# On Equivalences, Metrics, and Polynomial Time

***Ugo Dal Lago***
(Joint work with *Alberto Cappai*)

DICE 2016, Eindhoven, April 2nd.

# In this talk:

- We give **a few observations** on the structure of some (pretty old!) security proofs.
- We show how techniques for **program equivalence** could be applied to **ICC systems**.
- We do **not** define any new fancy technique for proving protocols secure;
- We do **not** study automation.

# Proofs by Reduction

$$\mathbf{Property}(\Phi) \stackrel{def}{=} (\forall \mathcal{D}.PPT(\mathcal{D}) \Rightarrow \mathsf{Pr}(Success(\mathcal{D} \mid \Phi)) = negl(n))\,.$$

$$\mathbf{Assumption}(\Phi) \quad \Rightarrow \quad \mathbf{Security}(\Pi)$$

Examples

$$\mathbf{OneWay}(f) \quad \Rightarrow \quad \mathbf{PRG}(G)$$
$$\mathbf{PRG}(G) \quad \Rightarrow \quad \mathbf{IND}(\Pi)$$

# Proofs by Reduction

$$\mathbf{Property}(\Phi) \stackrel{def}{=} (\forall \mathcal{D}.PPT(\mathcal{D}) \Rightarrow \mathsf{Pr}(Success(\mathcal{D} \mid \Phi)) = negl(n))\,.$$

$$\mathbf{Assumption}(\Phi) \quad \Rightarrow \quad \mathbf{Security}(\Pi)$$

Examples

$$\mathbf{OneWay}(f) \quad \Rightarrow \quad \mathbf{PRG}(G)$$
$$\mathbf{PRG}(G) \quad \Rightarrow \quad \mathbf{IND}(\Pi)$$

# Proofs by Reduction

$$\textbf{Property}(\Phi) \stackrel{def}{=} (\forall \mathcal{D}.PPT(\mathcal{D}) \Rightarrow \mathsf{Pr}(Success(\mathcal{D} \mid \Phi)) = negl(n))\,.$$

$$\textbf{Assumption}(\Phi) \quad \Rightarrow \quad \textbf{Security}(\Pi)$$

Examples

$$\textbf{OneWay}(f) \quad \Rightarrow \quad \textbf{PRG}(G)$$
$$\textbf{PRG}(G) \quad \Rightarrow \quad \textbf{IND}(\Pi)$$

# Proofs by Reduction

$$\mathbf{Property}(\Phi) \stackrel{def}{=} (\forall \mathcal{D}.PPT(\mathcal{D}) \Rightarrow \mathsf{Pr}(Success(\mathcal{D} \mid \Phi)) = negl(n))\,.$$

$$\mathbf{Assumption}(\Phi) \quad \Rightarrow \quad \mathbf{Security}(\Pi)$$

Examples

$$\mathbf{OneWay}(f) \quad \Rightarrow \quad \mathbf{PRG}(G)$$

$$\mathbf{PRG}(G) \quad \Rightarrow \quad \mathbf{IND}(\Pi)$$

# Proofs by Reduction

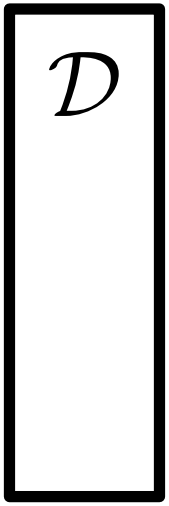
$$\mathbf{Property}(\Phi) \stackrel{def}{=} (\forall \mathcal{D}.PPT(\mathcal{D}) \Rightarrow \mathsf{Pr}(Success(\mathcal{D} \mid \Phi)) = negl(n))\,.$$

$$\mathbf{Assumption}(\Phi) \quad \Rightarrow \quad \mathbf{Security}(\Pi)$$

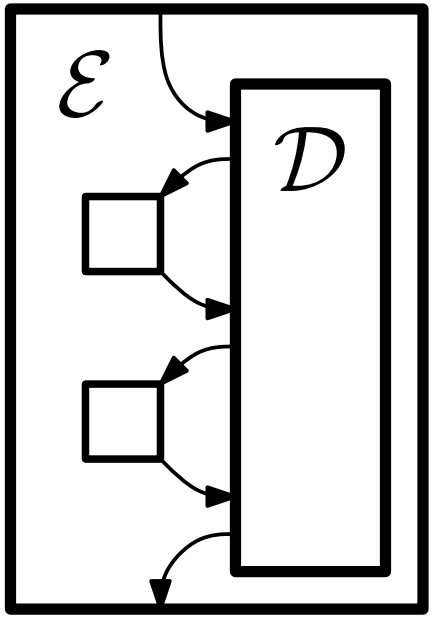
Examples

$$\mathbf{OneWay}(f) \quad \Rightarrow \quad \mathbf{PRG}(G)$$
$$\mathbf{PRG}(G) \quad \Rightarrow \quad \mathbf{IND}(\Pi)$$

