

# 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 Hérourard, 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.

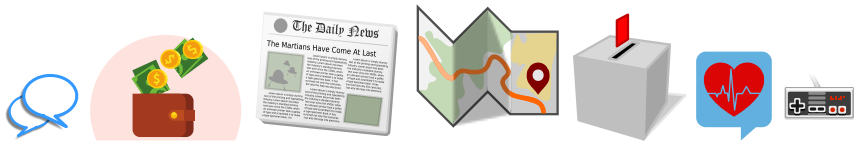
- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
- 3 Reasoning about protocols: local meta-logic
- 4 Global meta-logic: incorporating equivalences
- 5 Conclusion

# Outline

- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
- 3 Reasoning about protocols: local meta-logic
- 4 Global meta-logic: incorporating equivalences
- 5 Conclusion

# Security & Privacy

Increasingly many activities are becoming digitalized.



# Security & Privacy

Increasingly many activities are becoming digitalized.

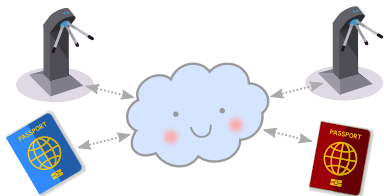


These systems must ensure important properties:

- **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.

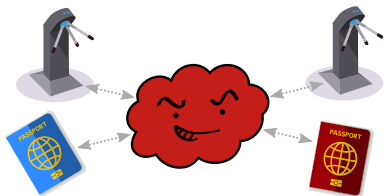
$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$

$R \rightarrow T_i : \text{ok}$

Scenario under consideration:

- roles  $R, T_1, \dots, T_n$ ; 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.

$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$

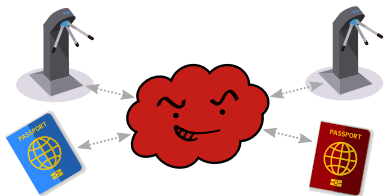
$R \rightarrow T_i : \text{ok}$

Scenario under consideration:

- roles  $R, T_1, \dots, T_n$ ; 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.

$$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$$
$$R \rightarrow T_i : \text{ok}$$

Scenario under consideration:

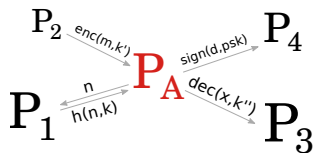
- roles  $R, T_1, \dots, T_n$ ; arbitrary number of sessions for each role
- attacker can intercept messages, inject new messages

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

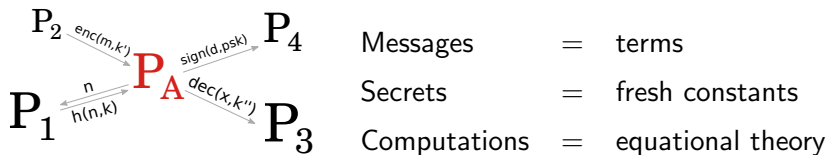
= fresh constants

Computations

= equational theory

# Symbolic model

An idealized setting, also known as Dolev-Yao model

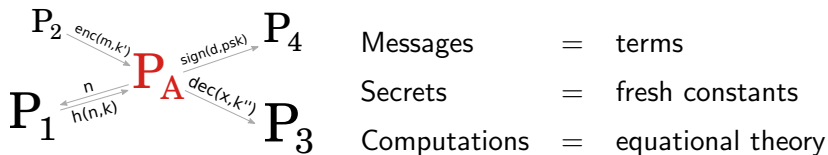


## Example (Equational theories)

- **Symmetric encryption:**  $sdec(senc(x, y), y) =_E x$ .
- **Hash function:** no equation.

# Symbolic model

An idealized setting, also known as Dolev-Yao model

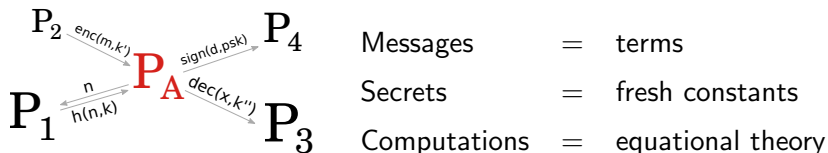


## Example (Equational theories)

- **Symmetric encryption:**  $sdec(senc(x, y), y) =_E x$ .
- **Hash function:** no equation.

