SQUIRREL, an interactive prover for protocol verification in the computational model


ANR TECAP - October 15, 2020
Security protocols

• A **protocol** is a set of rules detailing how entities interact:
  • contactless payment, HTTPS, access badges, etc.

• Security protocols must ensure some **security properties**:
  • authentication, anonymity, unlinkability, etc.

• How to have **guarantees** that protocols are secure?

  ⇒ **Use formal methods!**
Security protocols

• A protocol is a set of rules detailing how entities interact:
  • contactless payment, HTTPS, access badges, etc.

• Security protocols must ensure some security properties:
  • authentication, anonymity, unlinkability, etc.

• How to have guarantees that protocols are secure?

⇒ Use formal methods!
Formal verification of security protocols

Security protocols
- Specification

Is the goal satisfied?

Security goals
- Secrecy
- Authentication
- Unlinkability
- Privacy
- ...

Model
- Level of abstraction
- Attacker’s model

Formal methods and tools

Security properties
- Reachability
- Equivalence
Formal verification of security protocols

Security protocols
• Specification

Model
• Level of abstraction
• Attacker’s model

Security goals
• Secrecy
• Authentication
• Unlinkability
• Privacy
• ...

Is the goal satisfied?

Security properties
• Reachability
• Equivalence

Formal methods and tools
Formal verification of security protocols

Security protocols
- Specification

Is the goal satisfied?

Security goals
- Secrecy
- Authentication
- Unlinkability
- Privacy
- ...

Model
- Level of abstraction
- Attacker’s model

Security properties
- Reachability
- Equivalence

Formal methods and tools
Basic Hash protocol

- **Tag:**
  - $k_T$
  - new $n$

- **Reader:**
  - $\{k_i\}_i$

**Interaction:**

- $\langle n, H(n, k_T) \rangle$
- $\langle x_1, x_2 \rangle$

**Condition:**

- if $\exists k_i$, $x_2 = H(x_1, k_i)$
  - ok
- else
  - error
Different models: a big picture

<table>
<thead>
<tr>
<th>Model</th>
<th>Symbolic</th>
<th>CCSA*</th>
<th>Computational</th>
</tr>
</thead>
<tbody>
<tr>
<td>messages</td>
<td>terms</td>
<td>terms</td>
<td>bitstrings</td>
</tr>
<tr>
<td>crypto</td>
<td>perfect</td>
<td>implementation assumptions (e.g. EUF, PRF)</td>
<td>implementation assumptions (e.g. EUF, PRF)</td>
</tr>
<tr>
<td>attacker</td>
<td>can do</td>
<td>cannot do</td>
<td>cannot do</td>
</tr>
<tr>
<td>guarantees</td>
<td>weak (only logical attacks)</td>
<td>strong (all PPT** attacks)</td>
<td>strong (all PPT** attacks)</td>
</tr>
<tr>
<td>tools</td>
<td>well-suited for automation (e.g. ProVerif, Tamarin)</td>
<td>adapted for mechanization</td>
<td>limited support for automation (e.g. CryptoVerif, EasyCrypt)</td>
</tr>
</tbody>
</table>

* CCSA = Computationally Complete Symbolic Attacker
** PPT = Probabilistic Polynomial-Time
### Different models: a big picture

<table>
<thead>
<tr>
<th>Model</th>
<th>Symbolic</th>
<th>CCSA*</th>
<th>Computational</th>
</tr>
</thead>
<tbody>
<tr>
<td>messages</td>
<td>terms</td>
<td>terms</td>
<td>bitstrings</td>
</tr>
<tr>
<td>crypto</td>
<td>perfect (black-box)</td>
<td>implementation assumptions (e.g. EUF, PRF)</td>
<td>implementation assumptions (e.g. EUF, PRF)</td>
</tr>
<tr>
<td>attacker</td>
<td>can do</td>
<td>cannot do</td>
<td>cannot do</td>
</tr>
<tr>
<td>guarantees</td>
<td>weak (only logical attacks)</td>
<td>strong (all PPT** attacks)</td>
<td>strong (all PPT** attacks)</td>
</tr>
<tr>
<td>tools</td>
<td>well-suited for automation (e.g. ProVerif, Tamarin)</td>
<td>adapted for mechanization</td>
<td>limited support for automation (e.g. CryptoVerif, EasyCrypt)</td>
</tr>
</tbody>
</table>

