Formal Proofs of Crypto Protocols with Squirrel

David Baelde & Adrien Koutsos

ENS Rennes, IRISA, Inria Paris

What is Squirrel?

A proof assistant for
verifying cryptographic protocols,
based on the CCSA approach.

Bana & Comon. *A Computationally Complete Symbolic Attacker for Equivalence Properties.* CCS 2014.

Team

David Baelde, Stéphanie Delaune, Caroline Fontaine,
Clément Hérouard, Charlie Jacomme, Adrien Koutsos,
Joseph Lallemand, Solène Moreau, Tito Nguyen

(IRISA, LMF, Inria Paris, CISPA)

This talk

An informal introduction to the Squirrel system:

- How to formally model protocols and reason about their properties.
- Preparing the ground for hands-on learning!

This talk

An informal introduction to the Squirrel system:

- How to formally model protocols and reason about their properties.
- Preparing the ground for hands-on learning!

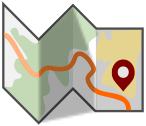
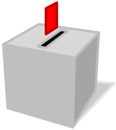
I'm not going to talk about the theory, open problems, related works. . .

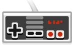
Outline

Security & Privacy

Increasingly many activities are becoming digitalized.

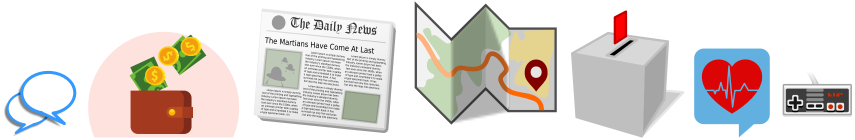




Security & Privacy

Increasingly many activities are becoming digitalized.

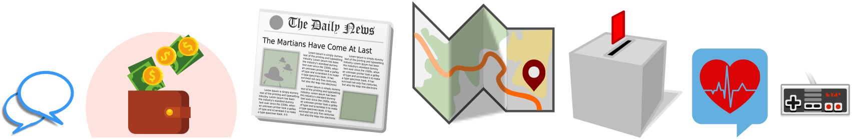


These systems must ensure important properties:

- security: secrecy, authenticity, no double-spending...
- privacy: anonymity, absence of tracking...

Security & Privacy

Increasingly many activities are becoming digitalized.



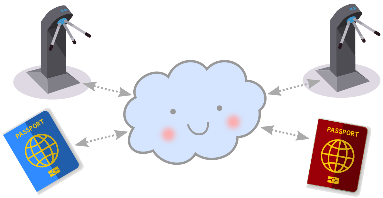
These systems must ensure important properties:

- security: secrecy, authenticity, no double-spending...
- privacy: anonymity, absence of tracking...

Frequent flaws at the hardware, software and specification levels can be discovered (and avoided) by using formal methods.

We focus on the analysis of protocols at the specification level.

Cryptographic protocols: a naive example



Each tag (T_i) owns a secret key k_i.

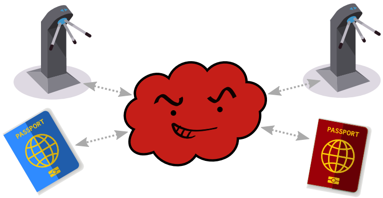
Reader (R) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \; n_R \\ T_i & \rightarrow & R & : \; \mathsf{h}(\mathsf{n}_R, \mathsf{k}_i) \\ R & \rightarrow & T_i & : \; \mathsf{ok} \end{array}$$

Scenario under consideration:

- roles R, T_1, ..., T_n; arbitrary number of sessions for each role

Cryptographic protocols: a naive example



Each tag (T_i) owns a secret key k_i.

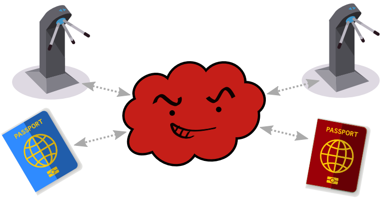
Reader (R) knows all legitimate keys.

$$\begin{array}{lcccl} R & \rightarrow & T_i & : & n_R \\ T_i & \rightarrow & R & : & \mathsf{h}(\mathsf{n}_R, \mathsf{k}_i) \\ R & \rightarrow & T_i & : & \mathsf{ok} \end{array}$$

Scenario under consideration:

- roles R, T_1, ..., T_n; arbitrary number of sessions for each role
- attacker can intercept messages, inject new messages

Cryptographic protocols: a naive example



Each tag (T_i) owns a secret key k_i.

Reader (R) knows all legitimate keys.

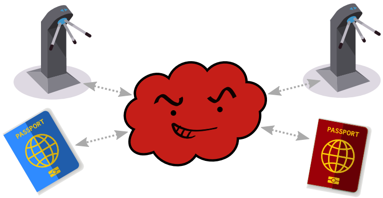
$$
\begin{array}{lcccl}
R & \rightarrow & T_i & : & n_R \\
T_i & \rightarrow & R & : & \mathsf{h}(\mathsf{n}_R, \mathsf{k}_i) \\
R & \rightarrow & T_i & : & \mathsf{ok}
\end{array}
$$

Scenario under consideration:

- roles R, T_1, ..., T_n; arbitrary number of sessions for each role
- attacker can intercept messages, inject new messages

Readers correctly authenticate tags.

Cryptographic protocols: a naive example



Each tag (T_i) owns a secret key k_i.

Reader (R) knows all legitimate keys.

$$
\begin{array}{lcccl}
R & \rightarrow & T_i & : & n_R \\
T_i & \rightarrow & R & : & \mathsf{h}(\mathsf{n}_R, \mathsf{k}_i) \\
R & \rightarrow & T_i & : & \mathsf{ok}
\end{array}
$$

Scenario under consideration:

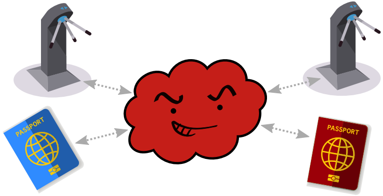
- roles R, T_1, ..., T_n; arbitrary number of sessions for each role
- attacker can intercept messages, inject new messages

Readers correctly authenticate tags.

Tags can be tracked: the protocol is not unlinkable.

- The attacker can obtain a pseudonym $\mathsf{h}(0, k_i)$ from T_i.