# Symbolic model

An idealized setting, also known as Dolev-Yao model



## Example (Equational theories)

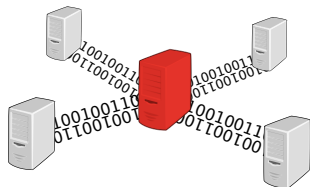
- **Symmetric encryption:**  $sdec(senc(x, y), y) =_E x$ .
- **Hash function:** no equation.

## Example (Basic Hash in the symbolic model)

Informally, both authentication and privacy hold.

# Computational model

The cryptographer's mathematical model for provable security



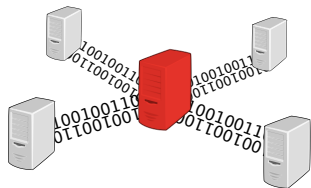
Messages = bitstrings

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

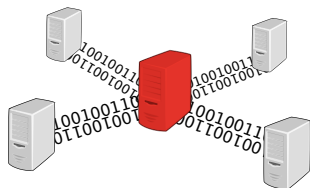
Secrets = random samplings

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  $\eta^{-k}$ .

# 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  $\eta^{-k}$ .

## 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  $x \mapsto \mathbf{h}(x, k)$ .
- Succeed if  $u = \mathbf{h}(v, k)$  and  $\mathcal{O}$  has not been called on  $v$ .



## Basic Hash in the computational model $T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$

### 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

$$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$$

### Authentication

Attacker can interact with tags and readers,

**wins** if some reader accepts a message that has not been emitted by a tag.

### Example (Basic Hash, when **h** is unforgeable)

Assume reader accepts some  $m$ :  $\text{snd}(m) = h(\text{fst}(m), k_i)$  for some  $i$ .

By unforgeability,  $\text{fst}(m) = n_T$  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

$$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$$

### 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

$$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$$

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

## Basic Hash in the computational model

$$T_i \rightarrow R : \langle n_T, h(n_T, k_i) \rangle$$

### 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:

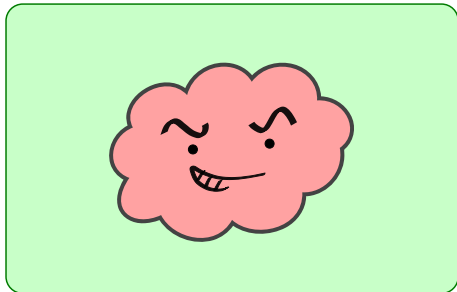
- Draw  $k_1, \dots, k_n$  uniformly at random. Flip a coin  $b$ .
- Consider oracles  $\mathcal{O}_i(x) = (\text{if } b \text{ then } h(x, k_i) \text{ else random}())$  that can only be queried once per message.
- Succeed if  $b = \mathcal{A}^{\mathcal{O}_1, \dots, \mathcal{O}_n}$ .

### 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	<b>Squirrel</b>	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

- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
  - Syntax and semantics
  - Axioms
  - Mechanization
- 3 Reasoning about protocols: local meta-logic
- 4 Global meta-logic: incorporating equivalences
- 5 Conclusion

# Terms of the CCSA logic (informally)

First-order terms interpreted as **probabilistic computations of bitstrings**.

## Names

Special constants used to represent random samplings.

Notation:  $n, r, k \dots$

## Honest functions symbols

Function symbols used to represent primitives, public constants. . .

Notation:  $f(m), g(m, n), ok \dots$

## Adversarial function symbols

Function symbols used to represent attacker computations.

Notation:  $\mathbf{att}(m_1, \dots, m_k)$ .

