SQUIRREL, an Interactive Prover for Protocol Verification in the Computational Model

October 28, 2020
Context

Security Protocols

**Distributed** programs which aim at providing some **security** properties.

Security Attacks

Attacks against security protocols can be very **damageable**, e.g. theft or privacy breach.

⇒ We need to check that protocols are secure:

  **formal methods** allow to do that, with strong guarantees.
Verification of Security Protocols

The goal is to completely rule-out classes of attacks.

\[ \forall A \in C, \quad (A \parallel P) \models \phi \]

Attacker Class

What is the class \( C \) of attackers?
Attacker Classes from the Literature

Adversary: any **arbitrary** function \( \mathcal{A} \) that satisfies some **axioms**, i.e. \( \mathcal{A} \models \text{Ax} \)
- Defined by what he **cannot** do, e.g.:
  \[ \{m_0\}_{pk} \sim \{m_1\}_{pk} \quad (\text{if } \text{len}(m_0) = \text{len}(m_1)) \]
- ✔ Strong security ✔ Good automation

Adversary: any **PPT** function \( \mathcal{A} \)
- models a real-world attacked.
- ✔ Strong security ✗ Limited automation

Adversary: **fixed** set of rules \( \mathcal{E} \)
- Defined by what he **can** do.
- E.g. \( \text{dec}(\{x\}_{pk}, \text{sk}) \rightarrow x \)
- ✗ Reduced security ✔ Good automation

---

† **Computationally Complete Symbolic Attacker**
[BC12], [CLCS14]

- First framework, **only for reachability properties**.
- A tool implementing a decision procedure, tested on a few protocols for a small number of sessions.

[BC14], [CK17]

- New framework, both for **reachability and equivalence** properties.
- Framework allows to do **manual** proofs, only for a **bounded number** of sessions.
Our Contributions (under submission, S&P).

Contribution 1:
A theoretical framework (a meta-logic) to express and prove security properties (reachability and equivalence) for an arbitrary number of sessions.

Contribution 2:
An interactive prover, SQUIRREL, to mechanize proofs.

Manual proofs, only for a bounded number of sessions.
Outline

1. The Base logic: CCSA
2. The Meta-Logic
3. An Interactive Theorem Prover, SQUIRREL
The Base logic: CCSA
Basic Hash Protocol

To formally model this protocol's security, we need to model:
- messages, i.e. distributions over bit-strings
- security properties (reachability or equivalence)

\[ \exists i, \pi_2(\text{input}) = h(\pi_1(\text{input}), k_i) \]

\[ \langle n, h(n, k) \rangle \]

\[ \text{ok()} \]

\[ \text{Otherwise} \]

\[ \text{ko()} \]
Basic Hash Protocol

To formally model this protocol’s security, we need to model:

- messages, i.e. distributions over bit-strings ⇒ terms
- security properties (reachability or equivalence) ⇒ formulas
We model protocol messages using terms built upon:

- **names** \( \mathcal{N} \), e.g. \( n_A, n_B \), for random samplings (including keys).
- **function symbols** \( \mathcal{F} \) for protocol functions, e.g.:
  
  \[
  h(_, _), \langle _, _ \rangle, \pi_i(_, ) \text{, ok()}, \text{if_then_else}_, \text{eq}(_, _)
  \]

- **function symbols** \( \mathcal{G} \) for adversarial computations, e.g.:
  
  \[
  \text{att}(_)
  \]

- **variables** \( x \in \mathcal{X} \).
Terms for Basic Hash with two tags (with keys $k_1, k_2$):

\[
\langle n, h(n, k_1) \rangle \quad \text{if } \pi_2(\text{input}) = h(\pi_1(\text{input}), k_1) \text{ then } \text{ok}() \\
\text{else if } \pi_2(\text{input}) = h(\pi_1(\text{input}), k_2) \text{ then } \text{ok}() \\
\text{else } \text{ko}() \]

where $\text{input} = \text{att}($frame$)$, and frame is the sequence of all messages sent over the network.
We model security properties using formulas, which are built using a predicate $\sim$ of arbitrary arity.

