Reducing search space for trace equivalence checking
FOSAD 2013

Lucca Hirschi
LSV, ENS Cachan

September 5, 2013

joint work with David Baelde and Stéphanie Delaune
LSV and LSV
Context

Prove automatically security properties of cryptographic protocols using formal methods.
Context

Prove automatically security properties of cryptographic protocols using formal methods.

Tools

- **Applied-\(\pi\)** models protocols (Dolev-Yao model);
Context

Prove automatically security properties of cryptographic protocols using formal methods.

Tools

- Applied-$\pi$ models protocols (Dolev-Yao model);
- reachability or equivalence model security properties;
Context

Prove automatically security properties of cryptographic protocols using formal methods.

<table>
<thead>
<tr>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ <strong>Applied-(\pi)</strong> models protocols (Dolev-Yao model);</td>
</tr>
<tr>
<td>✓ reachability or <strong>equivalence</strong> model security properties;</td>
</tr>
<tr>
<td>✓ <strong>algorithms</strong> check reachability or equivalence.</td>
</tr>
</tbody>
</table>
Context

Prove automatically security properties of cryptographic protocols using formal methods.

Tools

- Applied-$\pi$ models protocols (Dolev-Yao model);
- reachability or equivalence model security properties;
- algorithms check reachability or equivalence.

Issue

Main bottleneck: size of search space (interleavings).
Context

Prove automatically security properties of cryptographic protocols using formal methods.

Tools

- **Applied-π** models protocols (Dolev-Yao model);
- reachability or **equivalence** model security properties;
- **algorithms** check reachability or equivalence.

Issue

Main bottleneck: size of search space (**interleavings**).

Our Contribution

Reduce search space of **equivalence** checking using POR ideas by eliminating a lot of redundancies.
Our Contribution

Reduce search space of equivalence checking using POR ideas by eliminating a lot of redundancies.

Outline

1. Introduction
2. Model
3. Big Picture
4. Differentiation
5. Conclusion
Outline

1. Introduction
2. Model
3. Big Picture
4. Differentiation
5. Conclusion
Terms

$\mathcal{T}$: a given set of terms modulo an equational theory. E.g.
\[ \text{dec(\text{enc}(m, k), k)} = m. \]

Simple Processes

- $P_c ::= 0 \mid [T]\text{in}(c, x) \mid [T]\text{out}(c, m).P_c \quad m \in \mathcal{T}$
- $P_s ::= P_{c_1}|P_{c_2}| \ldots |P_{c_n} \quad c_i \neq c_j$
Terms

$\mathcal{T}$: a given set of terms modulo an equational theory. E.g.
$\text{dec}(\text{enc}(m, k), k) = m$.

Simple Processes

- $P_c ::= 0 \mid [T]\text{in}(c, x) \mid [T]\text{out}(c, m).P_c \quad m \in \mathcal{T}$
- $P_s ::= P_{c_1}\mid P_{c_2}\mid \ldots P_{c_n} \quad c_i \neq c_j$
- Process: $(P_s; \Phi)$ ($\Phi$ set of messages revealed to the intruder).
**Applied-\(\pi\)**

### Terms

\(\mathcal{T}\): a given set of terms modulo an equational theory. E.g.,

\[
\text{dec} (\text{enc}(m, k), k) = m.
\]

### Simple Processes

- \(P_c ::= 0 \mid [T]\text{in}(c, x) \mid [T]\text{out}(c, m).P_c \quad m \in \mathcal{T}\)
- \(P_s ::= P_{c_1} \mid P_{c_2} \mid \ldots P_{c_n} \quad c_i \neq c_j\)
- Process: \((P_s; \Phi)\) (\(\Phi\) set of messages revealed to the intruder).

### Semantics

\[
\begin{align*}
\text{if } T \land w \text{ fresh in } \Phi & \\
(\{[T].\text{out}(c, m).P\} \cup P; \Phi) & \xrightarrow{\nu w.\text{out}(c, w)} (\{P\} \cup P; \Phi \cup \{w \triangleright m\})
\end{align*}
\]

\[
\text{if } t\Phi = u \land \text{fv}(t) \subseteq \text{dom}(\Phi) & \\
(\{\text{in}(c, x).P\} \cup P; \Phi) & \xrightarrow{\text{in}(c, t)} (\{P[x \mapsto u]\} \cup P; \Phi)
\]

Trace equivalence

- \( \Phi \sim \Phi' \iff \forall M, N, M\Phi = N\Phi \iff M\Phi' = N\Phi' \) and conversely;
- \( A \sim B \iff \forall A \xrightarrow{s} A', \exists B', B \xrightarrow{s} B' \land \Phi_{A'} \sim \Phi_{B'} \) and conversely.

