Probabilistic I/O Automata: A promising framework for the analysis of security protocols?

Based on a work by:

Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira and Roberto Segala

October 12, 2005

Motivation

Analyzing cryptographic protocols involve dealing with:

- computational issues (inherent to crypto definitions)
- concurrency issues (inherent to protocols)

Two approaches have been proposed:

1. coming from the crypto community
2. coming from the security community

Motivation

1. Crypto approach
    - fine grained, based on (I)TM
    - Protocols involve computational issues ⇒ TM ☺
    - Protocols involve concurrency issues ⇒ ITM ☹
        - all concurrency aspects discussed "informally"
        - ITMs only provide a low level of abstraction, never used in reality
        - tapes probably not the most natural communication channels: connect tapes? compose ITMs? ...
        - "Sketch" proofs, error-prone [S02, HMS03, ...]

Motivation

2. Dolev-Yao approach
    - completely formal description
    - allows reasoning about much larger systems
    - systematic, often automated reasoning
    - strong assumptions about cryptography
        - too strong?
        - at least, not directly comparable

Motivation

First solution for these crypto-assumptions [AR00, CH04, . . .]:

- Prove that $\exists$ D-Y proof $\Rightarrow$ $\exists$ crypto proof (if we have good crypto primitives)
- Still need a way to formally formulate crypto proofs

Motivation

We propose a framework allowing to:

- express computational *and* concurrency aspects of crypto proofs
- prove systematically the security of cryptographic protocols
  - automatic proof checking?
- reason at several level of abstraction (TM $\longrightarrow$ DY-style)

Related Works

- Common motivations with [S04, H05, B05, ...], but:
  - They decompose (and automate) proofs as sequences of (computational) games
  - They do not consider protocols as realizing an ideal functionality
- most similar [PW01, LMMS98], but:
  - different ways to handle non-determinism
  - motivations are different

Why PIOAs?

Introduced by Lynch, Segala and Vaandrager [SL95, LSV03]

- Classical framework in the concurrency community
- Checking indistinguishability of systems is a classical issue
  - Proved through inductive simulation techniques
  - $\Rightarrow$ Positive arguments
- Composition of PIOAs is natural and well-known
- PIOAs allow to express protocols rigorously at *multiple* levels of abstraction
- *Probabilistic* I/O automata allow to describe random choices, . . .

Challenges

1. Need to find a way to resolve the non-determinism
2. Need to model resource-bounded computations
3. Need to model computational hardness assumptions
4. Need new notions of implementation
   ($\approx$ indistinguishability):
   - for identical distributions
   - for computationally indistinguishable distributions

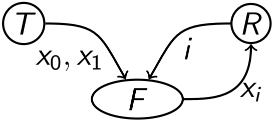
In this Talk...

- We extend the PIOA framework in order to be able to:
  - describe cryptographic protocol executions
  - describe computationally bounded PIOAs
  - prove (computational) indistinguishability of PIOAs
- We exemplify our approach by analyzing an OT protocol,
  - proof in the style of Canetti's UC framework
  - static, semi-honest adversary for now
- We will use this protocol as a running example

Our Example

Two-party Oblivious Transfer:

1. Transmitter has two messages x_0 and x_1
2. Receiver wants to read the i-th of them
3. Transmitter learns nothing
4. Receiver learns nothing but x_i



Our Example

Two-party Oblivious Transfer [GMW87]

- T has input bits x_0, x_1 – R has input bit i
- Passive, static, semi-honest adversary

$$
\begin{array}{cl}
T & \xrightarrow{\qquad f \qquad} \\
\Downarrow & \\
\bullet & \xleftarrow{\qquad z_0, z_1 \qquad} \\
\Downarrow & \\
\bullet & \xrightarrow[x_1 \oplus B(f^{-1}(z_1))]{x_0 \oplus B(f^{-1}(z_0))}
\end{array}
\qquad\qquad
\begin{array}{rc}
\xrightarrow{\qquad f \qquad} & R \\
 & \Downarrow \\
\xleftarrow{f^{1-i}(y_0),\, f^{i}(y_1)} & \bullet \\
 & \Downarrow \\
\xrightarrow{\qquad b_0, b_1 \qquad} & \bullet
\end{array}
$$

