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 \(<_1, _2>\)) 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. $A \rightarrow B : \nu n_A$.  
   \textbf{out}(c_A, \{\langle pk_A, n_A \rangle \}_B)

2. $B \rightarrow A : \nu n_B$. \textbf{in}(c_A, x). \textbf{out}(c_B, \{\langle \pi_2(\text{dec}(x, sk_A)), n_B \rangle \}_A)

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

\textit{Notation: we use }\equiv\textit{ to denote} \textbf{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.:

\[
A, B, \langle \_, \_ \rangle, \pi_1(\_), \pi_2(\_), \{\_\}_{-}, \text{pk}(\_), \text{sk}(\_), \\
\text{if}_\_\text{then}_\_\text{else}_\_, \_ \equiv \_, \_ \land \_, \_ \lor \_, \_ \rightarrow \_
\]

**Examples**

<table>
<thead>
<tr>
<th>(\text{pk}(k_A))</th>
<th>({\langle \text{pk}<em>A, n_A \rangle }</em>{\text{pk}_B})</th>
<th>(\pi_1(n_A))</th>
</tr>
</thead>
</table>

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

\[
\text{type}(f) = (\tau_1 \star \cdots \star \tau_n) \rightarrow \tau
\]

For now, we use the **message** and **bool** types. We require that terms are well-typed.
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: $A \rightarrow B: \nu n_A. \begin{align*} & \text{out}(c_A, \{\langle pk_A, n_A \rangle\}_{pk_B}) \end{align*}$

2: $B \rightarrow A: \nu n_B. \begin{align*} & \text{in}(c_A, x). \text{out}(c_B, \{\langle \pi_2(\text{dec}(\text{att}(t_1), sk_A)), n_B \rangle\}_{pk_A}) \end{align*}$

How do we represent the adversary’s inputs?

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

Example: Terms for PA, v1

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

$t_2 \equiv \{\langle \pi_2(\text{dec}(\text{att}(t_1), sk_A)), n_B \rangle\}_{pk_A}$
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{1cm} \text{out}(c_A, \{\langle pk_A, n_A \rangle \}_{pk_B})

2: \( B \rightarrow A : \nu n_B. \text{in}(c_A, x). \) if \( \pi_1(d) \equiv pk_A \)

\hspace{1cm} \text{then out}(c_B, \{\langle \pi_2(d), n_B \rangle \}_{pk_A})

\hspace{1cm} \text{else out}(c_B, \{0\}_{pk_A})

where \( d \equiv \text{dec}(x, sk_A). \)

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

The PA Protocol, v2

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

2: $B \rightarrow A : \nu n_B. \text{in}(c_A, x)$. if $\pi_1(d) \Downarrow pk_A$
then $\text{out}(c_B, \{\langle \pi_2(d), n_B \rangle \}_{pk_A})$
else $\text{out}(c_B, \{0\}_{pk_A})$

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.
Example: Terms for PA, v2

\[ t_1 \equiv \{ \langle \text{pk}_A, n_A \rangle \}_{\text{pk}_B} \]
\[ t_2 \equiv \text{if } \pi_1(d_1) \doteq \text{pk}_A \]
\[ \text{then } \{ \langle \pi_2(d_1), n_B \rangle \}_{\text{pk}_A} \]
\[ \text{else } \{ 0 \}_{\text{pk}_A} \]

where \( d_1 \equiv \text{dec(} \text{att}(t_1), \text{sk}_A \text{).} \)
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:

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

- 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.

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

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

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

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

- $\Phi$ is a sequence of terms (in $T(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})$).
- $\sigma$ 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:

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

Rule for new:

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

if $n_f$ does not appear in the lhs configuration

\textbf{-irreducibility}

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

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

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

if \(x \not\in \text{dom}(\sigma)\), the lhs folding configuration is \(\mapsto\)-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:

\[\Phi, \bigvee_{1 \leq i \leq n} b_i\]
Folding: Output Rule

Rule for outputs:

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

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

if the lhs folding configuration is \(\leftrightarrow\)-irreducible and if for every \(i\), \(\Pi_1\) does not start by an output on \(c\) and:

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

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

Remark: we omit the error message when \((\bigvee_{1 \leq i \leq n} b_i) \Leftrightarrow \text{true}\).
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:

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

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?

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

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

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

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

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

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.
We interpret $t \in T(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X})$ 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 $\llbracket t \rrbracket^\sigma_M$ is parameterized by:

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

We often omit $M$, as it is fixed throughout the interpretation.

We now define the machine $\llbracket t \rrbracket^\sigma \in D$, 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 \([f]_\mathcal{M}\):

\[
[f(t_1, \ldots, t_n)]^\sigma(1^n, \rho) \overset{\text{def}}{=} [f]_\mathcal{M}([t_1]^\sigma(1^n, \rho), \ldots, [t_n]^\sigma(1^n, \rho))
\]

**Adversarial function symbols** \( g \in \mathcal{G} \) also have access to \( \rho_a \):

\[
[g(t_1, \ldots, t_n)]^\sigma(1^n, \rho) \overset{\text{def}}{=} [g]_\mathcal{M}([t_1]^\sigma(1^n, \rho), \ldots, [t_n]^\sigma(1^n, \rho), \rho_a)
\]

**Remark:** \([f]_\mathcal{M}\) and \([g]_\mathcal{M}\) are **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 \( \rho_p \).

\( \Rightarrow \) Hence different names are independent random samplings.
We **force** the interpretation of some function symbols.

