Towards a Logic for Verification of Security Protocols

Work in progress
Comments welcome

Vincent Bernat

Laboratoire Spécification et Vérification
CNRS & ENS Cachan
Plan

1. Intro
   Existing models, caveats, reactive systems, properties

2. Transition system
   States, constraints, inference rules

3. Logics
   Temporal logics, expressiveness, decidability

4. Applications
Existing models

Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfies a property $\phi$?
Existing models

Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfy a property $\phi$?

To prove protocols, you may use:

- Process algebra
Verification of cryptographic protocols is a *model checking* problem: does a protocol $P$ satisfy a property $\phi$?

To prove protocols, you may use:

- Process algebra
- Trace models
Verification of cryptographic protocols is a *model checking* problem: does a protocol \( P \) satisfies a property \( \phi \)?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]

- Trace models
Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfies a property $\phi$?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]
  - Spi-calculus [Abadi Gordon 97]
- Trace models
Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfy a property $\phi$?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]
  - Spi-calculus [Abadi Gordon 97]
- Trace models
  - [Paulson 98] [Millen Rueß 2000]
Verification of cryptographic protocols is a *model checking* problem: does a protocol $P$ satisfies a property $\phi$?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]
  - Spi-calculus [Abadi Gordon 97]

- Trace models
  - [Paulson 98] [Millen Rueß 2000]
  - Rewriting rules [Rusinowitch Turuani 2001]
Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfies a property $\phi$?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]
  - Spi-calculus [Abadi Gordon 97]

- Trace models
  - [Paulson 98] [Millen Rueß 2000]
  - Rewriting rules [Rusinowitch Turuani 2001]
  - Horn clauses [Comon-Lundh Cortier 2003] [Blanchet 2002]
Verification of cryptographic protocols is a model checking problem: does a protocol $P$ satisfies a property $\phi$?

To prove protocols, you may use:

- Process algebra
  - CSP [Schneider 96]
  - Spi-calculus [Abadi Gordon 97]

- Trace models
  - [Paulson 98] [Millen Rueß 2000]
  - Rewriting rules [Rusinowitch Turuani 2001]
  - Horn clauses [Comon-Lundh Cortier 2003]
    [Blanchet 2002]

- Others...
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.

Secrecy \[ \text{Inst}(M) \simeq \text{Inst}(M') \] if \[ M \simeq M' \], for all \( M, M' \).
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.

**Secrecy** \( \text{Inst}(M) \sim \text{Inst}(M') \) if \( M \sim M' \), for all \( M, M' \).

**Authentication** \( \text{Inst}(M) \sim \text{Inst}_{\text{spec}}(M) \) for all \( M \).
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.

**Secrecy** \( \text{Inst}(M) \simeq \text{Inst}(M') \) if \( M \simeq M' \), for all \( M, M' \).

**Authentication** \( \text{Inst}(M) \simeq \text{Inst}_{\text{spec}}(M) \) for all \( M \).

Many problems:
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.

Secrecy \( \text{Inst}(M) \sim \text{Inst}(M') \) if \( M \sim M' \), for all \( M, M' \).

Authentication \( \text{Inst}(M) \sim \text{Inst}_{\text{spec}}(M) \) for all \( M \).

Many problems:

- Secrecy is tightly linked to observational equivalence: how to express another kind of secrecy property?

SPV'03 - Marseille – p.4/17
Caveats of spi-calculus

Introduced in [Abadi Gordon 97]. Properties are based on observational equivalence.

Secrecy \( \text{Inst}(M) \simeq \text{Inst}(M') \) if \( M \simeq M' \), for all \( M, M' \).

Authentication \( \text{Inst}(M) \simeq \text{Inst}_{\text{spec}}(M) \) for all \( M \).

Many problems:

- Secrecy is tightly linked to observational equivalence: how to express another kind of secrecy property?

- To express an authentication property, one has to build an \textit{ad hoc} process: difficulty to compare authentication properties between two different protocols.
Caveats for other models

Most of these models are primary targeted to express protocols.
Caveats for other models

Most of these models are primary targeted to express protocols.

Questions you may ask:

- Given two properties, one expressed with [Comon-Lundh Cortier 2003] and one with [Paulson 98], how could you compare them, since they do not work on the same abstraction?
- Given one protocol $P$ in [Schneider 96] and one property $\hat{A}$ expressed with [Comon-Lundh Cortier 2003], how can you check if $\hat{A}$ satisfies $P$?
- Given a set of properties and a protocol $P$ expressed with [Blanchet 2002], how could you check them without altering $P$?
Caveats for other models

Most of these models are primary targeted to express protocols.