Running example: the Basic Hash protocol



Each tag (T_i) owns a secret key k_i.

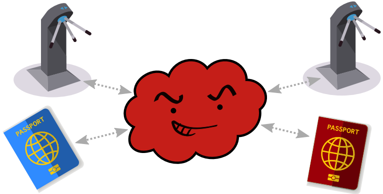
Reader (R) knows all legitimate keys.

$$\begin{array}{cccl} T_i & \rightarrow & R & : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, \mathsf{k}_i)\rangle \\ R & \rightarrow & T_i & : \mathsf{ok} \end{array}$$

Security properties:

- Authentication: readers must accept only legitimate inputs.
- Unlinkability: it must not be possible to track tags.

Running example: the Basic Hash protocol



Each tag (T_i) owns a secret key k_i.

Reader (R) knows all legitimate keys.

$$\begin{array}{lcl}
T_i & \rightarrow & R & : & \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, \mathsf{k}_i) \rangle \\
R & \rightarrow & T_i & : & \mathsf{ok}
\end{array}$$

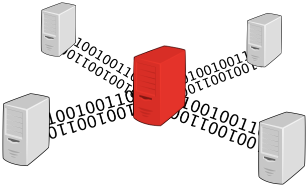
Security properties:

- Authentication: readers must accept only legitimate inputs.
- Unlinkability: it must not be possible to track tags.

Both properties hold... in a sense that needs to be made precise!

Computational model

The cryptographer's mathematical model for provable security



Messages = bitstrings

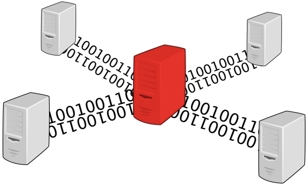
Secrets = random samplings

Participants = PPTIME Turing machines

+ assumptions on what cannot be achieved

Computational model

The cryptographer's mathematical model for provable security



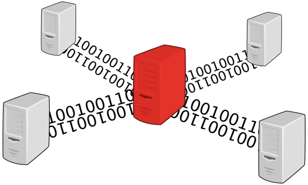
Messages	$=$	bitstrings
Secrets	$=$	random samplings
Participants	$=$	PPTIME Turing machines

$+$ assumptions on what cannot be achieved

The probability of an attack is negligible in the security parameter $\eta \in \mathbb{N}$ when it is asymptotically smaller than any η^{-k}.

Computational model

The cryptographer's mathematical model for provable security



Messages	$=$	bitstrings
Secrets	$=$	random samplings
Participants	$=$	PPTIME Turing machines

$+$ assumptions on what cannot be achieved

The probability of an attack is negligible in the security parameter $\eta \in \mathbb{N}$ when it is asymptotically smaller than any η^{-k}.

Definition (Unforgeability, EUF-CMA)

There is a negligible probability of success for the following game, for any attacker $\mathcal{A}$:

- Draw $k \in \{0,1\}^\eta$ uniformly at random.
- $\langle u, v\rangle := \mathcal{A}^{\mathcal{O}}$ where $\mathcal{O}$ is the oracle $x \mapsto \mathsf{h}(x, k)$.
- Succeed if $u = \mathsf{h}(v, k)$ and $\mathcal{O}$ has not been called on v.

Basic Hash in the computational model $\boxed{T_i \to R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i)\rangle}$

Authentication

Attacker can interact with tags and readers,
wins if some reader accepts a message that has not been emitted by a tag.

Basic Hash in the computational model $\boxed{T_i \to R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle}$

Authentication

Attacker can interact with tags and readers,
wins if some reader accepts a message that has not been emitted by a tag.

Example (Basic Hash, when h is unforgeable)

Assume reader accepts some m: $\mathsf{snd}(m) = \mathsf{h}(\mathsf{fst}(m), k_i)$ for some i.
By unforgeability, $\mathsf{fst}(m) = \mathsf{n}_T$ for some session of tag T_i.
The two projections of m are the two projections of the output of T_i:
authentication holds.

Basic Hash in the computational model $\boxed{T_i \rightarrow R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle}$

Privacy (simple scenario)

Attacker interacts with either T_1, T_2 or T_1, T_1
wins if he guesses in which situation he is.

Basic Hash in the computational model $\boxed{T_i \to R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle}$

Privacy (simple scenario)

Attacker interacts with either T_1, T_2 or T_1, T_1
wins if he guesses in which situation he is.

Definition (Pseudo-randomness, PRF)

The success probability for the following game is negligibly different from $\frac{1}{2}$:

- Draw $k_1, \ldots, k_n$ uniformly at random. Flip a coin b.
- Consider oracles $\mathcal{O}_i(x) = (\text{if } b \text{ then } \mathsf{h}(x, k_i) \text{ else random}())$ that can only be queried once per message.
- Succeed if $b = \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_n}$.

Basic Hash in the computational model $\boxed{T_i \to R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i)\rangle}$

Privacy (simple scenario)

Attacker interacts with either T_1, T_2 or T_1, T_1
wins if he guesses in which situation he is.

Definition (Pseudo-randomness, PRF)

The success probability for the following game is negligibly different from $\frac{1}{2}$:

- Draw $k_1, \ldots, k_n$ uniformly at random. Flip a coin b.
- Consider oracles $\mathcal{O}_i(x) = (\text{if } b \text{ then } \mathsf{h}(x, k_i) \text{ else random}())$ that can only be queried once per message.
- Succeed if $b = \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_n}$.

Example (Basic Hash, when h is pseudo-random)

Since tag nonces n_T are unlikely to collide, the second projections of tag outputs are indistinguishable from random samplings: privacy holds.

Comparison with related tools

	Akiss	DeepSec	Proverif	Tamarin	Scary	**Squirrel**	CryptoVerif	EasyCrypt
unbounded traces			✓	✓		✓	✓	✓
computational attacker					✓	✓	✓	✓
concrete security bounds							✓	✓
native concurrency	✓	✓	✓	✓	✓	✓	✓	
global mutable states	✓	✓	✓	✓	✓	✓		✓
automation	$\uparrow$	$\uparrow$	$\nearrow$	$\nearrow$	$\uparrow$	$\searrow$	$\nearrow$	$\downarrow$

- Squirrel only provides asymptotic guarantees *for each trace*.
- Automation is subjective. Differences in reasoning style are clearer.
- Squirrel is less mature than any of these tools.
  We have not verified anything like TLS, Signal or even Dolev-Yao!

