Formal Proofs of Crypto Protocols with Squirrel

David Baelde

ENS Rennes & IRISA

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

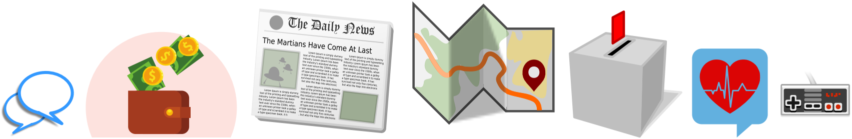
A little bit of security, a lot of logic, a few demos.

- Discover an important application of formal logic.
- A source for new problems in designing and studying logics.

Outline

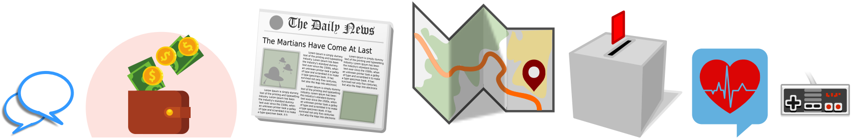
Security & Privacy

Increasingly many activities are becoming digitalized.



Security & Privacy

Increasingly many activities are becoming digitalized.

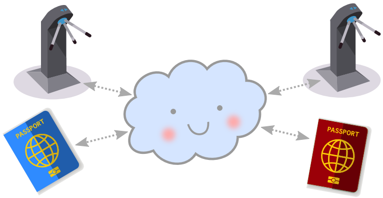


These systems must ensure important properties:

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

Frequent flaws at the hardware, software and specification levels.

Example protocol: Basic Hash



Each tag (T_i) owns a secret key k_i.

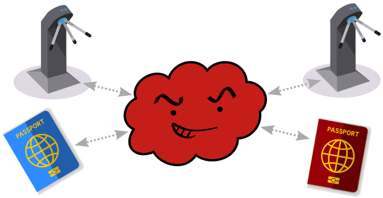
Reader (R) knows all legitimate keys.

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

Scenario under consideration:

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

Example protocol: Basic Hash



Each tag (T_i) owns a secret key k_i.

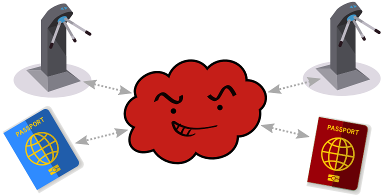
Reader (R) knows all legitimate keys.

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

Scenario under consideration:

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

Example protocol: Basic Hash



Each tag (T_i) owns a secret key k_i.

Reader (R) knows all legitimate keys.

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

Scenario under consideration:

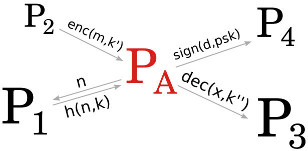
- roles R, T_1, ..., T_n; arbitrary number of sessions for each role
- attacker can intercept messages, inject new messages

Security properties:

- Readers must accept only legitimate inputs.
- It must not be possible to track tags.

Symbolic model

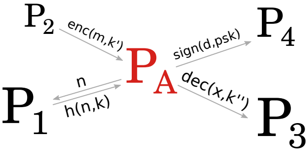
An idealized setting, also known as Dolev-Yao model



Messages	=	terms
Secrets	=	fresh constants
Computations	=	equational theory

Symbolic model

An idealized setting, also known as Dolev-Yao model



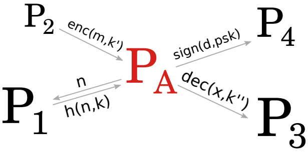
Messages	=	terms
Secrets	=	fresh constants
Computations	=	equational theory

Example (Equational theories)

- **Symmetric encryption**: $\mathsf{sdec}(\mathsf{senc}(x, y), y) =_{\mathsf{E}} x$.
- **Hash function**: no equation.

Symbolic model

An idealized setting, also known as Dolev-Yao model



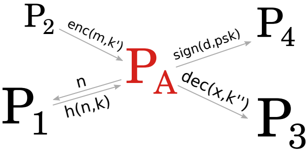
Messages	=	terms
Secrets	=	fresh constants
Computations	=	equational theory

Example (Equational theories)

- **Symmetric encryption**: $\mathsf{sdec}(\mathsf{senc}(x, y), y) =_{\mathsf{E}} x$.
- **Hash function**: no equation.

Symbolic model

An idealized setting, also known as Dolev-Yao model



Messages	=	terms
Secrets	=	fresh constants
Computations	=	equational theory

Example (Equational theories)

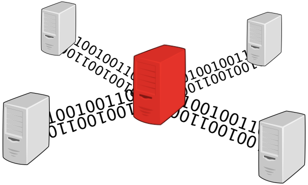
- **Symmetric encryption**: $\mathrm{sdec}(\mathrm{senc}(x, y), y) =_{\mathsf{E}} x$.
- **Hash function**: no equation.

Example (Basic Hash in the symbolic model)

Informally, both authentication and privacy hold.

Computational model

The cryptographer's mathematical model for provable security

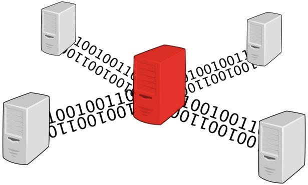


Messages	$=$	bitstrings
Secrets	$=$	random samplings
Participants	$=$	PPTIME Turing machines

$+$ assumptions on what cannot be achieved

Computational model

The cryptographer's mathematical model for provable security



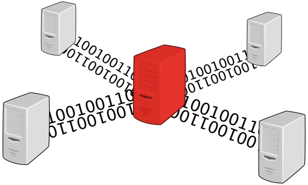
Messages	$=$	bitstrings
Secrets	$=$	random samplings
Participants	$=$	PPTIME Turing machines

$+$ 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 $\boxed{T_i \to R : \langle n_T, h(n_T, k_i)\rangle}$

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 $\boxed{T_i \to R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle}$

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.

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)

There is a negligible probability of success for the following game:

- 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)

There is a negligible probability of success for the following game:

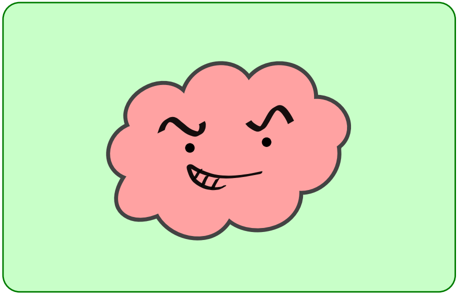
- Draw $k_1, \ldots, k_n$ uniformly at random. Flip a coin b.
- Consider oracles $\mathcal{O}_i(x) = (\text{if } b \text{ then } \mathsf{h}(x, k_i) \text{ else random}())$ that can only be queried once per message.
- Succeed if $b = \mathcal{A}^{\mathcal{O}_1, \ldots, \mathcal{O}_n}$.

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.

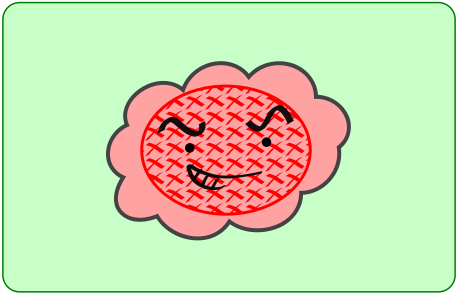
Limitations of symbolic model

- Security assumptions can be imprecise (cf. EUF-CMA and PRF).
- Obtaining computational guarantees from the symbolic model is hard!



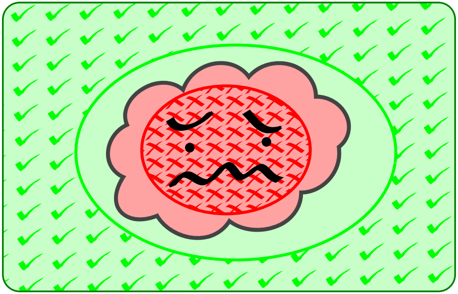
Limitations of symbolic model

- Security assumptions can be imprecise (cf. EUF-CMA and PRF).
- Obtaining computational guarantees from the symbolic model is hard!
- A fundamental problem:
  one should not specify what the attacker can do but what is safe.



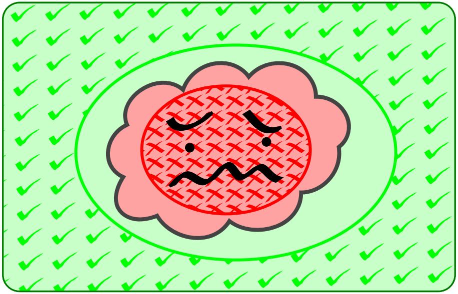
Limitations of symbolic model

- Security assumptions can be imprecise (cf. EUF-CMA and PRF).
- Obtaining computational guarantees from the symbolic model is hard!
- A fundamental problem:
  one should not specify what the attacker can do but what is safe.



Limitations of symbolic model

- Security assumptions can be imprecise (cf. EUF-CMA and PRF).
- Obtaining computational guarantees from the symbolic model is hard!
- A fundamental problem:
  one should not specify what the attacker can do but what is safe.
  The CCSA approach does just this, while keeping the modelling of messages as terms, to allow verification via automated reasoning.



