MPRI 2.30: Proofs of Security Protocols

1. The CCSA Approach to Computational Security

Adrien Koutsos

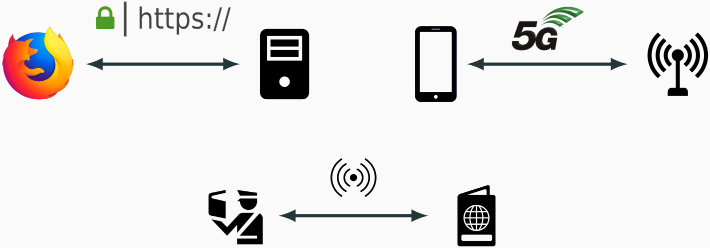
2023/2024

Introduction

Context

Security Protocols

- **Distributed programs** which aim at providing some **security properties**.
- Uses **cryptographic primitives**: e.g. encryption.

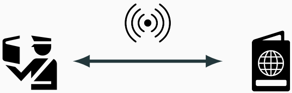


Context: Security Properties

There is a large variety of **security properties**.

Confidentiality
Authentication

Privacy



Privacy

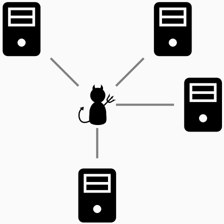
Context: Attacker Model

Against whom should these properties hold?

- **concretely**, in the **real world**: malicious individuals, corporations, state agencies, ...
- more **abstractly**, one (or many) computers sitting on the network.

Abstract attacker model

- **Network capabilities:** worst-case scenario: *eavesdrop*, *block* and *forge* messages.
- **Computational capabilities:** the adversary's *computational power*.
- **Side-channels capabilities:** observing the agents (e.g. time, power-consumption)
  ⇒ not in this lecture.



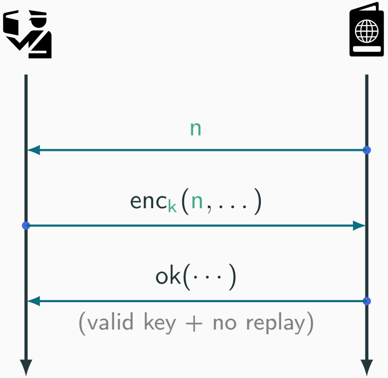
BAC Protocol (simplified)

The Basic Access Control protocol in **e-passports**:

- uses an RFID tag.
- guard access to information stored.
- should guarantee **data confidentiality** and **user privacy**.

Some security mechanisms:

- **integrity**: obtaining **key** k requires **physical access**.
- **no replay**: random nonce n, old messages cannot be re-used.

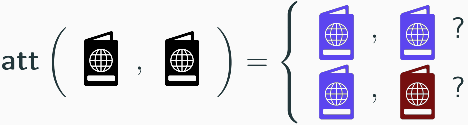


BAC Protocol (simplified)

Privacy: Unlinkability

No adversary can know whether it interacted with a particular user, **in any context**.

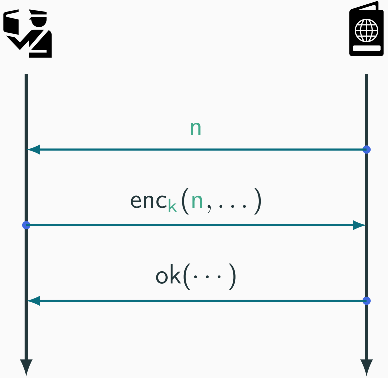
Example. For two user sessions:



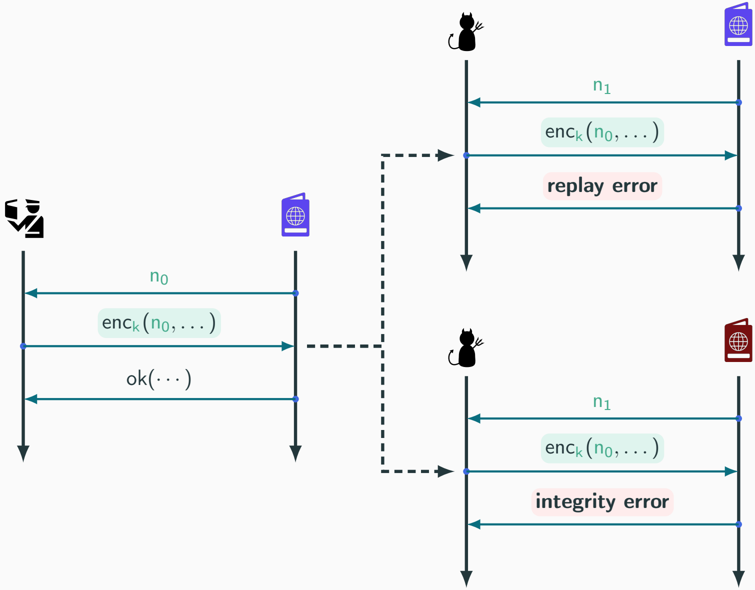
French version of BAC:

