still huge (exponential)!

2. A decision procedure for deciding (symbolic) equivalence between sets of symbolic traces.
   ----> 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 \textit{subtle security properties} (receipt-freeness, coercion-resistance, \ldots)
- algorithms (and tools!) for checking (automatically or not) testing equivalence for \textit{various} cryptographic primitives;
- more \textit{composition} results.

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

\rightarrow \textit{a postdoc position is available on this project.}