MPRI 2.30: Proofs of Security Protocols
The CCSA Approach to Computational Security

Adrien Koutsos
2021/2022
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
To illustrate what terms we need to consider, we consider a simple authentication protocol:

**The Private Authentication (PA) Protocol, v1**

1: $A \rightarrow B: \nu n_A$.  
   out($c_A, \{\langle pk_A, n_A \rangle \}_{pk_B}$)

2: $B \rightarrow A: \nu n_B. \text{in}(c_A, x). \text{out}(c_B, \{\langle \pi_2(d), n_B \rangle \}_{pk_A})$

where $pk_A \equiv pk(k_A)$, $pk_B \equiv pk(k_B)$ and $d \equiv \text{dec}(x, sk_A)$.

*Notation: we use $\equiv$ to denote syntactic equality of terms.*
We use **terms** to model *protocol messages*, built upon:

- **Names** $\mathcal{N}$, e.g. $n_A, n_B$, for random samplings.
- **Function symbols** $\mathcal{F}$, e.g.:

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

  $\text{if}_\_\text{then}_\_\text{else}_\_, \_ = \_, \_ \wedge \_, \_ \lor \_, \_ \rightarrow \_\$

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$pk(k_A)$</td>
</tr>
</tbody>
</table>
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 $\phi$ 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
The PA Protocol, v1

1: A → B : ν n_A. out(c_A, {⟨pk_A , n_A⟩}_{pk_B})
2: B → A : ν n_B. in(c_A, x). out(c_B, {⟨π_2(dec(att(t_1), sk_A)), n_B⟩}_{pk_A})

How do we represent the adversary’s inputs?

- We use adversarial functions symbols att ∈ G, which takes as input the current knowledge of the adversary.
- Intuitively, att can be any probabilistic PTIME computation.

Example: Terms for PA, v1

\[ t_1 \equiv \{⟨pk_A , n_A⟩\}_{pk_B} \]
\[ t_2 \equiv \{⟨π_2(dec(att(t_1)), sk_A)⟩, n_B⟩\}_{pk_A} \]
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:

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

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

$j$ allows to have different values for consecutive inputs.
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
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

1. $A \rightarrow B : \nu n_A$. \hspace{2cm} $\text{out}(c_A, \langle \langle \text{pk}_A, n_A \rangle \rangle_{\text{pk}_B})$

2. $B \rightarrow A : \nu n_B$. $\text{in}(c_A, x)$. If $\pi_1(d) \doteq \text{pk}_A$
   
   then $\text{out}(c_B, \langle \langle \pi_2(d), n_B \rangle \rangle_{\text{pk}_A})$
   
   else $\text{out}(c_B, \{0\}_{\text{pk}_A})$

where $d \equiv \text{dec}(x, \text{sk}_A)$.

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

⇒ branching is pushed (or folded) in the outputted terms, using the if_then_else_ function symbol.
### Example: Terms for PA, v2

\[ t_1 \equiv \{ \langle pk_A, n_A \rangle \}_{pk_B} \]
\[ t_2 \equiv \begin{cases} \{ \langle \pi_2(d_1), n_B \rangle \}_{pk_A} & \text{if } \pi_1(d_1) = pk_A \\ \{0\}_ {pk_A} & \text{else} \end{cases} \]

where \( d_1 \equiv \text{dec(} \text{att}(t_1), sk_A \text{)} \).
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.
First, we require that processes are deterministic.

Indeed, consider a simple process:

\[ P = \text{out}(c, t_0) \mid \text{out}(c, t_1) \]

- 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.
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 $A \xrightarrow{} A'$, for any observable action $\alpha$, if $A' \xrightarrow{\alpha} A_1$ and $A' \xrightarrow{\alpha} A_2$ then $A_1 = A_2$, 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.

\[
\text{out}(c, t_1) | \text{in}(c, x). \text{out}(c, t_2)
\]

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

\[
\text{out}(c, t_1) | \text{if } b\text{ then } \text{out}(c, t_2) \text{ else } \text{out}(c_0, t_3)
\]
Folding

Folding Algorithm
Folding configuration

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

- \(\Phi\) is a sequence of terms (in \(\mathcal{T}(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})\)).
- \(\sigma\) is a finite sequence of mappings \((x \mapsto t)\) 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.
In a **folding configuration** \( (\Phi; \sigma; j; \Pi_1, \ldots, \Pi_l) \):

- **\( \Phi \)** is the **frame**, i.e. the sequence of terms outputted since the execution started.
- **\( \sigma \)** **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:

\[(\Phi; \sigma; j; (\text{if } b \text{ then } P_1 \text{ else } P_2, b'), \Pi_1, \ldots, \Pi_l) \rightleftharpoons (\Phi; \sigma; j; (P_1, b' \land b), (P_2, b' \land \neg b), \Pi_1, \ldots, \Pi_l)\]

Rule for new:

\[(\Phi; \sigma; j; (\nu n, P, b), \Pi_1, \ldots, \Pi_l) \rightleftharpoons (\Phi; \sigma; j; (P[n \mapsto n_f], b), \Pi_1, \ldots, \Pi_l)\]

if \(n_f\) does not appear in the lhs configuration

\[\rightleftharpoons\text{-irreducibility}\]

A folding configuration \(K\) is \(\rightleftharpoons\text{-irreducible}\) if for any \(K'\), we have \(K \not\rightleftharpoons K'\).
Rule for inputs:

\[(\Phi; \sigma; j; (\text{in}(c, x). P_1, b_1), \ldots, (\text{in}(c, x). P_n, b_n), \Pi_1, \ldots, \Pi_l)\]

\[\xrightarrow{\text{in}(c)} (\Phi; \sigma[x \mapsto \text{att}_j(\Phi)]; j + 1; (P_1, b_1), \ldots, (P_n, b_n)\Pi_1, \ldots, \Pi_l)\]

if \(x \not\in \text{dom}(\sigma)\), the lhs folding configuration is \(\leftrightarrow\)-irreducible and if for every \(i\), \(\Pi_1\) 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 \(\Phi\) (in the rhs) by:

\[\Phi, \bigvee_{1 \leq i \leq n} b_i\]
Rule for outputs:

\[(\Phi; \sigma; j; (\text{out}(c, t_1).P_1, b_1), \ldots, (\text{out}(c, t_n).P_n, b_n), \Pi_1, \ldots, \Pi_l)\]

\[\text{out}(c) \mapsto (\Phi, t\sigma; \sigma; j; (P_1, b_1), \ldots, (P_n, b_n), \Pi_1, \ldots, \Pi_l)\]

if the lhs folding configuration is \(\leftrightarrow\)-irreducible and if for every \(i\), \(\Pi_i\) does not start by an output on \(c\) and:

\[t \equiv \text{if } b_1 \text{ then } t_1 \text{ else } \ldots \text{if } b_n \text{ then } t_n \text{ else error}\]

💡 The input and output rules make sense because we restrict ourselves to action-deterministic processes.
A folding observable action $a$ is either $\text{in}(c)$ or $\text{out}(c)$.

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

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

then $\Phi$ is the folding of $P$ over $\text{tr}$, denoted $\text{fold}(P, \text{tr})$. 
Exercise

What are all the possible foldings of the following protocols?

\[
\text{in}(c, x) \cdot \text{out}(c, t) \quad \text{out}(c, t_1) \mid \text{in}(c_0, x) \cdot \text{out}(c_0, t_2)
\]

\[
\text{if } b \text{ then out}(c, t_1) \text{ else out}(c, t_2)
\]

\[
\text{if } b \text{ then out}(c_1, t_1) \text{ else out}(c_2, t_2)
\]
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:

$$\nu n_0, \nu n_1, out(c, \langle n_0 , \langle 00 , n_1 \rangle \rangle)$$

yields $\langle n_0 , \langle 00 , n_1 \rangle \rangle$

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 $\eta$ and $\eta + 1$, which are always 0.
Semantics of Terms

We to 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).
The **interpretation** \([t]_M^\sigma\) 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 \([t]_M^\sigma \in \mathcal{D}\), 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 \textit{composition}.

