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

D. Baelde, S. Delaune, C. Jacomme, A. Koutsos, **S. Moreau**

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?

$\Rightarrow$ **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?

$\Rightarrow$ **Use formal methods!**

Formal verification of security protocols

Security protocols

- Specification

Is the goal satisfied?

Security goals

- Secrecy
- Authentication
- Unlinkability
- Privacy
- ...

Model

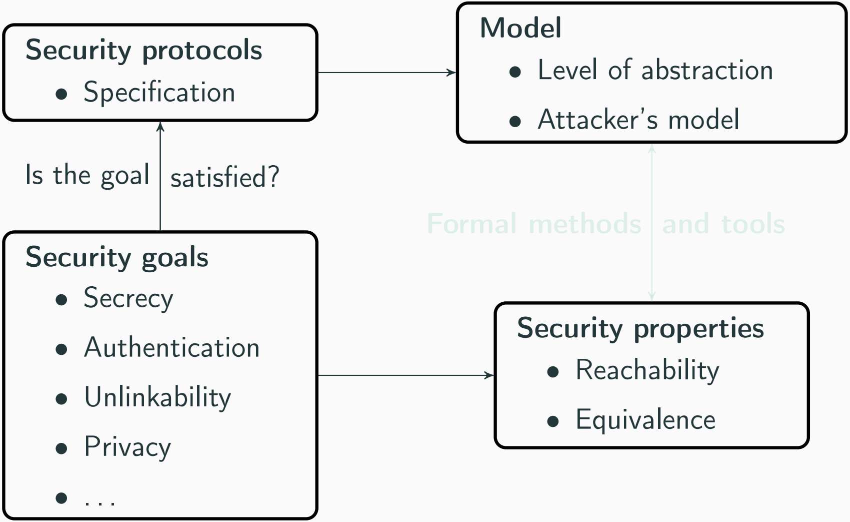
- Level of abstraction
- Attacker's model

Formal methods and tools

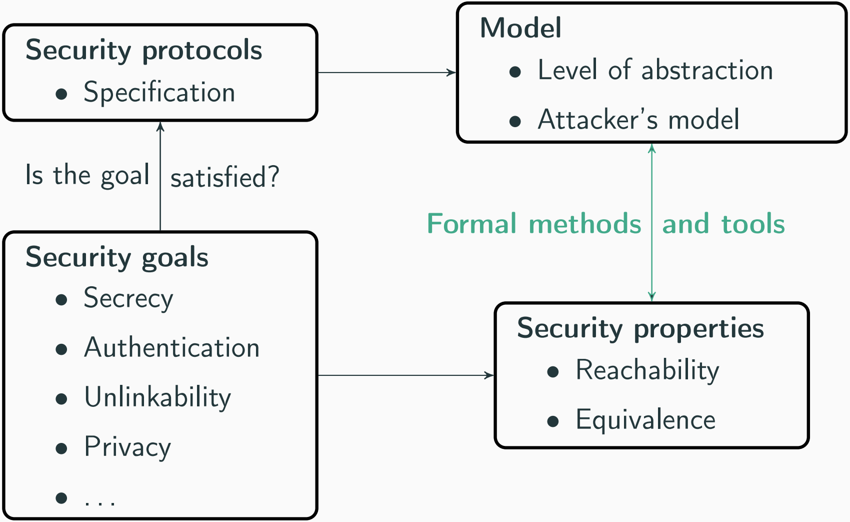
Security properties

- Reachability
- Equivalence

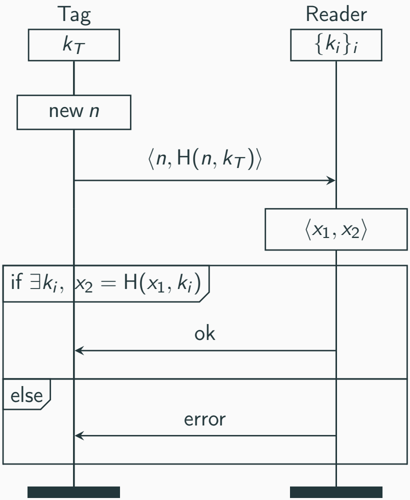
Formal verification of security protocols



Formal verification of security protocols



Basic Hash protocol



Different models: a big picture

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

* CCSA = Computationally Complete Symbolic Attacker
** PPT = Probabilistic Polynomial-Time

Different models: a big picture

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

* 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[1]

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

⇒ An interactive prover, Squirrel, to mechanize proofs.

[1] Paper under submission (Security and Privacy 2021).