- if _then _ else _ is interpreted as **branching**:

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

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

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

Similarly, we force the interpretations of \(\land, \lor, \rightarrow\), true, false.
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.

$$\phi := \top | \bot | \phi \land \phi | \phi \lor \phi | \phi \rightarrow \phi | \neg \phi$$

$$| \forall x. \phi | \exists x. \phi$$

$$(x \in \mathcal{X})$$

$$| t_1, \ldots, t_n \sim_n t_{n+1}, \ldots, t_{2n} \quad (t_1, \ldots, t_{2n} \in T(\mathcal{F}, \mathcal{G}, \mathcal{N}, \mathcal{X}))$$

**Remark:** we use $\hat{\land}, \hat{\lor}, \hat{\rightarrow}$ 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.:

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

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

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

\[
[t_1, \ldots, t_n \sim_n s_1, \ldots, s_n]_\mathcal{M} = \text{True}
\]

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

\[
\begin{align*}
\left| \Pr_{\rho} (A(1^n, ([t_i]_\mathcal{M}(1^n, \rho))_{1 \leq i \leq n}, \rho_a) = 1) 
- \Pr_{\rho} (A(1^n, ([s_i]_\mathcal{M}(1^n, \rho))_{1 \leq i \leq n}, \rho_a) = 1) \right| \quad \text{(\(*\))}
\end{align*}
\]

is a \textbf{negligible} function of \( \eta \).

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

$$f(\eta) \overset{\text{def}}{=} \Pr_\rho\left(\llbracket n_0 \rrbracket(1^n, \rho) = \llbracket n_1 \rrbracket(1^n, \rho)\right)$$

If $n_0 \not\equiv n_1$, 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?

true $\sim$ false  
$n_0 \sim n_0$  
$n_0 \sim n_1$  
$n_0 \div n_1 \sim$ false

$n_0, n_0 \sim n_0, n_1$  
$f(n_0) \sim f(n_1)$ where $f \in \mathcal{F} \cup \mathcal{G}$

$\pi_1(\langle n_0 , n_1 \rangle) \vdash n_0 \sim$ true
Exercise

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

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

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

\[ \not\models \pi_1(\langle n_0, n_1 \rangle) \equiv n_0 \sim \text{true} \]
$\mathcal{P}$ and $\mathcal{Q}$ are **indistinguishable**, written $\mathcal{P} \simeq \mathcal{Q}$, 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**:\(^2\)

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

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

- $\text{out}(c, t_1) \approx \text{out}(c, t_2)$
- $\text{out}(c, t) \approx \text{null}$
- $\text{in}(c, x) \approx \text{null}$

- $\text{out}(c, t) \approx \text{if } b \text{ then } \text{out}(c, t_1) \text{ else } \text{out}(c, t_2)$
- $\text{out}(c, t) \approx \text{if } b \text{ then } \text{out}(c, t) \text{ else } \text{out}(c_0, t_0)$
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.
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:

\[
\begin{align*}
\vec{u}_{\pi(1)}, \ldots, \vec{u}_{\pi(n)} & \sim \vec{v}_{\pi(1)}, \ldots, \vec{v}_{\pi(n)} \\
\vec{u}_1, \ldots, \vec{u}_n & \sim \vec{v}_1, \ldots, \vec{v}_n
\end{align*}
\]
Alpha-renaming.

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

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

Restriction. The adversary can throw away some values:

\[ \overrightarrow{u}, s \sim \overrightarrow{v}, t \]

\[ \frac{\overrightarrow{u} \sim \overrightarrow{v}}{\overrightarrow{u} \sim \overrightarrow{v}} \quad \text{RESTR} \]
**Duplication.** Giving twice the same value to the adversary is useless:

\[
\tilde{u}, s \sim \tilde{v}, t \\
\tilde{u}, s, s \sim \tilde{v}, t, t
\]

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

\[
\tilde{u}_1, \tilde{v}_1 \sim \tilde{u}_1, \tilde{v}_2 \\
f(\tilde{u}_1), \tilde{v}_1 \sim f(\tilde{u}_2), \tilde{v}_2
\]

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

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

\text{FA}

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

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

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

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

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

\[
\begin{align*}
\mathcal{B}(1^n, [\vec{u}_i]_{\mathcal{M}}(1^n, \rho), [\vec{v}_i]_{\mathcal{M}}(1^n, \rho), \rho_a) \\
= & \mathcal{A}(1^n, [f(\vec{u}_i)]_{\mathcal{M}}(1^n, \rho), [\vec{v}_i]_{\mathcal{M}}(1^n, \rho), \rho_a)
\end{align*}
\]

(\( i \in \{1, 2\} \))

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

\[
\frac{\vec{w}_1, b_0, u_0 \sim \vec{w}_1, b_1, u_1 \quad \vec{w}_0, b_0, v_0 \sim \vec{w}_1, b_1, v_1}{\vec{w}_0, \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \vec{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1}
\]

CS
Structural Rules: Proof of Case Study

\[
\begin{align*}
&b_0, u_0 \sim b_1, u_1, \quad b_0, v_0 \sim b_1, v_1, \\
&t_0 \equiv \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim t_1 \equiv \text{if } b_1 \text{ then } u_1 \text{ else } v_1 \quad \text{CS}
\end{align*}
\]

**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 \( i \in \{1, 2\} \):

\[
\Pr_\rho \left( \mathcal{B}_\top(1^n, [b_i]_\mathcal{M}(1^n, \rho), [u_i]_\mathcal{M}(1^n, \rho), \rho_a) = 1 \right)
= \Pr_\rho \left( \mathcal{A}(1^n, [t_i]_\mathcal{M}(1^n, \rho), \rho_a) = 1 \wedge [b_i]_\mathcal{M}(1^n, \rho) = 1 \right) \quad p_{\top,i}
\]
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 $B_{\perp}$ be the distinguisher:

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

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

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

$$
| (p^\top,1 + p^\bot,1) - (p^\top,0 + p^\bot,1) | \leq |p^\top,1 - p^\top,0| + |p^\bot,1 - p^\bot,1|
$$