Publications & case studies

- Baelde, Delaune, Jacomme, Koutsos & Moreau. *An Interactive Prover for Protocol Verification in the Computational Model.* S&P 2021.
- Jacomme, Scerri, Comon. *Oracle simulation: a technique for protocol composition with long term shared secrets.* CCS 2020.
- Baelde, Delaune, Koutsos & Moreau. *Cracking the Stateful Nut.* CSF 2022.
- Cremers, Fontaine & Jacomme. *A Logic and an Interactive Prover for the Computational Post-Quantum Security of Protocols.* S&P 2022.

Case studies

- Privacy and unlinkability properties of various protocols e.g. RFID.
- Parts of SSH protocol, YubiKey & YubiHSM.
- Post-quantum key exchanges.

Outline

1. Background: verifying security protocols
2. Reasoning about messages
   - Terms
   - Local formulas
   - Global formulas
3. Reasoning about protocols
4. Conclusion

Modelling messages

In our logic,

terms denote probabilistic polynomial-time computations of bitstrings.

We use terms to model messages computed by the protocol or attacker.

Honest functions symbols

Function symbols interpreted as deterministic computations,
used to represent primitives, public constants...
Notation: $\mathsf{f}(m)$, $\mathsf{g}(m, n)$, ok...
We assume builtin with a fixed, standard semantics: equals, ifthenelse, etc.

Example

- $\mathsf{if}\ u = v\ \mathsf{then}\ (\mathsf{if}\ v = u\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2)\ \mathsf{else}\ t_3$ and $\mathsf{if}\ u = v\ \mathsf{then}\ t_1\ \mathsf{else}\ t_3$ always compute the same thing.

Modelling messages

In our logic,

terms denote probabilistic polynomial-time computations of bitstrings.

We use terms to model messages computed by the protocol or attacker.

Honest functions symbols

Function symbols interpreted as deterministic computations,
used to represent primitives, public constants...
Notation: $\mathsf{f}(m)$, $\mathsf{g}(m, n)$, ok...
We assume builtin with a fixed, standard semantics: equals, ifthenelse, etc.

Example

- $\mathsf{if}\ u = v\ \mathsf{then}\ (\mathsf{if}\ v = u\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2)\ \mathsf{else}\ t_3$ and $\mathsf{if}\ u = v\ \mathsf{then}\ t_1\ \mathsf{else}\ t_3$ always compute the same thing.
- $\mathsf{enc}(u, v)$ might denote the symmetric encryption of a message u with some key v but v should not be a constant key (deterministic).

Modelling messages: names

Names

Interpreted as independent, uniform random samplings of length η.
Notation: n, r, k...

Names are used to model nonces, encryption randomization, etc.

Example

- When n and m are distinct name symbols,
  there is a negligible probability that m and n yield the same result.

Modelling messages: names

Names

Interpreted as independent, uniform random samplings of length η.
Notation: n, r, k...

Names are used to model nonces, encryption randomization, etc.

Example

- When n and m are distinct name symbols,
  there is a negligible probability that m and n yield the same result.
- There is a negligible probability that $t = \mathsf{n}$ returns true,
  provided that t represents a computation that cannot use n.
  - $\rightsquigarrow$ This is guaranteed if t does not contain n nor variables.
    Variables $x, y, z \ldots$ represent arbitrary probabilistic computations.

Modelling messages: names

Names

Interpreted as independent, uniform random samplings of length η.
Notation: n, r, k...

Names are used to model nonces, encryption randomization, etc.

Example

- When n and m are distinct name symbols,
  there is a negligible probability that m and n yield the same result.
- There is a negligible probability that $t = \mathsf{n}$ returns true,
  provided that t represents a computation that cannot use n.
  - $\leadsto$ This is guaranteed if t does not contain n nor variables.
    Variables $x, y, z \ldots$ represent arbitrary probabilistic computations.
- If h is interpreted as a hash function,
  there is a negligible probability that $\mathsf{h}(u, \mathsf{k}) = \mathsf{h}(v, \mathsf{k}) \wedge u \neq v \ldots$
  provided that u and v denote computations which cannot use k.

Modelling messages: adversarial function symbols

Adversarial function symbols

Function symbols used to represent attacker computations.
Interpreted as probabilistic computations that cannot access names.
Notation: $\mathbf{att}(m_1, \ldots, m_k)$.

Example (Modelling a trace of Basic Hash)

$$
\begin{array}{ccccccc}
T_i & \rightarrow & A & & : & \mathsf{out}_1 & = & \langle \mathsf{nT}, \mathsf{h}(\mathsf{nT}, \mathsf{k}_i) \rangle \\
 & & A & \rightarrow & R : & \mathsf{in}_2 & = & \mathbf{att}(\mathsf{out}_1) \\
 & & A & \leftarrow & R : & \mathsf{out}_2 & = & \mathsf{if} \ \ldots \ \mathsf{then\ ok\ else\ ko} \\
 & & A & \rightarrow & R : & \mathsf{in}_3 & = & \mathbf{att}'(\mathsf{out}_1, \mathsf{out}_2)
\end{array}
$$

Modelling messages: adversarial function symbols

Adversarial function symbols

Function symbols used to represent attacker computations.
Interpreted as probabilistic computations that cannot access names.
Notation: $\mathbf{att}(m_1, \dots, m_k)$.

Example (Modelling a trace of Basic Hash)

$$
\begin{array}{ccccccl}
T_i & \rightarrow & A & & : & \mathrm{out}_1 & = \langle \mathrm{nT}, \mathrm{h}(\mathrm{nT}, \mathrm{k}_i) \rangle \\
 & & A & \rightarrow R & : & \mathrm{in}_2 & = \mathbf{att}(\mathrm{out}_1) \\
 & & A & \leftarrow R & : & \mathrm{out}_2 & = \text{if } \dots \text{ then ok else ko} \\
 & & A & \rightarrow R & : & \mathrm{in}_3 & = \mathbf{att}'(\mathrm{out}_1, \mathrm{out}_2)
\end{array}
$$

Example (where h is interpreted as a hash function)

- There is a negligible probability that $\mathrm{h}(u, \mathrm{k}) = \mathrm{h}(v, \mathrm{k}) \wedge u \neq v \dots$ if u, v do not contain k nor variables (nothing to add for **att** symbols).

