

Using Task-Structured PIOAs to Analyze Cryptographic Protocols

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

University of Tokyo – July 2006



Motivation

Make:

- ▶ systematic proofs
- ▶ in a composable security setting
- ▶ considering probabilistic and nondeterministic behaviors
- ▶ including nondeterministic protocol specification



Nondeterministic behavior

Why? Experience from concurrency theory says:

- ▶ just specify what is needed for the protocol to work
- ▶ simplicity: avoids “clutter” in the specification
- ▶ generality: keeps freedom for the implementer



Example: Oblivious Transfer

OT functionality without internal nondeterminism:

Version 1:

- ▶ on input (x_0, x_1) from T , store (x_0, x_1)
- ▶ on input i from R : if input (x_0, x_1) was received, send x_i to R , else do nothing



Example: Oblivious Transfer

OT functionality without internal nondeterminism:

Version 2:

- ▶ on input (x_0, x_1) from T : if input i was received, send x_i to R , else store (x_0, x_1)
- ▶ on input i from R : if input (x_0, x_1) was received, send x_i to R , else store i

OT functionality with internal nondeterminism:

- ▶ on input (x_0, x_1) from T , store (x_0, x_1)
- ▶ on input i from R , store i
- ▶ if (x_0, x_1) and i have been received, send x_i to R



Example: Needham-Schroeder-Lowe

Receiver role:

Version 1:

1. Receive $\{N_a, A\}_{K_B}$
2. Select N_b
3. Send $\{N_a, N_b, B\}_{K_A}$

Version 2:

1. Select N_b
2. Receive $\{N_a, A\}_{K_B}$
3. Send $\{N_a, N_b, B\}_{K_A}$

- ▶ from a security point of view: who cares?
- ▶ according to the hardware, one solution might be better than the other



Motivation

Make:

- ▶ systematic proofs
- ▶ in a composable setting
- ▶ exhibiting probabilistic and nondeterministic behaviors
- ▶ including in protocol specification

We want to prove security *for every way to resolve the nondeterminism*



This work...

In this work, we propose:

- ▶ a new model for the analysis of crypto protocols
 - ▶ protocols can have internal nondeterminism
 - ▶ enables simulation based security for nondeterministic systems
- ▶ an analysis of an Oblivious Transfer protocol [EGL85,GMW87] in our model



Starting Point

PIOAs [Seg95, LSV03] are kinds of interacting, abstract, automata:

- ▶ state variables
- ▶ actions (input, output, internal)
- ▶ transitions: $(state \times action) \rightarrow \text{Disc}(states) \cup \perp$

Internal nondeterminism for output and internal actions

- ▶ not algorithmically resolved
- ▶ kept unresolved in the analyzed systems



Resolving nondeterminism

- ▶ PIOAs use schedulers with full knowledge of current state — way too powerful!
- ▶ We introduce *tasks*, i.e.,
 - ▶ equivalence classes on actions, abstracting from state variables (ex: send message 1, select key, ...)
 - ▶ given a task, at most one possible (probabilistic) action
- ▶ We introduce *task schedulers*: just sequences of tasks
- ▶ Execution: read first task, find and execute the enabled action (if there is one), go to next task, ...



Task-PIOAs

Task-PIOAs provide:

- ▶ realistic model of network behavior:
 - ▶ adversary (just one task-PIOA) still controls the network
 - ▶ adversary cannot decide when parties send messages
- ▶ freedom for implementers:
 - ▶ implementers can resolve internal nondeterminism as they want
 - ▶ not a concern in model or security proofs



Example: OT Functionality

Input actions:

$in(x)_{Trans}, x \in (\{0, 1\} \rightarrow \{0, 1\})$
 $in(i)_{Rec}, i \in \{0, 1\}$

Output actions:

$out(x)_{Rec}, x \in \{0, 1\}$

State:

$inval(Trans) \in (\{0, 1\} \rightarrow \{0, 1\}) \cup \{\perp\}$, initially \perp
 $inval(Rec) \in \{0, 1, \perp\}$, initially \perp

Transitions:

$in(x)_{Trans}$: if $inval(Trans) = \perp$ then $inval(Trans) := x$
 $in(i)_{Rec}$: if $inval(Rec) = \perp$ then $inval(Rec) := i$
 $out(x)_{Rec}$: Preconditions: $inval(Trans), inval(Rec) \neq \perp$
 $x = inval(Trans)(inval(Rec))$

