Time-Bounded Task-PIOAs: A Framework for Analyzing Security Protocols[⋆]

Ran Canetti[1,3], Ling Cheung[2,3], Dilsun Kaynar[3], Moses Liskov[4], Nancy Lynch[3], Olivier Pereira[5], and Roberto Segala[6]

[1] IBM T.J. Watson Center and Massachusetts Institute of Technology
[2] Radboud University of Nijmegen,
[3] Massachusetts Institute of Technology, lynch@csail.mit.edu
[4] The College of William and Mary
[5] Université Catholique de Louvain
[6] Università di Verona

Abstract. We present the *Time-Bounded Task-PIOA* modeling framework, an extension of the Probabilistic I/O Automata (PIOA) framework that is intended to support modeling and verification of security protocols. Time-Bounded Task-PIOAs directly model probabilistic and nondeterministic behavior, partial-information adversarial scheduling, and time-bounded computation. Together, these features are adequate to support modeling of key aspects of security protocols, including secrecy requirements and limitations on the knowledge and computational power of adversarial parties. They also support security protocol verification, using methods that are compatible with informal approaches used in the computational cryptography research community. We illustrate the use of our framework by outlining a proof of functional correctness and security properties for a well-known Oblivious Transfer protocol.

1 Introduction

Interacting abstract state machine modeling frameworks such as I/O Automata, and proof techniques based on invariant assertions, levels of abstraction, and composition, have long been used successfully for proving correctness of distributed algorithms. Security protocols are special cases of distributed algorithms—ones that use cryptographic primitives such as encryption and trap-door functions, and guarantee properties such as secrecy and authentication. Thus, one would expect the same kinds of models and methods to be useful for analyzing security protocols. However, making this work requires additions to the traditional frameworks, including mechanisms for modeling secrecy requirements, and for describing limitations on knowledge and computational power of adversarial parties.

[⋆] Canetti is supported by NSF CyberTrust Grant #430450; Cheung by DFG/NWO bilateral cooperation project 600.050.011.01 Validation of Stochastic Systems (VOSS); Kaynar and Lynch by DARPA/AFOSR MURI Award #F49620-02-1-0325, MURI AFOSR Award #SA2796PO 1-0000243658, NSF Awards #CCR-0326277 and #CCR-0121277, and USAF, AFRL Award #FA9550-04-1-0121; Pereira by the Belgian National Fund for Scientific Research (FNRS); and Segala by MURST project Constraint-based Verification of reactive systems (CoVer).

In this paper, we describe a modeling framework that extends Segala's Probabilistic I/O Automata (PIOA) framework [Seg95,SL95] and supports description of security-related features. Our extension, which we call the *Time-Bounded Task-PIOA* framework, directly models probabilistic and nondeterministic behavior, partial-information adversarial scheduling, and time-bounded computation. We define an *approximate implementation relation* for Time-Bounded Task-PIOAs, $\leq_{neg,pt}$, which captures the notion of *computational indistinguishability*—the idea that a polynomial-time-bounded observer cannot, with non-negligible probability, distinguish the behavior of one automaton from that of another. We prove that $\leq_{neg,pt}$ is transitive and compositional, and we define a type of probabilistic simulation relation that can be used to prove $\leq_{neg,pt}$. We believe that these features are adequate to support formal modeling and verification of typical security protocols, using methods that are compatible with the informal approaches used in the computational cryptography research community.

We illustrate the use of our framework by outlining a proof of functional correctness and security properties for the Oblivious Transfer (OT) protocol of [EGL85,GMW87]. Together, these properties are expressed by a statement, formulated using $\leq_{neg,pt}$, that every possible behavior of the OT protocol can also be realized by an abstract system representing the required functionality. The protocol model consists of the protocol parties, plus an adversary that acts as a message delivery system and hence has access to dynamic information such as ciphertexts. The abstract system includes an *ideal functionality*, i.e., a trusted third party whose security is assumed, together with a simulator.

Between the protocol and abstract system, we define intermediate systems at different levels of abstraction, and prove that each consecutive pair of levels satisfies $\leq_{neg,pt}$. This decomposes the security proof into several stages, each of which addresses some particular aspect of the protocol. In particular, computational reasoning is isolated to a single stage in the proof, where a system using hard-core bits of trap-door functions is shown to implement a system using random bits. For this interesting step, we reformulate the notion of hard-core bits using $\leq_{neg,pt}$, and prove that our reformulation is equivalent to a standard definition in the literature. The proof of $\leq_{neg,pt}$ for this stage is essentially a reduction to the security of hard-core bits, but the reduction is reformulated in terms of $\leq_{neg,pt}$ and composition results for Time-Bounded Task-PIOAs. Other stages are proved using probabilistic simulation relations.