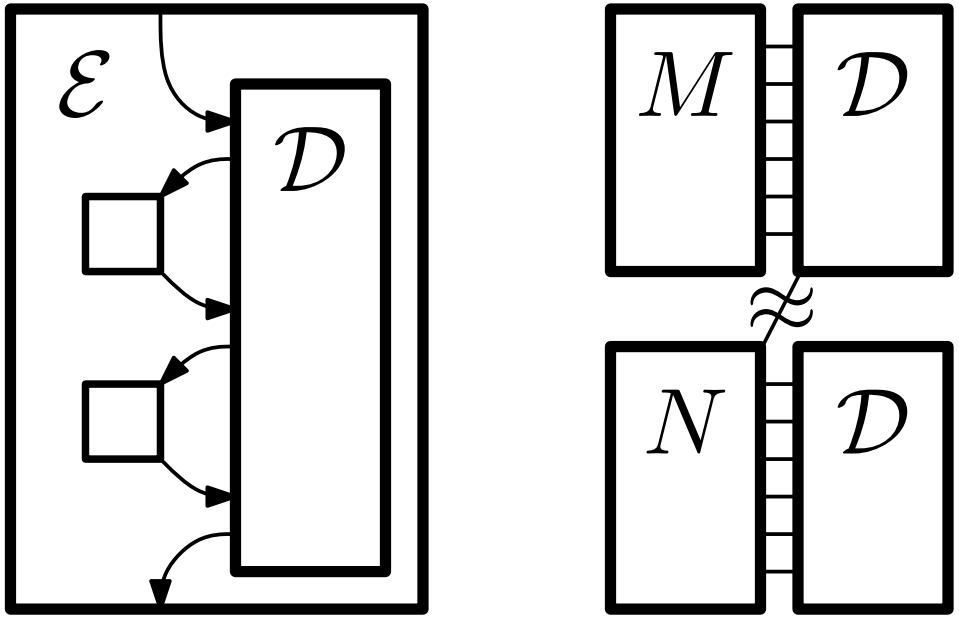
# Proofs by Reduction



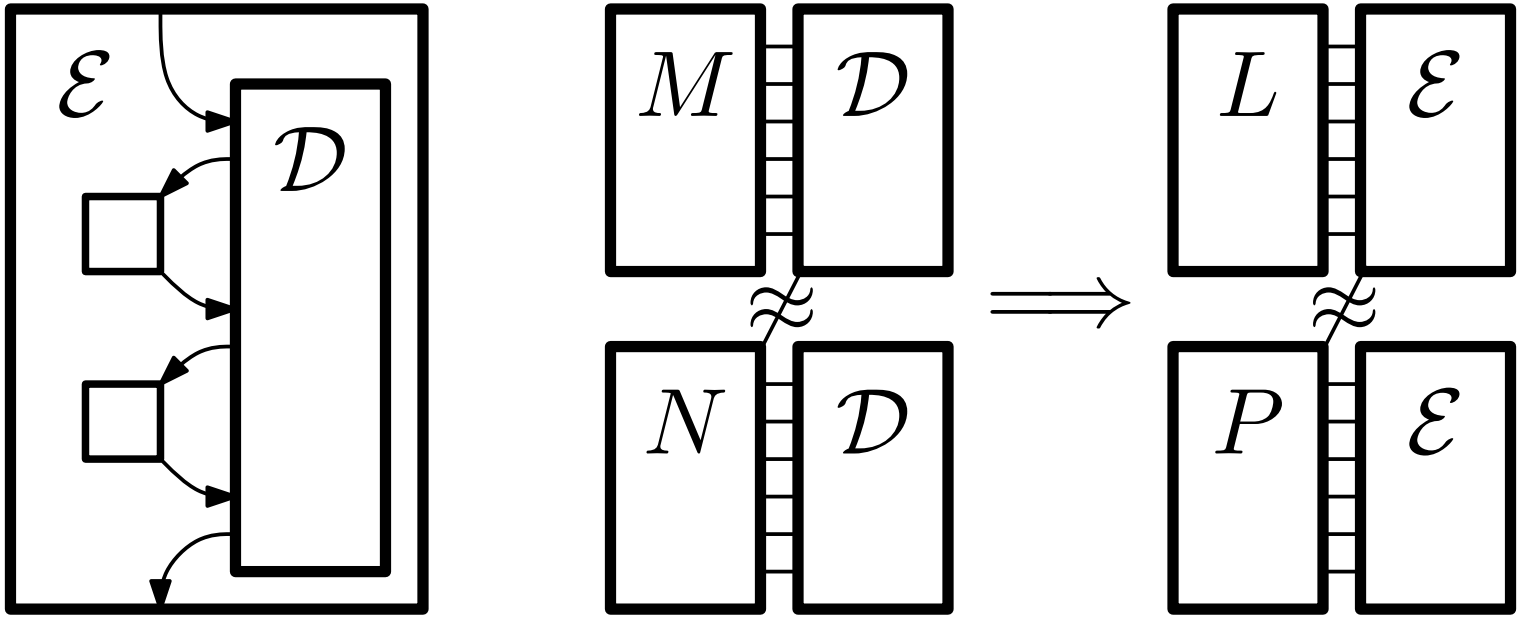
# Proofs by Reduction



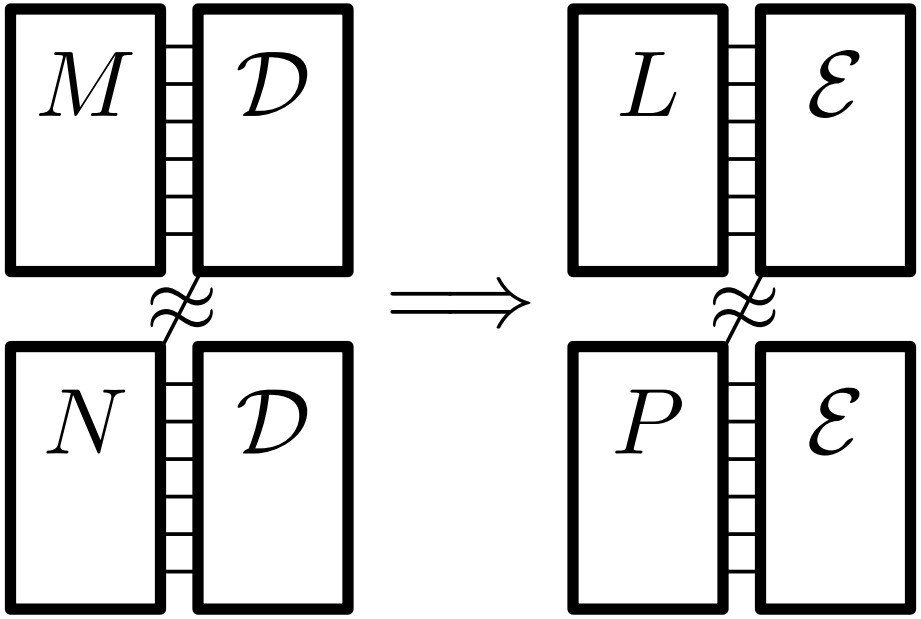
# Proofs by Reduction



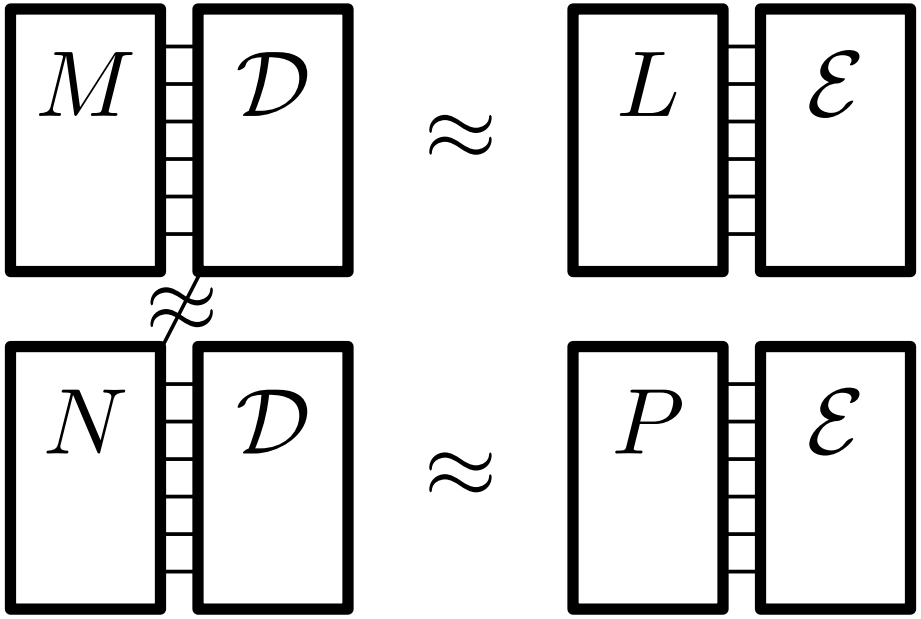
# Proofs by Reduction



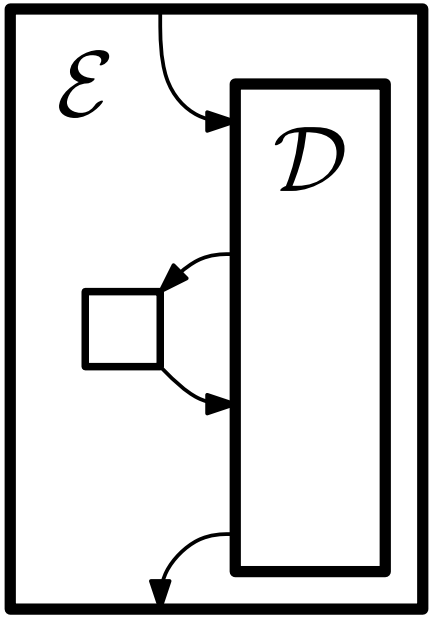
# Game-Based Proofs



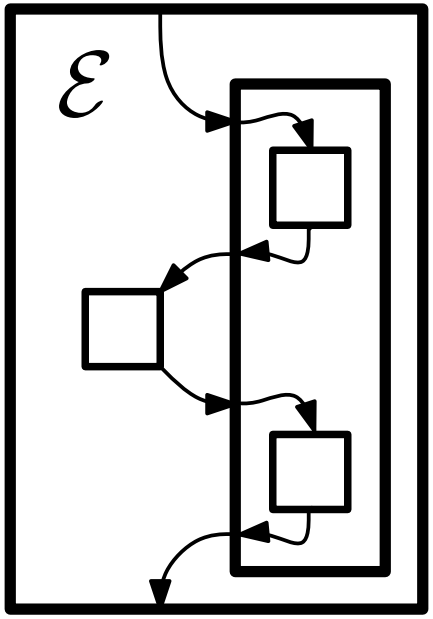
# Game-Based Proofs



# An Alternative Viewpoint

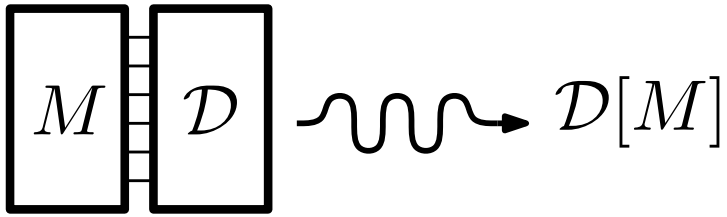


# An Alternative Viewpoint



# An Alternative Viewpoint

1. Introduce a $\lambda$-calculus RSLR, and prove it **complete** for probabilistic polynomial-time computation [Zhang10,DLPT11].