## Example

In reasonable models where  $h$  is a hash function,

$\mathbf{att}(h(\mathbf{true}, k))$  and  $h(\mathbf{false}, k)$  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:

- $\rho_h$  for **honest** samplings (by the protocol)
- $\rho_a$  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  $\sigma$  mapping variables to 2-PPTMs, we interpret any term  $t$  as a 2-PPTM  $\llbracket t \rrbracket_{\mathcal{M}}^{\sigma}$ :

- $\llbracket x \rrbracket_{\mathcal{M}}^{\sigma} = \sigma(x)$

## Terms of the CCSA logic (formally)

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

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

Given a semantic assignment  $\sigma$  mapping variables to 2-PPTMs, we interpret any term  $t$  as a 2-PPTM  $\llbracket t \rrbracket_{\mathcal{M}}^{\sigma}$ :

- $\llbracket x \rrbracket_{\mathcal{M}}^{\sigma} = \sigma(x)$
- $\llbracket \mathbf{n} \rrbracket_{\mathcal{M}}^{\sigma}(1^{\eta}, \rho_h, \rho_a) \stackrel{\text{def}}{=} \rho_h[\iota(\mathbf{n}) \times \eta, \iota(\mathbf{n}) \times (\eta + 1) - 1]$



## Terms of the CCSA logic (formally)

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

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

Given a semantic assignment  $\sigma$  mapping variables to 2-PPTMs, we interpret any term  $t$  as a 2-PPTM  $\llbracket t \rrbracket_{\mathcal{M}}^{\sigma}$ :

- $\llbracket x \rrbracket_{\mathcal{M}}^{\sigma} = \sigma(x)$
- $\llbracket \mathbf{n} \rrbracket_{\mathcal{M}}^{\sigma}(\mathbf{1}^{\eta}, \rho_h, \rho_a) \stackrel{\text{def}}{=} \rho_h[\iota(\mathbf{n}) \times \eta, \iota(\mathbf{n}) \times (\eta + 1) - 1]$
- $\llbracket \mathbf{f}(t_1, \dots, t_k) \rrbracket_{\mathcal{M}}^{\sigma}(\mathbf{1}^{\eta}, \rho_h, \rho_a) \stackrel{\text{def}}{=} \mathbf{f}_{\mathcal{M}}(\llbracket t_1 \rrbracket_{\mathcal{M}}^{\sigma}(\mathbf{1}^{\eta}, \rho_h, \rho_a), \dots, \llbracket t_k \rrbracket_{\mathcal{M}}^{\sigma}(\mathbf{1}^{\eta}, \rho_h, \rho_a), \mathbf{1}^{\eta})$

## Terms of the CCSA logic (formally)

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

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

Given a semantic assignment  $\sigma$  mapping variables to 2-PPTMs, we interpret any term  $t$  as a 2-PPTM  $\llbracket t \rrbracket_{\mathcal{M}}^{\sigma}$ :

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

## Terms of the CCSA logic (examples)

### Example (determinism, independence)

- $h(\text{cst})$  is interpreted as a deterministic computation:  
$$\llbracket h(\text{cst}) \rrbracket_{\mathcal{M}}(1^\eta, \rho_h, \rho_a) = \llbracket h(\text{cst}) \rrbracket_{\mathcal{M}}(1^\eta, \rho'_h, \rho'_a)$$
 for any  $\rho_h, \rho_a, \rho'_h, \rho'_a$
- however,  $\text{att}(\text{cst})$  may depend on the random tape  $\rho_a$

## Terms of the CCSA logic (examples)

### Example (determinism, independence)

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

## Terms of the CCSA logic (examples)

### Example (determinism, independence)

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

## Terms of the CCSA logic (examples)

### Example (determinism, independence)

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

## Terms of the CCSA logic (examples)

### Example (determinism, independence)

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

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

### Example (boolean builtins)

- $\mathbf{n} \stackrel{\bullet}{\neq} \mathbf{m}$  is true with overwhelming probability  $(1 - 2^{-\eta})$  for distinct names
- $(u \stackrel{\bullet}{=} v) \stackrel{\bullet}{\Rightarrow} (v \stackrel{\bullet}{=} w) \stackrel{\bullet}{\Rightarrow} (u \stackrel{\bullet}{=} v)$  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  $\eta$  for any 1-PPTM  $\mathcal{A}$ :