Background and prior work: Traditionally, security protocols have been analyzed using one of two approaches: *formal* or *computational*. In the formal approach, cryptographic operations are modeled purely symbolically, and security of a protocol is expressed in terms of absolute guarantees when the protocol is run against a Dolev-Yao adversary [DY83], which is incapable of breaking the cryptographic primitives. This approach lends itself to rigorous proofs using familiar methods; however, it neglects important computational issues that could render protocols invalid in practice. In contrast, in the computational approach, cryptographic operations are modeled as algorithms operating on bit strings, and security is expressed in terms of probabilistic guarantees when protocols are run against resource-bounded adversaries. This approach treats computational issues realistically, but it does not easily support rigorous proofs. For example, resource-bounded protocol components are often modeled as Interactive Turing Machines (ITMs) [GMR89,Can01]. But rigorous proofs in terms of ITMs are infeasible, because they represent components at too fine a level of detail.

A recent trend in security verification is to combine formal and computational analysis in one framework [LMMS98,PW00,BPW04,BCT04,RMST04,Bla05], by defining computational restrictions for abstract machines. This work provides formal syntax for specifying probabilistic polynomial-time (PPT) processes, and formulates computational security requirements in terms of semantic properties of these processes. Our work follows the same general approach, though with its own unique set of modeling choices.

Our starting point was the *Probabilistic I/O Automata (PIOA)* modeling framework [Seg95,SL95], which is based on abstract machines that allow both probabilistic and nondeterministic choices. In order to resolve nondeterministic choices (a prerequisite for stating probabilistic properties), PIOAs are combined with perfect-information schedulers, which can use full knowledge about the past execution in selecting the next action. This scheduling mechanism is too powerful for analyzing security protocols; e.g., a scheduler's choice of the next action may depend on "secret" information hidden in the states of honest protocol participants, and thus may reveal information about secrets to dishonest participants. Therefore, we augmented the PIOA framework to obtain the *Task-PIOA* framework [CCK$^+$06a,CCK$^+$06b], in which nondeterministic choices are resolved by oblivious *task schedules*, which schedule sets of actions (*tasks*) instead of individual actions. Task-PIOAs support familiar methods of abstraction and composition. They include an implementation relation, $\leq_0$, between automata, based on trace distributions, and probabilistic simulation relations that can be used to prove $\leq_0$.

In this paper, we define *Time-Bounded Task-PIOAs* by imposing time bounds on Task-PIOAs, expressed in terms of bit string encodings of automata constituents. This allows us to define polynomial-time Task-PIOAs and our approximate implementation relation $\leq_{neg,pt}$. We adapt the probabilistic simulation relations of [CCK$^+$06a,CCK$^+$06b] so that they can be used to prove $\leq_{neg,pt}$. Our analysis of Oblivious Transfer follows the style of *Universally Composable (UC) Security* [Can01] and *universal reactive simulatability* [PW01].

In [LMMS98,MMS03,RMST04], a process-algebraic syntax is used to specify protocols, and security properties are specified using an asymptotic observational equivalence on process terms. Nondeterministic choices are resolved by several types of probabilistic schedulers, e.g., special Markov chains or probability distributions on the set of actions. Various restrictions, such as environment- and history-independence, are imposed on these schedulers to enable them to support computational security arguments. In [PW00,PW01,BPW04], protocols are specified as interrupt-driven state machines that interact via a system of ports and buffers, and security is specified in terms of *reactive simulatability*, which expresses computational indistinguishability between the environment's views of the protocol and the abstract functional specification. A distributed protocol, in which machines activate each other by generating explicit "clock" signals, is used for scheduling among the (purely probabilistic) machines. This mechanism is similar to the ones typically used for ITMs [GMR89,Can01].

Our work differs from that of these two schools in our choice of underlying machine model and scheduling mechanism. Also, we follow a different modeling and proof methodology, based on the one typically used for distributed algorithms: we use nondeterminism extensively as a means of abstraction, and organize our proofs using invariants, levels of abstraction, and composition.