$$\phi ::= \phi \lor \phi \mid \neg \phi \mid \exists x, \phi \mid \vec{u} \sim \vec{v}$$

**Basic Hash Unlinkability (weak version)**

Weak unlinkability for two tags and two sessions:

$$\langle n, h(n, k_0) \rangle, \langle n', h(n', k_0) \rangle \sim \langle n, h(n, k_0) \rangle, \langle n', h(n', k_1) \rangle$$
Models of our logic are called computational models, where a computational model $\mathcal{M}$ interprets:

- **terms** as PPT Turing machines:
  - names as independent random samplings;
  - function symbols as deterministic machines.
- $\sim$ as **computational indistinguishability**.

**Validity**

We note $\mathcal{M} \models \phi$ when the base logic formula holds in the computational model $\mathcal{M}$.

A base logic formula $\phi$ is **valid** if $\mathcal{M} \models \phi$ for every $\mathcal{M}$. 
To prove that $\phi$ is valid, we axiomatize what the adversary cannot do. We restrict the models $\mathbb{M}$ using axioms $Ax$:

- structural axioms;
- implementation axioms, e.g. functional properties;
- cryptographic axioms, e.g. EUF-CMA, PRF.

Axioms are given as inference rules.

$$
\Delta_1 \vdash \phi_1 \ldots \Delta_1 \vdash \phi_1
$$

$$
\Delta \vdash \phi
$$
CCSA Approach: Axioms

\[ \vdash \phi \]

\[ \text{Ax sound (under crypto. assumptions)} \]

\[ \phi \text{ valid (under crypto. assumptions)} \]
**CCSA Approach: Examples of Axioms**

**DUP**

\[ \Delta \vdash \vec{u}, s \sim \vec{v}, t \]
\[ \Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t \]

**PRF**

- If \( \text{HFresh}^k(t; \vec{u}, t) \)
- Then \( n \sim \vec{v} \)
- Else \( h(t, k) \)
\[ \Delta \vdash \vec{u}, h(t, k) \sim \vec{v} \]

when \( \text{SC}^{n,k}(t, \vec{u}) \)
Weak unlinkability for two tags A, B with **three sessions**:

\[\sim\]
CCSA Approach: Limitations

Weak unlinkability for two tags $A, B$ with three sessions:

We have to manually prove all these equivalences!

- **Limited guarantees**: only proved for three sessions.
- Lots of redundant reasoning between cases.

Moreover, to prove security for a fixed $n$, e.g. 3, we often need to understand why security holds for any $n$. 
The Meta-Logic
**Goal:** prove security for any interleaving, in a single proof.
**Goal:** prove security for any interleaving, in a single proof.
Solution: a Meta-Logic

\[
\{ \phi_1, \phi_2, \phi_3, \phi_4, \ldots \} = \left\{ A_{i_1}, \ldots, A_{i_n} \mid \sim_{T_1, \ldots, T_n} \right\}
\]

for any \( n \in \mathbb{N} \) and

\[
(A_{i_j} \in \{\{\};\{\}\})_{1 \leq j \leq n}
\]
Solution: a Meta-Logic

\[
\left\{ \phi_1, \phi_2, \phi_3, \phi_4, \ldots \right\} = \left\{ A_{i_1}, \ldots, A_{i_n} \right\} \sim T_1, \ldots, T_n \text{ for any } n \in \mathbb{N} \text{ and } \left( A_{i_j} \in \{ \text{blue}; \text{green} \} \right)_{1 \leq j \leq n}
\]

\[
\psi = \forall \tau, \text{frame} @ \tau \sim \text{frame}' @ \tau
\]
Protocols as a Set of Actions

To do this, we need a **formal description of protocols**.

