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

\[
\text{Property}(\Phi) \overset{\text{def}}{=} (\forall \mathcal{D}. \text{PPT}(\mathcal{D}) \Rightarrow \Pr(\text{Success}(\mathcal{D} | \Phi)) = \text{negl}(n)).
\]

Assumption(\Phi) \Rightarrow \text{Security}(\Pi)

Examples

OneWay(f) \Rightarrow \text{PRG}(G)

\text{PRG}(G) \Rightarrow \text{IND}(\Pi)
Proofs by Reduction

\[ \text{Property}(\Phi) \overset{\text{def}}{=} (\forall \mathcal{D}. \text{PPT}(\mathcal{D}) \Rightarrow \Pr(\text{Success}(\mathcal{D} \mid \Phi)) = \text{negl}(n)) . \]

**Assumption(\Phi) \Rightarrow \text{Security}(\Pi)**

**Examples**

\[
\begin{align*}
\text{OneWay}(f) & \Rightarrow \text{PRG}(G) \\
\text{PRG}(G) & \Rightarrow \text{IND}(\Pi)
\end{align*}
\]
Proofs by Reduction

\[
\text{Property}(\Phi) \overset{\text{def}}{=} (\forall \mathcal{D}. PPT(\mathcal{D}) \Rightarrow \Pr(\text{Success}(\mathcal{D} | \Phi)) = \text{negl}(n)).
\]

Assumption$(\Phi)$ $\Rightarrow$ Security$(\Pi)$

Examples

OneWay$(f)$ $\Rightarrow$ PRG$(G)$
PRG$(G)$ $\Rightarrow$ IND$(\Pi)$
Proofs by Reduction

\[ \text{Property}(\Phi) \overset{\text{def}}{=} (\forall \mathcal{D}. \text{PPT}(\mathcal{D}) \Rightarrow \Pr(\text{Success}(\mathcal{D} | \Phi)) = \text{negl}(n)). \]

\[ \text{Assumption}(\Phi) \Rightarrow \text{Security}(\Pi) \]

Examples

\[ \text{OneWay}(f) \Rightarrow \text{PRG}(G) \]

\[ \text{PRG}(G) \Rightarrow \text{IND}(\Pi) \]
Proofs by Reduction

\[ \text{Property}(\Phi) \overset{\text{def}}{=} (\forall D. PPT(D) \Rightarrow \Pr(\text{Success}(D | \Phi)) = \negl(n)). \]

**Assumption**\((\Phi) \Rightarrow \text{Security}(\Pi)\)

**Examples**

\[ \text{OneWay}(f) \Rightarrow \text{PRG}(G) \]
\[ \text{PRG}(G) \Rightarrow \text{IND}(\Pi) \]
Proofs by Reduction
Proofs by Reduction
Proofs by Reduction
Proofs by Reduction
Game-Based Proofs
Game-Based Proofs

\[
M \approx D \quad \approx \quad L \approx E
\]

\[
N \approx D \quad \approx \quad P \approx E
\]
An Alternative Viewpoint
An Alternative Viewpoint
1. Introduce a $\lambda$-calculus RSLR, and prove it complete for probabilistic polynomial-time computation [Zhang10,DLPT11].

\[
\begin{array}{c}
M \\
\end{array} \xrightarrow{\sim} D[M]
\]

2. Define 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:**

\[
\begin{align*}
    t &::= x \mid v \mid 0(t) \mid 1(t) \mid \text{tail}(t) \mid tt \mid \text{case}_A(t, t, t, t) \\
    &\quad \mid \text{rec}_A(t, t, t) \mid \text{rand}; \\
    v &::= m \mid \lambda x : aA.t;
\end{align*}
\]

- **Types:**

\[
A ::= \text{Str} \mid \Box A \rightarrow A \mid \Box A \rightarrow A.
\]