In other related work, security analysis is sometimes carried out, informally, in terms of a sequence of *games*, which are similar to our levels of abstraction [Sho04,BR04,Bla05,Hal05].

Overview: Sections 2 and 3 review the PIOA and Task-PIOA frameworks, respectively. Section 4 defines Time-Bounded Task-PIOAs, and the approximate implementation relation $\leq_{neg,pt}$. Section 5 presents our definition of hard-core predicates for trapdoor permutations, in terms of $\leq_{neg,pt}$. Section 6 explains how we model cryptographic protocols and their requirements, illustrating this method with the OT protocol. Section 7 outlines our proofs for OT. Conclusions follow in Section 8. Complete details appear in [CCK$^+$06c].

2 PIOAs

In this section, we summarize basic definitions and results for PIOAs; full definitions, results, and proofs appear in [CCK$^+$06a,CCK$^+$06b].

We let $\mathsf{R}^{\geq 0}$ denote the set of nonnegative reals. We assume that the reader is comfortable with basic notions of probability, such as σ-fields and discrete probability measures. For a discrete probability measure μ on a set X, $supp(\mu)$ denotes the support of μ, that is, the set of elements $x \in X$ such that $\mu(x) \neq 0$. Given set X and element $x \in X$, the *Dirac* measure $\delta(x)$ is the discrete probability measure on X that assigns probability 1 to x.

A *Probabilistic I/O Automaton (PIOA)* $\mathcal{P}$ is a tuple $(Q, \bar{q}, I, O, H, D)$, where: (i) Q is a countable set of *states*, with *start state* $\bar{q} \in Q$; (ii) I, O and H are countable, pairwise disjoint sets of actions, referred to as *input, output and internal actions*, respectively; and (iii) $D \subseteq Q \times (I \cup O \cup H) \times Disc(Q)$ is a *transition relation*, where $Disc(Q)$ is the set of discrete probability measures on Q. An action a is *enabled* in a state q if $(q, a, \mu) \in D$ for some μ. The set $A := I \cup O \cup H$ is called the *action alphabet* of $\mathcal{P}$. If $I = \emptyset$, then $\mathcal{P}$ is *closed*. The set of *external* actions of $\mathcal{P}$ is $I \cup O$ and the set of *locally controlled* actions is $O \cup H$. We assume that $\mathcal{P}$ satisfies:

- **Input enabling:** For every $q \in Q$ and $a \in I$, a is enabled in q.
- **Transition determinism:** For every $q \in Q$ and $a \in A$, there is at most one $\mu \in Disc(Q)$ such that $(q, a, \mu) \in D$.

An *execution fragment* of $\mathcal{P}$ is a finite or infinite sequence $\alpha = q_0\, a_1\, q_1\, a_2\, \ldots$ of alternating states and actions, such that (i) if α is finite, it ends with a state; and (ii) for every non-final i, there is a transition $(q_i, a_{i+1}, \mu) \in D$ with $q_{i+1} \in supp(\mu)$. We write $fstate(\alpha)$ for q_0, and if α is finite, we write $lstate(\alpha)$ for its last state. $\mathsf{Frags}(\mathcal{P})$ (resp., $\mathsf{Frags}^*(\mathcal{P})$) denotes the set of all (resp., all finite) execution fragments of $\mathcal{P}$. An *execution* of $\mathcal{P}$ is an execution fragment α

with $fstate(\alpha) = \bar{q}$. $\mathsf{Execs}(\mathcal{P})$ (resp., $\mathsf{Execs}^*(\mathcal{P})$) denotes the set of all (resp., all finite) executions of $\mathcal{P}$. The *trace* of an execution fragment α, written $\mathsf{trace}(\alpha)$, is the restriction of α to the external actions of $\mathcal{P}$.

A PIOA, together with a *scheduler* that chooses the sequence of actions to be performed, gives rise to a unique *probabilistic execution*, and thereby, to a unique probability distribution on traces. Traditionally, the schedulers used for PIOAs have been perfect-information schedulers, which can use full knowledge about the past execution in selecting the next action.