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

(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  $\eta$  for any 1-PPTM  $\mathcal{A}$ :

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

(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)

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

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

- $(x \stackrel{\cdot}{=} y) \sim \text{true} \models x \sim y$  but not the converse

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

- $(x \stackrel{\cdot}{=} y) \sim \text{true} \models x \sim y$  but not the converse
- indeed,  $m \sim n$  but  $(m \stackrel{\cdot}{=} n) \sim \text{false}$  assuming  $m, n$  distinct

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

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

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

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

### Example (relating boolean connectives)

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

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

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

### Example (relating boolean connectives)

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



## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

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

### Example (relating boolean connectives)

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

## Example formulas

### Example (indistinguishability over booleans)

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

### Example (equality and indistinguishability)

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

### Example (relating boolean connectives)

- $(\phi \dot{\vee} \psi) \sim \text{true} \Leftrightarrow (\phi \sim \text{true}) \vee (\psi \sim \text{true})$  is valid
- $(\phi \dot{\wedge} \psi) \sim \text{true} \Leftrightarrow (\phi \sim \text{true}) \wedge (\psi \sim \text{true})$  is valid
- $(\phi \dot{\Rightarrow} \psi) \sim \text{true} \Rightarrow (\phi \sim \text{true}) \Rightarrow (\psi \sim \text{true})$  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.
- $\vec{u}_1, \vec{u}_2 \sim \vec{v}_1, \vec{v}_2 \Rightarrow \vec{u}_1, f(\vec{u}_2) \sim \vec{v}_1, f(\vec{v}_2)$       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.
- $\vec{u}_1, \vec{u}_2 \sim \vec{v}_1, \vec{v}_2 \Rightarrow \vec{u}_1, f(\vec{u}_2) \sim \vec{v}_1, f(\vec{v}_2)$       function application (FA)
- $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, n \sim \vec{v}, m$  when  $\vec{u}, \vec{v}$  are closed and do not contain  $n, m$
- $(t \doteq n) \sim \text{false}$  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.
- $\vec{u}_1, \vec{u}_2 \sim \vec{v}_1, \vec{v}_2 \Rightarrow \vec{u}_1, f(\vec{u}_2) \sim \vec{v}_1, f(\vec{v}_2)$       function application (FA)
- $\vec{u} \sim \vec{v} \Rightarrow \vec{u}, n \sim \vec{v}, m$  when  $\vec{u}, \vec{v}$  are closed and do not contain  $n, m$
- $(t \doteq n) \sim \text{false}$  when  $t$  is closed and does not contain  $n$

## Implementation axioms

Valid in models featuring reasonable implementations of some primitives.

Example:  $\forall x, y. (\text{fst}(\text{pair}(x, y)) \doteq x) \sim \text{true}$  and similarly for  $\text{snd}$ .

## 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:

$$\text{true} \sim ( u \dot{=} h(v, k) \dot{\Rightarrow} (\dot{\forall}_{s \in S} s \dot{=} v) )$$

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



# Crypto axioms

## Example (Unforgeability)

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

$$\text{true} \sim ( u \dot{=} h(v, k) \dot{\Rightarrow} (\dot{\forall}_{s \in S} s \dot{=} v) )$$

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

## Proof.

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

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

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

# Crypto axioms

## Example (Unforgeability)

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

$$\text{true} \sim ( u \dot{=} h(v, k) \Rightarrow (\dot{\forall}_{s \in S} s \dot{=} v) )$$

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

## Example (Pseudo-randomness)

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

$$\vec{v}, h(t, k) \sim \vec{v}, \text{ if } \dot{\forall}_{s \in S} s \dot{=} t \text{ then } h(t, k) \text{ else } n$$

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  $h(-, k)$ .