- $\neq$ **error messages** for replay and integrity checks.

$\Rightarrow$ **unlinkability attack.**



BAC Protocol: Privacy Attack



BAC Protocol: Lessons

Take-away lessons:

- This is a **protocol-level attack**: no issue with cryptography:
  $\Rightarrow$ cryptographic primitives are but an **ingredient**.
- **Innocuous-looking changes** can **break** security:
  $\Rightarrow$ designing security protocols is **hard**.

How to get a **strong confidence** in a protocol's **security guarantees**?

High-Confidence Security Guarantees

Verification

Formal mathematical proof of security protocols:

- Must be **sound**: proof $\Rightarrow$ property always holds.
- Usually **undecidable**: approaches either **incomplete** or **interactive**.
- **Machine-checked proofs** yield a high degree of confidence.
  - **general-purpose** tools (e.g. Coq and Lean).
  - in security protocol analysis, mostly **dedicated** tools.
    E.g. CryptoVerif, EasyCrypt, Squirrel.

Computer-aided Verification of Cryptographic Protocols

Goal

Design **formal frameworks** allowing for **mechanized verification** of **cryptographic protocols**.

- At the intersection of **cryptography** and **verification**.
- Particular verification challenges:
  - small or medium-sized programs
  - complex properties
  - probabilistic programs + arbitrary adversary

The CCSA Approach to Cryptographic Protocol Verification

The Computationally Complete Symbolic Attacker (**CCSA**) [1] is a framework in the **computational model** for the **verification** of cryptographic protocols.

Key ingredients

- **Protocol executions** models as **terms**.
- A **probabilistic logic**.
  $\Rightarrow$ interpret terms as PTIME-computable bitstring distributions.
- Translate **cryptographic hardness assumptions** as **logical rules**.
- **Reasoning rules** capturing **cryptographic arguments**.
- **Abstract approach**: no probabilities, no security parameter.

Protocols as Sequences of Terms

Example of a Protocol

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

The Private Authentication (PA) Protocol, v1

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

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

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

Terms

We use **terms** to model *protocol messages*, built upon a set of **symbols** $\mathcal{S}$ which includes:

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

$$\mathsf{A},\ \mathsf{B},\ \langle \_\ ,\ \_ \rangle,\ \pi_1(\_),\ \pi_2(\_),\ \{\_\}_{\_},\ \mathsf{pk}(\_),\ \mathsf{sk}(\_),$$
$$\mathsf{if\_then\_else\_},\ \_ \doteq \_,\ \_ \mathbin{\dot\wedge} \_,\ \_ \mathbin{\dot\vee} \_,\ \_ \mathbin{\dot\rightarrow} \_$$

Examples

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

Protocol Constructs

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

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

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

We will discuss how to retrieve replication later.

Protocols as Sequences of Terms

Protocol Inputs

Inputs

The PA Protocol, v1

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

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

How do we represent the adversary's inputs?

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

Example: Terms for PA, v1

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

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

Inputs

More generally, if:

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

then the input bitstring is represented by:

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

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

j allows to have different values for consecutive inputs.

Terms

Thus we extend our set of **term symbols** $\mathcal{S} = \mathcal{N} \uplus \mathcal{X} \uplus \mathcal{F} \uplus \mathcal{G}$:

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

We note $\mathcal{T}(\mathcal{S})$ the set of well-typed (see next slide) terms over symbols $\mathcal{S}$.