Questions you may ask:

- Given two properties, one expressed with [Comon-Lundh Cortier 2003] and one with [Paulson 98], how could you compare them, since they do not work on the same abstraction?
Caveats for other models

Most of these models are primary targeted to express protocols.

Questions you may ask:

Given two properties, one expressed with [Comon-Lundh Cortier 2003] and one with [Paulson 98], how could you compare them, since they do not work on the same abstraction?

Given one protocol $P$ in [Schneider 96] and one property $\phi$ expressed with [Comon-Lundh Cortier 2003], how can you check if $\phi$ satisfies $P$?
Caveats for other models

Most of these models are primary targeted to express protocols.

Questions you may ask:

- Given two properties, one expressed with [Comon-Lundh Cortier 2003] and one with [Paulson 98], how could you compare them, since they do not work on the same abstraction?
- Given one protocol $P$ in [Schneider 96] and one property $\phi$ expressed with [Comon-Lundh Cortier 2003], how can you check if $\phi$ satisfies $P$?
- Given a set of properties and a protocol $P$ expressed with [Blanchet 2002], how could you check them without altering $P$?
The transition system is the semantic glue between the abstraction and the temporal logic.
Reactive systems

Abstraction

petri nets

automata networks

process algebra

...

Temporal logic

LTL

CTL

PLTL

...

The transition system is the semantic glue between the abstraction and the temporal logic.
Reactive systems

Abstraction

petri nets  automata networks  process algebra ...

Transition system

Temporal logic

LTL  CTL  PLTL  ...

The transition system is the semantic glue between the abstraction and the temporal logic.
Reactive systems

Abstraction

petri nets  automata networks  process algebra ...

Transition system

Temporal logic

LTL  CTL  PLTL ...

The transition system is the semantic glue between the abstraction and the temporal logic.
Transposing the approach of reactive systems to cryptographic protocols has the following advantages:
Transposing the approach of \textit{reactive systems} to \textit{cryptographic protocols} has the following advantages:

- Changing the \textit{abstraction} layer while keeping the \textit{logic} is allowed, and vice versa;
Transposing the approach of reactive systems to cryptographic protocols has the following advantages:

- Changing the *abstraction* layer while keeping the *logic* is allowed, and vice versa;
- Easy comparison of *different protocols* for a given property;
Transposing the approach of reactive systems to cryptographic protocols has the following advantages:

- Changing the *abstraction* layer while keeping the *logic* is allowed, and vice versa;
- Easy comparison of *different protocols* for a given property;
- Easy comparison of *different properties* for a given protocol;
Transposing the approach of reactive systems to cryptographic protocols has the following advantages:

- Changing the *abstraction* layer while keeping the *logic* is allowed, and vice versa;
- Easy comparison of different protocols for a given property;
- Easy comparison of different properties for a given protocol;
- No new model needed, current models are fine if the transition system is general enough.
Goals

Conception of a logic for security properties covering at least:

- secrecy
  - from the intruder point of view
  - temporary secrecy
  - partial secrecy
Goals

Conception of a logic for security properties covering at least:

- secrecy
- authentication
  - vivacity
  - weak agreement
  - non-injective agreement
  - agreement
Goals

Conception of a logic for security properties covering at least:

- secrecy
- authentication

Conception of a transition system

- expressive enough to catch both the semantics of the protocols and the semantics of the logics,
- retaining cryptographic protocol specificities to be able to handle it.
1. **Intro**  
   Existing models, caveats, reactive systems, properties

2. **Transition system**  
   States, constraints, inference rules

3. **Logics**  
   Temporal logics, expressiveness, decidability

4. **Applications**
Choosing a transition system

- Powerful enough
Choosing a transition system

- Powerful enough
  - the intruder knowledge
Choosing a transition system

- Powerful enough
  - the intruder knowledge
  - for each triple (actor, session, role):
    - the step number,
    - the actor's memory
    - the last message received
Choosing a transition system

- Powerful enough
  - the intruder knowledge
  - for each triple (actor, session, role):
    - the step number,
    - the actor’s memory
    - the last message received
  - the last message sent on the network
Choosing a transition system

- Powerful enough
  - the intruder knowledge
  - for each triple (actor, session, role):
    - the step number,
    - the actor’s memory
    - the last message received
  - the last message sent on the network
- Specific enough: multiple sources of infinity, we have to work with it
Inference rules

Two built-in rules

Honest replay
If some transition is possible in some state, it should be possible to have a similar transition in some similar state; e.g., replaying a session between two actors.