# Crypto axioms

## Example (Unforgeability)

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

$$\text{true} \sim ( u \doteq h(v, k) \Rightarrow (\dot{\forall}_{s \in S} s \doteq v) )$$

where  $S = \{ s \mid h(s, k) \text{ occurs in } u, v \}$  and

$s, t$  are closed terms only containing  $k$  as  $h(-, k)$ .

## Example (Pseudo-randomness)

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

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

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  $h(-, k)$ .

## 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.  $\square$

# 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  $\phi \sim \text{true}$  we use sequent calculus, pretending these terms are formulas:

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

# 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  $\phi \sim \text{true}$  we use sequent calculus, pretending these terms are formulas:

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

- 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  $Ax \models \varphi_t$  but:

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

$$\text{base logic} \quad \varphi_{t_1}, \varphi_{t_2}, \dots \quad + \quad \frac{\varphi' \quad \varphi''}{\varphi} \quad = \quad \pi_{t_1}, \pi_{t_2}, \dots$$

## 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  $Ax \models \varphi_t$  but:

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

⇒ Develop a **meta-logic**

meta-logic

$\phi$

$\Downarrow$

base logic

$$\varphi_{t_1}, \varphi_{t_2}, \dots \quad + \quad \frac{\varphi' \quad \varphi''}{\varphi} \quad = \quad \pi_{t_1}, \pi_{t_2}, \dots$$

## 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  $Ax \models \varphi_t$  but:

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

↪ Develop a **meta-logic** suitable for interactive proofs, independent of  $b$ .

$$\begin{array}{ccccccc} \text{meta-logic} & \phi & + & \frac{\phi' \quad \phi''}{\phi} & = & \Pi \\ & \Downarrow & & \Downarrow & & \Downarrow \\ \text{base logic} & \varphi_{t_1}, \varphi_{t_2}, \dots & + & \frac{\varphi' \quad \varphi''}{\varphi} & = & \pi_{t_1}, \pi_{t_2}, \dots \end{array}$$



# Outline

- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
- 3 Reasoning about protocols: local meta-logic
  - Syntax
  - Semantics
  - Lifting axioms to the meta-logic
  - Protocols with dependencies and state
- 4 Global meta-logic: incorporating equivalences
- 5 Conclusion

## 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  $k(i)$ .

**Syntax:**  $i, j, k \dots \in \mathcal{X}_{\mathcal{I}}$

**Atoms over indices:**  $i = j$

### Timestamps

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

$T ::= \tau \mid \text{init} \mid \text{pred}(T) \mid A(\vec{i})$

$\tau \in \mathcal{X}_{\mathcal{T}}, A \in \mathcal{A}$

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

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  $\text{macro}@T$ .

### Messages

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

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  $\text{macro}@T$ .

### Messages

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

Some constructs ignored for simplicity.

### Formulas

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

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

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

## Local meta-logic formulas: examples

### Example (Input validation for Basic Hash)

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

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

## Local meta-logic formulas: examples

### Example (Input validation for Basic Hash)

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

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

### Example

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

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

## Local meta-logic formulas: examples

### Example (Input validation for Basic Hash)

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

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

### Example

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

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

### Intuitive semantics

We are now reasoning **over all traces** and all implementations of functions.

For a given **trace model**  $\mathbb{T}$ , a formula  $\phi$  becomes  $(\phi)^{\mathbb{T}} \sim \text{true}$ :

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



# Modelling protocols

## Definition (Action descriptions)

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

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

condition                  output  
(local formula)    (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  $A \in \mathcal{A}$  is given by an expression of the form

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

$\nearrow$  condition                       $\uparrow$  output  
(local formula)      (message term)

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

## Example (Basic Hash, over $T(i, k), R(j, i), R_1(j)$ )

Session  $k$  of tag  $T_i$ :

$$T(i, k).(\text{true}, \langle n_T(i, k), h(n_T(i, k), k(i)) \rangle)$$

## Modelling protocols

### Definition (Action descriptions)

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

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

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

### Example (Basic Hash, over $T(i, k), R(j, i), R_1(j)$ )

Reader session  $j$  identifies its input coming from tag  $T_i$ :

$$R(j, i).(\text{snd}(\text{input}@R(j, i)) = \text{h}(\text{fst}(\text{input}@R(j, i)), k(i)), \text{ok})$$



## 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  $A(\vec{i}).(\phi_{A(\vec{i})}, \sigma_{A(\vec{i})})$  is `input@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  $A(\vec{i}).(\phi_{A(\vec{i})}, \sigma_{A(\vec{i})})$  is `input@A( $\vec{i}$ )`.