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	↑	↑	↗	↗	↑	↘	↗	↓

- 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 1.3, 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

Terms of the CCSA logic (informally)

First-order terms interpreted as probabilistic computations of bitstrings.

Names

Special constants used to represent random samplings.
Notation: n, r, k...

Honest functions symbols

Function symbols used to represent primitives, public constants...
Notation: $\mathsf{f}(m)$, $\mathsf{g}(m, n)$, ok...

Adversarial function symbols

Function symbols used to represent attacker computations.
Notation: $\mathbf{att}(m_1, \ldots, m_k)$.

Example

In reasonable models where h is a hash function,
$\mathbf{att}(\mathsf{h}(\mathsf{true}, \mathsf{k}))$ and $\mathsf{h}(\mathsf{false}, \mathsf{k})$ are unlikely to compute the same bitstring.

Terms of the CCSA logic (formally)

We first need to fix a specific way of modelling probabilistic computations.

Definition (k-PPTM)

A k-PPTM is a polynomial-time Turing machine over the binary alphabet, with some number of regular input tapes + special read-only input tapes:

- a tape for receiving the security parameter $\eta \in \mathbb{N}$ in unary;
- k infinite binary tapes used as randomness sources.

We will use two randomness tapes:

- ρ_h for honest samplings (by the protocol)
- ρ_a for attacker samplings (by the probabilistic attacker)

Terms of the CCSA logic (formally)

A computational model $\mathcal{M}$ is given by:

Given a semantic assignment σ mapping variables to 2-PPTMs,
we interpret any term t as a 2-PPTM $[\![t]\!]^{\sigma}_{\mathcal{M}}$:

- $[\![x]\!]^{\sigma}_{\mathcal{M}} = \sigma(x)$

Terms of the CCSA logic (formally)

A computational model $\mathcal{M}$ is given by:

- an injective mapping ι associating to each name its position $\iota(\mathsf{n})$

Given a semantic assignment σ mapping variables to 2-PPTMs,
we interpret any term t as a 2-PPTM $[\![t]\!]^\sigma_\mathcal{M}$:

- $[\![x]\!]^\sigma_\mathcal{M} = \sigma(x)$
- $[\![\mathsf{n}]\!]^\sigma_\mathcal{M}(1^\eta, \rho_h, \rho_a) \stackrel{def}{=} \rho_h[\iota(\mathsf{n}) \times \eta, \iota(\mathsf{n}) \times (\eta + 1) - 1]$

Terms of the CCSA logic (formally)

A computational model $\mathcal{M}$ is given by:

- an injective mapping ι associating to each name its position $\iota(\mathsf{n})$
- for each honest function symbol f a 0-PPTM $\mathsf{f}_{\mathcal{M}}$

Given a semantic assignment σ mapping variables to 2-PPTMs,
we interpret any term t as a 2-PPTM $[\![t]\!]^{\sigma}_{\mathcal{M}}$:

- $[\![x]\!]^{\sigma}_{\mathcal{M}} = \sigma(x)$
- $[\![\mathsf{n}]\!]^{\sigma}_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) \stackrel{def}{=} \rho_h[\iota(\mathsf{n}) \times \eta, \iota(\mathsf{n}) \times (\eta + 1) - 1]$
- $[\![\mathsf{f}(t_1, \ldots, t_k)]\!]^{\sigma}_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) \stackrel{def}{=}$
  $\mathsf{f}_{\mathcal{M}}([\![t_1]\!]^{\sigma}_{\mathcal{M}}(1^\eta, \rho_h, \rho_a), \ldots, [\![t_k]\!]^{\sigma}_{\mathcal{M}}(1^\eta, \rho_h, \rho_a), 1^\eta)$

Terms of the CCSA logic (formally)

A computational model $\mathcal{M}$ is given by:

- an injective mapping ι associating to each name its position $\iota(\mathsf{n})$
- for each honest function symbol f a 0-PPTM $\mathsf{f}_{\mathcal{M}}$
- for each adversarial function symbol **att** a 1-PPTM $\mathbf{att}_{\mathcal{M}}$

Given a semantic assignment σ mapping variables to 2-PPTMs, we interpret any term t as a 2-PPTM $[\![t]\!]^{\sigma}_{\mathcal{M}}$:

- $[\![x]\!]^{\sigma}_{\mathcal{M}} = \sigma(x)$
- $[\![\mathsf{n}]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a) \stackrel{def}{=} \rho_h[\iota(\mathsf{n}) \times \eta, \iota(\mathsf{n}) \times (\eta + 1) - 1]$
- $[\![\mathsf{f}(t_1, \ldots, t_k)]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a) \stackrel{def}{=}$
  $\mathsf{f}_{\mathcal{M}}([\![t_1]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), \ldots, [\![t_k]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta})$
- $[\![\mathbf{att}(t_1, \ldots, t_k)]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a) \stackrel{def}{=}$
  $\mathbf{att}_{\mathcal{M}}([\![t_1]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), \ldots, [\![t_k]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta}, \rho_a)$

Terms of the CCSA logic (examples)

Example (determinism, independence)

- $\mathsf{h}(\mathsf{cst})$ is interpreted as a deterministic computation: $[\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$ for any $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however, **att**(cst) may depend on the random tape ρ_a

Terms of the CCSA logic (examples)

Example (determinism, independence)

- $\mathsf{h}(\mathsf{cst})$ is interpreted as a deterministic computation:
  $[\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$ for any $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however, $\mathbf{att}(\mathsf{cst})$ may depend on the random tape ρ_a
- consider probabilities over samplings of random tapes:
  $\Pr[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{m}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = ???$

Terms of the CCSA logic (examples)

Example (determinism, independence)

- $\mathsf{h}(\mathsf{cst})$ is interpreted as a deterministic computation:
  $[\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$ for any $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however, $\mathbf{att}(\mathsf{cst})$ may depend on the random tape ρ_a
- consider probabilities over samplings of random tapes:
  $\mathrm{Pr}[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{m}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = 2^{-\eta}$ if n, m distinct

Terms of the CCSA logic (examples)

Example (determinism, independence)

- h(cst) is interpreted as a deterministic computation:
  $[\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$ for any $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however, **att**(cst) may depend on the random tape ρ_a
- consider probabilities over samplings of random tapes:
  $\Pr[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{m}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = 2^{-\eta}$ if n, m distinct
  $\Pr[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{t}\ ]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = 2^{-\eta}$ if t closed, $\mathsf{n} \not\in t$

Terms of the CCSA logic (examples)

Example (determinism, independence)

- $\mathsf{h}(\mathsf{cst})$ is interpreted as a deterministic computation: $[\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{h}(\mathsf{cst})]\!]_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$ for any $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however, $\mathbf{att}(\mathsf{cst})$ may depend on the random tape ρ_a
- consider probabilities over samplings of random tapes:
  $\Pr[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{m}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = 2^{-\eta}$ if n, m distinct
  $\Pr[\ [\![\mathsf{n}]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = [\![\mathsf{t}\ ]\!]_{\mathcal{M}}(1^\eta, \rho_h, \rho_a)\ ] = 2^{-\eta}$ if t closed, $\mathsf{n} \notin t$

For convenience we assume that some builtin function symbols have their standard semantics: true, false, $\_ \dot{=} \_$, $\_ \dot{\wedge} \_$, $\_ \dot{\vee} \_$, $\_ \dot{\Rightarrow} \_$, etc.

Example (boolean builtins)

- $\mathsf{n} \mathrel{\dot{\neq}} \mathsf{m}$ is true with negligible probability $(2^{-\eta})$ for distinct names
- $(u \dot{=} v) \mathrel{\dot{\Rightarrow}} (v \dot{=} w) \mathrel{\dot{\Rightarrow}} (u \dot{=} v)$ is always true (probability 1)

Atoms of the CCSA logic

The logic features a single predicate:
$\vec{u} \sim \vec{v}$ can be formed for any sequences of terms $\vec{u}, \vec{v}$ of the same length.

Atoms of the CCSA logic

The logic features a single predicate:
$\vec{u} \sim \vec{v}$ can be formed for any sequences of terms $\vec{u}, \vec{v}$ of the same length.

Definition (Computational indistinguishability)

$\mathcal{M}, \sigma \models \vec{u} \sim \vec{v}$ when
the following quantity is negligible in η for any 1-PPTM $\mathcal{A}$:

$$| \Pr[\mathcal{A}([\![\vec{u}]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta}, \rho_a)] - \Pr[\mathcal{A}([\![\vec{v}]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta}, \rho_a)] |$$

(This is called the *advantage* of *distinguisher/attacker* $\mathcal{A}$.)

Atoms of the CCSA logic

The logic features a single predicate:
$\vec{u} \sim \vec{v}$ can be formed for any sequences of terms $\vec{u}, \vec{v}$ of the same length.

