Journées 2024 GT Scalp & GT Vérif, Lille

# Proof-Theoretical Investigations around Squirrel and CCSA logics

David Baelde and many co-authors

Univ Rennes, CNRS, IRISA

# Logical foundations of 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 introduction to cryptography and CCSA logics.
- A proof-system for synthesizing cryptographic reductions.
- Proof systems for CCSA logics,
  including completeness results for propositional/modal fragments.

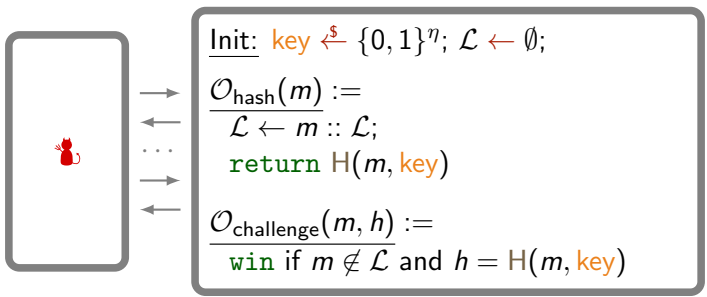
# Background: formal proofs of security protocols

# Unforgeability as a game

A cryptographic hash function $\mathrm{H}(m, \mathrm{key})$ is unforgeable when one cannot produce valid hashes without knowing key.

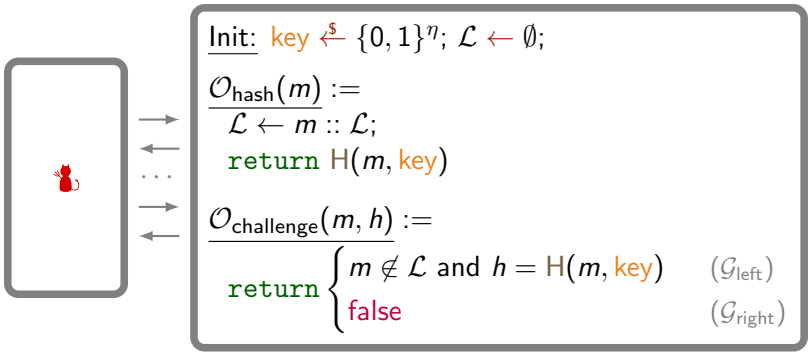
# Unforgeability as a game

A cryptographic hash function $\mathsf{H}(m, \mathsf{key})$ is unforgeable when one cannot produce valid hashes without knowing key.




For any probabilistic polynomial-time Turing machine (PPTM) $\mathcal{G}$, the probability that $\mathcal{G}$ wins is negligible in $\eta$,
i.e., asymptotically smaller than the inverse of any positive polynomial.

# Unforgeability as a pair of games



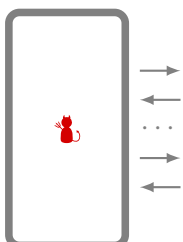

The advantage of any PPTM is negligible in $\eta$:

$$\mathsf{Adv}_{\mathcal{G}}(\text{}) = |\Pr(\text{}^{\mathcal{G}_{\mathsf{left}}} = 0) - \Pr(\text{}^{\mathcal{G}_{\mathsf{right}}} = 0)|$$

We say that $\mathcal{G}_{\mathsf{left}}$ and $\mathcal{G}_{\mathsf{right}}$ are computationally indistinguishable.

# Pseudo-randomness as a pair of games

A cryptographic hash function $\mathsf{H}(m, \mathsf{key})$ is pseudorandom when one cannot distinguish new hashes from random values.

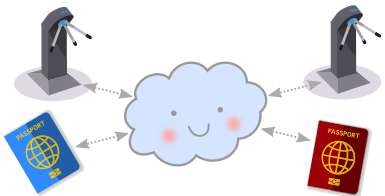


$\underline{\text{Init:}}\ \mathsf{key} \xleftarrow{\$} \{0,1\}^\eta;\ \mathcal{L} \leftarrow \emptyset;$

$\underline{\mathcal{O}_{\mathsf{hash}}(m)} :=$
- $\texttt{assert}\ m \notin \mathcal{L};$
- $\mathcal{L} \leftarrow m :: \mathcal{L};$
- $\texttt{return}\ \mathsf{H}(m, \mathsf{key})$

$\underline{\mathcal{O}_{\mathsf{real\text{-}or\text{-}random}}(m)} :=$
- $\texttt{assert}\ m \notin \mathcal{L};$
- $\mathcal{L} \leftarrow m :: \mathcal{L};$
- $\mathsf{random} \xleftarrow{\$} \{0,1\}^k;$
- $\texttt{return} \begin{cases} \mathsf{H}(m, \mathsf{key}) & (\mathcal{G}_{\mathsf{left}}) \\ \mathsf{random} & (\mathcal{G}_{\mathsf{right}}) \end{cases}$

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

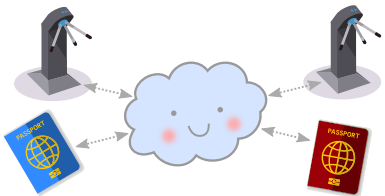
Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathrm{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathrm{ok} \end{array}$$

**Scenario under consideration**:

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

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

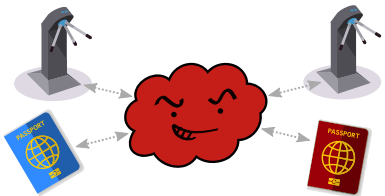
Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

**Scenario under consideration**:

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

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

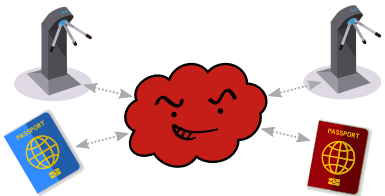
Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

**Scenario under consideration**:

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

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

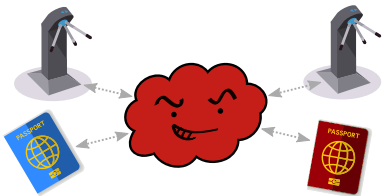
**Scenario under consideration**:

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

**Security properties**:

?? Authentication: $\mathsf{input@}R^j = \mathsf{h}(n_R^j, k_i) \Rightarrow \exists j'.\ \mathsf{input@}R^j = \mathsf{output@}T_i^{j'}$.

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

**Scenario under consideration**:

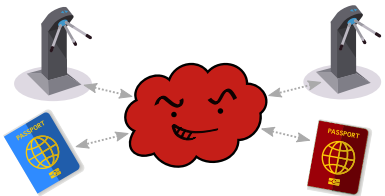
- roles $R^j$, $T_1^j$, ..., $T_n^j$; arbitrary number of sessions $j$ for each role
- attacker can intercept messages, inject new messages

**Security properties**:

✓ Authentication: $\mathsf{input@}R^j = \mathsf{h}(n_R^j, k_i) \Rightarrow \exists j'.\ \mathsf{input@}R^j = \mathsf{output@}T_i^{j'}$.

?? Injective auth.: two sessions of $R$ cannot accept with the same input

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

**Scenario under consideration**:

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

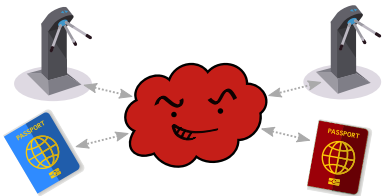
**Security properties**:

✓ Authentication: $\mathsf{input@}R^j = \mathsf{h}(n_R^j, k_i) \Rightarrow \exists j'.\ \mathsf{input@}R^j = \mathsf{output@}T_i^{j'}$.

✓ Injective auth.: two sessions of $R$ cannot accept with the same input

?? Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

# Example protocol (1/3)




Each tag ($T_i$) owns a secret key $k_i$.

Reader ($R$) knows all legitimate keys.

$$\begin{array}{lcll} R & \rightarrow & T_i & : \quad n_R \\ T_i & \rightarrow & R & : \quad \mathsf{h}(n_R, k_i) \\ R & \rightarrow & T_i & : \quad \mathsf{ok} \end{array}$$

**Scenario under consideration**:

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

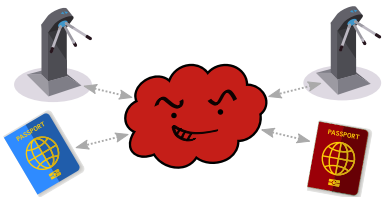
**Security properties**:

✓ Authentication: $\mathsf{input@}R^j = \mathsf{h}(n_R^j, k_i) \Rightarrow \exists j'.\ \mathsf{input@}R^j = \mathsf{output@}T_i^{j'}$.

