MPRI 2.30: Proofs of Security Protocols

1. The CCSA Approach to Computational Security

Adrien Koutsos

2022/2023

Introduction

Introduction

The Computationally Complete Symbolic Attacker (**CCSA**) [2] is a symbolic approach in the computational model to verify security protocols.

Its **key ingredients** are:

- Interpret a **protocol execution** as the **sequence of terms** seen by the adversary (the frame).
- Interpret terms as PTIME-computable bitstring distributions.
  - Functions symbol (e.g. the pair $< \_, \_ >$) are functions over bitstrings.
  - Names (e.g. n) are (uniform) distributions over bitstrings.
- Use cryptographic hardness assumptions (e.g. IND-CCA).
- Symbolic approach: no probabilities, no security parameter.

Protocols as Sequences of Terms

Example of a Protocol

To illustrate what terms we need to consider, we consider a simple authentication protocol:

The Private Authentication (PA) Protocol, v1

$$1 : A \to B : \nu\, n_A. \quad \textbf{out}(c_A, \{\langle pk_A\,,\, n_A\rangle\}_{pk_B})$$
$$2 : B \to A : \nu\, n_B.\,\textbf{in}(c_A, x).\ \textbf{out}(c_B, \{\langle \pi_2(\mathrm{dec}(x, sk_A))\,,\, n_B\rangle\}_{pk_A})$$

where $pk_A \equiv pk(k_A)$ and $pk_B \equiv pk(k_B)$.

*Notation: we use $\equiv$ to denote **syntactic** equality of terms.*

Terms

We use **terms** to model *protocol messages*, built upon:

- **Names** $\mathcal{N}$, e.g. $\mathsf{n}_A, \mathsf{n}_B$, for random samplings.
- **Function symbols** $\mathcal{F}$, e.g.:

$$\mathsf{A},\ \mathsf{B},\ \langle \_\ ,\ \_ \rangle,\ \pi_1(\_),\ \pi_2(\_),\ \{\_\}_{\_},\ \mathsf{pk}(\_),\ \mathsf{sk}(\_),$$

$$\mathsf{if\_then\_else\_},\ \_ \dot{=} \_,\ \_ \dot{\wedge} \_,\ \_ \dot{\vee} \_,\ \_ \dot{\rightarrow} \_$$

Examples

$\mathsf{pk}(\mathsf{k_A})$ $\{\langle \mathsf{pk_A}\ ,\ \mathsf{n_A} \rangle\}_{\mathsf{pk_B}}$ $\pi_1(\mathsf{n}_A)$

Types. Also, each function symbol $f \in \mathcal{F}$ comes with a type:

$$\mathsf{type}(f) = (\tau_1 \star \cdots \star \tau_n) \rightarrow \tau$$

For now, we use the `message` and `bool` types. We require that terms are well-typed.

Protocol Constructs

But this is not enough to **translate** a protocol **execution** into a **sequence of terms**. We also need to:

- model **inputs** of the protocol as **terms**.
- account for protocol **branching** (i.e. if ϕ then P_1 else P_2).

Moreover, we **forbid unbounded replication** !, since we want to build **finite** sequences of terms.

We will discuss how to retrieve replication briefly later.

Protocols as Sequences of Terms

Protocol Inputs

Inputs

The PA Protocol, v1

$$1 : \mathsf{A} \to \mathsf{B} : \nu\, \mathsf{n_A}. \qquad \mathbf{out}(\mathtt{c_A}, \{\langle \mathsf{pk_A}\,,\, \mathsf{n_A}\rangle\}_{\mathsf{pk_B}})$$

$$2 : \mathsf{B} \to \mathsf{A} : \nu\, \mathsf{n_B}. \mathbf{in}(\mathtt{c_A}, x). \mathbf{out}(\mathtt{c_B}, \{\langle \pi_2(\mathsf{dec}(\boxed{x}, \mathsf{sk_A}))\,,\, \mathsf{n_B}\rangle\}_{\mathsf{pk_A}})$$

How do we represent the adversary's inputs?

- We use **adversarial** functions symbols **att** $\in \mathcal{G}$, which takes as input the current knowledge of the adversary.
- Intuitively, **att** can be any probabilistic PTIME computation.

Example: Terms for PA, v1

$$t_1 \equiv \{\langle \mathsf{pk_A}\,,\, \mathsf{n_A}\rangle\}_{\mathsf{pk_B}}$$

$$t_2 \equiv \{\langle \pi_2(\mathsf{dec}(\boxed{\mathbf{att}(t_1)}, \mathsf{sk_A}))\,,\, \mathsf{n_B}\rangle\}_{\mathsf{pk_A}}$$

Inputs

More generally, if:

- there has already been n **outputs**, represented by the terms $t_1, \ldots, t_n$;
- and we are doing the j-th **input** since the protocol started;

then the input bitstring is represented by:

$$\mathbf{att}_j(t_1, \ldots, t_n)$$

where $\mathbf{att}_j \in \mathcal{G}$ is an **adversarial** function symbol of arity n.

j allows to have different values for consecutive inputs.

Terms

We extend our set of **terms** accordingly:

- **Names** $\mathcal{N}$.
- **Variables** $\mathcal{X}$.
- **Function symbols** $\mathcal{F}$.
- **Adversarial function symbols** $\mathcal{G}$, of any arity.

We note this set of terms $\mathcal{T}(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})$.

We will see the use of variables in $\mathcal{X}$ later.

Protocols as Sequences of Terms

Protocol Branching

Protocol Branching

In our first version of PA, B does not check that its comes from A. We propose a second version fixing this:

The PA Protocol, v2

$$\begin{aligned}
1 : \mathsf{A} \to \mathsf{B} : \nu\, \mathsf{n_A}. \qquad & \mathbf{out}(\mathtt{c_A}, \{\langle \mathsf{pk_A}\, ,\, \mathsf{n_A}\rangle\}_{\mathsf{pk_B}}) \\
2 : \mathsf{B} \to \mathsf{A} : \nu\, \mathsf{n_B}. \mathbf{in}(\mathtt{c_A}, x).\ & \text{if } \pi_1(d) \doteq \mathsf{pk_A} \\
& \quad \text{then } \mathbf{out}(\mathtt{c_B}, \{\langle \pi_2(d)\, ,\, \mathsf{n_B}\rangle\}_{\mathsf{pk_A}}) \\
& \quad \text{else } \ \mathbf{out}(\mathtt{c_B}, \{0\}_{\mathsf{pk_A}})
\end{aligned}$$

where $d \equiv \mathrm{dec}(\mathrm{x}, \mathsf{sk_A})$.

In the else branch, we return an encryption, to hide to the adversary which branch was taken.

Protocol Branching

The PA Protocol, v2

$$
\begin{aligned}
&1 : \mathsf{A} \to \mathsf{B} : \nu\, \mathsf{n_A}. && \mathbf{out}(\mathtt{c_A}, \{\langle \mathsf{pk_A}\, ,\, \mathsf{n_A} \rangle\}_{\mathsf{pk_B}}) \\
&2 : \mathsf{B} \to \mathsf{A} : \nu\, \mathsf{n_B}.\, \mathbf{in}(\mathtt{c_A}, x).\ && \text{if } \pi_1(d) \doteq \mathsf{pk_A} \\
& && \quad \text{then } \mathbf{out}(\mathtt{c_B}, \{\langle \pi_2(d)\, ,\, \mathsf{n_B} \rangle\}_{\mathsf{pk_A}}) \\
& && \quad \text{else } \ \mathbf{out}(\mathtt{c_B}, \{0\}_{\mathsf{pk_A}})
\end{aligned}
$$

The **bitstring outputted** in the second message of the protocol **depends** on which **branch** was taken.

Moreover, the adversary may **not know which branch** was taken.

$\Rightarrow$ **branching** is **pushed** (or **folded**) in the outputted terms, using the if_then_else_ function symbol.

Protocol Branching

Example: Terms for PA, v2

$$
\begin{aligned}
t_1 &\equiv \{\langle \mathsf{pk_A} , \mathsf{n_A} \rangle\}_{\mathsf{pk_B}} \\
t_2 &\equiv \text{if } \pi_1(d_1) \doteq \mathsf{pk_A} \\
&\qquad \text{then } \{\langle \pi_2(d_1) , \mathsf{n_B} \rangle\}_{\mathsf{pk_A}} \\
&\qquad \text{else } \{0\}_{\mathsf{pk_A}}
\end{aligned}
$$

where $d_1 \equiv \mathsf{dec}(\mathbf{att}(t_1), \mathsf{sk_A})$.

Folding

Folding

We describe a **systematic method** to compute, given a **process** P and a **trace** `tr` of **observable actions**, the **terms** representing the **outputted messages** during the execution of P over `tr`.

This is the **folding** of P over `tr`.

We deal with **inputs** and protocol **branching** using the two techniques we just saw.

Non-Determinism and Computational Semantics

First, we require that **processes** are **deterministic**.

Indeed, consider a simple process:

$$P = \mathbf{out}(\mathsf{c}, t_0) \mid \mathbf{out}(\mathsf{c}, t_1)$$

- in a symbolic setting, this is a **non-deterministic** choice between t_0 and t_1.
- in a computational setting, the semantics of P is unclear: how do **non-determinism** and **probabilities** interacts?

Hence, we choose to **forbid** such process: we only consider **action-deterministic** processes.

Action-Deterministic Processes

A process P is **action-deterministic** if the *observable* executions, starting from P, is described by a deterministic transition system.

Action-deterministic Process

A configuration A is action-deterministic iff for any $A \rightarrow^* A'$, for any observable action α, if $A' \xrightarrow{\alpha} A_1$ and $A' \xrightarrow{\alpha} A_2$ then $A_1 = A_1$, for any term interpretation domain.

P is action-deterministic if the initial configuration $(P, \emptyset, \emptyset)$ is.

Exercise

Determine if the following protocols are **action-deterministic**.

$$\mathbf{out}(\mathsf{c}, t_1) \mid \mathbf{in}(\mathsf{c}, \mathsf{x}).\, \mathbf{out}(\mathsf{c}, t_2)$$

$$\text{if } b \text{ then } \mathbf{out}(\mathsf{c}, t_1) \text{ else } \mathbf{in}(\mathsf{c}, \mathsf{x}).\, \mathbf{out}(\mathsf{c}, t_2)$$

$$\mathbf{out}(\mathsf{c}, t_1) \mid \text{if } b \text{ then } \mathbf{out}(\mathsf{c}, t_2) \text{ else } \mathbf{out}(\mathsf{c}_0, t_3)$$

Folding

Folding Algorithm

Folding Configuration

Folding configuration

A **folding configuration** is a tuple $(\Phi; \sigma; j; \Pi_1, \ldots, \Pi_l)$ where:

- Φ is a sequence of terms (in $\mathcal{T}(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})$).
- σ is a finite sequence of mappings $(\mathsf{x} \mapsto t)$ where t is a term.
- $j \in \mathbb{N}$.
- for every i, $\Pi_i = (P_i, b_i)$ where P_i is a protocol and b_i is a boolean term.

Folding Configuration: Intuition

In a **folding configuration** $(\Phi; \sigma; j; \Pi_1, \ldots, \Pi_l)$:

- Φ is the **frame**, i.e. the sequence of terms outputted since the execution started.
- σ **records inputs**, it maps input variable to their corresponding term.
- j **counts the number of inputs** since the execution started.
- (P, b) **represent the protocol** P if b is true (and is **null** otherwise).

  Using this interpretation, $\Pi_1, \ldots, \Pi_l$ is the **current process**.

Initial configuration: $(\epsilon; \emptyset; 0; (P, \top))$

Folding: New and Branching Rules

Rule for protocol branching:

$$(\Phi; \sigma; j; (\text{if } b \text{ then } P_1 \text{ else } P_2, b'), \Pi_1, \dots, \Pi_l)$$
$$\hookrightarrow (\Phi; \sigma; j; (P_1, b' \wedge b), (P_2, b' \wedge \neg b), \Pi_1, \dots, \Pi_l)$$

Rule for new:

$$(\Phi; \sigma; j; (\nu\, \mathsf{n}, P, b), \Pi_1, \dots, \Pi_l)$$
$$\hookrightarrow (\Phi; \sigma; j; (P[\mathsf{n} \mapsto \mathsf{n}_f], b), \Pi_1, \dots, \Pi_l)$$

if n_f does not appear in the lhs configuration

$\hookrightarrow$-irreducibility

A folding configuration K is $\hookrightarrow$-irreducible if for any K', we have $K \not\hookrightarrow K'$.

Folding: Input Rule

Rule for inputs:

$$(\Phi; \sigma; j; (\mathbf{in}(\mathsf{c}, x).P_1, b_1), \ldots, (\mathbf{in}(\mathsf{c}, x).P_n, b_n), \Pi_1, \ldots, \Pi_l)$$
$$\overset{\mathbf{in}(\mathsf{c})}{\hookrightarrow} \ (\Phi; \sigma[\mathsf{x} \mapsto \mathbf{att}_j(\Phi)]; j+1; (P_1, b_1), \ldots, (P_n, b_n), \Pi_1, \ldots, \Pi_l)$$

if $\mathsf{x} \notin \mathrm{dom}(\sigma)$, the lhs folding configuration is $\hookrightarrow$-irreducible and if for every i, Π_1 does not start by an input on c.

Alternative

If the **computational semantics** of processes tell the adversary if an **input succeeded or not**, we replace Φ (in the rhs) by:

$$\Phi, \dot{\bigvee}_{1 \leq i \leq n} b_i$$

Folding: Output Rule

Rule for outputs:

$$(\Phi; \sigma; j; (\mathbf{out}(\mathsf{c}, t_1).P_1, b_1), \ldots, (\mathbf{out}(\mathsf{c}, t_n).P_n, b_n), \Pi_1, \ldots, \Pi_l)$$
$$\xhookrightarrow{\mathbf{out}(\mathsf{c})} (\Phi, t\sigma; \sigma; j; (P_1, b_1), \ldots, (P_n, b_n), \Pi_1, \ldots, \Pi_l)$$

if the lhs folding configuration is $\hookrightarrow$-irreducible and if for every i, Π_1 does not start by an output on c and:

$$t \equiv \text{if } b_1 \text{ then } t_1 \text{ else } \ldots \text{if } b_n \text{ then } t_n \text{ else } \mathtt{error}$$

The input and output rules makes sense because we restrict ourselves to action-deterministic processes.

Remark: we omit the error message when $(\dot{\bigvee}_{1 \le i \le n} b_i) \Leftrightarrow \text{true}$.

Folding

A **folding observable action** a is either **in**(c) or **out**(c).

Given an **action-deterministic** process P and a trace $\mathtt{tr}$ of **folding observable**, if:

$$(\epsilon; \emptyset; 0; (P, \top)) \xhookrightarrow{\mathtt{tr}} (\Phi; \_; \_; \_)$$

then Φ is the **folding** of P over $\mathtt{tr}$, denoted $\mathrm{fold}(P, \mathtt{tr})$.

Folding: Exercises

Exercise

What are all the **possible foldings** of the following protocols?

$\mathbf{in}(\mathsf{c}, \mathsf{x}).\, \mathbf{out}(\mathsf{c}, t)$ $\qquad$ $\mathbf{out}(\mathsf{c}, t_1) \mid \mathbf{in}(\mathsf{c}_0, \mathsf{x}).\, \mathbf{out}(\mathsf{c}_0, t_2)$

$$\text{if } b \text{ then } \mathbf{out}(\mathsf{c}, t_1) \text{ else } \mathbf{out}(\mathsf{c}, t_2)$$

$$\text{if } b \text{ then } \mathbf{out}(\mathsf{c}_1, t_1) \text{ else } \mathbf{out}(\mathsf{c}_2, t_2)$$

Exercise

Extend the **folding** algorithm with a rule allowing to handle processes with let bindings.