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

Terms: Types

Types

Each symbol $s \in \mathcal{S}$ comes with a type $\mathtt{type}(s)$ of the form:

$$(\tau_{\mathsf{b}}^{1} \star \cdots \star \tau_{\mathsf{b}}^{n}) \to \tau_{\mathsf{b}} \qquad \text{or} \qquad \tau_{\mathsf{b}}$$

where $\tau_{\mathsf{b}}^{1}, \ldots, \tau_{\mathsf{b}}^{n}, \tau_{\mathsf{b}}$ are all **base types** in $\mathbb{B}$.

- We ask that $\mathbb{B}$ contains at least the `message` and `bool` types.
- We restrict names to type `message`:

$$\forall \mathsf{n} \in \mathcal{N}, \mathtt{type}(\mathsf{n}) = \mathtt{message}$$

- We restrict *variables* to base types, i.e.:

$$\forall x \in \mathcal{X}, \mathtt{type}(x) \in \mathbb{B}.$$

- We require that terms are well-typed and of a base type:

$$\vdash t : \tau_{\mathsf{b}} \qquad \text{where } \tau_{\mathsf{b}} \in \mathbb{B}.$$

Protocols as Sequences of Terms

Protocol Branching

Protocol Branching

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

The PA Protocol, v2

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

$$2 : \mathsf{B} \to \mathsf{A} : \nu\, \mathsf{n_B}. \mathbf{in}(\mathtt{c_A}, x).\ \text{if } \pi_1(d) \doteq \mathsf{pk_A}$$

$$\text{then } \mathbf{out}(\mathtt{c_B}, \{\langle \pi_2(d)\, ,\, \mathsf{n_B} \rangle\}_{\mathsf{pk_A}})$$

$$\text{else } \mathbf{out}(\mathtt{c_B}, \{0\}_{\mathsf{pk_A}})$$

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

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

Protocol Branching

The PA Protocol, v2

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

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

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

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

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

Protocol Branching

Example: Terms for PA, v2

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

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

Folding

Folding

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

This is the **folding** of P over $\texttt{tr}$.

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

Non-Determinism and Computational Semantics

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

Indeed, consider a simple process:

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

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

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

Action-Deterministic Processes

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

Action-deterministic Process

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

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

Action-Deterministic Processes: Exercise

Exercise

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

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

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

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

Folding

Folding Algorithm

Folding Configuration

Folding configuration

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

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

Folding Configuration: Intuition

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

- Φ is the **frame**, i.e. the sequence of terms outputted since the execution started.
- σ **records inputs**, it maps input variable to their corresponding term.
- j **counts the number of inputs** since the execution started.
- (P, b) **represent the protocol** P if b is true (and is **null** otherwise). Using this interpretation, $\Pi_1, \ldots, \Pi_l$ is the **current process**.

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

Folding: New and Branching Rules

Rule for protocol branching:

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

Rule for new:

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

if n_f does not appear in the lhs configuration

$\hookrightarrow$-irreducibility

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

Folding: Input Rule

Rule for inputs:

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

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

Alternative

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

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

Folding: Output Rule

Rule for outputs:

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

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

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

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

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

Folding

A **folding observable action** a is either **in**($\mathtt{c}$) or **out**($\mathtt{c}$).

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

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

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

Folding: Exercises

Exercise

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

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

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

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

Exercise

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

Semantics of Terms

Semantics of Terms

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

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

Example

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

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

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

Semantics of Terms

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

- a **working tape** (also used as input tape);
- two **read-only tapes** $\rho = (\rho_{\mathsf{a}}, \rho_{\mathsf{h}})$ for adversary and honest 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.

Terms Interpretation

The **interpretation** $[\![t]\!]_{\mathbb{M}} \in \mathcal{D}$ of a term t is parameterized by a **model** $\mathbb{M}$ which provides:

- the set of random tapes $\mathbb{T}_{\mathbb{M},\eta} = \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} \times \mathbb{T}^{\mathsf{h}}_{\mathbb{M},\eta}$, where $\mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta}$ and $\mathbb{T}^{\mathsf{h}}_{\mathbb{M},\eta}$ are **finite** same-length set of bit-strings.
  We equip it with the **uniform** probability measure.
  ($\mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta}$ for the adversary, $\mathbb{T}^{\mathsf{h}}_{\mathbb{M},\eta}$ for honest functions)
- the semantics $(\!|\cdot|\!)_{\mathbb{M}}$ of **symbols** in $\mathcal{S}$ (details on next slides).

