

A Higher-Order Indistinguishability Logic for Cryptographic Reasoning

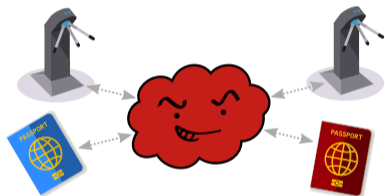
David Baelde Adrien Koutsos Joseph Lallemand

Univ Rennes, CNRS, IRISA & Inria Paris



LICS 2023, Boston

Reasoning about cryptographic protocols



Computational model

Probabilistic polynomial time machines (PPTM):

- Secrets = long-enough random bitstrings, size η .
- PTIME prevents brute force attacks.

Reasoning up to **negligible probability of failure**:
 $P(\eta)$ asymptotically smaller than any η^{-k} .

CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

First-order terms interpreted as PPTMs, explicit random tape $\rho \in \{0, 1\}^\infty$

- $\llbracket t \rrbracket_{\mathcal{M}}(\eta, \rho) \in \{0, 1\}^*$
- **Name constants** n, m, k, \dots extract from ρ dedicated sections of length η .

Example

$\text{att}_2(m, h(\text{att}_1(m), k))$: attacker computes 2nd message from m and hash of first message.

CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

First-order terms interpreted as PPTMs, explicit random tape $\rho \in \{0, 1\}^\infty$

- $\llbracket t \rrbracket_{\mathcal{M}}(\eta, \rho) \in \{0, 1\}^*$
- **Name constants** n, m, k, \dots extract from ρ dedicated sections of length η .

Predicates

- $t \sim t'$: “ t and t' are computationally indistinguishable”.
- $[\varphi]$ where φ is a boolean term : “ φ false with negligible probability”.

Example

$[n \neq \text{att}_2(m, h(\text{att}_1(m), k))]$ is valid.

CCSA meta-logic: reasoning about protocols

Foundation of the Squirrel proof-assistant [BDJKM, SP'21 & BDKM, CSF'22]

CCSA meta-logic: reasoning about protocols

Foundation of the Squirrel proof-assistant [BDJKM, SP'21 & BDKM, CSF'22]

Example (Local meta-logic formulas)

- $\text{output}@A(i, j) = \text{h}(\text{input}@A(i, j), \text{k}(i))$

CCSA meta-logic: reasoning about protocols

Foundation of the Squirrel proof-assistant [BDJKM, SP'21 & BDKM, CSF'22]

Example (Local meta-logic formulas)

- $\text{output@A}(i, j) = \text{h}(\text{input@A}(i, j), \text{k}(i))$
- $\forall k. \text{cond@B}(k) \Rightarrow \exists i, j. \text{A}(i, j) < \text{B}(k) \wedge \text{input@B}(k) = \text{output@A}(i, j)$

CCSA meta-logic: reasoning about protocols

Foundation of the Squirrel proof-assistant [BDJKM, SP'21 & BDKM, CSF'22]

Example (Local meta-logic formulas)

- $\text{output@A}(i, j) = \text{h}(\text{input@A}(i, j), \text{k}(i))$
- $\forall k. \text{cond@B}(k) \Rightarrow \exists i, j. \text{A}(i, j) < \text{B}(k) \wedge \text{input@B}(k) = \text{output@A}(i, j)$

Reasoning **over all trace models** \mathbb{T} for protocol \mathcal{P} , and **all computational models** \mathcal{M} .

Meta-logic term $t \xrightarrow{\mathbb{T}}$ base logic term $(t)^{\mathbb{T}} \xrightarrow{\mathcal{M}}$ PPTM returning bitstring

Local meta-logic formula $\varphi \xrightarrow{\mathbb{T}}$ base logic term $(\varphi)^{\mathbb{T}} \xrightarrow{\mathcal{M}}$ PPTM returning boolean

Global meta-logic formulas are first-order formulas over $[\varphi]$ and $t \sim t'$ atoms.

CCSA meta-logic: reasoning about protocols

Foundation of the Squirrel proof-assistant [BDJKM, SP'21 & BDKM, CSF'22]

Example (Local meta-logic formulas)

- $\text{output@A}(i, j) = \text{h}(\text{input@A}(i, j), \text{k}(i))$
- $\forall k. \text{cond@B}(k) \Rightarrow \exists i, j. \text{A}(i, j) < \text{B}(k) \wedge \text{input@B}(k) = \text{output@A}(i, j)$

Reasoning **over all trace models** \mathbb{T} for protocol \mathcal{P} , and **all computational models** \mathcal{M} .

Meta-logic term $t \xrightarrow{\mathbb{T}}$ base logic term $(t)^{\mathbb{T}} \xrightarrow{\mathcal{M}}$ PPTM returning bitstring

Local meta-logic formula $\varphi \xrightarrow{\mathbb{T}}$ base logic term $(\varphi)^{\mathbb{T}} \xrightarrow{\mathcal{M}}$ PPTM returning boolean

Global meta-logic formulas are first-order formulas over $[\varphi]$ and $t \sim t'$ atoms.