Definition (Computational indistinguishability)

$\mathcal{M}, \sigma \models \vec{u} \sim \vec{v}$ when
the following quantity is negligible in η for any 1-PPTM $\mathcal{A}$:

$$| \Pr[\mathcal{A}([\![\vec{u}]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta}, \rho_a)] - \Pr[\mathcal{A}([\![\vec{v}]\!]^{\sigma}_{\mathcal{M}}(1^{\eta}, \rho_h, \rho_a), 1^{\eta}, \rho_a)] |$$

(This is called the *advantage* of *distinguisher/attacker* $\mathcal{A}$.)

The rest is as usual in first-order logic: satisfaction for general formulas, validity, logical consequence, etc.

Example

The following formula is valid, i.e. satisfied in all computational models:
$\forall x, y, z, x', y', z'.\ (x, y, z \sim x', y', z') \Rightarrow (x', z', y' \sim x, z, y)$.

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse
- indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} \doteq \mathsf{n}) \sim \mathsf{false}$ assuming m, n distinct

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse
- indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} \doteq \mathsf{n}) \sim \mathsf{false}$ assuming m, n distinct
- more generally, the following formula is valid:
  $\big((x \doteq y) \sim \mathsf{true} \wedge \vec{u}[x] \sim \vec{v}[x]\big) \Rightarrow \vec{u}[y] \ \sim \vec{v}[y]$

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse
- indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} \doteq \mathsf{n}) \sim \mathsf{false}$ assuming m, n distinct
- more generally, the following formula is valid:
  $\big((x \doteq y) \sim \mathsf{true} \wedge \vec{u}[x] \sim \vec{v}[x]\big) \Rightarrow \vec{u}[y] \ \sim \vec{v}[y]$

Example (relating boolean connectives)

- $(\phi \mathrel{\dot\vee} \psi) \sim \mathsf{true} \ \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \vee (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathrel{\dot\wedge} \psi) \sim \mathsf{true} \ \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \wedge (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathrel{\dot\Rightarrow} \psi) \sim \mathsf{true} \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \Rightarrow (\psi \sim \mathsf{true})$ is valid

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse
- indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} \doteq \mathsf{n}) \sim \mathsf{false}$ assuming m, n distinct
- more generally, the following formula is valid:
  $\big((x \doteq y) \sim \mathsf{true} \wedge \vec{u}[x] \sim \vec{v}[x]\big) \Rightarrow \vec{u}[y] \ \sim \vec{v}[y]$

Example (relating boolean connectives)

- $(\phi \mathrel{\dot\vee} \psi) \sim \mathsf{true} \ \Leftarrow (\phi \sim \mathsf{true}) \vee (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathrel{\dot\wedge} \psi) \sim \mathsf{true} \ \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \wedge (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathrel{\dot\Rightarrow} \psi) \sim \mathsf{true} \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \Rightarrow (\psi \sim \mathsf{true})$ is valid

Example formulas

Example (indistinguishability over booleans)

$u \sim \mathsf{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \mathsf{true} \models x \sim y$ but not the converse
- indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} \doteq \mathsf{n}) \sim \mathsf{false}$ assuming m, n distinct
- more generally, the following formula is valid:
  $\big((x \doteq y) \sim \mathsf{true} \wedge \vec{u}[x] \sim \vec{v}[x]\big) \Rightarrow \vec{u}[y] \sim \vec{v}[y]$

Example (relating boolean connectives)

- $(\phi \mathbin{\dot\vee} \psi) \sim \mathsf{true} \Leftarrow (\phi \sim \mathsf{true}) \vee (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathbin{\dot\wedge} \psi) \sim \mathsf{true} \Leftrightarrow (\phi \sim \mathsf{true}) \wedge (\psi \sim \mathsf{true})$ is valid
- $(\phi \mathbin{\dot\Rightarrow} \psi) \sim \mathsf{true} \overset{?}{\Leftrightarrow} (\phi \sim \mathsf{true}) \Rightarrow (\psi \sim \mathsf{true})$ is valid

Example formulas

Example (indistinguishability over booleans)

$u \sim \text{true}$ means that u is true with overwhelming probability.

Example (equality and indistinguishability)

- $(x \doteq y) \sim \text{true} \models x \sim y$ but not the converse
- indeed, $\text{m} \sim \text{n}$ but $(\text{m} \doteq \text{n}) \sim \text{false}$ assuming m, n distinct
- more generally, the following formula is valid:
  $\big((x \doteq y) \sim \text{true} \wedge \vec{u}[x] \sim \vec{v}[x]\big) \Rightarrow \vec{u}[y] \sim \vec{v}[y]$

Example (relating boolean connectives)

- $(\phi \mathbin{\dot{\vee}} \psi) \sim \text{true} \Leftarrow (\phi \sim \text{true}) \vee (\psi \sim \text{true})$ is valid
- $(\phi \mathbin{\dot{\wedge}} \psi) \sim \text{true} \Leftrightarrow (\phi \sim \text{true}) \wedge (\psi \sim \text{true})$ is valid
- $(\phi \mathbin{\dot{\Rightarrow}} \psi) \sim \text{true} \Rightarrow (\phi \sim \text{true}) \Rightarrow (\psi \sim \text{true})$ is valid

Axioms

To prove that a formula of the CCSA logic holds in a class of models, it suffices to check (using your favorite first-order deduction technique) that it is a logical consequence of axioms that hold in this class of models.

Axioms

To prove that a formula of the CCSA logic holds in a class of models, it suffices to check (using your favorite first-order deduction technique) that it is a logical consequence of axioms that hold in this class of models.

Computational axioms

Some axioms hold in all computational models:

- Indistinguishability is an equivalence, and is stable by permutation.
- $\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2} \Rightarrow \vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})$ function application (FA)

Axioms

To prove that a formula of the CCSA logic holds in a class of models, it suffices to check (using your favorite first-order deduction technique) that it is a logical consequence of axioms that hold in this class of models.

Computational axioms

Some axioms hold in all computational models:

- Indistinguishability is an equivalence, and is stable by permutation.
- $\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2} \Rightarrow \vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})$ function application (FA)
- $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{m}$ when $\vec{u}, \vec{v}$ are closed and do not contain n, m
- $(t \doteq \mathsf{n}) \sim \mathsf{false}$ when t is closed and does not contain n

Axioms

To prove that a formula of the CCSA logic holds in a class of models, it suffices to check (using your favorite first-order deduction technique) that it is a logical consequence of axioms that hold in this class of models.

Computational axioms

Some axioms hold in all computational models:

- Indistinguishability is an equivalence, and is stable by permutation.
- $\vec{u_1}, \vec{u_2} \sim \vec{v_1}, \vec{v_2} \Rightarrow \vec{u_1}, \mathsf{f}(\vec{u_2}) \sim \vec{v_1}, \mathsf{f}(\vec{v_2})$ function application (FA)
- $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, \mathsf{n} \sim \vec{v}, \mathsf{m}$ when $\vec{u}, \vec{v}$ are closed and do not contain n, m
- $(t \doteq \mathsf{n}) \sim \mathsf{false}$ when t is closed and does not contain n

Implementation axioms

Valid in models featuring reasonable implementations of some primitives.
Example: $\forall x, y.\ (\mathsf{fst}(\mathsf{pair}(x, y)) \doteq x) \sim \mathsf{true}$ and similarly for snd.

Crypto axioms

Implementation axioms that specify security assumptions,
i.e. things that *cannot* be achieved.

Crypto axioms

Implementation axioms that specify security assumptions,
i.e. things that *cannot* be achieved.

Example (Unforgeability)

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

$$\mathsf{true} \sim \left(\, u \mathrel{\dot{=}} \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} (\dot{\bigvee}_{s \in S}\, s \mathrel{\dot{=}} v)\,\right)$$

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

Crypto axioms

Example (Unforgeability)

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

$$\mathsf{true} \sim \left(u \stackrel{\bullet}{=} \mathsf{h}(v, \mathsf{k}) \stackrel{\bullet}{\Rightarrow} (\stackrel{\bullet}{\bigvee}_{s \in S} s \stackrel{\bullet}{=} v) \right)$$

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

Proof.

Fix a model $\mathcal{M}$. Observe that $[\![u]\!]_{\mathcal{M}}$ and $[\![v]\!]_{\mathcal{M}}$ can be seen as attacker computations in the EUF-CMA game:

- occurrences $\mathsf{h}(s, \mathsf{k})$ computed via oracle queries on s;
- k is not accessed otherwise.