* CCSA = Computationally Complete Symbolic Attacker
** PPT = Probabilistic Polynomial-Time
CCSA model: related work

[BC12], [CLCS14]

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

[BC14], [CK17], [Kou19]

- New model, both for reachability and equivalence properties.
- Manual proofs, only for a bounded number of sessions.
- Decidability result.
Our contribution

⇒ A theoretical framework, called *meta-logic*, to express and prove security properties (reachability and equivalence) for an arbitrary number of sessions.

⇒ An interactive prover, **SQuRREL**, to *mechanize proofs*.

1Paper under submission (Security and Privacy 2021).
Outline of the talk

- Base logic (CCSA model)
- Meta-logic
- An interactive prover, SQUIRREL
Base logic (CCSA model)
A logic built over terms and a predicate $\sim$, where:

- **terms** are interpreted as PPT Turing machines;
- $\sim$ is interpreted as **computational indistinguishability**;
- **names** are independent random samplings;
- **function symbols** correspond to deterministic machines.

### Validity

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

A base logic formula $\phi$ is **valid** if $\forall \mathcal{M}, \mathcal{M} \models \phi$. 
Basic Hash protocol

Base logic formula expressing a (light) notion of unlinkability.

∀M, M ⊨ ⟨n_0, H(n_0, key_0)⟩, ⟨n'_0, H(n'_0, key_0)⟩ ∼ ⟨n_1, H(n_1, key_1)⟩, ⟨n_2, H(n_2, key_2)⟩
Axioms as inference rules

In order to prove $\forall M, M \models \phi$, we use an axiomatic approach.

- We restrict the models $M$ we want to consider using axioms:
  - structural axioms,
  - implementation axioms (e.g. EUF, PRF).
- Axioms are given as inference rules allowing to derive formulas of the logic.
- If we can derive a security property $\phi$ using a set of inference rules that are computationally valid for our models $M$, then we conclude that $\forall M, M \models \phi$. 
Axioms as inference rules: two examples

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

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

when \( SC^\text{key}(t, \vec{u}) \)
Axioms as inference rules: two examples

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

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

when \( \text{SC}^{\text{key}}(t, \vec{u}) \)
Limitations of the CCSA model

Let’s say we want to prove (a light notion of) unlinkability for the Basic Hash protocol for 2 tags ($T_A$ and $T_B$) that can each play 2 sessions.

We would have to **manually prove all these equivalences!**

\[
\begin{align*}
  m_{T_A} &\sim m_{T_1} \\
  m_{T_B} &\sim m_{T_1} \\
  m_{T_A}, m'_{T_A} &\sim m_{T_1}, m_{T_2} \\
  m_{T_A}, m_{T_B} &\sim m_{T_1}, m_{T_2} \\
  m_{T_B}, m'_{T_B} &\sim m_{T_1}, m_{T_2} \\
  m_{T_A}, m_{T_B}, m'_{T_B} &\sim m_{T_1}, m_{T_2}, m_{T_3} \\
  m_{T_A}, m'_{T_A}, m_{T_B} &\sim m_{T_1}, m_{T_2}, m_{T_3} \\
  m_{T_A}, m'_{T_A}, m'_{T_B}, m_{T_B} &\sim m_{T_1}, m_{T_2}, m_{T_3}, m_{T_4}
\end{align*}
\]
Building a meta-logic on the base logic

\[
\begin{align*}
\{ \phi_1, \phi_2 \} + \{ \text{base logic inference rules} \} & \rightarrow \{ \text{proof}_1, \text{proof}_2 \} \\
\{ \phi_3, \phi_4 \} + \{ \text{base logic inference rules} \} & \rightarrow \{ \text{proof}_3, \text{proof}_4 \}
\end{align*}
\]
Building a meta-logic on the base logic