We may omit $\mathbb{M}$ when it is clear from context.

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

Terms Interpretation: Function Symbols

Function symbols interpretations is just **composition**.

For **function symbols** in $f \in \mathcal{F}$, we simply apply $(\!| f |\!)_{\mathbb{M}}$:

$$[\![f(t_1, \ldots, t_n)]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} (\!| f |\!)_{\mathbb{M}}(1^\eta, [\![t_1]\!]_{\mathbb{M}}^{\eta,\rho}, \ldots, [\![t_n]\!]_{\mathbb{M}}^{\eta,\rho})$$

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

$$[\![g(t_1, \ldots, t_n)]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} (\!| g |\!)_{\mathbb{M}}(1^\eta, [\![t_1]\!]_{\mathbb{M}}^{\eta,\rho}, \ldots, [\![t_n]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_{\mathsf{a}})$$

Restrictions. $(\!| f |\!)_{\mathbb{M}}$ and $(\!| g |\!)_{\mathbb{M}}$ are:

- PTIME-computable;
- **deterministic** (all randomness must come explicitly, from ρ).

Terms Interpretation: Variables and Names

The interpretation $(\!|x|\!)_{\mathbb{M}}$ of a **variable** $x \in \mathcal{X}$ is an **arbitrary machine** in $\mathcal{D}$. Then:

$$[\![\mathsf{x}]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} (\!|x|\!)_{\mathbb{M}}(1^{\eta}, \rho).$$

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

$$[\![\mathsf{n}]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} (\!|\mathsf{n}|\!)_{\mathbb{M}}(1^{\eta}, \rho_{\mathsf{h}})$$

For every pair of different names $\mathsf{n}_0, \mathsf{n}_1$, we require that $(\!|\mathsf{n}_0|\!)_{\mathbb{M}}$ and $(\!|\mathsf{n}_1|\!)_{\mathbb{M}}$ extracts disjoint parts of ρ_{h}.

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

Terms Interpretation: Builtins

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

- if_then_else_ is interpreted as **branching**:

$$[\![\text{if } b \text{ then } t_1 \text{ else } t_2]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} \begin{cases} [\![t_1]\!]_{\mathbb{M}}^{\eta,\rho} & \text{if } [\![t_1]\!]_{\mathbb{M}}^{\eta,\rho} = 1 \\ [\![t_2]\!]_{\mathbb{M}}^{\eta,\rho} & \text{otherwise} \end{cases}$$

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

$$[\![t_1 \doteq t_2]\!]_{\mathbb{M}}^{\eta,\rho} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } [\![t_1]\!]_{\mathbb{M}}^{\eta,\rho} = [\![t_2]\!]_{\mathbb{M}}^{\eta,\rho} \\ 0 & \text{otherwise} \end{cases}$$

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

Terms Interpretation: Modeling and Randomness

$\neq$ in **how randomness** is sampled:

- **In the "real-world"**, the adversary $\mathcal{A}$ samples randomness on-the-fly, as needed.
  $\Rightarrow$ possibly $P(\eta)$ random bits, where P is the (polynomial) running-time of $\mathcal{A}$.
- **In the logic**, we restrict $\mathbb{T}_{\mathbb{M},\eta} = \mathbb{T}^{\mathsf{a}}_{\mathbb{M},\eta} \times \mathbb{T}^{\mathsf{h}}_{\mathbb{M},\eta}$ to be finite and fixed by $\mathbb{M}$.
  $\Rightarrow$ all randomness sampled eagerly according to $\mathbb{M}$, independently of the adversary $\mathcal{A}$.

This $\neq$ of behaviors is not an issue, i.e. the logic can **soundly model** real-world adversaries:

- Indeed, for any adversary $\mathcal{A}$, there exists a model $\mathbb{M}$ with enough randomness.