✓ Injective auth.: two sessions of $R$ cannot accept with the same input

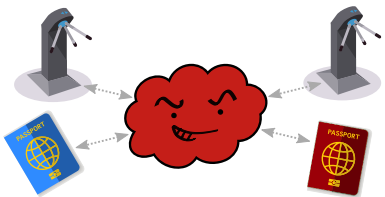
✘ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

# Example protocol (2/3)




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

# Example protocol (2/3)

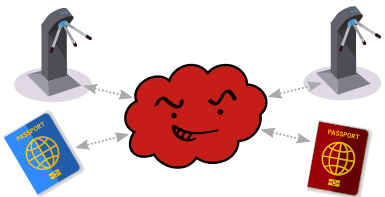



$$T_i \rightarrow R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle$$
$$R \rightarrow T_i : \mathsf{ok}$$

**Security properties**:

✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

# Example protocol (2/3)



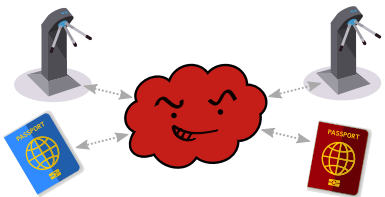

$$T_i \rightarrow R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle$$
$$R \rightarrow T_i : \mathsf{ok}$$

**Security properties**:

✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

?? Authentication: $\mathsf{snd}(\mathsf{input}@R^j) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@R^j), k_i) \Rightarrow \exists j'.\ \mathsf{input}@R^j = \mathsf{output}@T_i^{j'}$.

# Example protocol (2/3)

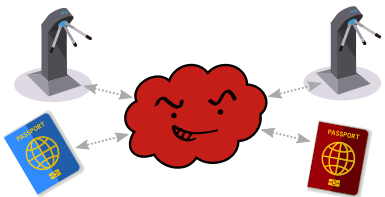



$$T_i \rightarrow R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle$$
$$R \rightarrow T_i : \mathsf{ok}$$

**Security properties**:

- ✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$
- ✓ Authentication: $\mathsf{snd}(\mathsf{input}@R^j) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@R^j), k_i) \Rightarrow \exists j'.\ \mathsf{input}@R^j = \mathsf{output}@T_i^{j'}$.
- ?? Injective auth.: two sessions of $R$ cannot accept with the same input

# Example protocol (2/3)

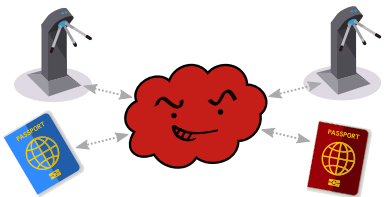



$$T_i \rightarrow R : \langle \mathsf{n}_T, \mathsf{h}(\mathsf{n}_T, k_i) \rangle$$
$$R \rightarrow T_i : \mathsf{ok}$$

**Security properties**:

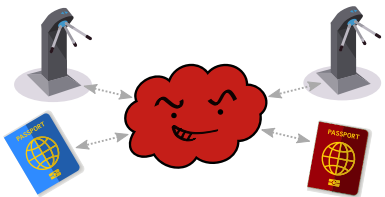
- ✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$
- ✓ Authentication: $\mathsf{snd}(\mathsf{input}@R^j) = \mathsf{h}(\mathsf{fst}(\mathsf{input}@R^j), k_i) \Rightarrow \exists j'.\ \mathsf{input}@R^j = \mathsf{output}@T_i^{j'}$.
- ✘ Injective auth.: two sessions of $R$ cannot accept with the same input

# Example protocol (3/3)




$$
\begin{array}{lcll}
R & \rightarrow & T_i & : \quad n_R \\
T_i & \rightarrow & R & : \quad \langle n_T, \mathsf{h}(\langle n_R, n_T \rangle, k_i) \rangle \\
R & \rightarrow & T_i & : \quad \mathsf{ok}
\end{array}
$$

# Example protocol (3/3)

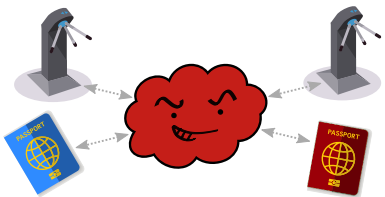



$$
\begin{array}{lcll}
R & \rightarrow & T_i & : \quad n_R \\
T_i & \rightarrow & R & : \quad \langle n_T, \mathsf{h}(\langle n_R, n_T \rangle, k_i) \rangle \\
R & \rightarrow & T_i & : \quad \mathsf{ok}
\end{array}
$$

**Security properties**:

✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

# Example protocol (3/3)



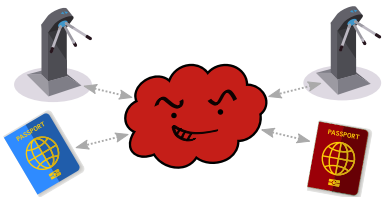

$$
\begin{array}{lcll}
R & \rightarrow & T_i & : \quad n_R \\
T_i & \rightarrow & R & : \quad \langle n_T, \mathsf{h}(\langle n_R, n_T \rangle, k_i) \rangle \\
R & \rightarrow & T_i & : \quad \mathsf{ok}
\end{array}
$$

**Security properties**:

✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$

?? Authentication: $R^j$ accepts $\Rightarrow \exists j'.\ \mathsf{input@}R^j = \mathsf{output@}T_i^{j'}$.

# Example protocol (3/3)




$$
\begin{array}{lcll}
R & \to & T_i & : \quad n_R \\
T_i & \to & R & : \quad \langle n_T, \mathsf{h}(\langle n_R, n_T \rangle, k_i) \rangle \\
R & \to & T_i & : \quad \mathsf{ok}
\end{array}
$$

**Security properties**:

- ✓ Unlinkability: adversary cannot distinguish $T_i, T_i$ from $T_i, T_j$
- ✓ Authentication: $R^j$ accepts $\Rightarrow \exists j'.\ \mathsf{input}@R^j = \mathsf{output}@T_i^{j'}$.
- ✓ Injective auth.: two sessions of $R$ cannot accept with the same input

# Formal proofs of protocols: summary

We have seen the **main ingredients for provable security** of protocols,
i.e. for formal proofs of protocols in the computational model.
As opposed to the symbolic/Dolev-Yao model used e.g. in Proverif and Tamarin.

**Cryptographic assumptions about primitives**

- Generally expressed as indistinguishability between two games.
- Sometimes as negligible chance of winning at single game.

**Security properties of protocols**

- Indistinguishability properties (e.g. unlinkability)
  state that an attacker cannot distinguish between two scenarios.
- Reachability properties (e.g. authentication)
  state that a condition is false with negligible probability (= is overwhelmingly true).

# The CCSA approach

# CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

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

- $[\![t]\!]_{\mathcal{M}}(1^\eta, \rho) \in \{0,1\}^*$
- Name constants n, m, k extract from $\rho$ dedicated sections of length $\eta$.

**Example**

$\mathsf{att}_1(\mathsf{m})$ : attacker computes first message from m.

# CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

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

- $[\![t]\!]_{\mathcal{M}}(1^\eta, \rho) \in \{0,1\}^*$
- Name constants n, m, k extract from $\rho$ dedicated sections of length $\eta$.

**Example**

$\mathsf{att}_2(\mathsf{m}, \mathsf{h}(\mathsf{att}_1(\mathsf{m}), \mathsf{k}))$ : attacker computes $2^{\text{nd}}$ 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 $\rho \in \{0,1\}^\infty$**

- $[\![t]\!]_{\mathcal{M}}(1^\eta, \rho) \in \{0,1\}^*$
- Name constants $\mathsf{n}, \mathsf{m}, \mathsf{k}$ extract from $\rho$ dedicated sections of length $\eta$.

**Example**

$[\![\mathsf{n}]\!]_{\mathcal{M}}(\eta, \rho)$ and $[\![\mathsf{att}_2(\mathsf{m}, \mathsf{h}(\mathsf{att}_1(\mathsf{m}), \mathsf{k}))]\!]_{\mathcal{M}}(\eta, \rho)$ are equal with probability $\leq 2^{-\eta}$.

# CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

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

- $[\![t]\!]_{\mathcal{M}}(1^\eta, \rho) \in \{0,1\}^*$
- Name constants $\mathsf{n}, \mathsf{m}, \mathsf{k}$ extract from $\rho$ dedicated sections of length $\eta$.

## Predicates

- $[\phi]$ where $\phi$ is boolean term : "$\phi$ is true with overwhelming probability"
- $t \sim t'$ for two terms of the same type : "$t$ and $t'$ are computationally indistinguishable"

## Example

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

# CCSA logic: Computationally Complete Symbolic Attacker

[Bana & Comon, CCS'14]

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

- $[\![t]\!]_{\mathcal{M}}(1^\eta, \rho) \in \{0,1\}^*$
- Name constants n, m, k extract from $\rho$ dedicated sections of length $\eta$.

## Predicates

- $[\phi]$ where $\phi$ is boolean term : "$\phi$ is true with overwhelming probability"
- $t \sim t'$ for two terms of the same type : "$t$ and $t'$ are computationally indistinguishable"

## Example

$[\mathsf{n} \neq t]$ is valid for any ground term $t$ where n does not occur.

# Example formulas

**Example (equality and indistinguishability)**

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.

# Example formulas

**Example (equality and indistinguishability)**

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct

# Example formulas

**Example (equality and indistinguishability)**

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

# Example formulas

## Example (equality and indistinguishability)

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

## Example (relating boolean connectives)

Which of the following implications are valid?

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

# Example formulas

## Example (equality and indistinguishability)

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

## Example (relating boolean connectives)

Which of the following implications are valid?

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

# Example formulas

## Example (equality and indistinguishability)

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

## Example (relating boolean connectives)

Which of the following implications are valid?

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

# Example formulas

## Example (equality and indistinguishability)

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

## Example (relating boolean connectives)

Which of the following implications are valid?

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

# Example formulas

## Example (equality and indistinguishability)

- We have $[x = y] \Rightarrow (x \sim y)$ but not the converse.
- Indeed, $\mathsf{m} \sim \mathsf{n}$ but $(\mathsf{m} = \mathsf{n}) \sim \mathsf{false}$. assuming $\mathsf{m}, \mathsf{n}$ distinct
- More generally, $[x = y] \Rightarrow (u[x] \sim v[x]) \Rightarrow (u[y] \sim v[y])$ is valid.

## Example (relating boolean connectives)

Which of the following implications are valid?

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

# Cryptographic axioms

## Example (Unforgeability)

In all models where h is unforgeable, we have

$$[\, u = \mathsf{h}(v, \mathsf{k}) \;\Rightarrow\; \bigvee_{m \in S} m = v \,]$$

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

# Cryptographic axioms

## Example (Unforgeability)

In all models where h is unforgeable, we have

$$[\, u = \mathsf{h}(v, \mathsf{k}) \;\Rightarrow\; \bigvee_{m \in S} m = v \,]$$

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

## Proof sketch.

Fix a model $\mathcal{M}$ where h is unforgeable.
The machines $[\![u]\!]_{\mathcal{M}}$ and $[\![v]\!]_{\mathcal{M}}$ can be simulated by some attacker wrt. the unforgeability game:

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

Submitting $v, u$ to challenge oracle yields a `win` iff our formula is false. □

# Cryptographic axioms

## Example (Unforgeability)

In all models where h is unforgeable, we have

$$[\, u = \mathsf{h}(v, \mathsf{k}) \;\Rightarrow\; \bigvee_{m \in S} m = v \,]$$

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

## Example (Pseudo-randomness)

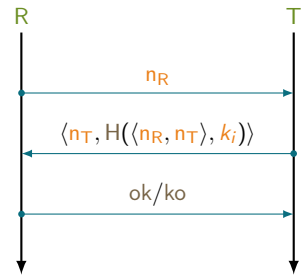
In all models where h is pseudorandom, we have

$$\vec{v},\ \mathsf{h}(t, \mathsf{k}) \;\sim\; \vec{v},\ \text{if } \bigvee_{m \in S} m = t \text{ then } \mathsf{h}(t, \mathsf{k}) \text{ else } \mathsf{n}$$

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

# Verifying protocols along fixed traces: an example

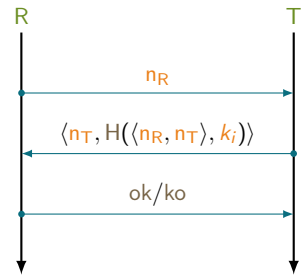
Authentication along trace $R^1_{\mathbf{challenge}} \cdot T^2_7 \cdot T^3_8 \cdot R^1_{\mathbf{test}}$.




$$\left(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}})\rangle, k_i)\right) \Rightarrow \left(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\right)$$

# Verifying protocols along fixed traces: an example

Authentication along trace $R^1_{\mathbf{challenge}} . T^2_7 . T^3_8 . R^1_{\mathbf{test}}$.

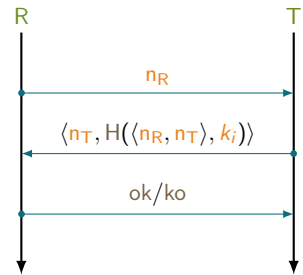



$$\mathsf{in}@R^1_{\mathsf{test}} \;=\; \mathbf{att}_3(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7, \mathsf{out}@T^3_8)$$

$$\Big(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}})\rangle, k_i)\Big) \Rightarrow \big(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\big)$$

# Verifying protocols along fixed traces: an example

Authentication along trace $R^1_{\mathbf{challenge}} . T^2_7 . T^3_8 . R^1_{\mathbf{test}}$.



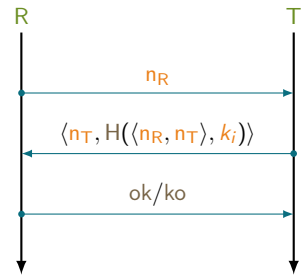

$$\mathsf{out}@T^3_8 \;=\; \langle n^3_T, \mathsf{H}(\langle \mathsf{in}@T^3_8, n^3_T \rangle, k_8) \rangle$$

$$\mathsf{in}@R^1_{\mathsf{test}} \;=\; \mathbf{att}_3(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7, \mathsf{out}@T^3_8)$$

$$\Big(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}}) \rangle, k_i)\Big) \Rightarrow \big(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\big)$$

# Verifying protocols along fixed traces: an example

Authentication along trace $R^1_{\mathbf{challenge}} . T^2_7 . T^3_8 . R^1_{\mathbf{test}}$.

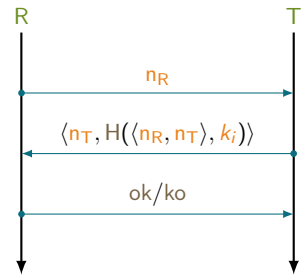



$$\begin{aligned}
\mathsf{in}@T^3_8 &= \mathbf{att}_2(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7) \\
\mathsf{out}@T^3_8 &= \langle n^3_T, \mathsf{H}(\langle \mathsf{in}@T^3_8, n^3_T \rangle, k_8) \rangle \\
\mathsf{in}@R^1_{\mathsf{test}} &= \mathbf{att}_3(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7, \mathsf{out}@T^3_8)
\end{aligned}$$

$$\Big(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}}) \rangle, k_i)\Big) \Rightarrow \big(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\big)$$

# Verifying protocols along fixed traces: an example

Authentication along trace $R^1_{\mathbf{challenge}} . T^2_7 . T^3_8 . R^1_{\mathbf{test}}$.