If $\mathsf{h}_{\mathcal{M}}$ satisfies EUF-CMA, then $[\![u]\!]_{\mathcal{M}}$ and $[\![\mathsf{h}(v, \mathsf{k})]\!]_{\mathcal{M}}$ can only be equal when $[\![v]\!]_{\mathcal{M}}$ has previously been used as a query – except for a negligible probability. Hence $[\![\_ \stackrel{\bullet}{\Rightarrow} \_]\!]_{\mathcal{M}}$ is true with overwhelming probability. $\square$

Crypto axioms

Example (Unforgeability)

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

$$\mathsf{true} \sim \left(u \doteq \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} (\dot{\bigvee}_{s \in S}\, s \doteq v) \right)$$

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

Example (Pseudo-randomness)

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

$$\vec{v}, \mathsf{h}(t, \mathsf{k}) \sim \vec{v}, \mathsf{if}\ \dot{\bigvee}_{s \in S}\, s \doteq t \ \mathsf{then}\ \mathsf{h}(t, \mathsf{k})\ \mathsf{else}\ \mathsf{n}$$

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 $\mathsf{h}(\_, \mathsf{k})$.

Crypto axioms

Example (Unforgeability)

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

$$\mathsf{true} \sim \left(u \doteq \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} (\dot{\bigvee}_{s \in S}\, s \doteq v) \right)$$

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

Example (Pseudo-randomness)

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

$$\vec{v}, \mathsf{h}(t, \mathsf{k}) \sim \vec{v}, \mathsf{if}\ \dot{\bigvee}_{s \in S}\, s \doteq t \ \mathsf{then}\ \mathsf{h}(t, \mathsf{k})\ \mathsf{else}\ \mathsf{n}$$

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 $\mathsf{h}(\_, \mathsf{k})$.

Proof.

Same idea as above but relying on a variant of PRF game where only the last oracle query is modified to return a random sampling. □

In Squirrel

Let's put this in practice on a simple analysis of the Basic Hash protocol.

`movep/basic-hash-two.sp`

A first proof system

To prove statements of the form $\phi \sim \mathsf{true}$ we use sequent calculus, pretending these terms are formulas:

$$\phi_1, \ldots, \phi_n \vdash \psi \quad \text{reads as} \quad (\phi_1 \mathrel{\dot{\wedge}} \ldots \mathrel{\dot{\wedge}} \phi_n \mathrel{\dot{\Rightarrow}} \psi) \sim \mathsf{true}.$$

In Squirrel

Let's put this in practice on a simple analysis of the Basic Hash protocol.

`movep/basic-hash-two.sp`

A first proof system

To prove statements of the form $\phi \sim \mathsf{true}$ we use sequent calculus, pretending these terms are formulas:

$$\phi_1, \ldots, \phi_n \vdash \psi \quad \text{reads as} \quad (\phi_1 \mathrel{\dot\wedge} \ldots \mathrel{\dot\wedge} \phi_n \mathrel{\dot\Rightarrow} \psi) \sim \mathsf{true}.$$

- All rules of classical sequent calculus are sound wrt. this semantics!
- We can also use extra rules corresponding to CCSA axioms.

Limitations of the CCSA logic

A security property needs to be verified for all traces t of a protocol.
We could check, for each trace, some entailment $\mathsf{Ax} \models \varphi_t$ but:

- So far, automatically verifying these obligations remains infeasible.
- This methodology assumes a fixed bound b on protocol traces.

base logic $\quad \varphi_{t_1},\ \varphi_{t_2},\ \ldots \quad + \quad \dfrac{\varphi' \quad \varphi''}{\varphi} \quad = \quad \pi_{t_1},\ \pi_{t_2},\ \ldots$

Limitations of the CCSA logic

A security property needs to be verified for all traces t of a protocol.
We could check, for each trace, some entailment $\mathsf{Ax} \models \varphi_t$ but:

- So far, automatically verifying these obligations remains infeasible.
- This methodology assumes a fixed bound b on protocol traces.

$\rightsquigarrow$ Develop a meta-logic

meta-logic	ϕ
	$\wr\downarrow$
base logic	$\varphi_{t_1},\ \varphi_{t_2},\ \ldots \quad + \quad \dfrac{\varphi' \quad \varphi''}{\varphi} \quad = \quad \pi_{t_1},\ \pi_{t_2},\ \ldots$

Limitations of the CCSA logic

A security property needs to be verified for all traces t of a protocol.
We could check, for each trace, some entailment $\mathsf{Ax} \models \varphi_t$ but:

- So far, automatically verifying these obligations remains infeasible.
- This methodology assumes a fixed bound b on protocol traces.

$\rightsquigarrow$ Develop a meta-logic suitable for interactive proofs, independent of b.

$$
\begin{tikzcd}
\text{meta-logic} & \phi \arrow[d, squiggly] & + & \dfrac{\phi' \quad \phi''}{\phi} \arrow[d, squiggly] & = & \Pi \arrow[d, squiggly] \\
\text{base logic} & \varphi_{t_1},\ \varphi_{t_2},\ \ldots & + & \dfrac{\varphi' \quad \varphi''}{\varphi} & = & \pi_{t_1},\ \pi_{t_2},\ \ldots
\end{tikzcd}
$$

Outline

Local meta-logic: indices and timestamps

We introduce a new logic (meta-logic) which is an enriched first-order logic, that we will interpret later in terms of the CCSA logic (base logic). The meta-logic internalizes the notion of protocol and trace.

Local meta-logic: indices and timestamps

We introduce a new logic (meta-logic) which is an enriched first-order logic, that we will interpret later in terms of the CCSA logic (base logic). The meta-logic internalizes the notion of protocol and trace.

The meta-logic features three sorts: indices, timestamps and messages.

Indices

Used to model unbounded collections, e.g. indexed names $\mathsf{k}(i)$.

Syntax: $i, j, k \ldots \in \mathcal{X}_{\mathcal{I}}$ Atoms over indices: $i = j$

Timestamps

Represent points in a trace of actions performed by the protocol.

$T ::= \tau \mid \mathsf{init} \mid \mathsf{pred}(T) \mid \mathsf{A}(\vec{i})$ $\qquad \tau \in \mathcal{X}_{\mathcal{T}}, \mathsf{A} \in \mathcal{A}$

Atoms over timestamps: $T = T'$, $T \leq T'$, $\mathsf{happens}(T)$

Quantification is only allowed over indices and timestamps.
Importantly, both indices and timestamps will be interpreted in finite sets.

Local meta-logic: messages and formulas

Some terms are dependent on the protocol's execution:
inputs, outputs, attacker's knowledge, execution conditions, etc.
This will be represented by terms of the form $\mathsf{macro}@T$.

Messages

$$t ::= x \mid \mathsf{n}(\vec{i}) \mid \mathsf{f}(\vec{t}) \mid \mathsf{input}@T \mid \mathsf{output}@T \mid \mathsf{frame}@T \mid \ldots$$

Some constructs ignored for simplicity.

Local meta-logic: messages and formulas

Some terms are dependent on the protocol's execution:
inputs, outputs, attacker's knowledge, execution conditions, etc.
This will be represented by terms of the form $\mathsf{macro}@T$.

Messages

$$t ::= x \mid \mathsf{n}(\vec{i}) \mid \mathsf{f}(\vec{t}) \mid \mathsf{input}@T \mid \mathsf{output}@T \mid \mathsf{frame}@T \mid \ldots$$

Some constructs ignored for simplicity.

Formulas

First-order formulas, without quantification over messages, over atoms

$$A ::= t = t' \mid i = i' \mid T = T' \mid T \leq T' \mid \mathsf{happens}(T) \mid \mathsf{cond}@T \mid \mathsf{exec}@T$$

The semantics of a local meta-logic formula ϕ is still of the form $t_\phi \sim \mathsf{true}$.

Local meta-logic formulas: examples

Example (Input validation for Basic Hash)

Session k of tag T_i outputs $\langle \mathsf{n}(i,k), \mathsf{h}(\mathsf{n}(i,k), \mathsf{k}(i))\rangle$.

$$\exists \tau, i.\ \mathsf{snd}(\mathsf{input}@\tau) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\tau), \mathsf{k}(i))$$

Local meta-logic formulas: examples

Example (Input validation for Basic Hash)

Session k of tag T_i outputs $\langle \mathsf{n}(i,k), \mathsf{h}(\mathsf{n}(i,k), \mathsf{k}(i))\rangle$.

$$\exists \tau, i.\ \mathsf{snd}(\mathsf{input}@\tau) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\tau), \mathsf{k}(i))$$

Example

All inputs of actions $\mathsf{A}(i)$ are outputs of actions $\mathsf{B}(j)$ that precede them:

$$\forall i.\ \mathsf{happens}(\mathsf{A}(i)) \Rightarrow \exists j.\ \mathsf{B}(j) \leq \mathsf{A}(i) \wedge \mathsf{input}@\mathsf{A}(i) = \mathsf{output}@\mathsf{B}(j)$$

Local meta-logic formulas: examples

Example (Input validation for Basic Hash)

Session k of tag T_i outputs $\langle \mathsf{n}(i,k), \mathsf{h}(\mathsf{n}(i,k), \mathsf{k}(i))\rangle$.

$$\exists \tau, i.\ \mathsf{snd}(\mathsf{input}@\tau) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\tau), \mathsf{k}(i))$$

Example

All inputs of actions $\mathsf{A}(i)$ are outputs of actions $\mathsf{B}(j)$ that precede them:

$$\forall i.\ \mathsf{happens}(\mathsf{A}(i)) \Rightarrow \exists j.\ \mathsf{B}(j) \leq \mathsf{A}(i) \wedge \mathsf{input}@\mathsf{A}(i) = \mathsf{output}@\mathsf{B}(j)$$

Intuitive semantics

We are now reasoning over all traces and all implementations of functions.
For a given trace model $\mathbb{T}$, a formula ϕ becomes $(\phi)^{\mathbb{T}} \sim \mathsf{true}$:

- existential quantifiers become finite disjunctions;
- atoms over timestamps become boolean constants.