Since $\mathcal{A}$’s advantage is non-negligible, at least one of the two quantity above is non-negligible. Hence either $B^\top$ or $B^\bot$ has a non-negligible advantage against a premise of the $\text{CS}$ rule. $\square$. 
Remark that \( b \) is necessary in CS

\[
\frac{\vec{w}_1, b_0, u_0 \sim \vec{w}_1, b_1, u_1 \quad \vec{w}_0, b_0, v_0 \sim \vec{w}_1, b_1, v_1}{\vec{w}_0, \text{if } b_0 \text{ then } u_0 \text{ else } v_0 \sim \vec{w}_1, \text{if } b_1 \text{ then } u_1 \text{ else } v_1}
\]

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

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

$$\begin{align*}
\vec{u}, t & \sim \vec{v} \\
\frac{s = t}{\vec{u}, s \sim \vec{v}} \quad & \text{R}
\end{align*}$$
Structural Rules: FO + Equality Reasoning

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

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

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

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

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

**Example**

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

Two rules exploiting the independence of bitstring distributions:

\[
\begin{align*}
(t \doteq n) &= \text{false} \quad \Rightarrow \text{IND} \quad \text{when } n \not\in \text{st}(t) \\
\vec{u} &\sim \vec{v} \quad \text{FRESH} \quad \text{when } n_0 \not\in \text{st}(\vec{u}) \text{ and } n_1 \not\in \text{st}(\vec{v})
\end{align*}
\]

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:

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

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

\[
\begin{align*}
\Pr_{\rho}([t \vdash n](1^n, \rho) = 1) & = \Pr_{\rho}([t](1^n, \rho) = [n](1^n, \rho)) \\
& = \sum_{w \in \{0,1\}^*} \Pr_{\rho}([t](1^n, \rho) = w \land [n](1^n, \rho) = w) \\
& = \frac{1}{2^n} \cdot \sum_{w \in \{0,1\}^*} \Pr_{\rho}([t](1^n, \rho) = w) \\
& = \frac{1}{2^n}
\end{align*}
\]

\(\square\)
Exercise

Give a derivation of the following formula:

\[ n_0 \sim \text{if } b \text{ then } n_0 \text{ else } n_1 \]  
\[ (\text{when } n_0, n_1 \not\in \text{st}(b)) \]
Implementation Rules
A rule is \textit{C-sound} if \( \phi \) is \textit{C-valid} whenever \( \phi_1, \ldots, \phi_n \) are \textit{C-valid}.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
\multicolumn{1}{|c|}{Example} \\
\hline
\begin{array}{c}
(\pi_1\langle x, y \rangle \div x) \sim \text{true}
\end{array} \\
\hline
\end{tabular}
\end{table}

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

Obviously, it is \( C_{\pi} \)-sound, where \( C_{\pi} \) 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:

- $\pi_i(\langle x_1, x_2 \rangle) = x_i$ for $i \in \{1, 2\}$
- $\text{dec}(\{x\}_z^{pk(y)}, sk(y)) = x$
- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
- ...
Cryptographic Rules
Cryptographic reductions are the main tool used in proofs of computational security.

**Cryptographic Reduction** $S \leq_{\text{red}} H$

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

- We assume that $H$ 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 $S \leq_{\text{red}} \mathcal{H}$

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

$$\forall A. \exists B. \text{Adv}^\eta_S(A) \leq P(\text{Adv}^\eta_H(B), \eta)$$

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 satisfy the functional equality:

$$\text{dec}(\{x\}_z^{pk(y)}, sk(y)) = x$$

---

\(^3\)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 $\mathcal{O}^{b,n}_{LR}(\cdot, \cdot)$:

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

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

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

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

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

59
Exercise

Show that if the encryption ignore its randomness, i.e. there exists $a_{enc}(\_,\_)$ s.t. for all $x, y, r$:

$$\{x\}^r_y = a_{enc}(x, y)$$

then the encryption does not satisfy IND-CCA$_1$. 
Indistinguishability Against Chosen Ciphertexts Attacks

If the encryption scheme is IND-CCA\textsubscript{1}, then the *ground* rule:

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

is sound, when:

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

\[
\left| \Pr_{\rho} (\mathcal{A}(1^n, \lceil \vec{u} \rceil, \mathcal{M}(1^n, \rho), \{\{ t_0 \}^{r}_{\text{pk}(n)} \}) \mathcal{M}(1^n, \rho), \rho_a) - \Pr_{\rho} (\mathcal{A}(1^n, \lceil \vec{u} \rceil, \mathcal{M}(1^n, \rho), \{\{ t_1 \}^{r}_{\text{pk}(n)} \}) \mathcal{M}(1^n, \rho), \rho_a) \right|
\]

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 IND-CCA_1 security game.
IND-CCA₁ Rule: Proof

Let \( B^{O_{LR}^b, n}, O_{\text{dec}}^n (1^n, [\text{pk}(n)] \mathcal{M}(1^n, \rho)) \) 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) compute\(^4^\):

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

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

iii) compute:

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

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

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

\(^4^\)we describe how later
Then $\mathcal{B}$ advantage against IND-CCA$_1$ is exactly $\mathcal{A}$ 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 \textbf{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 \( w_s := [s]_\mathcal{M}(1^n, \rho) \) in polynomial time.

Assuming this holds, we conclude by observing that \textbf{IND-CCA}_1 side conditions guarantees that \( \vec{u}, t_0, t_1 \) are not forbidden subterms.
**IND-CCA₁ 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.
- $s \in \{r, n\}$ 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 $\text{IND-CCA}_1$ 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 $\mathcal{B}$ receives $\llbracket \text{pk}(n) \rrbracket_\mathcal{M}(1^n, \rho)$ as argument.