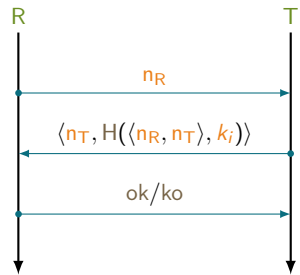

$$
\begin{aligned}
\mathsf{in}@T^2_7 &= \mathbf{att}_1(\mathsf{out}@R^1_{\mathsf{challenge}}) \\
\mathsf{out}@T^2_7 &= \langle n^2_T, \mathsf{H}(\langle \mathsf{in}@T^2_7, n^2_T\rangle, k_7)\rangle \\
\mathsf{in}@T^3_8 &= \mathbf{att}_2(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7) \\
\mathsf{out}@T^3_8 &= \langle n^3_T, \mathsf{H}(\langle \mathsf{in}@T^3_8, n^3_T\rangle, k_8)\rangle \\
\mathsf{in}@R^1_{\mathsf{test}} &= \mathbf{att}_3(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7, \mathsf{out}@T^3_8)
\end{aligned}
$$

$$
\Big(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}})\rangle, k_i)\Big) \Rightarrow \big(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\big)
$$

# Verifying protocols along fixed traces: an example

Authentication along trace $R^1_{\mathbf{challenge}} \cdot T^2_7 \cdot T^3_8 \cdot R^1_{\mathbf{test}}$.

$$
\begin{aligned}
\mathsf{out}@R^1_{\mathsf{challenge}} &= n^1_R \\
\mathsf{in}@T^2_7 &= \mathbf{att}_1(\mathsf{out}@R^1_{\mathsf{challenge}}) \\
\mathsf{out}@T^2_7 &= \langle n^2_T, \mathsf{H}(\langle \mathsf{in}@T^2_7, n^2_T\rangle, k_7)\rangle \\
\mathsf{in}@T^3_8 &= \mathbf{att}_2(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7) \\
\mathsf{out}@T^3_8 &= \langle n^3_T, \mathsf{H}(\langle \mathsf{in}@T^3_8, n^3_T\rangle, k_8)\rangle \\
\mathsf{in}@R^1_{\mathsf{test}} &= \mathbf{att}_3(\mathsf{out}@R^1_{\mathsf{challenge}}, \mathsf{out}@T^2_7, \mathsf{out}@T^3_8)
\end{aligned}
$$




$$
\Big(\bigvee_i \mathsf{snd}(\mathsf{in}@R^1_{\mathsf{test}}) = \mathsf{H}(\langle n^1_R, \mathsf{fst}(\mathsf{in}@R^1_{\mathsf{test}})\rangle, k_i)\Big) \Rightarrow \big(\mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^2_7 \vee \mathsf{in}@R^1_{\mathsf{test}} = \mathsf{out}@T^3_8\big)
$$

# Verifying protocols along all traces: CCSA meta-logic

With Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos and Solène Moreau [SP'21,CSF'22]

# Verifying protocols along all traces: CCSA meta-logic

With Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos and Solène Moreau [SP'21,CSF'22]

## Example (Local meta-logic formulas)

- $\mathsf{output}@\mathsf{A}(i,j) = \mathsf{h}(\mathsf{input}@\mathsf{A}(i,j), \mathsf{k}(i))$ $\leftarrow$ indexed timestamps and names

# Verifying protocols along all traces: CCSA meta-logic

With Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos and Solène Moreau [SP'21,CSF'22]

**Example (Local meta-logic formulas)**

- $\mathsf{output}@\mathsf{A}(i,j) = \mathsf{h}(\mathsf{input}@\mathsf{A}(i,j), \mathsf{k}(i))$ $\leftarrow$ indexed timestamps and names
- $\forall k.\ \mathsf{cond}@\mathsf{B}(k) \Rightarrow \exists i,j.\ \mathsf{A}(i,j) < \mathsf{B}(k) \wedge \mathsf{input}@\mathsf{B}(k) = \mathsf{output}@\mathsf{A}(i,j)$

# Verifying protocols along all traces: CCSA meta-logic

With Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos and Solène Moreau [SP'21,CSF'22]

**Example (Local meta-logic formulas)**

- $\mathsf{output@A}(i,j) = \mathsf{h}(\mathsf{input@A}(i,j), \mathsf{k}(i))$ $\leftarrow$ indexed timestamps and names
- $\forall k.\ \mathsf{cond@B}(k) \Rightarrow \exists i,j.\ \mathsf{A}(i,j) < \mathsf{B}(k) \wedge \mathsf{input@B}(k) = \mathsf{output@A}(i,j)$

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

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

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

# Verifying protocols along all traces: CCSA meta-logic

With Stéphanie Delaune, Charlie Jacomme, Adrien Koutsos and Solène Moreau [SP'21,CSF'22]

**Example (Local meta-logic formulas)**

- $\mathsf{output@A}(i,j) = \mathsf{h}(\mathsf{input@A}(i,j), \mathsf{k}(i))$ $\leftarrow$ indexed timestamps and names
- $\forall k.\ \mathsf{cond@B}(k) \Rightarrow \exists i,j.\ \mathsf{A}(i,j) < \mathsf{B}(k) \wedge \mathsf{input@B}(k) = \mathsf{output@A}(i,j)$

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

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

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

**Eliminating timestamps and indices in $(\phi)^{\mathbb{T}}$**

- Indices and timestamps interpreted in finite domains.
- Quantifications become finite boolean combinations.

# Higher-order CCSA logic (1/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

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

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

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

Terms of the new logic:
$\eta$-indexed families of random variables.

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

# Higher-order CCSA logic (1/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

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

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

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

Terms of the new logic:
$\eta$-indexed families of random variables.

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

## Benefits

- Ability to talk about useful non-PTIME functions, e.g. log.

# Higher-order CCSA logic (1/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

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

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

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

Terms of the new logic:
$\eta$-indexed families of random variables.

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

## Benefits

- Ability to talk about useful non-PTIME functions, e.g. log.
- Quantifiers at all types in local formulas:

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

# Higher-order CCSA logic (1/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

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

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

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

Terms of the new logic:
$\eta$-indexed families of random variables.

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

## Benefits

- Ability to talk about useful non-PTIME functions, e.g. log.
- Quantifiers at all types in local formulas:

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

- Express abstract reasoning schemes (e.g. hybrid argument) using higher-order logic.

# Higher-order CCSA logic (2/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

**Recursive definitions**

Terms are $\lambda$-terms with (well-founded) recursive definitions,
which allows to recover the protocol modelling "macros" $\mathsf{in}@T$, $\mathsf{out}@T \ldots$

# Higher-order CCSA logic (2/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

**Recursive definitions**

Terms are $\lambda$-terms with (well-founded) recursive definitions,
which allows to recover the protocol modelling "macros" $\mathsf{in}@T$, $\mathsf{out}@T\ldots$

**Local/global formulas**

- Local formulas are boolean terms, seen as higher-order formulas.
- Global formulas are first-order formulas over $u \sim v$ and $[\phi]$ where $\phi$ is local.
  We allow order-1 functions in computational indistinguishability.

# Higher-order CCSA logic (2/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

## Recursive definitions

Terms are $\lambda$-terms with (well-founded) recursive definitions,
which allows to recover the protocol modelling "macros" $\mathsf{in}@T$, $\mathsf{out}@T\ldots$

## Local/global formulas

- Local formulas are boolean terms, seen as higher-order formulas.
- Global formulas are first-order formulas over $u \sim v$ and $[\phi]$ where $\phi$ is local.
  We allow order-1 functions in computational indistinguishability.

## Advanced axioms

- In base logic, $[t \neq \mathsf{n}]$ is valid for any closed term $t$ that doesn't contain $\mathsf{n}$.

# Higher-order CCSA logic (2/2)

With Adrien Koutsos and Joseph Lallemand [LICS'23]

## Recursive definitions

Terms are $\lambda$-terms with (well-founded) recursive definitions,
which allows to recover the protocol modelling "macros" $\mathsf{in}@T$, $\mathsf{out}@T\ldots$

## Local/global formulas

- Local formulas are boolean terms, seen as higher-order formulas.
- Global formulas are first-order formulas over $u \sim v$ and $[\phi]$ where $\phi$ is local.
  We allow order-1 functions in computational indistinguishability.

## Advanced axioms

- In higher-order logic, $[\phi \Rightarrow t \neq \mathsf{n}(i)]$ is valid if
  in all models and $\eta, \rho$ such that $\phi$ holds,
  $\mathsf{n}(i)$ cannot occur in $t$ through unfoldings of recursive definitions.
- For cryptographic axioms, need to ensure that some functions are PPTM-computable:
  $\mathsf{adv}(\_)$ predicate.

# Universal quantifications

**Definition (Formulas)**

$\mathcal{M}, \sigma \models \forall(x : \tau).\Phi$ when $\mathcal{M}, \sigma\{x \mapsto A\} \models \Phi$ for any random variable $A$ on $[\![\tau]\!]$

# Universal quantifications

**Definition (Formulas)**

$\mathcal{M}, \sigma \models \forall(x : \tau).\Phi$ when $\mathcal{M}, \sigma\{x \mapsto A\} \models \Phi$ for any random variable $A$ on $[\![\tau]\!]$

**Definition (Boolean terms)**

$[\![\forall(x : \tau).\phi]\!]^{\eta,\rho}_{\mathcal{M},\sigma} = 1$ when $[\![\phi]\!]^{\eta,\rho}_{\mathcal{M},\sigma\{x \mapsto a\}} = 1$ for any constant $a \in [\![\tau]\!]$ (slight abuse)

$\mathcal{M}, \sigma \models [\psi]$ when $(\eta \mapsto \Pr_\rho[\, [\![\psi]\!]^{\eta,\rho}_{\mathcal{M},\sigma} = 1\,])$ overwhelming.