Modelling protocols

Definition (Action descriptions)

The semantics of an action $\mathsf{A} \in \mathcal{A}$ is given by an expression of the form

$$\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$$

condition (local formula) — output (message term)

The variables $\vec{i}$ are bound in this expression, which must be closed.

Modelling protocols

Definition (Action descriptions)

The semantics of an action $\mathsf{A} \in \mathcal{A}$ is given by an expression of the form

$$\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$$

condition (local formula)   output (message term)

The variables $\vec{i}$ are bound in this expression, which must be closed.

Example (Basic Hash, over $\mathsf{T}(i,k), \mathsf{R}(j,i), \mathsf{R}_1(j)$)

Session k of tag T_i:

$$\mathsf{T}(i,k).\big(\mathsf{true},\ \langle \mathsf{n}_T(i,k), \mathsf{h}(\mathsf{n}_T(i,k), \mathsf{k}(i))\rangle\big)$$

Modelling protocols

Definition (Action descriptions)

The semantics of an action $\mathsf{A} \in \mathcal{A}$ is given by an expression of the form

$$\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$$

condition (local formula) — output (message term)

The variables $\vec{i}$ are bound in this expression, which must be closed.

Example (Basic Hash, over $\mathsf{T}(i,k), \mathsf{R}(j,i), \mathsf{R}_1(j)$)

Reader session j identifies its input coming from tag T_i:

$$\begin{aligned}\mathsf{R}(j,i).\big(&\mathsf{snd}(\mathsf{input}@\mathsf{R}(j,i)) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\mathsf{R}(j,i)), \mathsf{k}(i)),\\ &\mathsf{ok}\big)\end{aligned}$$

Modelling protocols

Definition (Action descriptions)

The semantics of an action $\mathsf{A} \in \mathcal{A}$ is given by an expression of the form

$$\mathsf{A}(\vec{i}).(\underbrace{\phi_{\mathsf{A}(\vec{i})}}_{\substack{\text{condition} \\ \text{(local formula)}}}, \underbrace{o_{\mathsf{A}(\vec{i})}}_{\substack{\text{output} \\ \text{(message term)}}})$$

The variables $\vec{i}$ are bound in this expression, which must be closed.

Example (Basic Hash, over $\mathsf{T}(i,k), \mathsf{R}(j,i), \mathsf{R}_1(j)$)

Reader session j rejects its input:

$$\mathsf{R}_1(j).\big(\forall i.\ \mathsf{snd}(\mathsf{input}@\mathsf{R}(j,i)) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\mathsf{R}(j,i)), \mathsf{k}(i)), \\ \mathsf{ko}\big)$$

Modelling protocols

Definition (Protocol, simplified)

A protocol $\mathcal{P}$ is defined by giving a set of action symbols $\mathcal{A}$ and an action description for each action symbol.
The only macro allowed in $\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$ is $\mathsf{input}@\mathsf{A}(\vec{i})$.

Modelling protocols

Definition (Protocol, simplified)

A protocol $\mathcal{P}$ is defined by giving a set of action symbols $\mathcal{A}$ and an action description for each action symbol.
The only macro allowed in $\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$ is $\mathsf{input}@\mathsf{A}(\vec{i})$.

Definition (Trace model)

A trace model $\mathbb{T}$ for $\mathcal{P}$ consists of:

- an index domain $\mathcal{D}_{\mathcal{I}} \subseteq_{\mathsf{fin}} \mathbb{N}$;
- a timestamp domain $\mathcal{D}_{\mathcal{T}} \subseteq \{\mathsf{init}, \mathsf{undef}\} \cup \{\mathsf{A}(\vec{n}) \mid \mathsf{A} \in \mathcal{A}, \vec{n} \in \mathcal{D}_{\mathcal{I}}^{|\vec{n}|}\}$;
- a total order $<_{\mathcal{T}}$ over $\mathcal{D}_{\mathcal{T}} \setminus \{\mathsf{undef}\}$ with init as minimum element.

Modelling protocols

Definition (Protocol, simplified)

A protocol $\mathcal{P}$ is defined by giving a set of action symbols $\mathcal{A}$ and an action description for each action symbol.
The only macro allowed in $\mathsf{A}(\vec{i}).(\phi_{\mathsf{A}(\vec{i})}, o_{\mathsf{A}(\vec{i})})$ is $\mathsf{input}@\mathsf{A}(\vec{i})$.

Definition (Trace model)

A trace model $\mathbb{T}$ for $\mathcal{P}$ consists of:

- an index domain $\mathcal{D}_\mathcal{I} \subseteq_{\mathsf{fin}} \mathbb{N}$;
- a timestamp domain $\mathcal{D}_\mathcal{T} \subseteq \{\mathsf{init}, \mathsf{undef}\} \cup \{\mathsf{A}(\vec{n}) \mid \mathsf{A} \in \mathcal{A}, \vec{n} \in \mathcal{D}_\mathcal{I}^{|\vec{n}|}\}$;
- a total order $<_\mathcal{T}$ over $\mathcal{D}_\mathcal{T} \setminus \{\mathsf{undef}\}$ with init as minimum element.
- mappings $\sigma_\mathcal{I} : \mathcal{X}_\mathcal{I} \to \mathcal{D}_\mathcal{I}$ and $\sigma_\mathcal{T} : \mathcal{X}_\mathcal{T} \to \mathcal{D}_\mathcal{T}$.

Example

The trace model $\mathbb{T}$ with $\mathcal{D}_\mathcal{I} = \{1, 3, 12\}$, $\mathcal{D}_\mathcal{T} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$ corresponds to the execution trace $\mathsf{T}(1,3).\mathsf{T}(1,1)$.

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

$$(\mathsf{f}(t_1, \ldots, t_k))^{\mathbb{T}}_{\mathcal{P}} = \mathsf{f}((t_1)^{\mathbb{T}}_{\mathcal{P}}, \ldots, (t_k)^{\mathbb{T}}_{\mathcal{P}})$$

$$(\mathsf{n}(i_1, \ldots, i_k))^{\mathbb{T}}_{\mathcal{P}} = \mathsf{n}_{\sigma_{\mathcal{I}}(i_1), \ldots, \sigma_{\mathcal{I}}(i_k)}$$

$$(x)^{\mathbb{T}}_{\mathcal{P}} = x$$

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

If $\sigma_{\mathcal{I}}(i) = 3$ then $(\mathsf{h}(\mathsf{n}(i,i), \mathsf{k}(i)))^{\mathbb{T}}_{\mathcal{P}} = \mathsf{h}(\mathsf{n}_{3,3}, \mathsf{k}_3)$.

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

- $(\mathsf{A}(i_1, \ldots, i_k))^{\mathbb{T}}_{\mathcal{P}} = \begin{cases} \mathsf{A}(\sigma_{\mathcal{I}}(i_1), \ldots, \sigma_{\mathcal{I}}(i_k)) \text{ if it belongs to } \mathcal{D}_{\mathcal{T}} \\ \mathsf{undef} \text{ otherwise} \end{cases}$
- init interprets as itself, and pred(_) as the predecessor wrt. $<_{\mathcal{T}}$.

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

$$(\mathsf{T}(i,i))^{\mathbb{T}\{i\mapsto 3\}}_{\mathcal{P}} = \mathsf{undef}$$

$$(\mathsf{pred}(\mathsf{T}(i,i)))^{\mathbb{T}\{i\mapsto 1\}}_{\mathcal{P}} = \mathsf{T}(1,3)$$

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

$$(\phi \wedge \psi)^{\mathbb{T}}_{\mathcal{P}} = (\phi)^{\mathbb{T}}_{\mathcal{P}} \mathbin{\dot{\wedge}} (\psi)^{\mathbb{T}}_{\mathcal{P}}$$

$$(\forall i.\phi)^{\mathbb{T}}_{\mathcal{P}} = \dot{\bigwedge}_{n \in \mathcal{D}_{\mathcal{I}}} (\phi)^{\mathbb{T}\{i \mapsto n\}}_{\mathcal{P}}$$

$$(\mathsf{happens}(T))^{\mathbb{T}}_{\mathcal{P}} = \mathsf{true} \text{ when } (T)^{\mathbb{T}}_{\mathcal{P}} \neq \mathsf{undef}$$