Modelling messages: adversarial function symbols

Adversarial function symbols

Function symbols used to represent attacker computations.
Interpreted as probabilistic computations that cannot access names.
Notation: $\mathbf{att}(m_1, \ldots, m_k)$.

Example (Modelling a trace of Basic Hash)

$$
\begin{array}{cccclcl}
T_i & \rightarrow & A & & : & \mathsf{out}_1 & = & \langle \mathsf{nT}, \mathsf{h}(\mathsf{nT}, \mathsf{k}_i) \rangle \\
 & & A & \rightarrow & R & : & \mathsf{in}_2 & = & \mathbf{att}(\mathsf{out}_1) \\
 & & A & \leftarrow & R & : & \mathsf{out}_2 & = & \text{if } \ldots \text{ then ok else ko} \\
 & & A & \rightarrow & R & : & \mathsf{in}_3 & = & \mathbf{att}'(\mathsf{out}_1, \mathsf{out}_2)
\end{array}
$$

Example (where h is interpreted as a hash function)

- There is a negligible probability that $\mathsf{h}(u, \mathsf{k}) = \mathsf{h}(v, \mathsf{k}) \wedge u \neq v \ldots$ if u, v do not contain k nor variables (nothing to add for **att** symbols).
- There is a negligible probability that $\mathbf{att}(\mathsf{h}(\mathsf{true}, \mathsf{k})) = \mathsf{h}(\mathsf{false}, \mathsf{k})$.

Local formulas over messages

Syntax

First-order formulas over message equalities without quantifiers.

Local formulas are also terms, i.e. probabilistic computations of a boolean. We've seen several examples already!

Example (Simple Basic Hash trace)

Message produced by attacker:	$\mathsf{in}_2 := \mathbf{att}(\langle \mathsf{nT}, \mathsf{h}(\mathsf{nT}, \mathsf{k}_i)\rangle)$
Reader accepts as coming from T_j:	$\mathsf{snd}(\mathsf{in}_2) = \mathsf{h}(\mathsf{fst}(\mathsf{in}_2), \mathsf{k}_j)$

Local formulas over messages

Syntax

First-order formulas over message equalities without quantifiers.

Local formulas are also terms, i.e. probabilistic computations of a boolean. We've seen several examples already!

Example (Simple Basic Hash trace)

Message produced by attacker: $\mathsf{in}_2 := \mathbf{att}(\langle \mathsf{nT}, \mathsf{h}(\mathsf{nT}, \mathsf{k}_i)\rangle)$

Reader accepts as coming from T_j: $\mathsf{snd}(\mathsf{in}_2) = \mathsf{h}(\mathsf{fst}(\mathsf{in}_2), \mathsf{k}_j)$

Semantics

A formula is valid when it is true with overwhelming probability

- for any interpretation of primitives that satisfies the declared crypto assumptions,
- for any interpretation of attacker computations.

Local formulas over messages and indices

Introduce terms of sort index. Index terms can only be variables $i, j, k \ldots$

Syntax

First-order formulas over message and index equalities
with quantifiers over indices.

Example (Simple Basic Hash trace)

Message produced by attacker:	$\mathsf{in}_2 := \mathbf{att}(\langle \mathsf{nT}, \mathsf{h}(\mathsf{nT}, \mathsf{k}(i))\rangle)$
Reader accepts as coming from some T_j:	$\exists j.\ \mathsf{snd}(\mathsf{in}_2) = \mathsf{h}(\mathsf{fst}(\mathsf{in}_2), \mathsf{k}(j))$

Semantics

A formula is valid when it is true with overwhelming probability

- for any interpretation of primitives satisfying crypto assumptions,
- for any interpretation of attacker computations,
- for any interpretation of indices in some finite set.

(Local formulas are still probabilistic computations of a boolean.)

Crypto axioms

We need to translate cryptographic assumptions into logical axioms[1].

Example (Collision resistance)

The following axiom is not valid when h is interpreted as a collision-resistant keyed hash function:

$$\mathsf{h}(x, \mathsf{k}) = \mathsf{h}(y, \mathsf{k}) \Rightarrow x = y$$

[1] This slides over-simplifies things: think of an indexed key, or an occurrence of a hash with variables bound by quantifiers.

Crypto axioms

We need to translate cryptographic assumptions into logical axioms[1].

Example (Collision resistance)

The following axiom is not valid when h is interpreted as a collision-resistant keyed hash function:

$$\mathsf{h}(x, \mathsf{k}) = \mathsf{h}(y, \mathsf{k}) \Rightarrow x = y$$

It is valid when x, y are closed terms where k only occurs as $\mathsf{h}(\_, \mathsf{k})$.

[1] This slides over-simplifies things: think of an indexed key, or an occurrence of a hash with variables bound by quantifiers.

Crypto axioms

We need to translate cryptographic assumptions into logical axioms[1].

Example (Collision resistance)

The following axiom is not valid when h is interpreted as a collision-resistant keyed hash function:

$$\mathsf{h}(x, \mathsf{k}) = \mathsf{h}(y, \mathsf{k}) \Rightarrow x = y$$

It is valid when x, y are closed terms where k only occurs as $\mathsf{h}(\_, \mathsf{k})$.

Example (Unforgeability)

Axiom scheme that is valid in all models where h satisfies EUF-CMA:

$$u = \mathsf{h}(v, \mathsf{k}) \Rightarrow \bigvee_{s \in S} s = v$$

where $S = \{\ s \mid \mathsf{h}(s, \mathsf{k}) \text{ occurs in } u, v\}$
and u, v are closed terms only containing k as $\mathsf{h}(\_, \mathsf{k})$.

[1] This slides over-simplifies things: think of an indexed key, or an occurrence of a hash with variables bound by quantifiers.

Sequents

In Squirrel we prove formulas by organizing them in *sequents*:

$$\phi_1, \ldots, \phi_n \vdash \psi \qquad \text{reads as} \qquad (\textstyle\bigwedge_i \phi_i) \Rightarrow \psi$$

The concrete notation is as follows, with identifiers for hypotheses:

```
H_1 : phi_1
...
H_n : phi_n
--------------
psi
```

Example (Unforgeability)

Under the same assumptions as before,
we can reduce the goal $\phi_1, \ldots, \phi_n, u = \mathsf{h}(v, \mathsf{k}) \vdash \psi$
to the collection of subgoals $\phi_1, \ldots, \phi_n, s = v \vdash \psi$ for all $s \in S$.