\[
\begin{array}{c}
\{ \phi_1, \phi_2 \} \\
\{ \phi_3, \phi_4 \} \\
\ldots
\end{array}
+ \left\{ \text{base logic inference rules} \right\}
\rightarrow
\begin{array}{c}
\{ \text{proof}_1, \text{proof}_2 \} \\
\{ \text{proof}_3, \text{proof}_4 \} \\
\ldots
\end{array}
\]
Building a meta-logic on the base logic

\[
\begin{align*}
\{ \phi_1, \phi_2 \} & \quad + \quad \{ \text{base logic inference rules} \} \\
\phi_3, \phi_4 & \quad \Rightarrow \\
\ldots & \\
\psi & \quad + \\
\text{proof} & \quad \Rightarrow \\
\{ \text{proof}_1, \text{proof}_2 \} \\
\text{proof}_3, \text{proof}_4 & \quad \ldots
\end{align*}
\]
Building a meta-logic on the base logic

\[ \{ \phi_1, \phi_2, \phi_3, \phi_4, \ldots \} \]

\[ \rightarrow \]

\[ \{ \text{base logic inference rules} \} \]

\[ \{ \text{proof}_1, \text{proof}_2, \text{proof}_3, \text{proof}_4, \ldots \} \]

\[ \{ \psi \} \]

\[ \rightarrow \]

\[ \{ \text{meta-logic inference rules} \} \]

\[ \rightarrow \]

\[ \text{proof} \]
Meta-logic
Protocols as a set of actions

<table>
<thead>
<tr>
<th>An action is defined by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• a condition,</td>
</tr>
<tr>
<td>• and an <strong>output</strong> message.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A protocol is defined by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• a finite set of <strong>actions</strong>,</td>
</tr>
<tr>
<td>• equipped with a <strong>dependency relation</strong> to constrain the execution order of actions.</td>
</tr>
</tbody>
</table>

| A trace is a sequence of actions. |
A meta-logic built on the CCSA model

Extension of the base logic with:

- **index variables**, used to parameterize names $n[i_1, \ldots, i_k]$;
- **timestamps variables** $\tau$, to quantify over all possible instants of a trace;
- **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.
Basic Hash protocol

\[
\begin{align*}
\text{cond@T}[i,j] & := \text{true} \\
\text{output@T}[i,j] & := \langle n[i,j], H(n[i,j], \text{key}[i]) \rangle
\end{align*}
\]

\[
\begin{align*}
\text{cond@R}[k] & := \exists i, \text{snd(input@R}[k]) = H(\text{fst(input@R}[k]), \text{key}[i])) \\
\text{output@R}[k] & := \text{ok}
\end{align*}
\]

\[
\begin{align*}
\text{cond@R1}[k] & := \neg (\exists i, \text{snd(input@R1}[k]) = H(\text{fst(input@R1}[k]), \text{key}[i])) \\
\text{output@R1}[k] & := \text{error}
\end{align*}
\]
For each possible trace of a protocol, we can give a meaning to meta-logic terms and formulas.

A trace model $\mathbb{T}$ 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 $\mathcal{D}_T$;
- $\sigma_I : I \rightarrow \mathcal{D}_I$ maps index variables;
- $\sigma_T : T \rightarrow \mathcal{D}_T$ maps timestamp variables.
**Trace model $\mathcal{T}$**

For each possible trace of a protocol, we can give a meaning to **meta-logic terms and formulas**.

A trace model $\mathcal{T}$ is a tuple $(\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}, <_{\mathcal{T}}, \sigma_{\mathcal{I}}, \sigma_{\mathcal{T}})$:

- $\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}$ are index and timestamp domains;
- $<_{\mathcal{T}}$ is a total ordering on $\mathcal{D}_{\mathcal{T}}$;
- $\sigma_{\mathcal{I}} : \mathcal{I} \rightarrow \mathcal{D}_{\mathcal{I}}$ maps index variables;
- $\sigma_{\mathcal{T}} : \mathcal{T} \rightarrow \mathcal{D}_{\mathcal{T}}$ maps timestamp variables.
Translation from the meta-logic to the base logic: example


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

- $(n[i, j])^T := n_{3, 1}$
- $(n[i, j'])^T := n_{3, 2}$
- $(\text{output}@T[i, j])^T := \langle n_{3, 1}, H(n_{3, 1}, \text{key}_3) \rangle$
- $(\text{cond}@R[k])^T$
  