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

$$\begin{aligned} (\exists i.\ \mathsf{happens}(\mathsf{T}(i,i)))^{\mathbb{T}}_{\mathcal{P}} &= \dot{\bigvee}_{n \in \mathcal{D}_{\mathcal{I}}} (\mathsf{happens}(\mathsf{T}(i,i)))^{\mathbb{T}\{i \mapsto n\}}_{\mathcal{P}} \\ &= \mathsf{true} \mathbin{\dot{\vee}} \mathsf{false} \mathbin{\dot{\vee}} \mathsf{false} \end{aligned}$$

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

$$(\mathsf{output}@T)^{\mathbb{T}}_{\mathcal{P}} = \begin{cases} (o_{\mathsf{A}(\vec{i})})^{\mathbb{T}\{\vec{i}\mapsto\vec{n}\}}_{\mathcal{P}} & \text{when } (T)^{\mathbb{T}}_{\mathcal{P}} = \mathsf{A}(\vec{n}) \\ \mathsf{empty} & \text{when } (T)^{\mathbb{T}}_{\mathcal{P}} \in \{\mathsf{init}, \mathsf{undef}\} \end{cases}$$

$$(\mathsf{input}@T)^{\mathbb{T}}_{\mathcal{P}} = \mathbf{att}((\mathsf{frame}@T)^{\mathbb{T}}_{\mathcal{P}})$$

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

$$(\mathsf{output}@\mathsf{T}(i,i))^{\mathbb{T}\{i\mapsto 3\}}_{\mathcal{P}} = \mathsf{empty}$$

$$(\mathsf{output}@\mathsf{T}(i,i))^{\mathbb{T}\{i\mapsto 1\}}_{\mathcal{P}} = \langle \mathsf{n}_{1,1}, \mathsf{h}(\mathsf{n}_{1,1}, \mathsf{k}_1)\rangle$$

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

$$(\mathsf{frame}@T)^{\mathbb{T}}_{\mathcal{P}} = \mathsf{empty} \text{ when } (T)^{\mathbb{T}}_{\mathcal{P}} \in \{\mathsf{init}, \mathsf{undef}\}$$

$$(\mathsf{frame}@T)^{\mathbb{T}}_{\mathcal{P}} = (\langle \mathsf{frame}@\mathsf{pred}(T), \langle \mathsf{exec}@T, \mathsf{if}\ \mathsf{exec}@T\ \mathsf{then}\ \mathsf{output}@T\ \mathsf{else}\ \mathsf{empty} \rangle\rangle)^{\mathbb{T}}_{\mathcal{P}}$$

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

$$(\mathsf{frame}@\mathsf{T}(i,j))^{\mathbb{T}\{i \mapsto 1, j \mapsto 3\}}_{\mathcal{P}} = \langle \mathsf{empty}, \langle \ldots, \mathsf{if}\ \ldots\ \mathsf{then}\ \mathsf{n}_{1,3}, \mathsf{h}(\mathsf{n}_{1,3}, \mathsf{k}_1) \rangle\rangle$$

Semantics of local meta-logic

Definition (Interpretation $(t)^{\mathbb{T}}_{\mathcal{P}}$, $(\phi)^{\mathbb{T}}_{\mathcal{P}}$)

We simultaneously define translations for meta-logic terms and formulas:

$$\begin{array}{rcl} \text{message term } t & \rightsquigarrow & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \\ \text{index and timestamp terms} & \rightsquigarrow & \text{elements of } \mathcal{D}_{\mathcal{I}} \text{ and } \mathcal{D}_{\mathcal{T}} \\ \text{formula } \phi & \rightsquigarrow & \text{base logic boolean term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \end{array}$$

Key cases:

$$(\mathsf{exec}@T)^{\mathbb{T}}_{\mathcal{P}} = \mathsf{true} \text{ when } (T)^{\mathbb{T}}_{\mathcal{P}} \in \{\mathsf{init}, \mathsf{undef}\}$$

$$(\mathsf{exec}@T)^{\mathbb{T}}_{\mathcal{P}} = (\mathsf{cond}@T \wedge \mathsf{exec}@\mathsf{pred}(T))^{\mathbb{T}}_{\mathcal{P}}$$

$$(\mathsf{cond}@T)^{\mathbb{T}}_{\mathcal{P}} = (\phi_{\mathsf{A}(\vec{i})})^{\mathbb{T}\{\vec{i} \mapsto \vec{n}\}}_{\mathcal{P}} \text{ when } (T)^{\mathbb{T}}_{\mathcal{P}} = \mathsf{A}(\vec{n})$$

Example ($\mathbb{T}$ with $\mathcal{D}_{\mathcal{I}} = \{1, 3, 12\}$, $\mathcal{D}_{\mathcal{T}} = \{\mathsf{init} < \mathsf{T}(1,3) < \mathsf{T}(1,1)\}$)

$$(\mathsf{frame}@\mathsf{T}(i,j))^{\mathbb{T}\{i \mapsto 1, j \mapsto 3\}}_{\mathcal{P}} = \langle \mathsf{empty}, \langle \mathsf{true}, \mathsf{if\ true\ then\ } \mathsf{n}_{1,3}, \mathsf{h}(\mathsf{n}_{1,3}, \mathsf{k}_1) \rangle\rangle$$

Axioms of trace models

Example (Actions)

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

Example (Actions)

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

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

Example (Order on timestamps)

- $\mathsf{happens}(\tau) \wedge \mathsf{happens}(\tau') \Rightarrow \tau \leq \tau' \vee \tau' \leq \tau$ is valid.

Axioms of trace models

Example (Actions)

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

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

Example (Order on timestamps)

- $\mathsf{happens}(\tau) \wedge \mathsf{happens}(\tau') \Rightarrow \tau \leq \tau' \vee \tau' \leq \tau$ is valid.
- The converse is also valid.

Axioms of trace models

Example (Actions)

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

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

Example (Order on timestamps)

- $\mathsf{happens}(\tau) \wedge \mathsf{happens}(\tau') \Rightarrow \tau \leq \tau' \vee \tau' \leq \tau$ is valid.
- The converse is also valid.

Example (Case analysis and induction)

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

Lifting axioms to the meta-logic

Some axioms (e.g. $\mathsf{fst}(\langle t, t' \rangle) = t$) are trivially lifted to the meta-logic.
Axioms with occurrence constraints require some care.

Lifting axioms to the meta-logic

Some axioms (e.g. $\mathsf{fst}(\langle t, t' \rangle) = t$) are trivially lifted to the meta-logic.
Axioms with occurrence constraints require some care.

Example (Freshness in base logic)

$(t \mathrel{\dot{\neq}} \mathsf{n}) \sim \mathsf{true}$ is valid for any closed term t that doesn't contain n.

Example (Freshness in meta-logic)

$t \neq \mathsf{n}(\vec{i})$ is valid for any term t such that

- t does not contain message variables;
- such that, in any trace model $\mathbb{T}$, $(\mathsf{n}(i))^{\mathbb{T}}_{\mathcal{P}}$ does not occur in $(t)^{\mathbb{T}}_{\mathcal{P}}$.

Lifting axioms to the meta-logic

Some axioms (e.g. $\mathsf{fst}(\langle t, t' \rangle) = t$) are trivially lifted to the meta-logic. Axioms with occurrence constraints require some care.

Example (Freshness in base logic)

$(t \mathrel{\dot{\neq}} \mathsf{n}) \sim \mathsf{true}$ is valid for any closed term t that doesn't contain n.

Example (Freshness in meta-logic)

$t \neq \mathsf{n}(\vec{i})$ is valid for any term t such that

- t does not contain message variables;
- such that no $\mathsf{n}(\_)$ occurs in t and in action descriptions.

Lifting axioms to the meta-logic

Some axioms (e.g. $\mathsf{fst}(\langle t, t' \rangle) = t$) are trivially lifted to the meta-logic. Axioms with occurrence constraints require some care.

Example (Freshness in base logic)

$(t \mathrel{\dot{\neq}} \mathsf{n}) \sim \mathsf{true}$ is valid for any closed term t that doesn't contain n.

Example (Freshness in meta-logic)

$t \neq \mathsf{n}(\vec{i})$ is valid for any term t such that

- t does not contain message variables;
- such that no $\mathsf{n}(\_)$ occurs in t and, for any occurrence of $\mathsf{n}(\_)$ in the action description of some $\mathsf{A}(\vec{j})$, $\dot{\bigwedge}_{T \in t} \mathsf{not}(\mathsf{A}(\vec{j}) \leq T)$ is valid.

Lifting axioms to the meta-logic

Some axioms (e.g. $\mathsf{fst}(\langle t, t' \rangle) = t$) are trivially lifted to the meta-logic.
Axioms with occurrence constraints require some care.

Example (Freshness in base logic)

$(t \mathrel{\dot{\neq}} \mathsf{n}) \sim \mathsf{true}$ is valid for any closed term t that doesn't contain n.

Example (Freshness in meta-logic)

$t = \mathsf{n}(\vec{i}) \Rightarrow \bigvee_{\mathsf{A}(\vec{j}) \in S} \exists \vec{j}.\ \bigvee_{T \in t} \mathsf{A}(\vec{j}) \leq T$ is valid provided

- t does not contain message variables and occurrences of $\mathsf{n}(\_)$,
- S is the set of actions whose descriptions contains occurrences of $\mathsf{n}(\_)$.