Semantics of Terms

Semantics of Terms

We showed how to represent **protocol execution**, on some fixed trace of observables `tr`, as a **sequence of terms**.

Intuitively, the terms corresponds to **PTIME-computable bitstring distributions**.

Example

If $\langle \_ \,,\, \_ \rangle$ is the concatenation, and samplings are done uniformly at random among bitstrings of length $\eta \in \mathbb{N}$, then folding:

$$\nu\, \mathsf{n}_0, \nu\, \mathsf{n}_1, \mathbf{out}(\mathsf{c}, \langle \mathsf{n}_0 \,,\, \langle 00 \,,\, \mathsf{n}_1 \rangle\rangle) \quad \text{yields} \quad \langle \mathsf{n}_0 \,,\, \langle 00 \,,\, \mathsf{n}_1 \rangle\rangle$$

which represent a distribution over bitstrings of length $2 \cdot \eta + 2$, where all bits are sampled uniformly and independently, except for the bits at positions η and $\eta + 1$, which are always 0.

Semantics of Terms

We interpret $t \in \mathcal{T}(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})$ as a **Probabilistic Polynomial-time Turing machine** (PPTM), with:

- a **working tape** (also used as input tape);
- two **read-only infinite tapes** $\rho = (\rho_p, \rho_a)$ for protocol and adversary randomness.

We let $\mathcal{D}$ be the set of such machines.

The machine must be polynomial in the size of its input on the working tape only (obviously).

Term Interpretation

The **interpretation** $[\![t]\!]^{\sigma}_{\mathcal{M}}$ is parameterized by:

- a **valuation** $\sigma : \mathcal{X} \mapsto \mathcal{D}$ of variables as PPTMs;
- a **computational model** $\mathcal{M}$, which interprets function symbols.

We often omit $\mathcal{M}$, as it is fixed throughout the interpretation.

We now define the machine $[\![t]\!]^{\sigma} \in \mathcal{D}$, by defining its behavior for every $\eta \in \mathbb{N}$ and pairs of random tapes $\rho = (\rho_{p}, \rho_{a})$.

Term Interpretation: Function Symbols

Function symbols interpretations is just **composition**.

For **function symbols** in $f \in \mathcal{F}$, we simply apply $[\![f]\!]_{\mathcal{M}}$:

$$[\![f(t_1, \dots, t_n)]\!]^{\sigma}(1^{\eta}, \rho) \stackrel{\text{def}}{=} [\![f]\!]_{\mathcal{M}}([\![t_1]\!]^{\sigma}(1^{\eta}, \rho), \dots, [\![t_n]\!]^{\sigma}(1^{\eta}, \rho))$$

Adversarial function symbols $g \in \mathcal{G}$ also have access to ρ_a:

$$[\![g(t_1, \dots, t_n)]\!]^{\sigma}(1^{\eta}, \rho) \stackrel{\text{def}}{=} [\![g]\!]_{\mathcal{M}}([\![t_1]\!]^{\sigma}(1^{\eta}, \rho), \dots, [\![t_n]\!]^{\sigma}(1^{\eta}, \rho), \rho_a)$$

Remark: $[\![f]\!]_{\mathcal{M}}$ and $[\![g]\!]_{\mathcal{M}}$ are **deterministic** (all randomness must come explicitly, from ρ).

Term Interpretation: Variables and Names

For **variables** in $\mathsf{x} \in \mathcal{X}$, we use σ:

$$[\![\mathsf{x}]\!]^{\sigma}(1^{\eta}, \rho) \stackrel{\text{def}}{=} \sigma(x)(1^{\eta}, \rho),$$

Names $\mathsf{n} \in \mathcal{G}$ are interpreted as **uniform random samplings** among bitstrings of length η, extracted from $\rho_{\boldsymbol{p}}$:

$$[\![\mathsf{n}]\!]^{\sigma}(1^{\eta}, \rho) \stackrel{\text{def}}{=} \mathsf{M}_{\mathsf{n}}(\eta, \rho_{\boldsymbol{p}})$$

For every pair of different names $\mathsf{n}_0, \mathsf{n}_1$, we require that $\mathsf{M}_{\mathsf{n}_0}$ and $\mathsf{M}_{\mathsf{n}_1}$ extracts disjoint parts of $\rho_{\boldsymbol{p}}$.

*Hence different names are **independent** random samplings.*

Term Interpretation: Builtins

We **force** the interpretation of some **function symbols**.

- if_then_else_ is interpreted as **branching**:

$$[\![\text{if } b \text{ then } t_1 \text{ else } t_2]\!]^\sigma(1^\eta, \rho) \overset{\text{def}}{=} \begin{cases} [\![t_1]\!]^\sigma(1^\eta, \rho) & \text{if } [\![t_1]\!]^\sigma(1^\eta, \rho) = 1 \\ [\![t_2]\!]^\sigma(1^\eta, \rho) & \text{otherwise} \end{cases}$$

- $\_ \doteq \_$ is interpreted as an **equality** test:

$$[\![t_1 \doteq t_2]\!]^\sigma(1^\eta, \rho) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } [\![t_1]\!]^\sigma(1^\eta, \rho) = [\![t_2]\!]^\sigma(1^\eta, \rho) \\ 0 & \text{otherwise} \end{cases}$$

Similarly, we force the interpretations of $\dot{\wedge}, \dot{\vee}, \dot{\rightarrow}, \text{true}, \text{false}$.

A First-Order Logic for Indistinguishability

A First-Order Logic for Indistinguishability

We now present a logic, to state (and later prove) **properties** about **bitstring distributions**.

This is a first-order logic with a single predicate $\sim$,[1] representing **computational indistinguishability**.

$$
\begin{aligned}
\phi := {} & \top \mid \bot \\
& \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid \neg\phi \\
& \mid \forall \mathsf{x}.\phi \mid \exists \mathsf{x}.\phi && (\mathsf{x} \in \mathcal{X}) \\
& \mid t_1, \ldots, t_n \sim_n t_{n+1}, \ldots, t_{2n} && (t_1, \ldots, t_{2n} \in \mathcal{T}(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X}))
\end{aligned}
$$

Remark: we use $\dot\wedge, \dot\vee, \dot\rightarrow$ in for the boolean *function symbols* in terms, to avoid confusion with the boolean *connectives* in formulas.

[1] Actually, one predicate $\sim_n$ of arity $2n$ for every $n \in \mathbb{N}$.

Semantics of the Logic

The logic has a **standard FO semantics**, using $\mathcal{D}$ as interpretation domain and interpreting $\sim$ as **computational indistinguishability**.

$[\![\phi]\!]^\sigma_\mathcal{M} \in \{\mathsf{True}, \mathsf{False}\}$ is as expected for **boolean connective** and FO **quantifiers**. E.g.:

$$[\![\top]\!]^\sigma_\mathcal{M} \stackrel{\mathsf{def}}{=} \mathsf{True} \qquad [\![\phi \wedge \psi]\!]^\sigma_\mathcal{M} \stackrel{\mathsf{def}}{=} [\![\phi]\!]^\sigma_\mathcal{M} \text{ and } [\![\psi]\!]^\sigma_\mathcal{M}$$

$$[\![\neg\phi]\!]^\sigma_\mathcal{M} \stackrel{\mathsf{def}}{=} \text{not } [\![\phi]\!]^\sigma_\mathcal{M}$$

$$[\![\forall \mathsf{x}.\phi]\!]^\sigma_\mathcal{M} \stackrel{\mathsf{def}}{=} \mathsf{True} \quad \text{ if } \forall m \in \mathcal{D}, [\![\phi]\!]^{\sigma[\mathsf{x}\mapsto m]}_\mathcal{M} \stackrel{\mathsf{def}}{=} \mathsf{True}$$

Semantics of the Logic

Finally, $\sim_n$ is interpreted as **computational indistinguishability**.

$$[\![t_1, \ldots, t_n \sim_n s_1, \ldots, s_n]\!]^\sigma_{\mathcal{M}} = \mathsf{True}$$

if, for every PPTM $\mathcal{A}$ with a $n+1$ input (and working) tapes, and a **single** infinite random tape:

$$\left| \begin{array}{rl} & \Pr_\rho\left(\mathcal{A}(1^\eta, ([\![t_i]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho))_{1 \leq i \leq n}, \rho_a) = 1\right) \\ - & \Pr_\rho\left(\mathcal{A}(1^\eta, ([\![s_i]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho))_{1 \leq i \leq n}, \rho_a) = 1\right) \end{array} \right| \qquad (\star)$$

is a **negligible** function of η.

*The quantity in $(\star)$ is called the **advantage** of $\mathcal{A}$ against the left/right game $t_1, \ldots, t_n \sim_n s_1, \ldots, s_n$*

Negligible Functions

A function $f(\eta)$ is **negligible** if it is **asymptotically smaller** than the **inverse** of any **polynomial**, i.e.:

$$\forall c \in \mathbb{N}, \exists N \in \mathbb{N} \text{ s.t. } \forall n \geq N, f(n) \leq \frac{1}{n^c}$$

Example

Let f be the function defined by:

$$f(\eta) \stackrel{\text{def}}{=} \Pr_\rho\big(\llbracket \mathsf{n}_0 \rrbracket(1^\eta, \rho) = \llbracket \mathsf{n}_1 \rrbracket(1^\eta, \rho)\big)$$

If $\mathsf{n}_0 \not\equiv \mathsf{n}_1$, then $f(\eta) = \frac{1}{2^\eta}$, and f is negligible.

Satisfiability and Validity

A formula ϕ is **satisfied** by a computational model $\mathcal{M}$, written $\mathcal{M} \models \phi$, if $[\![\phi]\!]^{\sigma}_{\mathcal{M}} = \text{True}$ for every valuation σ.

ϕ is **valid**, denoted by $\models \phi$, if it is **satisfied** by **every computational model**.

ϕ is **$\mathcal{C}$-valid** if it is satisfied by every computational model $\mathcal{M} \in \mathcal{C}$.

Validity: Exercise

Exercise

Which of the formulas below are **valid**? Which are not?

$\mathsf{true} \sim \mathsf{false}$ $\quad$ $\mathsf{n}_0 \sim \mathsf{n}_0$ $\quad$ $\mathsf{n}_0 \sim \mathsf{n}_1$ $\quad$ $\mathsf{n}_0 \doteq \mathsf{n}_1 \sim \mathsf{false}$

$\mathsf{n}_0, \mathsf{n}_0 \sim \mathsf{n}_0, \mathsf{n}_1$ $\quad$ $f(\mathsf{n}_0) \sim f(\mathsf{n}_1)$ where $f \in \mathcal{F} \cup \mathcal{G}$

$$\pi_1(\langle \mathsf{n}_0 , \mathsf{n}_1 \rangle) \doteq \mathsf{n}_0 \sim \mathsf{true}$$

Validity: Exercise

Exercise

Which of the formulas below are **valid**? Which are not?

$\not\models \mathsf{true} \sim \mathsf{false}$ $\quad \models \mathsf{n}_0 \sim \mathsf{n}_0$ $\quad \models \mathsf{n}_0 \sim \mathsf{n}_1$ $\quad \models \mathsf{n}_0 \doteq \mathsf{n}_1 \sim \mathsf{false}$

$\not\models \mathsf{n}_0, \mathsf{n}_0 \sim \mathsf{n}_0, \mathsf{n}_1$ $\quad \models f(\mathsf{n}_0) \sim f(\mathsf{n}_1)$ where $f \in \mathcal{F} \cup \mathcal{G}$

$\not\models \pi_1(\langle \mathsf{n}_0 \,,\, \mathsf{n}_1 \rangle) \doteq \mathsf{n}_0 \sim \mathsf{true}$

Protocol Indistinguishability

$\mathcal{P}$ and $\mathcal{Q}$ are **indistinguishable**, written $\mathcal{P} \approx \mathcal{Q}$, if for any τ:

$$\models \mathsf{fold}(\mathcal{P}, \tau) \sim \mathsf{fold}(\mathcal{Q}, \tau)$$

Remark

While there are countably many observable traces τ, the **set** of **foldings** of a protocol P is always **finite**:[2]

$$\left|\left\{\mathsf{fold}(\mathcal{P}, \tau) \mid \tau\right\}\right| < +\infty$$

[2] If we remove trailing sequences of `error` terms.

Protocol Indistinguishability: Exercise

Exercise

Informally, determine which of the following protocols **indistinguishabilities** hold, and under what **assumptions**:

$$\mathbf{out}(\mathsf{c}, t_1) \approx \mathbf{out}(\mathsf{c}, t_2) \qquad \mathbf{out}(\mathsf{c}, t) \approx \mathbf{null} \qquad \mathbf{in}(\mathsf{c}, \mathsf{x}) \approx \mathbf{null}$$

$$\mathbf{out}(\mathsf{c}, t) \approx \text{if } b \text{ then } \mathbf{out}(\mathsf{c}, t_1) \text{ else } \mathbf{out}(\mathsf{c}, t_2)$$

$$\mathbf{out}(\mathsf{c}, t) \approx \text{if } b \text{ then } \mathbf{out}(\mathsf{c}, t) \text{ else } \mathbf{out}(\mathsf{c}_0, t_0)$$

Structural Rules

Rules: Soundness

A rule:

$$\frac{\phi_1 \quad \dots \quad \phi_n}{\phi}$$

is sound if ϕ is **valid** whenever $\phi_1, \dots, \phi_n$ are **valid**.

Example

$$\frac{y \sim x}{x \sim y} \quad \text{is sound}$$

These are typically **structural rules**, which are valid in all **computational models**.

Structural Rules

Computational indistinguishability is an **equivalence relation**:

$$\frac{}{\vec{u} \sim \vec{u}}\ \textsc{Refl} \qquad \frac{\vec{v} \sim \vec{u}}{\vec{u} \sim \vec{v}}\ \textsc{Sym} \qquad \frac{\vec{u} \sim \vec{w} \qquad \vec{w} \sim \vec{v}}{\vec{u} \sim \vec{v}}\ \textsc{Trans}$$

Permutation. If π is a permutation of $\{1, \ldots, n\}$ then:

$$\frac{u_{\pi(1)}, \ldots, u_{\pi(n)} \sim v_{\pi(1)}, \ldots, v_{\pi(n)}}{u_1, \ldots, u_n \sim v_1, \ldots, v_n}\ \textsc{Perm}$$

Structural Rules

Alpha-renaming.

$$\frac{}{\vec{u} \sim \vec{u}\alpha}\ \alpha\text{-EQU}$$

when α is an injective renaming of names in $\mathcal{N}$.

Restriction. The adversary can throw away some values:

$$\frac{\vec{u}, s \sim \vec{v}, t}{\vec{u} \sim \vec{v}}\ \textsc{Restr}$$

Structural Rules

Duplication. Giving twice the same value to the adversary is useless:

$$\frac{\vec{u}, s \sim \vec{v}, t}{\vec{u}, s, s \sim \vec{v}, t, t} \text{ Dup}$$

Function application. If the arguments of a function are indistinguishable, so is the image:

$$\frac{\vec{u}_1, \vec{v}_1 \sim \vec{u}_1, \vec{v}_2}{f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2} \text{ FA}$$

where $f \in \mathcal{F} \cup \mathcal{G}$.

