A Higher-Order Indistinguishability Logic for Cryptographic Reasoning

David Baelde, Adrien Koutsos, Joseph Lallemand

Univ Rennes, CNRS & IRISA   Inria Paris

GT MFS Days 2023

New foundation for the Squirrel prover

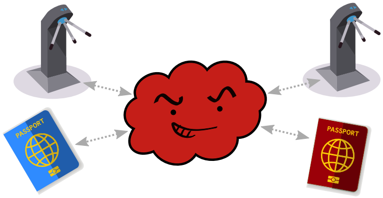
Squirrel is a proof assistant for verifying cryptographic protocols in the computational model.
It is based on the CCSA approach.

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

Outline

- Brief presentation of CCSA base logic, and Squirrel's meta-logic.
- How a higher-order CCSA logic solves several problems.

Example protocol: Basic Hash



Each tag (T_i) owns a secret key k_i.

Reader (R) knows all legitimate keys.

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

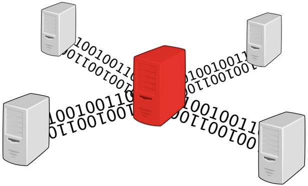
Scenario under consideration:

- Roles R, T_1, ..., T_n with 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.

Cryptographer's model for provable security



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

Rule out unavoidable, unimportant attacks:

- Attacks with negligible probability of success (asymptotically smaller than any η^{-k}).
- Attacks that cannot run in probabilistic polynomial-time.

Cryptographic assumptions

- Keyed hash function may be collision resistant, unforgeable, pseudo-random.
- Encryptions should ensure various forms of indistinguishability.
- Etc.

Outline

Base logic syntax

First-order logic with terms interpreted as probabilistic computations of bitstrings.

- Names n, r, k. . . : special constants represent random samplings.
- Honest function symbols f, enc, ok. . . : represent primitives, public constants, etc.
- Adversarial function symbols $\mathbf{att}_i$ represent attacker computations.

Example

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

Base logic syntax

First-order logic with terms interpreted as probabilistic computations of bitstrings.

- Names n, r, k...: special constants represent random samplings.
- Honest function symbols f, enc, ok...: represent primitives, public constants, etc.
- Adversarial function symbols $\mathbf{att}_i$ represent attacker computations.

Example

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

Formulas built from a single predicate $\vec{u} \sim \vec{v}$ expressing indistinguishability.

Example

$\exists x.\ \ (x \sim \mathsf{ok}) \Rightarrow \bot$

Application to Basic Hash protocol

Simple privacy scenario: T_1, T_2 vs. T_1, T'_1.

$$\begin{aligned} \text{Let } u_1, u_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle,\ \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_2})\rangle \\ v_1, v_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle,\ \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_1})\rangle \end{aligned}$$

Privacy expressed as $u_1, u_2 \sim v_1, v_2$.

Application to Basic Hash protocol

Simple privacy scenario: T_1, T_2 vs. T_1, T'_1.

$$\begin{aligned} \text{Let } u_1, u_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle,\ \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_2})\rangle \\ v_1, v_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle,\ \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_1})\rangle \end{aligned}$$

Privacy expressed as $u_1, u_2 \sim v_1, v_2$.

Authentication for T_1, T_2 expressed as $\phi \sim \mathsf{true}$ with ϕ as follows with $input_R = \mathbf{att}(u_1, u_2)$:

$$\begin{aligned} &\big(\mathsf{snd}(input_R) \doteq \mathsf{h}(\mathsf{fst}(input_R), \mathsf{k}_1) \mathbin{\dot\vee} \\ &\ \mathsf{snd}(input_R) \doteq \mathsf{h}(\mathsf{fst}(input_R), \mathsf{k}_2)\big) \\ \dot\Rightarrow\ &\big(input_R \doteq u_1 \mathbin{\dot\vee} input_R \doteq u_2\big) \end{aligned}$$

We assume some builtin connectives with their expected semantics: $\_ \doteq \_$, $\_ \mathbin{\dot\vee} \_$, $\_ \mathbin{\dot\Rightarrow} \_ \ldots$

Semantics of terms

Definition (Semantics in a model $\mathcal{M}$)

For a term t, $[\![t]\!]_{\mathcal{M}}$ is a deterministic polynomial time Turing machine taking as inputs:

- the security parameter η;
- two infinite randomness tapes $\rho = (\rho_h, \rho_a)$ for honest and attacker samplings.

Random samplings can be tracked syntactically:

- Honest randomness only accessed by names.
- Attacker randomness only accessed by $\mathbf{att}_i$ symbols.
- Variables can access anything!