- **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:**
  
  \[
  t ::= x \mid v \mid 0(t) \mid 1(t) \mid \text{tail}(t) \mid tt \mid \text{case}_A(t, t, t, t) \mid \\
  \text{rec}_A(t, t, t, t) \mid \text{rand};
  \]
  
  \[
  v ::= m \mid \lambda x : aA.t;
  \]

- **Types:** \( A ::= \text{Str} \mid \Box A \rightarrow A \mid \square A \rightarrow A. \)

- **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:**

\[
t ::= x \mid v \mid 0(t) \mid 1(t) \mid \text{tail}(t) \mid tt \mid \text{case}_A(t, t, t, t) \mid \\
\text{rec}_A(t, t, t, t) \mid \text{rand};
\]

\[
v ::= m \mid \lambda x : aA.t;
\]

- **Types:** \( A ::= \text{Str} \mid \square A \rightarrow A \mid \blacksquare A \rightarrow A. \)

- **Type System’s Main Ingredients:**

  - Higher-order terms *cannot* be duplicated, while strings *can*.
  - Recursion can only be performed on \( \square \)-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:

\[ t ::= x \mid v \mid 0(t) \mid 1(t) \mid \text{tail}(t) \mid tt \mid \text{case}_A(t, t, t) \mid \text{rec}_A(t, t, t) \mid \text{rand}; \]

\[ v ::= m \mid \lambda x : aA.t; \]

Types: \( A ::= \text{Str} \mid \Box A \to A \mid \blacksquare A \to A. \)

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:** $\mathcal{R} \subseteq T^A \times T^A$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90, DanosHarmer00, ...].
  - Too discriminating (e.g., $tt$ vs. $tt \oplus^e ff$).

- **A metric:** $\delta : T^A \times T^A \to \mathbb{R}$.
  - 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:** $\mathcal{R} \subseteq T^{N \to A} \times T^{N \to A}$.
  - The kind of concept one needs in cryptography.
How Should we Compare Terms in RSLR?

- **An equivalence relation**: $\mathcal{R} \subseteq T^A \times T^A$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90,DanosHarmer00,...].
  - Too discriminating (e.g., $tt$ vs. $tt \oplus^\epsilon ff$).

- **A metric**: $\delta : T^A \times T^A \to \mathbb{R}$.
  - 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**: $\mathcal{R} \subseteq T^{N\to A} \times T^{N\to A}$.
  - The kind of concept one needs in cryptography.
How Should we Compare Terms in RSLR?

- **An equivalence relation:** $\mathcal{R} \subseteq T^A \times T^A$.
  - This is the traditional approach.
  - It is pervasive, even in probabilistic $\lambda$-calculi [JonesPlotkin90,DanosHarmer00,...].
  - Too discriminating (e.g., $tt$ vs. $tt \oplus^e ff$).

- **A metric:** $\delta : T^A \times T^A \to \mathbb{R}$.
  - 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:**
  $\mathcal{R} \subseteq T^{N\rightarrow A} \times T^{N\rightarrow A}$.
  - The kind of concept one needs in cryptography.
Part I

Equivalences
Contexts

- **Contexts:**

  
  \[ C ::= t \mid [\cdot] \mid \lambda x.C \mid Ct \mid tC \mid 0(C) \mid 1(C) \mid \text{tail}(C) \]

  \[ \mid \text{case}_A(C, t, t, t) \mid \text{case}_A(t, C, C, C) \mid \text{rec}_A(C, t, t, t). \]

- Notice that we are working with *linear* contexts!

- One can easily define a formal system for judgements in the form \( \Gamma \vdash C[\vdash A] \), with the obvious meaning.

---

**Definition (Context Equivalence)**

Given two terms \( t, s \) such that \( \vdash t, s : A \), we say that \( t \) and \( s \) are context equivalent iff for every context \( C \) such that \( \vdash C[\vdash A] : \text{Str} \) we have that \( \llbracket C[t] \rrbracket(\epsilon) = \llbracket C[s] \rrbracket(\epsilon) \).
Contexts

▶ Contexts:

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

▶ Notice that we are working with linear contexts!