Structural Rules: Proof of Function Application

$$\frac{\vec{u}_1, \vec{v}_1 \sim \vec{u}_1, \vec{v}_2}{f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2}\ \text{FA}$$

Proof. The proof is by contrapositive. Assume $\mathcal{M}$, σ and $\mathcal{A}$ s.t. its advantage against:

$$f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2 \qquad (\dagger)$$

is not negligible. Let $\mathcal{B}$ be the *distinguisher* defined by, for any bitstrings $\vec{w}_u, \vec{w}_v$ and tape ρ_a:

$$\mathcal{B}(1^\eta, \vec{w}_u, \vec{w}_v, \rho_a) \stackrel{\text{def}}{=} \mathcal{A}(1^\eta, [\![f]\!]_{\mathcal{M}}(\vec{w}_u), \vec{w}_v, \rho_a)$$

$\mathcal{B}$ is a PPTM since $\mathcal{A}$ is and $[\![f]\!]_{\mathcal{M}}$ can be evaluated in pol. time. Then:

$$\begin{aligned} &\mathcal{B}(1^\eta, [\![\vec{u}_i]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho), [\![\vec{v}_i]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho), \rho_a) \\ = \ &\mathcal{A}(1^\eta, [\![f(\vec{u}_i)]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho), [\![\vec{v}_i]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho), \rho_a) \end{aligned} \qquad (i \in \{1, 2\})$$

Hence the advantage of $\mathcal{B}$ in distinguishing $\vec{u}_1, \vec{v}_1 \sim \vec{u}_1, \vec{v}_2$ is exactly the advantage of $\mathcal{A}$ in distinguishing $(\dagger)$. □

Structural Rules

Case Study. We can do case disjunction over branching terms:

$$\frac{\vec{w}_1, b_0, u_0 \sim \vec{w}_1, b_1, u_1 \qquad \vec{w}_0, b_0, v_0 \sim \vec{w}_1, b_1, v_1}{\vec{w}_0, \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \vec{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1} \text{ CS}$$

Structural Rules: Proof of Case Study

$$\frac{b_0, u_0 \sim b_1, u_1 \qquad b_0, v_0 \sim b_1, v_1}{t_0 \equiv \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim t_1 \equiv \text{if } b_1 \text{ then } u_1 \text{ else } v_1} \text{ CS}$$

Proof. (by contrapositive) Assume $\mathcal{M}$, σ and $\mathcal{A}$ s.t. its advantage against:

$$\text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \text{if } b_1 \text{ then } u_1 \text{ else } v_1 \qquad (\dagger)$$

is non-negligible. Let $\mathcal{B}_\top$ be the distinguisher:

$$\mathcal{B}_\top(1^\eta, w_b, w, \rho_a) \stackrel{\text{def}}{=} \begin{cases} \mathcal{A}(1^\eta, w, \rho_a) & \text{if } w_b = 1 \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{B}_\top$ is trivially a PPTM. Moreover, for any $i \in \{1, 2\}$:

$$\left.\begin{aligned} &\Pr_\rho\Big(\mathcal{B}_\top(1^\eta, [\![b_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho), [\![u_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho), \rho_a) = 1\Big) \\ = \; &\Pr_\rho\Big(\mathcal{A}(1^\eta, [\![t_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho), \rho_a) = 1 \wedge [\![b_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho) = 1\Big) \end{aligned}\right\} p_{\top,i}$$

Structural Rules: Proof of Case Study (continued)

Hence the advantage of $\mathcal{B}_\top$ against $b_0, u_0 \sim b_1, u_1$ is $|p_{\top,1} - p_{\top,0}|$.

Similarly, let $\mathcal{B}_\bot$ be the distinguisher:

$$\mathcal{B}_\bot(1^\eta, w_b, w, \rho_a) \stackrel{\text{def}}{=} \begin{cases} \mathcal{A}(1^\eta, w, \rho_a) & \text{if } w_b \neq 1 \\ 0 & \text{otherwise} \end{cases}$$

By an identical reasoning, we get that the advantage of $\mathcal{B}_\bot$ against $b_0, v_0 \sim b_1, v_1$ is $|p_{\bot,1} - p_{\bot,0}|$, where $p_{\bot,i}$ is:

$$\Pr_\rho\Big(\mathcal{A}(1^\eta, [\![t_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho), \rho_a) = 1 \wedge [\![b_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho) \neq 1\Big)$$

Structural Rules: Proof of Case Study (continued)

The advantage of $\mathcal{A}$ against $t_0 \sim t_1$ is, by partitioning and triangular inequality:

$$|(p_{\top,1} + p_{\perp,1}) - (p_{\top,0} + p_{\perp,1})| \leq |p_{\top,1} - p_{\top,0}| + |p_{\perp,1} - p_{\perp,1}|$$

Since $\mathcal{A}$'s advantage is non-negligible, at least one of the two quantity above is non-negligible. Hence either $\mathcal{B}_\top$ or $\mathcal{B}_\perp$ has a non-negligible advantage against a premise of the CS rule. $\square$.

Counter-Examples

Remark that b is **necessary** in CS

$$\frac{\vec{w}_1, b_0, u_0 \sim \vec{w}_1, b_1, u_1 \qquad \vec{w}_0, b_0, v_0 \sim \vec{w}_1, b_1, v_1}{\vec{w}_0, \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \vec{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1}\ \mathrm{CS}$$

We have:

$$\models 0 \sim 0 \qquad\qquad \models \mathsf{n}_0 \sim \mathsf{n}_1 \qquad\qquad \models \mathtt{even}(\mathsf{n}_0) \sim \mathtt{even}(\mathsf{n}_0)$$

But:

$$\not\models \text{if } \mathtt{even}(\mathsf{n}_0) \text{ then } \mathsf{n}_0 \text{ else } 0 \sim \text{if } \mathtt{even}(\mathsf{n}_0) \text{ then } \mathsf{n}_1 \text{ else } 0$$

Why is the later formula not valid?

Structural Rules: FO + Equality Reasoning

If $\models (s \doteq t) \sim \text{true}$, then s and t are **equal with overwhelming probability**. Hence we can **safely replace** s by t in **any context**.

Let $(s = t) \stackrel{\text{def}}{=} (s \doteq t) \sim \text{true}$. Then the following rule is sound:

$$\frac{\vec{u}, t \sim \vec{v} \qquad s = t}{\vec{u}, s \sim \vec{v}}\ \mathrm{R}$$

Structural Rules: FO + Equality Reasoning

To prove $\models s = t$, we use the following rule:

$$\frac{\mathcal{A}_{\mathsf{th}} \vdash_{\mathrm{FO}_=} s = t}{s = t}\ \mathrm{FO}$$

where $\vdash_{\mathrm{FO}_=}$ is any **sound proof system** for (classical) first-order logic with equality:

$$\mathcal{F}_{\mathrm{FO}}(\dot{\to}, \mathsf{false}, \doteq, \mathcal{F} \cup \mathcal{G})$$

We allow additional FO axioms using $\mathcal{A}_{\mathsf{th}}$ (e.g. for if_then_else_).

Example

$$\mathcal{A}_{\mathsf{th}} \vdash_{\mathrm{FO}_=} \left(v \doteq w \dot{\to} \text{if } u \doteq v \text{ then } u \text{ else } t \doteq s\right) = \left(v \doteq w \dot{\to} \text{if } u \doteq v \text{ then } w \text{ else } t \doteq s\right)$$

Structural Rules: Probabilistic Independence

Two rules exploiting the **independence** of bitstring distributions:

$$\frac{}{(t \doteq \mathsf{n}) = \mathsf{false}}\ \textsc{=-Ind} \quad \text{when } \mathsf{n} \not\in \mathsf{st}(t)$$

$$\frac{\vec{u} \sim \vec{v}}{\vec{u}, \mathsf{n}_0 \sim \vec{v}, \mathsf{n}_1}\ \textsc{Fresh} \quad \text{when } \mathsf{n}_0 \not\in \mathsf{st}(\vec{u}) \text{ and } \mathsf{n}_1 \not\in \mathsf{st}(\vec{v})$$

Remark

To check that the rules side-conditions hold, we require that they do not contain free variables. Hence we actually have a countable, recursive, set of **ground rules** (i.e. rule **schemata**).

Structural Rules: Probability Independence

We give the proof of the first rule:

$$\frac{}{(t \doteq \mathsf{n}) = \mathsf{false}} =\text{-IND} \quad \text{when } \mathsf{n} \notin \mathsf{st}(t)$$

Proof. For any computational model $\mathcal{M}$ (we omit it below):

$$\begin{aligned}
& \Pr_\rho([\![t \doteq \mathsf{n}]\!](1^\eta, \rho) = 1) \\
= \ & \Pr_\rho([\![t]\!](1^\eta, \rho) = [\![\mathsf{n}]\!](1^\eta, \rho)) \\
= \ & \textstyle\sum_{w \in \{0,1\}^*} \Pr_\rho([\![t]\!](1^\eta, \rho) = w \wedge [\![\mathsf{n}]\!](1^\eta, \rho) = w) \\
= \ & \textstyle\sum_{w \in \{0,1\}^*} \Pr_\rho([\![t]\!](1^\eta, \rho) = w) \cdot \Pr_\rho([\![\mathsf{n}]\!](1^\eta, \rho) = w) \\
= \ & \tfrac{1}{2^\eta} \cdot \textstyle\sum_{w \in \{0,1\}^*} \Pr_\rho([\![t]\!](1^\eta, \rho) = w) \\
= \ & \frac{1}{2^\eta}
\end{aligned}$$

□

Structural Rules: Exercise

Exercise

Give a **derivation** of the following formula:

$$\mathsf{n}_0 \sim \text{if } b \text{ then } \mathsf{n}_0 \text{ else } \mathsf{n}_1 \quad (\text{when } \mathsf{n}_0, \mathsf{n}_1 \notin \mathrm{st}(b))$$

Implementation Rules

Rules: Soundness

A rule is $\mathcal{C}$-**sound** if ϕ is $\mathcal{C}$-**valid** whenever $\phi_1, \ldots, \phi_n$ are $\mathcal{C}$-**valid**.

Example

$$\overline{(\pi_1 \langle x\,,\, y\rangle \doteq x) \sim \mathsf{true}}$$

is **not** sound, because we do not require anything on the interpretation of π_1 and the pair.

Obviously, it is $\mathcal{C}_\pi$-sound, where $\mathcal{C}_\pi$ is the set of computational model where π_1 computes the first projection of the pair $\langle \_\,,\, \_\rangle$.

Implementation Assumptions

The **general philosophy** of the CCSA approach is to make the minimum number of assumptions possible on the interpretations of function symbols in a computational model.

Any additional necessary assumption is added through rules, which restrict the set of computation model for which the formula holds (hence limit the scope of the final security result).

Typically, this is used for:

- **functional properties**, which must be satisfied by the protocol functions (e.g. the projection/pair rule).
- **cryptographic hardness assumptions**, which must be satisfied by the cryptographic primitives (e.g. IND-CCA).

Functional Properties

Example. **Equational theories** for protocol functions:

- $\pi_i\left(\langle x_1, x_2 \rangle\right) = x_i$ $\quad i \in \{1, 2\}$
- $\mathsf{dec}(\{x\}^z_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x$
- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
- ...

Cryptographic Rules

Cryptographic Reduction

Cryptographic reductions are the main tool used in proofs of computational security.

Cryptographic Reduction $\mathcal{S} \leq_{\textbf{red}} \mathcal{H}$

*If you can break the **cryptographic design** $\mathcal{S}$, then you can break the **hardness assumption** $\mathcal{H}$ using roughly the same **time**.*

- We assume that $\mathcal{H}$ cannot be broken in a reasonable time:
  - Low-level assumptions: D-Log, DDH, ...
  - Higher-level assumptions: IND-CCA, EUF-MAC, PRF, ...
- Hence, $\mathcal{S}$ **cannot be broken in a reasonable time**.

Cryptographic Reduction $\mathcal{S} \leq_{\mathsf{red}} \mathcal{H}$

$\mathcal{S}$ reduces to a hardness hypothesis $\mathcal{H}$ (e.g. IND-CCA, DDH) if:

$$\forall \mathcal{A}.\, \exists \mathcal{B}.\ \mathsf{Adv}^{\eta}_{\mathcal{S}}(\mathcal{A}) \leq P(\mathsf{Adv}^{\eta}_{\mathcal{H}}(\mathcal{B}), \eta)$$

where $\mathcal{A}$ and $\mathcal{B}$ are taken among PPTMs and P is a polynomial.

Cryptographic Rules

We are now going to give **rules** which capture some **cryptographic hardness hypotheses**.

The validity of these rules will be established through a **cryptographic reduction**.

- Asymmetric encryption: indistinguishability (IND-CCA_1) and key-privacy (KP-CCA_1);
- Hash function: collision-resistance (CR-HK);
- MAC: unforgeability (EUF-CMA);

Cryptographic Rules

Asymmetric Encryption

Asymmetric Encryption Scheme

An **asymmetric encryption scheme** contains:

- public and private key generation functions $\mathsf{pk}(\_), \mathsf{sk}(\_)$;
- **randomized**[3] encryption function $\{\_\}_{\_}^{\_}$;
- a decryption function $\mathsf{dec}(\_, \_)$

It must satisfies the functional equality:

$$\mathsf{dec}(\{x\}_{\mathsf{pk}(y)}^{z}, \mathsf{sk}(y)) = x$$

[3] The role of the randomization will become clear later.

IND-CCA$_1$ Security

An encryption scheme is **indistinguishable against chosen cipher-text attacks** (IND-CCA$_1$) iff. for every PPTM $\mathcal{A}$ with access to:

- a left-right oracle $\mathcal{O}_{\mathsf{LR}}^{b,\mathsf{n}}(\cdot,\cdot)$:

$$\mathcal{O}_{\mathsf{LR}}^{b,\mathsf{n}}(m_0, m_1) \stackrel{\text{def}}{=} \begin{cases} \{m_b\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}} & \text{if } \mathsf{len}(m_1) = \mathsf{len}(m_2) \quad (\mathsf{r}\ \textit{fresh}) \\ 0 & \text{otherwise} \end{cases}$$

- and a decryption oracle $\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}}(\cdot)$,

where $\mathcal{A}$ can call $\mathcal{O}_{\mathsf{LR}}$ once, and cannot call $\mathcal{O}_{\mathsf{dec}}$ after $\mathcal{O}_{\mathsf{LR}}$, then:

$$\left|\ \mathsf{Pr}_{\mathsf{n}}\left(\mathcal{A}^{\mathcal{O}_{\mathsf{LR}}^{\mathbf{1},\mathsf{n}},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}}}(1^\eta, \mathsf{pk}(\mathsf{n})) = 1\right) - \mathsf{Pr}_{\mathsf{n}}\left(\mathcal{A}^{\mathcal{O}_{\mathsf{LR}}^{\mathbf{0},\mathsf{n}},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}}}(1^\eta, \mathsf{pk}(\mathsf{n})) = 1\right)\ \right|$$

is negligible in η, where n is drawn uniformly in $\{0,1\}^\eta$.

Exercise

Show that if the encryption **ignore its randomness**, i.e. there exists aenc(_, _) s.t. for all x, y, r:

$$\{x\}_y^r = \mathrm{aenc}(x, y)$$

then the encryption does not satisfy IND-CCA$_1$.

IND-CCA$_1$ Rule

Indistinguishability Against Chosen Ciphertexts Attacks

If the encryption scheme is IND-CCA$_1$, then the *ground* rule:

$$\frac{\mathsf{len}(t_0) = \mathsf{len}(t_1)}{\vec{u}, \{t_0\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})} \sim \vec{u}, \{t_1\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})}} \ \text{IND-CCA}_1$$

is sound, when:

- r does not appear in $\vec{u}, t_0, t_1$;
- n appears only in $\mathsf{pk}(\cdot)$ or $\mathsf{dec}(\_, \mathsf{sk}(\cdot))$ positions in $\vec{u}, t_0, t_1$.