For \textbf{function symbols} in $f \in \mathcal{F}$, we simply apply $\llbracket f \rrbracket_M$:

$$\llbracket f(t_1, \ldots, t_n) \rrbracket^\sigma_{1^n, \rho} \overset{\text{def}}{=} \llbracket f \rrbracket_M(\llbracket t_1 \rrbracket^\sigma_{1^n, \rho}, \ldots, \llbracket t_n \rrbracket^\sigma_{1^n, \rho})$$

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

$$\llbracket g(t_1, \ldots, t_n) \rrbracket^\sigma_{1^n, \rho} \overset{\text{def}}{=} \llbracket g \rrbracket_M(\llbracket t_1 \rrbracket^\sigma_{1^n, \rho}, \ldots, \llbracket t_n \rrbracket^\sigma_{1^n, \rho}, \rho_a)$$

\textit{Remark:} $\llbracket f \rrbracket_M$ and $\llbracket g \rrbracket_M$ are \textit{deterministic} (all randomness must come explicitly, from $\rho$).
For **variables** in \( x \in \mathcal{X} \), we use \( \sigma \):

\[
[x]^{\sigma}(1^\eta, \rho) \overset{\text{def}}{=} \sigma(x)(1^\eta, \rho),
\]

**Names** \( n \in \mathcal{G} \) are interpreted as **uniform random samplings** among bitstrings of length \( \eta \), extracted from \( \rho_p \):

\[
[n]^{\sigma}(1^\eta, \rho) \overset{\text{def}}{=} M_n(\eta, \rho_p)
\]

For every pair of different names \( n_0, n_1 \), we require that \( M_{n_0} \) and \( M_{n_1} \) extracts disjoint parts of \( \rho_p \).

\[\text{\textbullet \ Hence different names are independent random samplings.}\]

We force the interpretation of some function symbols.

- if_then_else_ is interpreted as branching:

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

- _ \overset{=} {_} is interpreted as an equality test:

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

Similarly, we force the interpretations of \( \wedge, \lor, \to, \text{true}, \text{false} \).
A First-Order Logic for Indistinguishability
We now present a logic, to state (and latter prove) properties about bitstring distributions.

This is a first-order logic with a single predicate $\sim$,\(^1\) representing computational indistinguishability.

$$\phi := \top \mid \bot \mid \phi \land \phi \mid \phi \lor \phi \mid \phi \rightarrow \phi \mid \neg \phi \mid \forall x. \phi \mid \exists x. \phi \mid t_1, \ldots, t_n \sim t_{n+1}, \ldots, t_{2n} \quad (t_1, \ldots, t_{2n} \in T(F, G, N, X))$$

Remark: we use $\land, \lor, \rightarrow$ 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 $n \in \mathbb{N}$. 

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

\[ [\phi]_\mathcal{M} \in \{ \text{True, False} \} \] is as expected for boolean connective and FO quantifiers. E.g.:

\[ [\top]_\mathcal{M} \overset{\text{def}}{=} \text{True} \]

\[ [\phi \land \psi]_\mathcal{M} \overset{\text{def}}{=} [\phi]_\mathcal{M} \text{ and } [\psi]_\mathcal{M} \]

\[ [\neg \phi]_\mathcal{M} \overset{\text{def}}{=} \text{not } [\phi]_\mathcal{M} \]

\[ [\forall x. \phi]_\mathcal{M} \overset{\text{def}}{=} \text{True } \text{ if } \forall m \in \mathcal{D}, [\phi]_\mathcal{M}[x \mapsto m] \overset{\text{def}}{=} \text{True} \]
Finally, $\sim_n$ is interpreted as **computational indistinguishability**.

$$[ t_1, \ldots , t_n \sim_n s_1, \ldots , s_n ]^\sigma_M = \text{True}$$

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

$$\left| \Pr_\rho ( A(1^n, ([t_i]^\sigma_M(1^n, \rho))_{1 \leq i \leq n, \rho_a} = 1) \right| - \left| \Pr_\rho ( A(1^n, ([s_i]^\sigma_M(1^n, \rho))_{1 \leq i \leq n, \rho_a} = 1) \right|$$

is a **negligible** function of $\eta$.

*The quantity in (⋆) is called the **advantage** of $A$ 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:

$$f(\eta) \overset{\text{def}}{=} \Pr_\rho([n_0](1^n, \rho) = [n_1](1^n, \rho))$$

If $n_0 \neq n_1$, then $f(\eta) = \frac{1}{2^n}$, and $f$ is negligible.
A formula $\phi$ is **satisfied** by a computational model $\mathcal{M}$, written $\mathcal{M} \models \phi$, if $\llbracket \phi \rrbracket^\sigma_{\mathcal{M}} = \text{True}$ for every valuation $\sigma$.

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

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

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

\begin{align*}
\text{true} & \sim \text{false} \\
n_0 & \sim n_0 \\
n_0 & \sim n_1 \\
n_0 \equiv n_1 & \sim \text{false} \\
n_0, n_0 & \sim n_0, n_1 \\
 f(n_0) & \sim f(n_1) \quad \text{where} \quad f \in \mathcal{F} \cup \mathcal{G} \\
\pi_1(\langle n_0, n_1 \rangle) & \equiv n_0 \sim \text{true}
\end{align*}
Exercise

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

\[ \not\models \text{true} \sim \text{false} \quad \models n_0 \sim n_0 \quad \models n_0 \sim n_1 \quad \models n_0 \equiv n_1 \sim \text{false} \]

\[ \not\models n_0, n_0 \sim n_0, n_1 \quad \models f(n_0) \sim f(n_1) \quad \text{where } f \in \mathcal{F} \cup \mathcal{G} \]

\[ \not\models \pi_1(\langle n_0, n_1 \rangle) \parallel n_0 \sim \text{true} \]
\( \mathcal{P} \) and \( \mathcal{Q} \) are **indistinguishable**, written \( \mathcal{P} \approx \mathcal{Q} \), if for any \( \tau \):

\[
\models \text{fold}(\mathcal{P}, \tau) \sim \text{fold}(\mathcal{Q}, \tau)
\]

**Remark**

While there are countably many observable traces \( \tau \), the set of **foldings** of a protocol \( \mathcal{P} \) is always **finite**:\(^2\)

\[
\left| \{ \text{fold}(\mathcal{P}, \tau) \mid \tau \} \right| < +\infty
\]

\(^2\)If we remove trailing sequences of error terms.
Exercise
Informally, determine which of the following protocols indistinguishabilities hold, and under what assumptions:

\[
\begin{align*}
\text{out}(c, t_1) & \approx \text{out}(c, t_2) & \text{out}(c, t) & \approx \text{null} & \text{in}(c, x) & \approx \text{null} \\
\text{out}(c, t) & \approx \text{if } b \text{ then } \text{out}(c, t_1) \text{ else } \text{out}(c, t_2) \\
\text{out}(c, t) & \approx \text{if } b \text{ then } \text{out}(c, t) \text{ else } \text{out}(c_0, t_0)
\end{align*}
\]
Structural Rules
A rule:

\[
\frac{\phi_1 \ldots \phi_n}{\phi}
\]

is sound if \( \phi \) is valid whenever \( \phi_1, \ldots, \phi_n \) are valid.

**Example**

\[
\frac{y \sim x}{x \sim y}
\]

is sound

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

Computational indistinguishability is an **equivalence relation**:

\[
\begin{align*}
\vec{u} &\sim \vec{u} \quad \text{REFL} \\
\vec{v} &\sim \vec{u} \quad \vec{u} \sim \vec{v} \quad \text{SYM} \\
\vec{u} &\sim \vec{w} \quad \vec{w} \sim \vec{v} \quad \text{TRANS}
\end{align*}
\]

**Permutation.** If \( \pi \) is a permutation of \( \{1, \ldots, n\} \) then:

\[
\begin{align*}
\vec{u}_{\pi(1)}(1), \ldots, \vec{u}_{\pi(n)}(n) &\sim \vec{v}_{\pi(1)}(1), \ldots, \vec{v}_{\pi(n)}(n) \\
\vec{u}_{1}, \ldots, \vec{u}_{n} &\sim \vec{v}_{1}, \ldots, \vec{v}_{n} \quad \text{PERM}
\end{align*}
\]
Alpha-renaming.

\[
\frac{}{\vec{u} \sim \vec{u}\alpha} \quad \alpha\text{-EQU}
\]

when \(\alpha\) is an injective renaming of names in \(\mathcal{N}\).