- f is a random trapdoor permutation
- y_0 and y_1 are random elements of the domain of f
- B is a hard-core predicate for f
- R outputs $x_i = b_i \oplus B(y_i)$ – T outputs nothing

Motivation for OT

1. Complete primitive [GMW87]
2. Two flavors of secrecy
   - x_{1-i} computationally hidden to R
   - i perfectly hidden to T

$$
\begin{tikzcd}
T \arrow[r, "f"] \arrow[d, Rightarrow] & {} \\
\bullet \arrow[d, Rightarrow] & {} \arrow[l, "{z_0, z_1}"'] \\
\bullet \arrow[r, "x_0 \oplus B(f^{-1}(z_0))", "x_1 \oplus B(f^{-1}(z_1))"'] & {}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{} \arrow[r, "f"] & R \arrow[d, Rightarrow] \\
{} & \bullet \arrow[l, "{f^{1-i}(y_0), f^{i}(y_1)}"'] \arrow[d, Rightarrow] \\
{} \arrow[r, "{b_0, b_1}"] & \bullet
\end{tikzcd}
$$

Our Goal

We want to prove that:

- For every adversary A corrupting $C \subseteq \{T, R\}$,
  obtaining I/O of parties in C and
  seeing the protocol execution by the honest parties
- There is a simulator S having access to the same I/O as A
  able to simulate a protocol exection as convincing as the previous one

$\Rightarrow$ we are sure that the protocol does not disclose anything not disclosed by the specification

In this Talk...

- We extend the PIOA framework in order to be able to:
  - describe cryptographic protocol executions
  - describe computationally bounded PIOAs
  - prove (computational) indistinguishability of PIOAs
- We exemplify our approach by analyzing an OT protocol,
  - proof in the style of Canetti's UC framework
  - static, semi-honest adversary for now
- We will use this protocol as a running example

What are PIOAs?

Probabilistic I/O Automata are described through:

- state (and a start state)
- actions, partitioned into:
  - input actions
  - output actions
  - internal (hidden) actions
- transition function:
  $(\mathit{state} \times \mathit{action}) \longrightarrow$ distribution on *states*

Example: Transmitter's role

Input actions:

$in(x)_{Trans}$, $x \in (\{0,1\} \to \{0,1\})$
$receive(2,z)_{Trans}$, $z \in (\{0,1\} \to D)$

Output actions:

$send(1,f)_{Trans}$, $f \in Tdp$
$send(3,b)_{Trans}$, $b \in (\{0,1\} \to \{0,1\})$

Internal actions:

$choose-tdppval_{Trans}$
$fix-bval_{Trans}$

State:

$inval \in (\{0,1\} \to \{0,1\}) \cup \{\perp\}$, initially $\perp$
$tdpp \in Tdpp \cup \{\perp\}$, initially $\perp$
$zval \in (\{0,1\} \to D) \cup \{\perp\}$, initially $\perp$
$bval \in (\{0,1\} \to \{0,1\}) \cup \{\perp\}$, initially $\perp$

Example: Transmitter's role

Transitions:

$\mathbf{in(x)_{Trans}}$
Effect:
if $inval = \perp$ then
$inval := x$

$\mathbf{choose - tdppval_{Trans}}$
Effect:
if $tdpp = \perp$ then
$tdpp :=$ random tdpp

$\mathbf{send(1, f)_{Trans}}$
Precondition:
$tdpp \neq \perp$,
$f = tdpp.funct$
Effect:
none

$\mathbf{receive(2, z)_{Trans}}$
Effect:
if $zval = \perp$ then $zval := z$

$\mathbf{fix - bval_{Trans}}$
Precondition:
$tdpp, zval, inval \neq \perp$
$bval = \perp$
Effect:
$bval = B(tdpp.inv(zval)) \oplus inval$

$\mathbf{send(3, b)_{Trans}}$
Precondition:
$b = bval \neq \perp$
Effect:
none

What can we do with PIOAs?