IND-CCA$_1$ Rule: Proof

Proof sketch

Proof by contrapositive. Let $\mathcal{M}$ be a comp. model, $\mathcal{A}$ an adversary and $\vec{u}, t_0, t_1$ ground terms such that:

$$\Big| \quad \mathrm{Pr}_\rho(\mathcal{A}(1^\eta, [\![\vec{u}]\!]_\mathcal{M}(1^\eta, \rho), [\![\{t_0\}^{\mathsf{r}}_{\mathsf{pk(n)}}]\!]_\mathcal{M}(1^\eta, \rho), \rho_a) - \mathrm{Pr}_\rho(\mathcal{A}(1^\eta, [\![\vec{u}]\!]_\mathcal{M}(1^\eta, \rho), [\![\{t_1\}^{\mathsf{r}}_{\mathsf{pk(n)}}]\!]_\mathcal{M}(1^\eta, \rho), \rho_a) \Big|$$

is not negligible, and $\mathcal{M} \models \mathrm{len}(t_0) = \mathrm{len}(t_1)$.

We must build a PPTM $\mathcal{B}$ s.t. $\mathcal{B}$ wins the IND-CCA$_1$ security game.

IND-CCA$_1$ Rule: Proof

Let $\mathcal{B}^{\mathcal{O}_{\mathbf{LR}}^{b,\mathbf{n}},\mathcal{O}_{\mathbf{dec}}^{\mathbf{n}}}(1^\eta, [\![\mathsf{pk}(\mathsf{n})]\!]_{\mathcal{M}}(1^\eta,\rho))$ be the following program:

i) **lazily** samples the infinite random tapes (ρ_a, ρ'_p) where:

$$\rho'_p := \rho_p[\mathsf{n} \mapsto 0, \mathsf{r} \mapsto 0]$$

ii) compute[4]:

$$w_{\vec{u}}, w_{t_0}, w_{t_1} := [\![\vec{u}, t_0, t_1]\!]_{\mathcal{M}}(1^\eta, \rho)$$

using (ρ_a, ρ'_p), $[\![\mathsf{pk}(\mathsf{n})]\!]_{\mathcal{M}}(1^\eta,\rho)$ and calls to $\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}}$.

iii) compute:

$$w_{lr} := \mathcal{O}_{\mathsf{LR}}^{b,\mathsf{n}}(w_{t_0}, w_{t_1}) = [\![\{t_b\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}}]\!]_{\mathcal{M}}$$

(since $\mathcal{M} \models \mathsf{len}(t_0) = \mathsf{len}(t_1)$)

iv) return $\mathcal{A}(1^\eta, w_{\vec{u}}, w_{lr}, \rho_a)$.

[4] we describe how later

IND-CCA$_1$ Rule: Proof

Then $\mathcal{B}$ advantage against IND-CCA$_1$ is exactly $\mathcal{A}$ advantage against:

$$\vec{u}, \{t_0\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})} \sim \vec{u}, \{t_1\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})}$$

which we assumed non-negligible.

IND-CCA$_1$ Rule: Proof

It only remains to explain how to do step *ii*) in polynomial time.

We prove by **structural induction** that for any subterm s of $\vec{u}, t_0, t_1$:

- either s is a forbidden subterm n, $\mathsf{sk}(\mathsf{n})$ or r;
- or $\mathcal{B}$ can compute $w_s := [\![s]\!]_{\mathcal{M}}(1^\eta, \rho)$ in polynomial time.

Assuming this holds, we conclude by observing that IND-CCA$_1$ side conditions guarantees that $\vec{u}, t_0, t_1$ are not forbidden subterms.

IND-CCA$_1$ Rule: Proof

Induction. We are in one of the following cases:

- $s \in \mathcal{X}$ is not possible, since $\vec{u}, t_0, t_1$ are ground.
- $s \in \{\mathsf{r}, \mathsf{n}\}$ are forbidden, hence the induction hypothesis holds.
- $s \in \mathcal{N} \backslash \{\mathsf{r}, \mathsf{n}\}$, then $\mathcal{B}$ computes s directly from $\rho'_p = \rho_p[\mathsf{n} \mapsto 0, \mathsf{r} \mapsto 0]$.
- $s \equiv f(t_1, \ldots, t_n)$ and $t_1, \ldots, t_n$ are not forbidden. Then, by induction hypothesis, $\mathcal{B}$ can compute $w_i := [\![t_i]\!]_{\mathcal{M}}(1^\eta, \rho)$ for any $1 \leq i \leq n$. Then $\mathcal{B}$ simply computes:

$$w_s := \begin{cases} [\![f]\!]_{\mathcal{M}}(w_1, \ldots, w_n) & \text{if } f \in \mathcal{F} \\ [\![f]\!]_{\mathcal{M}}(w_1, \ldots, w_n, \rho_a) & \text{if } f \in \mathcal{G} \end{cases}$$

IND-CCA$_1$ Rule: Proof

case disjunction (continued):

- $s \equiv f(t_1, \ldots, t_n)$ and at least one of the t_i is forbidden.

  Using IND-CCA$_1$ side conditions, either s is either $\mathsf{pk}(\mathsf{n})$, $\mathsf{sk}(\mathsf{n})$ or $\mathsf{dec}(m, \mathsf{sk}(\mathsf{n}))$.

  The first case is immediate since $\mathcal{B}$ receives $[\![\mathsf{pk}(\mathsf{n})]\!]_{\mathcal{M}}(1^\eta, \rho)$ as argument.

  The second case is a forbidden subterm, hence the induction hypothesis holds.

  For the last case, from IND-CCA$_1$ side conditions, we know that $m \neq \mathsf{r}$ and $m \neq \mathsf{n}$. Hence, by **induction hypothesis**, $\mathcal{B}$ can compute $w_m = [\![m]\!]_{\mathcal{M}}(1^\eta, \rho)$. We conclude using:

  $$w_s := \mathcal{O}^{\mathsf{n}}_{\mathsf{dec}}(w_m) \qquad \square$$

Exercise

Which of the following formulas can be proven using IND-CCA$_1$?

$$\mathsf{pk}(n), \{0\}^{r}_{\mathsf{pk}(n)} \sim \mathsf{pk}(n), \{1\}^{r}_{\mathsf{pk}(n)}$$

$$\mathsf{pk}(n), \{0\}^{r}_{\mathsf{pk}(n)}, \{0\}^{r_0}_{\mathsf{pk}(n)} \sim \mathsf{pk}(n), \{1\}^{r}_{\mathsf{pk}(n)}, \{0\}^{r_0}_{\mathsf{pk}(n)}$$

$$\mathsf{pk}(n), \{0\}^{r}_{\mathsf{pk}(n)}, \{0\}^{r}_{\mathsf{pk}(n)} \sim \mathsf{pk}(n), \{0\}^{r}_{\mathsf{pk}(n)}, \{1\}^{r}_{\mathsf{pk}(n)}$$

$$\mathsf{pk}(n), \{0\}^{r}_{\mathsf{pk}(n)} \sim \mathsf{pk}(n), \{\mathsf{sk}(n)\}^{r}_{\mathsf{pk}(n)}$$

IND-CCA$_1$ Rule: Exercise

Exercise (Hybrid Argument)

Prove the following formula using IND-CCA$_1$:

$$\{0\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_0}, \{1\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_1}, \dots, \{n\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_n} \sim \{0\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_0}, \{0\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_1}, \dots, \{0\}_{\mathsf{pk}(\mathsf{n})}^{\mathsf{r}_n}$$

Note: we assume that all plain-texts above have the same length (e.g. they are all represented over L bits, for L large enough)

KP-CCA$_1$ Security

A scheme provides **key privacy against chosen cipher-text attacks** (KP-CCA$_1$) iff for every PPTM $\mathcal{A}$ with access to:

- a left-right encryption oracle $\mathcal{O}_{\mathsf{LR}}^{b,\mathsf{n}_0,\mathsf{n}_1}(\cdot)$:

$$\mathcal{O}_{\mathsf{LR}}^{b,\mathsf{n}_0,\mathsf{n}_1}(m) \stackrel{\text{def}}{=} \{m\}_{\mathsf{pk}(\mathsf{n}_b)}^{\mathsf{r}} \qquad (\mathsf{r}\ \textit{fresh})$$

- and two decryption oracles $\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_0}(\cdot)$ and $\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_1}(\cdot)$,

where $\mathcal{A}$ can call $\mathcal{O}_{\mathsf{LR}}$ once, and cannot call the decryption oracles after $\mathcal{O}_{\mathsf{LR}}$, then:

$$\left| \begin{array}{l} \quad \Pr_{\mathsf{n}_0,\mathsf{n}_1}\left(\mathcal{A}^{\mathcal{O}_{\mathsf{LR}}^{1,\mathsf{n}_0,\mathsf{n}_1},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_0},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_1}}(1^\eta, \mathsf{pk}(\mathsf{n}_0), \mathsf{pk}(\mathsf{n}_1)) = 1\right) \\ - \Pr_{\mathsf{n}_0,\mathsf{n}_1}\left(\mathcal{A}^{\mathcal{O}_{\mathsf{LR}}^{0,\mathsf{n}_0,\mathsf{n}_1},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_0},\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}_1}}(1^\eta, \mathsf{pk}(\mathsf{n}_0), \mathsf{pk}(\mathsf{n}_1)) = 1\right) \end{array} \right|$$

is negligible in η, where $\mathsf{n}_0, \mathsf{n}_1$ are drawn in $\{0,1\}^\eta$.

Security Notions: Exercise

Exercise

Show that $\text{IND-CCA}_1 \not\Rightarrow \text{KP-CCA}_1$ and $\text{KP-CCA}_1 \not\Rightarrow \text{IND-CCA}_1$.

KP-CCA$_1$ Rule

Key Privacy Against Chosen Ciphertexts Attacks

If the encryption scheme is KP-CCA$_1$, then the *ground* rule:

$$\frac{}{\vec{u}, \{t\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n}_0)} \sim \vec{u}, \{t\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n}_1)}}\ \textsc{KP-CCA}_1$$

is sound, when:

- r does not appear in $\vec{u}, t$;
- $\mathsf{n}_0, \mathsf{n}_1$ appear only in $\mathsf{pk}(\cdot)$ or $\mathsf{dec}(\_, \mathsf{sk}(\cdot))$ positions in $\vec{u}, t$.

The **proof** is similar to the IND-CCA$_1$ soundness proof. We omit it.

Security Proof

Private Authentication: Anonymity

Lets now try to prove that PA v2 provides **anonymity**:

- I_X is the initiator with identity X;
- S_X is the server, accepting messages from X;

The adversary must not be able to distinguish $I_A \mid S_A$ from $I_C \mid S_A$.

$$
\begin{array}{ll}
I_X \;: \nu\, r.\;\; \nu\, n_I. & \mathbf{out}(c_I, \{\langle pk_X , n_I \rangle\}^{r}_{pk_S}) \\
S_X : \nu\, r_0. \nu\, n_S. \mathbf{in}(c_I, x). & \text{if } \pi_1(d) \doteq pk_X \\
& \quad \text{then } \mathbf{out}(c_S, \{\langle \pi_2(d) , n_S \rangle\}^{r_0}_{pk_X}) \\
& \quad \text{else } \;\mathbf{out}(c_S, \{0\}^{r_0}_{pk_X})
\end{array}
$$

We assume the encryption is IND-CCA_1 and KP-CCA_1.

Private Authentication: Anonymity

As we saw, an encryption **does not hide the length** of the plain-text. Hence, since $\mathsf{len}(\langle \mathsf{n_I}, \mathsf{n_S} \rangle) \neq \mathsf{len}(0)$, there is an attack:

$$\not\models \{\langle \mathsf{n_I}, \mathsf{n_S} \rangle\}_{\mathsf{pk_A}}^{\mathsf{r_0}} \sim \{0\}_{\mathsf{pk_C}}^{\mathsf{r_0}}$$

even if the encryption is $\mathsf{IND\text{-}CCA_1}$ and $\mathsf{KP\text{-}CCA_1}$.

Private Authentication: Anonymity

We **fix** the protocol by:

- adding a **length check**;
- using a **decoy** message of the correct length.

The PA Protocol, v3

$$\begin{array}{ll} \mathsf{I_X} : \nu\, \mathsf{r}.\ \nu\, \mathsf{n_I}. & \mathbf{out}(\mathtt{c_I}, \{\langle \mathsf{pk_X}\,,\, \mathsf{n_I}\rangle\}^{\mathsf{r}}_{\mathsf{pk_S}}) \\ \mathsf{S_X} : \nu\, \mathsf{r_0}.\, \nu\, \mathsf{n_S}.\, \mathbf{in}(\mathtt{c_I}, x).\ & \text{if } \pi_1(d) \doteq \mathsf{pk_X} \,\dot{\wedge}\, \mathsf{len}(\pi_2(d)) \doteq \mathsf{len}(\mathsf{n_S}) \\ & \quad \text{then } \mathbf{out}(\mathtt{c_S}, \{\langle \pi_2(d)\,,\, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}}) \\ & \quad \text{else } \ \mathbf{out}(\mathtt{c_S}, \{\langle \mathsf{n_S}\,,\, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}}) \end{array}$$

Private Authentication: Anonymity

$$\mathsf{I_X} : \nu\, \mathsf{r}.\ \nu\, \mathsf{n_I}. \qquad \mathbf{out}(\mathrm{c_I}, \{\langle \mathsf{pk_X}, \mathsf{n_I}\rangle\}^{\mathsf{r}}_{\mathsf{pk_S}})$$

$$\mathsf{S_X} : \nu\, \mathsf{r_0}. \nu\, \mathsf{n_S}. \mathbf{in}(\mathrm{c_I}, x). \text{ if } \pi_1(d) \doteq \mathsf{pk_X} \mathbin{\dot\wedge} \mathsf{len}(\pi_2(d)) \doteq \mathsf{len}(\mathsf{n_S})$$

$$\text{then } \mathbf{out}(\mathrm{c_S}, \{\langle \pi_2(d), \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}})$$

$$\text{else } \mathbf{out}(\mathrm{c_S}, \{\langle \mathsf{n_S}, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}})$$

To prove $\mathsf{I_A} \mid \mathsf{S_A} \approx \mathsf{I_C} \mid \mathsf{S_A}$, we have several **traces**:

$\mathbf{in}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_S})$ $\qquad$ $\mathbf{in}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_S}), \mathbf{out}(\mathrm{c_I})$

$\mathbf{out}(\mathrm{c_I}), \mathbf{in}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_S})$ $\qquad$ $\mathbf{out}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_S}), \mathbf{in}(\mathrm{c_I})$

$\mathbf{out}(\mathrm{c_S}), \mathbf{in}(\mathrm{c_I}), \mathbf{out}(\mathrm{c_I})$ $\qquad$ $\mathbf{out}(\mathrm{c_S}), \mathbf{out}(\mathrm{c_S}), \mathbf{in}(\mathrm{c_I})$

Private Authentication: Anonymity

$$\mathsf{I_X} : \nu\, \mathsf{r}.\ \nu\, \mathsf{n_I}.\qquad \mathbf{out}(c_{\mathtt{I}}, \{\langle \mathsf{pk_X}, \mathsf{n_I}\rangle\}^{\mathsf{r}}_{\mathsf{pk_S}})$$

$$\mathsf{S_X} : \nu\, \mathsf{r_0}.\ \nu\, \mathsf{n_S}.\ \mathbf{in}(c_{\mathtt{I}}, x).\ \text{if } \pi_1(d) \doteq \mathsf{pk_X} \mathbin{\dot\wedge} \mathsf{len}(\pi_2(d)) \doteq \mathsf{len}(\mathsf{n_S})$$

$$\text{then } \mathbf{out}(c_{\mathtt{S}}, \{\langle \pi_2(d), \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}})$$

$$\text{else } \mathbf{out}(c_{\mathtt{S}}, \{\langle \mathsf{n_S}, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}})$$

To prove $\mathsf{I_A} \mid \mathsf{S_A} \approx \mathsf{I_C} \mid \mathsf{S_A}$, we have several **traces**:

$\mathbf{in}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{S}})$ $\qquad$ $\mathbf{in}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{S}}), \mathbf{out}(c_{\mathtt{I}})$

$\mathbf{out}(c_{\mathtt{I}}), \mathbf{in}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{S}})$ $\qquad$ $\mathbf{out}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{S}}), \mathbf{in}(c_{\mathtt{I}})$

$\mathbf{out}(c_{\mathtt{S}}), \mathbf{in}(c_{\mathtt{I}}), \mathbf{out}(c_{\mathtt{I}})$ $\qquad$ $\mathbf{out}(c_{\mathtt{S}}), \mathbf{out}(c_{\mathtt{S}}), \mathbf{in}(c_{\mathtt{I}})$

But there is a **more general trace**: its security implies the security of the other traces.