Quantifiers in local meta-logic formulas

- Only allowed over **index** and **timestamp**, which are interpreted in finite domains.
- Quantifications translate to finite boolean combinations.

Higher-order CCSA logic

Letting go (at first) of PTIME, computability, bitstrings, protocols. . .

Terms of the old base logic:
probabilistic polynomial-time machines.

$$\llbracket t \rrbracket_{\mathcal{M}}(\eta, \rho) \in \{0, 1\}^*$$

Terms of the new logic:
 η -indexed families of random variables.

$$\llbracket t \rrbracket_{\mathcal{M}}^{\eta, \rho} \in \llbracket \tau \rrbracket_{\mathcal{M}}^{\eta}$$

Higher-order CCSA logic

Letting go (at first) of PTIME, computability, bitstrings, protocols...

Terms of the old base logic:
probabilistic polynomial-time machines.

$$\llbracket t \rrbracket_{\mathcal{M}}(\eta, \rho) \in \{0, 1\}^*$$

Terms of the new logic:
 η -indexed families of random variables.

$$\llbracket t \rrbracket_{\mathcal{M}}^{\eta, \rho} \in \llbracket \tau \rrbracket_{\mathcal{M}}^{\eta}$$

Benefits

- Quantifiers at all types in local formulas, e.g. $\forall_{\tau} : (\tau \rightarrow \text{bool}) \rightarrow \text{bool}$:

$$\llbracket \forall_{\tau}(\lambda x : \tau. \varphi) \rrbracket_{\mathcal{M}}^{\eta, \rho} = \text{true} \quad \text{when} \quad \llbracket \lambda x : \tau. \varphi \rrbracket_{\mathcal{M}}^{\eta, \rho} = a \in \llbracket \tau \rrbracket_{\mathcal{M}}^{\eta} \mapsto \text{true}$$

Higher-order CCSA logic

Letting go (at first) of PTIME, computability, bitstrings, protocols. . .

Terms of the old base logic:
probabilistic polynomial-time machines.

$$\llbracket t \rrbracket_{\mathcal{M}}(\eta, \rho) \in \{0, 1\}^*$$

Terms of the new logic:
 η -indexed families of random variables.

$$\llbracket t \rrbracket_{\mathcal{M}}^{\eta; \rho} \in \llbracket \tau \rrbracket_{\mathcal{M}}^{\eta}$$

Benefits

- Quantifiers at all types in local formulas, e.g. $\forall_{\tau} : (\tau \rightarrow \text{bool}) \rightarrow \text{bool}$:

$$\llbracket \forall_{\tau}(\lambda x : \tau. \varphi) \rrbracket_{\mathcal{M}}^{\eta; \rho} = \text{true} \quad \text{when} \quad \llbracket \lambda x : \tau. \varphi \rrbracket_{\mathcal{M}}^{\eta; \rho} = a \in \llbracket \tau \rrbracket_{\mathcal{M}}^{\eta} \mapsto \text{true}$$

- Ability to talk about useful non-PTIME functions, e.g. discrete logarithm.
- Express abstract reasoning schemes using HOL, e.g. hybrid argument.

Recovering the meta-logic

Restricting types and terms:

- Types **index** and **timestamp** fixed and finite: $\llbracket \tau \rrbracket_{\mathcal{M}}^{\eta}$ is the same finite set for all η .
- One can restrict some terms to be constant, deterministic, PTIME, adversarial, etc.

$$\mathcal{M} \models \text{const}(t) \quad \text{when} \quad \llbracket t \rrbracket_{\mathcal{M}}^{\eta, \rho} \text{ independent of } \eta, \rho$$

Recovering the meta-logic

Restricting types and terms:

- Types **index** and **timestamp** fixed and finite: $\llbracket \tau \rrbracket_{\mathcal{M}}^{\eta}$ is the same finite set for all η .
- One can restrict some terms to be constant, deterministic, PTIME, adversarial, etc.

$$\mathcal{M} \models \text{const}(t) \quad \text{when} \quad \llbracket t \rrbracket_{\mathcal{M}}^{\eta, \rho} \text{ independent of } \eta, \rho$$

Recursive definitions:

- We allow recursive definitions with a semantical well-foundedness criterion.
- Macros of the meta-logic can be recovered, e.g. $\text{input}_{\mathcal{P}}, \text{output}_{\mathcal{P}} : \text{timestamp} \rightarrow \text{message}$.