We can:

- compose PIOAs
  - compatibility conditions on the action's names
  - input actions which are output actions of another PIOA are not available anymore
  - output actions remain available
- hide output actions
  - output actions become internal actions

Resolving Nondeterministic Choices

Problem: A lot of actions are enabled at the same time (inside a protocol party, between protocol parties)

Solution: Use *task-schedulers*!

- A task is an equivalence class on actions
- Tasks abstract from state variables
- At most one action is enabled in a specific task
- A task-scheduler is a (maybe infinite) sequence of tasks

Example: Tasks for the transmitter:
$\{in(*)_{Trans}\}$, $\{choose - tdppval_{Trans}\}$, $\{send(1,*)_{Trans}\}$, $\{receive(2,*)_{Trans}\}$, $\{fix - bval_{Trans}\}$, $\{send(3,*)_{Trans}\}$.

When a task-scheduler is defined, we have pure probabilistic executions!

In this Talk...

- We extend the PIOA framework in order to be able to:
  - describe cryptographic protocol executions
  - describe computationally bounded PIOAs
  - prove (computational) indistinguishability of PIOAs
- We exemplify our approach by analyzing an OT protocol,
  - proof in the style of Canetti's UC framework
  - static, semi-honest adversary for now
- We will use this protocol as a running example

Proving Security of Protocols

We want to prove that a protocol P *realizes* a functionality F, which means:

- For every *efficient adversary* A for P,
- there is an efficient adversary S for F such that:
- no *environment* can *efficiently distinguish* $P||A$ from $F||S$.

What do we mean by:

- an *efficient adversary*?
- an *environment*?
- *efficiently distinguish* task-PIOAs?

Efficient Adversary

We introduce *time-bounded* task-PIOAs.

Suppose we represent all parts of the task-PIOA T as bit strings.
T is b-time-bounded iff

1. all parts of the task-PIOA can be decoded by a TM in time $\leq b$
2. $\exists$ a *TM* running in time $\leq b$ that, given a task and a state, computes the unique enabled action
3. $\exists$ a *TM* running in time $\leq b$ that, given an action and a state, computes the next state
4. all these TM have description $\leq b$ (in some standard encoding)

Efficient Adversary

We introduce *polynomial-time* task-PIOA families.

$\overline{T} = \{T_k\}_{k \in \mathbb{N}}$ is a polynomial-time task-PIOA family iff $\exists$ a polynomial p such that T_k is a $p(k)$-time-bounded task-PIOA.

- an efficient adversary is a polynomial-time task-PIOA family
- transmitters and receivers are polynomial-time task-PIOA families

Proving Security of Protocols

We want to prove that a protocol P *realizes* a functionality F, which means:

- For every *efficient adversary* A for P,
- there is an efficient adversary S for F such that:
- no *environment* can *efficiently distinguish* $P||A$ from $F||S$.

What do we mean by:

- an *efficient adversary*?
- an *environment*?
- *efficiently distinguish* task-PIOAs?

Environment

A task-PIOA E is an environment for T iff

1. it closes T ($E||T$ has no input actions)
2. E has a special output *accept*, which we use to measure ability of distinguishing

Proving Security of Protocols

We want to prove that a protocol P *realizes* a functionality F, which means:

- For every *efficient adversary* A for P,
- there is an efficient adversary S for F such that:
- no *environment* can *efficiently distinguish* $P||A$ from $F||S$.

What do we mean by:

- an *efficient adversary*?
- an *environment*?
- *efficiently distinguish* task-PIOAs?