2. Define a notion a notion of **equivalence** close enough to *computational indistinguishability*.
3. Prove an appropriate **context lemma** in the form of full-abstraction for trace equivalence.

# RSLR in One Slide

- **Terms** and **Values**:

$$t ::= x \mid v \mid \mathsf{0}(t) \mid \mathsf{1}(t) \mid \mathsf{tail}(t) \mid tt \mid \mathsf{case}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rec}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rand};$$

$$v ::= \underline{m} \mid \lambda x : \mathsf{aA}.t;$$

- **Types**: $\mathsf{A} ::= \mathsf{Str} \mid \Box\mathsf{A} \to \mathsf{A} \mid \blacksquare\mathsf{A} \to \mathsf{A}$.
- **Type System's Main Ingredients**:
  - Higher-order terms *cannot* be duplicated, while strings *can*.
  - Recursion can only be performed on $\Box$-free types [BC92].

**Theorem (Polytime Completeness)**

*The set of probabilistic functions which can be computed by* RSLR *terms coincides with the polytime computable ones.*

# RSLR in One Slide

- **Terms** and **Values**:

$$t ::= x \mid v \mid \mathsf{0}(t) \mid \mathsf{1}(t) \mid \mathsf{tail}(t) \mid tt \mid \mathsf{case}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rec}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rand};$$

$$v ::= \underline{m} \mid \lambda x : \mathsf{aA}.t;$$

- **Types**: $\mathsf{A} ::= \mathsf{Str} \mid \Box\mathsf{A} \to \mathsf{A} \mid \blacksquare\mathsf{A} \to \mathsf{A}$.
- **Type System's Main Ingredients**:
  - Higher-order terms *cannot* be duplicated, while strings *can*.
  - Recursion can only be performed on $\Box$-free types [BC92].

**Theorem (Polytime Completeness)**

*The set of probabilistic functions which can be computed by* RSLR *terms coincides with the polytime computable ones.*

# RSLR in One Slide

- **Terms** and **Values**:

$$t ::= x \mid v \mid \mathsf{0}(t) \mid \mathsf{1}(t) \mid \mathsf{tail}(t) \mid tt \mid \mathsf{case}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rec}_{\mathsf{A}}(t,t,t,t) \mid \mathsf{rand};$$

$$v ::= \underline{m} \mid \lambda x : \mathsf{aA}.t;$$

- **Types**: $\mathsf{A} ::= \mathsf{Str} \mid \Box\mathsf{A} \to \mathsf{A} \mid \blacksquare\mathsf{A} \to \mathsf{A}$.
- **Type System's Main Ingredients**:
  - Higher-order terms *cannot* be duplicated, while strings *can*.
  - Recursion can only be performed on $\Box$-free types [BC92].

**Theorem (Polytime Completeness)**

*The set of probabilistic functions which can be computed by* RSLR *terms coincides with the polytime computable ones.*

# RSLR in One Slide

- **Terms** and **Values**:

$$t \ ::= x \ \Big| \ v \ \Big| \ \mathsf{0}(t) \ \Big| \ \mathsf{1}(t) \ \Big| \ \mathsf{tail}(t) \ \Big| \ tt \ \Big| \ \mathsf{case}_{\mathsf{A}}(t,t,t,t) \ \Big| \ \mathsf{rec}_{\mathsf{A}}(t,t,t,t) \ \Big| \ \mathsf{rand};$$

$$v \ ::= \underline{m} \ \Big| \ \lambda x : \mathsf{aA}.t;$$

- **Types**: $\mathsf{A} ::= \mathsf{Str} \ \Big| \ \Box\mathsf{A} \to \mathsf{A} \ \Big| \ \blacksquare\mathsf{A} \to \mathsf{A}$.
- **Type System's Main Ingredients**:
  - Higher-order terms *cannot* be duplicated, while strings *can*.
  - Recursion can only be performed on $\Box$-free types [BC92].

**Theorem (Polytime Completeness)**

*The set of probabilistic functions which can be computed by* RSLR *terms coincides with the polytime computable ones.*

# How Should we Compare Terms in RSLR?

- An **equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}}$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90,DanosHarmer00,...].
  - Too discriminating (e.g., $\mathtt{tt}$ vs. $\mathtt{tt} \oplus^{\epsilon} \mathtt{ff}$).
- A **metric**: $\delta : \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}} \to \mathbb{R}$.
  - How to define it? How could one compare two terms in such a way that their distance reflects their behaviour?
  - Not much is known [CrübilléDL2015].
- A **parametric equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathbb{N}\to\mathsf{A}} \times \mathrm{T}^{\mathbb{N}\to\mathsf{A}}$.
  - The kind of concept one needs in cryptography.

# How Should we Compare Terms in RSLR?

- An **equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}}$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90,DanosHarmer00,...].
  - Too discriminating (e.g., $\mathtt{tt}$ vs. $\mathtt{tt} \oplus^{\epsilon} \mathtt{ff}$).
- A **metric**: $\delta : \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}} \to \mathbb{R}$.
  - How to define it? How could one compare two terms in such a way that their distance reflects their behaviour?
  - Not much is known [CrübilléDL2015].
- A **parametric equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathbb{N}\to\mathsf{A}} \times \mathrm{T}^{\mathbb{N}\to\mathsf{A}}$.
  - The kind of concept one needs in cryptography.

# How Should we Compare Terms in RSLR?

- An **equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}}$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90,DanosHarmer00,...].
  - Too discriminating (e.g., $\mathtt{tt}$ vs. $\mathtt{tt} \oplus^{\epsilon} \mathtt{ff}$).
- A **metric**: $\delta : \mathrm{T}^{\mathsf{A}} \times \mathrm{T}^{\mathsf{A}} \to \mathbb{R}$.
  - How to define it? How could one compare two terms in such a way that their distance reflects their behaviour?
  - Not much is known [CrübilléDL2015].
- A **parametric equivalence relation**: $\mathcal{R} \subseteq \mathrm{T}^{\mathbb{N}\to\mathsf{A}} \times \mathrm{T}^{\mathbb{N}\to\mathsf{A}}$.
  - The kind of concept one needs in cryptography.

# Part I

# Equivalences

# Contexts

- **Contexts**:

$$C ::= t \mid [\cdot] \mid \lambda x.C \mid Ct \mid tC \mid \mathsf{0}(C) \mid \mathsf{1}(C) \mid \mathsf{tail}(C)$$
$$\mid \mathsf{case}_{\mathsf{A}}(C,t,t,t) \mid \mathsf{case}_{\mathsf{A}}(t,C,C,C) \mid \mathsf{rec}_{\mathsf{A}}(C,t,t,t).$$

- Notice that we are working with *linear* contexts!
- One can easily define a formal system for judgements in the form $\Gamma \vdash C[\vdash \mathsf{A}]$, with the obvious meaning.

**Definition (Context Equivalence)**

Given two terms $t, s$ such that $\vdash t, s : \mathsf{A}$, we say that $t$ and $s$ are *context equivalent* iff for every context $C$ such that $\vdash C[\vdash \mathsf{A}] : \mathsf{Str}$ we have that $[\![C[t]]\!](\underline{\epsilon}) = [\![C[s]]\!](\underline{\epsilon})$.

# Contexts

- **Contexts**:

$$C ::= t \mid [\cdot] \mid \lambda x.C \mid Ct \mid tC \mid \mathsf{0}(C) \mid \mathsf{1}(C) \mid \mathsf{tail}(C)$$
$$\mid \mathsf{case}_{\mathsf{A}}(C,t,t,t) \mid \mathsf{case}_{\mathsf{A}}(t,C,C,C) \mid \mathsf{rec}_{\mathsf{A}}(C,t,t,t).$$

- Notice that we are working with *linear* contexts!
- One can easily define a formal system for judgements in the form $\Gamma \vdash C[\vdash \mathsf{A}]$, with the obvious meaning.