  := $(\exists i, \text{snd}(\text{input}@R[k]) = H(\text{fst}(\text{input}@R[k]), \text{key}[i]))^T$
  := $\text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_1)$
  $\forall \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_2)$
  $\forall \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_3)$
Translation from the meta-logic to the base logic: example


- $\mathcal{D}_I := \{1, 2, 3\}$
- $\sigma_I := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$
- $(n[i, j])^T := n_{3, 1}$
- $(n[i, j'])^T := n_{3, 2}$
- $(\text{output}@T[i, j])^T := \langle n_{3, 1}, H(n_{3, 1}, \text{key}_3) \rangle$
- $(\text{cond}@R[k])^T$
  
  $$
  := (\exists i, \text{snd}(\text{input}@R[k]) = H(\text{fst}(\text{input}@R[k]), \text{key}[i]))^T
  := \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_1)
  \lor \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_2)
  \lor \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_3)
  $$
Translation from the meta-logic to the base logic: example


- \( D_I := \{1, 2, 3\} \)
- \( \sigma_I := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\} \)
- \((n[i, j])^T := n_{3,1}\)
- \((n[i, j'])^T := n_{3,2}\)
- \((\text{output}@T[i, j])^T := \langle n_{3,1}, H(n_{3,1}, \text{key}_3) \rangle\)
- \((\text{cond}@R[k])^T := \begin{cases} \exists i, \text{snd}(\text{input}@R[k]) = H(\text{fst}(\text{input}@R[k]), \text{key}[i])^T \\ \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_1) \\ \forall \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_2) \\ \forall \text{snd}(\text{att}(\ldots)) = H(\text{fst}(\text{att}(\ldots)), \text{key}_3) \end{cases} \)
Translation from the meta-logic to the base logic: example


- \( \mathcal{D}_T := \{1, 2, 3\} \)
- \( \sigma_T := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\} \)
- \( (n[i,j])^T := n_{3,1} \)
- \( (n[i,j'])^T := n_{3,2} \)
- \( (\text{output}@T[i,j])^T := \langle n_{3,1}, H(n_{3,1}, \text{key}_3) \rangle \)
- \( (\text{cond}@R[k])^T := (\exists i, \text{snd}(\text{input}@R[k]) = H(\text{fst}(\text{input}@R[k]), \text{key}[i]))^T \)
  \( := \text{snd}(\text{att}(...)) = H(\text{fst}(\text{att}(...)), \text{key}_1) \)
  \( \lor \text{snd}(\text{att}(...)) = H(\text{fst}(\text{att}(...)), \text{key}_2) \)
  \( \lor \text{snd}(\text{att}(...)) = H(\text{fst}(\text{att}(...)), \text{key}_3) \)
Reasoning with the meta-logic on protocols

(Quick reminder: in the base logic)

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

**A base logic formula $\phi$ is valid if $\forall \mathcal{M}, \mathcal{M} \models \phi$.**

In the meta-logic

We note $\mathcal{T}, \mathcal{M} \models \psi$ when the meta-logic formula $\psi$ is satisfied in the trace model $\mathcal{T}$ and in the computational model $\mathcal{M}$.

**A meta-logic formula $\psi$ is valid if $\forall \mathcal{T}, \forall \mathcal{M}, \mathcal{T}, \mathcal{M} \models \psi$.**
Reasoning with the meta-logic on protocols

(Quick reminder: in the base logic)

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

A base logic formula $\phi$ is valid if $\forall \mathcal{M}, \mathcal{M} \models \phi$.

In the meta-logic

We note $\mathcal{T}, \mathcal{M} \models \psi$ when the meta-logic formula $\psi$ is satisfied in the trace model $\mathcal{T}$ and in the computational model $\mathcal{M}$.