Trace equivalence allows to model anonymity, unlikability, etc.
## Equivalence

### Trace equivalence

<table>
<thead>
<tr>
<th>( \Phi \sim \Phi' )</th>
<th>( \iff \forall M, N, \ M\Phi = N\Phi \iff M\Phi' = N\Phi' ) and conversely;</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \simeq B )</td>
<td>( \iff \forall A \xrightarrow{s} A', \ \exists B', \ B \xrightarrow{s} B' \land \Phi_{A'} \sim \Phi_{B'} ) and conversely.</td>
</tr>
</tbody>
</table>

Trace equivalence allows to model anonymity, unlikability, etc.

### Our aim

Improve algorithms/programs checking trace equivalence (for simple processes).
Symbolic calculus - 1

Inputs messages: infinitely branching $\rightsquigarrow$ symbolic calculus.
Symbolic calculus - 1

Inputs messages: infinitely branching $\leadsto$ symbolic calculus.

System of Constraints

- Constraints: $(X \triangleright x); u = v, (fv^?(X) : \text{dom}(\Phi))$;
- System of constraints: $(\Phi, \mathcal{D})$. 
Symbolic calculus - 1

Inputs messages: infinitely branching $\bowtie$ symbolic calculus.

System of Constraints

- Constraints: $(X \triangleright x); u = v, (fv^?(X) : \text{dom}(\Phi));$
- System of constraints: $(\Phi, \mathcal{D})$.

$$P = \text{out}(c, k).\text{in}(c, x).\text{out}(c, \langle k, x \rangle).\text{in}(c, y)$$

leads to

$$\mathcal{D} = \{X \triangleright x; Y \triangleright y; (fv^?(X) : \{w\}); (fv^?(Y) = \{w; w'\})\}$$

$$\Phi = \{w \triangleright k; w' \triangleright \langle k, x \rangle\}$$
Inputs messages: infinitely branching $\leadsto$ symbolic calculus.

System of Constraints

- Constraints: $(X \triangleright x); u = v, (fv^?(X) : \text{dom}(\Phi))$;
- System of constraints: $(\Phi, \mathcal{D})$.

$$P = \text{out}(c, k).\text{in}(c, x).\text{out}(c, \langle k, x \rangle).\text{in}(c, y)$$

leads to

$$\mathcal{D} = \{X \triangleright x; Y \triangleright y; (fv^?(X) : \{w\}); (fv^?(Y) = \{w; w'\})\}$$

$$\Phi = \{w \triangleright k; w' \triangleright \langle k, x \rangle\}$$

Symbolic processes

$$(P; \Phi; \mathcal{D}; tr)$$
Symbolic Calculus - 2

Semantics:

\[
\begin{align*}
\left\{\left[ T \right].\text{out}(c, m).P \right\} \cup P; \Phi; D; tr & \xrightarrow{s} \nu w.\text{out}(c, X) \{ P \} \cup P; \Phi \cup \{ w \triangleright m \}; D \cup \{ T \}; tr.\nu w.\text{out}(c, X) \\
\text{if } w \text{ fresh in } \phi
\end{align*}
\]

\[
\begin{align*}
\left\{\left[ T \right].\text{in}(c, x).P \right\} \cup P; \Phi; D; tr & \xrightarrow{s} \text{in}(c, X) \{ P; \Phi; D \cup \{ T; (X \triangleright x); (fv^?(X) : \text{dom}(\Phi)) \}; tr.\text{in}(c, X) \}
\end{align*}
\]
Symbolic Calculus - 2

Semantics:

\[
(\{[T].\text{out}(c,m).P}\cup \mathcal{P}; \Phi; \mathcal{D}; tr) \xrightarrow{\nu \text{w}.\text{out}(c,X)}_s \\
(\{P\} \cup \mathcal{P}; \Phi \cup \{w \triangleright m\}; \mathcal{D} \cup \{T\}; tr.\nu \text{w}.\text{out}(c,X))
\]

if \(w\) fresh in \(\phi\)

\[
(\{[T].\text{in}(c,x).P\} \cup \mathcal{P}; \Phi; \mathcal{D}; tr) \xrightarrow{\text{in}(c,X)}_s \\
(\mathcal{P}; \Phi; \mathcal{D} \cup \{T; (X \triangleright x); (fv^?(X) : \text{dom}(\Phi))\}; tr.\text{in}(c,X))
\]