<table>
<thead>
<tr>
<th>An Action is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ a condition,</td>
</tr>
<tr>
<td>■ and an output message.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A Protocol is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ a finite set of actions,</td>
</tr>
<tr>
<td>■ equipped with a dependency relation to constrain the execution order of actions.</td>
</tr>
</tbody>
</table>
Protocols as a Set of Actions

Action $T(i, j)$: session $j$ of the tag $i$
Condition: true
Output: $\langle n[i, j], h(n[i, j], k[i]) \rangle$
Protocols as a Set of Actions

Action $R_{ok}[k]$: session $k$ of the reader, accept

Condition: $\exists i, \pi_2(input@R_{ok}[k]) = h(\pi_1(input@R_{ok}[k]), k[i])$

Output: $ok()$
Protocols as a Set of Actions

Action $R_{ko}[k]$: session $k$ of the reader, reject

Condition: $\neg (\exists i, \pi_2(\text{input} \circ R_{ko}[k]) = h(\pi_1(\text{input} \circ R_{ko}[k]), k[i]))$

Output: $ko()$
Protocols as a Set of Actions

**Bonus Action** $R[k]$: session $k$ of the reader, accept or reject

**Condition:** true

**Output:** find $i$ s.t. $\pi_2(\text{input} @ R[k]) = h(\pi_1(\text{input} @ R[k]), k[i])$ then ok() 
else ko()
Extension of the base logic with:

- **index variables** $I$, to represent session numbers, agent numbers, etc: $i_1, \ldots, i_n$
- **indexed names**, e.g. $n[i_1, \ldots, i_k]$
- **timestamps variables** $\tau$ and terms, to quantify over all possible instants of a trace: $\tau$ or $T(i, j)$.
- **macros** $\text{cond}@\tau$, $\text{input}@\tau$, $\text{output}@\tau$ to talk about the condition, input and output of the action at instant $\tau$
- **quantifications** over timestamps and indices.
A meta-formula $\psi$ represents a set of base-formulas. Roughly:

$$\psi \text{ represents } \left\{ (\psi)^T \mid \text{for any "trace" } T \right\}$$

**Trace Model** $T$

A trace model $T$ is a tuple $(\mathcal{D}_I, \mathcal{D}_T, <_T, \sigma_I, \sigma_T)$:

- $\mathcal{D}_I, \mathcal{D}_T$ are index and timestamp domains;
- $<_T$ is a total ordering on $\mathcal{D}_T$;
- $\sigma_I : I \rightarrow \mathcal{D}_I$ interprets index variables;
- $\sigma_T : T \rightarrow \mathcal{D}_T$ interprets timestamp variables.

Translation function $(_)^T$ from meta-formulas and terms to base-formulas and terms.

- $\mathcal{D}_I = \{1, 2, 3\}$.
- $\sigma_I = \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $\mathcal{D}_I = \{1, 2, 3\}$.
- $\sigma_I = \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(n[i, j])^\mathcal{T} := n_{3, 1}$
- $(n[i, j'])^\mathcal{T} := n_{3, 2}$
- $(\text{output@ } T[i, j])^\mathcal{T} := \langle n_{3, 1}, h(n_{3, 1}, k_3) \rangle$

- $\mathcal{D}_I = \{1, 2, 3\}$.
- $\sigma_I = \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(n[i, j])^T := n_{3,1}$
- $(n[i, j'])^T := n_{3,2}$
- $(\text{output}@T[i, j])^T := \langle n_{3,1}, h(n_{3,1}, k_3) \rangle$
- $(\text{cond}@R_{ok}[k])^T$
  
  $$:= (\exists i, \pi_2(\text{input}@R[k]) = h(\pi_1(\text{input}@R[k]), k[i]))^T$$
  $$:= \pi_2(\text{att}(...)) = h(\pi_1(\text{att}(...)), k_1)$$
  $$\lor \pi_2(\text{att}(...)) = h(\pi_1(\text{att}(...)), k_2)$$
  $$\lor \pi_2(\text{att}(...)) = h(\pi_1(\text{att}(...)), k_3)$$
Selected (simplified) rules:

\[(f(t_1, \ldots, t_n))^\mathcal{T} = f((t_1)^\mathcal{T}, \ldots, (t_n)^\mathcal{T})\]

\[(\exists i. \phi)^\mathcal{T} = \bigvee_{k \in \mathcal{D}_\mathcal{I}} (\phi)^\mathcal{T}_{i \mapsto k}\]

\[(\forall \tau. \phi)^\mathcal{T} = \bigwedge_{\nu \in \mathcal{D}_\mathcal{T}} (\phi)^\mathcal{T}_{\tau \mapsto \nu}\]
Meta-Logic: Translation to the Base Logic, Some Details