**Definition (Context Equivalence)**

Given two terms $t, s$ such that $\vdash t, s : \mathsf{A}$, we say that $t$ and $s$ are *context equivalent* iff for every context $C$ such that $\vdash C[\vdash \mathsf{A}] : \mathsf{Str}$ we have that $[\![C[t]]\!](\underline{\epsilon}) = [\![C[s]]\!](\underline{\epsilon})$.

# Contexts

- **Contexts**:

$$C ::= t \mid [\cdot] \mid \lambda x.C \mid Ct \mid tC \mid \mathsf{0}(C) \mid \mathsf{1}(C) \mid \mathsf{tail}(C)$$
$$\mid \mathsf{case}_{\mathsf{A}}(C,t,t,t) \mid \mathsf{case}_{\mathsf{A}}(t,C,C,C) \mid \mathsf{rec}_{\mathsf{A}}(C,t,t,t).$$

- Notice that we are working with *linear* contexts!
- One can easily define a formal system for judgements in the form $\Gamma \vdash C[\vdash \mathsf{A}]$, with the obvious meaning.

**Definition (Context Equivalence)**

Given two terms $t, s$ such that $\vdash t, s : \mathsf{A}$, we say that $t$ and $s$ are *context equivalent* iff for every context $C$ such that $\vdash C[\vdash \mathsf{A}] : \mathsf{Str}$ we have that $[\![C[t]]\!](\underline{\epsilon}) = [\![C[s]]\!](\underline{\epsilon})$.

# Traces

- **Trace**: a sequence of actions $l_1 \cdot l_2 \cdot \ldots \cdot l_n$ such that $l_i \in \{\mathsf{pass}(v), \mathsf{view}(\underline{m}) \mid v \in \mathrm{V}, \underline{m} \in \mathrm{V}^{\mathsf{Str}}\}$.
- **Compatibility**: the trace $\mathsf{T}$ is *compatible* $\mathsf{A}$ iff

$$\mathsf{T} = \begin{cases} \mathsf{view}(\underline{m}) & \text{if } \mathsf{A} = \mathsf{Str}; \\ \mathsf{pass}(v) \cdot \mathsf{S} & \text{if } \mathsf{A} = \mathsf{aB} \to \mathsf{C},\ v \in \mathrm{V}^{\mathsf{B}} \text{ and } \mathsf{S} \text{ is compatible with } \mathsf{C}. \end{cases}.$$

- **Example**: a term of type $\mathsf{A} = \mathsf{aStr} \to \mathsf{bStr} \to \mathsf{Str}$ requires a trace like $\mathsf{T} = \mathsf{pass}(\underline{m}_1) \cdot \mathsf{pass}(\underline{m}_2) \cdot \mathsf{view}(\underline{m})$.

## Definition

Given two terms $t, s$ we say that they are *trace equivalent* (and we write $t \simeq^T s$) if, for all traces $\mathsf{T}$ it holds that $\Pr(t, \mathsf{T}) = \Pr(s, \mathsf{T})$.

# Full Abstraction

**Theorem**

*Context and trace equivalences coincide, i.e., for all $t, s$:*
$t \equiv s \iff t \simeq^T s$.

- This theorem cannot be proved by a simple induction (e.g. on contexts).
- We need a *stronger invariant*...
- ...which in turns requires some auxiliary concepts to be introduced:
  - Term Distributions.
  - Context pairs and their semantics.

# Full Abstraction

**Theorem**

*Context and trace equivalences coincide, i.e., for all $t, s$:*
$t \equiv s \iff t \simeq^T s$.

- This theorem cannot be proved by a simple induction (e.g. on contexts).
- We need a *stronger invariant*...
- ...which in turns requires some auxiliary concepts to be introduced:
  - Term Distributions.
  - Context pairs and their semantics.

# Full Abstraction

**Theorem**

*Context and trace equivalences coincide, i.e., for all $t, s$:*
$t \equiv s \iff t \simeq^T s$.

- This theorem cannot be proved by a simple induction (e.g. on contexts).
- We need a *stronger invariant*...
- ...which in turns requires some auxiliary concepts to be introduced:
  - Term Distributions.
  - Context pairs and their semantics.

# Full Abstraction

**Theorem**

*Context and trace equivalences coincide, i.e., for all $t, s$:*
$t \equiv s \iff t \simeq^T s$.

- This theorem cannot be proved by a simple induction (e.g. on contexts).
- We need a *stronger invariant*...
- ...which in turns requires some auxiliary concepts to be introduced:
  - Term Distributions.
  - Context pairs and their semantics.

# Term Distribution Labelled Semantics

$$\frac{}{\mathcal{T} \Rightarrow^{\epsilon} \mathcal{T}} \qquad \frac{\mathcal{T} \Rightarrow^{\mathsf{S}} \{(\lambda x.t_i)^{p_i}\}}{\mathcal{T} \Rightarrow^{\mathsf{S}\cdot\mathsf{pass}(v)} \{(t_i\{v/x\})^{p_i}\}} \qquad \frac{\mathcal{T} \Rightarrow^{\mathsf{S}} \mathcal{S} \qquad \mathcal{S} \Rrightarrow \mathcal{U}}{\mathcal{T} \Rightarrow^{\mathsf{S}} \mathcal{U}}$$

$$\frac{\mathcal{T} \Rightarrow^{\mathsf{S}} \{(\underline{m_i})^{p_i}\}}{\mathcal{T} \mapsto^{\mathsf{S}\cdot\mathsf{view}(\underline{m})} \sum_{\underline{m_i}=\underline{m}} p_i} \qquad \frac{t \to \{(t_i)^{p_i}\}}{\mathcal{T} + \{(t)^{p}\} \Rrightarrow \mathcal{T} + \{(t_i)^{p\cdot p_i}\}}$$

- **Further Generalization**: same semantics, but on pairs in the form $(C, \mathcal{T})$.
- **The Invariant**: Two distributions $\mathcal{P}, \mathcal{Q}$ on pairs $(C, \mathcal{T})$ are *related* (written $\mathcal{P} \triangledown \mathcal{Q}$) iff

$$\mathcal{P} = \sum_{i=1}^{n} p_i \cdot (C_i, \mathcal{T}_i); \qquad \mathcal{Q} = \sum_{i=1}^{n} p_i \cdot (C_i, \mathcal{S}_i);$$

and $\mathcal{T}_i$ and $\mathcal{S}_i$ are trace equivalent (for $1 \leq i \leq n$).

# Applicative Bisimilarity

- Trace equivalence does not really allow to get rid of the universal quantification on all contexts.
  - Contexts, however, become traces, which have a nicer structure.
- How about **bisimulation** [Abramsky90]?
- Following [DLSangioriAlberti14,CrubilléDL14], one can indeed:
  - See RSLR as a labelled Markov chain.
  - Define a notion of probabilistic bisimulation [LarsenSkou92] over it.
  - Prove congruence (and **soundness**) of bisimilarity, using Howe's method.
- Bisimilarity is **not complete**, because RSLR terms do not have the expressive power which is necessary to simulate all tests [vBMMO04].

# Applicative Bisimilarity

- Trace equivalence does not really allow to get rid of the universal quantification on all contexts.
  - Contexts, however, become traces, which have a nicer structure.
- How about **bisimulation** [Abramsky90]?
- Following [DLSangioriAlberti14,CrubilléDL14], one can indeed:
  - See RSLR as a labelled Markov chain.
  - Define a notion of probabilistic bisimulation [LarsenSkou92] over it.
  - Prove congruence (and **soundness**) of bisimilarity, using Howe's method.
- Bisimilarity is **not complete**, because RSLR terms do not have the expressive power which is necessary to simulate all tests [vBMMO04].

# Applicative Bisimilarity

- Trace equivalence does not really allow to get rid of the universal quantification on all contexts.
  - Contexts, however, become traces, which have a nicer structure.
- How about **bisimulation** [Abramsky90]?
- Following [DLSangioriAlberti14,CrubilléDL14], one can indeed:
  - See RSLR as a labelled Markov chain.
  - Define a notion of probabilistic bisimulation [LarsenSkou92] over it.
  - Prove congruence (and **soundness**) of bisimilarity, using Howe's method.
- Bisimilarity is **not complete**, because RSLR terms do not have the expressive power which is necessary to simulate all tests [vBMMO04].