## Definition (Trace model)

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

- an index domain  $\mathcal{D}_{\mathcal{I}} \subseteq_{\text{fin}} \mathbb{N}$ ;
- a timestamp domain  $\mathcal{D}_{\mathcal{T}} \subseteq \{\text{init}, \text{undef}\} \cup \{A(\vec{n}) \mid A \in \mathcal{A}, \vec{n} \in \mathcal{D}_{\mathcal{I}}^{|\vec{n}|}\}$ ;
- a total order  $<_{\mathcal{T}}$  over  $\mathcal{D}_{\mathcal{T}} \setminus \{\text{undef}\}$  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  $A(\vec{i}).(\phi_{A(\vec{i})}, \sigma_{A(\vec{i})})$  is `input@A( $\vec{i}$ )`.

## Definition (Trace model)

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

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

## Example

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

## Semantics of local meta-logic

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

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

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

Key cases:

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

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

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

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

If  $\sigma_{\mathcal{I}}(i) = 3$  then  $(h(n(i, i), k(i)))_{\mathcal{P}}^{\mathbb{T}} = h(n_{3,3}, k_3)$ .



## Semantics of local meta-logic

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

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

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

Key cases:

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

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

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

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

## Semantics of local meta-logic

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

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

message term  $t \rightsquigarrow$  base logic term  $(t)_{\mathcal{P}}^{\mathbb{T}}$

index and timestamp terms  $\rightsquigarrow$  elements of  $\mathcal{D}_{\mathcal{I}}$  and  $\mathcal{D}_{\mathcal{T}}$

formula  $\phi \rightsquigarrow$  base logic boolean term  $(\phi)_{\mathcal{P}}^{\mathbb{T}}$

Key cases:

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

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

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

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

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

## Semantics of local meta-logic

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

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

message term  $t \rightsquigarrow$  base logic term  $(t)_{\mathcal{P}}^{\mathbb{T}}$

index and timestamp terms  $\rightsquigarrow$  elements of  $\mathcal{D}_{\mathcal{I}}$  and  $\mathcal{D}_{\mathcal{T}}$

formula  $\phi \rightsquigarrow$  base logic boolean term  $(\phi)_{\mathcal{P}}^{\mathbb{T}}$

Key cases:

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

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

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

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

$$(\text{output}@T(i, i))_{\mathcal{P}}^{\mathbb{T}\{i \mapsto 1\}} = \langle n_{1,1}, h(n_{1,1}, k_1) \rangle$$

## Semantics of local meta-logic

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

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

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

Key cases:

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

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

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

$$(\text{frame@}T(i, j))_{\mathcal{P}}^{\mathbb{T}\{i \mapsto 1, j \mapsto 3\}} = \langle \text{empty}, \langle \dots, \text{if } \dots \text{ then } n_{1,3}, h(n_{1,3}, k_1) \rangle \rangle$$

## Semantics of local meta-logic

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

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

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

Key cases:

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

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

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

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

$$(\text{frame}@T(i, j))_{\mathcal{P}}^{\mathbb{T}\{i \mapsto 1, j \mapsto 3\}} = \langle \text{empty}, \langle \text{true}, \text{if true then } n_{1,3}, h(n_{1,3}, k_1) \rangle \rangle$$