Symbolic equivalence

\[A \approx_s B \iff \forall A \xrightarrow{s} A' \forall \Theta \in \text{Sol}(\Phi_{A'}, \mathcal{D}_{A'}) \exists B' B \xrightarrow{s} B', \Theta \in \text{Sol}(\Phi_{B'}, \mathcal{D}_{B'}) \text{ and } \Phi_{A'} \sim \Phi_{B'} \text{ and conversely.}\]
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Compression

\[ \vdash \approx \quad \text{Thm 1: } \approx = \approx_c \]

Symbolic

\[ \approx_c = \approx_s \]

Differentiation

\[ \approx_s = \approx_{ds} \quad \text{Thm 2: } \approx_s = \approx_{ds} \]
Apply optimizations to SPEC:

- adapt its formalism;
- constraints solving.
Apply optimizations to SPEC:
- adapt its formalism;
- constraints solving.

Implementation
Thm 1: $\varepsilon_c = \varepsilon_s$ 

Simplify the symbolic representation:

Thm 2: $\varepsilon_s = \varepsilon_s^d$

Apply optimizations to SPEC:
- adapt its formalism;
- constraint reduction.

Implementation
Outline

1. Introduction
2. Model
3. Big Picture
4. Differentiation
5. Conclusion
\[ P = \text{in}(a, x).\text{out}(a, k).P_a \mid \text{in}(b, y).\text{out}(b, k').P_b \]
\[ P = \text{in}(a, x).\text{out}(a, k).P_a \mid \text{in}(b, y).\text{out}(b, k').P_b \]
Dependency constraints

Dependency constraint: $w \in \text{message of } x$

We can add constraints on the fly.
Dependency constraints

Dependency constraint: \( w \in \text{message of } x \)

We can add constraints on the fly.

- Eliminate symmetric traces;
Dependency constraints:

- Dependency constraint: \( w \in \) message of \( x \)
- We can add constraints on the fly.
- Eliminate symmetric traces;
- Do not remove too much information (intruder can observe the order).
\[ P = IO(a) | IO(b) | IO(c) \] where \[ IO(x) = \text{in}(x, X).\text{out}(x, w_x) \]
\[ P = \text{IO}(a) \parallel \text{IO}(b) \parallel \text{IO}(c) \quad \text{where} \quad \text{IO}(x) = \text{in}(x, X).\text{out}(x, w_x) \]
\[ P = IO(a) | IO(b) | IO(c) \text{ where } IO(x) = in(x, X).out(x, w_x) \]
\[ P = IO(a)|IO(b)|IO(c) \text{ where } IO(x) = \text{in}(x, X).\text{out}(x, w_x) \]

\[ t = IO(c_1).IO(c_2)\ldots IO(c_n) \overset{\sim}{\longrightarrow} IO(c_n).IO(c_1)\ldots IO(c_{n-1}) \]

- \( c_n < c_1 \);
- \( c_2, c_3 \ldots c_{n-1} < c_n \)

\[ g(t) = \text{there exists } 1 \leq i < n \text{ such that } w_i \in \text{message of } x_n \]
Differentiation

Differentiated semantics
Symbolic semantics + dependency constraints built on the fly.
Differentiation

Differentiated semantics
Symbolic semantics $+$ dependency constraints built on the fly.

$$(\{in(c,x).out(c,m).P\} \cup P; D; \Phi; t) \xrightarrow{io(c,X,w)} s$$

$$(\{P\} \cup P; D \cup \{(X \triangleright x), G(t.io(c,X,w))\}; \Phi \cup \{w \triangleright m\}; t.io(c,X,w))$$

$\rightsquigarrow$ less solutions, less traces/interleavings to check.
Differentiation

Differentiated semantics

Symbolic semantics + dependency constraints built on the fly.

\[
\begin{align*}
\{\text{in}(c, x).\text{out}(c, m).P\} \uplus P; D; \Phi; t & \xrightarrow{\text{io}(c, X, w)} d \\
\{P\} \uplus P; D \cup \{(X \triangleright x), G(t.\text{io}(c, X, w))\}; \Phi \cup \{w \triangleright m\}; t.\text{io}(c, X, w) & \end{align*}
\]

\[\leadsto \text{less solutions, less traces/interleavings to check.}\]

Theorem

\[\approx_d^s = \approx_s\]
Idea of the proof

- $[t]$: set of traces modulo valid permutations;
- $\text{Min}([t])$: lexico. minimum of the class.