# Part II

# Metrics

# From Equivalences to Metrics

- **Context Metric**:
  $\delta^C_{\mathsf{A}}(t,s) = \sup_{\vdash C[\vdash \mathsf{A}]:\mathsf{Str}} |[\![C[t]]\!](\underline{\epsilon}) - [\![C[s]]\!](\underline{\epsilon})|$.
  - This is a pseudometric:

$$\delta^C(t,t) = 0;$$
$$\delta^C(t,s) = \delta^C(s,t);$$
$$\delta^C(t,r) \leq \delta^C(t,s) + \delta^C(s,r).$$

  - $\delta^C(t,s) = 0$ iff $t$ and $s$ are contxt equivalent.
- **Trace Metric**: $\delta^T_{\mathsf{A}}(t,s) = \sup_{\mathsf{T}:\mathsf{A}} |\Pr(t,\mathsf{T}) - \Pr(s,\mathsf{T})|$.
- The notion of a trace stays *the same*?
  - *No*!
  - The action $\mathsf{view}(\cdot)$ must be performed on *finite sets of strings* rather than on single strings.
  - This is necessary to manage the $\mathsf{case}$ and $\mathsf{rec}$ constructs.
  - Consider, e.g., contexts in the form $(\lambda x.t)[\cdot]$, where $t$ tests the value of $x$ in many different ways.

# From Equivalences to Metrics

- **Context Metric**:
  $\delta^C_{\mathsf{A}}(t,s) = \sup_{\vdash C[\vdash \mathsf{A}]:\mathsf{Str}} |[\![C[t]]\!](\underline{\epsilon}) - [\![C[s]]\!](\underline{\epsilon})|$.
  - This is a pseudometric:

$$\delta^C(t,t) = 0;$$
$$\delta^C(t,s) = \delta^C(s,t);$$
$$\delta^C(t,r) \leq \delta^C(t,s) + \delta^C(s,r).$$

  - $\delta^C(t,s) = 0$ iff $t$ and $s$ are contxt equivalent.
- **Trace Metric**: $\delta^T_{\mathsf{A}}(t,s) = \sup_{\mathsf{T}:\mathsf{A}} |\Pr(t,\mathsf{T}) - \Pr(s,\mathsf{T})|$.
- The notion of a trace stays *the same*?
  - *No*!
  - The action $\mathsf{view}(\cdot)$ must be performed on *finite sets of strings* rather than on single strings.
  - This is necessary to manage the $\mathsf{case}$ and $\mathsf{rec}$ constructs.
  - Consider, e.g., contexts in the form $(\lambda x.t)[\cdot]$, where $t$ tests the value of $x$ in many different ways.

# From Equivalences to Metrics

- **Context Metric**:
  $\delta^C_{\mathsf{A}}(t,s) = \sup_{\vdash C[\vdash \mathsf{A}]:\mathsf{Str}} |[\![C[t]]\!](\underline{\epsilon}) - [\![C[s]]\!](\underline{\epsilon})|$.
  - This is a pseudometric:

$$\delta^C(t,t) = 0;$$
$$\delta^C(t,s) = \delta^C(s,t);$$
$$\delta^C(t,r) \leq \delta^C(t,s) + \delta^C(s,r).$$

  - $\delta^C(t,s) = 0$ iff $t$ and $s$ are contxt equivalent.
- **Trace Metric**: $\delta^T_{\mathsf{A}}(t,s) = \sup_{\mathsf{T}:\mathsf{A}} |\Pr(t,\mathsf{T}) - \Pr(s,\mathsf{T})|$.
- The notion of a trace stays *the same*?
  - *No*!
  - The action $\mathsf{view}(\cdot)$ must be performed on *finite sets of strings* rather than on single strings.
  - This is necessary to manage the $\mathsf{case}$ and $\mathsf{rec}$ constructs.
  - Consider, e.g., contexts in the form $(\lambda x.t)[\cdot]$, where $t$ tests the value of $x$ in many different ways.

# From Equivalences to Metrics

- **Context Metric**:
  $\delta^C_{\mathsf{A}}(t,s) = \sup_{\vdash C[\vdash \mathsf{A}]:\mathsf{Str}} |[\![C[t]]\!](\underline{\epsilon}) - [\![C[s]]\!](\underline{\epsilon})|$.
  - This is a pseudometric:

$$\delta^C(t,t) = 0;$$
$$\delta^C(t,s) = \delta^C(s,t);$$
$$\delta^C(t,r) \leq \delta^C(t,s) + \delta^C(s,r).$$

  - $\delta^C(t,s) = 0$ iff $t$ and $s$ are contxt equivalent.
- **Trace Metric**: $\delta^T_{\mathsf{A}}(t,s) = \sup_{\mathsf{T}:\mathsf{A}} |\Pr(t,\mathsf{T}) - \Pr(s,\mathsf{T})|$.
- The notion of a trace stays *the same*?
  - *No*!
  - The action $\mathsf{view}(\cdot)$ must be performed on *finite sets of strings* rather than on single strings.
  - This is necessary to manage the $\mathsf{case}$ and $\mathsf{rec}$ constructs.
  - Consider, e.g., contexts in the form $(\lambda x.t)[\cdot]$, where $t$ tests the value of $x$ in many different ways.

# From Equivalences to Metrics

- **Context Metric**:
  $\delta^C_{\mathsf{A}}(t,s) = \sup_{\vdash C[\vdash \mathsf{A}]:\mathsf{Str}} |[\![C[t]]\!](\underline{\epsilon}) - [\![C[s]]\!](\underline{\epsilon})|$.
  - This is a pseudometric:

$$\delta^C(t,t) = 0;$$
$$\delta^C(t,s) = \delta^C(s,t);$$
$$\delta^C(t,r) \leq \delta^C(t,s) + \delta^C(s,r).$$

  - $\delta^C(t,s) = 0$ iff $t$ and $s$ are contxt equivalent.
- **Trace Metric**: $\delta^T_{\mathsf{A}}(t,s) = \sup_{\mathsf{T}:\mathsf{A}} |\Pr(t,\mathsf{T}) - \Pr(s,\mathsf{T})|$.
- The notion of a trace stays *the same*?
  - *No*!
  - The action $\mathsf{view}(\cdot)$ must be performed on *finite sets of strings* rather than on single strings.
  - This is necessary to manage the $\mathsf{case}$ and $\mathsf{rec}$ constructs.
  - Consider, e.g., contexts in the form $(\lambda x.t)[\cdot]$, where $t$ tests the value of $x$ in many different ways.

# From Equivalences to Metrics

**Theorem (Full Abstraction)**

*Context and trace metrics coincide, i.e., for all $t, s$:*
$\delta^C(t, s) = \delta^T(t, s)$.

- How do we get this result?
- The structure of the proof is similar to the one for equivalences.
  - What plays the rôle of $\nabla$ is $\nabla^\varepsilon$, a notion of equivalence parametrized by $\varepsilon > 0$.