Two PIOAs $\mathcal{P}_i = (Q_i, \bar{q}_i, I_i, O_i, H_i, D_i)$, $i \in \{1, 2\}$, are said to be *compatible* if $A_i \cap H_j = O_i \cap O_j = \emptyset$ whenever $i \neq j$. In that case, we define their *composition* $\mathcal{P}_1 \| \mathcal{P}_2$ to be the PIOA $(Q_1 \times Q_2,\ (\bar{q}_1, \bar{q}_2),\ (I_1 \cup I_2) \setminus (O_1 \cup O_2),\ O_1 \cup O_2,\ H_1 \cup H_2, D)$, where D is the set of triples $((q_1, q_2), a, \mu_1 \times \mu_2)$ such that (i) a is enabled in some q_i; and (ii) for every i, if $a \in A_i$ then $(q_i, a, \mu_i) \in D_i$, and otherwise $\mu_i = \delta(q_i)$. A *hiding* operator is also available for PIOAs: given $\mathcal{P} = (Q, \bar{q}, I, O, H, D)$ and $S \subseteq O$, $hide(\mathcal{P}, S)$ is defined to be $(Q, \bar{q}, I, O', H', D)$, where $O' = O \setminus S$ and $H' = H \cup S$.

3 Task-PIOAs

The perfect-information schedulers that have been used to resolve nondeterministic choices in PIOAs are too powerful for computational analysis of security protocols; e.g., a scheduler's choice of the next action may depend on "secret" information hidden in the states of honest protocol participants, and thus may reveal information about secrets to dishonest participants. To avoid this problem, we resolve nondeterminism using a more restrictive, oblivious task mechanism. Again, full definitions, results, and proofs appear in [CCK$^+$06a,CCK$^+$06b].

Basic definitions: A *Task-PIOA* $\mathcal{T}$ is a pair $(\mathcal{P}, R)$, where $\mathcal{P} = (Q, \bar{q}, I, O, H, D)$ is a PIOA (satisfying transition determinism and input enabling), and R is an equivalence relation on the locally-controlled actions $O \cup H$. The equivalence classes of R are called *tasks*. A task T is an *output task* if $T \subseteq O$, and similarly for *internal tasks*. Unless otherwise stated, we will use terminology inherited from the PIOA setting. We require the following axiom for task-PIOAs:

- **Action determinism:** For every state $q \in Q$ and every task $T \in R$, there is at most one action $a \in T$ that is enabled in q.

In case some $a \in T$ is enabled in q, we say that T is *enabled* in q. If T is enabled in every state from a set S, then T is *enabled* in S.

A *task schedule* for $\mathcal{T} = (\mathcal{P}, R)$ is a finite or infinite sequence $\rho = T_1\, T_2\, \ldots$ of tasks in R. A task schedule is *oblivious*, in the sense that it does not depend on dynamic information generated during execution. Because of the action-determinism assumption for task-PIOAs and the transition-determinism assumption for PIOAs, ρ can be used to generate a unique probabilistic execution, and hence, a unique trace distribution, of $\mathcal{P}$. One can do this by repeatedly scheduling tasks, each of which determines at most one transition of $\mathcal{P}$.

Formally, we define an operation *apply* that "applies" a task schedule to a finite execution fragment, by applying tasks one at a time. To apply a task T to an execution fragment α: (i) if T is not enabled in $lstate(\alpha)$, then the result

```
Automaton $Src(D, \mu)$:
Signature:
Input:                          Internal:
    none                            $chooserand$
Output:
    $rand(d), d \in D$

State:
    $chosenval \in D \cup \{\perp\}$, initially $\perp$

Transitions:
$chooserand$                              $rand(d)$
Pre: $chosenval = \perp$                  Pre: $d = chosenval \neq \perp$
Eff:                                      Eff:
    $chosenval :=$ choose-random$(D, \mu)$      none

Tasks: $\{chooserand\}$, $\{rand(*)\}$
```

Fig. 1. Code for $Src(D, \mu)$

is α itself; (ii) otherwise, due to action- and transition-determinism, there is a unique transition from $lstate(\alpha)$ with a label in T, and the result is α extended with that transition. We generalize this construction from a single fragment α to a discrete probability measure η on execution fragments.

Now consider η of the form $\delta(\bar{q})$. For every task schedule ρ, $apply(\delta(\bar{q}), \rho)$, the results of applying ρ to $\bar{q}$, is said to be a *probabilistic execution* of $\mathcal{T}$. This can be viewed as a probabilistic tree generated by running $\mathcal{P}$ from its start state, resolving nondeterministic choices according to ρ. The *trace distribution* induced by ρ, $tdist(\rho)$, is the image measure of $apply(\delta(\bar{q}), \rho)$ under the measurable function trace. A *trace distribution* of $\mathcal{T}$ is $tdist(\rho)$ for any ρ, and we define $tdists(\mathcal{T}) := \{tdist(\rho) \mid \rho \text{ is a task schedule for } \mathcal{T}\}$.