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

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

$$w_s := O_{\text{dec}}^n(w_m)$$
Exercise

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

\begin{align*}
\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)}
\end{align*}
Exercise (Hybrid Argument)

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

\[
\{0\}_0^{r_0}_{pk(n)}, \{1\}_1^{r_1}_{pk(n)}, \ldots, \{n\}_n^{r_n}_{pk(n)} \sim \{0\}_0^{r_0}_{pk(n)}, \{0\}_1^{r_1}_{pk(n)}, \ldots, \{0\}_n^{r_n}_{pk(n)}
\]

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)
A scheme provides **key privacy against chosen cipher-text attacks** (KP-CCA\textsubscript{1}) iff for every PPTM \( \mathcal{A} \) with access to:

- a left-right encryption oracle \( \mathcal{O}_{LR}^{b,n_0,n_1}(.) \):
  \[
  \mathcal{O}_{LR}^{b,n_0,n_1}(m) \overset{\text{def}}{=} \{ m \}^r_{pk(n_b)} \quad (r \text{ fresh})
  \]

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

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

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

is negligible in \( \eta \), where \( n_0, n_1 \) are drawn in \( \{0, 1\}^\eta \).
Exercise

Show that \( \text{IND-CCA}_1 \nRightarrow \text{KP-CCA}_1 \) and \( \text{KP-CCA}_1 \nRightarrow \text{IND-CCA}_1 \).
Key Privacy Against Chosen Ciphertexts Attacks

If the encryption scheme is KP-CCA₁, then the ground rule:

\[ \vec{u}, \{t\}_r \sim \vec{u}, \{t\}_r \]

is sound, when:

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

The proof is similar to the IND-CCA₁ soundness proof. We omit it.
Security Proof
Private Authentication: Anonymity

Let's 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 $I_A \parallel S_A$ from $I_C \parallel S_A$.

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

$S_X: \nu r_0. \nu n_S. \text{in}(c_I, x). \quad \text{if } \pi_1(d) = pk_X$
then $\text{out}(c_S, \{\langle \pi_2(d), n_S \rangle\}^{ro}_{pk_X})$
else $\text{out}(c_S, \{0\}^{ro}_{pk_X})$

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:

\[
\not\equiv \{\langle n_I, n_S \rangle\}_{pk_A}^{ro} \sim \{0\}_{pk_C}^{ro}
\]

even if the encryption is IND-CCA\(_1\) and KP-CCA\(_1\).
We fix the protocol by:

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

The PA Protocol, v3

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

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

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

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

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

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

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

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

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

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

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

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

I_X : ν r. ν n_I.

out(c_I, \{⟨pk_X, n_I⟩\}^r_{pk_S})

S_X : ν r_0. ν n_S. in(c_I, x).

if \pi_1(d) \equiv pk_X \land \text{len}(\pi_2(d)) \equiv \text{len}(n_S)

then out(c_S, \{⟨\pi_2(d), n_S⟩\}^{r_0}_{pk_X})

else out(c_S, \{⟨n_S, n_S⟩\}^{r_0}_{pk_X})

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

\begin{align*}
\text{in}(c_I), \text{out}(c_I), \text{out}(c_S) &\quad \text{in}(c_I), \text{out}(c_S), \text{out}(c_I) \\
\text{out}(c_I), \text{in}(c_I), \text{out}(c_S) &\quad \text{out}(c_I), \text{out}(c_S), \text{in}(c_I) \\
\text{out}(c_S), \text{in}(c_I), \text{out}(c_I) &\quad \text{out}(c_S), \text{out}(c_S), \text{in}(c_I)
\end{align*}

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:

\[ \text{out}_1^A, \text{out}_{2, A}^A[\text{out}_1^A] \sim \text{out}_1^C, \text{out}_{2, A}^A[\text{out}_1^C] \]

where:

\[ \text{out}_1^X \equiv \{\langle \text{pk}_X, n_I \rangle \}_p^r \]

\[ \text{out}_{2, Y}^X [M] \equiv \text{if } \pi_1(d[M]) \vdash \text{pk}_X \land \text{len}(\pi_2(d[M])) \vdash \text{len}(n_S) \]

\[ \text{then } \{\langle \pi_2(d[M]), n_S \rangle \}_p^r \]

\[ \text{else } \{\langle n_S, n_S \rangle \}_p^r \]

\[ d[M] \equiv \text{dec(\text{att}_0([M]), sk_S)} \]
Private Authentication: Anonymity

First, we push the branching under the encryption:

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

where:

\[
\text{out}_2^{X,Y}[M] \equiv \begin{cases} 
\text{if } \pi_1(d[M]) \vdash \text{pk}_X \wedge \text{len}(\pi_2(d[M])) \vdash \text{len}(n_S) & \text{then } \langle \pi_2(d[M]), n_S \rangle \\
\text{else } \langle n_S, n_S \rangle & \end{cases}
\]

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

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

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

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:

\[
\text{out}_{1}^A, \text{out}_{2}^A, \text{out}_{1}^A \sim \text{out}_{1}^C, \text{out}_{2}^C, \text{out}_{1}^C
\]

\[
\text{out}_{1}^A, \text{out}_{2}^A, \text{out}_{1}^A \sim \text{out}_{1}^C, \text{out}_{2}^C, \text{out}_{1}^C
\]

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:

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

then \( \langle \pi_2(d[M]), n_S \rangle \)

else \( \langle n_S, n_S \rangle \)

Then:

\[
\frac{\text{len}(m_C[\text{out}_1^C]) = \text{len}(m_A[\text{out}_1^A])}{\text{len}(m_C[\text{out}_1^C]) = \text{len}(m_A[\text{out}_1^A])} \quad \text{FO}
\]

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

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

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.

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

$\alpha$-EQU + R + 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?