Restriction. The adversary can throw away some values:

\[
\frac{\vec{u}, s \sim \vec{v}, t}{\vec{u} \sim \vec{v}} \quad \text{RESTR}
\]
**Structural Rules**

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

\[
\vec{u}, s \sim \vec{v}, t \\
\vec{u}, s, s \sim \vec{v}, t, t \\
\text{DUP}
\]

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

\[
\vec{u}_1, \vec{v}_1 \sim \vec{u}_1, \vec{v}_2 \\
f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2 \\
\text{FA}
\]

where \( f \in \mathcal{F} \cup \mathcal{G} \).
Structural Rules: Proof of Function Application

\[
\begin{align*}
\vec{u}_1, \vec{v}_1 &\sim \vec{u}_1, \vec{v}_2 \\
\frac{f(\vec{u}_1), \vec{v}_1 &\sim f(\vec{u}_2), \vec{v}_2}{\text{FA}}
\end{align*}
\]

**Proof.** The proof is by contrapositive. Assume \(\mathcal{M}, \sigma\) and \(\mathcal{A}\) s.t. its advantage against:

\[
f(\vec{u}_1), \vec{v}_1 \sim f(\vec{u}_2), \vec{v}_2 \tag{†}
\]

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

\[
\mathcal{B}(1^n, \vec{w}_u, \vec{w}_v, \rho_a) \overset{\text{def}}{=} \mathcal{A}(1^n, \lceil f \rceil_{\mathcal{M}}(\vec{w}_u), \vec{w}_v, \rho_a)
\]

\(\mathcal{B}\) is a PPTM since \(\mathcal{A}\) is and \(\lceil f \rceil_{\mathcal{M}}\) can be evaluated in pol. time. Then:

\[
\begin{align*}
\mathcal{B}(1^n, \lceil \vec{u}_i \rceil_{\mathcal{M}}(1^n, \rho), \lceil \vec{v}_i \rceil_{\mathcal{M}}(1^n, \rho), \rho_a) \\
= \mathcal{A}(1^n, \lceil f(\vec{u}_i) \rceil_{\mathcal{M}}(1^n, \rho), \lceil \vec{v}_i \rceil_{\mathcal{M}}(1^n, \rho), \rho_a) \quad (i \in \{1, 2\})
\end{align*}
\]

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 (†). \(\square\)
Case Study. We can do case disjunction over branching terms:

\[
\begin{align*}
\vec{w}_1, b_0, u_0 & \sim \vec{w}_1, b_1, u_1 \\
\vec{w}_0, b_0, v_0 & \sim \vec{w}_1, b_1, v_1 \\
\vec{w}_0, \text{if } b_0 \text{ then } u_0 \text{ else } v_0 & \sim \vec{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1
\end{align*}
\]
Structural Rules: Proof of Case Study

\begin{align*}
  & b_0, u_0 \sim b_1, u_1 \quad b_0, v_0 \sim b_1, v_1 \\
  \quad 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
\end{align*}

CS

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

\[
\text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \text{if } b_1 \text{ then } u_1 \text{ else } v_1 
\]

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

\[
\mathcal{B}_\top(1^n, w_b, w, \rho_a) \overset{\text{def}}{=} \begin{cases} 
\mathcal{A}(1^n, w, \rho_a) & \text{if } w_b = 1 \\
0 & \text{otherwise}
\end{cases}
\]

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

\[
\Pr_{\rho} \left( \mathcal{B}_\top(1^n, [b_i]_{\mathcal{M}}(1^n, \rho), [u_i]_{\mathcal{M}}(1^n, \rho), \rho_a) = 1 \right) = \Pr_{\rho} \left( \mathcal{A}(1^n, [t_i]_{\mathcal{M}}(1^n, \rho), \rho_a) = 1 \wedge [b_i]_{\mathcal{M}}(1^n, \rho) = 1 \right) \overset{p_{\top}, i}{=} 
\]
Hence the advantage of $B_{\top}$ against $b_0, u_0 \sim b_1, u_1$ is $|p_{\top,1} - p_{\top,0}|$.

Similarly, let $B_{\bot}$ be the distinguisher:

$$B_{\bot}(1^n, w_b, w, \rho_a) \overset{\text{def}}{=} \begin{cases} A(1^n, w, \rho_a) & \text{if } w_b \neq 1 \\ 0 & \text{otherwise} \end{cases}$$

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

$$\Pr_\rho\left(A(1^n, [t]_\mathcal{M}(1^n, \rho), \rho_a) = 1 \land [b]_\mathcal{M}(1^n, \rho) \neq 1 \right)$$
The advantage of $A$ against $t_0 \sim t_1$ is, by partitioning and triangular inequality:

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

Since $A$’s advantage is non-negligible, at least one of the two quantity above is non-negligible. Hence either $B^{\top}$ or $B_{\perp}$ has a non-negligible advantage against a premise of the CS rule. □.
Counter-Examples

Remark that $b$ is **necessary** in CS

$$\vec{w}_1, b_0, u_0 \sim \vec{w}_1, b_1, u_1 \quad \vec{w}_0, b_0, v_0 \sim \vec{w}_1, b_1, v_1$$

CS

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

We have:

$$\models 0 \sim 0 \quad \models n_0 \sim n_1 \quad \models \text{even}(n_0) \sim \text{even}(n_0)$$

But:

$$\not\models \text{if even}(n_0) \text{ then } n_0 \text{ else } 0 \sim \text{if even}(n_0) \text{ then } n_1 \text{ else } 0$$

*Why is the later formula not valid?*
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 $(s = t) \overset{\text{def}}{=} (s \doteq t) \sim \text{true}$. Then the following rule is sound:

$$
\frac{\vec{u}, t \sim \vec{v} \quad s = t}{\vec{u}, s \sim \vec{v}} \quad \mathsf{R}
$$
Structural Rules: FO + Equality Reasoning

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

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

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

$$\mathcal{F}_{\text{FO}}(\rightarrow, \text{false}, \vdash, \mathcal{F} \cup \mathcal{G})$$

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

Example

$\mathcal{A}_{th} \vdash_{\text{FO}=} (v \doteq w \rightarrow \text{if } u \doteq v \text{ then } u \text{ else } t \doteq s) =
(v \doteq w \rightarrow \text{if } u \doteq v \text{ then } w \text{ else } t \doteq s)$
Structural Rules: Probability Independence

Two rules exploiting the independence of bitstring distributions:

\[ \frac{(t \equiv n) = \text{false}}{\equiv \text{-IND}} \quad \text{when } n \not\in \text{st}(t) \]

\[ \frac{\vec{u} \sim \vec{v}}{\vec{u}, n_0 \sim \vec{v}, n_1} \quad \text{FRESH} \quad \text{when } n_0 \not\in \text{st}(\vec{u}) \text{ and } n_1 \not\in \text{st}(\vec{v}) \]

**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).
We give the proof of the first rule:

\[ (t \equiv n) = \text{false} \quad \Rightarrow \text{IND} \quad \text{when } n \not\in \text{st}(t) \]

**Proof.** For any computational model \( M \) (we omit it below):

\[
\Pr_{\rho}([t \equiv n](1^n, \rho) = 1) \\
= \Pr_{\rho}([t](1^n, \rho) = [n](1^n, \rho)) \\
= \sum_{w \in \{0,1\}^*} \Pr_{\rho}([t](1^n, \rho) = w \land [n](1^n, \rho) = w) \\
= \sum_{w \in \{0,1\}^*} \Pr_{\rho}([t](1^n, \rho) = w) \cdot \Pr_{\rho}([n](1^n, \rho) = w) \\
= \frac{1}{2^n} \cdot \sum_{w \in \{0,1\}^*} \Pr_{\rho}([t](1^n, \rho) = w) \\
= \frac{1}{2^n}
\]
Exercise

Give a derivation of the following formula:

\[ n_0 \sim \text{if } b \text{ then } n_0 \text{ else } n_1 \quad (\text{when } n_0, n_1 \notin \text{st}(b)) \]
Implementation Rules
A rule is \textbf{C-sound} if \( \phi \) is \textbf{C-valid} whenever \( \phi_1, \ldots, \phi_n \) are \textbf{C-valid}.

\textbf{Example}

\[
(\pi_1\langle x, y \rangle \div x) \sim \text{true}
\]

is \textbf{not} \textbf{sound}, because we do not require anything on the interpretation of \( \pi_1 \) and the pair.