# Universal quantifications

**Definition (Formulas)**

$\mathcal{M}, \sigma \models \forall(x : \tau).\Phi$ when $\mathcal{M}, \sigma\{x \mapsto A\} \models \Phi$ for any random variable $A$ on $[\![\tau]\!]$

**Definition (Boolean terms)**

$[\![\forall(x : \tau).\phi]\!]^{\eta,\rho}_{\mathcal{M},\sigma} = 1$ when $[\![\phi]\!]^{\eta,\rho}_{\mathcal{M},\sigma\{x \mapsto a\}} = 1$ for any constant $a \in [\![\tau]\!]$ (slight abuse)

$\mathcal{M}, \sigma \models [\psi]$ when $(\eta \mapsto \mathrm{Pr}_\rho[\, [\![\psi]\!]^{\eta,\rho}_{\mathcal{M},\sigma} = 1\,])$ overwhelming.

**Theorem**

*For any $\phi$ : bool, and because all probability spaces are discrete, we have:*

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

# Foundations for cryptographic reductions in CCSA logics

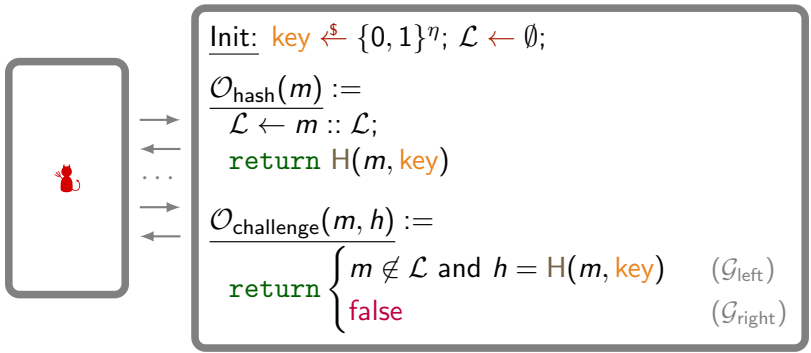
With Adrien Koutsos[1] and Justine Sauvage [CCS'24]

1. Background: formal proofs of security protocols
2. The CCSA approach
3. Foundations for cryptographic reductions in CCSA logics
   - Bi-deduction and its proof system
   - Proof-search implementation in Squirrel
4. Proof systems for CCSA logics
5. Conclusion

[1] Credits to Adrien for the slides.

# Recall: unforgeability as a pair of games

Cryptographic hash function $\mathsf{H}(\_, \mathsf{key})$ is **unforgeable** when the two games below are indistinguishable:




Let $\mathsf{Adv}_{\mathsf{UF}}(\text{😈})$ be the adversary's advantage for this pair of games.

# Hardness assumption: example

**Example**

$$\text{😈}\Big(\mathsf{H}(0,\mathsf{k}),\mathsf{H}(1,\mathsf{k})\Big) = \mathsf{H}(m,\mathsf{k}) \quad \Rightarrow \quad m = 0 \ \vee \ m = 1$$

**Proof by reduction**

Build an adversary 😈 against unforgeability (UF):

- compute $h_0 \leftarrow \mathcal{O}_{\mathsf{hash}}(0)$ and $h_1 \leftarrow \mathcal{O}_{\mathsf{hash}}(1)$;
- call 😈 to compute $h \leftarrow \text{😈}(h_0, h_1)$;
- compute $m$; (must be computable, cf. syntactic condition)
- return $\mathcal{O}_{\mathsf{challenge}}(m, h)$.

$\mathsf{Adv}_{\mathsf{UF}}(\text{😈})$ is the probability that our formula is false: it must be negligible.

# From hardness assumptions to logical rules, by hand

Until recently:

- Squirrel supported a limited set of hardness assumptions (symmetric/asymmetric encryption, signature, hash, DH, ...)
- Built-in tactics for each such assumptions:

hardness assumption (imperative, stateful programs)

$\Downarrow$

reasoning rules (pure, logic)

- Adding rules for new hardness assumptions is: tedious, error-prone, and not in user-space (Ocaml code).

# From hardness assumptions to logical rules, formalized

Systematically translate hardness assumptions into cryptographic rules.

**Inputs:**

- An (imperative, stateful) hardness assumption $\mathcal{G}_0 \approx \mathcal{G}_1$.
- A (pure) indistinguishability property $u_0 \sim u_1$ to prove.

**Goal:**

- Synthesize PPTM $\mathcal{S}$ such that $\mathcal{S}^{\mathcal{G}_i} = [\![ u_i ]\!]$ for each $i \in \{0, 1\}$.

# From hardness assumptions to logical rules, formalized

Systematically translate hardness assumptions into cryptographic rules.

**Inputs:**

- An (imperative, stateful) hardness assumption $\mathcal{G}_0 \approx \mathcal{G}_1$.
- A (pure) indistinguishability property $u_0 \sim u_1$ to prove.

**Goal:**

- Synthesize PPTM $\mathcal{S}$ such that $\mathcal{S}^{\mathcal{G}_i} = [\![ u_i ]\!]$ for each $i \in \{0, 1\}$.
- Ensure $\mathsf{Adv}_{u_0 \sim u_1}(\mathcal{A}) = \mathsf{Adv}_{\mathcal{G}_0 \approx \mathcal{G}_1}(\mathcal{A} \circ \mathcal{S})$ and is thus negligible.

# Bi-deduction

## Bi-Terms

The **bi-terms** $u_{\#} = \#(u_0; u_1)$ represent a pair of left/right scenarios.

Factorize common behavior, e.g. $f(v, \#(u_0; u_1)) = \#(f(v, u_0); f(v, u_1))$

# Bi-deduction

## Bi-Terms

The **bi-terms** $u_\# = \#(u_0; u_1)$ represent a pair of left/right scenarios.

Factorize common behavior, e.g. $f(v, \#(u_0; u_1)) = \#(f(v, u_0); f(v, u_1))$

## Bi-deduction

New judgment $u_\# \triangleright_{\mathcal{G}_0 \approx \mathcal{G}_1} v_\#$ which means:

$$\exists \mathcal{S} \in \mathrm{PPTM}. \begin{cases} \mathcal{S}^{\mathcal{G}_0}(\llbracket u_0 \rrbracket) = \llbracket v_0 \rrbracket \\ \text{and } \mathcal{S}^{\mathcal{G}_1}(\llbracket u_1 \rrbracket) = \llbracket v_1 \rrbracket \end{cases}$$

# Bi-deduction

## Bi-Terms

The **bi-terms** $u_\# = \#(u_0; u_1)$ represent a pair of left/right scenarios.
Factorize common behavior, e.g. $f(v, \#(u_0; u_1)) = \#(f(v, u_0); f(v, u_1))$

## Bi-deduction

New judgment $u_\# \triangleright_{\mathcal{G}_0 \approx \mathcal{G}_1} v_\#$ which means:

$$\exists \mathcal{S} \in \mathrm{PPTM}. \begin{cases} \mathcal{S}^{\mathcal{G}_0}(\llbracket u_0 \rrbracket) = \llbracket v_0 \rrbracket \\ \text{and } \mathcal{S}^{\mathcal{G}_1}(\llbracket u_1 \rrbracket) = \llbracket v_1 \rrbracket \end{cases}$$

## Inference Rule

$$\frac{\emptyset \triangleright_{\mathcal{G}_0 \approx \mathcal{G}_1} \#(u_0; u_1)}{u_0 \sim u_1} \textsc{Bi-Deduce}$$

# Bi-deduction: rules

A few simple **bi-deduction rules**:

- **Transitivity**

$$\frac{\vec{u}_{\#} \rhd \vec{v}_{\#} \qquad \vec{u}_{\#}, \vec{v}_{\#} \rhd \vec{w}_{\#}}{\vec{u}_{\#} \rhd \vec{v}_{\#}, \vec{w}_{\#}}$$

$$\begin{aligned} \underline{\mathcal{S}(\vec{u})} := \ & \vec{v} \leftarrow \mathcal{S}_1(\vec{u}) \\ & \vec{w} \leftarrow \mathcal{S}_2(\vec{u}, \vec{v}) \\ & \textit{return } (\vec{v}, \vec{w}) \end{aligned}$$