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

$$\begin{align*}
R & : \nu n_R. \quad \text{out}(c_R, n_R) \\
T_X & : \nu n_T. \text{in}(c_R, x). \text{out}(c_I, \langle X \oplus n_T, n_T \oplus H(x, k_X) \rangle)
\end{align*}$$

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

---

85
We just saw an attack against:

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

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

**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 word**.
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:

$$\nu \vec{k}_S. \nu \vec{k}_0, \ldots, \vec{k}_N. \mid_{id \leq N} \mid_{sid \leq M} (P(\vec{k}_S, \vec{k}_{id}) \mid S(\vec{k}_S, \vec{k}_{id}))$$

$$\approx \nu \vec{k}_S. \nu \vec{k}_0,0, \ldots, \vec{k}_{N,M}. \mid_{id \leq N} \mid_{sid \leq M} (P(\vec{k}_S, \vec{k}_{id_{sid}}) \mid S(\vec{k}_S, \vec{k}_{id_{sid}}))$$
**Generic server**, accepting all identities.
No changes for the user process $P(\mathbf{k}_S, \mathbf{k}_U)$.
The server $S(\mathbf{k}_S, \mathbf{k}_{U_1}, \ldots, \mathbf{k}_{U_M})$ is parameterized by:

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

The we require that:

$$\nu \mathbf{k}_S, \nu \mathbf{k}_0, \ldots, \mathbf{k}_N. (\!_\mathrm{id} \leq N \!_\mathrm{sid} \leq M P(\mathbf{k}_S, \mathbf{k}_{\mathrm{id}}) | (\!_\leq L S(\mathbf{k}_S, \mathbf{k}_0, \ldots, \mathbf{k}_N))$$

$$\approx \nu \mathbf{k}_S, \nu \mathbf{k}_{0,0}, \ldots, \mathbf{k}_{N,M}. (\!_\mathrm{id} \leq N \!_\mathrm{sid} \leq M P(\mathbf{k}_S, \mathbf{k}_{\mathrm{id},\mathrm{sid}}) | (\!_\leq L S(\mathbf{k}_S, \mathbf{k}_{0,0}, \ldots, \mathbf{k}_{N,M}))$$
Private Authentication: Unlinkability

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

\[
I(\kappa_S, k_X) : \nu r. \nu n_I. \quad \text{out}(c_I, \{\langle \text{pk}_X, n_I \rangle \}_{\text{pk}_S})
\]

\[
S(\kappa_S, k_X) : \nu r_0. \nu n_S. \text{in}(c_I, x). \text{if } \pi_1(d) \equiv \text{pk}_X \land \text{len}(\pi_2(d)) \equiv \text{len}(n_S)
\]

\[
\text{then out}(c_S, \{\langle \pi_2(d), n_S \rangle \}_{\text{pk}_X})
\]

\[
\text{else out}(c_S, \{\langle n_S, n_S \rangle \}_{\text{pk}_X})
\]

where \( \text{sk}_X \equiv \text{sk}(k_X) \), \( \text{pk}_X \equiv \text{pk}(k_X) \) and \( d \equiv \text{dec}(x, \text{sk}_S) \).
**Theorem**

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

$$\nu k_S. \nu k_0, \ldots, k_N. \; !id \leq N \; !sid \leq M \; (I(k_S, k_id) \; | \; S(k_S, k_id)) \equiv \nu k_S. \nu k_{0,0}, \ldots, k_{N,M}. \; !id \leq N \; !sid \leq M \; (I(k_S, k_id_{sid}) \; | \; S(k_S, k_id_{sid}))$$

**Proof**

For all $N, M$, for all trace of observables $tr$, we show that:

$$\models \text{fold}(P_L, tr) \sim \text{fold}(P_R, tr)$$

by induction over $tr$, where $P_L$ and $P_R$ 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**:

\[
(P(\vec{k}) \mid S(\vec{k})) \mid (P(\vec{k}) \mid S(\vec{k})) \approx (P(\vec{k}_0) \mid S(\vec{k}_0)) \mid (P(\vec{k}_1) \mid S(\vec{k}_1))
\]
Authentication Protocols
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 $\tau$ 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.

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

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

  if $\forall A \in \mathcal{I} \pi_{2}(y) = H(\langle n_{R,j}, \pi_{1}(y) \rangle, k_{A})$
  then **out**($c_{j}^{R2}$, ok)
  else **out**($c_{j}^{R2}$, ko)

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

\[
\nu (k_{A})_{A \in \mathcal{I}}. (\forall A \in \mathcal{I} \forall i < N \text{T}(A, i)) \mid (\forall j < M \text{R}(j))
\]

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

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

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

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

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

**Remark:** $tr \diamond c \leq tr'$ denotes that $tr \diamond c \land tr \leq tr'$. 
We let $T_{io}$ be the set of observable traces where all outputs are always directly preceded by an input on the same channel, i.e.:

$$tr \in T_{io} \text{ iff. } \forall tr' \diamond c \leq tr. \exists tr''. tr' = tr''; in(c); out(c)$$

**Assumption: POR**

We admit that to analyze the Hash-Lock protocol, it is sufficient to consider only observables traces in $T_{io}$. 
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.*
Authentication of the Hash-Lock Protocol

For any $\mathbf{tr} \triangle c^j_{R^2} \in \mathcal{T}_{io}$, we let $\text{accept}^{A}@\mathbf{tr}$ be a term stating that the reader accepts the tag $A$ at the end of the trace $\mathbf{tr}$ (defined later).
Informally, Hash-Lock provides authentication if for all \( tr \in T_{io} \), \( tr_1 \diamond c_j^{R_1} \) and \( tr_3 \diamond c_j^{R_2} \) such that:

\[
tr_1 < tr_3 \leq tr \quad \text{and} \quad \text{accept}^A@tr_3
\]

there must exist \( tr_2 \diamond c_T^{A,i} \) such that \( tr_1 \leq tr_2 \leq tr_3 \) and:

\[
\text{out}^@tr_1 = \text{in}^@tr_2 \wedge \text{out}^@tr_2 = \text{in}^@tr_3
\]

Graphically:
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.
For any observable trace $tr$ and observable $\alpha$, we let:

$$\text{pred}(tr; \alpha) \overset{\text{def}}{=} tr$$
We now define some generic terms, called macros, by induction of the observable trace $tr$.