Demo

Let's see it in action on a naive and painful example:

`basic-hash-two.sp`

Global formulas over messages and indices

Syntax

First-order formulas Φ over the following atoms:

- $[\phi]$: "local formula ϕ is almost always true"
- $[\vec{u} \sim \vec{v}]$: "$\vec{u}$ and $\vec{v}$ are indistinguishable"

where $\vec{u}, \vec{v}$ are sequences of messages of same length

Quantifications allowed over indices *and messages*.

We use $\wedge$, $\Rightarrow$, $\forall$, ... to distinguish from local formulas.
As before, valid = true in all interpretations.

Example (Valid global formulas)

- $\forall x, y, z, x', y', z'.\ [x, y, z \sim x', y', z'] \Rightarrow [x', z', y' \sim x, z, y]$
- $\forall x, y.\ [x = y] \wedge [\vec{u}[x] \sim \vec{v}[x]] \Rightarrow [\vec{u}[y] \sim \vec{v}[y]]$
- $[\phi \sim \mathsf{true}] \Leftrightarrow [\phi]$

Global formula examples

Example (Equality vs. indistinguishability)

$\left[x = y\right] \Rightarrow \left[x \sim y\right]$ but not the converse:

Global formula examples

Example (Equality vs. indistinguishability)

$\left[x = y\right] \Rightarrow \left[x \sim y\right]$ but not the converse: $\left[\mathsf{n} \sim \mathsf{m}\right]$ but $\left[\mathsf{n} \neq \mathsf{m}\right]$

Global formula examples

Example (Equality vs. indistinguishability)

$[x = y] \Rightarrow [x \sim y]$ but not the converse: $[\mathsf{n} \sim \mathsf{m}]$ but $[\mathsf{n} \neq \mathsf{m}]$

Example (Relating local and global connectives)

- $[\phi \wedge \psi] \overset{?}{\Leftrightarrow} ([\phi] \wedge [\psi])$
- $[\phi \vee \psi] \overset{?}{\Leftrightarrow} ([\phi] \vee [\psi])$
- $[\phi \Rightarrow \psi] \overset{?}{\Leftrightarrow} ([\phi] \Rightarrow [\psi])$

Global formula examples

Example (Equality vs. indistinguishability)

$[x = y] \Rightarrow [x \sim y]$ but not the converse: $[\mathsf{n} \sim \mathsf{m}]$ but $[\mathsf{n} \neq \mathsf{m}]$

Example (Relating local and global connectives)

- $[\phi \wedge \psi] \Leftrightarrow ([\phi] \wedge [\psi])$
- $[\phi \vee \psi] \overset{?}{\Leftrightarrow} ([\phi] \vee [\psi])$
- $[\phi \Rightarrow \psi] \overset{?}{\Leftrightarrow} ([\phi] \Rightarrow [\psi])$

Global formula examples

Example (Equality vs. indistinguishability)

$\left[x = y\right] \Rightarrow \left[x \sim y\right]$ but not the converse: $\left[\mathsf{n} \sim \mathsf{m}\right]$ but $\left[\mathsf{n} \neq \mathsf{m}\right]$

Example (Relating local and global connectives)

- $\left[\phi \wedge \psi\right] \Leftrightarrow (\left[\phi\right] \wedge \left[\psi\right])$
- $\left[\phi \vee \psi\right] \Leftarrow (\left[\phi\right] \vee \left[\psi\right])$
- $\left[\phi \Rightarrow \psi\right] \overset{?}{\Leftrightarrow} (\left[\phi\right] \Rightarrow \left[\psi\right])$

Global formula examples

Example (Equality vs. indistinguishability)

$[x = y] \Rightarrow [x \sim y]$ but not the converse: $[\mathsf{n} \sim \mathsf{m}]$ but $[\mathsf{n} \neq \mathsf{m}]$

Example (Relating local and global connectives)

- $[\phi \wedge \psi] \Leftrightarrow ([\phi] \wedge [\psi])$
- $[\phi \vee \psi] \Leftarrow ([\phi] \vee [\psi])$
- $[\phi \Rightarrow \psi] \Rightarrow ([\phi] \Rightarrow [\psi])$

Axioms on equivalence

Example (Freshness)

$\left[\vec{u} \sim \vec{v}\right] \Rightarrow \left[\vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{m}\right]$ valid when $\vec{u}, \vec{v}$ do not contain variables nor n, m.

Example (Function application)

- $\left[\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2}\right] \Rightarrow \left[\vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})\right]$

Axioms on equivalence

Example (Freshness)

$\left[\vec{u} \sim \vec{v}\right] \Rightarrow \left[\vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{m}\right]$ valid when $\vec{u}, \vec{v}$ do not contain variables nor n, m.

Example (Function application)

- $\left[\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2}\right] \Rightarrow \left[\vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})\right]$
- More generally, we have $\left[\vec{u} \sim \vec{v}\right] \Rightarrow \left[\vec{u'} \sim \vec{v'}\right]$ when $\vec{u'}$ and $\vec{v'}$ can be computed in the same way from $\vec{u}$ and $\vec{v}$.

Axioms on equivalence

Example (Freshness)

$[\vec{u} \sim \vec{v}] \Rightarrow [\vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{m}]$ valid when $\vec{u}, \vec{v}$ do not contain variables nor n, m.

Example (Function application)

- $[\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2}] \Rightarrow [\vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})]$
- More generally, we have $[\vec{u} \sim \vec{v}] \Rightarrow [\vec{u'} \sim \vec{v'}]$ when $\vec{u'}$ and $\vec{v'}$ can be computed in the same way from $\vec{u}$ and $\vec{v}$.

Example (Pseudo-randomness)

Axiom scheme that holds in all models where h satisfies PRF:

$$[\vec{v}, \mathsf{h}(t, \mathsf{k}) \quad \sim \quad \vec{v}, \mathsf{if} \ \vee_{s \in S} \, s = t \ \mathsf{then} \ \mathsf{h}(t, \mathsf{k}) \ \mathsf{else} \ \mathsf{n}]$$

where S is the set of hashes in $\vec{v}, t$,
n is fresh and $\vec{v}, t$ are closed terms only containing k as $\mathsf{h}(\_, \mathsf{k})$.