Outline of the talk

Base logic (CCSA model)

Meta-logic

An interactive prover, SQUIRREL

Base logic (CCSA model)

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.

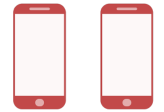
Validity

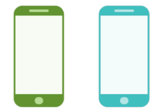
We note $\mathbb{M} \models \phi$ when the base logic formula is satisfied in the computational model $\mathbb{M}$.

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

Basic Hash protocol

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





$$\forall \mathbb{M}, \mathbb{M} \models \quad \langle \mathsf{n}_0, \mathsf{H}(\mathsf{n}_0, \mathsf{key}_0)\rangle, \langle \mathsf{n}'_0, \mathsf{H}(\mathsf{n}'_0, \mathsf{key}_0)\rangle \quad \sim \quad \langle \mathsf{n}_1, \mathsf{H}(\mathsf{n}_1, \mathsf{key}_1)\rangle, \langle \mathsf{n}_2, \mathsf{H}(\mathsf{n}_2, \mathsf{key}_2)\rangle$$

Axioms as inference rules

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

- We restrict the models $\mathbb{M}$ 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 ϕ using a set of inference rules that are computationally valid for our models $\mathbb{M}$, then we conclude that $\forall \mathbb{M}, \mathbb{M} \models \phi$.

Axioms as inference rules: two examples

DUP

$$\frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}$$

PRF

$$\frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}^{\mathsf{key}}(t; \vec{u}, t) \\ \text{then n} \\ \text{else } \mathsf{H}(t, \mathsf{key})\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}) \sim \vec{v}}$$

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

Axioms as inference rules: two examples

$$\text{DUP}\ \frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}$$

$$\text{PRF}\ \frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}^{\mathsf{key}}(t; \vec{u}, t)\\ \text{then } \mathsf{n}\\ \text{else } \mathsf{H}(t, \mathsf{key})\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}) \sim \vec{v}}$$

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

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!**

$$m_{T_A} \sim m_{T_\mathbf{1}}$$
$$m_{T_B} \sim m_{T_\mathbf{1}}$$

$$m_{T_A}, m'_{T_A} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}$$
$$m_{T_A}, m_{T_B} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}$$
$$m_{T_B}, m'_{T_B} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}$$

$$m_{T_A}, m'_{T_A}, m_{T_B} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}, m_{T_\mathbf{3}}$$
$$m_{T_A}, m_{T_B}, m'_{T_B} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}, m_{T_\mathbf{3}}$$

$$m_{T_A}, m'_{T_A}, m_{T_B}, m'_{T_B} \sim m_{T_\mathbf{1}}, m_{T_\mathbf{2}}, m_{T_\mathbf{3}}, m_{T_\mathbf{4}}$$

Building a meta-logic on the base logic

$$\begin{Bmatrix} \phi_1 & \phi_2 \\ \phi_3 & \phi_4 \\ \dots & \end{Bmatrix} \quad + \quad \begin{Bmatrix} \text{base logic} \\ \text{inference rules} \end{Bmatrix} \rightarrow \quad \begin{Bmatrix} \text{proof}_1 & \text{proof}_2 \\ \text{proof}_3 & \text{proof}_4 \\ \dots & \end{Bmatrix}$$

Building a meta-logic on the base logic

$$\left\{\begin{matrix}\phi_1 & \phi_2 \\ \phi_3 & \phi_4 \\ \dots & \end{matrix}\right\} + \left\{\begin{matrix}\text{base logic} \\ \text{inference rules}\end{matrix}\right\} \to \left\{\begin{matrix}\text{proof}_1 & \text{proof}_2 \\ \text{proof}_3 & \text{proof}_4 \\ \dots & \end{matrix}\right\}$$

Building a meta-logic on the base logic

$$\left\{\begin{matrix}\phi_1 & \phi_2\\ \phi_3 & \phi_4\\ \dots & \end{matrix}\right\} + \left\{\begin{matrix}\text{base logic}\\ \text{inference rules}\end{matrix}\right\} \rightarrow \left\{\begin{matrix}\text{proof}_1 & \text{proof}_2\\ \text{proof}_3 & \text{proof}_4\\ \dots & \end{matrix}\right\}$$

$$\psi \quad + \qquad\qquad \rightarrow \quad \text{proof}$$

Building a meta-logic on the base logic

$$\left\{\begin{matrix}\phi_1 & \phi_2 \\ \phi_3 & \phi_4 \\ \dots & \end{matrix}\right\} + \left\{\begin{matrix}\text{base logic} \\ \text{inference rules}\end{matrix}\right\} \to \left\{\begin{matrix}\text{proof}_1 & \text{proof}_2 \\ \text{proof}_3 & \text{proof}_4 \\ \dots & \end{matrix}\right\}$$

$$\psi + \left\{\begin{matrix}\text{meta-logic} \\ \text{inference rules}\end{matrix}\right\} \to \text{proof}$$

Meta-logic

Protocols as a set of actions

An action is defined by:

- a **condition**,
- and an **output** message.