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

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

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

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

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

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

\begin{itemize}
  \item \textbf{Remark:} we omit $\mathcal{P}$ when it is clear from context.
\end{itemize}
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:

\[ \pi_1\langle x, y \rangle \doteq x \sim \text{true} \quad \pi_2\langle x, y \rangle \doteq y \sim \text{true} \]

**Proof**

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

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

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

\[ R(j) : \nu n_{R,j}. \text{in}(c_{j}^{R_1}, \_). \text{out}(c_{j}^{R_1}, n_{R,j}). \text{in}(c_{j}^{R_2}, y). \]

\[
\begin{align*}
\text{if } \bigvee_{A \in I} \pi_2(y) = H(\langle n_{R,j}, \pi_1(y) \rangle, k_A) \\
\text{then } \text{out}(c_{j}^{R_2}, \text{ok}) \\
\text{else } \text{out}(c_{j}^{R_2}, \text{ko})
\end{align*}
\]

To be able to state some authentication property of Hash-Lock, we need an additional macro. For all \( tr \diamond c_{j}^{R_2} \in T_{io} \), we let:

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

💡 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**:

\[
\forall A \in \mathcal{I}. \forall tr \in \mathcal{T}_o. \forall tr_1 \diamond c^{R_1}_j, tr_3 \diamond c^{R_2}_j \text{ s.t. } tr_1 < tr_3 \leq tr, \\
\text{accept}^A_{tr_3} \rightarrow \forall tr_2 \diamond c^{T}_{A,i} \text{ s.t. } tr_1 \leq tr_2 \leq tr_3 \\
\text{out}^{tr_1} = \text{in}^{tr_2} \wedge \text{out}^{tr_2} = \text{in}^{tr_3} \sim \text{true}
\]

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 $\Gamma = t_1 \rightarrow \cdots \rightarrow t_n$ and a (boolean) term $t$.

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

$$(t_1 \rightarrow \cdots \rightarrow t_n \rightarrow t) \sim true$$
Careful not to confuse the boolean connectives at the reachability and equivalence levels!

Exercise
Determine which directions are correct.

\[
\begin{align*}
t_\phi \land t_\psi \sim \text{true} & \iff t_\phi \sim \text{true} \land t_\psi \sim \text{true} \\
t_\phi \lor t_\psi \sim \text{true} & \iff t_\phi \sim \text{true} \lor t_\psi \sim \text{true} \\
t_\phi \rightarrow t_\psi \sim \text{true} & \iff t_\phi \sim \text{true} \rightarrow t_\psi \sim \text{true}
\end{align*}
\]
Careful not to confuse the boolean connectives at the reachability and equivalence levels!

**Exercise**

Determine which directions are correct.

\[ t_\phi \land t_\psi \sim \text{true} \iff t_\phi \sim \text{true} \land t_\psi \sim \text{true} \]

\[ t_\phi \lor t_\psi \sim \text{true} \iff t_\phi \sim \text{true} \lor t_\psi \sim \text{true} \]

\[ t_\phi \rightarrow t_\psi \sim \text{true} \Rightarrow t_\phi \sim \text{true} \rightarrow t_\psi \sim \text{true} \]

The second relation works both ways when \( t_\phi \) or \( t_\psi \) is a **constant** formula.
Our reachability judgements can be trivially equipped with a sequent calculus.

\( \Gamma, t_\phi \vdash t_\phi \)

\( \Gamma \vdash t_\psi \quad \Gamma, t_\psi \vdash t_\phi \)

\( \Gamma \vdash t_\phi \)

\( \Gamma \vdash t_\psi \quad \Gamma \vdash t_\phi \)

\( \Gamma \vdash t_\psi \land t_\phi \)

\( \Gamma \vdash t_\psi \land t_\phi \vdash t_\theta \)

\( \Gamma, t_\psi \land t_\phi \vdash t_\theta \)

\( \Gamma \vdash t_\phi \)

\( \Gamma \vdash t_\psi \quad \Gamma \vdash t_\phi \)

\( \Gamma \vdash t_\psi \lor t_\phi \)

\( \Gamma \vdash t_\psi \lor t_\phi \)

\( \Gamma, t_\psi \lor t_\phi \vdash t_\theta \)

\( \Gamma, t_\psi \lor t_\phi \vdash t_\theta \)

\( \Gamma, t_\psi \vdash t_\theta \quad \Gamma, t_\phi \vdash t_\theta \)

\( \Gamma \vdash t_\psi \rightarrow t_\phi \rightarrow t_\theta \)

\( \Gamma, t_\psi \rightarrow t_\phi \rightarrow t_\theta \)
\[ \Gamma, t_\phi \vdash \bot \]
\[ \Gamma \vdash \neg t_\phi \]
\[ \Gamma, \bot \vdash t_\phi \]

\[ \Gamma_1, t_\phi, t_\psi, \Gamma_2 \vdash t_\theta \]
\[ \Gamma_1, t_\psi, t_\phi, \Gamma_2 \vdash t_\theta \]

\[ \Gamma, t_\psi, t_\psi \vdash t_\phi \]
\[ \Gamma, t_\psi \vdash t_\phi \]
The reachability proof system is \textit{sound}.

\textbf{Proof}

First, remark that any $\Gamma$ and $t_\theta$,

\[ \Gamma \vdash t_\theta \text{ is valid iff. } \Pr_\rho (\mathcal{M}(\Gamma, \neg t_\phi)_{\mathcal{M}}(1^n, \rho)) \text{ is negligible.} \quad (\dagger) \]

- Left-to-right:
  
  \[ \Gamma \vdash t_\theta \text{ valid} \]
  \[ \Rightarrow \forall A \in D. \ Pr_\rho (A(1^n, \mathcal{M}(\Gamma, \neg t_\phi)_{\mathcal{M}}(1^n, \rho), \rho_a) \in \text{negl}(\eta)) \]
  \[ \Rightarrow \Pr_\rho (\mathcal{M}(\Gamma, \neg t_\phi)_{\mathcal{M}}(1^n, \rho)) \in \text{negl}(\eta) \]
  \[ \quad \text{(taking } A(1^n, w, \rho_a) = w) \]

- Right-to-left is straightforward.
We only prove only rule, say
\[ \Gamma, t_\psi \vdash t_\theta \quad \Gamma, t_\phi \vdash t_\theta \]
\[ \implies \Gamma, t_\psi \lor t_\phi \vdash t_\theta \]
By the previous remark (†), since \((\Gamma, t_\psi \vdash t_\theta)\) and \((\Gamma, t_\phi \vdash t_\theta)\) are valid

- \(\Pr_{\rho} \left( \llbracket (\hat{\bigwedge} \Gamma) \hat{\land} t_\psi \hat{\land} \neg t_\theta \rrbracket_{M}(1^\eta, \rho) \right) \) is negligible.
- \(\Pr_{\rho} \left( \llbracket (\hat{\bigwedge} \Gamma) \hat{\land} t_\phi \hat{\land} \neg t_\theta \rrbracket_{M}(1^\eta, \rho) \right) \) is negligible.