Selected (simplified) rules:

\[
(f(t_1, \ldots, t_n))^T = f((t_1)^T, \ldots, (t_n)^T)
\]

\[
(\exists i. \phi)^T = \bigvee_{k \in \mathcal{D}_I} (\phi)^T_{i \mapsto k}
\]

\[
(\forall \tau. \phi)^T = \bigwedge_{v \in \mathcal{D}_T} (\phi)^T_{\tau \mapsto v}
\]

and for macros:

\[
(output@\tau)^T = \text{specified by the protocol}
\]

\[
(cond@\tau)^T = \text{specified by the protocol}
\]

\[
(input@\tau)^T = \text{att}((frame@\text{pred}(\tau))^T)
\]

\[
(frame@\tau)^T \approx \langle (frame@\text{pred}(\tau))^T, (output@\tau)^T \rangle
\]
Validity: Meta-Logic

We note $T, M \models \psi$ when the meta-logic formula $\psi$ holds in trace model $T$ and in the computational model $M$:

$$T, M \models \psi \ \text{iff.} \ \ M \models (\psi)^T$$

A meta-logic formula $\psi$ is valid if $(\psi)^T$ is valid for every $T$. 
Meta-Logic: Lifting Axioms

Base logic rule

\[
\text{DUP} \quad \frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}
\]

Meta-logic rule

\[
\text{DUP} \quad \frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}
\]
Meta-Logic: Lifting Axioms

**Base logic rule**

\[
\text{PRF} \quad \text{if } \text{HFresh}^k(t; \vec{u}, t) \\
\Delta \vdash \vec{u}, \quad \text{then } n \sim \vec{v} \\
\text{else } h(t, k) \\
\Delta \vdash \vec{u}, h(t, k) \sim \vec{v}
\]

when \( SC^{n,k}(t, \vec{u}) \)

**Meta-logic rule**

\[
\text{PRF} \quad \text{if } \text{HFresh}_{\mathcal{P}}^k[i](t; \vec{u}, t) \\
\Delta \vdash \vec{u}, \quad \text{then } n \sim \vec{v} \\
\text{else } h(t, k[i]) \\
\Delta \vdash \vec{u}, h(t, k[i]) \sim \vec{v}
\]

when \( SC^\mathcal{P}_{n,k[i]}(t, \vec{u}) \)
Meta-Logic: Lifting Axioms

Base logic rule

\[
\text{PRF} \quad \begin{array}{l}
\text{if HFresh}^k(t; \bar{u}, t) \\
\Delta \vdash \bar{u}, \quad \text{then } n \\
\text{else } h(t, k)
\end{array}
\quad \begin{array}{l}
\sim \bar{v} \\
\Delta \vdash \bar{u}, h(t, k) \sim \bar{v}
\end{array}
\]

when \( \text{SC}^n,k(t, \bar{u}) \)

Meta-logic rule

\[
\text{PRF} \quad \begin{array}{l}
\text{if HFresh}_{\mathcal{P}}^{\bar{i}}(t; \bar{u}, t) \\
\Delta \vdash \bar{u}, \quad \text{then } n \\
\text{else } h(t, k[\bar{i}])
\end{array}
\quad \begin{array}{l}
\sim \bar{v} \\
\Delta \vdash \bar{u}, h(t, k[\bar{i}]) \sim \bar{v}
\end{array}
\]

when \( \text{SC}_{\mathcal{P}}^{n,k}[\bar{i}](t, \bar{u}) \)

HFresh^k(t; \bar{u}, t) and \( \text{SC}^n,k(t, \bar{u}) \)
can be checked/computed syntactically.