Obviously, it is \( \textbf{C}_\pi \)-\textbf{sound}, where \( \textbf{C}_\pi \) is the set of computational model where \( \pi_1 \) computes the first projection of the pair \( \langle \_, \_ \rangle \).
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).
Example. Equational theories for protocol functions:

- $\pi_i (\langle x_1, x_2 \rangle) = x_i$  
  \[ i \in \{1, 2\} \]

- $\text{dec}(\{x\}_{pk(y)}, sk(y)) = x$

- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

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

<table>
<thead>
<tr>
<th>Cryptographic Reduction $S \leq_{\text{red}} \mathcal{H}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you can break the <strong>cryptographic design</strong> $S$, then you can break the <strong>hardness assumption</strong> $\mathcal{H}$ using roughly the same time.</td>
</tr>
</tbody>
</table>

- 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, $S$ cannot be broken in a reasonable time.
Cryptographic Reduction $S \leq_{\text{red}} H$

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

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

where $A$ and $B$ are taken among PPTMs and $P$ is a polynomial.
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
An **asymmetric encryption scheme** contains:

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

It must satisfies the functional equality:

\[
\text{dec}(\{x\}_{pk(y)}^z, sk(y)) = x
\]

\(^3\)The role of the randomization will become clear later.
An encryption scheme is indistinguishable against chosen cipher-text attacks (IND-CCA₁) iff for every PPTM \( A \) with access to:

- a left-right oracle \( \mathcal{O}_{LR}^{b,n}(\cdot, \cdot) \):
  \[
  \mathcal{O}_{LR}^{b,n}(m_0, m_1) \overset{\text{def}}{=} \begin{cases} 
  \{m_b\}_r^{pk(n)} & \text{if len}(m_1) = \text{len}(m_2) \quad (r \text{ fresh}) \\
  0 & \text{otherwise}
  \end{cases}
  \]

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

where \( A \) can call \( \mathcal{O}_{LR} \) once, and cannot call \( \mathcal{O}_{\text{dec}} \) after \( \mathcal{O}_{LR} \), then:

\[
\left| \Pr_n \left( A^{\mathcal{O}_{LR}^{1,n}, \mathcal{O}_{\text{dec}}^{n}}(1^n, pk(n)) = 1 \right) - \Pr_n \left( A^{\mathcal{O}_{LR}^{0,n}, \mathcal{O}_{\text{dec}}^{n}}(1^n, pk(n)) = 1 \right) \right|
\]

is negligible in \( \eta \), where \( n \) is drawn uniformly in \( \{0, 1\}^\eta \).
Exercise
Show that if the encryption ignore its randomness, i.e. there exists $\text{aenc}(\_,\_)$ s.t. for all $x, y, r$:

$$\{x\}^r_y = \text{aenc}(x, y)$$

then the encryption does not satisfy IND-CCA₁.
Indistinguishability Against Chosen Ciphertexts Attacks
If the encryption scheme is $\text{IND-CCA}_1$, then the ground rule:

\[
\frac{\text{len}(t_0) = \text{len}(t_1)}{	ilde{u}, \{t_0\}_r^{pk(n)} \sim \tilde{u}, \{t_1\}_r^{pk(n)}} \quad \text{IND-CCA}_1
\]

is sound, when:

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

Proof sketch

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

$$\left| \Pr_\rho(\mathcal{A}(1^n, \llbracket \bar{u} \rrbracket, \mathcal{M}(1^n, \rho), \llbracket \{t_0\}\}_{pk(n)}^r \mathcal{M}(1^n, \rho, \rho_a)) - \Pr_\rho(\mathcal{A}(1^n, \llbracket \bar{u} \rrbracket, \mathcal{M}(1^n, \rho), \llbracket \{t_1\}\}_{pk(n)}^r \mathcal{M}(1^n, \rho, \rho_a)) \right|$$

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

We must build a PPTM $\mathcal{B}$ s.t. $\mathcal{B}$ wins the $\text{IND-CCA}_1$ security game.
IND-CCA₁ Rule: Proof

Let $B^{O_{LR}, O_{\text{dec}}^n}(1^n, [\text{pk}(n)] \mathcal{M}(1^n, \rho))$ be the following program:

i) **lazily** samples the infinite random tapes $(\rho_a, \rho'_p)$ where:

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

ii) compute⁴:

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

using $(\rho_a, \rho'_p), [\text{pk}(n)] \mathcal{M}(1^n, \rho)$ and calls to $O_{\text{dec}}^n$.

iii) compute:

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

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

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

---

⁴we describe how later
IND-CCA₁ Rule: Proof

Then $B$ advantage against IND-CCA₁ is exactly $A$ advantage against:

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

which we assumed non-negligible.
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 $n$, $sk(n)$ or $r$;
- or $B$ can compute $w_s := [s]_{M}(1^n, \rho)$ 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.
**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.
- $s \in \{r, n\}$ are forbidden, hence the induction hypothesis holds.
- $s \in \mathcal{N}\backslash\{r, n\}$, then $B$ computes $s$ directly from
  $\rho'_p = \rho_p[n \mapsto 0, r \mapsto 0]$.
- $s \equiv f(t_1, \ldots, t_n)$ and $t_1, \ldots, t_n$ are not forbidden. Then, by
  induction hypothesis, $B$ can compute $w_i := \llbracket t_i \rrbracket_M(1^n, \rho)$ for any
  $1 \leq i \leq n$. Then $B$ simply computes:

$$w_s := \begin{cases} \llbracket f \rrbracket_M(w_1, \ldots, w_n) & \text{if } f \in \mathcal{F} \\ \llbracket f \rrbracket_M(w_1, \ldots, w_n, \rho_a) & \text{if } f \in \mathcal{G} \end{cases}$$
case disjunction (continued):

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

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

The first case is immediate since $B$ receives $\left[\text{pk}(n)\right]_M(1^n, \rho)$ as argument.

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

For the last case, from \texttt{IND-CCA}$_1$ side conditions, we know that $m \neq r$ and $m \neq n$. Hence, by \textit{induction hypothesis}, $B$ can compute $w_m = \left[m\right]_M(1^n, \rho)$. We conclude using:

$$w_s := \mathcal{O}_{\text{dec}}^n(w_m)$$
Exercise

Which of the following formulas can be proven using \textsc{IND-CCA}_1?

\[
\text{pk}(n), \{0\}^r_{\text{pk}(n)} \sim \text{pk}(n), \{1\}^r_{\text{pk}(n)}
\]

\[
\text{pk}(n), \{0\}^r_{\text{pk}(n)}, \{0\}^{ro}_{\text{pk}(n)} \sim \text{pk}(n), \{1\}^r_{\text{pk}(n)}, \{0\}^{ro}_{\text{pk}(n)}
\]

\[
\text{pk}(n), \{0\}^r_{\text{pk}(n)}, \{0\}^r_{\text{pk}(n)} \sim \text{pk}(n), \{0\}^r_{\text{pk}(n)}, \{1\}^r_{\text{pk}(n)}
\]

\[
\text{pk}(n), \{0\}^r_{\text{pk}(n)} \sim \text{pk}(n), \{\text{sk}(n)\}^r_{\text{pk}(n)}
\]
Exercise (Hybrid Argument)

Prove the following formula using \( \text{IND-CCA}_1 \):

\[
\{0\}_{pk(n)}^{r_0}, \{1\}_{pk(n)}^{r_1}, \ldots, \{n\}_{pk(n)}^{r_n} \sim \{0\}_{pk(n)}^{r_0}, \{0\}_{pk(n)}^{r_1}, \ldots, \{0\}_{pk(n)}^{r_n}
\]

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 $A$ with access to:

- a left-right encryption oracle $O^{b,n_0,n_1}_{LR}(\cdot)$:

  $$O^{b,n_0,n_1}_{LR}(m) \overset{\text{def}}{=} \{m\}^r_{pk(n_b)} \quad (r \text{ fresh})$$

- and two decryption oracles $O^{n_0}_{dec}(\cdot)$ and $O^{n_1}_{dec}(\cdot)$,

where $A$ can call $O_{LR}$ once, and cannot call the decryption oracles after $O_{LR}$, then:

$$\left| \Pr_{n_0,n_1}(A^{O^{1,n_0,n_1}_{LR},O^{n_0}_{dec},O^{n_1}_{dec}}(1^n, pk(n_0), pk(n_1))) = 1) - \Pr_{n_0,n_1}(A^{O^{0,n_0,n_1}_{LR},O^{n_0}_{dec},O^{n_1}_{dec}}(1^n, pk(n_0), pk(n_1))) = 1) \right|$$

is negligible in $\eta$, where $n_0, n_1$ are drawn in $\{0, 1\}^\eta$. 
Exercise
Show that $\text{IND-CCA}_1 \not\Rightarrow \text{KP-CCA}_1$ and $\text{KP-CCA}_1 \not\Rightarrow \text{IND-CCA}_1$. 
Key Privacy Against Chosen Ciphertexts Attacks

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

$$\vec{u}, \{t\}^r_{pk(n_0)} \sim \vec{u}, \{t\}^r_{pk(n_1)}$$

is sound, when:

- $r$ does not appear in $\vec{u}, t$;
- $n_0, n_1$ appear only in $pk(\cdot)$ or $\text{dec}(\_, sk(\cdot))$ positions in $\vec{u}, t$.