See **partial order reduction** (POR) techniques [1].

Private Authentication: Anonymity

We must prove that:

$$\mathsf{out}_1^{\mathsf{A}}, \mathsf{out}_2^{\mathsf{A},\mathsf{A}}[\mathsf{out}_1^{\mathsf{A}}] \sim \mathsf{out}_1^{\mathsf{C}}, \mathsf{out}_2^{\mathsf{A},\mathsf{A}}[\mathsf{out}_1^{\mathsf{C}}]$$

where:

$$\begin{aligned}
\mathsf{out}_1^{\mathsf{X}} &\equiv \{\langle \mathsf{pk_X}\,,\,\mathsf{n_I}\rangle\}_{\mathsf{pk_S}}^{\mathsf{r}}) \\
\mathsf{out}_2^{\mathsf{X},\mathsf{Y}}[\mathsf{M}] &\equiv \text{if } \pi_1(d[\mathsf{M}]) \doteq \mathsf{pk_X} \mathrel{\dot\wedge} \mathsf{len}(\pi_2(d[\mathsf{M}])) \doteq \mathsf{len}(\mathsf{n_S}) \\
&\qquad \text{then } \{\langle \pi_2(d[\mathsf{M}])\,,\,\mathsf{n_S}\rangle\}_{\mathsf{pk_Y}}^{\mathsf{r_0}} \\
&\qquad \text{else } \{\langle \mathsf{n_S}\,,\,\mathsf{n_S}\rangle\}_{\mathsf{pk_Y}}^{\mathsf{r_0}} \\
d[\mathsf{M}] &\equiv \mathsf{dec}(\mathbf{att}_0([\mathsf{M}]), \mathsf{sk_S})
\end{aligned}$$

Private Authentication: Anonymity

First, we push the branching under the encryption:

$$\frac{\mathsf{out}_1^{\mathsf{A}}, \mathsf{out}_2^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{A}}] \sim \mathsf{out}_1^{\mathsf{C}}, \underline{\mathsf{out}_2}^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{C}}] \quad \overline{\mathsf{out}_2^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{C}}] = \underline{\mathsf{out}_2}^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{C}}]}}{\mathsf{out}_1^{\mathsf{A}}, \mathsf{out}_2^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{A}}] \sim \mathsf{out}_1^{\mathsf{C}}, \mathsf{out}_2^{\mathsf{A,A}}[\mathsf{out}_1^{\mathsf{C}}]}\ \mathrm{R}$$

where:

$$\underline{\mathsf{out}_2}^{\mathsf{X,Y}}[\mathsf{M}] \equiv \left\{ \begin{array}{l} \text{if } \pi_1(d[\mathsf{M}]) \doteq \mathsf{pk}_{\mathsf{X}} \mathbin{\dot\wedge} \mathsf{len}(\pi_2(d[\mathsf{M}])) \doteq \mathsf{len}(\mathsf{n}_{\mathsf{S}}) \\ \text{then } \langle \pi_2(d[\mathsf{M}])\, ,\, \mathsf{n}_{\mathsf{S}} \rangle \\ \text{else } \langle \mathsf{n}_{\mathsf{S}}\, ,\, \mathsf{n}_{\mathsf{S}} \rangle \end{array} \right\}_{\mathsf{pk}_{\mathsf{Y}}}^{\mathsf{r}_0}$$

We let $m_{\mathsf{X}}[\mathsf{M}]$ be the content of the encryption above.

Private Authentication: Anonymity

Then, we use KP-CCA$_1$ to change the encryption key:

$$\dfrac{\begin{array}{l} \mathsf{out}_1^{A}, \mathsf{out}_2^{A,A}[\mathsf{out}_1^{A}] \\ \sim\ \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,C}[\mathsf{out}_1^{C}] \end{array} \qquad \begin{array}{l} \dfrac{}{\mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,C}[\mathsf{out}_1^{C}]}\ \text{KP-CCA}_1 \\ \sim\ \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,A}[\mathsf{out}_1^{C}] \end{array}}{\mathsf{out}_1^{A}, \mathsf{out}_2^{A,A}[\mathsf{out}_1^{A}] \sim \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,A}[\mathsf{out}_1^{C}]}\ \textsc{Trans}$$

since:

- the encryption randomness r_0 is correctly used;
- the key randomness n_A and n_B appear only in pk($\cdot$) and dec(_, sk($\cdot$)) positions.

Private Authentication: Anonymity

Then, we use IND-CCA$_1$ to change the encryption content:

$$\dfrac{\begin{array}{l} \mathsf{out}_1^{A}, \mathsf{out}_2^{A,A}[\mathsf{out}_1^{A}] \\ \sim\ \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{C,C}[\mathsf{out}_1^{C}] \end{array} \qquad \begin{array}{r} \dfrac{\mathsf{len}(m_C[\mathsf{out}_1^{C}]) = \mathsf{len}(m_A[\mathsf{out}_1^{A}])}{\mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{C,C}[\mathsf{out}_1^{A}]}\ \textsc{ind-cca}_1 \\ \sim\ \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,C}[\mathsf{out}_1^{C}] \end{array}}{\mathsf{out}_1^{A}, \mathsf{out}_2^{A,A}[\mathsf{out}_1^{A}] \sim \mathsf{out}_1^{C}, \underline{\mathsf{out}_2}^{A,C}[\mathsf{out}_1^{C}]}\ \textsc{Trans}$$

since:

- the encryption randomness r_0 is correctly used;
- the key randomness n_C appear only in $\mathsf{pk}(\cdot)$ and $\mathsf{dec}(\_, \mathsf{sk}(\cdot))$ positions.

Private Authentication: Anonymity

Recall that:

$$m_X[M] \equiv \text{if } \pi_1(d[M]) \doteq \mathsf{pk}_X \mathbin{\dot\wedge} \mathsf{len}(\pi_2(d[M])) \doteq \mathsf{len}(n_S) \\ \text{then } \langle \pi_2(d[M])\,,\, n_S \rangle \\ \text{else } \langle n_S\,,\, n_S \rangle$$

Then:

$$\frac{\mathsf{len}(m_C[\mathsf{out}_1^C]) = \mathsf{len}(m_A[\mathsf{out}_1^A])}{\mathsf{len}(m_C[\mathsf{out}_1^C]) = \mathsf{len}(m_A[\mathsf{out}_1^A])}\ \mathrm{FO}$$

if $\mathcal{A}_{\mathsf{th}}$ contains the axiom[5]:

$$\forall x, y. \mathsf{len}(\langle x\,,\, y \rangle) = c_{\langle\_\,,\,\_\rangle}(\mathsf{len}(x), \mathsf{len}(y))$$

where $c_{\langle\_\,,\,\_\rangle}(\cdot,\cdot)$ is left unspecified.

[5]This axiom must be satisfied by the protocol implementation for the security proof to apply.

Private Authentication: Anonymity

Then, we α-rename the key randomness $\mathsf{n_C}$, rewrite back the encryption, and conclude.

$$\frac{}{\mathsf{out}_1^{\mathsf{A}}, \mathsf{out}_2^{\mathsf{A},\mathsf{A}}[\mathsf{out}_1^{\mathsf{A}}] \sim \mathsf{out}_1^{\mathsf{C}}, \underline{\mathsf{out}_2}^{\mathsf{C},\mathsf{C}}[\mathsf{out}_1^{\mathsf{C}}]}\ \alpha\text{-}\textsc{equ} + \mathrm{R} + \textsc{Refl}$$

Privacy

Privacy

We proved **anonymity** of the Private Authentication protocol, which we defined as:

$$I_A \mid S_A \approx I_C \mid S_A$$

But does this really guarantees that this protocol protects the privacy of its users?

$\Rightarrow$ **No, because of linkability attacks**

Linkability Attacks

Consider the following authentication protocol, called KCL, between a reader R and a tag T_X with identity X:

$$
\begin{aligned}
R \; &: \nu\, n_R. \qquad\qquad\qquad \mathbf{out}(c_R, n_R) \\
T_X \; &: \nu\, n_T.\, \mathbf{in}(c_R, x).\, \mathbf{out}(c_I, \langle X \oplus n_T\,,\, n_T \oplus H(x, k_X)\rangle)
\end{aligned}
$$

Assuming H is a PRF (Pseudo-Random Function), and $\oplus$ is the exclusive-or, we can prove that KCL provides **anonymity**.

$$T_A \mid R \approx T_B \mid R$$

Linkability Attacks

But there are **privacy attacks** against KCL, using two sessions:

$1: \mathsf{E} \rightarrow \mathsf{T_A} : \mathsf{n}_R$	$\mathsf{E} \rightarrow \mathsf{T_A} : \mathsf{n}_R$
$2: \mathsf{T_A} \rightarrow \mathsf{E} : \langle \mathsf{A} \oplus \mathsf{n}_T, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k_A})\rangle$	$\mathsf{T_A} \rightarrow \mathsf{E} : \langle \mathsf{A} \oplus \mathsf{n}_T, \mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k_A})\rangle$
$3: \mathsf{E} \rightarrow \mathsf{T_A} : \mathsf{n_R}$	$\mathsf{E} \rightarrow \mathsf{T_B} : \mathsf{n_R}$
$4: \mathsf{T_A} \rightarrow \mathsf{E} : \langle \mathsf{A} \oplus \mathsf{n}'_\mathsf{T}, \mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n_R}, \mathsf{k_A})\rangle$	$\mathsf{T_B} \rightarrow \mathsf{E} : \langle \mathsf{B} \oplus \mathsf{n}'_\mathsf{T}, \mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n_R}, \mathsf{k_B})\rangle$

Let t_2 and t_4 be the outputs of T. Then, on the left scenario:

$$\begin{aligned}\pi_2(t_2) \oplus \pi_2(t_4) &= \left(\mathsf{n}_T \oplus \mathsf{H}(\mathsf{n}_R, \mathsf{k_A})\right) \oplus \left(\mathsf{n}'_T \oplus \mathsf{H}(\mathsf{n_R}, \mathsf{k_A})\right) \\ &= \mathsf{n}_T \oplus \mathsf{n}'_T \\ &= \pi_1(t_2) \oplus \pi_1(t_4)\end{aligned}$$

The same equality check will almost never hold on the right, under reasonable assumption on H.

Linkability Attacks

We just saw an **attack** against:

$$(\mathsf{T_A} \mid \mathsf{R}) \mid (\mathsf{T_A} \mid \mathsf{R}) \approx (\mathsf{T_A} \mid \mathsf{R}) \mid (\mathsf{T_B} \mid \mathsf{R})$$

Unlinkability

To prevent such attacks, we need to prove a stronger property, called **unlinkability**. It requires to prove the **equivalence** between:

- a **real-world**, where each agent can run **many sessions**:

$$\nu\, \vec{k}_0, \ldots, \vec{k}_N.\ !_{\mathrm{id} \le N}\ !_{\mathtt{sid} \le M}\ P(\vec{k}_{\mathrm{id}})$$

- and an **ideal-world**, where each agent run at most a **single session**:

$$\nu\, \vec{k}_{0,0}, \ldots, \vec{k}_{N,M}.\ !_{\mathrm{id} \le N}\ !_{\mathtt{sid} \le M}\ P(\vec{k}_{\mathrm{id},\mathtt{sid}})$$

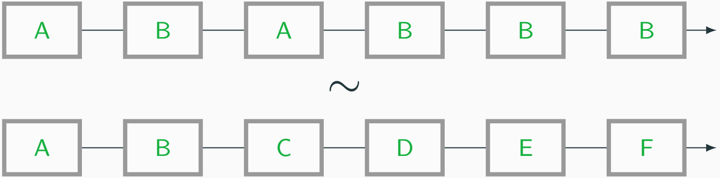
Remark

The processes above are parameterized by $N, M \in \mathbb{N}$. Unlinkability holds if the equivalence holds for any N, M.

For the sack of simplicity, we omit channel names.

Unlinkability

Example An unlinkability scenario.



Unlinkability: Intuition

In the **ideal-world**, relations between sessions **cannot leak** any **information** on identities.
$\Rightarrow$ hence **no link** can be **efficiently found** in the **real word**.

Unlinkability: Adding Servers

Our definition of **unlinkability** did not account for the **server**.

User-specific server, accepting a single identity.
The processes $P(\vec{k}_S, \vec{k}_U)$ and $S(\vec{k}_S, \vec{k}_U)$ are parameterized by:

- some **global** key material $\vec{k}_S$;
- and some **user-specific** key material $\vec{k}_U$.

Then, we require that:

$$\begin{aligned} & \nu\,\vec{k}_S.\ \nu\,\vec{k}_0, \ldots, \vec{k}_N. \quad !_{\mathrm{id}\leq N}\ !_{\mathtt{sid}\leq M}\ \big(P(\vec{k}_S, \vec{k}_{\mathrm{id}}) \ \mid S(\vec{k}_S, \vec{k}_{\mathrm{id}})\big) \\ \approx\ & \nu\,\vec{k}_S.\ \nu\,\vec{k}_{0,0}, \ldots, \vec{k}_{N,M}.\ !_{\mathrm{id}\leq N}\ !_{\mathtt{sid}\leq M}\ \big(P(\vec{k}_S, \vec{k}_{\mathrm{id}_{\mathtt{sid}}}) \mid S(\vec{k}_S, \vec{k}_{\mathrm{id}_{\mathtt{sid}}})\big) \end{aligned}$$

Unlinkability: Adding Servers

Generic server, accepting all identities.

No changes for the user process $P(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{\mathsf{U}})$.

The server $S(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{\mathsf{U}_1}, \dots, \vec{\mathsf{k}}_{\mathsf{U}_M})$ is parameterized by:

- some **global** key material $\vec{\mathsf{k}}_{\mathsf{S}}$;
- **all users** key material $\vec{\mathsf{k}}_{\mathsf{U}_1}, \dots, \vec{\mathsf{k}}_{\mathsf{U}_M}$.