In Squirrel

Let's go back to our naive and painful example:

`basic-hash-two.sp`

Since equivalences are often between terms with many similarities, we write $\left[\vec{u} \sim \vec{v}\right]$ as `equiv(diff(u1,v1),..,diff(uN,vN))` where `diff` operators can be pushed inside terms to only be used where the left and right versions differ.

When proving an equivalence, we only display the list of bi-terms:

```
... (* Hypotheses *)
----------------------
0: diff(u1,v1)
1: diff(u2,v2)
2: diff(u3,v3)
```

Outline

Modelling protocols

A protocol is modelled by a set of actions.
Each action is identified by an indexed action symbol $\mathsf{A}(\vec{i})$.

The semantics of action $\mathsf{A}(\vec{i})$ is given by:

- a local formula describing the executability condition;
- a message term describing its output.

Both can use a special message term $\mathsf{input}@\mathsf{A}(\vec{i})$.

Example (Basic Hash)

Action $\mathsf{T}(i, k)$ for session k of T_i:

- Executes if true.
- Outputs $\langle \mathsf{nT}(i, k), \mathsf{h}(\mathsf{nT}(i, k), \mathsf{k}(i))\rangle$.

Modelling protocols

A protocol is modelled by a set of actions.
Each action is identified by an indexed action symbol $\mathsf{A}(\vec{i})$.

The semantics of action $\mathsf{A}(\vec{i})$ is given by:

- a local formula describing the executability condition;
- a message term describing its output.

Both can use a special message term $\mathsf{input}@\mathsf{A}(\vec{i})$.

Example (Basic Hash)

Action $\mathsf{R}(j, i)$ when reader session j recognizes a message from tag i:

- Executes if $\mathsf{snd}(\mathsf{input}@\mathsf{R}(j,i)) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\mathsf{R}(j,i)), \mathsf{k}(i))$
- Outputs ok.

Modelling protocols

A protocol is modelled by a set of actions.
Each action is identified by an indexed action symbol $\mathsf{A}(\vec{i})$.

The semantics of action $\mathsf{A}(\vec{i})$ is given by:

- a local formula describing the executability condition;
- a message term describing its output.

Both can use a special message term $\mathsf{input}@\mathsf{A}(\vec{i})$.

Example (Basic Hash)

Action $\mathsf{R}_1(j)$ when reader session j rejects its input:

- Executes if $\forall i.\ \mathsf{snd}(\mathsf{input}@\mathsf{R}(j,i)) \neq \mathsf{h}(\mathsf{fst}(\mathsf{input}@\mathsf{R}(j,i)), \mathsf{k}(i))$.
- Outputs ko.

Full local meta-logic

A new sort of terms, to represent points in an abstract execution trace.

Terms of sort timestamp

$T ::= \tau \mid \mathsf{pred}(T) \mid \mathsf{A}(\vec{i})$ $\qquad$ τ variable, $\mathsf{A} \in \mathcal{A}$ action symbol

Local formulas over all sorts

Enrich syntax with:

- Quantification over timestamps.
- Atoms over timestamps: $T = T'$, $T \leq T'$, $\mathsf{happens}(T)$.
- Macros $\mathsf{input}@T$, $\mathsf{output}@T$, $\mathsf{cond}@T$, etc.

Semantics:

- Meaning of macros defined wrt. a system of actions.
- Timestamps interpreted in arbitrary trace + undefined timestamp.

Local meta-logic formulas: examples

Example

$\exists j, i.\ \ \mathsf{happens}(\mathsf{R}(j, i))$ says that some R action is schedule in the trace:

- It is true for trace $\mathsf{init}.\mathsf{T}(12, 27).\mathsf{R}(42, 13).\mathsf{R}_1(99)$.
- It is false for trace $\mathsf{init}.\mathsf{T}(12, 27).\mathsf{R}_1(99)$.

Local meta-logic formulas: examples

Example

$\exists j, i.\ \ \mathsf{happens}(\mathsf{R}(j, i))$ says that some R action is schedule in the trace:

- It is true for trace $\mathsf{init}.\mathsf{T}(12, 27).\mathsf{R}(42, 13).\mathsf{R}_1(99)$.
- It is false for trace $\mathsf{init}.\mathsf{T}(12, 27).\mathsf{R}_1(99)$.

Example (Mutual exclusion for R and R_1 in a session)

$\forall j, i.\ \ \neg(\mathsf{happens}(\mathsf{R}(j, i)) \wedge \mathsf{happens}(\mathsf{R}_1(j)))$

- It is not valid: not satisfied in trace $\mathsf{init}.\mathsf{T}(12, 27).\mathsf{R}(42, 13).\mathsf{R}_1(42)$.
- Reasonable axiom: the two scheduled actions wouldn't execute.

Local meta-logic formulas: examples

Example

$\exists j, i.\ \ \mathsf{happens}(\mathsf{R}(j,i))$ says that some R action is schedule in the trace:

- It is true for trace $\mathsf{init}.\mathsf{T}(12,27).\mathsf{R}(42,13).\mathsf{R}_1(99)$.
- It is false for trace $\mathsf{init}.\mathsf{T}(12,27).\mathsf{R}_1(99)$.

Example (Mutual exclusion for R and R_1 in a session)

$\forall j, i.\ \ \neg(\mathsf{happens}(\mathsf{R}(j,i)) \wedge \mathsf{happens}(\mathsf{R}_1(j)))$

- It is not valid: not satisfied in trace $\mathsf{init}.\mathsf{T}(12,27).\mathsf{R}(42,13).\mathsf{R}_1(42)$.
- Reasonable axiom: the two scheduled actions wouldn't execute.