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

Collision Resistance
A **keyed cryptographic hash** $H(\_\_, \_\_)$ is **computationally collision resistant** if no PPTM adversary can build 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 $A$, the following quantity:

$$\Pr_k\left(A^{O_{H(\_,\_)}}(1^n) = \langle m_1, m_2 \rangle, m_1 \neq m_2 \text{ and } H(m_1, k) = H(m_2, k)\right)$$

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

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

\[
H(m_1, k) \neq H(m_2, k) \rightarrow m_1 \neq m_2 \sim \text{true} \quad \text{CR}
\]

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

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

$$
\therefore (H(m_1, k) \equiv H(m_2, k)) \sim \text{true} \quad \text{CR}
$$

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

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 satisfy the functional equality:

\[
\text{verify}_k(\text{mac}_k(m), m) = \text{true}
\]
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 $A$, the following quantity:

$$
\Pr_k \left( A^{O_{\text{mac}_k}(\cdot), O_{\text{verify}_k}(\cdot, \cdot)} (1^n) = \langle m, \sigma \rangle, \text{m not queried to } O_{\text{mac}_k}(\cdot) \right)
$$

$$
\text{and verify}_k (\sigma, m) = 1
$$

is negligible, where $k$ is drawn uniformly in $\{0,1\}^\eta$. 
**Message Authentication Code Unforgeability**

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

\[
\text{verify}_k(s, m) \rightarrow \bigvee_{u \in S} m \doteq u \sim \text{true}
\]

is sound, when:

- \( S = \{ u \mid \text{mac}_k(u) \in \text{st}(s, m) \} \);
- \( k \) 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:

\[
\Phi \equiv \text{mac}_k(t_1), \text{mac}_k(t_2), \text{mac}_{k_0}(t_3)
\]

\[
\models \text{verify}_k(g(\Phi), n) \rightarrow (n \doteq t_1 \lor n \doteq t_2) \sim \text{true}
\]
Exercise
Assume mac is EUF-CMA. Show that the following rule is sound:

\[ \text{verify}_k(\text{if } b \text{ then } s_0 \text{ else } s_1, m) \rightarrow \bigvee_{u \in S_1 \cup S_2} m \models u \sim \text{true} \]

when \( b, s_0, s_1, m \) are ground terms, and:

- \( S_i = \{ u \mid \text{mac}_k(u) \in \text{st}(s_i, m) \} \), 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 \( b \equiv (\text{att}(k) \models \text{mac}_k(0)) \).
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 $I_A | S_A$ from $I_C | S_A$.

$l_X : \nu r. \nu n_I$. \hspace{1cm} out$(c_I, \{\langle pk_X , n_I \rangle \}^r_{pk_S})$

$s_X : \nu r_0. \nu n_S. in(c_I, x)$. if $\pi_1(d) \doteq pk_X$
then $\textbf{out}(c_S, \{\langle \pi_2(d) , n_S \rangle \}^{r_0}_{pk_X})$
else $\textbf{out}(c_S, \{0\}^{r_0}_{pk_X})$

We assume the encryption is IND-CCA$_1$ and KP-CCA$_1$. 
As we saw, an encryption does not hide the length of the plain-text. Hence, since $\text{len}(\langle n_I, n_S \rangle) \neq \text{len}(0)$, there is an attack:

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

even if the encryption is IND-CCA$_1$ and KP-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**

\[
\begin{align*}
I_X &: \nu r. \nu n_I. \quad \text{out}(c_I, \{\langle pk_X, n_I \rangle \}^r_{pk_S}) \\
S_X &: \nu r_0. \nu n_S. \text{in}(c_I, x). \quad \text{if } \pi_1(d) = pk_X \land \text{len}(\pi_2(d)) = \text{len}(n_S) \\
&\quad \text{then out}(c_S, \{\langle \pi_2(d), n_S \rangle \}^{r_0}_{pk_X}) \\
&\quad \text{else out}(c_S, \{\langle n_S, n_S \rangle \}^{r_0}_{pk_X})
\end{align*}
\]
Private Authentication: Anonymity

\[ I_X : \nu r. \nu n_I. \quad \text{out}(c_I, \{\langle pk_X, n_I \rangle\}^r_{pk_S}) \]

\[ S_X : \nu r_0. \nu n_S. \text{in}(c_I, x). \text{if } \pi_1(d) \doteq pk_X \land \text{len}(\pi_2(d)) \doteq \text{len}(n_S) \]
\[ \text{then } \text{out}(c_S, \{\langle \pi_2(d), n_S \rangle\}^{r_0}_{pk_X}) \]
\[ \text{else } \text{out}(c_S, \{\langle n_S, n_S \rangle\}^{r_0}_{pk_X}) \]

To prove \( I_A \mid S_A \approx I_C \mid S_A \), we have several traces:

\[ \text{in}(c_I), \text{out}(c_I), \text{out}(c_S) \]
\[ \text{out}(c_I), \text{in}(c_I), \text{out}(c_S) \]
\[ \text{out}(c_S), \text{in}(c_I), \text{out}(c_I) \]
Private Authentication: Anonymity

\[ I_X : \nu r. \nu n_I. \quad \text{out}(c_I, \{\langle pk_X, n_I \rangle \}_r) \]

\[ S_X : \nu r_0. \nu n_S. \text{in}(c_I, x). \text{if } \pi_1(d) \equiv pk_X \land \text{len}(\pi_2(d)) \equiv \text{len}(n_S) \]

\[ \quad \text{then out}(c_S, \{\langle \pi_2(d), n_S \rangle \}_r) \]

\[ \quad \text{else out}(c_S, \{\langle n_S, n_S \rangle \}_r) \]

To prove \( I_A | S_A \approx I_C | S_A \), we have several traces:

\[ \text{in}(c_I), \text{out}(c_I), \text{out}(c_S) \]

\[ \text{in}(c_I), \text{out}(c_S), \text{out}(c_I) \]

\[ \text{out}(c_I), \text{in}(c_I), \text{out}(c_S) \]

\[ \text{out}(c_I), \text{out}(c_S), \text{in}(c_I) \]

\[ \text{out}(c_S), \text{in}(c_I), \text{out}(c_I) \]

\[ \text{out}(c_S), \text{out}(c_S), \text{in}(c_I) \]

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

See partial order reduction (POR) techniques [1].
We must prove that:

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

where:

\[ \text{out}_1^X \equiv \{ \langle \text{pk}_X, \text{n}_I \rangle \}^{r}_{\text{pk}_S} \]

\[ \text{out}_2^{X,Y} \equiv \begin{cases} 
\{ \langle \pi_2(d_X), \text{n}_S \rangle \}^{r_0}_{\text{pk}_Y} & \text{if } \pi_1(d_X) = \text{pk}_X \land \text{len}(\pi_2(d_X)) = \text{len}(\text{n}_S) \\
\{ \langle \text{n}_S, \text{n}_S \rangle \}^{r_0}_{\text{pk}_Y} & \text{else} \end{cases} \]

\[ d_X \equiv \text{dec}(\text{att}_0(\text{out}_1^X), \text{sk}_S) \]
First, we push the branching under the encryption:

\[
\begin{align*}
\text{out}_1^A, \text{out}_2^{A,A} &\sim \text{out}_1^C, \text{out}_2^{A,A} \\
\text{out}_1^A, \text{out}_2^{A,A} &\sim \text{out}_1^C, \text{out}_2^{A,A} \\
\text{out}_1^A, \text{out}_2^{A,A} &\sim \text{out}_1^C, \text{out}_2^{A,A}
\end{align*}
\]

\[
R \\
\]

where:

\[
\text{out}_{2}^{X,Y} \equiv \begin{cases} 
\text{if } \pi_1(d_X) \equiv \text{pk}_X \land \text{len}(\pi_2(d_X)) \equiv \text{len}(n_S) \quad \text{then } \langle \pi_2(d_X), n_S \rangle \\
\quad \text{else } \langle n_S, n_S \rangle
\end{cases}^{r_0} \quad \text{pk}_Y
\]

