Reducing memory consumption of ProVerif

Vincent Cheval

Inria Paris
vincent.cheval@inria.fr

Master 2 internship of Margot Catinaud

GT FMS - Rump Session
28/03/2023
TLS with Encrypted Client Hello extension

Many features
- HelloRetryRequest
- Certificate-based Client Authentication
- Pre-Shared Keys and Tickets
- 0RTT
- Post Handshake Authentication
- Other TLS extensions (e.g. SNI)

Many security properties
- Server Authentication
- Client Authentication
- Key and Transcript Agreement
- Data Stream Integrity
- Key Uniqueness
- Downgrade Resilience
- Key Secrecy
- Key Indistinguishability
- 1RTT Data Forward Secrecy
- 0RTT Data Secrecy
- Client Identity Privacy
- Client Unlinkability
- Server Extension Privacy
- Client Extension Privacy
- Server Identity Privacy

<table>
<thead>
<tr>
<th>Status</th>
<th>Reachability</th>
<th>Equivalence</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verified</td>
<td>358</td>
<td>208</td>
<td>566</td>
<td>63 %</td>
</tr>
<tr>
<td>Stopped mostly due to OM (200-300GB)</td>
<td>230</td>
<td>87</td>
<td>317</td>
<td>36 %</td>
</tr>
<tr>
<td>Total</td>
<td>592</td>
<td>300</td>
<td>892</td>
<td></td>
</tr>
</tbody>
</table>
Main idea to reduce memory

1. Process
2. Translation into Horn clauses
3. Saturation of Horn clauses
4. Verification of query

Our solution: Hash consing and representation of terms in DAG

Required to modify several internal algorithms of ProVerif

Unification
Unification modulo equational theory
Matching
Subsumption of Horn Clauses
…
## Preliminary results

<table>
<thead>
<tr>
<th>Query</th>
<th>ProVerif 2.04</th>
<th>Prototype</th>
<th>Memory ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key secrecy &amp; Uniqueness</td>
<td>162 GB</td>
<td>6 GB</td>
<td>28.9</td>
</tr>
<tr>
<td>Authentication</td>
<td>141 GB</td>
<td>22 GB</td>
<td>6.4</td>
</tr>
<tr>
<td>Secrecy &amp; Authenticity</td>
<td>162 GB</td>
<td>2 GB</td>
<td>67.5</td>
</tr>
<tr>
<td>Forward secrecy &amp; Stream integrity</td>
<td>46 GB</td>
<td>11 GB</td>
<td>4.2</td>
</tr>
<tr>
<td>Post-handshake authentication</td>
<td>61 GB</td>
<td>39 GB</td>
<td>1.6</td>
</tr>
<tr>
<td>Key indistinguishability</td>
<td>34 GB</td>
<td>2 GB</td>
<td>18.9</td>
</tr>
</tbody>
</table>
Next steps?

What remains to be done?

- Generalise to all types of queries
- Adapt all the saturation transformation rules

Lot of implementation required

What remains to be done?

Release end of 2023?

Theoretical questions

Set $S$ of Horn Clauses

\[
F_1^1 \land \ldots \land F_{n_1}^1 \rightarrow C_1^1 \\
F_1^2 \land \ldots \land F_{n_2}^2 \rightarrow C_2^2 \\
\vdots \\
F_1^k \land \ldots \land F_{n_k}^k \rightarrow C_k^k
\]

How to find the best variable renaming that minimise the size (in DAG) of $S$?

How to find a variable renaming that approximate the minimal size (in DAG) of $S$?