# Part III

# Parametrized Equivalences

# Computational Indistinguishability (CI)

## Definition

Two distribution ensembles $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ and $\{\mathcal{E}_n\}_{n\in\mathbb{N}}$ (where both $\mathcal{D}_n$ and $\mathcal{E}_n$ are distributions on binary strings) are said to be *computationally indistinguishable* iff for every PPT algorithm $\mathcal{A}$ the following quantity is a negligible function of $n \in \mathbb{N}$:

$$\left|\Pr_{x\leftarrow\mathcal{D}_n}(\mathcal{A}(x,1^n)=\epsilon) - \Pr_{x\leftarrow\mathcal{E}_n}(\mathcal{A}(x,1^n)=\epsilon)\right|.$$

- This is a key notion in modern cryptography. As an example, pseudorandomness is a form of CI.
- The algorithm $\mathcal{A}$ can be assumed to sample from $x$ just *once* without altering the definition itself (provided the two involved ensembles are efficiently computable)

# Computational Indistinguishability (CI)

## Definition

Two distribution ensembles $\{\mathcal{D}_n\}_{n\in\mathbb{N}}$ and $\{\mathcal{E}_n\}_{n\in\mathbb{N}}$ (where both $\mathcal{D}_n$ and $\mathcal{E}_n$ are distributions on binary strings) are said to be *computationally indistinguishable* iff for every PPT algorithm $\mathcal{A}$ the following quantity is a negligible function of $n \in \mathbb{N}$:

$$\left|\Pr_{x\leftarrow\mathcal{D}_n}(\mathcal{A}(x,1^n)=\epsilon) - \Pr_{x\leftarrow\mathcal{E}_n}(\mathcal{A}(x,1^n)=\epsilon)\right|.$$

- This is a key notion in modern cryptography. As an example, pseudorandomness is a form of CI.
- The algorithm $\mathcal{A}$ can be assumed to sample from $x$ just *once* without altering the definition itself (provided the two involved ensembles are efficiently computable)

# Parametric Context Equivalence (PCE)

- **Two Observataions**:
  1. While context distance is an *absolute* notion of distance, CI depends on a parameter $n$, the so-called *security parameter*.
  2. In computational indistinguishability, one can compare distributions over *strings*, while the context distance can evaluate how far terms of *arbitrary* types are.
- One is lead, quite naturally, to the following definition:

**Definition (Parametric Context Equivalence)**

Given two terms $t, s$ such that $\vdash t, s : \mathsf{aStr} \to \mathsf{A}$, we say that $t$ and $s$ are *parametrically context equivalent* iff for every context $C$ such that $\vdash C[\vdash \mathsf{A}] : \mathsf{Str}$ we have that $|[\![C[t1^{\mathbf{n}}]]\!](\underline{\epsilon}) - [\![C[s1^{\mathbf{n}}]]\!](\underline{\epsilon})|$ is negligible in $\mathbf{n}$.

# Parametric Context Equivalence (PCE)

- **Two Observataions**:
  1. While context distance is an *absolute* notion of distance, CI depends on a parameter $n$, the so-called *security parameter*.
  2. In computational indistinguishability, one can compare distributions over *strings*, while the context distance can evaluate how far terms of *arbitrary* types are.
- One is lead, quite naturally, to the following definition:

**Definition (Parametric Context Equivalence)**

Given two terms $t, s$ such that $\vdash t, s : \mathsf{aStr} \to \mathsf{A}$, we say that $t$ and $s$ are *parametrically context equivalent* iff for every context $C$ such that $\vdash C[\vdash \mathsf{A}] : \mathsf{Str}$ we have that $|[\![C[t1^{\mathbf{n}}]]\!](\underline{\epsilon}) - [\![C[s1^{\mathbf{n}}]]\!](\underline{\epsilon})|$ is negligible in $\mathbf{n}$.

# CI vs. PCE

**Theorem**

*Let $t, s$ be two terms of type $\mathsf{aStr} \to \mathsf{Str}$. Then $t, s$ are parametric context equivalent iff the distribution ensembles $\{[\![t1^{\mathbf{n}}]\!]\}_{\mathbf{n}\in\mathbb{N}}$ and $\{[\![s1^{\mathbf{n}}]\!]\}_{\mathbf{n}\in\mathbb{N}}$ are computationally indistinguishable.*

- This only holds for **base types**.
- How about **higher-order types**?
  - The literature of cryptography does not offer a precise definition of *higher-order* CI.
  - Sometimes it is necessary to enforce linear access to cryptographic primitives. (e.g. security for passive adversaries).
  - Sometime, taking multiple accesses is necessary (e.g. in pseudorandom functions).

# CI vs. PCE

**Theorem**

*Let $t, s$ be two terms of type $\mathsf{aStr} \to \mathsf{Str}$. Then $t, s$ are parametric context equivalent iff the distribution ensembles $\{[\![t1^{\mathbf{n}}]\!]\}_{\mathbf{n}\in\mathbb{N}}$ and $\{[\![s1^{\mathbf{n}}]\!]\}_{\mathbf{n}\in\mathbb{N}}$ are computationally indistinguishable.*

- This only holds for **base types**.
- How about **higher-order types**?
  - The literature of cryptography does not offer a precise definition of *higher-order* CI.
  - Sometimes it is necessary to enforce linear access to cryptographic primitives. (e.g. security for passive adversaries).
  - Sometime, taking multiple accesses is necessary (e.g. in pseudorandom functions).

# Parametric Trace Equivalence

- Contexts test families of terms rather than terms.
- The action $\mathsf{view}(\cdot)$ takes in input closed $\mathsf{RSLR}$ terms $\mathtt{D} : \mathsf{aStr} \to \mathsf{Str}$, that we call *distinguishers*.

**Definition**

Two terms $t, s : \mathsf{aStr} \to \mathsf{A}$ are *parametrically trace equivalent*, and we write $t \simeq_n^T s$, iff for every trace $\mathsf{T}$ which is parametrically compatible with $\mathsf{A}$, there is a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}_{[0,1]}$ such that $|\Pr(t, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T}) - \Pr(s, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T})| \leq \varepsilon(\mathbf{n})$.

**Theorem**

*Parametric trace equivalence and parametric context equivalence coincide.*

# Parametric Trace Equivalence

- Contexts test families of terms rather than terms.
- The action $\mathsf{view}(\cdot)$ takes in input closed $\mathsf{RSLR}$ terms $\mathtt{D} : \mathsf{aStr} \to \mathsf{Str}$, that we call *distinguishers*.

**Definition**

Two terms $t, s : \mathsf{aStr} \to \mathsf{A}$ are *parametrically trace equivalent*, and we write $t \simeq_n^T s$, iff for every trace $\mathsf{T}$ which is parametrically compatible with $\mathsf{A}$, there is a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}_{[0,1]}$ such that $|\Pr(t, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T}) - \Pr(s, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T})| \leq \varepsilon(\mathbf{n})$.

**Theorem**

*Parametric trace equivalence and parametric context equivalence coincide.*

# Parametric Trace Equivalence

- Contexts test families of terms rather than terms.
- The action $\mathsf{view}(\cdot)$ takes in input closed $\mathsf{RSLR}$ terms $\mathtt{D} : \mathsf{aStr} \to \mathsf{Str}$, that we call *distinguishers*.

## Definition

Two terms $t, s : \mathsf{aStr} \to \mathsf{A}$ are *parametrically trace equivalent*, and we write $t \simeq_n^T s$, iff for every trace $\mathsf{T}$ which is parametrically compatible with $\mathsf{A}$, there is a negligible function $\varepsilon : \mathbb{N} \to \mathbb{R}_{[0,1]}$ such that $|\Pr(t, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T}) - \Pr(s, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T})| \leq \varepsilon(\mathbf{n})$.

## Theorem

*Parametric trace equivalence and parametric context equivalence coincide.*

# An Example

- Let's consider these two terms:

$$t = \lambda n.(\lambda k.\lambda x.\lambda y.\mathsf{ENC}(xn, k))\mathsf{GEN}(n);$$
$$s = \lambda n.(\lambda k.\lambda x.\lambda y.\mathsf{ENC}(yn, k))\mathsf{GEN}(n);$$
$$\vdash t, s : \Box\mathsf{Str} \to (\Box\mathsf{Str} \to \mathsf{Str}) \to (\Box\mathsf{Str} \to \mathsf{Str}) \to \mathsf{Str};$$

  where $\mathsf{ENC}(x, k) = x \oplus G(k)$ and $G$ is a pseudorandom generator.
- We know that $G$ is pseudorandom iff, given a source of randomness $R$, it holds that $G \simeq R$.
- We would like to show that if $G \simeq R$, then $t \simeq s$.

# An Example



***THEOREM 3.18*** *If $G$ is a pseudorandom generator, then Construction 3.17 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

**PROOF** Let $\Pi$ denote Construction 3.17. We show that $\Pi$ satisfies Definition 3.8. Namely, we show that for any probabilistic polynomial-time adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}$ such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n). \tag{3.2}$$

The intuition is that if $\Pi$ used a uniform pad in place of the pseudorandom pad $G(k)$, then the resulting scheme would be identical to the one-time pad encryption scheme and $\mathcal{A}$ would be unable to correctly guess which message was encrypted with probability any better than 1/2. Thus, if Equation (3.2) does not hold then $\mathcal{A}$ must implicitly be distinguishing the output of $G$ from a random string. We make this explicit by showing a *reduction*; namely, by showing how to use $\mathcal{A}$ to construct an efficient distinguisher $D$, with the property that $D$'s ability to distinguish the output of $G$ from a uniform string is directly related to $\mathcal{A}$'s ability to determine which message was encrypted by $\Pi$. Security of $G$ then implies security of $\Pi$.