# Axioms of trace models

## Example (Actions)

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

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

# Axioms of trace models

## Example (Actions)

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

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

## Example (Order on timestamps)

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

# Axioms of trace models

## Example (Actions)

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

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

## Example (Order on timestamps)

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



# Axioms of trace models

## Example (Actions)

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

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

## Example (Order on timestamps)

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

## Example (Case analysis and induction)

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

## Lifting axioms to the meta-logic

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

## Lifting axioms to the meta-logic

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

### Example (Freshness in base logic)

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

### Example (Freshness in meta-logic)

$t \neq n(i)$  is valid for any term  $t$  such that

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

## Lifting axioms to the meta-logic

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

### Example (Freshness in base logic)

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

### Example (Freshness in meta-logic)

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

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

## Lifting axioms to the meta-logic

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

### Example (Freshness in base logic)

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

### Example (Freshness in meta-logic)

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

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

## Lifting axioms to the meta-logic

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

### Example (Freshness in base logic)

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

### Example (Freshness in meta-logic)

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

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

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:

$$\text{true} \sim ( u \doteq h(v, k) \Rightarrow (\dot{\forall}_{h(s,k) \in \text{subterm}(u,v)} s \doteq v) )$$

where  $u, v$  are closed terms only containing  $k$  as  $h(-, k)$ .

## Lifting axioms to the meta-logic

### Example (Unforgeability in base logic)

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

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

where  $u, v$  are closed terms only containing  $k$  as  $h(-, k)$ .

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



## Lifting axioms to the meta-logic

### Example (Unforgeability in base logic)

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

$$\text{true} \sim ( u \dot{=} h(v, k) \Rightarrow (\dot{\forall}_{h(s,k) \in \text{subterm}(u,v)} s \dot{=} v) )$$

where  $u, v$  are closed terms only containing  $k$  as  $h(-, k)$ .

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

### Example (Unforgeability in meta-logic)

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

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

when  $u, v$  contain no message variable and, for all  $\mathbb{T}$ ,  $(k(\vec{i}))^{\mathbb{T}}$  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  $A(\vec{i}) <_{\mathcal{P}} B(\vec{j})$  implies  $A(\sigma(\vec{i})) <_{\mathcal{P}} B(\sigma(\vec{j}))$  for any  $\sigma : \mathcal{X}_{\mathcal{I}} \rightarrow \mathcal{X}_{\mathcal{I}}$ . Elements in action descriptions of  $A(\vec{i})$  can also mention macros  $m@pred^n(B(\vec{j}))$  for  $B(\vec{j}) < A(\vec{i})$ .

## 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  $A(\vec{i}) <_{\mathcal{P}} B(\vec{j})$  implies  $A(\sigma(\vec{i})) <_{\mathcal{P}} B(\sigma(\vec{j}))$  for any  $\sigma : \mathcal{X}_{\mathcal{I}} \rightarrow \mathcal{X}_{\mathcal{I}}$ . Elements in action descriptions of  $A(\vec{i})$  can also mention macros  $m@pred^n(B(\vec{j}))$  for  $B(\vec{j}) < A(\vec{i})$ .

### Definition (Trace model, continued)

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

### Example (More axioms)

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

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

## Application: LAK protocol (variant)

$$\begin{aligned} R &\rightarrow T : nR \\ T &\rightarrow R : \langle nT, h(\langle nR, nT, \text{tag1} \rangle, \text{key}) \rangle \\ R &\rightarrow T : \dots \end{aligned}$$

### Actions for LAK

- $T(i, j)$  for session  $j$  of  $T_i$
- $R(k)$  for the first output of reader session  $k$
- $R_1(k, i)$  when reader session  $k$  accepts tag  $i$   $R(k) < R_a(k)$
- $R_2(k)$  when reader session  $k$  rejects  $R(k) < R_r(k)$



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  $s(\vec{i})@T$ .

### Definition (Protocol, continued)

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

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

The interpretation is naturally extended to handle state macros.

### Example

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

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

## Application: OSK protocol (variant)

An RFID protocol where tags update their state:

- Tag  $T_i$  maintains a state  $s(i)$  initialized with  $s_0(i)$ .
- Each session of  $T_i$  updates  $s(i) := h(s(i), k)$  and outputs  $g(s(i), k')$ .
- Readers know all initial values, accept all inputs  $g(h^n(s_0(i), k), k')$ .



stateful/running-ex-oracle.sp



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

# What we have achieved so far

## Summary

We have 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 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 success is negligible.
- Wanted: for any attacker against  $\mathcal{P}$  that adaptatively chooses the trace, the probability of success is negligible.