# Bi-deduction: rules

A few simple **bi-deduction rules**:

- **Transitivity**

$$\frac{\vec{u}_{\#} \rhd \vec{v}_{\#} \qquad \vec{u}_{\#}, \vec{v}_{\#} \rhd \vec{w}_{\#}}{\vec{u}_{\#} \rhd \vec{v}_{\#}, \vec{w}_{\#}}$$

$$\underline{\mathcal{S}}(\vec{u}) := \begin{array}{l} \vec{v} \leftarrow \mathcal{S}_1(\vec{u}) \\ \vec{w} \leftarrow \mathcal{S}_2(\vec{u}, \vec{v}) \\ \textit{return } (\vec{v}, \vec{w}) \end{array}$$

- **Function application**

$$\frac{\vec{u}_{\#} \rhd \vec{v}_{\#} \qquad \mathsf{adv}(\mathsf{f})}{\vec{u}_{\#} \rhd \mathsf{f}(\vec{v}_{\#})}$$

$$\underline{\mathcal{S}}(\vec{u}) := \begin{array}{l} \vec{v} \leftarrow \mathcal{S}_1(\vec{u}) \\ x \leftarrow \mathbb{M}_{\mathsf{f}}(\vec{v}) \\ \textit{return } x \end{array}$$

# Bi-deduction: rules

**Bi-deduction rules** handling **randomness**:

ORACLE

$$\frac{\vec{u}_{\#} \rhd v_{\#}}{\vec{u}_{\#} \rhd \mathsf{H}(v_{\#}, \mathsf{k})}$$

$$\underline{\mathcal{S}(\vec{u})} := \begin{array}{l} \vec{v} \leftarrow \mathcal{S}_1(\vec{u}) \\ x \xleftarrow{\$} \mathcal{O}_{\mathsf{hash}}(\vec{v}) \\ \textit{return } x \end{array}$$

NAME

$$\frac{\vec{u}_{\#} \rhd v_{\#}}{\vec{u}_{\#} \rhd \mathsf{n}(v_{\#})}$$

$$\underline{\mathcal{S}(\vec{u})} := \begin{array}{l} v \leftarrow \mathcal{S}_1(\vec{u}) \\ x \xleftarrow{\$} \mathbb{M}_{\mathsf{n_r}}(v, \rho_h) \\ \textit{return } x \end{array}$$

# Bi-deduction: rules

**Bi-deduction rules** handling **randomness**:

$$\textsc{Oracle}\quad \frac{\vec{u}_{\#} \rhd v_{\#}}{\vec{u}_{\#} \rhd \mathsf{H}(v_{\#}, \mathsf{k})}$$

$$\underline{\mathcal{S}}(\vec{u}) := \begin{array}{l} \vec{v} \leftarrow \mathcal{S}_1(\vec{u}) \\ x \xleftarrow{\$} \mathcal{O}_{\mathsf{hash}}(\vec{v}) \\ \textit{return } x \end{array}$$

$$\textsc{Name}\quad \frac{\vec{u}_{\#} \rhd v_{\#}}{\vec{u}_{\#} \rhd \mathsf{n}(v_{\#})}$$

$$\underline{\mathcal{S}}(\vec{u}) := \begin{array}{l} v \leftarrow \mathcal{S}_1(\vec{u}) \\ x \xleftarrow{\$} \mathbb{M}_{\mathsf{n_r}}(v, \rho_h) \\ \textit{return } x \end{array}$$

**Problem:** the $\textsc{Name}$ rule allows $\mathcal{S}$ to read $\mathsf{k}$!

# Bi-deduction: constraints

- Problem: $\mathcal{S}$ should not access the game secret keys.
- Solution: track **randomness usage** using logical **constraints**.
  E.g. ensures that $\mathcal{S}$ does not directly use key.
- Enriched bi-deduction judgment:

$$\textsc{Oracle}\ \frac{\vdash \vec{u}_{\#} \rhd v_{\#}}{(\mathsf{k} : \mathsf{T}_{\mathsf{G}}^{\mathsf{key}}) \vdash \vec{u}_{\#} \rhd \mathsf{H}(v_{\#}, \mathsf{k})} \qquad \textsc{Name}\ \frac{}{(\mathsf{n} : \mathsf{T}_{\mathcal{S}}) \vdash \vec{u}_{\#} \rhd \mathsf{n}}$$

# Bi-deduction: constraints

Eventually, check that the **constraints** are **valid**:

$$\frac{\mathcal{C} \vdash \emptyset \rhd \#(u_0; u_1) \qquad \models [\mathsf{Valid}(\mathcal{C})]}{u_0 \sim u_1} \textsc{Bi-Deduce}$$

Example:

$$\not\models [\mathsf{Valid}((\mathsf{k} : \mathsf{T}_{\mathsf{G}}^{\mathsf{key}}), (\mathsf{k} : \mathsf{T}_{\mathcal{S}}))]$$

# Bi-deduction: constraints

Eventually, check that the **constraints** are **valid**:

$$\frac{\mathcal{C} \vdash \emptyset \rhd \#(u_0; u_1) \qquad \models [\mathsf{Valid}(\mathcal{C})]}{u_0 \sim u_1} \textsc{Bi-Deduce}$$

Example:

$$\not\models [\mathsf{Valid}((\mathsf{k} : \mathsf{T}_{\mathsf{G}}^{\mathsf{key}}), (\mathsf{k} : \mathsf{T}_{\mathcal{S}}))]$$

Some additional difficulties:

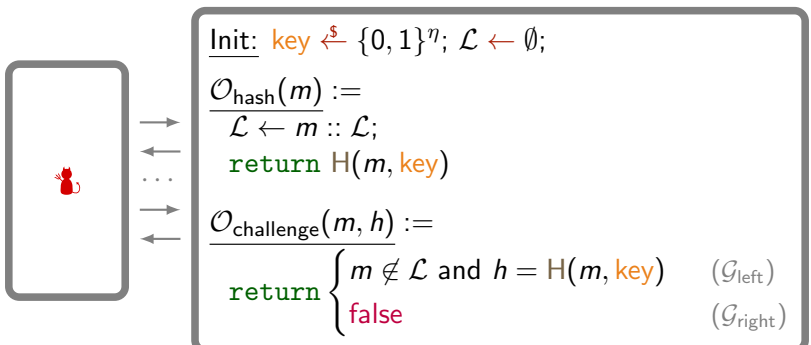
- We need to handle *indexednames* and *conditions*:

$$(\mathsf{n}, i, \phi : \mathsf{T})$$

- Non-well-founded constraints must be avoided, e.g. when condition relies on name itself.

# Bi-deduction: statefulness

We also need to account for $\mathcal{G}$'s **statefulness**.



# Bi-deduction: statefulness

We **track the state** of $\mathcal{G}$:

- Add **Hoare pre- and post- conditions**:

$$(\phi, \psi) \vdash u_{\#} \rhd v_{\#}$$

- **Semantics**:

$$\exists \mathcal{S} \in \mathrm{PPTM}.\ \forall \mu \models \phi. \quad (\!|\mathcal{S}|\!)_{\mu}^{\mathcal{G}_i}(u_i) = (\mu', [\![v_i]\!]) \quad (\forall i \in \{0,1\})$$
$$\text{where } \mu' \models \psi$$

# Bi-deduction: statefulness

We **track the state** of $\mathcal{G}$:

- Add **Hoare pre- and post- conditions**:

$$(\phi, \psi) \vdash u_\# \rhd v_\#$$

- **Semantics**:

$$\exists \mathcal{S} \in \mathrm{PPTM}.\ \forall \mu \models \phi. \quad (\!|\mathcal{S}|\!)_\mu^{\mathcal{G}_i}(u_i) = (\mu', [\![v_i]\!]) \quad (\forall i \in \{0,1\})$$
$$\text{where } \mu' \models \psi$$

- Modified **proof-system**:

$$\frac{(\phi, \chi) \vdash \vec{u}_\# \rhd \vec{v}_\# \qquad (\chi, \psi) \vdash \vec{u}_\#, \vec{v}_\# \rhd \vec{w}_\#}{(\phi, \psi) \vdash \vec{u}_\# \rhd \vec{v}_\#, \vec{w}_\#}\ \textsc{Trans}$$

# Wrapping up on bi-deduction

With Adrien Koutsos and Justine Sauvage [CCS'24]

## In summary

- Framework for reducing logical indistinguishability to cryptographic assumptions.
- Proof-system for bi-deduction.
- Proof-search algorithm for deriving bi-deduction proofs, implemented in Squirrel.
  Includes induction rule, whose invariant is synthesized.