Let $\mathcal{A}$ be an arbitrary PPT adversary. We construct a distinguisher $D$ that takes a string $w$ as input, and whose goal is to determine whether $w$ was chosen uniformly (i.e., $w$ is a "random string") or whether $w$ was generated by choosing a uniform $k$ and computing $w := G(k)$ (i.e., $w$ is a "pseudorandom string"). We construct $D$ so that it emulates the eavesdropping experiment for $\mathcal{A}$, as described below, and observes whether $\mathcal{A}$ succeeds or not. If $\mathcal{A}$ succeeds then $D$ guesses that $w$ must be a pseudorandom string, while if $\mathcal{A}$ does not succeed then $D$ guesses that $w$ is a random string. In detail:

> **Distinguisher** $D$:
> $D$ is given as input a string $w \in \{0,1\}^{\ell(n)}$. (We assume that $n$ can be determined from $\ell(n)$.)
>
> 1. Run $\mathcal{A}(1^n)$ to obtain a pair of messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$.
> 2. Choose a uniform bit $b \in \{0,1\}$. Set $c := w \oplus m_b$.
> 3. Give $c$ to $\mathcal{A}$ and obtain output $b'$. Output 1 if $b' = b$, and output 0 otherwise.

$D$ clearly runs in polynomial time (assuming $\mathcal{A}$ does).

Before analyzing the behavior of $D$, we define a modified encryption scheme $\widetilde{\Pi} = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ that is exactly the one-time pad encryption scheme, except that we now incorporate a security parameter that determines the length of the message to be encrypted. That is, $\widetilde{\mathsf{Gen}}(1^n)$ outputs a uniform key $k$ of length $\ell(n)$, and the encryption of message $m \in 2^{\ell(n)}$ using key $k \in \{0,1\}^{\ell(n)}$



is the ciphertext $c := k \oplus m$. (Decryption can be performed as usual, but is inessential to what follows.) Perfect secrecy of the one-time pad implies

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] = \frac{1}{2}. \tag{3.3}$$

To analyze the behavior of $D$, the main observations are:

1. If $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$, then the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0,1\}^{\ell(n)}$ is uniform. Since $D$ outputs 1 exactly when $\mathcal{A}$ succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))

$$\Pr_{w \leftarrow \{0,1\}^{\ell(n)}}[D(w) = 1] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] = \frac{1}{2}. \tag{3.4}$$

   (The subscript on the first probability just makes explicit that $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$ there.)

2. If $w$ is instead generated by choosing uniform $k \in \{0,1\}^n$ and then setting $w := G(k)$, the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$. This is because $\mathcal{A}$, when run as a subroutine by $D$, is now given a ciphertext $c = w \oplus m_b$ where $w = G(k)$ for a uniform $k \in \{0,1\}^n$. Thus,

$$\Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]. \tag{3.5}$$

Since $G$ is a pseudorandom generator (and since $D$ runs in polynomial time), we know there is a negligible function $\mathsf{negl}$ such that

$$\left|\Pr_{w \leftarrow \{0,1\}^{\ell(n)}}[D(w) = 1] - \Pr_{k \leftarrow \{0,1\}^n}[D(G(k)) = 1]\right| \le \mathsf{negl}(n).$$

Using Equations (3.4) and (3.5), we thus see that

$$\left|\frac{1}{2} - \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]\right| \le \mathsf{negl}(n),$$

which implies $\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n)$. Since $\mathcal{A}$ was an arbitrary PPT adversary, this completes the proof that $\Pi$ has indistinguishable encryptions in the presence of an eavesdropper. ∎

It is easy to get lost in the details of the proof and wonder whether anything has been gained as compared to the one-time pad; after all, the one-time pad also encrypts an $\ell$-bit message by XORing it with an $\ell$-bit string! The point of the construction, of course, is that the $\ell$-bit string $G(k)$ can be *much*

# An Example



***THEOREM 3.18*** *If $G$ is a pseudorandom generator, then Construction 3.17 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

**PROOF** Let $\Pi$ denote Construction 3.17. We show that $\Pi$ satisfies Definition 3.8. Namely, we show that for any probabilistic polynomial-time adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}$ such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n). \tag{3.2}$$

The intuition is that if $\Pi$ used a uniform pad in place of the pseudorandom



is the ciphertext $c := k \oplus m$. (Decryption can be performed as usual, but is inessential to what follows.) Perfect secrecy of the one-time pad implies

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] = \frac{1}{2}. \tag{3.3}$$

To analyze the behavior of $D$, the main observations are:

1. If $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$, then the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0,1\}^{\ell(n)}$ is uni-

If $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$, then the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0,1\}^{\ell(n)}$ is uniform. Since $D$ outputs 1 exactly when $\mathcal{A}$ succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))

$$\Pr_{w \in \{0,1\}^{\ell(n)}}[D(w) = 1] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] = \frac{1}{2} \tag{3.4}$$

1. Run $\mathcal{A}(1^n)$ to obtain a pair of messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$.
2. Choose a uniform bit $b \in \{0,1\}$. Set $c := w \oplus m_b$.
3. Give $c$ to $\mathcal{A}$ and obtain output $b'$. Output 1 if $b' = b$, and output 0 otherwise.

$D$ clearly runs in polynomial time (assuming $\mathcal{A}$ does).

Before analyzing the behavior of $D$, we define a modified encryption scheme $\widetilde{\Pi} = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ that is exactly the one-time pad encryption scheme, except that we now incorporate a security parameter that determines the length of the message to be encrypted. That is, $\widetilde{\mathsf{Gen}}(1^n)$ outputs a uniform key $k$ of length $\ell(n)$, and the encryption of message $m \in 2^{\ell(n)}$ using key $k \in \{0,1\}^{\ell(n)}$

$$\left|\frac{1}{2} - \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]\right| \le \mathsf{negl}(n),$$

which implies $\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n)$. Since $\mathcal{A}$ was an arbitrary PPT adversary, this completes the proof that $\Pi$ has indistinguishable encryptions in the presence of an eavesdropper. ∎

It is easy to get lost in the details of the proof and wonder whether anything has been gained as compared to the one-time pad; after all, the one-time pad also encrypts an $\ell$-bit message by XORing it with an $\ell$-bit string! The point of the construction, of course, is that the $\ell$-bit string $G(k)$ can be *much*

# An Example



***THEOREM 3.18*** *If $G$ is a pseudorandom generator, then Construction 3.17 is a fixed-length private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper.*

**PROOF** Let $\Pi$ denote Construction 3.17. We show that $\Pi$ satisfies Definition 3.8. Namely, we show that for any probabilistic polynomial-time adversary $\mathcal{A}$ there is a negligible function $\mathsf{negl}$ such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n). \tag{3.2}$$

The intuition is that if $\Pi$ used a uniform pad in place of the pseudorandom



is the ciphertext $c := k \oplus m$. (Decryption can be performed as usual, but is inessential to what follows.) Perfect secrecy of the one-time pad implies

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] = \frac{1}{2}. \tag{3.3}$$

To analyze the behavior of $D$, the main observations are:

1. If $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$, then the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0,1\}^{\ell(n)}$ is uni-

If $w$ is chosen uniformly from $\{0,1\}^{\ell(n)}$, then the view of $\mathcal{A}$ when run as a subroutine by $D$ is distributed identically to the view of $\mathcal{A}$ in experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0,1\}^{\ell(n)}$ is uniform. Since $D$ outputs 1 exactly when $\mathcal{A}$ succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))

$$\Pr_{w \in \{0,1\}^{\ell(n)}}[D(w) = 1] = \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1\right] = \frac{1}{2} \tag{3.4}$$

1. Run $\mathcal{A}(1^n)$ to obtain a pair of messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$.
2. Choose a uniform bit $b \in \{0,1\}$. Set $c := w \oplus m_b$.
3. Give $c$ to $\mathcal{A}$ and obtain output $b'$. Output 1 if $b' = b$, and output 0 otherwise.