Further precision improvements are possible and implemented.

Lifting axioms to the meta-logic

Example (Unforgeability in base logic)

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

$$\mathsf{true} \sim \left(u \mathrel{\dot{\dot{=}}} \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} \left(\dot{\dot{\bigvee}}_{\mathsf{h}(s,\mathsf{k}) \in \mathsf{subterm}(u,v)} s \mathrel{\dot{\dot{=}}} v\right) \right)$$

where u, v are closed terms only containing k as $\mathsf{h}(\_, \mathsf{k})$.

Lifting axioms to the meta-logic

Example (Unforgeability in base logic)

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

$$\mathsf{true} \sim \left(u \doteq \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} (\dot{\bigvee}_{\mathsf{h}(s,\mathsf{k}) \in \mathsf{subterm}(u,v)} s \doteq v) \right)$$

where u, v are closed terms only containing k as $\mathsf{h}(\_, \mathsf{k})$.

Assume $\mathsf{ST}_{\mathcal{P}}(t)$ is a set of meta-logic terms such that, for all $\mathbb{T}$, any occurrence of $\mathsf{h}(\_, \mathsf{k}_{\_})$ in $(t)^{\mathbb{T}}_{\mathcal{P}}$ is the interpretation in $\mathbb{T}$ of an occurrence of $\mathsf{h}(\_, \mathsf{k}(\_))$ in a term of $\mathsf{ST}_{\mathcal{P}}(t)$.

Lifting axioms to the meta-logic

Example (Unforgeability in base logic)

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

$$\mathsf{true} \sim \left(u \doteq \mathsf{h}(v, \mathsf{k}) \mathrel{\dot{\Rightarrow}} (\dot{\bigvee}_{\mathsf{h}(s,\mathsf{k}) \in \mathsf{subterm}(u,v)} s \doteq v) \right)$$

where u, v are closed terms only containing k as $\mathsf{h}(\_, \mathsf{k})$.

Assume $\mathsf{ST}_{\mathcal{P}}(t)$ is a set of meta-logic terms such that, for all $\mathbb{T}$, any occurrence of $\mathsf{h}(\_, \mathsf{k}\_)$ in $(t)^{\mathbb{T}}_{\mathcal{P}}$ is the interpretation in $\mathbb{T}$ of an occurrence of $\mathsf{h}(\_, \mathsf{k}(\_))$ in a term of $\mathsf{ST}_{\mathcal{P}}(t)$.

Example (Unforgeability in meta-logic)

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

$$u = \mathsf{h}(v, \mathsf{k}(\vec{i})) \Rightarrow (\dot{\bigvee}_{\mathsf{h}(s,\mathsf{k}(\vec{j})) \in \mathsf{subterm}(\mathsf{ST}_{\mathcal{P}}(u,v))} s \doteq v) \,)$$

when u, v contain no message variable and, for all $\mathbb{T}$, $(\mathsf{k}(\vec{i}))^{\mathbb{T}}$ only occurs as a key in $(u, v)^{\mathbb{T}}$.

Basic Hash

We can finally put everything together!

`movep/basic-hash-wa.sp`

Adding sequential dependencies

We add a partial order specifying sequential dependencies between actions.

Definition (Protocol, continued)

A protocol $\mathcal{P}$ also specifies a partial order $<_{\mathcal{P}}$ over indexed actions, such that $\mathsf{A}(\vec{i}) <_{\mathcal{P}} \mathsf{B}(\vec{j})$ implies $\mathsf{A}(\sigma(\vec{i})) <_{\mathcal{P}} \mathsf{B}(\sigma(\vec{j}))$ for any $\sigma : \mathcal{X}_{\mathcal{I}} \to \mathcal{X}_{\mathcal{I}}$. Elements in action descriptions of $\mathsf{A}(\vec{i})$ can also mention macros $\mathsf{m}@\mathsf{pred}^n(\mathsf{B}(\vec{j}))$ for $\mathsf{B}(\vec{j}) < \mathsf{A}(\vec{i})$.

Adding sequential dependencies

We add a partial order specifying sequential dependencies between actions.

Definition (Protocol, continued)

A protocol $\mathcal{P}$ also specifies a partial order $<_{\mathcal{P}}$ over indexed actions, such that $\mathsf{A}(\vec{i}) <_{\mathcal{P}} \mathsf{B}(\vec{j})$ implies $\mathsf{A}(\sigma(\vec{i})) <_{\mathcal{P}} \mathsf{B}(\sigma(\vec{j}))$ for any $\sigma : \mathcal{X}_{\mathcal{I}} \to \mathcal{X}_{\mathcal{I}}$. Elements in action descriptions of $\mathsf{A}(\vec{i})$ can also mention macros $\mathsf{m}@\mathsf{pred}^n(\mathsf{B}(\vec{j}))$ for $\mathsf{B}(\vec{j}) < \mathsf{A}(\vec{i})$.

Definition (Trace model, continued)

We require that $<_{\mathcal{T}}$ is downward-closed wrt. $<_{\mathcal{P}}$.

Example (More axioms)

For any actions such that $\mathsf{A}(\vec{i}) <_{\mathcal{P}} \mathsf{B}(\vec{j})$ we have:

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

Application: LAK protocol (variant)

$$
\begin{array}{lcllll}
R & \rightarrow & T & : & \mathsf{nR} \\
T & \rightarrow & R & : & \langle \mathsf{nT}, \mathsf{h}(\langle \mathsf{nR}, \mathsf{nT}, \mathsf{tag1} \rangle, \mathsf{key}) \rangle \\
R & \rightarrow & T & : & \ldots
\end{array}
$$

Actions for LAK

- $\mathsf{T}(i, j)$ for session j of T_i
- $\mathsf{R}(k)$ for the first output of reader session k
- $\mathsf{R}_1(k, i)$ when reader session k accepts tag i — $\mathsf{R}(k) < \mathsf{R}_a(k)$
- $\mathsf{R}_2(k)$ when reader session k rejects — $\mathsf{R}(k) < \mathsf{R}_r(k)$

`lak-tags-full-wa.sp`

Adding mutable state

Some protocols use memory cells to update information from one session to the next. We model it by adding state macros of the form $\mathsf{s}(\vec{i})@T$.

Definition (Protocol, continued)

Protocols also need to define, for each state macro $\mathsf{s}(\vec{i})$:

- an initial value $i_{\mathsf{s}(\vec{i})}$;
- for each action $\mathsf{A}(\vec{j})$, an update term $u_{\mathsf{s}(\vec{i}),\mathsf{A}(\vec{j})}$.

The interpretation is naturally extended to handle state macros.

Example

Cells $\mathsf{s}(i)$ containing messages of the form $\mathsf{f}^k(\mathsf{c})$, with $\mathsf{A}(i)$ updating $\mathsf{s}(i)$:

- $i_{\mathsf{s}(i)} = \mathsf{c}$
- $u_{\mathsf{s}(\vec{i}),\mathsf{A}(\vec{j})} = \mathsf{if}\ i = j\ \mathsf{then}\ \mathsf{f}(\mathsf{s}(i)@\mathsf{pred}(\mathsf{A}(\vec{i})))\ \mathsf{else}\ \mathsf{s}(i)@\mathsf{pred}(\mathsf{A}(\vec{i}))$

Application: OSK protocol (variant)

An RFID protocol where tags update their state:

- Tag T_i maintains a state $\mathsf{s}(i)$ initialized with $\mathsf{s}_0(i)$.
- Each session of T_i updates $\mathsf{s}(i) := \mathsf{h}(\mathsf{s}(i), \mathsf{k})$ and outputs $\mathsf{g}(\mathsf{s}(i), \mathsf{k}')$.
- Readers know all initial values, accept all inputs $\mathsf{g}(\mathsf{h}^n(\mathsf{s}_0(i), \mathsf{k}), \mathsf{k}')$.

`stateful/running-ex-oracle.sp`

Main difficulty: deriving high-levels lemmas...and proving them.

What we have achieved so far

Summary

We have a defined a meta-logic over the CCSA logic,
mechanizing its use to verify protocols with unbounded traces.
This construction enables concise, high-level proofs,
enabling formal trace-based reasoning in the computational model.

- Intuitive style, blends well with mutable state.

What we have achieved so far

Summary

We have a defined a meta-logic over the CCSA logic,
mechanizing its use to verify protocols with unbounded traces.
This construction enables concise, high-level proofs,
enabling formal trace-based reasoning in the computational model.

- Intuitive style, blends well with mutable state.

A fundamental limitation

The provided guarantees are not what a cryptographer would expect.

- Squirrel: for any trace $\mathbb{T}$, for any attacker against $\mathcal{P}_{\mathbb{T}}$, the probability of sucess is negligible.
- Wanted: for any attacker against $\mathcal{P}$ that adaptatively chooses the trace, the probability of success is negligible.

Outline

Global meta-logic formulas

Local meta-logic formula = trace property of single protocol $\mathcal{P}$.
We need to express equivalences, possibly relating several protocols.