▶ One can easily define a formal system for judgements in the form \( \Gamma \vdash C[\vdash A] \), with the obvious meaning.

Definition (Context Equivalence)

Given two terms \( t, s \) such that \( \vdash t, s : A \), we say that \( t \) and \( s \) are context equivalent iff for every context \( C \) such that \( \vdash C[\vdash A] : \text{Str} \) we have that \( \llbracket C[t] \rrbracket(\epsilon) = \llbracket C[s] \rrbracket(\epsilon) \).
Contexts

- **Contexts:**

  \[
  C ::= t \mid [] \mid \lambda x.C \mid Ct \mid tC \mid 0(C) \mid 1(C) \mid \text{tail}(C) \\
  \mid \text{case}_A(C, t, t, t) \mid \text{case}_A(t, C, C, C) \mid \text{rec}_A(C, t, t, t).
  \]

- Notice that we are working with *linear* contexts!

- One can easily define a formal system for judgements in the form \( \Gamma \vdash C[\vdash A] \), with the obvious meaning.

**Definition (Context Equivalence)**

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

- **Trace**: a sequence of actions $l_1 \cdot l_2 \cdot \ldots \cdot l_n$ such that $l_i \in \{\text{pass}(v), \text{view}(m) \mid v \in V, m \in V^{\text{Str}}\}$.

- **Compatibility**: the trace $T$ is compatible $A$ iff

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

- **Example**: a term of type $A = a\text{Str} \rightarrow b\text{Str} \rightarrow \text{Str}$ requires a trace like $T = \text{pass}(m_1) \cdot \text{pass}(m_2) \cdot \text{view}(m)$.

**Definition**

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

*Context and trace equivalences coincide, i.e., for all \( t, s \):* 
\[
\quad t \equiv s \iff t \simeq^T s. 
\]

- 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

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{I} \Rightarrow^\varepsilon \mathcal{I} )</td>
<td>( \mathcal{I} \Rightarrow^S { (\lambda x.t_i)^{p_i} } )</td>
</tr>
<tr>
<td>( \mathcal{I} \Rightarrow^S { (t_i{v/x})^{p_i} } )</td>
<td>( \mathcal{I} \Rightarrow^S S \rightarrow S \Rightarrow U )</td>
</tr>
<tr>
<td>( \mathcal{I} \Rightarrow^S { (m_i)^{p_i} } )</td>
<td>( \mathcal{I} \rightarrow^S \text{view}(m) \sum_{m_i=m} p_i )</td>
</tr>
<tr>
<td>( \mathcal{I} \rightarrow^S \text{view}(m) \sum_{m_i=m} p_i )</td>
<td>( t \rightarrow { (t_i)^{p_i} } )</td>
</tr>
<tr>
<td>( \mathcal{I} + { (t)^p } \Rightarrow \mathcal{I} + { (t_i)^{p.p_i} } )</td>
<td></td>
</tr>
</tbody>
</table>

**Further Generalization**: same semantics, but on pairs in the form \((C, \mathcal{I})\).

**The Invariant**: Two distributions \( \mathcal{P}, \mathcal{Q} \) on pairs \((C, \mathcal{I})\) are related (written \( \mathcal{P} \triangledown \mathcal{Q} \)) iff

\[
\mathcal{P} = \sum_{i=1}^{n} p_i \cdot (C_i, \mathcal{I}_i); \quad \mathcal{Q} = \sum_{i=1}^{n} p_i \cdot (C_i, S_i);
\]

and \( \mathcal{I}_i \) and \( S_i \) 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:**
\[ \delta^C_A(t, s) = \sup_{\vdash C[\vdash A]:\text{Str}} |\llbracket C[t]\rrbracket(\epsilon) - \llbracket C[s]\rrbracket(\epsilon)|. \]

▶ 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 context equivalent.