# Outline

- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
- 3 Reasoning about protocols: local meta-logic
- 4 **Global meta-logic: incorporating equivalences**
  - Syntax and semantics
  - Proof systems
  - An advanced example: OSK
- 5 Conclusion

## 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  $\Phi$  over the following atoms:

- $[\phi]_{\mathcal{P}}$  where  $\phi$  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:**  $\tilde{\forall}, \tilde{\exists} \dots$  to distinguish from local meta-logic.

### Example (Strong secrecy of OSK states)

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

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

$$\begin{aligned}([\phi]_{\mathcal{P}})^{\mathbb{T}} &= (\phi)_{\mathcal{P}}^{\mathbb{T}} \sim \text{true} \\ ([\vec{u} \sim \vec{v}]_{\mathcal{P}, \mathcal{P}'})^{\mathbb{T}} &= (\vec{u})_{\mathcal{P}}^{\mathbb{T}} \sim (\vec{v})_{\mathcal{P}'}^{\mathbb{T}}\end{aligned}$$

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:

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

## 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

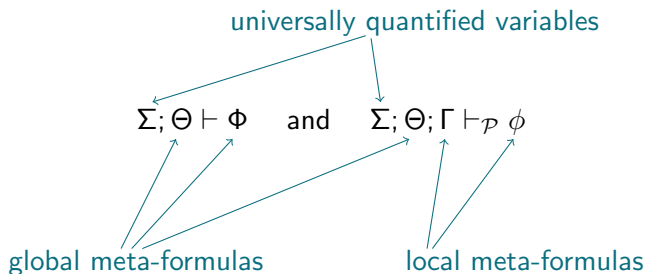


## Key points:

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

# Meta-logic sequents

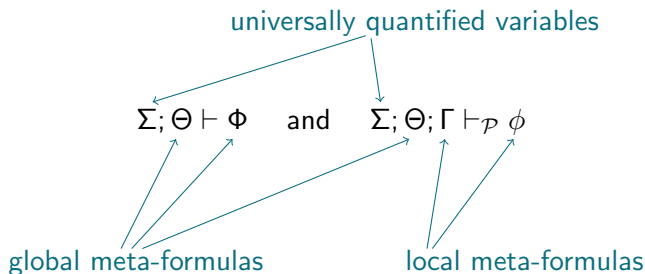
The full proof system relies on **two kinds of sequents**.



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

# Meta-logic sequents

The full proof system relies on **two kinds of sequents**.



Semantics given by  $\forall \Sigma. (\tilde{\Lambda} \Theta) \Rightarrow \phi$     and     $\forall \Sigma. (\tilde{\Lambda} \Theta) \Rightarrow [(\wedge \Gamma) \Rightarrow \phi]_P$ .

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  $\phi$  is **deterministic**,  
i.e. features no names or **att**, even through macros:

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

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

## 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  $\phi$  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**):

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



## 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**):

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

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

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

$$\frac{\exists \mathcal{B} \text{ which computes } \llbracket \vec{v}_i \rrbracket_{\mathcal{P}_i} \text{ from } \llbracket \vec{u}_i \rrbracket_{\mathcal{P}_i}}{\Sigma; \Theta, [\vec{u}_1 \sim \vec{u}_2]_{\mathcal{P}_1, \mathcal{P}_2} \vdash [\vec{v}_1 \sim \vec{v}_2]_{\mathcal{P}_1, \mathcal{P}_2}}$$

## An advanced example

Strong secrecy for OSK in the random oracle model.



`stateful/running-ex-oracle.sp`



# Outline

- 1 Background: verifying security protocols
- 2 Reasoning about messages: the CCSA logic
- 3 Reasoning about protocols: local meta-logic
- 4 Global meta-logic: incorporating equivalences
- 5 Conclusion**

# 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.