The we require that:

$$\begin{aligned}
&\nu\, \vec{\mathsf{k}}_{\mathsf{S}}.\, \nu\, \vec{\mathsf{k}}_{0}, \dots, \vec{\mathsf{k}}_{N}. && \left(!_{\mathsf{id} \le N}\ !_{\mathtt{sid} \le M}\ P(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{\mathsf{id}})\right) \ \mid \\
& && \left(!_{\le L}\ S(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{0}, \dots, \vec{\mathsf{k}}_{N})\right) \\
\approx\ &\nu\, \vec{\mathsf{k}}_{\mathsf{S}}.\, \nu\, \vec{\mathsf{k}}_{0,0}, \dots, \vec{\mathsf{k}}_{N,M}. && \left(!_{\mathsf{id} \le N}\ !_{\mathtt{sid} \le M}\ P(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{\mathsf{id},\mathtt{sid}})\right) \ \mid \\
& && \left(!_{\le L}\ S(\vec{\mathsf{k}}_{\mathsf{S}}, \vec{\mathsf{k}}_{0,0}, \dots, \vec{\mathsf{k}}_{N,M})\right)
\end{aligned}$$

Private Authentication: Unlinkability

Private Authentication

We parameterize the initiator and server in **PA** by the key material:

$$\mathsf{I}(\mathsf{k_S}, \mathsf{k_X}) \;:\; \nu\, \mathsf{r}.\; \nu\, \mathsf{n_I}.\; \mathbf{out}(\mathsf{c_I}, \{\langle \mathsf{pk_X}\,,\, \mathsf{n_I}\rangle\}^{\mathsf{r}}_{\mathsf{pk_S}})$$

$$\begin{aligned}\mathsf{S}(\mathsf{k_S}, \mathsf{k_X}) \;:\; \nu\, \mathsf{r_0}.\, \nu\, \mathsf{n_S}.\, \mathbf{in}(\mathsf{c_I}, x).\; & \text{if } \pi_1(d) \doteq \mathsf{pk_X} \mathbin{\dot\wedge} \mathsf{len}(\pi_2(d)) \doteq \mathsf{len}(\mathsf{n_S}) \\ & \quad \text{then } \mathbf{out}(\mathsf{c_S}, \{\langle \pi_2(d)\,,\, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}}) \\ & \quad \text{else } \; \mathbf{out}(\mathsf{c_S}, \{\langle \mathsf{n_S}\,,\, \mathsf{n_S}\rangle\}^{\mathsf{r_0}}_{\mathsf{pk_X}})\end{aligned}$$

where $\mathsf{sk_X} \equiv \mathsf{sk}(\mathsf{k_X})$, $\mathsf{pk_X} \equiv \mathsf{pk}(\mathsf{k_X})$ and $d \equiv \mathsf{dec}(x, \mathsf{sk_S})$.

Private Authentication: Unlinkability

Theorem

Private Authentication, v3 satisfies the **unlinkability** property (with user-specific server). I.e., for all $N, M \in \mathbb{N}$:

$$\begin{aligned} & \nu\, \mathsf{k_S}.\, \nu\, \mathsf{k}_0, \ldots, \mathsf{k}_N. \quad !_{\mathsf{id} \le N}\; !_{\mathtt{sid} \le M} \left(I(\mathsf{k_S}, \mathsf{k_{id}}) \quad \mid S(\mathsf{k_S}, \mathsf{k_{id}}) \right) \\ \approx\; & \nu\, \mathsf{k_S}.\, \nu\, \mathsf{k}_{0,0}, \ldots, \mathsf{k}_{N,M}.\; !_{\mathsf{id} \le N}\; !_{\mathtt{sid} \le M} \left(I(\mathsf{k_S}, \mathsf{k}_{\mathsf{id}_{\mathtt{sid}}}) \mid S(\mathsf{k_S}, \mathsf{k}_{\mathsf{id}_{\mathtt{sid}}}) \right) \end{aligned}$$

Proof

For all N, M, for all trace of observables $\mathtt{tr}$, we show that:

$$\models \mathsf{fold}(\mathcal{P}_{\mathcal{L}}, \mathtt{tr}) \sim \mathsf{fold}(\mathcal{P}_{\mathcal{R}}, \mathtt{tr})$$

by induction over $\mathtt{tr}$, where $\mathcal{P}_{\mathcal{L}}$ and $\mathcal{P}_{\mathcal{R}}$ are, resp., the left and right protocols in the theorem above.

For details, see the Squirrel file `private-authentication-many.sp`.

Unlinkability: Remark

Note that **user-specific unlinkability** is a very strong property, that do not often hold.

Example

Assume S **leaks whether it succeeded** or not. This models the fact that the adversary can **distinguish success from failure**:

- e.g. because a door opens, which can be observed;
- or because success is followed by further communication, while failure is followed by a new authentication attempt.

Then the following unlinkability scenario **does not hold**:

$$(\underline{P(\vec{\mathsf{k}})} \mid S(\vec{\mathsf{k}})) \mid (P(\vec{\mathsf{k}}) \mid \underline{S(\vec{\mathsf{k}})}) \approx (\underline{P(\vec{\mathsf{k}}_0)} \mid S(\vec{\mathsf{k}}_0)) \mid (P(\vec{\mathsf{k}}_1) \mid \underline{S(\vec{\mathsf{k}}_1)})$$

Authentication Protocols

Authentication Protocol

We now focus on another class of security properties: **reachability** and **correspondance properties** (e.g. **authentication**)

These are properties on a **single** protocol, often expressed as a **temporal** property on **events** of the protocol. E.g.

> *If **Alice** accepts **Bob** at time τ then **Bob** must have initiated a session with Alice at time $\tau' < \tau$.*

To formalize the **cryptographic arguments** proving such properties, we will design a specialized **framework** and **proof system**.

The Hash-Lock Protocol

Let $\mathcal{I}$ be a finite set of identities.

$$\begin{aligned}
\mathsf{T}(\mathsf{A},\mathtt{i}) &: \nu\, n_{T,i}.\ \mathbf{in}(c^{T}_{A,i}, x).\ \mathbf{out}(c^{T}_{A,i}, \langle n_{T,i}\,,\ \mathsf{H}(\langle x\,,\ n_{T,i}\rangle, k_A)\rangle) \\
\mathsf{R}(\mathtt{j}) &: \nu\, n_{R,j}.\ \mathbf{in}(c^{R_1}_{j}, _).\ \mathbf{out}(c^{R_1}_{j}, n_{R,j}).\ \mathbf{in}(c^{R_2}_{j}, y). \\
&\qquad \text{if } \dot{\bigvee}_{A\in\mathcal{I}} \pi_2(y) \doteq \mathsf{H}(\langle n_{R,j}\,,\ \pi_1(y)\rangle, k_A) \\
&\qquad \text{ then } \mathbf{out}(c^{R_2}_{j}, ok) \\
&\qquad \text{ else } \mathbf{out}(c^{R_2}_{j}, ko)
\end{aligned}$$

We consider the N session of each tag, and M session of the reader:

$$\nu\,(k_A)_{A\in\mathcal{I}}.\ \big(!_{A\in\mathcal{I}}\ !_{i<N}\ \mathsf{T}(\mathsf{A},\mathtt{i})\big)\ \mid\ \big(!_{j<M}\ \mathsf{R}(\mathtt{j})\big)$$

Remark: we let the adversary do the scheduling between parties.

Notations

- we let $\leq$ be the **prefix relation** over observable traces:

$$\mathtt{tr}_0 \leq \mathtt{tr}_1 \text{ iff. } \exists \mathtt{tr}'.\ \mathtt{tr}_1 = \mathtt{tr}_0; \mathtt{tr}'$$

- $\mathtt{tr} \diamond \mathtt{c}$ states that $\mathtt{tr}$ **ends with an output** on $\mathtt{c}$:

$$\mathtt{tr} \diamond \mathtt{c} \text{ iff. } \exists \mathtt{tr}'.\ \mathtt{tr} = \mathtt{tr}'; \mathbf{out}(\mathtt{c})$$

Remark: $\mathtt{tr} \diamond \mathtt{c} \leq \mathtt{tr}'$ denotes that $\mathtt{tr} \diamond \mathtt{c} \wedge \mathtt{tr} \leq \mathtt{tr}'$.

POR Result (Assumed)

We let $\mathcal{T}_{\mathsf{io}}$ be the set of observable traces where all outputs are always **directly preceded** by an input on the same channel, i.e.:

$$\mathtt{tr} \in \mathcal{T}_{\mathsf{io}} \quad \text{iff.} \quad \forall \mathtt{tr}' \diamond \mathtt{c} \leq \mathtt{tr}.\ \exists \mathtt{tr}''.\ \mathtt{tr}' = \mathtt{tr}''; \mathbf{in}(\mathtt{c}); \mathbf{out}(\mathtt{c})$$

Assumption: POR

We admit that to analyze the Hash-Lock protocol, it is sufficient to consider only observables traces in $\mathcal{T}_{\mathsf{io}}$.

Authentication

Informal Definition

If the j-th session of R accepts believing it talked to tag A, then:

- there exists a session i of tag A **properly interleaved** with the j-th session of R;
- **messages** have been **properly forwarded** between the i-th session of tag A and the j-th session of R.

The second condition is often relaxed to require only a partial correspondence between messages.

Authentication of the Hash-Lock Protocol

For any $\mathtt{tr} \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R_2}} \in \mathcal{T}_{\mathsf{io}}$, we let $\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}$ be a term stating that the reader accepts the tag A at the end of the trace $\mathtt{tr}$ (defined later).

Authentication of the Hash-Lock Protocol

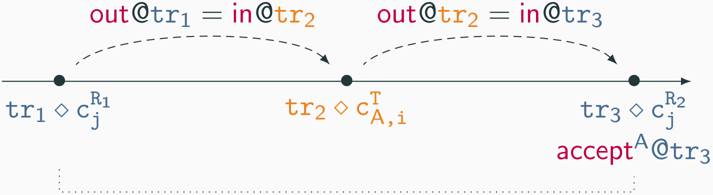
Informally, Hash-Lock provides authentication if for all $\mathtt{tr} \in \mathcal{T}_{\text{io}}$, $\mathtt{tr}_1 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_1}$ and $\mathtt{tr}_3 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_2}$ such that:

$$\mathtt{tr}_1 < \mathtt{tr}_3 \leq \mathtt{tr} \qquad \text{and} \qquad \mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3$$

there must exists $\mathtt{tr}_2 \diamond \mathtt{c}_{\mathtt{A,i}}^{\mathtt{T}}$ such that $\mathtt{tr}_1 \leq \mathtt{tr}_2 \leq \mathtt{tr}_3$ and:

$$\mathsf{out}@\mathtt{tr}_1 = \mathsf{in}@\mathtt{tr}_2 \wedge \mathsf{out}@\mathtt{tr}_2 = \mathsf{in}@\mathtt{tr}_3$$

Graphically:



Authentication of the Hash-Lock Protocol

What do we lack to formalize and prove the **authentication** of the **Hash-Lock** protocol?

- define the (generic) **terms representing** the output, input and acceptance, which we need to state the property;
- have a set of sound **one-sided** rules, to do the proof.

Authentication Protocols

Macro Terms

Notations: Predecessor

For any **observable trace** $\mathtt{tr}$ and **observable** α, we let:

$$\mathsf{pred}(\mathtt{tr}; \alpha) \overset{\text{def}}{=} \mathtt{tr}$$

Macro Terms

We now define some **generic** terms, called **macros**, by **induction** of the observable trace $\mathtt{tr}$.

Let $\mathcal{P}$ be a action-deterministic protocol and $\mathtt{tr} \in \mathcal{T}_{\mathsf{io}}$ with j inputs. If $\mathsf{fold}(\mathcal{P}, \mathtt{tr}) = t_1, \ldots, t_n$ then we let:

$$\mathsf{out}_{\mathcal{P}}@\mathtt{tr} \stackrel{\text{def}}{=} \begin{cases} t_n & \text{if } \exists \mathtt{c}.\ t_n \diamond \mathtt{c} \\ \mathsf{empty} & \text{otherwise} \end{cases}$$

$$\mathsf{frame}_{\mathcal{P}}@\mathtt{tr} \stackrel{\text{def}}{=} \begin{cases} \langle \mathsf{frame}_{\mathcal{P}}@\mathsf{pred}(\mathtt{tr}), \mathsf{out}_{\mathcal{P}}@\mathtt{tr} \rangle & \text{if } \mathtt{tr} \neq \epsilon \\ \mathsf{empty} & \text{if } \mathtt{tr} = \epsilon \end{cases}$$

$$\mathsf{in}_{\mathcal{P}}@(\mathtt{tr}; \mathbf{in}(\mathtt{c}); \mathbf{out}(\mathtt{c})) \stackrel{\text{def}}{=} \begin{cases} \mathbf{att}_j(\mathsf{frame}_{\mathcal{P}}@\mathtt{tr}) & \text{if } \mathtt{tr} \neq \epsilon \\ \mathbf{att}_0() & \text{if } \mathtt{tr} = \epsilon \end{cases}$$

Remark: we omit $\mathcal{P}$ when it is clear from context.

The restriction to traces in $\mathcal{T}_{io}$ simplifies the definition of $in_{\mathcal{P}}@tr$.

Macro Terms

$\mathsf{frame}_{\mathcal{P}}@\mathtt{tr}$ contains **all** the **information known** to an **adversary** against $\mathcal{P}$ after the execution of $\mathtt{tr}$.

More precisely, we can show that for all action-deterministic processes $\mathcal{P}$ and $\mathcal{Q}$, for all $\mathtt{tr} \in \mathcal{T}_{\mathsf{io}}$:

$$\mathcal{M} \models \mathsf{fold}(\mathcal{P}, \mathtt{tr}) \sim \mathsf{fold}(\mathcal{Q}, \mathtt{tr}) \ \text{ iff. } \ \mathcal{M} \models \mathsf{frame}_{\mathcal{P}}@\mathtt{tr} \sim \mathsf{frame}_{\mathcal{Q}}@\mathtt{tr}$$

for any $\mathcal{M}$ satisfying:

$$\pi_1\langle x\,,\, y\rangle \doteq x \sim \mathsf{true} \qquad\qquad \pi_2\langle x\,,\, y\rangle \doteq y \sim \mathsf{true}$$

Proof

$\Rightarrow$ apply FA to build $\mathsf{frame}_{\mathcal{R}}@\mathtt{tr}$ from $\mathsf{fold}(\mathcal{R}, \mathtt{tr})$ for $\mathcal{R} \in \{\mathcal{P}, \mathcal{Q}\}$
$\Leftarrow$ apply FA + DUP + the pair injectivity rules to compute all terms in $\mathsf{fold}(\mathcal{R}, \mathtt{tr})$ from $\mathsf{frame}_{\mathcal{R}}@\mathtt{tr}$ for $\mathcal{R} \in \{\mathcal{P}, \mathcal{Q}\}$

Hash-Lock: Accept

$$\begin{aligned} T(A,\mathtt{i}) &: \nu\, n_{T,\mathtt{i}}.\ \mathbf{in}(c^{\mathtt{T}}_{A,\mathtt{i}}, x).\ \mathbf{out}(c^{\mathtt{T}}_{A,\mathtt{i}}, \langle n_{T,\mathtt{i}}, H(\langle x, n_{T,\mathtt{i}}\rangle, k_A)\rangle) \\ R(\mathtt{j}) &: \nu\, n_{R,\mathtt{j}}.\ \mathbf{in}(c^{\mathtt{R_1}}_{\mathtt{j}}, \_).\ \mathbf{out}(c^{\mathtt{R_1}}_{\mathtt{j}}, n_{R,\mathtt{j}}).\ \mathbf{in}(c^{\mathtt{R_2}}_{\mathtt{j}}, y). \\ &\quad \text{if } \dot{\bigvee}_{A\in\mathcal{I}} \pi_2(y) \doteq H(\langle n_{R,\mathtt{j}}, \pi_1(y)\rangle, k_A) \\ &\quad \text{then } \mathbf{out}(c^{\mathtt{R_2}}_{\mathtt{j}}, \mathrm{ok}) \\ &\quad \text{else } \mathbf{out}(c^{\mathtt{R_2}}_{\mathtt{j}}, \mathrm{ko}) \end{aligned}$$

To be able to state some **authentication** property of Hash-Lock, we need an additional macro. For all $\mathtt{tr} \diamond c^{\mathtt{R_2}}_{\mathtt{j}} \in \mathcal{T}_{io}$, we let:

$$\mathrm{accept}^{A}@\mathtt{tr} \stackrel{\text{def}}{=} \pi_2(\mathrm{in}@\mathtt{tr}) \doteq H(\langle n_{R,\mathtt{j}}, \pi_1(\mathrm{in}@\mathtt{tr})\rangle, k_A)$$

We made sure that all names in the protocol are unique, so that they don't have to be renamed during the folding.