▶ **Trace Metric:** \( \delta^T_A(t, s) = \sup_{\vdash T:A} |\text{Pr}(t, T) - \text{Pr}(s, T)|. \)

▶ The notion of a trace stays the same?

▶ *No!*

▶ The action \( \text{view}(\cdot) \) must be performed on finite sets of strings rather than on single strings.

▶ This is necessary to manage the case and 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:**
\[ \delta^C_A(t, s) = \sup_{C[t]:\text{Str}} \| [C[t]](\epsilon) - [C[s]](\epsilon) \| . \]

- This is a pseudometric:
\[
\begin{align*}
\delta^C(t, t) &= 0; \\
\delta^C(t, s) &= \delta^C(s, t); \\
\delta^C(t, r) &\leq \delta^C(t, s) + \delta^C(s, r). 
\end{align*}
\]

- \( \delta^C(t, s) = 0 \) iff \( t \) and \( s \) are context equivalent.

**Trace Metric:**
\[ \delta^T_A(t, s) = \sup_{T:A} |\text{Pr}(t, T) - \text{Pr}(s, T)| . \]

- The notion of a trace stays the same?

  - No!
  - The action \( \text{view}(\cdot) \) must be performed on finite sets of strings rather than on single strings.
  - This is necessary to manage the case and 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:**
  \[ \delta^C_{A}(t, s) = \sup_{T:C[\vdash A]:\text{Str}} |\llbracket C[t] \rrbracket(\epsilon) - \llbracket C[s] \rrbracket(\epsilon)|. \]
  - This is a pseudometric:
    \[
    \begin{align*}
    \delta^C(t, t) &= 0; \\
    \delta^C(t, s) &= \delta^C(s, t); \\
    \delta^C(t, r) &\leq \delta^C(t, s) + \delta^C(s, r).
    \end{align*}
    \]
    - \( \delta^C(t, s) = 0 \) iff \( t \) and \( s \) are context equivalent.

- **Trace Metric:**
  \[ \delta^T_{A}(t, s) = \sup_{T:A}|\Pr(t, T) - \Pr(s, T)|. \]
  - The notion of a trace stays *the same*?
    - No!
    - The action \( \text{view}(\cdot) \) must be performed on *finite sets of strings* rather than on single strings.
    - This is necessary to manage the case and 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:**
\[ \delta^C(t, s) = \sup_{C[\vdash A] : Str} |\llbracket C[t] \rrbracket(\epsilon) - \llbracket C[s] \rrbracket(\epsilon)|. \]
▶ This is a pseudometric:
\[
\begin{align*}
\delta^C(t, t) &= 0; \\
\delta^C(t, s) &= \delta^C(s, t); \\
\delta^C(t, r) &\leq \delta^C(t, s) + \delta^C(s, r).
\end{align*}
\]
▶ \(\delta^C(t, s) = 0\) iff \(t\) and \(s\) are context equivalent.

▶ **Trace Metric:**
\[ \delta^T_A(t, s) = \sup_{T:A} |Pr(t, T) - Pr(s, T)|. \]
▶ The notion of a trace stays the same?
▶ *No!*
▶ The action view(\(\cdot\)) must be performed on finite sets of strings rather than on single strings.
▶ This is necessary to manage the case and 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:**
  \[ \delta^C_A(t, s) = \sup_{C \vdash C[\vdash A] : \text{Str}} \| [C[t]](\epsilon) - [C[s]](\epsilon) \|. \]
  - This is a pseudometric:
    \[ \begin{align*}
    \delta^C(t, t) &= 0; \\
    \delta^C(t, s) &= \delta^C(s, t); \\
    \delta^C(t, r) &\leq \delta^C(t, s) + \delta^C(s, r).
    \end{align*} \]
    - \( \delta^C(t, s) = 0 \) iff \( t \) and \( s \) are context equivalent.