A protocol is defined by:

- a finite set of **actions**,
- equiped with a **dependency relation** to constrain the execution order of actions.

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 $\mathsf{n}[i_1, \ldots. i_k]$;
- **timestamps variables** τ, to quantify over all possible instants of a trace;
- **macros** $\mathsf{cond}@\tau, \mathsf{input}@\tau, \mathsf{ouput}@\tau$ to talk about the condition, input and ouput of the action at instant τ;
- **quantifications** over timestamps and indices.

Basic Hash protocol

$$\begin{array}{lcl}
\text{cond@}\mathbf{T[i,j]} & := & \text{true} \\
\text{output@}\mathbf{T[i,j]} & := & \langle \mathsf{n}[i,j], \mathsf{H}(\mathsf{n}[i,j], \mathsf{key}[i]) \rangle
\end{array}$$

$$\begin{array}{lcl}
\text{cond@}\mathbf{R[k]} & := & \exists i, \mathsf{snd}(\mathsf{input@}R[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input@}R[k]), \mathsf{key}[i])) \\
\text{output@}\mathbf{R[k]} & := & \mathsf{ok}
\end{array}$$

$$\begin{array}{lcl}
\text{cond@}\mathbf{R1[k]} & := & \neg(\exists i, \mathsf{snd}(\mathsf{input@}R1[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input@}R1[k]), \mathsf{key}[i])) \\
\text{output@}\mathbf{R1[k]} & := & \mathsf{error}
\end{array}$$

Translation from the meta-logic to the base logic

Trace model $\mathbb{T}$

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 $(\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}, <_{\mathcal{T}}, \sigma_{\mathcal{I}}, \sigma_{\mathcal{T}})$:

- $\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}$ are index and timestamp domains;
- $<_{\mathcal{T}}$ is a total ordering on $\mathcal{D}_{\mathcal{T}}$;
- $\sigma_{\mathcal{I}} : \mathcal{I} \to D_{\mathcal{I}}$ maps index variables;
- $\sigma_{\mathcal{T}} : \mathcal{T} \to D_{\mathcal{T}}$ maps timestamp variables.

Translation from the meta-logic to the base logic

Trace model $\mathbb{T}$

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 $(\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}, <_{\mathcal{T}}, \sigma_{\mathcal{I}}, \sigma_{\mathcal{T}})$:

- $\mathcal{D}_{\mathcal{I}}, \mathcal{D}_{\mathcal{T}}$ are index and timestamp domains;
- $<_{\mathcal{T}}$ is a total ordering on $\mathcal{D}_{\mathcal{T}}$;
- $\sigma_{\mathcal{I}} : \mathcal{I} \to D_{\mathcal{I}}$ maps index variables;
- $\sigma_{\mathcal{T}} : \mathcal{T} \to D_{\mathcal{T}}$ maps timestamp variables.

Translation from the meta-logic to the base logic: example

Let's consider a trace of the Basic Hash protocol: $T[3,1].R[2].T[3,2]$.

- $\mathcal{D}_{\mathcal{I}} := \{1,2,3\}$
- $\sigma_{\mathcal{I}} := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(\mathsf{n}[i,j])^{\mathbb{T}} := \mathsf{n}_{3,1}$
- $(\mathsf{n}[i,j'])^{\mathbb{T}} := \mathsf{n}_{3,2}$
- $(\mathsf{output}@T[i,j])^{\mathbb{T}} := \langle \mathsf{n}_{3,1}, \mathsf{H}(\mathsf{n}_{3,1}, \mathsf{key}_3)\rangle$
- $(\mathsf{cond}@R[k])^{\mathbb{T}}$

$$
\begin{aligned}
&:= (\exists i, \mathsf{snd}(\mathsf{input}@R[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input}@R[k]), \mathsf{key}[i]))^{\mathbb{T}} \\
&:= \mathsf{snd}(\mathsf{att}(\ldots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\ldots)), \mathsf{key}_1) \\
&\quad\ \vee \mathsf{snd}(\mathsf{att}(\ldots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\ldots)), \mathsf{key}_2) \\
&\quad\ \vee \mathsf{snd}(\mathsf{att}(\ldots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\ldots)), \mathsf{key}_3)
\end{aligned}
$$

Translation from the meta-logic to the base logic: example

Let's consider a trace of the Basic Hash protocol: $T[3,1].R[2].T[3,2]$.