A First-Order Logic for Indistinguishability

A First-Order Logic for Indistinguishability

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

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

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

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

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

Semantics of the Logic

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

The **satisfaction** $\mathbb{M} \models \Phi$ of Φ in $\mathbb{M}$ is as expected for **boolean connective** and FO **quantifiers**. E.g.:

$$\mathbb{M} \models \top \qquad\qquad \mathbb{M} \models \Phi \wedge \Psi \quad \text{if } \mathbb{M} \models \Phi \text{ and } \mathbb{M} \models \Psi$$

$$\mathbb{M} \models \neg\Phi \quad \text{if not } \mathbb{M} \models \Phi \qquad\qquad \mathbb{M} \models \forall \mathsf{x}.\Phi \quad \text{if } \forall m \in \mathcal{D}, \mathbb{M}[\mathsf{x} \mapsto m] \models \Phi$$

Semantics of the Logic

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

$$\mathbb{M} \models t_1, \ldots, t_n \sim_n s_1, \ldots, s_n$$

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

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

is a **negligible** function of η.

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

Negligible Functions

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

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

Example

Let f be the function defined by:

$$f(\eta) \stackrel{\text{def}}{=} \Pr_\rho\left([\![\mathsf{n}_0]\!]^{\eta,\rho} = [\![\mathsf{n}_1]\!]^{\eta,\rho}\right)$$

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

Satisfiability and Validity

A formula Φ is **satisfied** by a model $\mathbb{M}$ when $\mathbb{M} \models \Phi$.

Φ is **valid**, denoted by $\models \Phi$, if it is **satisfied** by **every model**.

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

Validity: Exercise

Exercise

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

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

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

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

Validity: Exercise

Exercise

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

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

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

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

Protocol Indistinguishability

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

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

Remark

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

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

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

Protocol Indistinguishability: Exercise

Exercise

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

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

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

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

Structural Rules

Rules: Soundness

A rule:

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

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

Example

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

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

Structural Rules

Computational indistinguishability is an **equivalence relation**:

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

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

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

Structural Rules

Alpha-renaming.

$$\frac{}{\vec{u} \sim \vec{u}\alpha}\ \alpha\text{-}\textsc{Equ}$$

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

Restriction. The adversary can throw away some values:

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

Structural Rules

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

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

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

$$\frac{\vec{u_1}, \vec{v_1} \sim \vec{u_1}, \vec{v_2}}{f(\vec{u_1}), \vec{v_1} \sim f(\vec{u_2}), \vec{v_2}} \text{ FA}$$

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

Structural Rules: Proof of Function Application

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

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

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

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

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

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

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

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

Structural Rules

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

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

Structural Rules: Proof of Case Study

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

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

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

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

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

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

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

Structural Rules: Proof of Case Study (continued)

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

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

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

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

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

Structural Rules: Proof of Case Study (continued)

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

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

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

Counter-Examples

Remark that b is **necessary** in CS

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

We have:

$$\models \langle 0\,,\, \mathsf{n}_0 \rangle \sim \langle 0\,,\, \mathsf{n}_0 \rangle \qquad \models \langle 1\,,\, \mathsf{n}_0 \rangle \sim \langle 1\,,\, \mathsf{n}_0 \rangle \qquad \models \mathtt{even}(\mathsf{n}_0) \sim \mathtt{odd}(\mathsf{n}_0)$$

But:

$$\not\models \begin{array}{l} \quad \text{if } \mathtt{even}(\mathsf{n}_0) \text{ then } \langle 0\,,\, \mathsf{n}_0 \rangle \text{ else } \langle 1\,,\, \mathsf{n}_0 \rangle \\ \sim \text{if } \ \mathtt{odd}(\mathsf{n}_0) \text{ then } \langle 0\,,\, \mathsf{n}_0 \rangle \text{ else } \langle 1\,,\, \mathsf{n}_0 \rangle \end{array}$$

Why is the later formula not valid?

Structural Rules: Equality Reasoning

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

If ϕ is a term of type `bool`, let $[\phi] \stackrel{\text{def}}{=} \phi \sim \text{true}$.
$\Rightarrow$ i.e. ϕ is *overwhelmingly true* (equivalently, $\neg\phi$ is *negligible*).