Figure 1 contains a specification of a *random source* task-PIOA $Src(D, \mu)$, which we use in our OT models. Src draws an element d from the distribution μ using the action $chooserand$ and outputs that element using the action $rand(d)$. It has two tasks: internal task $\{chooserand\}$ and output task $\{rand(d) \mid d \in D\}$. Notice, since all $rand(d)$ actions are grouped into a single task, a task scheduler decides whether or not to output the random element without "knowing" which element has been drawn.

Given compatible task-PIOAs $\mathcal{T}_i = (\mathcal{P}_i, R_i)$, $i \in \{1, 2\}$, we define their *composition* $\mathcal{T}_1 \| \mathcal{T}_2$ to be the task-PIOA $(\mathcal{P}_1 \| \mathcal{P}_2, R_1 \cup R_2)$. Note that $R_1 \cup R_2$ is an equivalence relation because compatibility requires sets of locally controlled actions to be disjoint. It is also easy to check that action determinism is preserved under composition. The hiding operation for task-PIOAs hides all actions in the specified tasks: given task-PIOA $\mathcal{T} = (\mathcal{P}, R)$ and a set $U \subseteq R$ of output tasks, $hide(\mathcal{T}, U)$ is defined to be $(hide(\mathcal{P}, \bigcup U), R)$.

Implementation: An implementation relation expresses the idea that every behavior of one automaton is also a behavior of a second automaton. In that case, it is "safe" to replace the second automaton with the first in a larger system. This notion mades sense only if the two automata interact with the environment via the same interface: thus, two task-PIOAs $\mathcal{T}_1$ and $\mathcal{T}_2$ are *comparable* if $I_1 = I_2$ and $O_1 = O_2$.

If $\mathcal{T}$ and $\mathcal{E}$ are task-PIOAs, then $\mathcal{E}$ is said to be an *environment* for $\mathcal{T}$ if $\mathcal{T}$ and $\mathcal{E}$ are compatible and $\mathcal{T} \| \mathcal{E}$ is closed. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are comparable task-PIOAs, then $\mathcal{T}_1$ *implements* $\mathcal{T}_2$, written $\mathcal{T}_1 \leq_0 \mathcal{T}_2$, if $tdists(\mathcal{T}_1 \| \mathcal{E}) \subseteq tdists(\mathcal{T}_2 \| \mathcal{E})$ for every environment $\mathcal{E}$ for both $\mathcal{T}_1$ and $\mathcal{T}_2$. Equivalently, given any task schedule ρ_1 for $\mathcal{T}_1 \| \mathcal{E}$, there is a task schedule ρ_2 for $\mathcal{T}_2 \| \mathcal{E}$ such that $tdist(\rho_1) = tdist(\rho_2)$. Since we require equality of corresponding trace distributions, this is also referred to as *perfect implementation*. Transitivity of $\leq_0$ is trivial and compositionality is proved in [CCK+06a,CCK+06b].

Simulation relations: In [CCK+06a,CCK+06b], we define a new type of probabilistic simulation relation for task-PIOAs, and prove that such simulation relations are sound for proving $\leq_0$. Here we outline the definition. Let $\mathcal{T}_1 = (\mathcal{P}_1, R_1)$ and $\mathcal{T}_2 = (\mathcal{P}_2, R_2)$ be two comparable closed task-PIOAs. A simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$ is a relation R from discrete probability measures on $\mathsf{Frags}^*(\mathcal{P}_1)$ to discrete probability measures on $\mathsf{Frags}^*(\mathcal{P}_2)$ satisfying a number of conditions: (i) If $(\eta_1, \eta_2) \in R$, then η_1 and η_2 induce the same trace distribution; (ii) The Dirac measures $\delta(\bar{q}_1)$ and $\delta(\bar{q}_2)$ are related by R; and (iii) There is a function $corr : ({R_1}^* \times R_1) \rightarrow {R_2}^*$ such that, given $(\eta_1, \eta_2) \in R$, a task schedule ρ for $\mathcal{T}_1$ and a task T of $\mathcal{T}_1$, the measures $apply(\eta_1, T)$ and $apply(\eta_2, corr(\rho, T))$ are related by the expansion of R. (Expansion is a standard operation on relations between discrete measures (cf. [CCK+06b]).) That is, given a task schedule ρ for $\mathcal{T}_1$ that has already been matched in $\mathcal{T}_2$, and a new task T, $corr$ matches T with a finite sequence of tasks of $\mathcal{T}_2$, and the result of scheduling T after η_1 is again related to the result of scheduling the sequence $corr(\rho, T)$ after η_2.