Definition (Syntax of global meta-logic formulas)

First-order logic formulas Φ over the following atoms:

- $[\phi]_{\mathcal{P}}$ where ϕ is a local meta-logic formula;
- $[\vec{u} \sim \vec{v}]_{\mathcal{P},\mathcal{P}'}$ where $\vec{u}, \vec{v}$ are same-length sequences of meta-logic terms.

Quantifications allowed over indices, timestamps *and messages*.
Notations: $\tilde{\forall}$, $\tilde{\vee}$... to distinguish from local meta-logic.

Example (Strong secrecy of OSK states)

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

Global meta-logic formulas

Definition (Compatible protocols)

Two protocols are compatible if they have the same trace models: same partially ordered action symbols $(\mathcal{A}, <_{\mathcal{P}})$.

Protocols occurring in a global meta-logic formula must be compatible.

Definition (Semantics of global meta-logic formulas)

A global meta-logic formula Φ interprets in $\mathbb{T}$ as base-logic formula $(\Phi)^{\mathbb{T}}$, with straightforward translation of all logical connectives and:

$$([\phi]_{\mathcal{P}})^{\mathbb{T}} = (\phi)^{\mathbb{T}}_{\mathcal{P}} \sim \mathsf{true}$$

$$([\vec{u} \sim \vec{v}]_{\mathcal{P},\mathcal{P}'})^{\mathbb{T}} = (\vec{u})^{\mathbb{T}}_{\mathcal{P}} \sim (\vec{v})^{\mathbb{T}}_{\mathcal{P}'}$$

The formula is valid when, for all $\mathcal{M}$ and $\mathbb{T}$, we have $\mathcal{M} \models (\Phi)^{\mathbb{T}}$.

Observational equivalence

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

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

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
(**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`

LAK protocol

We can also prove unlinkability for our variant of LAK:

`lak-tags.sp`

Key points:

- Model using find i suchthat … in messages.
- Reasoning about these more complex terms.

Meta-logic sequents

The full proof system relies on two kinds of sequents.

universally quantified variables

$$\Sigma; \Theta \vdash \Phi \quad \text{and} \quad \Sigma; \Theta; \Gamma \vdash_{\mathcal{P}} \phi$$

global meta-formulas

local meta-formulas

Semantics given by $\quad \tilde{\forall}\Sigma.\ (\tilde{\bigwedge}\Theta) \mathrel{\tilde{\Rightarrow}} \Phi \quad$ and $\quad \tilde{\forall}\Sigma.\ (\tilde{\bigwedge}\Theta) \mathrel{\tilde{\Rightarrow}} [(\wedge\Gamma) \Rightarrow \phi]_{\mathcal{P}}.$

Meta-logic sequents

The full proof system relies on two kinds of sequents.

universally quantified variables

$$\Sigma; \Theta \vdash \Phi \quad \text{and} \quad \Sigma; \Theta; \Gamma \vdash_{\mathcal{P}} \phi$$

global meta-formulas

local meta-formulas

Semantics given by $\tilde{\forall}\Sigma.\ (\tilde{\bigwedge}\Theta) \mathrel{\tilde{\Rightarrow}} \Phi$ and $\tilde{\forall}\Sigma.\ (\tilde{\bigwedge}\Theta) \mathrel{\tilde{\Rightarrow}} [(\wedge\Gamma) \Rightarrow \phi]_{\mathcal{P}}$.

Proof system features rules for deriving the two kind of sequents.
Each kind can be useful to derive the other kind.

Proof system (1)

Purely local reasoning using classical inferences, for instance:

$$\frac{\Sigma; \Theta; \Gamma, \phi_1 \vdash_{\mathcal{P}} \psi \quad \Sigma; \Theta; \Gamma, \phi_2 \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta; \Gamma, \phi_1 \vee \phi_2 \vdash_{\mathcal{P}} \psi}$$

Purely global reasoning using classical inferences:

$$\frac{\Sigma; \Theta, \Phi_1; \Gamma \vdash_{\mathcal{P}} \psi \quad \Sigma; \Theta, \Phi_2; \Gamma \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2; \Gamma \vdash_{\mathcal{P}} \psi}$$

$$\frac{\Sigma; \Theta, \Phi_1 \vdash \Psi \quad \Sigma; \Theta, \Phi_2 \vdash \Psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2 \vdash \Psi}$$

Proof system (2)

From global to local hypotheses:

$$\frac{\Sigma; \Theta; \phi, \Gamma \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta, [\phi]_{\mathcal{P}}; \Gamma \vdash_{\mathcal{P}} \psi}$$

The opposite direction requires that ϕ is deterministic,
i.e. features no names or **att**, even through macros:

$$\frac{\Sigma; \Theta, [\phi]_{\mathcal{P}}; \Gamma \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta; \phi, \Gamma \vdash_{\mathcal{P}} \psi} \qquad \frac{\Sigma; \Theta, [\phi]_{\mathcal{P}} \vdash \Psi \quad \Sigma; \Theta, [\psi]_{\mathcal{P}} \vdash \Psi}{\Sigma; \Theta, [\phi \vee \psi]_{\mathcal{P}} \vdash \Psi}$$

Proof system (2)

From global to local hypotheses:

$$\frac{\Sigma; \Theta; \phi, \Gamma \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta, [\phi]_{\mathcal{P}}; \Gamma \vdash_{\mathcal{P}} \psi}$$

The opposite direction requires that ϕ is deterministic,
i.e. features no names or **att**, even through macros:

$$\frac{\Sigma; \Theta, [\phi]_{\mathcal{P}}; \Gamma \vdash_{\mathcal{P}} \psi}{\Sigma; \Theta; \phi, \Gamma \vdash_{\mathcal{P}} \psi} \qquad \frac{\Sigma; \Theta, [\phi]_{\mathcal{P}} \vdash \Psi \quad \Sigma; \Theta, [\psi]_{\mathcal{P}} \vdash \Psi}{\Sigma; \Theta, [\phi \vee \psi]_{\mathcal{P}} \vdash \Psi}$$

$$\frac{}{\Sigma; \Theta \vdash [\phi]_{\mathcal{P}} \vee [\neg\phi]_{\mathcal{P}}}$$

Proof system (3)

From local to global sequents:

$$\frac{\Sigma; \Theta; \vdash_{\mathcal{P}} u = v \quad \Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{C}[v] \sim \vec{t}}{\Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{C}[u] \sim \vec{t}}$$

From global to local sequents (`rewrite equiv`):

$$\frac{\Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{u} \sim \vec{v} \quad \Sigma; \Theta; \Gamma[\vec{v}] \vdash_{\mathcal{P}'} \phi[\vec{v}]}{\Sigma; \Theta; \Gamma[\vec{u}] \vdash_{\mathcal{P}} \phi[\vec{u}]} \; \Gamma, \phi \text{ without macros/names}$$

Proof system (3)

From local to global sequents:

$$\frac{\Sigma; \Theta; \vdash_{\mathcal{P}} u = v \quad \Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{C}[v] \sim \vec{t}}{\Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{C}[u] \sim \vec{t}}$$

From global to local sequents (`rewrite equiv`):

$$\frac{\Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} \vec{u} \sim \vec{v} \quad \Sigma; \Theta; \Gamma[\vec{v}] \vdash_{\mathcal{P}'} \phi[\vec{v}]}{\Sigma; \Theta; \Gamma[\vec{u}] \vdash_{\mathcal{P}} \phi[\vec{u}]} \; \Gamma, \phi \text{ without macros/names}$$

Decomposes thanks to bi-deduction rule (which also justifies `fadup`):

$$\frac{\Sigma; \Theta \vdash_{\mathcal{P},\mathcal{P}'} (\Gamma \Rightarrow \phi) \sim (\Delta \Rightarrow \psi) \quad \Sigma; \Theta; \Delta \vdash_{\mathcal{P}'} \psi}{\Sigma; \Theta; \Gamma \vdash_{\mathcal{P}} \phi}$$

$$\frac{\exists \mathcal{B} \text{ which computes } [\![\vec{v_i}]\!]_{\mathcal{P}_i} \text{ from } [\![\vec{u_i}]\!]_{\mathcal{P}_i}}{\Sigma; \Theta, [\vec{u_1} \sim \vec{u_2}]_{\mathcal{P}_1,\mathcal{P}_2} \vdash [\vec{v_1} \sim \vec{v_2}]_{\mathcal{P}_1,\mathcal{P}_2}}$$

An advanced example

Strong secrecy for OSK in the random oracle model.

`stateful/running-ex-oracle.sp`

Outline

What's next?

Challenge us with your favorite (small) protocol/mechanism.

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

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

We are looking for postdocs and engineers!

Ongoing work:

- More complex protocols: many toy challenges, Signal as a target.
- More powerful automation using SMT solvers / FO prover.
- Study of translation from pi-calculus processes to systems of actions.
- Formally deriving crypto axioms / tactics from games.
- Theoretical steps towards concrete security and polynomial security.