Then the following rule is sound:

$$\frac{\vec{u}, t \sim \vec{v} \qquad [s \doteq t]}{\vec{u}, s \sim \vec{v}}\ \mathrm{R}$$

Structural Rules: Equality Reasoning

Proof

First, for any model $\mathbb{M}$, we have:

$$\mathbb{M} \models [\phi] \text{ iff. } \Pr_\rho \left([\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} \right) \text{ is overwhelming.}$$

- Left-to-right:

$$\begin{aligned}
& \mathbb{M} \models [\phi] \\
\Rightarrow\ & \forall A \in \mathcal{D}.\ \left| \Pr_\rho \left(\mathcal{A}(1^\eta, [\![\phi]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) \right) - \Pr_\rho \left(\mathcal{A}(1^\eta, [\![\mathsf{true}]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) \right) \right| \in \mathsf{negl}(\eta) \\
\Rightarrow\ & \left| \Pr_\rho \left([\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} \right) - 1 \right| \in \mathsf{negl}(\eta) && \text{(taking } \mathcal{A}(1^\eta, w, \rho_a) = w\text{)} \\
\Rightarrow\ & \Pr_\rho \left([\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} \right) \in \mathsf{o.w.}(\eta)
\end{aligned}$$

- Right-to-left, assume $\Pr_\rho \left([\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} \right) \in \mathsf{o.w.}(\eta)$ and take $\mathcal{A} \in \mathcal{D}$:

$$\begin{aligned}
& \left| \Pr_\rho \left(\mathcal{A}(1^\eta, [\![\phi]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) \right) - \Pr_\rho \left(\mathcal{A}(1^\eta, [\![\mathsf{true}]\!]_{\mathbb{M}}^{\eta,\rho}, \rho_a) \right) \right| \\
\leq\ & \Pr_\rho \left(\neg [\![\phi]\!]_{\mathbb{M}}^{\eta,\rho} \right) && \text{(up-to-bad)} \\
\in\ & \mathsf{negl}(\eta)
\end{aligned}$$

Structural Rules: Equality Reasoning

This allows to conclude immediately since:

$$\begin{aligned} & |\Pr(\mathcal{A}(\llbracket \vec{u}, t \rrbracket)) - \Pr(\mathcal{A}(\llbracket \vec{v} \rrbracket))| \\ \leq \ & |\Pr(\mathcal{A}(\llbracket \vec{u}, s \rrbracket)) - \Pr(\mathcal{A}(\llbracket \vec{v} \rrbracket))| + \Pr\left(\llbracket s \rrbracket \neq \llbracket t \rrbracket\right) \qquad \text{(up-to-bad)} \end{aligned}$$

Reminder: up-to-bad argument

If B, E, E' are events such that:

$$(E \land \neg B) \Leftrightarrow (E' \land \neg B), \qquad (\diamond)$$

then $|\Pr(E) - \Pr(E')| \leq \Pr(B)$.

Indeed, by triangular inequality and total probabilities:

$$|\Pr(E) - \Pr(E')| \leq |\Pr(E \land B) - \Pr(E' \land B)| + |\Pr(E \land \neg B) - \Pr(E' \land \neg B)|$$

We conclude by observing that:

- $|\Pr(E \land \neg B) - \Pr(E' \land \neg B)| = 0$ by $(\diamond)$;
- $|\Pr(E \land B) - \Pr(E' \land B)| \leq \max(\Pr(E \land B), \Pr(E' \land B)) \leq \Pr(B)$.

Structural Rules: Generic Equality Reasoning

To prove $\models [s \doteq t]$ (or more generally $\models [\phi]$), we use the rule:

$$\frac{\mathcal{A}_{\mathsf{th}} \vdash_{\textsc{Gen}} \phi}{[\phi]} \; \textsc{gen}$$

where $\vdash_{\textsc{Gen}}$ is any **sound proof system** for generic mathematical reasoning (e.g. higher-order logic).

This allows **exact** (i.e. non-probabilistic) mathematical reasoning.

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

Example

$$\mathcal{A}_{\mathsf{th}} \vdash_{\textsc{Gen}} v \doteq w \dot{\to} \begin{pmatrix} \text{if } u \doteq v \text{ then } u \text{ else } t & \doteq \\ \text{if } u \doteq v \text{ then } w \text{ else } t & \end{pmatrix}$$

Structural Rules: Probabilistic Independence

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

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

$$\frac{\vec{u} \sim \vec{v}}{\vec{u}, \mathsf{n}_0 \sim \vec{v}, \mathsf{n}_1}\ \textsc{Fresh} \quad \text{when } \mathsf{n}_0 \notin \mathsf{st}(\vec{u}) \text{ and } \mathsf{n}_1 \notin \mathsf{st}(\vec{v})$$