**Lemma 1**
If $P$ has an trace $t$ then it has all traces of $[t]$.

**Lemma 2**
- If $P$ has an trace $t$ then it has a differentiated trace $\text{Min}(t)$;
- $P$ has no other differentiated trace in $[t]$. 
Conclusion

- Better differentiation (compression, semantics, extended patterns) for simple processes;
- applied to trace equivalence checking.
- implementation in SPEC.
Conclusion

Better differentiation (compression, semantics, extended patterns) for simple processes;

applied to trace equivalence checking.

implementation in SPEC.

<table>
<thead>
<tr>
<th>Protocol</th>
<th># ac.</th>
<th>T. REF (s)</th>
<th>T. OPT (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 parallels</td>
<td>8</td>
<td>44.59</td>
<td>5.88</td>
</tr>
<tr>
<td>7 parallels</td>
<td>16</td>
<td>∞</td>
<td>370.65</td>
</tr>
<tr>
<td>depth 4</td>
<td>10</td>
<td>42.87</td>
<td>8.42</td>
</tr>
<tr>
<td>depth 10</td>
<td>22</td>
<td>∞</td>
<td>122.27</td>
</tr>
<tr>
<td>WMF, auth. false, 1 sess.</td>
<td>12</td>
<td>30.89</td>
<td>1.87</td>
</tr>
<tr>
<td>WMF, auth., 1 sess.</td>
<td>14</td>
<td>51.54</td>
<td>6.43</td>
</tr>
<tr>
<td>WMF, strong secr., 1 sess.</td>
<td>16</td>
<td>65.20</td>
<td>8.09</td>
</tr>
<tr>
<td>WMF, false, 2 sess.</td>
<td>24</td>
<td>7742.24</td>
<td>3.30</td>
</tr>
<tr>
<td>NSSK, auth., 1 session</td>
<td>10</td>
<td>76.68</td>
<td>22.99</td>
</tr>
<tr>
<td>Yahalom, auth., 1 session</td>
<td>10</td>
<td>6602.82</td>
<td>237.10</td>
</tr>
</tbody>
</table>
Better differentiation (compression, semantics, extended patterns) for simple processes;
- applied to trace equivalence checking.
- implementation in SPEC.

<table>
<thead>
<tr>
<th>Protocol</th>
<th># ac.</th>
<th>T. REF (s)</th>
<th>T. OPT (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 parallels</td>
<td>8</td>
<td>44.59</td>
<td>5.88</td>
</tr>
<tr>
<td>7 parallels</td>
<td>16</td>
<td>∞</td>
<td>370.65</td>
</tr>
<tr>
<td>depth 4</td>
<td>10</td>
<td>42.87</td>
<td>8.42</td>
</tr>
<tr>
<td>depth 10</td>
<td>22</td>
<td>∞</td>
<td>122.27</td>
</tr>
<tr>
<td>WMF, auth. false, 1 sess.</td>
<td>12</td>
<td>30.89</td>
<td>1.87</td>
</tr>
<tr>
<td>WMF, auth., 1 sess.</td>
<td>14</td>
<td>51.54</td>
<td>6.43</td>
</tr>
<tr>
<td>WMF, strong secr., 1 sess.</td>
<td>16</td>
<td>65.20</td>
<td>8.09</td>
</tr>
<tr>
<td>WMF, false, 2 sess.</td>
<td>24</td>
<td>7742.24</td>
<td>3.30</td>
</tr>
<tr>
<td>NSSK, auth., 1 session</td>
<td>10</td>
<td>76.68</td>
<td>22.99</td>
</tr>
<tr>
<td>Yahalom, auth., 1 session</td>
<td>10</td>
<td>6602.82</td>
<td>237.10</td>
</tr>
</tbody>
</table>

Future Work
- Richer class of processes;
- improve constraints solving.