Automation of the Computationally Complete Symbolic Attacker (CCSA)

Simon Jeanteur
March 28, 2023
TU Wien, Vienna, Austria
• Proofs in the CCSA make use of the first-order BC-Logic
• There exists some fairly good automated first-order theorem prover
Proofs in the CCSA make use of the first-order BC-Logic.

There exists some fairly good automated first-order theorem prover.

Can we leverage those provers to automatically prove computationally sound properties on cryptographic protocols?
Speedrunning an example
We consider a simple protocol inspired by the RFID protocols.

- A tag outputs a fresh nonce $n$ in plain text and hashed with a key $k$ shared with the reader.
- The reader then tries to authenticate the tag by checking that the message it received was properly constructed using a key within its database.

$$T_{ij} \rightarrow R_i : \nu n. \langle n, \mathcal{H}(n, k_j) \rangle$$

We model a single reader and multiple tags (parametrized by $j$) over multiple sessions (parametrized by $i$).

We are concerned with the authentication of a tag by the reader. It intuitively means that when a reader accepts an authentication of the $j^{th}$ tag, then the latter really did try to authenticate.
As steps

```
let condition(j:tag_idx, in: Message)
  { verify(sel2of2(in), sel1of2(in), key(j)) }
step reader(i:session, j:tag_idx)
  { condition(j, input(reader(i, j))) } { ok }
step reader_fail(i:session)
  { not(exists (j:tag_idx) {condition(j, input(reader_fail(i)))}) } { ko }
step tag(i:session, j:tag_idx) /* the tag */
  { true } { tpl(nt(i,j), hash(nt(i,j), key(j))) }

/* ordering */
order forall (i:session, j:tag_idx) { reader_fail(i) <> reader(i, j) }
order forall (i:session, j:tag_idx, j2:tag_idx)
  { reader(i, j2) <> reader(i, j) }
```

Speedrunning an example
An axiom example: euf-cma

When `hash` and `verify` form a message authentication code\(^1\) (MAC) scheme that is existentially unforgeable under chosen message attacks (Euf-Cma), the protocol \( \mathcal{P} \) verifies:

\[
\forall m, \sigma, k. \mid verify(\sigma, m, k) \mid \\
\Rightarrow \left( k \sqsubseteq_{hash(\_\_\_, \_\_)} m, \sigma, \mathcal{P} \lor \exists u. (\text{hash}(u, k) \sqsubseteq m, \sigma \land |u| = |m|) \right)
\]

where \( k \) is a nonce and all \( \sqsubseteq \) relations are variations of subterm relations.

\(^1\)i.e. A symmetric signature scheme
$T_{i,j} \rightarrow R_i : \forall n. \langle n, H(n, k_j) \rangle$

The protocol gets encoded in the logic via its interaction with our axioms:

$u \sqsubseteq \text{input}(\tau) \Rightarrow \bigvee$

$\exists i, j. T_{i,j} < \tau \land (u \sqsubseteq \text{msg}(T_{i,j}) \lor u \sqsubseteq \text{cond}(T_{i,j}))$

$\exists i, j. R_{i,j}^{\text{succ}} < \tau \land (u \sqsubseteq \text{msg}(R_{i,j}^{\text{succ}}) \lor u \sqsubseteq \text{cond}(R_{i,j}^{\text{succ}}))$

$\exists i. R_{i}^{\text{fail}} < \tau \land \left( u \sqsubseteq \text{msg}(R_{i}^{\text{fail}}) \lor u \sqsubseteq \text{cond}(R_{i}^{\text{fail}}) \right)$

$\text{init} < \tau \land (u \sqsubseteq \text{msg}(\text{init}) \lor u \sqsubseteq \text{cond}(\text{init}))$
Some preprocessing

\[ T_{ij} \rightarrow R_i : \quad \forall n. \langle n, \mathcal{H}(n, k_j) \rangle \]

We can preprocess the euf-cma axiom to some extent by pre-instantiating it. For instance:

\[
\forall i, j. \left| \text{verify} \left( \text{sel2of2} \left( \text{input} \left( R_{ij}^{\text{succ}} \right) \right) \right), \text{sel1of2} \left( \text{input} \left( R_{ij}^{\text{succ}} \right) \right) \right|, k[j] \right|
\Rightarrow \exists k, j'. \left( T_{k, j'} < R_{ij}^{\text{succ}} \land j = j' \land |n_{T}[k, j']| = |\text{sel1of2} \left( \text{input} \left( R_{ij}^{\text{succ}} \right) \right)| \right)
\]
## Preliminary Results

<table>
<thead>
<tr>
<th>Protocol</th>
<th>cryptoVerif (ms)</th>
<th>us (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic-hash</td>
<td>51.8 ± 6.7</td>
<td>22.8 ± 1.9</td>
</tr>
<tr>
<td>feldhofer</td>
<td>59.4 ± 5.1</td>
<td>82.4 ± 19.4</td>
</tr>
<tr>
<td>hash-lock</td>
<td>60.6 ± 6.2</td>
<td>37.5 ± 3.3</td>
</tr>
<tr>
<td>lak-tag</td>
<td>60.4 ± 4.6</td>
<td>60.7 ± 10.9</td>
</tr>
<tr>
<td>mw</td>
<td>71.0 ± 7.9</td>
<td>59.3 ± 9.3</td>
</tr>
</tbody>
</table>