- $\mathcal{D}_{\mathcal{I}} := \{1, 2, 3\}$
- $\sigma_{\mathcal{I}} := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(\mathsf{n}[i,j])^{\mathbb{T}} := \mathsf{n}_{3,1}$
- $(\mathsf{n}[i,j'])^{\mathbb{T}} := \mathsf{n}_{3,2}$
- $(\mathsf{output}@T[i,j])^{\mathbb{T}} := \langle \mathsf{n}_{3,1}, \mathsf{H}(\mathsf{n}_{3,1}, \mathsf{key}_3)\rangle$
- $(\mathsf{cond}@R[k])^{\mathbb{T}}$

$$
\begin{aligned}
&:= (\exists i, \mathsf{snd}(\mathsf{input}@R[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input}@R[k]), \mathsf{key}[i]))^{\mathbb{T}} \\
&:= \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_1) \\
&\quad\ \vee \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_2) \\
&\quad\ \vee \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_3)
\end{aligned}
$$

Translation from the meta-logic to the base logic: example

Let's consider a trace of the Basic Hash protocol: $T[3,1].R[2].T[3,2]$.

- $\mathcal{D}_{\mathcal{I}} := \{1, 2, 3\}$
- $\sigma_{\mathcal{I}} := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(\mathsf{n}[i, j])^{\mathbb{T}} := \mathsf{n}_{3,1}$
- $(\mathsf{n}[i, j'])^{\mathbb{T}} := \mathsf{n}_{3,2}$
- $(\mathsf{output}@T[i, j])^{\mathbb{T}} := \langle \mathsf{n}_{3,1}, \mathsf{H}(\mathsf{n}_{3,1}, \mathsf{key}_3)\rangle$
- $(\mathsf{cond}@R[k])^{\mathbb{T}}$

$$\begin{aligned} &:= (\exists i, \mathsf{snd}(\mathsf{input}@R[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input}@R[k]), \mathsf{key}[i]))^{\mathbb{T}} \\ &:= \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_1) \\ &\qquad \vee \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_2) \\ &\qquad \vee \mathsf{snd}(\mathsf{att}(\dots)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(\dots)), \mathsf{key}_3) \end{aligned}$$

Translation from the meta-logic to the base logic: example

Let's consider a trace of the Basic Hash protocol: $T[3,1].R[2].T[3,2]$.

- $\mathcal{D}_{\mathcal{I}} := \{1,2,3\}$
- $\sigma_{\mathcal{I}} := \{i \mapsto 3, j \mapsto 1, j' \mapsto 2, k \mapsto 2\}$

- $(\mathsf{n}[i,j])^{\mathbb{T}} := \mathsf{n}_{3,1}$
- $(\mathsf{n}[i,j'])^{\mathbb{T}} := \mathsf{n}_{3,2}$
- $(\mathsf{output}@T[i,j])^{\mathbb{T}} := \langle \mathsf{n}_{3,1}, \mathsf{H}(\mathsf{n}_{3,1}, \mathsf{key}_3)\rangle$
- $(\mathsf{cond}@R[k])^{\mathbb{T}}$

$$\begin{aligned} &:= (\exists i, \mathsf{snd}(\mathsf{input}@R[k]) = \mathsf{H}(\mathsf{fst}(\mathsf{input}@R[k]), \mathsf{key}[i]))^{\mathbb{T}} \\ &:= \mathsf{snd}(\mathsf{att}(...)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(...)), \mathsf{key}_1) \\ &\qquad \vee\ \mathsf{snd}(\mathsf{att}(...)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(...)), \mathsf{key}_2) \\ &\qquad \vee\ \mathsf{snd}(\mathsf{att}(...)) = \mathsf{H}(\mathsf{fst}(\mathsf{att}(...)), \mathsf{key}_3) \end{aligned}$$

Reasoning with the meta-logic on protocols

(Quick reminder: in the base logic)

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

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

In the meta-logic

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

A meta-logic formula ψ is valid if $\forall \mathbb{T}, \forall \mathbb{M}, \mathbb{T}, \mathbb{M} \models \psi$.

Reasoning with the meta-logic on protocols

(Quick reminder: in the base logic)

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

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

In the meta-logic

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

A meta-logic formula ψ is valid if $\forall \mathbb{T}, \forall \mathbb{M}, \mathbb{T}, \mathbb{M} \models \psi$.