Authentication: Hash-Lock

The following formulas encode the fact that the **Hash-Lock** protocol provides **authentication**:

$$\forall \mathsf{A} \in \mathcal{I}.\ \forall \mathtt{tr} \in \mathcal{T}_{\mathsf{io}}.\ \forall \mathtt{tr}_1 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_1}, \mathtt{tr}_3 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_2} \text{ s.t. } \mathtt{tr}_1 < \mathtt{tr}_3 \leq \mathtt{tr},$$

$$\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3 \mathrel{\dot{\rightarrow}} \dot{\bigvee_{\substack{\mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}} \\ \mathtt{tr}_1 \leq \mathtt{tr}_2 \leq \mathtt{tr}_3}}} \begin{array}{l} \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \mathrel{\dot{\wedge}} \\ \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3 \end{array} \sim \mathsf{true}$$

This kind of one-sided formulas are called **reachability formulas**. Proving the validity of such formulas requires **additional rules**, to allow for **propositional reasoning**.

Authentication Protocols

Reachability Proof System

Reachability Judgements

We define a **judgments** dedicated to **reachability correspondance properties**.

Definition

A **reachability judgement** $\Gamma \vdash t$ comprises a sequence of terms $\Gamma = t_1 \dot{\rightarrow} \cdots \dot{\rightarrow} t_n$ and a (boolean) term t.

$\Gamma \vdash t$ is **valid** if and only if the following formula is valid:

$$(t_1 \dot{\rightarrow} \cdots \dot{\rightarrow} t_n \dot{\rightarrow} t) \sim \mathrm{true}$$

Boolean Connectives in Reachability Judgements

Careful not to confuse the boolean connectives at the **reachability** and **equivalence** levels!

Exercise

Determine which directions are correct.

$$t_\phi \mathbin{\dot\wedge} t_\psi \sim \mathsf{true} \quad \overset{?}{\Leftrightarrow} \quad t_\phi \sim \mathsf{true} \wedge t_\psi \sim \mathsf{true}$$

$$t_\phi \mathbin{\dot\vee} t_\psi \sim \mathsf{true} \quad \overset{?}{\Leftrightarrow} \quad t_\phi \sim \mathsf{true} \vee t_\psi \sim \mathsf{true}$$

$$t_\phi \mathbin{\dot\rightarrow} t_\psi \sim \mathsf{true} \quad \overset{?}{\Leftrightarrow} \quad t_\phi \sim \mathsf{true} \rightarrow t_\psi \sim \mathsf{true}$$

Boolean Connectives in Reachability Judgements

Careful not to confuse the boolean connectives at the **reachability** and **equivalence** levels!

Exercise

Determine which directions are correct.

$$t_\phi \mathbin{\dot\wedge} t_\psi \sim \mathsf{true} \quad \Leftrightarrow \quad t_\phi \sim \mathsf{true} \wedge t_\psi \sim \mathsf{true}$$

$$t_\phi \mathbin{\dot\vee} t_\psi \sim \mathsf{true} \quad \Leftarrow \quad t_\phi \sim \mathsf{true} \vee t_\psi \sim \mathsf{true}$$

$$t_\phi \mathbin{\dot\rightarrow} t_\psi \sim \mathsf{true} \quad \Rightarrow \quad t_\phi \sim \mathsf{true} \rightarrow t_\psi \sim \mathsf{true}$$

The second relation works both ways when t_ϕ or t_ψ is a **constant** formula.

Reachability Proof System

Our **reachability judgements** can be trivially equipped with a **sequent calculus**.

$$\frac{}{\Gamma, t_\phi \vdash t_\phi} \qquad \frac{\Gamma \vdash t_\psi \qquad \Gamma, t_\psi \vdash t_\phi}{\Gamma \vdash t_\phi}$$

$$\frac{\Gamma \vdash t_\psi \qquad \Gamma \vdash t_\phi}{\Gamma \vdash t_\psi \mathbin{\dot\wedge} t_\phi} \qquad \frac{\Gamma, t_\psi, t_\phi \vdash t_\theta}{\Gamma, t_\psi \mathbin{\dot\wedge} t_\phi \vdash t_\theta}$$

$$\frac{\Gamma \vdash t_\phi}{\Gamma \vdash t_\psi \mathbin{\dot\vee} t_\phi} \qquad \frac{\Gamma \vdash t_\psi}{\Gamma \vdash t_\psi \mathbin{\dot\vee} t_\phi} \qquad \frac{\Gamma, t_\psi \vdash t_\theta \qquad \Gamma, t_\phi \vdash t_\theta}{\Gamma, t_\psi \mathbin{\dot\vee} t_\phi \vdash t_\theta}$$

$$\frac{\Gamma \vdash t_\psi \qquad \Gamma, t_\phi \vdash t_\theta}{\Gamma, t_\psi \mathbin{\dot\to} t_\phi \vdash t_\theta} \qquad \frac{\Gamma, t_\psi \vdash t_\phi}{\Gamma \vdash t_\psi \mathbin{\dot\to} t_\phi}$$

Reachability Proof System (cont.)

$$\frac{\Gamma, t_\phi \vdash \bot}{\Gamma \vdash \neg t_\phi} \qquad \frac{}{\Gamma, \bot \vdash t_\phi}$$

$$\frac{\Gamma_1, t_\phi, t_\psi, \Gamma_2 \vdash t_\theta}{\Gamma_1, t_\psi, t_\phi, \Gamma_2 \vdash t_\theta} \qquad \frac{\Gamma, t_\psi, t_\psi \vdash t_\phi}{\Gamma, t_\psi \vdash t_\phi}$$

Reachability Proof System: Soundness

The reachability proof system is **sound**.

Proof

First, remark that any Γ and t_θ,

$$\Gamma \vdash t_\theta \text{ is valid iff. } \Pr_\rho \left([\![(\dot{\wedge}\Gamma) \mathrel{\dot{\wedge}} \dot{\neg} t_\phi]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho) \right) \text{ is negligible.} \qquad (\dagger)$$

- Left-to-right:

$$\begin{aligned} & \Gamma \vdash t_\theta \text{ valid} \\ \Rightarrow\ & \forall A \in \mathcal{D}.\ \Pr_\rho \left(\mathcal{A}(1^\eta, [\![(\dot{\wedge}\Gamma) \mathrel{\dot{\wedge}} \dot{\neg} t_\phi]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho), \rho_a \right) \in \mathsf{negl}(\eta) \\ \Rightarrow\ & \Pr_\rho \left([\![(\dot{\wedge}\Gamma) \mathrel{\dot{\wedge}} \dot{\neg} t_\phi]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho) \right) \in \mathsf{negl}(\eta) \\ & \qquad\qquad (\text{taking } \mathcal{A}(1^\eta, w, \rho_a) = w) \end{aligned}$$

- Right-to-left is straightforward.

Reachability Proof System: Soundness

We only prove only rule, say

$$\frac{\Gamma, t_\psi \vdash t_\theta \qquad \Gamma, t_\phi \vdash t_\theta}{\Gamma, t_\psi \mathbin{\dot\vee} t_\phi \vdash t_\theta}$$

By the previous remark (†), since $(\Gamma, t_\psi \vdash t_\theta)$ and $(\Gamma, t_\phi \vdash t_\theta)$ are valid

- $\mathsf{Pr}_\rho\left(\llbracket(\dot\bigwedge\Gamma) \mathbin{\dot\wedge} t_\psi \mathbin{\dot\wedge} \dot\neg t_\theta\rrbracket^\sigma_{\mathcal{M}}(1^\eta, \rho)\right)$ is negligible.
- $\mathsf{Pr}_\rho\left(\llbracket(\dot\bigwedge\Gamma) \mathbin{\dot\wedge} t_\phi \mathbin{\dot\wedge} \dot\neg t_\theta\rrbracket^\sigma_{\mathcal{M}}(1^\eta, \rho)\right)$ is negligible.

Since the union of two negligible (η-indexed families of) events is a negligible (η-indexed families of) events,

$$\begin{aligned} & \mathsf{Pr}_\rho\left(\llbracket((\dot\bigwedge\Gamma) \mathbin{\dot\wedge} t_\psi \mathbin{\dot\wedge} \dot\neg t_\theta) \mathbin{\dot\vee} ((\dot\bigwedge\Gamma) \mathbin{\dot\wedge} t_\phi \mathbin{\dot\wedge} \dot\neg t_\theta)\rrbracket^\sigma_{\mathcal{M}}(1^\eta, \rho)\right) \text{ is negligible} \\ \Leftrightarrow\; & \mathsf{Pr}_\rho\left(\llbracket(\dot\bigwedge\Gamma) \mathbin{\dot\wedge} (t_\psi \mathbin{\dot\vee} t_\phi) \mathbin{\dot\wedge} \dot\neg t_\theta\rrbracket^\sigma_{\mathcal{M}}(1^\eta, \rho)\right) \text{ is negligible} \end{aligned}$$

Hence using (†) again, $\Gamma, t_\psi \mathbin{\dot\vee} t_\phi \vdash t_\theta$ is valid.

Authentication Protocols

Cryptographic Rule: Collision Resistance

Cryptographic Hash

A **keyed cryptographic hash** $\mathsf{H}(\_,\_)$ is **computationally collision resistant** if no PPTM adversary can built collisions, even when it has access to a hashing **oracle**.

More precisely, a hash is *collision resistant under hidden key attacks* (CR-HK) iff for every PPTM $\mathcal{A}$, the following quantity:

$$\mathsf{Pr}_{\mathsf{k}}\left(\mathcal{A}^{\mathcal{O}_{\mathsf{H}(\cdot,\mathsf{k})}}(1^\eta) = \langle m_1\,,\, m_2\rangle, m_1 \neq m_2 \text{ and } \mathsf{H}(m_1,\mathsf{k}) = \mathsf{H}(m_2,\mathsf{k})\right)$$

is negligible, where k is drawn uniformly in $\{0,1\}^\eta$.

CR Rule

Collision Resistance

If H is a CR-HK function, then the *ground* rule:

$$\frac{}{\mathsf{H}(m_1, \mathsf{k}) \doteq \mathsf{H}(m_2, \mathsf{k}) \dot{\to} m_1 \doteq m_2 \sim \mathsf{true}}\ \mathrm{CR}$$

is sound, when k appears only in H key positions in m_1, m_2.

CR Rule: Exercise

Exercise

Let H be CR-HK. Show that the following rule is **not** sound:

$$\frac{}{\dot{\neg}(\mathsf{H}(m_1, \mathsf{k}) \doteq \mathsf{H}(m_2, \mathsf{k})) \sim \mathsf{true}} \text{ CR}$$

when k appears only in H key positions in m_1, m_2 and $m_1 \not\equiv m_2$.

Authentication Protocols

Cryptographic Rule: Message Authentication Code

Message Authentication Code

A **message authentication code** is a symmetric cryptographic schema which:

- create **message authentication codes** using mac_(_)
- **verifies** mac using verify_(_, _)

It must satisfies the functional equality:

$$\text{verify}_{\text{k}}(\text{mac}_{\text{k}}(m), m) = \text{true}$$

MAC Security

A MAC must be **computationally unforgeable**, even when the adversary has access to a mac and verify **oracles**.

A MAC is *unforgeable against chosen-message attacks* (EUF-CMA) iff for every PPTM $\mathcal{A}$, the following quantity:

$$\mathsf{Pr}_{\mathsf{k}}\left(\begin{array}{r}\mathcal{A}^{\mathcal{O}_{\mathsf{mac}_{\mathsf{k}}(\cdot)},\mathcal{O}_{\mathsf{verify}_{\mathsf{k}}(\cdot,\cdot)}}(1^\eta) = \langle m\,,\,\sigma\rangle,\ m \text{ not queried to } \mathcal{O}_{\mathsf{mac}_{\mathsf{k}}(\cdot)}\\ \text{and } \mathsf{verify}_{\mathsf{k}}(\sigma, m) = 1\end{array}\right)$$

is negligible, where k is drawn uniformly in $\{0,1\}^\eta$.

EUF-MAC Rule

Take two messages s, m and a key $\mathsf{k} \in \mathcal{N}$ such that

- s and m are ground.
- $\mathsf{k} \in \mathcal{N}$ appears only in mac or verify key positions in s, m.

Key Idea

To build a rule for EUF-CMA, we proceed as follow:

- Compute $[\![s, m]\!]$ bottum-up, calling $\mathcal{O}_{\mathsf{mac}_\mathsf{k}(\cdot)}$ and $\mathcal{O}_{\mathsf{verify}_\mathsf{k}(\cdot,\cdot)}$ if necessary.
- Log all sub-terms $\mathbb{S}_{\mathsf{mac}}(s, m)$ sent to $\mathcal{O}_{\mathsf{mac}_\mathsf{k}(\cdot)}$.

$\Rightarrow$ If $\mathsf{verify}_\mathsf{k}(s, m)$ then $m = u$ for some $u \in \mathbb{S}_{\mathsf{mac}}(s, m)$.

*$\mathbb{S}_{mac}(s, m)$ are the **calls** to $\mathcal{O}_{mac_k(\cdot)}$ needed to compute s, m.*

EUF-MAC Rule

$\mathbb{S}_{\mathsf{mac}}(\cdot)$ defined by induction on ground terms:

$$\begin{aligned}
\mathbb{S}_{\mathsf{mac}}(\mathsf{n}) &\stackrel{\text{def}}{=} \emptyset \\
\mathbb{S}_{\mathsf{mac}}(\mathsf{verify}_{\mathsf{k}}(u_1, u_2)) &\stackrel{\text{def}}{=} \mathbb{S}_{\mathsf{mac}}(u_1) \cup \mathbb{S}_{\mathsf{mac}}(u_2) \\
\mathbb{S}_{\mathsf{mac}}(\mathsf{mac}_{\mathsf{k}}(u)) &\stackrel{\text{def}}{=} \{u\} \cup \mathbb{S}_{\mathsf{mac}}(u) \\
\mathbb{S}_{\mathsf{mac}}(f(u_1, \dots, u_n)) &\stackrel{\text{def}}{=} \bigcup_{1 \leq i \leq n} \mathbb{S}_{\mathsf{mac}}(u_i) \qquad \text{(for other cases)}
\end{aligned}$$

EUF-MAC Rule

Message Authentication Code Unforgeability

If mac is an EUF-CMA function, then the *ground* rule:

$$\frac{}{\mathsf{verify}_{\mathsf{k}}(s, m) \dot{\rightarrow} \dot{\bigvee}_{u \in \mathcal{S}} m \doteq u \sim \mathsf{true}} \text{ EUF-MAC}$$

is sound, when:

- $\mathcal{S} = \{u \mid \mathsf{mac}_{\mathsf{k}}(u) \in \mathbb{S}_{\mathsf{mac}}(s, m)\}$;
- $\mathsf{k} \in \mathcal{N}$ appears only in mac or verify key positions in s, m.