Perfect Implementation

A first implementation relation:

$T_1 \leq_0 T_2$ means

- for every environment E for T_1 and T_2,
- for every scheduler ρ_1 for $E||T_1$
- there is a scheduler ρ_2 for $E||T_2$ and
- $Pr[E||T_1$ scheduled by ρ_1 outputs *accept*$] =$
  $Pr[E||T_2$ scheduled by ρ_2 outputs *accept*$]$

$T_1 \leq_0 T_2$ iff any trace distribution of $E||T_1$ is also a trace distribution of $E||T_2$

Efficient Distinguisher

Our first implementation relation is too restrictive:

1. environments can distinguish computationally indistinguishable trace distributions
2. environments can receive unbounded computational help from a PT adversary (there is no bound on the length of the schedulers)

Efficient Distinguisher

Approximate implementation relation: $T_1 \leq_{b,b_1,b_2,\epsilon} T_2$ means:

- for every b-bounded environment E for T_1 and T_2,
- for every b_1-bounded scheduler ρ_1 for $E||T_1$,
- there is a b_2-bounded scheduler ρ_2 for $E||T_2$ such that:
- $|Pr[E||T_1$ scheduled by ρ_1 outputs *accept*$]-$
  $Pr[E||T_2$ scheduled by ρ_2 outputs *accept*$]| \leq \epsilon$

Efficient Distinguisher

Natural extension to families:
Suppose b, b_0, b_1, ϵ are functions $\mathrm{N} \longrightarrow \mathrm{R}^+$, then:
$\overline{T_1} \leq_{b,b_1,b_2,\epsilon} \overline{T_2}$ means:

- for every $b(k)$-bounded environment E_k for $(T_1)_k$ and $(T_2)_k$,
- for every $b_1(k)$-bounded scheduler $(\rho_1)_k$ for $E_k||(T_1)_k$,
- there is a $b_2(k)$-bounded scheduler $(\rho_2)_k$ for $E_k||(T_2)_k$ such that:

$$|Pr[E_k||(T_1)_k \text{ scheduled by } (\rho_1)_k \text{ outputs } \mathit{accept}] - Pr[E_k||(T_2)_k \text{ scheduled by } (\rho_2)_k \text{ outputs } \mathit{accept}]| \leq \epsilon(k)$$

Efficient Distinguisher

Specializing this to polynomials:

$\overline{T_1} \leq_{neg,pt} \overline{T_2}$ means:

- for every polynomial p,
- for every polynomial p_1,
- there is a polynomial p_2 and
- a negligible function ϵ such that: $\overline{T_1} \leq_{p,p_1,p_2,\epsilon} \overline{T_2}$

Proving Security of Protocols...

P realizes F means:

- For every polynomial-time bounded A for P,
- there is a polynomial-time bounded S for F such that $P||A \leq_{neg,pt} F||S$.

Proving Security of Protocols...

P realizes F means:

- For every polynomial-time bounded A for P,
- there is a polynomial-time bounded S for F such that $P||A \leq_{neg,pt} F||S$.

How do we prove this?

The $\leq_{neg,pt}$-relation

The $\leq_{neg,pt}$ enjoys a lot of convenient properties:

- Transitivity:
  if $T_1 \leq_{neg,pt} T_2$ and $T_2 \leq_{neg,pt} T_3$ then $T_1 \leq_{neg,pt} T_3$
- Composition:
  if $T_1 \leq_{neg,pt} T_2$ and T_3 is PT-bounded then
  $T_1 || T_3 \leq_{neg,pt} T_2 || T_3$
- Hiding:
  if $T_1 \leq_{neg,pt} T_2$ and U is an output task for T_1 and T_2
  then $hide_U(T_1) \leq_{neg,pt} hide_U(T_2)$
- Relation with $\leq_0$:
  if $T_1 \leq_0 T_2$ and the required task schedulers only increase by a polynomial factor then $T_1 \leq_{neg,pt} T_2$

Proof of the OT Protocol

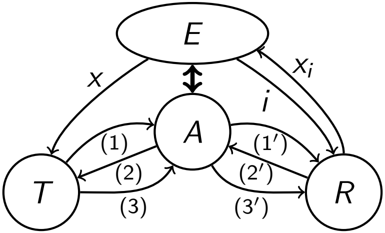
Outline:

- We want to prove that:
  - $T||R$ realizes F
  - $\forall$ PT $A, \exists$ PT S : $T||R||A \leq_{neg,pt} F||S$
- Actually, we prove that:
  - $\forall$ PT A,
    $T||R||A \leq_0 F||TR_1||A \leq_{neg,pt} F||TR_2||A \leq_0 F||TR||A$
    and we have adequate bounds on the schedulers for the $\leq_0$ relations

Proof of the OT Protocol

Transitivity of $\leq_{neg,pt}$ allows to split proofs in different parts!

Real system (RS):

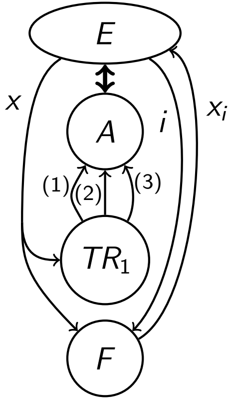


$$
\begin{aligned}
(1) &= f \\
(2) &= z_0, z_1 \\
(3) &= x \oplus B(f^{-1}(z))
\end{aligned}
\qquad
\begin{aligned}
(1') &= f \\
(2') &= f^{1-i}(y_0), f^{i}(y_1) \\
(3') &= b_0, b_1
\end{aligned}
$$

Int_1

First intermediate system (Int_1):

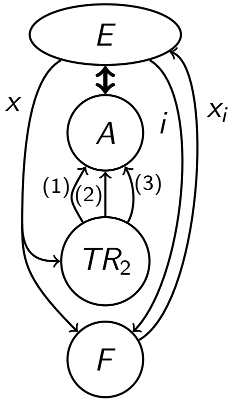
$$(1) = f$$
$$(2) = z_0, z_1$$
$$(3) = x \oplus B(f^{-1}(z))$$



- We prove: $\forall A,\ T||R||A \leq_0 F||TR_1||A$
- Note that we really use the asymmetry of $\leq$!!!

Int_2

Second intermediate system (Int_2):



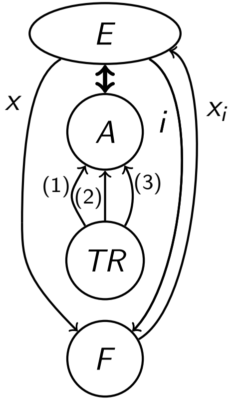
$$(1) = f$$
$$(2) = z_0, z_1$$
$$(3) = x_0 \oplus c_0, x_1 \oplus c_1$$
$$(c \text{ random})$$

- We prove: $F||TR_1||A \leq_{neg,pt} F||TR_2||A$
- This is an approximate implementation!

SIS

Ideal system (SIS):

$(1) = f$
$(2) = z_0, z_1$
$(3) = c_0, c_1$



- We prove: $F||TR_2||A \leq_0 F||TR||A$

Proving $T_1 \leq_0 T_2$

How do we prove that $T_1 \leq_0 T_2$?

or How do we prove that, for every environment E for T_1 and T_2, every trace distribution of $T_1||E$ is also a trace distribution of $T_2||E$?

$\Rightarrow$ We use a simulation relation!

- Standard tool in the concurrency community... extended to our framework!

Simulation Relation

What is a simulation relation R?

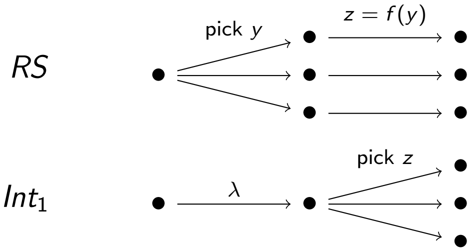
- Suppose E is fixed. R relates:
  - distributions on states of $T_1||E$ to
  - distributions on states of $T_2||E$
- R is a simulation relation iff
  - start state of $T_1||E$ related to start state of $T_2||E$
  - for every task of $T_1||E$, there is a sequence of tasks for $T_2||E$ such that:
    - executing these tasks on both systems preserves traces
    - the resulting distributions on states are also related

Simulation Relation

What is a simulation relation R?