Since the union of two negligible (\(\eta\)-indexed families of) events is a negligible (\(\eta\)-indexed families of) events,
\(\Pr_{\rho} \left( \llbracket ((\hat{\bigwedge} \Gamma) \hat{\land} t_\psi \hat{\land} \neg t_\theta) \lor ((\hat{\bigwedge} \Gamma) \hat{\land} t_\phi \hat{\land} \neg t_\theta) \rrbracket_{M}(1^\eta, \rho) \right) \) is negligible
\[ \Leftrightarrow \Pr_{\rho} \left( \llbracket (\hat{\bigwedge} \Gamma) \hat{\land} (t_\psi \lor t_\phi) \hat{\land} \neg t_\theta \rrbracket_{M}(1^\eta, \rho) \right) \) is negligible
Hence using (†) again, \(\Gamma, t_\psi \lor t_\phi \vdash t_\theta\) is valid.
Authentication Protocols

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

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

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

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

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

$$H(m_1, k) \not\approx H(m_2, k) \implies m_1 \not\approx m_2 \sim \text{true}$$

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:

$$\begin{array}{c}
\vdash (H(m_1, k) \vdash H(m_2, k)) \sim \text{true} \\
\hline
\hline
\end{array}$$

when $k$ appears only in $H$ key positions in $m_1$, $m_2$ and $m_1 \neq m_2$. 
Authentication Protocols

Cryptographic Rule: 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:

$$\Pr_k \left( A^{\mathcal{O}_{\text{mac}}(\cdot), \mathcal{O}_{\text{verify}}(\cdot, \cdot)}(1^\eta) = \langle m, \sigma \rangle, \text{ m not queried to } \mathcal{O}_{\text{mac}}(\cdot) \right)$$

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

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

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

- $s$ and $m$ are ground.
- $k \in \mathbb{N}$ 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 $[s, m]$ bottom-up, calling $O_{mac_k}(\cdot)$ and $O_{verify_k}(\cdot, \cdot)$ if necessary.
- Log all sub-terms $S_{mac}(s, m)$ sent to $O_{mac_k}(\cdot)$.

$\Rightarrow$ If $verify_k(s, m)$ then $m = u$ for some $u \in S_{mac}(s, m)$.

$\therefore S_{mac}(s, m)$ are the calls to $O_{mac_k}(\cdot)$ needed to compute $s, m$. 

119
$S_{mac}(\cdot)$ defined by induction on ground terms:

- $S_{mac}(n) \overset{\text{def}}{=} \emptyset$
- $S_{mac}(\text{verify}_k(u_1, u_2)) \overset{\text{def}}{=} S_{mac}(u_1) \cup S_{mac}(u_2)$
- $S_{mac}(\text{mac}_k(u)) \overset{\text{def}}{=} \{ u \} \cup S_{mac}(u)$
- $S_{mac}(f(u_1, \ldots, u_n)) \overset{\text{def}}{=} \bigcup_{1 \leq i \leq n} S_{mac}(u_i)$ (for other cases)
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:

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

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

\[
\models \text{verify}_k(g(\Phi), n) \rightarrow (n \doteq t_1 \lor n \doteq t_2) \sim \text{true}
\]
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:

- \( S_i = \{ u \mid \text{mac}_k(u) \in \mathcal{S}_{\text{mac}}(s_i, m) \}, \) 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 \( b \equiv (\text{att}(k) \models \text{mac}_k(0)) \).
Authentication Protocols

Authentication of the Hash-Lock Protocol
Theorem
Assuming that the hash function is EUF-CMA\textsuperscript{6}, the Hash-Lock protocol provides authentication, i.e. for any identity $a \in \mathcal{I}$, for any $tr \in T_{io}$, $tr_1 \diamond c_{j}^{R_1}$ and $tr_3 \diamond c_{j}^{R_2}$ s.t.:

$$tr_1 < tr_3 \leq tr$$

the following formula is valid:

$$\text{accept}^{A} @ tr_3 \rightarrow \bigvee_{tr_2 \diamond c_{A,i}^{T}, tr_1 \leq tr_2 \leq tr_3} \begin{array}{l}
\text{out} @ tr_1 \Rightarrow \text{in} @ tr_2 \wedge \\
\text{out} @ tr_2 \Rightarrow \text{in} @ tr_3
\end{array} \sim \text{true}$$

\textsuperscript{6}Taking $\text{verify}_k (s, m) \overset{\text{def}}{=} s \Rightarrow H(m, k)$. 