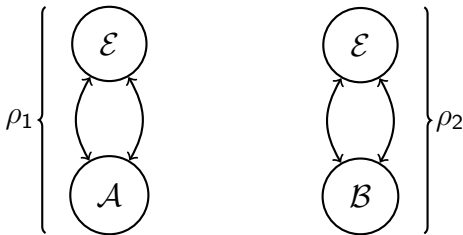
Tasks: $\{out(*)_{Rec}\}$



Indistinguishability

Implementation relation for task-PIOAs:

- ▶ $\mathcal{A} \leq \mathcal{B}$ means:
 - \forall environment \mathcal{E} for \mathcal{A} and \mathcal{B} ,
 - and \forall task scheduler ρ_1 for $\mathcal{A}||\mathcal{E}$,
 - \exists task scheduler ρ_2 for $\mathcal{B}||\mathcal{E}$
 - s.t. \mathcal{E} cannot distinguish \mathcal{A} from \mathcal{B}



Indistinguishability

Implementation relation for task-PIOAs:

- ▶ $\mathcal{A} \leq \mathcal{B}$ means:
 - \forall environment \mathcal{E} for \mathcal{A} and \mathcal{B} ,
 - and \forall task scheduler ρ_1 for $\mathcal{A}||\mathcal{E}$,
 - \exists task scheduler ρ_2 for $\mathcal{B}||\mathcal{E}$
 - s.t. \mathcal{E} cannot distinguish \mathcal{A} from \mathcal{B}
- ▶ Indistinguishability for nondeterministic systems
- ▶ \leq transitive: $A \leq B, B \leq C \Rightarrow A \leq C$
- ▶ \leq composable: $A \leq B \Rightarrow A||C \leq B||C$



Proving Security

Two variants of \leq :

- ▶ \leq_0 , for perfect indistinguishability
- ▶ $\leq_{neg,pt}$ for computational indistinguishability

\leq_0 proved using a sound simulation relation

- ▶ \approx matching (distributions on) states
- ▶ very systematic

$\leq_{neg,pt}$ proved using computational assumptions

- ▶ Express computational assumptions as $C_1 \leq_{neg,pt} C_2$
- ▶ Composition: $C_1 \leq_{neg,pt} C_2 \Rightarrow C_1 || Ifc \leq_{neg,pt} C_2 || Ifc$



Proving Security

Proofs are modular:

- ▶ $A \leq B$ proved as $A \leq A_1 \leq \dots \leq A_n \leq B$
 - ▶ \approx sequences of games, but for automata
- ▶ Composition properties allow reusing proofs for small systems in bigger ones

Reasoning at multiple abstraction levels:

- ▶ systematic proof techniques in terms of task-PIOAs
- ▶ can be used at computational level *and* at D-Y level



Establishing $C_1 \leq_{neg,pt} C_2$

Example: Hard-core predicates for trapdoor permutations

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

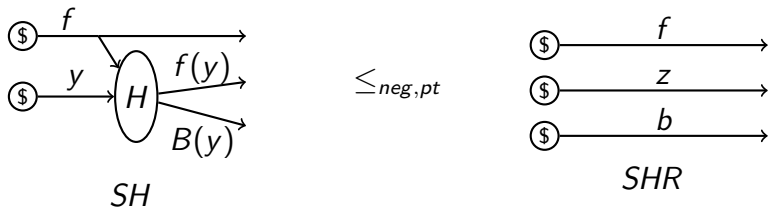
$$\left| \begin{array}{l} \Pr[f \leftarrow Tdp; \\ y \leftarrow D; \\ b \leftarrow B(y) : \\ G(f, f(y), b) = 1] \end{array} \right. - \left. \begin{array}{l} \Pr[f \leftarrow Tdp; \\ z \leftarrow D; \\ b \leftarrow \{0, 1\} : \\ G(f, z, b) = 1] \end{array} \right| \leq \epsilon$$



Defining H-C Predicates in terms of PIOAs

We transpose this classical crypto assumption to task-PIOAs.

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



Theorem: Both formulations are equivalent!



Using Computational Assumptions

What's happening if we use 2 hard-core bits?