Example

$\Pr[\ [\![\mathsf{n} \doteq t]\!]_{\mathcal{M}}(\eta, \rho) = \mathsf{true}\ ] = 2^{-\eta}$ if t closed, $\mathsf{n} \notin t$

Semantics of formulas

Definition (Computational indistinguishability)

$\mathcal{M} \models \vec{u} \sim \vec{v}$ when, for any polynomial time Turing machine $\mathcal{D}$,

$$|\ \mathsf{Pr}[\mathcal{D}([\![\vec{u}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] - \mathsf{Pr}[\mathcal{D}([\![\vec{v}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)]\ | \text{ is negligible.}$$

The rest is as usual in first-order logic.

Semantics of formulas

Definition (Computational indistinguishability)

$\mathcal{M} \models \vec{u} \sim \vec{v}$ when, for any polynomial time Turing machine $\mathcal{D}$,

$$|\ \Pr[\mathcal{D}([\![\vec{u}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] - \Pr[\mathcal{D}([\![\vec{v}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)]\ | \text{ is negligible.}$$

The rest is as usual in first-order logic.

Example

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

Semantics of formulas

Definition (Computational indistinguishability)

$\mathcal{M} \models \vec{u} \sim \vec{v}$ when, for any polynomial time Turing machine $\mathcal{D}$,

$$\mid \Pr[\mathcal{D}([\![\vec{u}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] - \Pr[\mathcal{D}([\![\vec{v}]\!]_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] \mid \text{ is negligible.}$$

The rest is as usual in first-order logic.

Example

- $u \sim \mathsf{true}$ means that u is true with overwhelming probability.
- $\mathsf{true} \sim \left(t \mathrel{\dot{\neq}} \mathsf{n} \right)$ is valid when t closed and does not contain n.

Semantics of formulas

Definition (Computational indistinguishability)

$\mathcal{M} \models \vec{u} \sim \vec{v}$ when, for any polynomial time Turing machine $\mathcal{D}$,

$$\mid \Pr[\mathcal{D}(\llbracket \vec{u} \rrbracket_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] - \Pr[\mathcal{D}(\llbracket \vec{v} \rrbracket_{\mathcal{M}}(\eta, \rho_h, \rho_a), \eta, \rho_a)] \mid \text{ is negligible.}$$

The rest is as usual in first-order logic.

Example

- $u \sim \mathsf{true}$ means that u is true with overwhelming probability.
- $\mathsf{true} \sim \left(t \mathrel{\dot{\neq}} \mathsf{n} \right)$ is valid when t closed and does not contain n.
- $\forall \vec{x}, \vec{y}, \vec{x'}, \vec{y'}. \ (\vec{x}, \vec{y} \sim \vec{x'}, \vec{y'}) \Rightarrow (\vec{x}, \mathsf{f}(\vec{y}) \sim \vec{x'}, \mathsf{f}(\vec{y'}))$ is valid for any function symbol f.

Outline

Basic Hash protocol in the meta-logic

Simple privacy scenario: T_1, T_2 vs. T_1, T_1'.

$$\begin{aligned} \text{Let } u_1, u_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle, \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_2})\rangle \\ v_1, v_2 &= \langle \mathsf{n}_t, \mathsf{h}(\mathsf{n}_t, \mathsf{k}_1)\rangle, \langle \mathsf{n}'_t, \mathsf{h}(\mathsf{n}'_t, \boxed{\mathsf{k}_1})\rangle \end{aligned}$$

Privacy expressed as $u_1, u_2 \sim v_1, v_2$.

Basic Hash protocol in the meta-logic

Privacy expressed as $\text{frame}@\tau \sim_{\mathcal{P},\mathcal{P}'} \text{frame}@\tau$.

- Timestamp τ refers to an arbitrary point in any arbitrary execution trace.
- Macro $\text{frame}@\tau$ stands for the sequence of all protocol outputs before τ.
- Protocol $\mathcal{P}$ = multiple tags playing multiple sessions.
- Protocol $\mathcal{P}'$ = multiple tags playing single sessions.

Basic Hash protocol in the meta-logic

Privacy expressed as $\mathsf{frame}@\tau \sim_{\mathcal{P},\mathcal{P}'} \mathsf{frame}@\tau$.

- Timestamp τ refers to an arbitrary point in any arbitrary execution trace.
- Macro $\mathsf{frame}@\tau$ stands for the sequence of all protocol outputs before τ.
- Protocol $\mathcal{P}$ = multiple tags playing multiple sessions.
- Protocol $\mathcal{P}'$ = multiple tags playing single sessions.