A state $s$ similar to a state $s_0$ is roughly the same state to which we apply a substitution $\sigma$ on the session numbers.

Nonces are parameterized by session numbers.

Intruder attack
If some transition is done by an agent expecting a term $t$, the intruder can force it to do the transition by providing himself $t$. 
Two built-in rules

**Honest replay**  If some transition is possible in some state, it should be possible to have a similar transition in some similar state; e.g., replaying a session between two actors.
Two built-in rules

**Honest replay**  If some transition is possible in some state, it should be possible to have a similar transition in some similar state; e.g., replaying a session between two actors. A state \( s \) similar to a state \( s' \) is roughly the same state to which we apply a substitution \( \sigma \) on the session numbers. **Nonces** are parameterized by session numbers.
Inference rules

Two built-in rules

**Honest replay**  If some transition is possible in some state, it should be possible to have a similar transition in some similar state; e.g., replaying a session between two actors.
A state $s$ similar to a state $s'$ is roughly the same state to which we apply a substitution $\sigma$ on the session numbers. Nonces are parameterized by session numbers.

**Intruder attack**  If some transition is done by an agent expecting a term $t$, the intruder can force it to do the transition by providing himself $t$. 
Honest replay

\[
q \rightarrow q'
\]

\[
q'' \oplus q\sigma \rightarrow q'' \oplus q'\sigma
\]
Honest replay

\[
q \rightarrow q' \\
q'' \oplus q\sigma \rightarrow q'' \oplus q'\sigma
\]

Partial example:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( s_1 )</td>
<td>( n - 1 )</td>
<td>A, B</td>
<td>( N_{A,s_1} )</td>
</tr>
<tr>
<td>B</td>
<td>( s_2 )</td>
<td>( m - 1 )</td>
<td>B, A</td>
<td>( N_{B,s_2} )</td>
</tr>
<tr>
<td>A</td>
<td>( s_1 )</td>
<td>( n )</td>
<td>A, B</td>
<td>( N_{A,s_1} )</td>
</tr>
<tr>
<td>B</td>
<td>( s_2 )</td>
<td>( m - 1 )</td>
<td>B, A</td>
<td>( N_{B,s_2} )</td>
</tr>
</tbody>
</table>
Partial example:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>s₁</td>
<td>n − 1</td>
<td>A, B</td>
<td>N_A,s₁</td>
</tr>
<tr>
<td>B</td>
<td>s₂</td>
<td>m − 1</td>
<td>B, A</td>
<td>N_B,s₂</td>
</tr>
<tr>
<td>A</td>
<td>s₁</td>
<td>n</td>
<td>A, B</td>
<td>N_A,s₁</td>
</tr>
<tr>
<td>B</td>
<td>s₂</td>
<td>m − 1</td>
<td>B, A</td>
<td>N_B,s₂</td>
</tr>
</tbody>
</table>

Honest replay

\[
q \rightarrow q'
\]

\[
q'' \oplus q\sigma \rightarrow q'' \oplus q'\sigma
\]
### Honest replay

\[
q \rightarrow q' \quad \quad q'' \oplus q\sigma \rightarrow q'' \oplus q'\sigma
\]

**Partial example:**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(s_1)</td>
<td>(n - 1)</td>
<td>A, B</td>
<td>(N_{A,s_1})</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>(s_2)</td>
<td>(m - 1)</td>
<td>B, A</td>
<td>(N_{B,s_2})</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>(s_1)</td>
<td>(n)</td>
<td>A, B</td>
<td>(N_{A,s_1})</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>(s_2)</td>
<td>(m - 1)</td>
<td>B, A</td>
<td>(N_{B,s_2})</td>
<td>B</td>
</tr>
</tbody>
</table>

\[
\sigma = \left\{ \begin{array}{c}
    s_1 \rightarrow s'_1 \\
    s_2 \rightarrow s'_2
\end{array} \right.\]
**Honest replay**

\[
\begin{align*}
q & \rightarrow q' \\
q'' \oplus q\sigma & \rightarrow q'' \oplus q'\sigma
\end{align*}
\]