Example

If t_1 t_2 and t_3 are terms which do not contain k, then:

$$\Phi \equiv \mathsf{mac}_{\mathsf{k}}(t_1), \mathsf{mac}_{\mathsf{k}}(t_2), \mathsf{mac}_{\mathsf{k}_0}(t_3)$$

$$\models \mathsf{verify}_{\mathsf{k}}(g(\Phi), \mathsf{n}) \;\dot{\rightarrow}\; \left(\mathsf{n} \doteq t_1 \,\dot{\vee}\, \mathsf{n} \doteq t_2\right) \sim \mathsf{true}$$

EUF-MAC Rule: Exercise

Exercise

Assume mac is EUF-CMA. Show that the following rule is sound:

$$\frac{}{\mathsf{verify}_{\mathsf{k}}(\text{if } b \text{ then } s_0 \text{ else } s_1, m) \dot{\rightarrow} \dot{\bigvee}_{u \in \mathcal{S}_1 \cup \mathcal{S}_2} m \doteq u \sim \mathsf{true}}$$

when b, s_0, s_1, m are *ground* terms, and:

- $\mathcal{S}_i = \{u \mid \mathsf{mac}_{\mathsf{k}}(u) \in \mathbb{S}_{\mathsf{mac}}(s_i, m)\}$, for $i \in \{0, 1\}$;
- k appears only in mac or verify key positions in s_0, s_1, m.

Remark: we do not make *any* assumption on b, except that it is ground. E.g., we can have $b \equiv (\mathbf{att}(\mathsf{k}) \doteq \mathsf{mac}_{\mathsf{k}}(0))$.

Authentication Protocols

Authentication of the Hash-Lock Protocol

Authentication: Hash-Lock

Theorem

Assuming that the hash function is EUF-CMA[6], the Hash-Lock protocol provides **authentication**, i.e. for any identity $a \in \mathcal{I}$, for any $\mathtt{tr} \in \mathcal{T}_{io}$, $\mathtt{tr}_1 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_1}$ and $\mathtt{tr}_3 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_2}$ s.t.:

$$\mathtt{tr}_1 < \mathtt{tr}_3 \leq \mathtt{tr}$$

the following formula is valid:

$$\mathsf{accept}^{\mathsf{A}} @\mathtt{tr}_3 \mathrel{\dot{\rightarrow}} \dot{\bigvee_{\substack{\mathtt{tr}_2 \diamond \mathtt{c}_{\mathtt{A},\mathtt{i}}^{\mathtt{T}} \\ \mathtt{tr}_1 \leq \mathtt{tr}_2 \leq \mathtt{tr}_3}}} \begin{array}{l} \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \mathrel{\dot{\wedge}} \\ \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3 \end{array} \sim \mathsf{true}$$

[6] Taking $\mathsf{verify}_{\mathsf{k}}(s, m) \stackrel{\mathsf{def}}{=} s \doteq \mathsf{H}(m, \mathsf{k})$.

Authentication: Hash-Lock

Proof. Let $a \in \mathcal{I}$, and let $\mathtt{tr} \in \mathcal{T}_{io}$, $\mathtt{tr}_1 \diamond c_j^{\mathtt{R_1}}$ and $\mathtt{tr}_3 \diamond c_j^{\mathtt{R_2}}$ be s.t.:

$$\mathtt{tr}_1 < \mathtt{tr}_3 \leq \mathtt{tr}$$

We let:

$$t_{conc} \stackrel{\text{def}}{=} \dot{\bigvee_{\substack{\mathtt{tr_2} \diamond c^{\mathtt{T}}_{\mathbf{A},\mathtt{i}} \\ \mathtt{tr_1} \leq \mathtt{tr_2} \leq \mathtt{tr_3}}}} \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \mathbin{\dot\wedge} \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3$$

We must prove that the following reachability judgement is valid:

$$\mathsf{accept}^{A}@\mathtt{tr}_3 \vdash t_{conc}$$

i.e. that:

$$\pi_2(\mathsf{in}@\mathtt{tr}_3) \doteq \mathsf{H}(\langle \mathsf{n}_{R,j}\,,\, \pi_1(\mathsf{in}@\mathtt{tr}_3)\rangle, \mathsf{k}_A) \vdash t_{conc}$$

Authentication: Hash-Lock

We use the EUF-MAC rule on the equality:

$$\pi_2(\mathsf{in}@\mathtt{tr}_3) \doteq \mathsf{H}(\langle \mathsf{n}_{\mathsf{R},\mathsf{j}}\,,\, \pi_1(\mathsf{in}@\mathtt{tr}_3)\rangle, \mathsf{k}_\mathsf{A}) \qquad (\dagger)$$

The terms above are ground, and the key k_A is correctly used in them. Moreover, the set of *honest* hashes using key k_A appearing in (†), excluding the top-level hash, is:

$$\begin{aligned} & \mathbb{S}_{\mathsf{mac}}(\pi_2(\mathsf{in}@\mathtt{tr}_3), \langle \mathsf{n}_{\mathsf{R},\mathsf{j}}\,,\, \pi_1(\mathsf{in}@\mathtt{tr}_3)\rangle) \\ = \; & \mathbb{S}_{\mathsf{mac}}(\mathsf{in}@\mathtt{tr}_3) \\ = \; & \left\{ \mathsf{H}(\langle \mathsf{in}@\mathtt{tr}_2\,,\, \mathsf{n}_{\mathsf{T},\mathtt{i}}\rangle, \mathsf{k}_\mathsf{A}) \mid \mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}} < \mathtt{tr}_3 \right\} \end{aligned}$$

The hashes in the reader's outputs can be seen as verify checks, and can therefore be ignored.

Authentication: Hash-Lock

Hence using EUF-MAC plus some basic reasoning, we have:

$$\dfrac{\dfrac{\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3, \begin{matrix}\langle \mathsf{in}@\mathtt{tr}_2\,,\, \mathsf{n}_{\mathsf{T},\mathtt{i}}\rangle \doteq \\ \langle \mathsf{n}_{\mathsf{R},\mathtt{j}}\,,\, \pi_1(\mathsf{in}@\mathtt{tr}_3)\rangle\end{matrix} \vdash t_{\mathsf{conc}} \qquad \text{for every } \mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}} < \mathtt{tr}_3}{\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3, \dot{\bigvee}_{\mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}} < \mathtt{tr}_3} \begin{matrix}\langle \mathsf{in}@\mathtt{tr}_2\,,\, \mathsf{n}_{\mathsf{T},\mathtt{i}}\rangle \doteq \\ \langle \mathsf{n}_{\mathsf{R},\mathtt{j}}\,,\, \pi_1(\mathsf{in}@\mathtt{tr}_3)\rangle\end{matrix} \vdash t_{\mathsf{conc}}}}{\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3 \vdash t_{\mathsf{conc}}}$$

Authentication: Hash-Lock

Assuming that the pair and projections satisfy:

$$\frac{}{(\pi_1\langle x\,,\,y\rangle \doteq x) \sim \mathsf{true}} \qquad \frac{}{(\pi_2\langle x\,,\,y\rangle \doteq y) \sim \mathsf{true}}$$

We only have to show that for every $\mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A,i}} < \mathtt{tr}_3$:

$$\Gamma \vdash t_{\mathsf{conc}}$$

is valid, where:

$$\Gamma \stackrel{\mathsf{def}}{=} \mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3,\ \mathsf{in}@\mathtt{tr}_2 \doteq \mathsf{n}_{\mathsf{R,j}},\ \mathsf{n}_{\mathsf{T,i}} \doteq \pi_1(\mathsf{in}@\mathtt{tr}_3)$$

Authentication: Hash-Lock

Since $\mathtt{tr}_1 \diamond \mathsf{c}_{\mathtt{j}}^{\mathtt{R}_1} < \mathtt{tr}_3$ we know that:

$$\mathtt{out}@\mathtt{tr}_1 \stackrel{\text{def}}{=} \mathsf{n}_{\mathsf{R},\mathtt{j}}$$

Moreover:

$$\mathtt{out}@\mathtt{tr}_2 \stackrel{\text{def}}{=} \langle \mathsf{n}_{\mathsf{T},\mathtt{i}} \,, \mathsf{H}(\langle \mathsf{in}@\mathtt{tr}_2 \,, \mathsf{n}_{\mathsf{T},\mathtt{i}} \rangle, \mathsf{k}_\mathsf{A}) \rangle$$

Hence:

$$\Gamma \vdash \pi_1(\mathtt{out}@\mathtt{tr}_2) \doteq \pi_1(\mathsf{in}@\mathtt{tr}_3) \qquad (\diamond)$$

Similarly:

$$\begin{aligned}\Gamma \vdash \pi_2(\mathtt{out}@\mathtt{tr}_2) &\doteq \mathsf{H}(\langle \mathsf{in}@\mathtt{tr}_2 \,, \mathsf{n}_{\mathsf{T},\mathtt{i}} \rangle, \mathsf{k}_\mathsf{A}) \\ &\doteq \mathsf{H}(\langle \mathsf{n}_{\mathsf{R},\mathtt{j}} \,, \pi_1(\mathsf{in}@\mathtt{tr}_3) \rangle, \mathsf{k}_\mathsf{A}) \\ &\doteq \pi_2(\mathsf{in}@\mathtt{tr}_3)\end{aligned}$$

Consequently:

$$\Gamma \vdash \pi_2(\mathtt{out}@\mathtt{tr}_2) \doteq \pi_2(\mathsf{in}@\mathtt{tr}_3) \qquad (\star)$$

Assuming that the pair and projections satisfy the property:

$$\frac{}{\pi_1\ x \doteq \pi_1\ y \mathrel{\dot{\to}} \pi_2\ x \doteq \pi_2\ y \mathrel{\dot{\to}} x \doteq y}$$

We deduce from $(\star)$ and $(\diamond)$ that:

$$\Gamma \vdash \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3$$

Putting everything together, we get:

$$\Gamma \vdash \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \mathbin{\dot{\wedge}} \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3 \qquad (\ddagger)$$

Authentication: Hash-Lock

Recall that:

$$t_{\text{conc}} \stackrel{\text{def}}{=} \dot{\bigvee}_{\substack{\mathtt{tr_2} \diamond \mathtt{c}^{\mathtt{T}}_{\mathbf{A},\mathtt{i}} \\ \mathtt{tr_1} \leq \mathtt{tr_2} \leq \mathtt{tr_3}}} \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \mathbin{\dot{\wedge}} \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3$$

and we must show that $\Gamma \vdash t_{\text{conc}}$. Hence, using (‡), it only remains to prove that whenever $\mathtt{tr}_2 < \mathtt{tr}_1$, we have:

$$\Gamma,\ \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2,\ \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3 \vdash \bot$$

This follows from the independence rule:

$$\frac{}{(t \doteq \mathsf{n}) = \mathsf{false}}\ {=}\text{-IND} \qquad \text{when } t \text{ is ground and } \mathsf{n} \not\in \mathsf{st}(t)$$

using the fact that:

$$\mathsf{out}@\mathtt{tr}_1 \stackrel{\text{def}}{=} \mathsf{n}_{\mathsf{R},\mathsf{j}}$$

and that if $\mathtt{tr}_2 < \mathtt{tr}_1$ then $\mathsf{n}_{\mathsf{R},\mathsf{j}} \not\in \mathsf{st}(\mathsf{in}@\mathtt{tr}_2)$.

Authentication Protocols

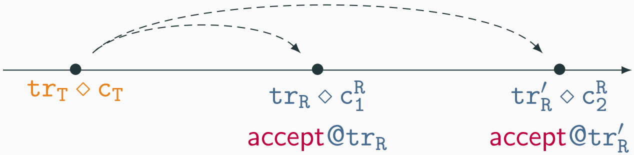
Beyond Authentication

Beyond Authentication

Authentication, which states that we must have:

$\forall \mathtt{tr}_\mathtt{R} \diamond \mathtt{c}_\mathtt{R}.\ \exists \mathtt{tr}_\mathtt{T} \diamond \mathtt{c}_\mathtt{T}.$

does not exclude the scenario:



Replay Attack

This is a **replay attack**: the **same message** (or partial transcript), when replayed, is **accepted again** by the server.

This can yield real-word **attacks**. E.g. an adversary can open a door at will once it eavesdropped one honest interaction.

Example

The following protocol, called Basic Hash, suffer from such attacks:

$$\begin{aligned}
\mathsf{T}(\mathsf{A}, \mathtt{i}) &: \nu\, \mathsf{n}_{\mathsf{T},\mathtt{i}}.\ \mathbf{out}(\mathsf{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}}, \langle \mathsf{n}_{\mathsf{T},\mathtt{i}}\,,\ \mathsf{H}(\mathsf{n}_{\mathsf{T},\mathtt{i}}, \mathsf{k}_{\mathsf{A}})\rangle) \\
\mathsf{R}(\mathtt{j}) &: \mathbf{in}(\mathsf{c}^{\mathtt{R}_2}_{\mathtt{j}}, \mathsf{y}).\ \text{if } \dot{\bigvee}_{\mathsf{A} \in \mathcal{I}} \pi_2(\mathsf{y}) \doteq \mathsf{H}(\pi_1(\mathsf{y}), \mathsf{k}_{\mathsf{A}}) \\
&\qquad\qquad\qquad \text{then } \mathbf{out}(\mathsf{c}^{\mathtt{R}_2}_{\mathtt{j}}, \mathsf{ok}) \\
&\qquad\qquad\qquad \text{else } \mathbf{out}(\mathsf{c}^{\mathtt{R}_2}_{\mathtt{j}}, \mathsf{ko})
\end{aligned}$$

Injective Authentication

The **authentication** property is too *weak* for many real-world application.

To prevent replay attacks, we require that the protocol provides a **stronger** property, **injective authentication**.

Injective Authentication: Hash-Lock

The following formulas encode the fact that the **Hash-Lock** protocol provides **injective authentication**:

$\forall \mathsf{A} \in \mathcal{I}.\ \forall \mathtt{tr} \in \mathcal{T}_{\mathsf{io}}.\ \forall \mathtt{tr}_1 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_1}, \mathtt{tr}_3 \diamond \mathtt{c}_{\mathtt{j}}^{\mathtt{R}_2}$ s.t. $\mathtt{tr}_1 < \mathtt{tr}_3 \leq \mathtt{tr}$

$$\mathsf{accept}^{\mathsf{A}}@\mathtt{tr}_3 \ \dot{\rightarrow} \ \dot{\bigvee_{\substack{\mathtt{tr}_2 \diamond \mathtt{c}^{\mathtt{T}}_{\mathtt{A},\mathtt{i}} \\ \mathtt{tr}_1 \leq \mathtt{tr}_2 \leq \mathtt{tr}_3}}} \begin{array}{l} \mathsf{out}@\mathtt{tr}_1 \doteq \mathsf{in}@\mathtt{tr}_2 \ \dot{\wedge} \\ \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}_3 \end{array}$$

$$\dot{\wedge} \ \dot{\bigwedge_{\substack{\mathtt{tr}'_1 \diamond \mathtt{c}_{\mathtt{k}}^{\mathtt{R}_1},\ \mathtt{tr}'_3 \diamond \mathtt{c}_{\mathtt{k}}^{\mathtt{R}_2} \\ \mathtt{tr}'_1 < \mathtt{tr}'_3 \leq \mathtt{tr}}}} \left(\begin{array}{l} \mathsf{accept}^{\mathsf{A}}@\mathtt{tr}'_3 \ \dot{\wedge} \\ \mathsf{out}@\mathtt{tr}_2 \doteq \mathsf{in}@\mathtt{tr}'_3 \end{array} \dot{\rightarrow}\ j = k \right)$$

References i

[1] D. Baelde, S. Delaune, and L. Hirschi.
Partial order reduction for security protocols.
In *CONCUR*, volume 42 of *LIPIcs*, pages 497–510. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[2] G. Bana and H. Comon-Lundh.
A computationally complete symbolic attacker for equivalence properties.
In *CCS*, pages 609–620. ACM, 2014.