We let \( m_X \) be the content of the encryption above.
Then, we use $\text{KP-CCA}_1$ to change the encryption key:

$$
\begin{align*}
\text{out}_1^A, \text{out}_2^A, A & \sim \text{out}_1^C, \text{out}_2^A, C \\
\text{out}_1^C, \text{out}_2^A, C & \sim \text{out}_1^C, \text{out}_2^A, A
\end{align*}
$$

\begin{align*}
\text{ KP-CCA}_1 \\
\text{ TRANS }
\end{align*}

since:

- the encryption randomness $r_0$ is correctly used;
- the key randomness $n_A$ and $n_B$ appear only in $\text{pk}(\cdot)$ and $\text{dec}(\_, \text{sk}(\cdot))$ positions.
Then, we use $\text{IND-CCA}_1$ to change the encryption content:

$$\begin{align*}
\text{out}_1^A, \text{out}_2^{A,A} &\sim \text{out}_1^C, \text{out}_2^{C,C} \\
\text{out}_1^C, \text{out}_2^{C,C} &\sim \text{out}_1^C, \text{out}_2^{A,C} \\
\text{out}_1^A, \text{out}_2^{A,A} &\sim \text{out}_1^C, \text{out}_2^{A,C}
\end{align*}$$

since:

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

\[ m_X \equiv \text{if } \pi_1(d_X) \equiv \text{pk}_X \land \text{len}(\pi_2(d_X)) \equiv \text{len}(n_S) \]

then \( \langle \pi_2(d_X), n_S \rangle \)
else \( \langle n_S, n_S \rangle \)

Then:

\[
\frac{\mathcal{A}_{th} \vdash_{\text{FO}} \text{len}(m_C) = \text{len}(m_A)}{\text{FO}} \quad \text{len}(m_C) = \text{len}(m_A)
\]

if \( \mathcal{A}_{th} \) contains the axiom\(^5\):

\[ \forall x, y. \text{len}((x, y)) = c_{\langle \_ , \_ \rangle}(\text{len}(x), \text{len}(y)) \]

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.
Then, we \( \alpha \)-rename the key randomness \( n_C \), rewrite back the encryption, and conclude.

\[
\begin{align*}
\text{out}^A_1, \text{out}^A_2 & \sim \text{out}^A_1, \text{out}^A_2 & \text{Refl} \\
\text{out}^A_1, \text{out}^A_2 & \sim \text{out}^A_1, \text{out}^A_2 & \alpha\text{-EQU} \\
\text{out}^A_1, \text{out}^A_2 & \sim \text{out}^C_1, \text{out}^C_2 & \end{align*}
\]
Privacy
We proved **anonymity** of the Private Authentication protocol, which we defined as:

\[
I_A | S_A \approx I_C | S_A
\]

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

⇒ *No, because of linkability attacks*
Consider the following authentication protocol, called \textit{KCL}, between a reader $R$ and a tag $T_X$ with identity $X$:

\[
R : \nu n_R. \text{ out}(c_R, n_R) \\
T_X : \nu n_T. \text{ in}(c_R, x). \text{ out}(c_I, \langle X \oplus n_T , n_T \oplus H(x, k_X) \rangle )
\]

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

\[
T_A | R \approx T_B | R
\]
But there are **privacy attacks** against KCL, using two sessions:

1. \(E \rightarrow T_A : n_R\)
2. \(T_A \rightarrow E : \langle A \oplus n_T, n_T \oplus H(n_R, k_A) \rangle\)

3. \(E \rightarrow T_A : n_R\)
4. \(T_A \rightarrow E : \langle A \oplus n'_T, n'_T \oplus H(n_R, k_A) \rangle\)

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

\[
\pi_2(t_2) \oplus \pi_2(t_4) = (n_T \oplus H(n_R, k_A)) \oplus (n'_T \oplus H(n_R, k_A)) \\
= n_T \oplus n'_T \\
= \pi_1(t_2) \oplus \pi_1(t_4)
\]

The same equality check will almost never hold on the right, under reasonable assumption on \(H\).
We just saw an attack against:

\[(T_A | R) | (T_A | R) \approx (T_A | R) | (T_B | R)\]
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**:
  \[ \nu \vec{k}_0, \ldots, \vec{k}_N. \ !_{\text{id} \leq N} !_{\text{sid} \leq M} P(\vec{k}_{\text{id}}) \]

- and an **ideal-world**, where each agent run at most a **single session**:
  \[ \nu \vec{k}_{0,0}, \ldots, \vec{k}_{N,M}. \ !_{\text{id} \leq N} !_{\text{sid} \leq M} P(\vec{k}_{\text{id},\text{sid}}) \]

**Remark**

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

For the sake of simplicity, we omit channel names.
Example  An unlinkability scenario.

A  B  A  B  B  B

∼

A  B  C  D  E  F
In the **ideal-world**, relations between sessions **cannot leak** any information on identities.

⇒ **hence no link** can be **efficiently found** in the **real world**.
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:

\[
\nu \vec{k}_S, \nu \vec{k}_0, \ldots, \nu \vec{k}_N, !\text{id} \leq N !\text{sid} \leq M (P(\vec{k}_S, \vec{k}_\text{id}) | S(\vec{k}_S, \vec{k}_\text{id})) \\
\approx \nu \vec{k}_S, \nu \vec{k}_{0,0}, \ldots, \nu \vec{k}_{N,M}, !\text{id} \leq N !\text{sid} \leq M (P(\vec{k}_S, \vec{k}_{\text{id}\text{sid}}) | S(\vec{k}_S, \vec{k}_{\text{id}\text{sid}}))
\]
Unlinkability: Adding Servers

Generic server, accepting all identities. No changes for the user process \( P(\vec{k}_S, \vec{k}_U) \).

The server \( S(\vec{k}_S, \vec{k}_{U_1}, \ldots, \vec{k}_{U_M}) \) is parameterized by:

- some **global** key material \( \vec{k}_S \);
- all **users** key material \( \vec{k}_{U_1}, \ldots, \vec{k}_{U_M} \).

The we require that:

\[
\nu \vec{k}_S, \nu \vec{k}_0, \ldots, \vec{k}_N. \quad (\text{id} \leq N \text{ id} \leq M P(\vec{k}_S, \vec{k}_{\text{id}})) \mid (\text{id} \leq L S(\vec{k}_S, \vec{k}_0, \ldots, \vec{k}_N))
\]

\[
\approx \nu \vec{k}_S, \nu \vec{k}_{0,0}, \ldots, \vec{k}_{N,M}. \quad (\text{id} \leq N \text{ sid} \leq M P(\vec{k}_S, \vec{k}_{\text{id},\text{sid}})) \mid (\text{id} \leq L S(\vec{k}_S, \vec{k}_{0,0}, \ldots, \vec{k}_{N,M}))
\]
**Private Authentication**

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

\[ I(k_S, k_X) : \nu r. \nu n_I. \quad \text{out}(c_I, \{\langle pk_X, n_I \rangle\}_{pk_S}^r) \]

\[ S(k_S, k_X) : \nu r_0. \nu n_S. \text{in}(c_I, x). \text{if } \pi_1(d) = pk_X \land \text{len}(\pi_2(d)) = \text{len}(n_S) \]

\[ \text{then } \text{out}(c_S, \{\langle \pi_2(d), n_S \rangle\}_{pk_X}^{r_0}) \]

\[ \text{else } \text{out}(c_S, \{\langle n_S, n_S \rangle\}_{pk_X}^{r_0}) \]

where \( sk_X = sk(k_X) \), \( pk_X = pk(k_X) \) and \( d = \text{dec}(x, sk_S) \).
Private Authentication: Unlinkability

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

\[
\nu k_S. \nu k_0, \ldots, k_N. \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \! \!
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**:

\[
(P(\vec{k}) | S(\vec{k})) \approx (P(\vec{k}_0) | S(\vec{k}_0)) \approx (P(\vec{k}_1) | S(\vec{k}_1))
\]
Authentication Protocols
The Hash-Lock Protocol