- **Trace Metric:**
  \[ \delta^T_A(t, s) = \sup_{T : A} |\Pr(t, T) - \Pr(s, T)|. \]
  - The notion of a trace stays *the same*?
    - *No!*
    - The action \( \text{view}(\cdot) \) must be performed on *finite sets of strings* rather than on single strings.
    - This is necessary to manage the *case* and *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 \( \{D_n\}_{n \in \mathbb{N}} \) and \( \{E_n\}_{n \in \mathbb{N}} \) (where both \( D_n \) and \( E_n \) are distributions on binary strings) are said to be \textit{computationally indistinguishable} iff for every PPT algorithm \( A \) the following quantity is a negligible function of \( n \in \mathbb{N} \):

\[
| \Pr_{x \leftarrow D_n} (A(x, 1^n) = \epsilon) - \Pr_{x \leftarrow E_n} (A(x, 1^n) = \epsilon) |.
\]

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

Definition

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

\[
|\Pr_{x \leftarrow D_n}(A(x, 1^n) = \epsilon) - \Pr_{x \leftarrow E_n}(A(x, 1^n) = \epsilon)|.
\]

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

- **Two Observations:**
  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 $\vdash t, s : \text{aStr} \rightarrow A$, we say that $t$ and $s$ are *parametrically context equivalent* iff for every context $C$ such that $\vdash C[\vdash A] : \text{Str}$ we have that
\[
\| [C[t^{1,n}]](\epsilon) - [C[s^{1,n}]](\epsilon) \| \text{ is negligible in } n.
\]
Parametric Context Equivalence (PCE)

Two Observations:
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 $\vdash t, s : \text{aStr} \rightarrow A$, we say that $t$ and $s$ are parametrically context equivalent iff for every context $C$ such that $\vdash C[\vdash A] : \text{Str}$ we have that $|\tilde{\mathbf{J}}_C[t^1_n](\epsilon) - \tilde{\mathbf{J}}_C[s^1_n](\epsilon)|$ is negligible in $n$. 
Theorem

Let \( t, s \) be two terms of type \( a\text{Str} \rightarrow \text{Str} \). Then \( t, s \) are parametric context equivalent iff the distribution ensembles \( \{[t^n]_1\}_{n \in \mathbb{N}} \) and \( \{[s^n]_1\}_{n \in \mathbb{N}} \) 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).
Theorem

Let \( t, s \) be two terms of type \( aStr \rightarrow Str \). Then \( t, s \) are parametric context equivalent iff the distribution ensembles \( \{ [t1^n] \}_{n \in \mathbb{N}} \) and \( \{ [s1^n] \}_{n \in \mathbb{N}} \) 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 $\text{view}(\cdot)$ takes in input closed RSLR terms $D: \text{aStr} \rightarrow \text{Str}$, that we call *distinguishers*.

**Definition**

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

**Theorem**

Parametric trace equivalence and parametric context equivalence coincide.
Parametric Trace Equivalence

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

**Definition**

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

**Theorem**

Parametric trace equivalence and parametric context equivalence coincide.
Parametric Trace Equivalence

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

**Definition**

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

**Theorem**

*Parametric trace equivalence and parametric context equivalence coincide.*
An Example

Let’s consider these two terms:

\[ t = \lambda n. (\lambda k. \lambda x. \lambda y. \text{ENC}(xn, k)) \text{GEN}(n); \]
\[ s = \lambda n. (\lambda k. \lambda x. \lambda y. \text{ENC}(yn, k)) \text{GEN}(n); \]
\[ \vdash t, s : \Box \text{Str} \rightarrow (\Box \text{Str} \rightarrow \text{Str}) \rightarrow (\Box \text{Str} \rightarrow \text{Str}) \rightarrow \text{Str}; \]

where \( \text{ENC}(x, k) = x \oplus G(k) \) 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 \( t \simeq s \).
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 Def. 3.8. Namely, we show that for any probabilistic polynomial-time adversary $\mathcal{A}$ there is a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\text{PrivK}_{\mathcal{A},\Pi}^{\text{c}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