Lifting axioms from the base logic to the meta-logic (1)

Base logic rule

DUP
$$\frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}$$

Meta-logic rule

DUP
$$\frac{\Delta \vdash \vec{u}, s \sim \vec{v}, t}{\Delta \vdash \vec{u}, s, s \sim \vec{v}, t, t}$$

Lifting axioms from the base logic to the meta-logic (2)

Base logic rule

PRF

$$\frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}^{\mathsf{key}}(t; \vec{u}, t) \\ \text{then } \mathsf{n} \\ \text{else } \mathsf{H}(t, \mathsf{key})\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}) \sim \vec{v}}$$

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

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

Meta-logic rule

PRF

$$\frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t; \vec{u}, t) \\ \text{then } \mathsf{n} \\ \text{else } \mathsf{H}(t, \mathsf{key}[\vec{i}])\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}[\vec{i}]) \sim \vec{v}}$$

when $\mathsf{SC}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t, \vec{u})$

$\mathsf{HFresh}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t; \vec{u}, t)$ and $\mathsf{SC}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t, \vec{u})$ 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

$$\frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}^{\mathsf{key}}(t; \vec{u}, t) \\ \text{then n} \\ \text{else } \mathsf{H}(t, \mathsf{key})\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}) \sim \vec{v}}$$

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

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

Meta-logic rule

PRF

$$\frac{\Delta \vdash \vec{u}, \begin{array}{l}\text{if } \mathsf{HFresh}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t; \vec{u}, t) \\ \text{then n} \\ \text{else } \mathsf{H}(t, \mathsf{key}[\vec{i}])\end{array} \sim \vec{v}}{\Delta \vdash \vec{u}, \mathsf{H}(t, \mathsf{key}[\vec{i}]) \sim \vec{v}}$$

when $\mathsf{SC}_{\mathcal{P}}^{\mathsf{key}[\vec{i}]}(t, \vec{u})$

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

- **direct** occurrences (syntactically),
- and **indirect** occurrences (any action of the protocol).

Basic Hash 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:

$$\dfrac{\dfrac{\dfrac{\cdots}{\tau = T[i,j]} \qquad \dfrac{\cdots}{\tau = R[k]} \qquad \dfrac{\cdots}{\tau = R1[k]}}{\mathsf{frame@pred}(\tau) \vdash \mathsf{frame@}\tau}}{\vdash \mathsf{frame@}\tau}\ \text{Induction}$$

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

Protocol	Crypto. assumptions	Security properties
Basic Hash	PRF, EUF-CMA	Authentication & Unlinkability
Hash Lock	PRF, EUF-CMA	Authentication & Unlinkability
LAK (pairs)	PRF, EUF-CMA	Authentication & Unlinkability
MW	PRF, EUF-CMA, XOR	Authentication & Unlinkability
Feldhofer	CCA_1, PRF, EUF-CMA	Authentication & Unlinkability
Private Auth.	CCA_1, EUF-CMA, ENC-KP	Anonymity
Signed DDH	EUF-CMA, DDH	Authentication & Strong Secrecy
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

[demo]

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!

References

Gergei Bana and Hubert Comon-Lundh. "Towards Unconditional Soundness: Computationally Complete Symbolic Attacker". In: *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*. Ed. by Pierpaolo Degano and Joshua D. Guttman. Vol. 7215. Lecture Notes in Computer Science. Springer, 2012, pp. 189–208.

Gergei Bana and Hubert Comon-Lundh. "A Computationally Complete Symbolic Attacker for Equivalence Properties". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 609–620.

Hubert Comon, Charlie Jacomme, and Guillaume Scerri. *Oracle simulation: a technique for protocol composition with long term shared secrets*. Research Report. INRIA ; LSV, ENS Paris Saclay, Université Paris-Saclay ; Université Versailles Saint-Quentin, Aug. 2020. url: `https://hal.inria.fr/hal-02913866`.

Hubert Comon and Adrien Koutsos. “Formal Computational Unlinkability Proofs of RFID Protocols”. In: *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*. IEEE Computer Society, 2017, pp. 100–114.

Hubert Comon-Lundh, Véronique Cortier, and Guillaume Scerri. “A tool for automating the computationally complete symbolic attacker (Extended Abstract)”. In: *Joint Workshop on Foundations of Computer Security and Formal and Computational Cryptography (FCS-FCC’14)*. Vienne, Austria, July 2014. url: `https://hal.inria.fr/hal-01080296`.

Adrien Koutsos. “Decidability of a Sound Set of Inference Rules for Computational Indistinguishability”. In: *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*. IEEE, 2019, pp. 48–61.