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

$T(A, i) : \nu \ n_{T,i}. \ in(c^T_{A,i}, x). \ out(c^T_{A,i}, \langle n_{T,i}, H(\langle x, n_{T,i}, k_A \rangle) \rangle)$

$R(j) : \nu \ n_{R,j}. \ in(c^R_1, \_). \ out(c^R_1, n_{R,j}). \ in(c^R_2, y).$

if $\forall A \in \mathcal{I} \ \pi_2(y) = H(\langle n_{R,j}, \pi_1(y) \rangle, k_A)$

then $out(c^R_2, ok)$

else $out(c^R_2, ko)$

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

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

Remark: we let the adversary do the scheduling between parties.
• we let $\leq$ be the **prefix relation** over observable traces:

$$tr_0 \leq tr_1 \text{ iff. } \exists tr'. \; tr_1 = tr_0 ; tr'$$

• $tr \bowtie c$ states that $tr$ **ends with an output** on $c$:

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

**Remark:** $tr \bowtie c \leq tr'$ denotes $tr \bowtie c \land tr \leq tr'$. 


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

$$tr \in \mathcal{T}_{\text{io}} \iff \forall tr' : c \leq tr. \exists tr''. tr' = tr''; \text{in}(c); \text{out}(c)$$

**Assumption: POR**

We admit that to analyze the Hash-Lock protocol, it is sufficient to consider only observables traces in $\mathcal{T}_{\text{io}}$. 
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.
For any $tr \diamond c_j^{R_2} \in \mathcal{T}_{io}$, we let $\text{accept}^{A}@tr$ be a term stating that the reader accepts the tag A at the end of the trace $tr$ (defined later).
Informally, **Hash-Lock** provides authentication if for all $\text{tr} \in T_{io}$, $\text{tr}_1 \diamond c_{j}^{R_1}$ and $\text{tr}_3 \diamond c_{j}^{R_2}$ such that:

$$\text{tr}_1 < \text{tr}_3 \leq \text{tr} \quad \text{and} \quad \text{accept}^{A@\text{tr}_3}$$

there must exist $\text{tr}_2 \diamond c_{A,i}^{T}$ such that $\text{tr}_1 \leq \text{tr}_2 \leq \text{tr}_3$ and:

$$\text{out}@\text{tr}_1 = \text{in}@\text{tr}_2 \land \text{out}@\text{tr}_2 = \text{in}@\text{tr}_3$$

Graphically:

\[
\begin{align*}
\text{out}@\text{tr}_1 &= \text{in}@\text{tr}_2 \\
\text{out}@\text{tr}_2 &= \text{in}@\text{tr}_3
\end{align*}
\]
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 $tr$ and observable $\alpha$, we let:

$$\text{pred}(tr; \alpha) \overset{\text{def}}{=} tr$$
Macro Terms

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

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

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

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

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

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

💡 The restriction to traces in $\mathcal{T}_{io}$ simplifies the definition of $\text{in}_\mathcal{P}@tr$. 110
frame_P@tr contains all the information known to an adversary against P after the execution of tr.

More precisely, we can show that for all action-deterministic processes P and Q, for all tr ∈ T_io:

\[ M \models \text{fold}(P, tr) \sim \text{fold}(Q, tr) \iff M \models \text{frame}_P@tr \sim \text{frame}_Q@tr \]

for any M satisfying:

\[ \pi_1(x, y) \triangleright x \sim \text{true} \quad \pi_2(x, y) \triangleright y \sim \text{true} \]

**Proof**

⇒ apply FA to build \( \text{frame}_R@tr \) from \( \text{fold}(R, tr) \) for \( R \in \{P, Q\} \)

⇐ apply FA + DUP + the pair injectivity rules to compute all terms in \( \text{fold}(R, tr) \) from \( \text{frame}_R@tr \) for \( R \in \{P, Q\} \)
Hash-Lock: Accept

\[ T(A, i) : \nu n_{T,i}. \text{in}(c_{A,i}^T, x). \text{out}(c_{A,i}^T, \langle n_{T,i}, H(\langle x, n_{T,i} \rangle, k_A) \rangle) \]

\[ R(j) : \nu n_{R,j}. \text{in}(c_{j}^{R1}, _\_). \text{out}(c_{j}^{R1}, n_{R,j}). \text{in}(c_{j}^{R2}, y). \]

\[ \text{if } \bigvee_{A \in I} \pi_2(y) \equiv H(\langle n_{R,j}, \pi_1(y) \rangle, k_A) \]

\[ \text{then } \text{out}(c_{j}^{R2}, \text{ok}) \]

\[ \text{else } \text{out}(c_{j}^{R2}, \text{ko}) \]

To be able to state some authentication property of Hash-Lock, we need an additional macro. For all \( \text{tr} \diamond c_{j}^{R2} \in \mathcal{T}_{io} \), we let:

\[ \text{accept}_{\text{tr}}^{A} \overset{\text{def}}{=} \pi_2(\text{in}_{\text{tr}}) \equiv H(\langle n_{R,j}, \pi_1(\text{in}_{\text{tr}}) \rangle, k_A) \]

💡 We made sure that all names in the protocol are unique, so that they don’t have to be renamed during the folding.
The following formulas encode the fact that the **Hash-Lock** protocol provides **authentication**:

\[
\forall A \in \mathcal{I}. \forall \text{tr} \in \mathcal{T}_{io}. \forall \text{tr}_1 \diamond c_{j}^{R_1}, \text{tr}_3 \diamond c_{j}^{R_2} \text{ s.t. } \text{tr}_1 < \text{tr}_3 \leq \text{tr},
\]

\[
\text{accept}^A @ \text{tr}_3 \rightarrow \bigvee \begin{array}{l}
\text{out} @ \text{tr}_1 = \text{in} @ \text{tr}_2 \wedge \\
\text{out} @ \text{tr}_2 = \text{in} @ \text{tr}_3 \\
\text{tr}_2 \diamond c_{A,i}^T \\
\text{tr}_1 \leq \text{tr}_2 \leq \text{tr}_3
\end{array}
\]

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
A reachability judgement $\Gamma \vdash t$ comprises a sequence of terms $\Gamma = t_1 \rightarrow \ldots t_n$ and a (boolean) term $t$.

$\Gamma \vdash t$ is valid if and only if:

$$t_1 \rightarrow \ldots t_{n-1} \rightarrow t_n \rightarrow t \sim true$$

is valid.
Our reachability judgements can be trivially equipped with a sound sequent calculus.

\[
\begin{align*}
\Gamma, t_\phi \vdash t_\phi & \quad & \Gamma \vdash t_\phi \\
\Gamma \vdash t_\psi & \quad & \Gamma, t_\psi \vdash t_\phi \\
\Gamma \vdash t_\phi & \quad & \Gamma\vdash t_\phi \\
\Gamma \vdash t_\psi & \quad & \Gamma \vdash t_\phi \\
\Gamma \vdash t_\psi \land t_\phi & \quad & \Gamma \vdash t_\psi \land t_\phi \vdash t_\theta \\
\Gamma \vdash t_\psi \lor t_\phi & \quad & \Gamma \vdash t_\psi \lor t_\phi \\
\Gamma, t_\psi \land t_\phi \vdash t_\theta & \quad & \Gamma, t_\psi \lor t_\phi \vdash t_\theta \\
\Gamma, t_\psi \vdash t_\theta & \quad & \Gamma, t_\psi \lor t_\phi \vdash t_\theta \\
\Gamma, t_\psi \vdash t_\theta & \quad & \Gamma, t_\psi \lor t_\phi \vdash t_\theta
\end{align*}
\]
\[
\Gamma, t_\phi \vdash \bot \\
\Gamma \vdash \neg t_\phi \\
\Gamma_1, t_\phi, t_\psi, \Gamma_2 \vdash t_\theta \\
\Gamma_1, t_\psi, t_\phi, \Gamma_2 \vdash t_\theta \\
\Gamma, t_\psi, t_\psi \vdash t_\phi \\
\Gamma, t_\psi \vdash t_\phi
\]
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 \(tr \in \mathcal{T}_{io}\), \(tr_1 \diamond c_{R1}^j\) and \(tr_3 \diamond c_{R2}^j\) s.t.:

\[
tr_1 < tr_3 \leq tr
\]

the following formula is valid:

\[
\text{accept}^A_{tr_3} \rightarrow \bigvee_{tr_2 \diamond c_{T,A,1}^j, \text{tr}_1 \leq \text{tr}_2 \leq \text{tr}_3} \text{out}_{tr_1} = \text{in}_{tr_2} \wedge \text{out}_{tr_2} = \text{in}_{tr_3} \sim \text{true}
\]