Partial example:

\[
\begin{array}{cccc|cccc}
A & s_1 & n - 1 & A, B & N_{A,s_1} & A & s'_{1} & n - 1 & A, B & N_{A,s'_1} \\
B & s_2 & m - 1 & B, A & N_{B,s_2} & B & s'_{2} & m - 1 & B, A & N_{B,s'_2} \\
A & s_1 & n & A, B & N_{A,s_1} & A & s'_{1} & n & A, B & N_{A,s'_1} \\
B & s_2 & m - 1 & B, A & N_{B,s_2} & B & s'_{2} & m - 1 & B, A & N_{B,s'_2}
\end{array}
\]

\[
\sigma = \left\{ \begin{array}{c}
s_1 \rightarrow s'_{1} \\
s_2 \rightarrow s'_{2}
\end{array} \right\}
\]
Intruder attack

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \]

\[ \rightarrow q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2 \]

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \]
Intruder attack

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \rightarrow q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \]
\[ \rightarrow q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2 \]

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \rightarrow q \oplus q_A^1 \oplus q_B^2 \oplus e_3 \oplus I_2 \]
Intruder attack

\[
q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \\
\quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2 \\
q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^1 \oplus q_B^2 \oplus e_3 \oplus I_2 \\
\quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^2 \oplus e_2 \oplus I_2
\]

\[e_2 \in q|I\]
Intruder attack

\[
q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \longrightarrow \quad q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \\
\longrightarrow \quad q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2
\]

Partial example:

<table>
<thead>
<tr>
<th>A</th>
<th>s_1</th>
<th>n - 1</th>
<th>A</th>
<th>s_1</th>
<th>n - 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>s_2</td>
<td>m - 1</td>
<td>B</td>
<td>s_2</td>
<td>m - 1</td>
</tr>
<tr>
<td>A</td>
<td>s_1</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>s_2</td>
<td>m - 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Intruder attack

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \]
\[ q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2 \]

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^1 \oplus q_B^2 \oplus e_3 \oplus I_2 \]
\[ q \oplus q_A^2 \oplus q_B^2 \oplus e_2 \oplus I_2 \]

Partial example:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>s₁</th>
<th>n  - 1</th>
<th></th>
<th>A</th>
<th>s₁</th>
<th>n  - 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>s₂</td>
<td>m  - 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>s₁</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>s₁</th>
<th>n  - 1</th>
<th></th>
<th>A</th>
<th>s₁</th>
<th>n  - 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>s₂</td>
<td>m  - 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>s₁</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Intruder attack

\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^1 \oplus e_2 \oplus I_1 \]
\[ \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^2 \oplus e_3 \oplus I_2 \]
\[ q \oplus q_A^1 \oplus q_B^1 \oplus e_1 \quad \rightarrow \quad q \oplus q_A^1 \oplus q_B^2 \oplus e_3 \oplus I_2 \]
\[ \quad \rightarrow \quad q \oplus q_A^2 \oplus q_B^2 \oplus e_2 \oplus I_2 \]

Partial example:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>s_1</td>
<td>n - 1</td>
</tr>
<tr>
<td>B</td>
<td>s_2</td>
<td>m - 1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>s_1</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>s_2</td>
</tr>
<tr>
<td>A</td>
<td>s_1</td>
<td>n</td>
</tr>
<tr>
<td>B</td>
<td>s_2</td>
<td>m - 1</td>
</tr>
<tr>
<td>A</td>
<td>s_1</td>
<td>n</td>
</tr>
<tr>
<td>B</td>
<td>s_2</td>
<td>m</td>
</tr>
</tbody>
</table>
1. Intro
   Existing models, caveats, reactive systems, properties

2. Transition system
   States, constraints, inference rules

3. Logics
   Temporal logics, expressiveness, decidability

4. Applications
Choosing a logic

- Powerful enough
Choosing a logic

- Powerful enough
  - able to express properties on states
Choosing a logic

- Powerful enough
  - able to express properties on states
  - temporal dimension for secrecy
Choosing a logic

- Powerful enough
  - able to express properties on states
  - temporal dimension for secrecy
  - possibility to count to express strong authentication properties
Choosing a logic

- Powerful enough
  - able to express properties on states
  - temporal dimension for secrecy
  - possibility to count to express strong authentication properties

- Specific enough
  - decidability results with bounded number of sessions
  - ?
1. Intro
   Existing models, caveats, reactive systems, properties

2. Transition system
   States, constraints, inference rules

3. Logics
   Temporal logics, expressiveness, decidability

4. Applications
Better understanding of authentication properties

expression and proof of properties independently of the protocol and of the model, e.g. nonces are needed for injective agreement

decidability of a class of properties for a bounded number of sessions