Remark

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

Structural Rules: Probability Independence

We give the proof of the first rule:

$$\frac{}{[t \mathrel{\dot{\neq}} \mathsf{n}]}\ \text{=-IND} \qquad \text{when } \mathsf{n} \notin \mathsf{st}(t)$$

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

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

□

Exercise

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

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

Implementation Rules

Rules: Soundness

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

Example

$$\overline{[\pi_1 \langle x\,,\, y \rangle \doteq x]}$$

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

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

Implementation Assumptions

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

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

Typically, this is used for:

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

Functional Properties

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

- $\pi_i(\langle x_1, x_2 \rangle) = x_i$ $\qquad i \in \{1, 2\}$
- $\mathsf{dec}(\{x\}^{z}_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x$
- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
- $\ldots$

Cryptographic Rules

Cryptographic Reduction

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

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

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

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

Cryptographic Reduction

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

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

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

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

Cryptographic Rules

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

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

- Asymmetric encryption: indistinguishability ($\mathsf{IND\text{-}CCA}_1$) and key-privacy ($\mathsf{KP\text{-}CCA}_1$);
- Hash function: collision-resistance (CR-HK);
- MAC: unforgeability (EUF-CMA);

Cryptographic Rules

Asymmetric Encryption

Asymmetric Encryption Scheme

An asymmetric encryption scheme contains:

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

It must satisfies the functional equality:

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

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

IND-CCA$_1$ Security

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

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

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

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

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

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

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

IND-CCA$_1$ Security: Exercise

Exercise

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

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

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

IND-CCA$_1$ Rule

Indistinguishability Against Chosen Ciphertexts Attacks

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

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

is sound, when:

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

$$\mathsf{n} \sqsubseteq_{\mathsf{pk}(\cdot),\mathsf{dec}(\_,\mathsf{sk}(\cdot))} \vec{u}, t_0, t_1$$

IND-CCA$_1$ Rule: Conditions

Definition: Positions

We write $\text{pos}(t) \in \{\epsilon\} \cup \mathbb{N} (\cdot \mathbb{N})^*$ the set of *positions* of t and $t_{|p}$ the sub-term of t at position p.

Example

if $t \equiv f(g(a, b), h(c))$ then $\text{pos}(t) = \{\epsilon, 0, 1, 0 \cdot 0, 0 \cdot 1, 1, 1 \cdot 0\}$ and:

$$t_{|\epsilon} \equiv t \qquad t_{|0} \equiv g(a, b) \qquad t_{|0\cdot 0} \equiv a \qquad t_{|0\cdot 1} \equiv b \qquad t_{|1} \equiv h(c)$$

$$t_{|1\cdot 0} \equiv c$$

IND-CCA$_1$ Rule: Conditions

Definition: CCA$_1$ Side-Condition

$(\mathsf{n} \sqsubseteq_{\mathsf{pk}(\cdot),\mathsf{dec}(\_,\mathsf{sk}(\cdot))} u)$ iff. for any $p \in \mathsf{pos}(u)$, if $t_{|p} \equiv \mathsf{n}$, either:

- $p = p_0 \cdot 0$ and $t_{|p_0} \equiv \mathsf{pk}(\mathsf{n})$;
- or $p = p_0 \cdot 1 \cdot 0$ and $t_{|p_0} \equiv \mathsf{dec}(s, \mathsf{sk}(\mathsf{n}))$.

Examples (writing $\sqsubseteq$ instead of $\sqsubseteq_{\mathsf{pk}(\cdot),\mathsf{dec}(\_,\mathsf{sk}(\cdot))}$)

$\mathsf{n} \not\sqsubseteq \mathsf{n}$ $\quad$ $\mathsf{n} \sqsubseteq \mathsf{pk}(\mathsf{pk}(\mathsf{n}))$ $\quad$ $\mathsf{n} \sqsubseteq \mathsf{dec}(\mathsf{pk}(\mathsf{n}), \mathsf{sk}(\mathsf{n}))$

$\mathsf{n} \not\sqsubseteq \mathsf{dec}(\mathsf{sk}(\mathsf{n}), \mathsf{sk}(\mathsf{n}))$ $\quad$ $\mathsf{n} \sqsubseteq t$ if $\mathsf{n} \not\in \mathsf{st}(t)$