\(^6\)Taking \(\text{verify}_k(s, m) \defeq s = H(m, k)\).
**Proof.** Let \( a \in I \), and let \( t_r \in T_{io} \), \( t_{r_1} \cdot c_j^{R_1} \) and \( t_{r_3} \cdot c_j^{R_2} \) be s.t.:

\[
tr_1 < tr_3 \leq tr
\]

We let:

\[
t_{conc} \overset{\text{def}}{=} \bigvee_{tr_2 \cdot c_{A,i}^{T} \overset{tr_1 \leq tr_2 \leq tr_3}{} \in \cdot c_{A,i}^{T}} \text{out}@tr_1 = \text{in}@tr_2 \land \text{out}@tr_2 = \text{in}@tr_3
\]

We must prove that the following reachability judgement is valid:

\[
\text{accept}^{A}@tr_3 \vdash t_{conc}
\]

i.e. that:

\[
\pi_2(\text{in}@tr_3) \equiv H(\langle n_{R,j}, \pi_1(\text{in}@tr_3) \rangle, k_A) \vdash t_{conc}
\]
We use the **EUF-MAC** rule on the equality:

\[ \pi_2(\text{in} \oplus \text{tr}_3) \doteq H(\langle n_{R,j} , \pi_1(\text{in} \oplus \text{tr}_3) \rangle, k_A) \]  

(†)

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:

\[ \{ H(\langle \text{in} \oplus \text{tr}_2 , n_{T,i} \rangle, k_A) \mid \text{tr}_2 \bowtie c_{A,i}^{T} < \text{tr}_3 \} \]

💡 The hashes in the reader’s outputs can be seen as verify checks, and can therefore be ignored.
Hence using **EUF-MAC** plus some basic reasoning, we have:

\[
\text{accept}^A_{@tr_3}, \langle \text{in}@tr_2, n_{T,i} \rangle \vdash t_{\text{conc}} \quad \text{for every } tr_2 \diamond c_{A,i}^T < tr_3
\]

\[
\text{accept}^A_{@tr_3}, \bigvee_{tr_2 \diamond c_{A,i}^T < tr_3} \langle \text{in}@tr_2, n_{T,i} \rangle \vdash t_{\text{conc}}
\]

\[
\text{accept}^A_{@tr_3} \vdash t_{\text{conc}}
\]
Assuming that the pair and projections satisfy:

\[(\pi_1\langle x, y \rangle \doteq x) \sim true\]  \[\quad (\pi_2\langle x, y \rangle \doteq y) \sim true\]

We only have to show that for every \(tr_2 \diamond c_{A,i}^T < tr_3\):

\[\Gamma \vdash t_{conc}\]

is valid, where:

\[\Gamma \overset{\text{def}}{=} \text{accept}^A@tr_3, \text{in}@tr_2 \doteq n_{R,j}, \ n_{T,i} \doteq \pi_1(\text{in}@tr_3)\]
Authentication: Hash-Lock

Since $tr_1 \diamond c_j^{R_1} < tr_3$ we know that:

$$\text{out}@tr_1 \overset{\text{def}}{=} n_{R,j}$$

Moreover:

$$\text{out}@tr_2 \overset{\text{def}}{=} \langle n_{T,i} , H(\langle \text{in}@tr_2 , n_{T,i} \rangle , k_A) \rangle$$

Hence:

$$\Gamma \vdash \pi_1(\text{out}@tr_2) \overset{\text{def}}{=} \pi_1(\text{in}@tr_3) \quad \text{(◊)}$$

Similarly:

$$\Gamma \vdash \pi_2(\text{out}@tr_2) \overset{\text{def}}{=} H(\langle \text{in}@tr_2 , n_{T,i} \rangle , k_A)$$
$$\overset{\text{def}}{=} H(\langle n_{R,j} , \pi_1(\text{in}@tr_3) \rangle , k_A)$$
$$\overset{\text{def}}{=} \pi_2(\text{in}@tr_3)$$

Consequently:

$$\Gamma \vdash \pi_2(\text{out}@tr_2) \overset{\text{def}}{=} \pi_2(\text{in}@tr_3) \quad \text{(*)}$$
Authentication: Hash-Lock

Assuming that the pair and projections satisfy the property:

\[ \pi_1 x \Downarrow= \pi_1 y \Rightarrow \pi_2 x \Downarrow= \pi_2 y \Rightarrow x \Downarrow= y \]

We deduce from (⋆) and (⋄) that:

\[ \Gamma \vdash \text{out}@\text{tr}_2 \Downarrow= \text{in}@\text{tr}_3 \]

Putting everything together, we get:

\[ \Gamma \vdash \text{out}@\text{tr}_1 \Downarrow= \text{in}@\text{tr}_2 \wedge \text{out}@\text{tr}_2 \Downarrow= \text{in}@\text{tr}_3 \quad (‡) \]
Recall that:

$$t_{\text{conc}} \overset{\text{def}}{=} \bigvee_{\text{tr}_2 \circ c_{A,i}^{T}} \text{out} @ \text{tr}_1 = \text{in} @ \text{tr}_2 \land \text{out} @ \text{tr}_2 = \text{in} @ \text{tr}_3$$

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

$$\Gamma, \text{out} @ \text{tr}_1 = \text{in} @ \text{tr}_2, \text{out} @ \text{tr}_2 = \text{in} @ \text{tr}_3 \vdash \bot$$

This follows from the independence rule:

$$\frac{(t \models n) = \text{false}}{-\text{IND}} \quad \text{when } t \text{ is ground and } n \notin \text{st}(t)$$

using the fact that:

$$\text{out} @ \text{tr}_1 \overset{\text{def}}{=} n_{R,j}$$

and that if $\text{tr}_2 < \text{tr}_1$ then $n_{R,j} \notin \text{st}(\text{in} @ \text{tr}_2)$. 
Authentication Protocols

Beyond Authentication
**Authentication**, which states that we must have:

\[ \forall \text{tr}_R \diamond c_R. \exists \text{tr}_T \diamond c_T. \]

does not exclude the scenario:

\[ \text{tr}_T \diamond c_T \quad \text{tr}_R \diamond c_R \]

\[ \text{accept}@\text{tr}_R \]

\[ \text{tr}_T \diamond c_T \quad \text{tr}_R \diamond c_1^R \quad \text{tr}_R \diamond c_2^R \]

\[ \text{accept}@\text{tr}_R \quad \text{accept}@\text{tr}_R' \]
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:

\[
T(A, i) : \nu n_{T,i}. \text{out}(c_{A,i}^T, \langle n_{T,i}, H(n_{T,i}, k_A) \rangle)
\]

\[
R(j) : \text{in}(c_j^{R_2}, y). \text{if } \forall A \in \mathcal{I} \pi_2(y) = H(\pi_1(y), k_A)
\]

then **out**(\(c_j^{R_2}, \text{ok}\))

else **out**(\(c_j^{R_2}, \text{ko}\))
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*:

\[ \forall A \in I. \forall tr \in T_{io}. \forall tr_1 \diamond c_j^{R_1}, tr_3 \diamond c_j^{R_2} \text{ s.t. } tr_1 < tr_3 \leq tr \]

\[
\begin{align*}
\text{accept}^A@tr_3 \rightarrow \bigvee_{tr_2 \vdash c_{A,i}^T} \quad \text{out}@tr_1 = \text{in}@tr_2 \land \\
&\text{out}@tr_2 = \text{in}@tr_3 \\
&\bigwedge_{tr'_1 \vdash c_k^{R_1}, tr'_3 \vdash c_k^{R_2}} \bigwedge_{tr'_1 < tr'_3 \leq tr} \\
&\left( \text{accept}^A@tr'_3 \land \\
&\text{out}@tr_2 = \text{in}@tr'_3 \rightarrow j = k \right)
\end{align*}
\]
Partial order reduction for security protocols. 

A computationally complete symbolic attacker for equivalence properties.  