Authentication for T_1, T_2 expressed as $\phi \sim \mathsf{true}$ with ϕ as follows with $input_R = \mathbf{att}(u_1, u_2)$:

$$
\begin{aligned}
&\big(\mathsf{snd}(input_R) \doteq \mathsf{h}(\mathsf{fst}(input_R), \mathsf{k}_1) \mathbin{\dot\vee} \\
&\ \mathsf{snd}(input_R) \doteq \mathsf{h}(\mathsf{fst}(input_R), \mathsf{k}_2)\big) \\
\dot\Rightarrow\ &\big(input_R \doteq u_1 \mathbin{\dot\vee} input_R \doteq u_2\big)
\end{aligned}
$$

Basic Hash protocol in the meta-logic

Privacy expressed as $\mathsf{frame}@\tau \sim_{\mathcal{P},\mathcal{P}'} \mathsf{frame}@\tau$.

- Timestamp τ refers to an arbitrary point in any arbitrary execution trace.
- Macro $\mathsf{frame}@\tau$ stands for the sequence of all protocol outputs before τ.
- Protocol $\mathcal{P}$ = multiple tags playing multiple sessions.
- Protocol $\mathcal{P}'$ = multiple tags playing single sessions.

Authentication expressed as meta-logic formula:

$$\forall i.\ \ \mathsf{snd}(\mathsf{input}@\tau) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@\tau), \mathsf{k}(i)) \Rightarrow \exists j.\ \ \mathsf{T}(i,j) < \tau\ \wedge\ \mathsf{input}@\tau = \mathsf{output}@\mathsf{T}(i,j)$$

- Macro $\mathsf{input}@\tau$ defined as $\mathbf{att}(\mathsf{frame}@\tau)$. Macro $\mathsf{output}@\tau$ defined depending on τ.
- Indices i, j are elements of an arbitrary finite set.
- Total order $<$ on timestamps corresponds to execution order.

Meta-logic overview

We are now reasoning over all traces (all trace models $\mathbb{T}$)
and all implementations of functions (all computational models $\mathcal{M}$).

$$\begin{array}{rcr} \text{Meta-logic term } t \xrightarrow{\mathbb{T}} & \text{base logic term } (t)^{\mathbb{T}}_{\mathcal{P}} \xrightarrow{\mathcal{M}} & \text{PPTM returning bitstring} \\ \text{Local meta-logic formula } \phi \xrightarrow{\mathbb{T}} & \text{base logic term } (\phi)^{\mathbb{T}}_{\mathcal{P}} \xrightarrow{\mathcal{M}} & \text{PPTM returning boolean} \\ \text{Global meta-logic formula } \Phi \xrightarrow{\mathbb{T}} & \text{base logic formula } (\Phi)^{\mathbb{T}} \xrightarrow{\mathcal{M}} & \text{SAT/UNSAT} \end{array}$$

Important points

- Local formula quantifications only over indices and timestamps: translated to finite disjunctions and conjunctions once $\mathbb{T}$ fixed.
- Translation of macros depends on $\mathcal{P}$: definition of logic tied to a class of protocols.

Proving meta-logic formulas

Custom proof system:

- hides probabilistic reasoning when proving local formulas;
- most axioms are liftings of base logic axioms.

Proving meta-logic formulas

Custom proof system:

- hides probabilistic reasoning when proving local formulas;
- most axioms are liftings of base logic axioms.

Freshness in base logic

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

Freshness in meta-logic

Local meta-logic formula $\phi \Rightarrow t \neq \mathsf{n}(i)$ is valid if
for any $\mathbb{T}$ such that $(\phi)^{\mathbb{T}} \neq \mathsf{false}$, $(\mathsf{n}(i))^{\mathbb{T}}$ does not occur in $(t)^{\mathbb{T}}$.

Example (Basic Hash)

$\forall i, j, \tau.\ \ \tau < \mathsf{T}(i,j) \Rightarrow \mathsf{n}_t(i,j) \neq \mathsf{fst}(\mathsf{input}@\tau)$

Outline

From machines to random variables

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

$$[\![t]\!]_{\mathcal{M}}(\eta,\rho) \in \{0,1\}^*$$

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

$$[\![t]\!]_{\mathcal{M}}(\eta,\rho) \in [\![\tau]\!]_{\mathcal{M}}$$

From machines to random variables

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

$$[\![t]\!]_{\mathcal{M}}(\eta, \rho) \in \{0,1\}^*$$

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

$$[\![t]\!]_{\mathcal{M}}(\eta, \rho) \in [\![\tau]\!]_{\mathcal{M}}$$

Non-PTIME function symbols

E.g. talking about discrete logarithm is useful in specifications and proofs.