(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 $D$:
1. Run $\mathcal{A}(\ell(n))$ 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 $\Pi' = (\text{Gen'}, \text{Enc'}, \text{Dec'})$ 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, $\text{Gen'}(\ell(n))$ outputs a uniform key $k$ of length $\ell(n)$, and the encryption of message $m \in \{0,1\}^{\ell(n)}$ using key $k \in \{0,1\}^{\ell(n)}$ is

$$\text{Enc'}(m, k) := m \oplus k.$$

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

$$\left| \frac{1}{2} - \Pr[\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1] \right| \leq \text{negl}(n),$$

which implies $\Pr[\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$.

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

$$\Pr[D(w) = 1] = \Pr[\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

(3.4)

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

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 $\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n)$. This is because when $\mathcal{A}$ is run as a subroutine by $D(w)$ in this case, $\mathcal{A}$ is given a ciphertext $c \leftarrow w \oplus m_{\mathcal{A}}$, 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))

$$\Pr_{\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1} = \frac{1}{2}.$$

(3.3)

2. If $w$ is instead generated by choosing uniform $k \in \{0,1\}^{\ell(n)}$ 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 $\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n)$. This is because, when run as a subroutine by $D$, it is given a ciphertext $c \leftarrow w \oplus m_k$ where $w = G(k)$ for a uniform $k \in \{0,1\}^{\ell(n)}$. Thus,

$$\Pr_{\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1} = \Pr[D(\ell(n)) = 1] = \frac{1}{2} + \text{negl}(n).$$

(3.5)

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

$$\left| \frac{1}{2} - \Pr[\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1] \right| \leq \text{negl}(n),$$

which implies $\Pr[\text{PrivK}_{\mathcal{A},\Pi'}^{\text{c}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n)$. 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
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 \( \text{PrivK}^{\text{eav}}_{A, \Pi}(n) \). 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 uniform. Since \( D \) outputs 1 exactly when \( \mathcal{A} \) succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))

\[
\Pr_{(m_0, m_1) \leftarrow \mathcal{R}(\ell(n))}[D(w) = 1] = \Pr[\text{PrivK}^{\text{eav}}_{\Pi}(n) = 1] = \frac{1}{2}
\]
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 $A$ there is a negligible function $\text{neg}(n)$ such that
\[
\Pr[\text{PrivK}_{A,\Pi}(n) \ni 1] \leq \frac{1}{2} + \text{neg}(n).
\] (3.2)

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

If $w$ is chosen uniformly from $\{0, 1\}^\ell(n)$, then the view of $A$ when run as a subroutine by $D$ is distributed identically to the view of $A$ in experiment $\text{PrivK}^{\text{eav}}_{A,\Pi}(n)$. This is because when $A$ is run as a subroutine by $D(w)$ in this case, $A$ is given a ciphertext $c = w \oplus m_b$ where $w \in \{0, 1\}^\ell(n)$ is uniform. Since $D$ outputs 1 exactly when $A$ succeeds in its eavesdropping experiment, we therefore have (cf. Equation (3.3))
\[
\Pr[\text{PrivK}_{A,\Pi}(n) \ni 1] = \Pr[\text{PrivK}^{\text{eav}}_{A,\Pi}(n) \ni 1] = \frac{1}{2}
\] (3.4)

1. Run $A(1^n)$ 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 $A$ and obtain output $b'$. Output 1 if $b' = b$, and output 0 otherwise.

$D$ clearly runs in polynomial time (assuming $A$ does).