IND-CCA$_1$ Rule: Proof

Proof sketch

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

$$\left| \begin{aligned} &\mathsf{Pr}_\rho(\mathcal{A}(1^\eta, [\![\vec{u}]\!]^{\eta,\rho}_{\mathbb{M}}, [\![\{t_0\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})}]\!]^{\eta,\rho}_{\mathbb{M}}, \rho_{\mathsf{a}}) \\ -\ &\mathsf{Pr}_\rho(\mathcal{A}(1^\eta, [\![\vec{u}]\!]^{\eta,\rho}_{\mathbb{M}}, [\![\{t_1\}^{\mathsf{r}}_{\mathsf{pk}(\mathsf{n})}]\!]^{\eta,\rho}_{\mathbb{M}}, \rho_{\mathsf{a}}) \end{aligned} \right|$$

is not negligible, and $\mathbb{M} \models [\mathsf{len}(t_0) \doteq \mathsf{len}(t_1)]$.

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

IND-CCA$_1$ Rule: Proof

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

i) **lazily**[4] samples the random tapes $(\rho_\mathsf{a}, \rho'_\mathsf{h})$ where:

$$\rho'_\mathsf{h} := \rho_\mathsf{h}[\mathsf{n} \mapsto 0, \mathsf{r} \mapsto 0]$$

ii) compute[5]:

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

using $(\rho_\mathsf{a}, \rho'_\mathsf{h})$, $[\![\mathsf{pk}(\mathsf{n})]\!]_{\mathbb{M}}^{\eta,\rho}$ and calls to $\mathcal{O}_{\mathsf{dec}}^{\mathsf{n}}$.

iii) return 0 if $\mathsf{len}(t_0) \dot{\neq} \mathsf{len}(t_1)$.

iii) otherwise, compute:

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

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

[4] Why do we need this?
[5] We describe how later.

IND-CCA$_1$ Rule: Proof

Then:

$$\begin{aligned} \mathsf{Adv}(\mathcal{A}) &\leq \mathsf{Adv}(\mathcal{A} \wedge \mathsf{len}(t_0) \doteq \mathsf{len}(t_1)) + \mathsf{Pr}(\mathsf{len}(t_0) \dot{\neq} \mathsf{len}(t_1)) \qquad \text{(up-to-bad)} \\ &= \mathsf{Adv}(\mathcal{B} \wedge \mathsf{len}(t_0) \doteq \mathsf{len}(t_1)) + \mathsf{Pr}(\mathsf{len}(t_0) \dot{\neq} \mathsf{len}(t_1)) \\ &= \mathsf{Adv}(\mathcal{B}) + \mathsf{Pr}(\mathsf{len}(t_0) \dot{\neq} \mathsf{len}(t_1)) \end{aligned}$$

Hence $\mathcal{B}$'s advantage against IND-CCA$_1$ is at least $\mathcal{A}$'s advantage against:

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

up-to a negligible quantity (the probability that $\mathsf{len}(t_0) \dot{\neq} \mathsf{len}(t_1)$).

Since ($\dagger$) is assumed non-negligible, so is $\mathcal{B}$'s advantage.

IND-CCA$_1$ Rule: Proof

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

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

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

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

IND-CCA$_1$ Rule: Proof

Induction. We are in one of the following cases:

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

$$w_s := \begin{cases} (\!|f|\!)_{\mathbb{M}}(1^\eta, w_1, \ldots, w_n) & \text{if } f \in \mathcal{F} \\ (\!|f|\!)_{\mathbb{M}}(1^\eta, w_1, \ldots, w_n, \rho_{\mathsf{a}}) & \text{if } f \in \mathcal{G} \end{cases}$$

IND-CCA$_1$ Rule: Proof

case disjunction (continued):

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

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

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

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

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

IND-CCA_1 Rule: Exercise

Exercise

Which of the following formulas can be proven using IND-CCA_1?

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

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

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

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

IND-CCA$_1$ Rule: Exercise

Exercise (Hybrid Argument)

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

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

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

KP-CCA$_1$ Security

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

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

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

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

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

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

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

Exercise

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

KP-CCA$_1$ Rule

Key Privacy Against Chosen Ciphertexts Attacks

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

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

is sound, when:

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

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

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