From machines to random variables

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

$$[\![t]\!]_{\mathcal{M}}(\eta, \rho) \in \{0,1\}^*$$

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

$$[\![t]\!]_{\mathcal{M}}(\eta, \rho) \in [\![\tau]\!]_{\mathcal{M}}$$

Non-PTIME function symbols

E.g. talking about discrete logarithm is useful in specifications and proofs.

Arbitrary quantifiers

Easily defined since terms need not be computable:

$$[\![\forall x : \alpha.\ \phi]\!]^{\sigma}_{\mathcal{M}}(\eta, \rho) = \mathsf{true} \quad \text{when} \quad [\![\phi]\!]^{\sigma, x \mapsto a}_{\mathcal{M}}(\eta, \rho) = \mathsf{true} \text{ for all } a \in [\![\alpha]\!]_{\mathcal{M}}.$$

Going higher-order

Quantifications allow to express reasoning steps internally:

$$\forall x, y : \text{message}. \qquad \text{fst}\ \langle x, y\rangle = x$$

Going higher-order

Terms are arbitrary typed λ-terms: functions, functions taking functions taking ...
Not for fun, but because it provides simple, uniform foundations.

Quantifications allow to express reasoning steps internally:

$$\forall x, y : \text{message}. \qquad \text{fst}\ \langle x, y \rangle = x$$

$$\forall f : \alpha \to \beta, c : \text{bool}, x, y : \alpha. \quad (\text{if } c \text{ then } f\ x \text{ else } f\ y) = (f\ (\text{if } c \text{ then } x \text{ else } \ \text{else } y))$$

Going higher-order

Terms are arbitrary typed λ-terms: functions, functions taking functions taking ...
Not for fun, but because it provides simple, uniform foundations.

Quantifications allow to express reasoning steps internally:

$$\forall x, y : \text{message}. \quad \text{fst}\ \langle x, y \rangle = x$$

$$\forall f : \alpha \to \beta, c : \text{bool}, x, y : \alpha. \quad (\text{if } c \text{ then } f\ x \text{ else } f\ y) = (f\ (\text{if } c \text{ then } x \text{ else } \text{ else } y))$$

$$\forall p : \text{timestamp} \to \text{bool}. \quad \big(\forall \tau.\ (\forall \tau'.\ \ \tau' < \tau \Rightarrow p\ \tau') \Rightarrow p\ \tau\big) \Rightarrow \forall \tau.\ \ p\ \tau$$

$$\forall p : \alpha \to \text{bool}. \quad \big(\text{not}\ (\exists a : \alpha.\ \ p\ a)\big) = \big(\forall a : \alpha.\ \text{not}\ (p\ a)\big)$$

Generalizing indistinguishability

Global formulas are still first-order formulas, but over higher-order terms.
Arbitrary predicates can be considered:

- $\vec{u} \sim \vec{v}$: distinguisher still polynomial time.
- $\mathsf{adv}(t)$: t can be computed in polynomial time without access to ρ_h.
- ...

Higher-order indistinguishability allows to decompose old axioms:

$$\forall \vec{x}, \vec{y}, \vec{x'}, \vec{y'}. \qquad \mathsf{adv}(\mathsf{f}) \Rightarrow (\vec{x}, \vec{y} \sim \vec{x'}, \vec{y'}) \Rightarrow (\vec{x}, \mathsf{f}(\vec{y}) \sim \vec{x'}, \mathsf{f}(\vec{y'}))$$

Generalizing indistinguishability

Global formulas are still first-order formulas, but over higher-order terms.
Arbitrary predicates can be considered:

- $\vec{u} \sim \vec{v}$: distinguisher still polynomial time, takes functions (oracles) as inputs.
- $\mathsf{adv}(t)$: t can be computed in polynomial time without access to ρ_h.
- ...

Higher-order indistinguishability allows to decompose old axioms:

$$\forall \vec{x}, \vec{y}, \vec{x'}, \vec{y'}. \quad \mathsf{adv}(\mathsf{f}) \Rightarrow (\vec{x}, \vec{y} \sim \vec{x'}, \vec{y'}) \Rightarrow (\vec{x}, \mathsf{f}(\vec{y}) \sim \vec{x'}, \mathsf{f}(\vec{y'}))$$

$$\forall f, x, y, f', x', y'. \quad (x, f, y \sim x', f', y') \Rightarrow (x, f\ y \sim x', f'\ y')$$

$$\forall f, x, y, x', y'. \quad \mathsf{adv}(f) \Rightarrow (x, y \sim x', y') \Rightarrow (x, f, y \sim x', f, y')$$

Recursive definitions

Equip our λ-calculus with recursive definitions to recover the macros of the old meta-logic.