## Further topics

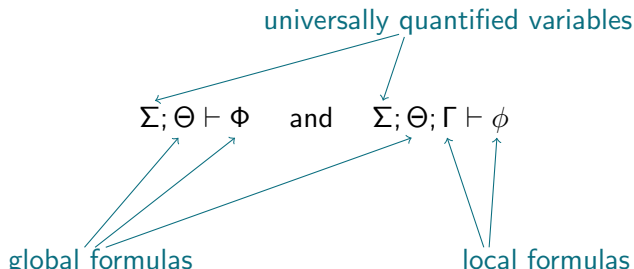
- Bi-deduction (without oracles) is also used in Squirrel to prove implications between indistinguishabilities in the logic. Single-deduction also used to reason about secrecy.
- Can we synthesize crypto reductions in other settings, e.g. Easycrypt?
- Relate to proof-as-program paradigm, e.g. relational typing?

# Proof systems for CCSA logics

With Antoine, Delaune, Koutsos, Lallemand and Moreau [CSF'22,LICS'23,CSL'25]

# Higher-order CCSA sequents

The proof system relies on two kinds of sequents.




Semantics given by $\forall \Sigma.\ (\wedge \Theta) \Rightarrow \Phi$ and $\forall \Sigma.\ (\wedge \Theta) \Rightarrow [(\wedge \Gamma) \Rightarrow \phi]$.

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

# Proof system (1)

Purely global reasoning using classical inferences:

$$\frac{\Sigma; \Theta, \Phi_1; \Gamma \vdash \psi \quad \Sigma; \Theta, \Phi_2; \Gamma \vdash \psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2; \Gamma \vdash \psi} \qquad \frac{\Sigma; \Theta \vdash \neg\neg F}{\Sigma; \Theta \vdash F}$$

$$\frac{\Sigma; \Theta, \Phi_1 \vdash \Psi \quad \Sigma; \Theta, \Phi_2 \vdash \Psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2 \vdash \Psi} \qquad \frac{\Sigma, x; \Theta \vdash F}{\Sigma; \Theta \vdash \forall x.F}\ x \notin \Sigma$$

# Proof system (1)

Purely global reasoning using classical inferences:

$$\frac{\Sigma; \Theta, \Phi_1; \Gamma \vdash \psi \quad \Sigma; \Theta, \Phi_2; \Gamma \vdash \psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2; \Gamma \vdash \psi} \qquad \frac{\Sigma; \Theta \vdash \neg\neg F}{\Sigma; \Theta \vdash F}$$

$$\frac{\Sigma; \Theta, \Phi_1 \vdash \Psi \quad \Sigma; \Theta, \Phi_2 \vdash \Psi}{\Sigma; \Theta, \Phi_1 \vee \Phi_2 \vdash \Psi} \qquad \frac{\Sigma, x; \Theta \vdash F}{\Sigma; \Theta \vdash \forall x.F}\ x \notin \Sigma$$

Purely local reasoning using classical inferences, for instance:

$$\frac{\Sigma; \Theta; \Gamma, \phi_1 \vdash \psi \quad \Sigma; \Theta; \Gamma, \phi_2 \vdash \psi}{\Sigma; \Theta; \Gamma, \phi_1 \vee \phi_2 \vdash \psi} \qquad \frac{\Sigma, x; \Theta; \Gamma \vdash \phi}{\Sigma; \Theta; \Gamma \vdash \forall x.\phi}\ x \notin \Sigma$$

# Proof system (2)

From global to local hypotheses:

$$\frac{\Sigma; \Theta; \phi, \Gamma \vdash \psi}{\Sigma; \Theta, [\phi]; \Gamma \vdash \psi}$$

Some (ugly) inferences require that $\phi$ is deterministic:

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

# Proof system (2)

From global to local hypotheses:

$$\frac{\Sigma; \Theta; \phi, \Gamma \vdash \psi}{\Sigma; \Theta, [\phi]; \Gamma \vdash \psi}$$

Some (ugly) inferences require that $\phi$ is deterministic:

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

These rules can be derived when $\Sigma; \Theta \vdash [\phi] \vee [\neg\phi]$ is valid ($= \phi$ is determined).

# Proof system (3)

From reachability to equivalence properties:

$$\frac{\Sigma; \Theta; \vdash u = v \quad \Sigma; \Theta \vdash [\vec{C}[v]] \sim \vec{t}}{\Sigma; \Theta \vdash [\vec{C}[u]] \sim \vec{t}}$$

From equivalence to reachability properties:

$$\frac{\Sigma; \Theta \vdash [\vec{u} \sim \vec{v}] \quad \Sigma; \Theta; \Gamma[\vec{v}] \vdash \phi[\vec{v}]}{\Sigma; \Theta; \Gamma[\vec{u}] \vdash \phi[\vec{u}]} \; \Gamma, \phi \text{ without names}$$

# Towards completeness

With Thibaut Antoine [CSL'25]

**Problem**

The gradual, organic growth of our proof system is unsatisfying.

Look for well-behaved and complete proof system, for a fragment of the logic.

Non-trivial even if forget about protocols, messages, quantifiers...
keeping only a propositional local logic, and $[\phi]$ atoms in the global logic.

- Why don't we need a mixed *reductio ad absurdum* rule?

$$\frac{\Sigma; \Theta \vdash \neg\neg\Phi}{\Sigma; \Theta \vdash \Phi} \qquad \frac{\Sigma; \Theta; \Gamma \vdash \neg\neg\phi}{\Sigma; \Theta; \Gamma \vdash \phi} \qquad \frac{\Sigma; \Theta \vdash \neg\neg[\Gamma \Rightarrow \phi]}{\Sigma; \Theta; \Gamma \vdash \phi} \; ??$$

- How to incorporate determinism assumptions in the proof system?
  Adding $[P] \vee [\neg P]$ as hypotheses yields exponential blowup.

# Modal logic of overwhelming truth

## Syntax

$\varphi ::= \bot \mid p \mid \varphi \Rightarrow \varphi' \mid \Box\varphi$ where $p \in \mathcal{P}$

$\wedge$, $\vee$, $\Diamond$... as syntactic sugar

## Models

Cryptographic structures $\mathcal{S}$ given by

- for each $\eta \in \mathbb{N}$, a set of tapes $X_\eta$;
- for each $p \in \mathcal{P}$ and $\eta \in \mathbb{N}$, a random variable $\hat{p}_\eta : X_\eta \to \{0,1\}$.

several inessential variants in the paper

## Satisfaction

$$
\begin{array}{lll}
\mathcal{S},\eta,\rho \models p & \text{iff} & \hat{p}_\eta(\rho) = 1 \\
\mathcal{S},\eta,\rho \models \varphi \Rightarrow \psi & \text{iff} & \mathcal{S},\eta,\rho \models \varphi \text{ implies } \mathcal{S},\eta,\rho \models \psi \\
\mathcal{S},\_,\_ \models \Box\varphi & \text{iff} & (\eta \mapsto Pr_\rho(\mathcal{S},\eta,\rho \models \varphi)) \text{ is overwhelming}
\end{array}
$$

# Modal logic of overwhelming truth

## Syntax

$\varphi ::= \bot \mid p \mid \varphi \Rightarrow \varphi' \mid \Box\varphi$ where $p \in \mathcal{P}$

$\wedge$, $\vee$, $\Diamond$... as syntactic sugar

## Models

Cryptographic structures $\mathcal{S}$ given by

- for each $\eta \in \mathbb{N}$, a set of tapes $X_\eta$;
- for each $p \in \mathcal{P}$ and $\eta \in \mathbb{N}$, a random variable $\hat{p}_\eta : X_\eta \to \{0,1\}$.

several inessential variants in the paper

## Satisfaction

$$
\begin{array}{lll}
\mathcal{S},\eta,\rho \models p & \text{iff} & \hat{p}_\eta(\rho) = 1 \\
\mathcal{S},\eta,\rho \models \varphi \Rightarrow \psi & \text{iff} & \mathcal{S},\eta,\rho \models \varphi \text{ implies } \mathcal{S},\eta,\rho \models \psi \\
\mathcal{S},\_,\_ \models \Box\varphi & \text{iff} & (\eta \mapsto Pr_\rho(\mathcal{S},\eta,\rho \models \varphi)) \text{ is overwhelming}
\end{array}
$$

Formula $\varphi$ is valid if $\mathcal{S},\_,\_ \models \Box\varphi$ for all $\mathcal{S}$.