HFresh^k_{\mathcal{P}}(t; \bar{u}, t) and \( \text{SC}_{\mathcal{P}}^{n,k}[\bar{i}](t, \bar{u}) \)
need to be checked/computed for:
- **direct** occurrences (syntactically),
- and **indirect** occurrences (any action of the protocol).
Using the meta-logic inference rules, we are able to derive all at once a family of base logic formulas.

(Do you recall the long list of equivalences shown previously?)

It starts like this:

\[
\begin{align*}
\tau = T[i, j], \ldots &\vdash \ldots \\
\tau = R_{ok}[k], \ldots &\vdash \ldots \\
\tau = R_{ko}[k], \ldots &\vdash \ldots \\
\text{frame@}\text{pred}(\tau) &\sim \text{frame}^\text{u}@\text{pred}(\tau) \vdash \text{frame@}\tau \sim \text{frame}^\text{u}@\tau \\
\vdash \text{frame@}\tau &\sim \text{frame}^\text{u}@\tau
\end{align*}
\]

Ind
An Interactive Theorem Prover, SQUIRREL
The Tool

- The **input language** is a variant of the applied-pi calculus.
- We have implemented:
  - the **translation** of the specification of the protocol from this input language to actions,
  - **proof tactics**, corresponding to inference rules,
  - **automated reasoning** to ease the proof effort.
- The **user** interacts with the prover by **calling tactics** to derive formulas step by step.
Case Studies

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Crypto. assumptions</th>
<th>Security properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Hash</td>
<td>PRF, EUF-CMA</td>
<td>Authentication &amp; Unlinkability</td>
</tr>
<tr>
<td>Hash Lock</td>
<td>PRF, EUF-CMA</td>
<td>Authentication &amp; Unlinkability</td>
</tr>
<tr>
<td>LAK (pairs)</td>
<td>PRF, EUF-CMA</td>
<td>Authentication &amp; Unlinkability</td>
</tr>
<tr>
<td>MW</td>
<td>PRF, EUF-CMA, XOR</td>
<td>Authentication &amp; Unlinkability</td>
</tr>
<tr>
<td>Feldhofer</td>
<td>CCA₁, PRF, EUF-CMA</td>
<td>Authentication &amp; Unlinkability</td>
</tr>
<tr>
<td>Private Auth.</td>
<td>CCA₁, EUF-CMA, ENC-KP</td>
<td>Anonymity</td>
</tr>
<tr>
<td>Signed DDH</td>
<td>EUF-CMA, DDH</td>
<td>Authentication &amp; Strong Secrecy</td>
</tr>
</tbody>
</table>

Additional case studies, using the composition framework from [CJS20]

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Crypto. assumptions</th>
<th>Security properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed DDH</td>
<td>EUF-CMA, DDH</td>
<td>Authentication &amp; Strong Secrecy</td>
</tr>
<tr>
<td>SSH (fwd agent)</td>
<td>EUF-CMA, DDH</td>
<td>Authentication &amp; Strong Secrecy</td>
</tr>
</tbody>
</table>
[demo]
Our contribution

- **Meta-logic** built on the **CCSA model**.

- Set of **meta-logic inference rules** for proving reachability and equivalence properties.

- **SQUIRREL**, an interactive prover **implementing** these inference rules, used on various **case studies**.
Conclusion

Our contribution

- **Meta-logic** built on the **CCSA model**.
- Set of **meta-logic inference rules** for proving reachability and equivalence properties.
- **Squirrel**, an interactive prover implementing these inference rules, used on various **case studies**.

Current and future work

- Extend support to **stateful** and **more complex** protocols.
- More **proof automation**.
Conclusion

Our contribution

- **Meta-logic** built on the **CCSA model**.
- Set of **meta-logic inference rules** for proving reachability and equivalence properties.
- **Squirrel**, an interactive prover implementing these inference rules, used on various **case studies**.

Current and future work

- Extend support to **stateful** and **more complex** protocols.
- More **proof automation**.

Thank you for your attention!
References