Proof system

$\mathcal{E}; \Theta \vdash \Phi$	reads as	$\forall \mathcal{E}. \wedge \Theta \Rightarrow \Phi$	\mathcal{E}	:	environment
$\mathcal{E}; \Theta; \Gamma \vdash \varphi$	reads as	$\forall \mathcal{E}. \wedge \Theta \Rightarrow [\wedge \Gamma \Rightarrow \varphi]$	Θ	:	global formulas (FO formulas)
			Γ, φ	:	local formulas (boolean terms)

$$\frac{\mathcal{E}; \Theta; \Gamma, \varphi_1 \vdash \psi \quad \mathcal{E}; \Theta; \Gamma, \varphi_2 \vdash \psi}{\mathcal{E}; \Theta; \Gamma, \varphi_1 \vee \varphi_2 \vdash \psi}$$

$$\frac{\mathcal{E}; \Theta, [\varphi_1]; \Gamma \vdash \psi \quad \mathcal{E}; \Theta, [\varphi_2]; \Gamma \vdash \psi \quad \mathcal{E}; \Theta \vdash \text{const}(\varphi_1) \vee \text{const}(\varphi_2)}{\mathcal{E}; \Theta, [\varphi_1 \vee \varphi_2]; \Gamma \vdash \psi}$$

Proof system

$\mathcal{E}; \Theta \vdash \Phi$ reads as $\forall \mathcal{E}. \wedge \Theta \Rightarrow \Phi$
 $\mathcal{E}; \Theta; \Gamma \vdash \varphi$ reads as $\forall \mathcal{E}. \wedge \Theta \Rightarrow [\wedge \Gamma \Rightarrow \varphi]$

\mathcal{E} : environment
 Θ : global formulas (FO formulas)
 Γ, φ : local formulas (boolean terms)

$$\frac{\mathcal{E}, x; \Theta \vdash \Phi}{\mathcal{E}; \Theta \vdash \forall x. \Phi} \quad x \notin \mathcal{E} \qquad \frac{\mathcal{E}, x; \Theta; \Gamma \vdash \varphi}{\mathcal{E}; \Theta; \Gamma \vdash \forall x. \varphi} \quad x \notin \mathcal{E}$$

Theorem (Equivalence between local and global quantifiers)

$$\mathcal{M} \models \forall (x : \tau). [\varphi]$$

“for any random variable x over $[\tau]$,
 φ holds almost surely”

iff $\mathcal{M} \models [\forall (x : \tau). \varphi]$

“almost surely,
 φ holds for any value $x \in [\tau]$ ”

Proof system

$\mathcal{E}; \Theta \vdash \Phi$ reads as $\forall \mathcal{E}. \wedge \Theta \Rightarrow \Phi$
 $\mathcal{E}; \Theta; \Gamma \vdash \varphi$ reads as $\forall \mathcal{E}. \wedge \Theta \Rightarrow [\wedge \Gamma \Rightarrow \varphi]$

\mathcal{E} : environment
 Θ : global formulas (FO formulas)
 Γ, φ : local formulas (boolean terms)

$$\frac{\mathcal{E}, x; \Theta \vdash \Phi}{\mathcal{E}; \Theta \vdash \forall x. \Phi} \quad x \notin \mathcal{E} \qquad \frac{\mathcal{E}, x; \Theta; \Gamma \vdash \varphi}{\mathcal{E}; \Theta; \Gamma \vdash \forall x. \varphi} \quad x \notin \mathcal{E}$$

Theorem (Equivalence between local and global quantifiers)

$$\mathcal{M} \models \forall (x : \tau). [\varphi] \qquad \text{iff} \qquad \mathcal{M} \models [\forall (x : \tau). \varphi]$$

*“for any random variable x over $[\tau]$,
 φ holds almost surely”*

*“almost surely,
 φ holds for any value $x \in [\tau]$ ”*

If τ is assumed fixed and finite, this is also equivalent to $\mathcal{M} \models \forall (x : \tau). \text{const}(x) \Rightarrow [\varphi]$.

Advanced axioms

The base logic comes with axioms for reasoning about names and crypto primitives.

E.g. $[n \neq t]$ valid when t is ground and does not contain n .

We lifted these axioms to the meta-logic:

- Occurrence conditions : account for potential macros unrollings.
- Take into account **constant** local formulas conditioning occurrences.

For our higher-order logic, we justify improved axioms from first principles:

- Occurrence conditions : account for unrollings of recursive definitions.
- Take into account **arbitrary** local formulas as conditions.
- **Resulting axioms systematically handle bad occurrences, i.e. corruption.**
Case study on forward secrecy for a DH key exchange.

Conclusion

Higher-order CCSA logic

- Strictly generalizes former CCSA meta-logic and proof system.
- Decouples core logic from protocol-specific declarations and recursive definitions.
- Fragment corresponding to former meta-logic implemented in **Squirrel** proof assistant. All past proof developments have been ported; new ones added.

Future work

- More complex proofs of protocols.
- Proof-theoretical investigations, automated reasoning.
- Modelling other classes of protocols and attacker models.
- Relationship with other works in higher-order crypto.