Before analyzing the behavior of $D$, we define a modified encryption scheme $\tilde{\Pi} = (\text{Gen}, \tilde{\text{Enc}}, \tilde{\text{Dec}})$ 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, $\text{Gen}(1^n)$ outputs a uniform key $k$ of length $\ell(n)$, and the encryption of message $m \in 2^n$ using key $k \in \{0, 1\}^\ell(n)$

\[
\Pr[\text{PrivK}_{A,\Pi}(n) \ni 1] \leq \frac{1}{2} + \text{neg}(n),
\]

which implies $\Pr[\text{PrivK}_{A,\Pi}(n) \ni 1] \leq \frac{1}{2} + \text{neg}(n)$. Since $A$ 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
Suppose now that $t \not\equiv s$.

Then there exists a trace $T$ such that:
$$|\Pr(t, \text{pass}(1^n) \cdot T) - \Pr(s, \text{pass}(1^n) \cdot T)|$$
is not negligible, where $T = \text{pass}(v) \cdot \text{pass}(u) \cdot \text{view}(D)$.

If this is true we can easily construct a trace $S$ such that
$$|\Pr(G, \text{pass}(1^n) \cdot S) - \Pr(R, \text{pass}(1^n) \cdot S)|$$
is itself not negligible.

As a consequence, $G \not\equiv^T R$;
And this is a contradiction.

The trace $S$ only consists of a distinguisher $E$, which can be built out of $v$, $u$ and $D$:

$$E = \lambda x. \text{if } \text{rand} \text{ then } \left( D(x \oplus v(i \ell x)) \right) \text{ else } \left( D(x \oplus u(i \ell x)) \right)$$
An Example

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

\[
E = \lambda x. \text{if rand then } (D(x \oplus v(i \ell x))) \text{ else } (D(x \oplus u(i \ell x)))
\]
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?

\[ A \rightarrow B \]
\[ B \rightarrow C \]
\[ C \rightarrow A \]
\[ A \rightarrow C \]

\[ t_A : \text{Str} \otimes (\text{Str} \rightarrow \text{Str}) \]
\[ t_B : \text{Str} \rightarrow \text{Str} \]
\[ t_C : \text{Str} \rightarrow \text{Str} \otimes (\text{Str} \rightarrow \text{Str}) \]

\[ C[\cdot_A, \cdot_B, \cdot_C] \]
Why Pairs?

\[ A \rightarrow B \]
\[ B \rightarrow C \]
\[ C \rightarrow A \]
\[ A \rightarrow C \]

\[ t_A : \text{Str} \otimes (\text{Str} \rightarrow \circ \text{Str}) \]
\[ t_B : \text{Str} \rightarrow \circ \text{Str} \]
\[ t_C : \text{Str} \rightarrow \circ \text{Str} \otimes (\text{Str} \rightarrow \circ \text{Str}) \]

\[ C[\cdot_A, \cdot_B, \cdot_C] \]
Why Pairs?

\[\begin{align*}
A & \rightarrow B \\
B & \rightarrow C \\
C & \rightarrow A \\
A & \rightarrow C
\end{align*}\]

\[
t_A : Str \otimes (Str \rightarrow \circ Str)
\]

\[
t_B : Str \rightarrow \circ Str
\]

\[
t_C : Str \rightarrow \circ Str \otimes (Str \rightarrow \circ Str)
\]

\[
C[\cdot_A, \cdot_B, \cdot_C]
\]
Why Modalities?

Pseudorandom Functions:

$$\text{Str} \rightarrow \text{Str} \otimes !_p(\text{Str} \rightarrow \text{Str})$$

CPA-IND

$$\text{Str} \rightarrow \text{Str} \otimes !_p(\text{Str} \rightarrow \text{Str}) \otimes (\text{Str} \otimes \text{Str} \rightarrow \text{Str})$$
Why Modalities?

Pseudorandom Functions:

\[ \text{Str} \rightarrow \text{Str} \otimes !_p(\text{Str} \rightarrow \text{Str}) \]

CPA-IND

\[ \text{Str} \rightarrow \text{Str} \otimes !_p(\text{Str} \rightarrow \text{Str}) \otimes (\text{Str} \otimes \text{Str} \rightarrow \text{Str}) \]
Thank You!

Questions?