A meta-logic formula $\psi$ is valid if $\forall \mathcal{T}, \forall \mathcal{M}, \mathcal{T}, \mathcal{M} \models \psi$. 
Lifting axioms from the base logic to the meta-logic (1)

Base logic rule

\[
\text{DUP}
\]

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

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

Meta-logic rule

\[
\text{DUP}
\]

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

\[
\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t
\]
Lifting axioms from the base logic to the meta-logic (2)

Base logic rule

\[ \text{PRF} \]

\[ \Delta \vdash \vec{u}, \quad \begin{array}{l}
\text{if } \text{HFresh}^{\text{key}}(t; \vec{u}, t) \\
\text{then } n \\
\text{else } H(t, \text{key})
\end{array} \sim \vec{v} \\
\hline
\Delta \vdash \vec{u}, H(t, \text{key}) \sim \vec{v} \]

when \( \text{SC}^{\text{key}}(t, \vec{u}) \)

Meta-logic rule

\[ \text{PRF} \]

\[ \Delta \vdash \vec{u}, \quad \begin{array}{l}
\text{if } \text{HFresh}^{\text{key}[\vec{i}]}(t; \vec{u}, t) \\
\text{then } n \\
\text{else } H(t, \text{key}[\vec{i}])
\end{array} \sim \vec{v} \\
\hline
\Delta \vdash \vec{u}, H(t, \text{key}[\vec{i}]) \sim \vec{v} \]

when \( \text{SC}^{\text{key}[\vec{i}]}(t, \vec{u}) \)

\[ \text{HFresh}^{\text{key}}(t; \vec{u}, t) \text{ and } \text{SC}^{\text{key}}(t, \vec{u}) \text{ can be checked syntactically.} \]

\[ \text{HFresh}^{\text{key}[\vec{i}]}(t; \vec{u}, t) \text{ and } \text{SC}^{\text{key}[\vec{i}]}(t, \vec{u}) \text{ need to be checked for:} \]

- **direct** occurrences (syntactically),
- and **indirect** occurrences (any action of the protocol).
Lifting axioms from the base logic to the meta-logic (2)

**Base logic rule**

**PRF**

\[ \Delta \vdash \vec{u}, \text{ if } \text{HFresh}^\text{key}(t; \vec{u}, t) \]  
\[ \text{then } n \sim \vec{v} \]
\[ \text{else } H(t, \text{key}) \]

\[ \Delta \vdash \vec{u}, H(t, \text{key}) \sim \vec{v} \]

when \( \text{SC}^{\text{key}}(t, \vec{u}) \)

**Meta-logic rule**

**PRF**

\[ \Delta \vdash \vec{u}, \text{ if } \text{HFresh}^\text{key}[\vec{i}](t; \vec{u}, t) \]  
\[ \text{then } n \sim \vec{v} \]
\[ \text{else } H(t, \text{key}[\vec{i}]) \]

\[ \Delta \vdash \vec{u}, H(t, \text{key}[\vec{i}]) \sim \vec{v} \]

when \( \text{SC}^\text{key}[\vec{i}](t, \vec{u}) \)

HFresh\text{key}^P(t; \vec{u}, t) and SC\text{key}^P[\vec{i}](t, \vec{u}) can be checked syntactically. need to be checked 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:

\[
\begin{align*}
\tau &= T[i,j] \\
\tau &= R[k] \\
\tau &= R1[k]
\end{align*}
\]

\[
\text{frame@pred(}\tau\text{)} \vdash \text{frame@}\tau \quad \text{Induction}
\]

\[
\vdash \text{frame@}\tau
\]
An interactive prover, SQUIRREL
The tool

- Approximately 10,000 lines of OCaml code.
- 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]
| Signed DDH        | EUF-CMA, DDH                 | Authentication & Strong Secrecy |
| SSH (fwd agent)   | EUF-CMA, DDH                 | Authentication & Strong Secrecy |
Basic Hash protocol
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

- Deepen understanding of differences with EasyCrypt, CryptoVerif.
- Extend support to **stateful** and more complex protocols.

Thank you for your attention!
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

- Deepen understanding of differences with EasyCrypt, CryptoVerif.
- Extend support to stateful and more complex protocols.

Thank you for your attention!
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

- Deepen understanding of differences with EasyCrypt, CryptoVerif.
- Extend support to **stateful** and more complex protocols.

Thank you for your attention!