Example (Authentication for Basic Hash)

$$\begin{aligned}\forall j, i.\ \ \mathsf{cond}@\mathsf{R}(j,i) \Rightarrow \exists k.\ \ & \mathsf{T}(i,k) < \mathsf{R}(j,i) \wedge \\ & \mathsf{fst}(\mathsf{input}@\mathsf{R}(j,i)) = \mathsf{fst}(\mathsf{output}@\mathsf{T}(i,k)) \wedge \\ & \mathsf{snd}(\mathsf{input}@\mathsf{R}(j,i)) = \mathsf{snd}(\mathsf{output}@\mathsf{T}(i,k))\end{aligned}$$

Axioms of trace models

Injectivity (part of `auto` tactic)

For any two actions $\mathsf{A}, \mathsf{B} \in \mathcal{A}$:

- $\forall \vec{i}.\ \ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{B}(\vec{j})) \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{B}(\vec{j})$
- $\forall \vec{i}.\ \ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{A}(\vec{j})) \wedge \vec{i} \neq \vec{j} \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{A}(\vec{j})$

Axioms of trace models

Injectivity (part of `auto` tactic)

For any two actions $\mathsf{A}, \mathsf{B} \in \mathcal{A}$:

- $\forall \vec{i}.\ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{B}(\vec{j})) \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{B}(\vec{j})$
- $\forall \vec{i}.\ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{A}(\vec{j})) \wedge \vec{i} \neq \vec{j} \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{A}(\vec{j})$

Order (part of `auto` tactic)

- $\mathsf{happens}(\tau) \wedge \mathsf{happens}(\tau') \Leftrightarrow \tau \leq \tau' \vee \tau' \leq \tau$
- $\mathsf{happens}(\mathsf{pred}(\tau)) \Rightarrow \mathsf{pred}(\tau) < \tau.$

Axioms of trace models

Injectivity (part of `auto` tactic)

For any two actions $\mathsf{A}, \mathsf{B} \in \mathcal{A}$:

- $\forall \vec{i}.\ \ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{B}(\vec{j})) \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{B}(\vec{j})$
- $\forall \vec{i}.\ \ \forall \vec{j}.\ \mathsf{happens}(\mathsf{A}(\vec{i})) \wedge \mathsf{happens}(\mathsf{A}(\vec{j})) \wedge \vec{i} \neq \vec{j} \Rightarrow \mathsf{A}(\vec{i}) \neq \mathsf{A}(\vec{j})$

Order (part of `auto` tactic)

- $\mathsf{happens}(\tau) \wedge \mathsf{happens}(\tau') \Leftrightarrow \tau \leq \tau' \vee \tau' \leq \tau$
- $\mathsf{happens}(\mathsf{pred}(\tau)) \Rightarrow \mathsf{pred}(\tau) < \tau.$

Case analysis and induction (`case` and `induction`)

- $\forall \tau.\ \ \mathsf{happens}(\tau) \Rightarrow \tau = \mathsf{init} \vee \bigvee_{\mathsf{A} \in \mathcal{A}} \exists \vec{i}.\ \ \tau = A(\vec{i})$
- $(\forall\ \tau.\ (\forall \tau'.\ \ \tau' < \tau \Rightarrow\ \ \phi[\tau']) \Rightarrow \phi[\tau]) \Rightarrow \forall \tau.\ \ \phi[\tau]$

Occurrences in terms with macros

Adding macros complicates axioms that come with occurrence constraints.

Example (Freshness without macros nor indices)

$t \neq \mathsf{n}$ is valid for any term t that does not contain message variables nor n.

Occurrences in terms with macros

Adding macros complicates axioms that come with occurrence constraints.

Example (Freshness without macros nor indices)

$t \neq \mathsf{n}$ is valid for any term t that does not contain message variables nor n.

Example (Freshness without macros)

$t = \mathsf{n}(\vec{i}) \Rightarrow \bigvee_{\mathsf{n}(\vec{j}) \in t} \vec{i} = \vec{j}$ valid for any term t without message variables.

Occurrences in terms with macros

Adding macros complicates axioms that come with occurrence constraints.

Example (Freshness without macros nor indices)

$t \neq \mathsf{n}$ is valid for any term t that does not contain message variables nor n.

Example (Freshness without macros)

$t = \mathsf{n}(\vec{i}) \Rightarrow \bigvee_{\mathsf{n}(\vec{j}) \in t} \vec{i} = \vec{j}$ valid for any term t without message variables.

Example (Freshness with macros)

$t = \mathsf{n}(\vec{i}) \quad \Rightarrow \quad \bigvee_{\mathsf{n}(\vec{j}) \in t} \vec{i} = \vec{j} \quad \vee \quad \bigvee_{\mathsf{n}(\vec{j}) \in \mathsf{A}(\vec{k})} \exists \vec{k}.\ \mathsf{A}(\vec{k}) \leq T \wedge \vec{i} = \vec{j}$

valid for any term t without message variables and only macros@T...

Occurrences in terms with macros

Adding macros complicates axioms that come with occurrence constraints.

Example (Freshness without macros nor indices)

$t \neq \mathsf{n}$ is valid for any term t that does not contain message variables nor n.

Example (Freshness without macros)

$t = \mathsf{n}(\vec{i}) \Rightarrow \bigvee_{\mathsf{n}(\vec{j}) \in t} \vec{i} = \vec{j}$ valid for any term t without message variables.

Example (Freshness with macros)

$$t = \mathsf{n}(\vec{i}) \quad \Rightarrow \quad \bigvee_{\mathsf{n}(\vec{j}) \in t} \vec{i} = \vec{j} \quad \vee \quad \bigvee_{\mathsf{n}(\vec{j}) \in \mathsf{A}(\vec{k})} \exists \vec{k}.\ \mathsf{A}(\vec{k}) \leq T \wedge \vec{i} = \vec{j}$$

valid for any term t without message variables and only macros@T...
assuming that indices of $\mathsf{n}(\_)$ are not bound by inner quantifications...

Further refinements are possible, and even desirable.

Basic Hash

We can now nicely model and reason about Basic Hash!

`basic-hash-wa.sp`

Observe that the user does not specify the actions directly.
They are compiled from a description of the system using process algebra.

The full story

Sequential dependencies

Actions are actually equipped with a partial order expressing dependencies. $\mathsf{A}(\vec{i}) \prec \mathsf{B}(\vec{i}, \vec{j})$ imposes that

> in all traces, any instance of $\mathsf{B}(\vec{i}, \vec{j})$ must be preceded by $\mathsf{A}(\vec{i})$.

The output and condition of $\mathsf{B}(\vec{i}, \vec{j})$ can mention $\mathsf{input}@\mathsf{A}(\vec{i})$.