# Characterizing our modal logic

The following formulas are valid:

- $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
- $\Box\varphi \Rightarrow \varphi$
- $\Box\varphi \Leftrightarrow \Box\Box\varphi$

# Characterizing our modal logic

The following formulas are valid:

- $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
- $\Box\varphi \Rightarrow \varphi$
- $\Box\varphi \Leftrightarrow \Box\Box\varphi$ and $\Diamond\varphi \Rightarrow \Box\Diamond\varphi$

# Characterizing our modal logic

The following formulas are valid:

- $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
- $\Box\varphi \Rightarrow \varphi$
- $\Box\varphi \Leftrightarrow \Box\Box\varphi$ and $\Diamond\varphi \Rightarrow \Box\Diamond\varphi$

**Theorem**

*A formula is valid in our sense iff it is a theorem of S5.*

**Proof.**

- Axioms defining S5 are sound in modal logic of ow. truth.
- S5 valid = valid in Kripke frames that are finite cliques.
  Finite cliques can be turned into cryptographic structures. □

# Hypersequents

S5 enjoys a very nice hypersequent calculus [Poggiolesi 2008].

$$\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_n \vdash \Delta_n \qquad \text{reads as} \qquad \bigvee_i \Box(\wedge\Gamma_i \Rightarrow \vee\Delta_i)$$

# Hypersequents

S5 enjoys a very nice hypersequent calculus [Poggiolesi 2008].

$$\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_n \vdash \Delta_n \qquad \text{reads as} \qquad \bigvee_i \Box(\wedge\Gamma_i \Rightarrow \vee\Delta_i)$$

Selected rules:

$$\frac{}{\ldots \mid \Gamma, \varphi \vdash \varphi, \Delta} \qquad \frac{\ldots \mid \Gamma, \psi \vdash \Delta \quad \ldots \mid \Gamma \vdash \varphi, \Delta}{\ldots \mid \Gamma, \varphi \Rightarrow \psi \vdash \Delta} \qquad \frac{\ldots \mid \Gamma \vdash \Delta \mid \vdash \varphi}{\ldots \mid \Gamma \vdash \Delta, \Box\varphi}$$

# Hypersequents

S5 enjoys a very nice hypersequent calculus [Poggiolesi 2008].

$$\Gamma_1 \vdash \Delta_1 \mid \ldots \mid \Gamma_n \vdash \Delta_n \qquad \text{reads as} \qquad \bigvee_i \Box(\wedge\Gamma_i \Rightarrow \vee\Delta_i)$$

Selected rules:

$$\frac{}{\ldots \mid \Gamma, \varphi \vdash \varphi, \Delta} \qquad \frac{\ldots \mid \Gamma, \psi \vdash \Delta \quad \ldots \mid \Gamma \vdash \varphi, \Delta}{\ldots \mid \Gamma, \varphi \Rightarrow \psi \vdash \Delta} \qquad \frac{\ldots \mid \Gamma \vdash \Delta \mid \vdash \varphi}{\ldots \mid \Gamma \vdash \Delta, \Box\varphi}$$

Modified axiom yields variant that is complete for S5 + $(\Box p \vee \Box\neg p)$:

$$\frac{}{\ldots \mid \Gamma, p \vdash \Delta \mid \Gamma' \vdash p, \Delta'}$$

# Back to earth

Little hope to do better for S5... but we don't need all of S5!

## Propositional CCSA logic

$$\begin{array}{rrcl} \text{Global formulas} & \Phi & ::= & \bot \mid [\varphi] \mid \Phi \Rightarrow \Phi' \\ \text{Local formulas} & \varphi & ::= & \bot \mid p \mid \varphi \Rightarrow \varphi' \end{array}$$

Two kinds of sequents, similar to previous ones but multiple-conclusion:

$$\begin{array}{ll} \Theta_1, \ldots, \Theta_n \vdash \Psi_1, \ldots, \Psi_m & \text{reads as } \wedge\Theta \Rightarrow \vee\Psi \\ \Theta, \ldots, \Theta_n; \gamma_1, \ldots, \gamma \vdash \delta_1 \ldots, \delta_m & \text{reads as } \wedge\Theta \Rightarrow \Box(\wedge\gamma \vdash \vee\delta) \end{array}$$

# A sound and complete calculus for propositional CCSA logic

$$\frac{}{\Theta, F \vdash F, \Pi} \qquad \frac{}{\Theta, \bot \vdash \Pi} \qquad \frac{}{\Theta; \Gamma, \phi \vdash \phi, \Delta} \qquad \frac{}{\Theta; \Gamma, \bot \vdash \Delta}$$

$$\frac{\Theta \vdash F, \Pi \quad \Theta, G \vdash \Pi}{\Theta, F \Rightarrow G \vdash \Pi} \qquad \frac{\Theta, F \vdash G, \Pi}{\Theta \vdash F \Rightarrow G, \Pi}$$

$$\frac{\Theta; \Gamma \vdash \phi, \Delta \quad \Theta; \Gamma, \psi \vdash \Delta}{\Theta; \Gamma, \phi \Rightarrow \psi \vdash \Delta} \qquad \frac{\Theta; \Gamma, \phi \vdash \psi, \Delta}{\Theta; \Gamma \vdash \phi \Rightarrow \psi, \Delta}$$

$$\frac{\Theta; \Gamma, \phi \vdash \Delta}{\Theta, \Box\phi; \Gamma \vdash \Delta} \qquad \frac{\Theta; \cdot \vdash \phi}{\Theta \vdash \Box\phi, \Pi}$$

# A sound and complete calculus for propositional CCSA logic

$$\frac{}{\Theta, F \vdash F, \Pi} \qquad \frac{}{\Theta, \bot \vdash \Pi} \qquad \frac{}{\Theta; \Gamma, \phi \vdash \phi, \Delta} \qquad \frac{}{\Theta; \Gamma, \bot \vdash \Delta}$$

$$\frac{\Theta \vdash F, \Pi \quad \Theta, G \vdash \Pi}{\Theta, F \Rightarrow G \vdash \Pi} \qquad \frac{\Theta, F \vdash G, \Pi}{\Theta \vdash F \Rightarrow G, \Pi}$$

$$\frac{\Theta; \Gamma \vdash \phi, \Delta \quad \Theta; \Gamma, \psi \vdash \Delta}{\Theta; \Gamma, \phi \Rightarrow \psi \vdash \Delta} \qquad \frac{\Theta; \Gamma, \phi \vdash \psi, \Delta}{\Theta; \Gamma \vdash \phi \Rightarrow \psi, \Delta}$$

$$\frac{\Theta; \Gamma, \phi \vdash \Delta}{\Theta, \Box\phi; \Gamma \vdash \Delta} \qquad \frac{\Theta; \cdot \vdash \phi}{\Theta \vdash \Box\phi, \Pi}$$

**Lemma**

*If* $[\varphi_1], \ldots, [\varphi_n] \vdash [\psi_1], \ldots, [\psi_m]$ *is valid, there exists* $k \in [1; m]$ *such that* $\varphi_1, \ldots, \varphi_n \vdash \psi_k$ *is classically valid.*

# Conclusion

## Summary

- CCSA methodology for proving cryptographic protocols.
- Over time, CCSA logics (resp. axioms) have become both more expressive (resp. precise) and conceptually clearer.
- Opportunities for proof theory and semantics (e.g. beyond discrete proba. in HO CCSA).

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

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

# Conclusion

## Summary

- CCSA methodology for proving cryptographic protocols.
- Over time, CCSA logics (resp. axioms) have become both more expressive (resp. precise) and conceptually clearer.
- Opportunities for proof theory and semantics (e.g. beyond discrete proba. in HO CCSA).

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

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

## Other topics

- Proof automation, including using SMT solvers for the polymorphic, higher-order logic.
- Concrete bounds rather than asymptotic guarantees (cf. work of Théo Vignon [CSF'23]).
- Working on more complex protocols and post-quantum security.