$D$ clearly runs in polynomial time (assuming $\mathcal{A}$ does).

Before analyzing the behavior of $D$, we define a modified encryption scheme $\widetilde{\Pi} = (\widetilde{\mathsf{Gen}}, \widetilde{\mathsf{Enc}}, \widetilde{\mathsf{Dec}})$ that is exactly the one-time pad encryption scheme, except that we now incorporate a security parameter that determines the length of the message to be encrypted. That is, $\widetilde{\mathsf{Gen}}(1^n)$ outputs a uniform key $k$ of length $\ell(n)$, and the encryption of message $m \in 2^{\ell(n)}$ using key $k \in \{0,1\}^{\ell(n)}$

$$\left|\frac{1}{2} - \Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right]\right| \le \mathsf{negl}(n),$$

which implies $\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \le \frac{1}{2} + \mathsf{negl}(n)$. Since $\mathcal{A}$ was an arbitrary PPT adversary, this completes the proof that $\Pi$ has indistinguishable encryptions in the presence of an eavesdropper. ∎

It is easy to get lost in the details of the proof and wonder whether anything has been gained as compared to the one-time pad; after all, the one-time pad also encrypts an $\ell$-bit message by XORing it with an $\ell$-bit string! The point of the construction, of course, is that the $\ell$-bit string $G(k)$ can be *much*

# An Example

- Suppose now that $t \not\simeq s$.
- Then there exists a trace $\mathsf{T}$ such that: $|\Pr(t, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T}) - \Pr(s, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T})|$ is not negligible, where $\mathsf{T} = \mathsf{pass}(v) \cdot \mathsf{pass}(u) \cdot \mathsf{view}(\mathtt{D})$.
- If this is true we can easily construct a trace $\mathsf{S}$ such that $|\Pr(G, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{S}) - \Pr(R, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{S})|$ is itself not negligible.
  - As a consequence, $G \not\simeq^T R$;
  - And this is a contradiction.
  - The trace $\mathsf{S}$ only consists of a distinguisher $\mathtt{E}$, which can be built out of $v$, $u$ and $\mathtt{D}$:

$$\mathtt{E} = \lambda x.\mathtt{if}\ \mathsf{rand}\ \mathtt{then}\ (\mathtt{D}(x \oplus v(\mathtt{i}_\ell x)))\ \mathtt{else}\ (\mathtt{D}(x \oplus u(\mathtt{i}_\ell x)))$$

# An Example

- Suppose now that $t \not\simeq s$.
- Then there exists a trace $\mathsf{T}$ such that: $|\Pr(t, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T}) - \Pr(s, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{T})|$ is not negligible, where $\mathsf{T} = \mathsf{pass}(v) \cdot \mathsf{pass}(u) \cdot \mathsf{view}(\mathtt{D})$.
- If this is true we can easily construct a trace $\mathsf{S}$ such that $|\Pr(G, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{S}) - \Pr(R, \mathsf{pass}(1^{\mathbf{n}}) \cdot \mathsf{S})|$ is itself not negligible.
  - As a consequence, $G \not\simeq^T R$;
  - And this is a contradiction.
  - The trace $\mathsf{S}$ only consists of a distinguisher $\mathtt{E}$, which can be built out of $v$, $u$ and $\mathtt{D}$:

$$\mathtt{E} = \lambda x.\mathtt{if}\ \mathsf{rand}\ \mathtt{then}\ (\mathtt{D}(x \oplus v(\mathtt{i}_\ell x)))\ \mathtt{else}\ (\mathtt{D}(x \oplus u(\mathtt{i}_\ell x)))$$

# Challenges

1. **Handling Pairs**
   - If we endow the language with a tensor product $A \otimes B$, the structure of traces becomes much more complicated.
   - Projection actions are simply not sound!
   - We would be able to handle *protocols*, and not only primitives.
2. **Handling Copying**
   - Linear contexts are not adequate when modeling adversaries having access to oracles.
   - RSLR is not complete for type-2 polytime complexity!
3. **Higher-Order Pseudorandomness?**
   - Cryptographers are only interested in *ground* and *first-order* pseudorandomness.
   - Parametric equivalences make sense at every type. But they are too lax.
   - But, again, what if we have a calculus which is complete for higher-order polynomial time?

# Challenges

1. **Handling Pairs**
   - If we endow the language with a tensor product $A \otimes B$, the structure of traces becomes much more complicated.
   - Projection actions are simply not sound!
   - We would be able to handle *protocols*, and not only primitives.
2. **Handling Copying**
   - Linear contexts are not adequate when modeling adversaries having access to oracles.
   - RSLR is not complete for type-2 polytime complexity!
3. **Higher-Order Pseudorandomness?**
   - Cryptographers are only interested in *ground* and *first-order* pseudorandomness.
   - Parametric equivalences make sense at every type. But they are too lax.
   - But, again, what if we have a calculus which is complete for higher-order polynomial time?

# Challenges

1. **Handling Pairs**
   - If we endow the language with a tensor product $A \otimes B$, the structure of traces becomes much more complicated.
   - Projection actions are simply not sound!
   - We would be able to handle *protocols*, and not only primitives.
2. **Handling Copying**
   - Linear contexts are not adequate when modeling adversaries having access to oracles.
   - RSLR is not complete for type-2 polytime complexity!
3. **Higher-Order Pseudorandomness?**
   - Cryptographers are only interested in *ground* and *first-order* pseudorandomness.
   - Parametric equivalences make sense at every type. But they are too lax.
   - But, again, what if we have a calculus which is complete for higher-order polynomial time?

# Why Pairs?

$$
\begin{array}{ccc}
A & \rightarrow & B \\
B & \rightarrow & C \\
C & \rightarrow & A \\
A & \rightarrow & C
\end{array}
$$

$$
\begin{array}{l}
t_A : \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str}) \\
t_B : \mathsf{Str} \multimap \mathsf{Str} \\
t_C : \mathsf{Str} \multimap \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str})
\end{array}
$$

$$C[\cdot_A, \cdot_B, \cdot_C]$$

# Why Pairs?

$$
\begin{array}{ccc}
A & \rightarrow & B \\
B & \rightarrow & C \\
C & \rightarrow & A \\
A & \rightarrow & C
\end{array}
$$

$$
\begin{aligned}
t_A &: \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str}) \\
t_B &: \mathsf{Str} \multimap \mathsf{Str} \\
t_C &: \mathsf{Str} \multimap \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str})
\end{aligned}
$$

$$C[\cdot_A, \cdot_B, \cdot_C]$$

# Why Pairs?

$$\begin{array}{ccc} A & \rightarrow & B \\ B & \rightarrow & C \\ C & \rightarrow & A \\ A & \rightarrow & C \end{array}$$

$$\begin{array}{l} t_A : \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str}) \\ t_B : \mathsf{Str} \multimap \mathsf{Str} \\ t_C : \mathsf{Str} \multimap \mathsf{Str} \otimes (\mathsf{Str} \multimap \mathsf{Str}) \end{array}$$

$$C[\cdot_A, \cdot_B, \cdot_C]$$

# Why Modalities?

**Pseudorandom Functions**:

$$\mathsf{Str} \multimap \mathsf{Str} \otimes !_p(\mathsf{Str} \multimap \mathsf{Str})$$

**CPA-IND**

$$\mathsf{Str} \multimap \mathsf{Str} \otimes !_p(\mathsf{Str} \multimap \mathsf{Str}) \ \otimes \ (\mathsf{Str} \otimes \mathsf{Str} \multimap \mathsf{Str})$$

# Why Modalities?

**Pseudorandom Functions**:

$$\mathsf{Str} \multimap \mathsf{Str} \otimes !_p(\mathsf{Str} \multimap \mathsf{Str})$$

**CPA-IND**

$$\mathsf{Str} \multimap \mathsf{Str} \otimes !_p(\mathsf{Str} \multimap \mathsf{Str}) \ \otimes \ (\mathsf{Str} \otimes \mathsf{Str} \multimap \mathsf{Str})$$

# Thank You!

Questions?