- Suppose E is fixed. R relates:
  - distributions on states of $T_1||E$ to
  - distributions on states of $T_2||E$
- R is a simulation relation iff
  - start state of $T_1||E$ related to start state of $T_2||E$
  - for every task of $T_1||E$, there is a sequence of tasks for $T_2||E$ such that:
    - executing these tasks on both systems preserves traces
    - the resulting distributions on states are also related
- *Theorem:* If, $\forall E$, such an R exists, then $T_1 \leq_0 T_2$

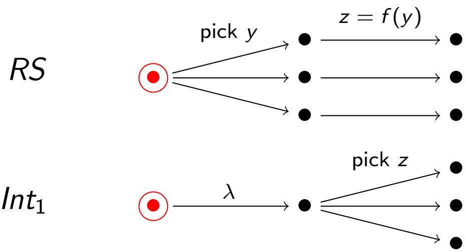
Simulation Relation



R usually contains requirements like:

- if $Int_1.zval = \perp$ then (1) or (2) hold:
  (1) $RS.yval = \perp$
  (2) $RS.yval$ is the uniform distribution on $Dom(f)$
- $Int_1.zval = RS.zval$

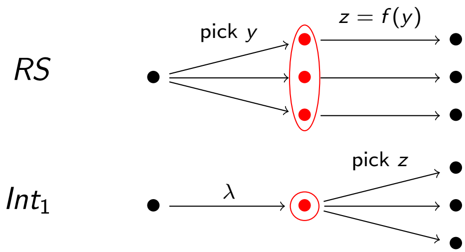
Simulation Relation



R usually contains requirements like:

- if $Int_1.zval = \perp$ then (1) or (2) hold:
  (1) $RS.yval = \perp$
  (2) $RS.yval$ is the uniform distribution on $Dom(f)$
- $Int_1.zval = RS.zval$

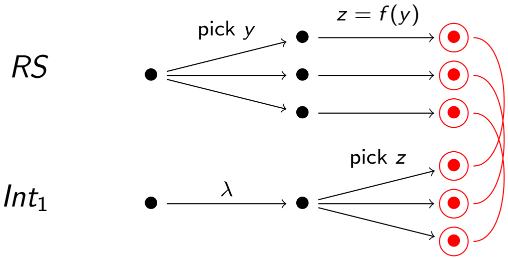
Simulation Relation



R usually contains requirements like:

- if $Int_1.zval = \perp$ then (1) or (2) hold:
  (1) $RS.yval = \perp$
  (2) $RS.yval$ is the uniform distribution on $Dom(f)$
- $Int_1.zval = RS.zval$

Simulation Relation



R usually contains requirements like:

- if $Int_1.zval = \perp$ then (1) or (2) hold:
  (1) $RS.yval = \perp$
  (2) $RS.yval$ is the uniform distribution on $Dom(f)$
- $Int_1.zval = RS.zval$

Proving $T_1 \leq_{neg,pt} T_2$

This is where we need computational hardness assumptions.

For our OT protocol, we transpose the classical crypto assumption for hard-core predicates to our framework.

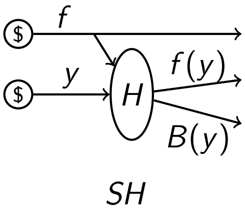
Crypto: for every PPT G, there is a negligible ϵ:

$$\left| \begin{array}{l} \Pr[f \leftarrow Tdp; \\ \quad z \leftarrow D; \\ \quad b \leftarrow B(f^{-1}(z)) : \\ \quad G(f,z,b) = 1 \quad] \end{array} - \begin{array}{l} \Pr[\quad f \leftarrow Tdp; \\ \qquad z \leftarrow D; \\ \qquad b \leftarrow \{0,1\} : \\ \qquad G(f,z,b) = 1 \quad] \end{array} \right| \leq \epsilon$$

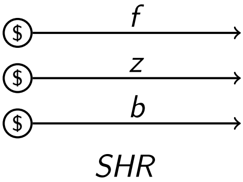
Defining H-C Predicates in terms of PIOAs

We transpose the classical crypto assumption to task-PIOAs.