123
**Proof.** Let \( a \in \mathcal{A} \), and let \( tr \in \mathcal{T}_{io} \), \( tr_1 \circ c^R_1 \) and \( tr_3 \circ c^R_2 \) be s.t.:

\[
tr_1 \prec tr_3 \preceq tr
\]

We let:

\[
t_{conc} \overset{\text{def}}{=} \bigvee_{\substack{tr_2 \circ c^T_{A,i} \\tr_1 \leq tr_2 \leq tr_3}} \text{out} @ tr_1 = \text{in} @ tr_2 \land \text{out} @ tr_2 = \text{in} @ tr_3
\]

We must prove that the following reachability judgement is valid:

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

i.e. that:

\[
\pi_2(\text{in} @ tr_3) = H(\langle n_{R,j}, \pi_1(\text{in} @ tr_3) \rangle, k_A) \vdash t_{conc}
\]
Authentication: Hash-Lock

We use the **EUF-MAC** rule on the equality:

\[
\pi_2(\text{in} @ \text{tr}_3) \equiv H(\langle n_{R,j}, \pi_1(\text{in} @ \text{tr}_3) \rangle, k_A) \quad (\dagger)
\]

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 \((\dagger)\), excluding the top-level hash, is:

\[
S_{\text{mac}}(\pi_2(\text{in} @ \text{tr}_3), \langle n_{R,j}, \pi_1(\text{in} @ \text{tr}_3) \rangle) = S_{\text{mac}}(\text{in} @ \text{tr}_3) = \{ H(\langle \text{in} @ \text{tr}_2, n_{T,i} \rangle, k_A) | \text{tr}_2 \diamond c_{A,i}^T < \text{tr}_3 \}
\]

⚠️ *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:

\[
\text{accept}^A_{@\text{tr}_3}, \langle \text{in}_{@\text{tr}_2}, n_{T,i} \rangle \doteq \langle n_{R,j}, \pi_1(\text{in}_{@\text{tr}_3}) \rangle \vdash t_{\text{conc}} \quad \text{for every } \text{tr}_2 \diamond c_{A,i}^T < \text{tr}_3
\]

\[
\text{accept}^A_{@\text{tr}_3}, \bigvee_{\text{tr}_2 \diamond c_{A,i}^T < \text{tr}_3} \langle \text{in}_{@\text{tr}_2}, n_{T,i} \rangle \doteq \langle n_{R,j}, \pi_1(\text{in}_{@\text{tr}_3}) \rangle \vdash t_{\text{conc}}
\]

\[
\text{accept}^A_{@\text{tr}_3} \vdash t_{\text{conc}}
\]
assuming that the pair and projections satisfy:

\[
\begin{align*}
(\pi_1\langle x, y \rangle \doteq x) & \sim true \\
(\pi_2\langle x, y \rangle \doteq y) & \sim true
\end{align*}
\]

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

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

is valid, where:

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

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

Similarly:

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

$$\overset{\diamond}{=} H(\langle n_{R,j}, \pi_1(\text{in}@tr_3) \rangle, k_A)$$

$$\overset{\diamond}{=} \pi_2(\text{in}@tr_3)$$

Consequently:

$$\Gamma \vdash \pi_2(\text{out}@tr_2) \overset{\star}{=} \pi_2(\text{in}@tr_3)$$
Assuming that the pair and projections satisfy the property:

\[
\pi_1 x \equiv \pi_1 y \rightarrow \pi_2 x \equiv \pi_2 y \rightarrow x \equiv y
\]

We deduce from (⋆) and (⋄) that:

\[\Gamma \vdash \text{out}_{@\text{tr}_2} \equiv \text{in}_{@\text{tr}_3}\]

Putting everything together, we get:

\[\Gamma \vdash \text{out}_{@\text{tr}_1} \equiv \text{in}_{@\text{tr}_2} \wedge \text{out}_{@\text{tr}_2} \equiv \text{in}_{@\text{tr}_3}\] (‡)
Authentication: Hash-Lock

Recall that:

\[ t_{\text{conc}} \overset{\text{def}}{=} \bigvee_{\begin{array}{c} \text{tr}_2 \odot_{A,i}^T \text{out@tr}_1 = \text{in@tr}_2 \land \text{out@tr}_2 = \text{in@tr}_3 \\ \text{tr}_1 \leq \text{tr}_2 \leq \text{tr}_3 \end{array}} \]

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:

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

This follows from the independence rule:

\[
(t = n) = \text{false} \rightleftharpoons \text{IND} \quad \text{when } t \text{ is ground and } n \not\in \text{st}(t)
\]

using the fact that:

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

and that if \( \text{tr}_2 < \text{tr}_1 \) then \( n_{R,j} \not\in \text{st}(\text{in@tr}_2) \).
Authentication Protocols

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

\[ \forall \text{tr}_R \diamond \text{c}_R, \exists \text{tr}_T \diamond \text{c}_T. \]

This does not exclude the scenario:

\[ \forall \text{tr}_R \diamond \text{c}_R, \exists \text{tr}_T \diamond \text{c}_T. \]

\[ \text{tr}_R \diamond \text{c}_R \]

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

\[ \text{tr}_T \diamond \text{c}_T \]

\[ \text{tr}_R \diamond \text{c}_1 \]

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

\[ \text{tr}_R \diamond \text{c}_2 \]

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

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

\[
R(j) : \text{in}(c_{j}^{R2}, y). \quad \text{if} \quad \bigvee_{A\in\mathcal{I}} \pi_2(y) = H(\pi_1(y), k_A)
\]

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

else \text{out}(c_{j}^{R2}, \text{ko})
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.
The following formulas encode the fact that the Hash-Lock protocol provides *injective authentication*:

\[
\forall A \in \mathcal{I}. \ \forall tr \in \mathcal{T}_{io}. \ \forall tr_1 \diamond c_{j}^{R_1}, \ tr_3 \diamond c_{j}^{R_2} \ s.t. \ tr_1 < tr_3 \leq tr
\]

\[
\begin{align*}
\text{accept}^A_{@tr_3} \rightarrow & \bigvee_{tr_2 \diamond c_{A,i}^T, \ tr_1 \leq tr_2 \leq tr_3} \\
\text{out}_{@tr_1} = \text{in}_{@tr_2} \land \\
\text{out}_{@tr_2} = \text{in}_{@tr_3} \\
\land \\
\text{out}_{@tr_2} = \text{in}_{@tr_3}' \rightarrow j = k
\end{align*}
\]
Partial order reduction for security protocols.  

A computationally complete symbolic attacker for equivalence properties.  