In some protocol, we:

- ▶ select one trapdoor permutation f
- ▶ select two elements of the domain of f , say, (y_0, y_1)
- ▶ transmit $f(y_0), f(y_1), B(y_0), B(y_1)$

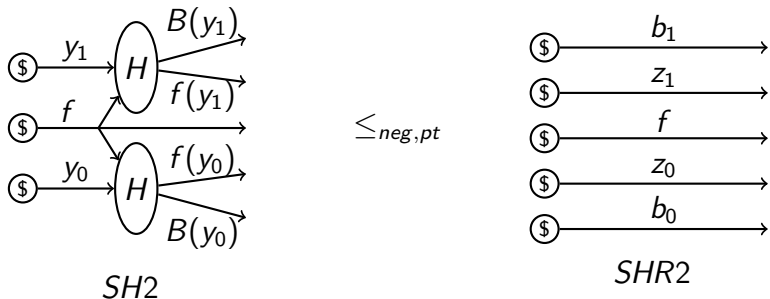
Do we keep the same indistinguishability guarantee?

- ▶ That is, $B(y_0)$ and $B(y_1)$ cannot be distinguished from random bits



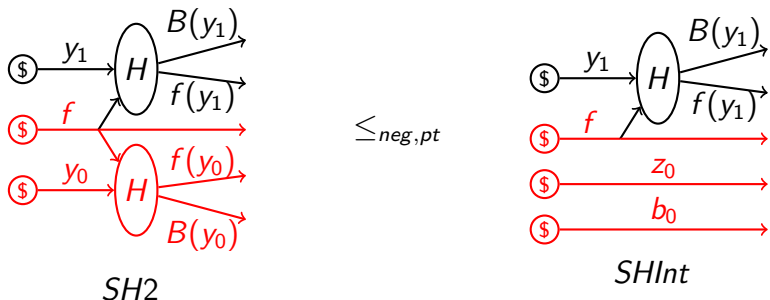
Using our PIOAs Hardness Assumption

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



Using our PIOAs Hardness Assumption

Consider the *SHInt* intermediate system. We have:

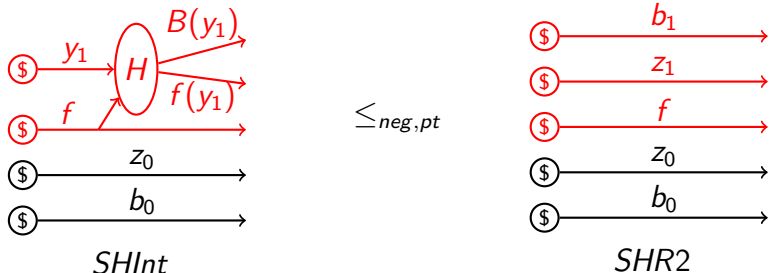


SH2 and *SHInt* are just *SH* and *SHR* composed with the same systems!



Using our PIOAs Hardness Assumption

We also have:



$SH2 \leq_{neg,pt} SHR2$ follows from our transitivity result!



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 $T_2 || E$?

⇒ We use a simulation relation!

- ▶ Standard technique in the concurrency community, extended to our framework



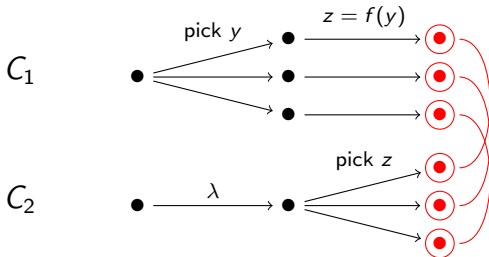
Simulation Relation

How does our simulation relation R look?

- ▶ Suppose E is fixed. R relates (\approx):
 - ▶ 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 $C_2.zval = \perp$ then (1) or (2) holds:
 - (1) $C_1.yval = \perp$
 - (2) $C_1.yval$ is the uniform distribution on $Dom(f)$
- ▶ $C_2.zval = C_1.zval$



Conclusion

Case-study on a simple OT protocol [GMW87] available:
MIT-CSAIL-TR-2006-019, Mar. 2006.

We hope task-PIOAs provide a framework for:

- ▶ More general, expressive, specifications
- ▶ More general, systematic, security proofs



Conclusion

Further works:

- ▶ General theorem for secure protocol composition
- ▶ Deal with other computational assumptions
- ▶ New case studies (key exchange, ...)
- ▶ Mechanization
- ▶ New simulation relations
- ▶ ...