The full story

Sequential dependencies

Actions are actually equipped with a partial order expressing dependencies. $\mathsf{A}(\vec{i}) \prec \mathsf{B}(\vec{i}, \vec{j})$ imposes that

in all traces, any instance of $\mathsf{B}(\vec{i}, \vec{j})$ must be preceded by $\mathsf{A}(\vec{i})$.

The output and condition of $\mathsf{B}(\vec{i}, \vec{j})$ can mention $\mathsf{input}@\mathsf{A}(\vec{i})$.

Mutable states

We can model mutable memory cells (shared memory) using more macros.

$$\mathsf{s}(\vec{i})@T \; : \; \text{contents of cell } \mathsf{s} \text{ at time } T$$

Each action comes with update terms describing
how $\mathsf{s}(\vec{i})@\mathsf{A}(\vec{j})$ is obtained from $\mathsf{s}(\vec{i})@\mathsf{pred}(\mathsf{A}(\vec{j}))$.

Protocols in global formulas

Syntax & semantics

First-order formulas Φ over the following atoms:

- $[\phi]_{\mathcal{P}}$: "ϕ is almost always true" when interpreted wrt. $\mathcal{P}$
- $[\vec{u} \sim \vec{v}]_{\mathcal{P},\mathcal{P}'}$: "$\vec{u}$ and $\vec{v}$ are indistinguishable" when interpreted wrt. $\mathcal{P}$ and $\mathcal{P}'$ respectively

All protocols mentionned in a global formula must have the same sets of traces (same partially ordered action symbols).

This is a condition on which actions can be scheduled, **not** on their actual executability.

Example (Privacy for two tags of Basic Hash)

- $\big[\mathsf{output@T}(i,j), \mathsf{output@T}(i,j') \sim \mathsf{output@T}(i,j), \mathsf{output@T}(i',j')\big]_{\mathcal{P},\mathcal{P}}$

Protocols in global formulas

Syntax & semantics

First-order formulas Φ over the following atoms:

- $[\phi]_{\mathcal{P}}$: "ϕ is almost always true" when interpreted wrt. $\mathcal{P}$
- $[\vec{u} \sim \vec{v}]_{\mathcal{P},\mathcal{P}'}$: "$\vec{u}$ and $\vec{v}$ are indistinguishable"
  when interpreted wrt. $\mathcal{P}$ and $\mathcal{P}'$ respectively

All protocols mentionned in a global formula must have
the same sets of traces (same partially ordered action symbols).

This is a condition on which actions can be scheduled,
not on their actual executability.

Example (Privacy for two tags of Basic Hash)

- $\big[\mathsf{output}@\mathsf{T}(i,j), \mathsf{output}@\mathsf{T}(i,j') \sim \mathsf{output}@\mathsf{T}(i,j), \mathsf{output}@\mathsf{T}(i',j')\big]_{\mathcal{P},\mathcal{P}}$
- $\big[\mathsf{output}@\mathsf{T}(i,j), \mathsf{output}@\mathsf{T}(i,j') \sim \mathsf{output}@\mathsf{T}(i,j), \mathsf{output}@\mathsf{T}(i,j')\big]_{\mathcal{P},\mathcal{P}'}$
  with $\mathcal{P}'$ where $\mathsf{T}(i,j)$ uses (i,j) as identity,
  i.e. uses key $\mathsf{k}'(i,j)$ rather than $\mathsf{k}(i)$.

More macros

Cumulative executability condition

Macro $\mathsf{cond}@\tau$ only gives the condition of τ alone.
We accumulate it to know if the trace can execute thus far:

$$\begin{aligned} \mathsf{exec}@\mathsf{init} &= \mathsf{true} \\ \mathsf{exec}@\tau &= \mathsf{exec}@\mathsf{pred}(\tau) \wedge \mathsf{cond}@\tau \end{aligned}$$

Frame

Define what an attacker has observed at a given point in a trace:

$$\begin{aligned} \mathsf{frame}@\mathsf{init} &= \mathsf{empty} \\ \mathsf{frame}@\tau &= \langle \mathsf{frame}@\mathsf{pred}(\tau), \mathsf{exec}@\tau, \mathsf{if}\ \mathsf{exec}@\tau\ \mathsf{then}\ \mathsf{output}@\tau \rangle \end{aligned}$$

We can then define $\mathsf{input}@\tau = \mathbf{att}(\mathsf{frame}@\tau)$.

Observational equivalence

Two protocols $\mathcal{P}$ and $\mathcal{P}'$ are indistinguishable when:

$$\forall \tau.\ [\mathsf{happens}(\tau)]_{\mathcal{P}} \Rightarrow [\mathsf{frame}@\tau \sim \mathsf{frame}@\tau]_{\mathcal{P},\mathcal{P}'}$$

Threat model

Attackers choose a trace, i.e. a sequence of actions to execute.
At each step of the trace, they:

- compute the input of the action from past observables
  ($\mathbf{att}(\_)$ in input, same on both sides)
- obtain new observables: executability bit and output message
  (def. of frame)

At the end, they attempt to distinguish observables for $\mathcal{P}$ and $\mathcal{P}'$.
(def. of $\sim$)

Basic Hash protocol

Let's prove unlinkability:

> *"Ensuring that a user may make multiple uses of a service without others being able to link these uses together." (ISO/IEC 15408)*

Basic Hash protocol

The multiple-session system, where multiple tags play multiple sessions, must be indistinguishable from a single-session system where multiple tags play one session each.

Basic Hash protocol

The multiple-session system, where multiple tags play multiple sessions, must be indistinguishable from a single-session system where multiple tags play one session each.

First attempt:

`movep/basic-hash-fail.sp`

Basic Hash protocol

The multiple-session system, where multiple tags play multiple sessions, must be indistinguishable from a single-session system where multiple tags play one session each.

First attempt:

`movep/basic-hash-fail.sp`

Proper model, with an interesting proof:

`basic-hash.sp`

Note: a sequent $[\phi_1], \ldots, [\phi_n], \phi'_1, \ldots, \phi'_m \vdash \psi$ reads as

$$[\phi_1] \wedge \ldots \wedge [\phi_n] \Rightarrow [\phi'_1 \wedge \ldots \wedge \phi'_m \Rightarrow \psi]$$

Outline

What's next?

Learn some more on our website, with tutorials and interactive examples:

https://squirrel-prover.github.io/

Hands on experience in practical sessions!