$SH \leq_{neg,pt} SHR$:



$\leq_{neg,pt}$



Theorem: Both formulations are equivalent!

Proving $T_1 \leq_{neg,pt} T_2$

We need to prove $Int_1 \leq_{neg,pt} Int_2$.

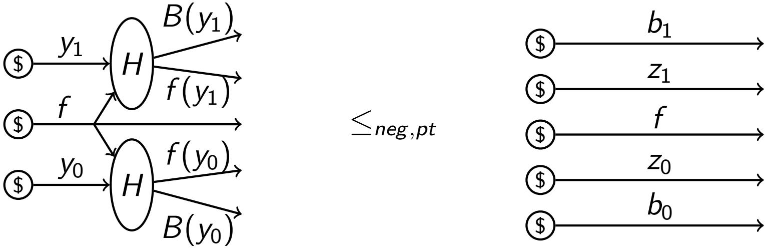
The only difference between the two systems is that

- in Int_1, the third message is $x_0 \oplus B(f^{-1}(z_0)), x_1 \oplus B(f^{-1}(z_1))$
- in Int_2, the third message is $x_0 \oplus c_0, x_1 \oplus x_1$

We need to replace two hard-core bits with random bits!

Using our PIOAs Hardness Assumption

Our composition and transitivity properties allow proving $SH2 \leq_{neg,pt} SHR2$:



$SH2$ $SHR2$

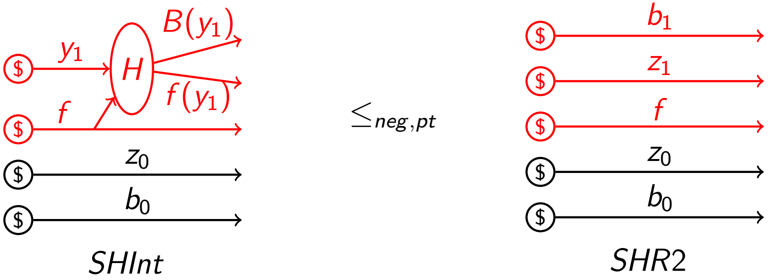
Using our PIOAs Hardness Assumption

Consider the *SHInt* intermediate system. We have:

*SH*2 and *SHInt* are just *SH* and *SHR* composed with the same systems!

Using our PIOAs Hardness Assumption

We also have:



$SH2 \leq_{neg,pt} SHR2$ follows from our transitivity result!
Further compositions allow proving $Int_1 \leq_{neg,pt} Int_2 \ldots$

Summary

We propose a new framework for the analysis of cryptographic protocols:

- We extended the PIOA theory with tasks to manage non-determinism in a cryptographic context
- We extended the PIOA theory to manage computational assumptions
- We can express classical hardness assumptions in terms of PIOAs
- Our task-PIOA formalism allow to describe and analyze protocols

Summary

We proved the security of the [GMW87] OT Protocol in the presence of a semi-honest, static adversary:

- Imagination still needed for building the right simulator, but
- Systematic techniques used to prove its correctness:
    - Decompose the proof into different steps
    - Perfectly indistinguishable steps are proved through our simulation relation
    - Computationally indistinguishable steps are proved by composing PIOAs on top of those expressing classical crypto assumptions

Further works

- Composable security
  - Composition is a natural operation for PIOAs

  $\Rightarrow$ Composition theorems much easier than those based on ITMs!
- New cryptographic assumptions
  - Pseudo-random functions, ...

  $\Rightarrow$ Crypto assumptions involve adaptative behaviors!
- Active adversaries
  - Key exchange protocols?
- Mechanization, automation of the proof process?

Thank you!