Example

$\mathsf{frame}_{\mathcal{P}}@\tau \stackrel{def}{=} \mathsf{if}\ \tau = \mathsf{init}\ \mathsf{then}\ \mathsf{empty}\ \mathsf{else}\ \langle \mathsf{output}_{\mathcal{P}}@\tau, \mathsf{frame}_{\mathcal{P}}@(\mathsf{pred}\ \tau)\rangle$

$\mathsf{output}_{\mathcal{P}}@\tau \stackrel{def}{=} \mathsf{match}\ \tau\ \mathsf{with}\ \mathsf{T}(i,j) \mapsto \langle \mathsf{n}_t(i,j), \mathsf{h}(\mathsf{n}_t(i,j), \mathsf{k}(i))\rangle \mid \_ \mapsto \mathsf{default}$

Benefits

- Same term can mix macros for different protocols.
- Same logic can deal with different classes of protocols, different attacker models. . .

Advanced axioms

Axioms cannot be derived as liftings of base logic axioms.

- Recover (adaptations of) previous axioms, justifying them from first principles.
- Opportunity to generalize!

Freshness in higher-order logic

$\phi \Rightarrow t \neq (\mathsf{n}\ i)$ if,

for any $\mathcal{M}, \eta, \rho$ such that $[\![\phi]\!]_{\mathcal{M}}(\eta, \rho) = \mathsf{true}$, $[\![t]\!]_{\mathcal{M}}(\eta, \rho)$ does not sample n at $[\![i]\!]_{\mathcal{M}}(\eta, \rho)$.

Advanced axioms

Axioms cannot be derived as liftings of base logic axioms.

- Recover (adaptations of) previous axioms, justifying them from first principles.
- Opportunity to generalize!

Freshness in higher-order logic

$\phi \Rightarrow t \neq (\mathsf{n}\ i)$ if,

for any $\mathcal{M}, \eta, \rho$ such that $[\![\phi]\!]_{\mathcal{M}}(\eta, \rho) = \mathsf{true}$, $[\![t]\!]_{\mathcal{M}}(\eta, \rho)$ does not sample n at $[\![i]\!]_{\mathcal{M}}(\eta, \rho)$.

The same can be done for cryptographic axioms, e.g. IND-CCA.

Example

Assume A inputs x and sends back n only if $x = \mathsf{ok}$.

We are now able to prove $\mathsf{input@A} \neq \mathsf{ok} \Rightarrow \mathsf{n} \neq \mathbf{att}(\mathsf{output@A})$.

Relating global and local quantifiers

Theorem

For any local formula ϕ we have:

$$\mathcal{M}, \sigma \models \forall (x : \tau).[\phi] \qquad \textit{iff} \qquad \mathcal{M}, \sigma \models [\dot{\forall} (x : \tau).\phi]$$

- On the left: for any random variable over $[\![\tau]\!]$, ϕ holds with overwhelming probability.
- On the right: it is overwhelmingly true that ϕ holds for any value in $[\![\tau]\!]$.
- Right $\Rightarrow$ left immediate, other direction requires carefulness with probability theory.

Relating global and local quantifiers

Theorem

For any local formula ϕ we have:

$$\mathcal{M}, \sigma \models \forall(x : \tau).[\phi] \qquad \textit{iff} \qquad \mathcal{M}, \sigma \models [\dot{\forall}\,(x : \tau).\phi]$$

- On the left: for any random variable over $[\![\tau]\!]$, ϕ holds with overwhelming probability.
- On the right: it is overwhelmingly true that ϕ holds for any value in $[\![\tau]\!]$.
- Right $\Rightarrow$ left immediate, other direction requires carefulness with probability theory.

Same result holds for existential quantifier.

Refined version available when types and random variables are constant,
which is the case when we mirror meta-logic reasoning over timestamps and indices.

Conclusion

Theory

- General foundation for Squirrel.
- Conceptually simpler: no more meta-logic, not tied to fixed notion of protocol.
- Complete proof system hiding much of the complexity.

Practice

- Current version of Squirrel implements fragment of new higher-order logic.
- All past case studies carry over.
- New: examples involving corruptions, generic hybrid argument.

Learn more on our website: tech report, tutorials and interactive examples.

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