Adversarial scheduling: The standard scheduling mechanism in the security protocol community is an *adversarial scheduler*—a resource-bounded algorithmic entity that determines the next move adaptively, based on its own view of the computation so far. Our oblivious task schedules do not directly capture the adaptivity of adversarial schedulers. To address this issue, we separate scheduling concerns into two parts: We model the adaptive adversarial scheduler as a system component, e.g., a message delivery service that can eavesdrop on the communications and control the order of message delivery. Such a service has access to partial information about the execution: it sees information that other components communicate to it during execution, but not "secret information" that these components hide. Its choices may be essential to the analysis of the protocol. On the other hand, basic scheduling choices are resolved by a task schedule sequence, chosen nondeterministically in advance. These choices are less important; for example, in the OT protocol, both the transmitter and receiver make random choices, but it is inconsequential which does so first.

4 Time-Bounded Task-PIOAs

A key assumption of computational cryptography is that certain problems cannot be solved with non-negligible probability by resource-bounded entities. In particular, adversaries are assumed to be resource-bounded. To express such bounds formally, we introduce the notion of a *time-bounded task-PIOA*, which assumes bit-string representations of automata constituents and imposes time bounds on Turing machines that decode the representations and compute the next action and next state. Complete details appear in [CCK$^+$06c].

Basic definitions: We assume a standard bit-string representation for actions and tasks of task-PIOAs. A task-PIOA $\mathcal{T}$ is said to be *b-time-bounded*, where $b \in \mathsf{R}^{\geq 0}$, provided: (i) every state and transition has a bit-string representation, and the length of the representation of every automaton part is at most b; (ii) there is a deterministic Turing machine that decides whether a given representation of a candidate automaton part is indeed such an automaton part, and this machine runs in time at most b; (iii) there is a deterministic Turing machine that, given a state and a task of $\mathcal{T}$, determines the next action (or indicates "no action"), in time at most b; and (iv) there is a probabilistic Turing machine that, given a state and an action of $\mathcal{T}$, determines the next state of $\mathcal{T}$, in time at most b. Furthermore, each of these Turing machines can be described using a bit string of length at most b, according to some standard encoding of Turing machines.

Composing two compatible time-bounded task-PIOAs yields a time-bounded task-PIOA with a bound that is linear in the sum of the original bounds. Similarly, hiding changes the time bound by a linear factor. We say that task schedule ρ is *b-bounded* if $|\rho| \leq b$, that is, ρ is finite and contains at most b tasks.

Task-PIOA families: Typically, a computational hardness assumption states that, as the size of a problem grows, the success probability of a resource-bounded entity trying to solve the problem diminishes quickly. The size of a problem is expressed in terms of a *security parameter* $k \in \mathsf{N}$, e.g., the key length for an encryption scheme. Accordingly, we define families of task-PIOAs indexed by a security parameter: a *task-PIOA family* $\overline{\mathcal{T}}$ is an indexed set $\{\mathcal{T}_k\}_{k \in \mathsf{N}}$ of task-PIOAs. The notion of time bound is also expressed in terms of the security parameter; namely, given $b : \mathsf{N} \to \mathsf{R}^{\geq 0}$, we say that $\overline{\mathcal{T}}$ is *b-time-bounded* if every $\mathcal{T}_k$ is $b(k)$ time-bounded. Task-PIOA family $\overline{\mathcal{T}}$ is said to be *polynomial-time-bounded* provided that $\overline{\mathcal{T}}$ is p-time-bounded for some polynomial p. Compatibility and parallel composition for task-PIOA families are defined pointwise. Results for composition and hiding carry over easily from those for time-bounded task-PIOAs.

Approximate implementation: Our notion of approximate implementation allows errors in the emulation and takes into account time bounds of various automata involved. Let $\mathcal{T}$ be a closed task-PIOA with a special output action *accept* and let ρ be a task schedule for $\mathcal{T}$. The *acceptance probability* with respect to $\mathcal{T}$ and ρ is defined to be:

$$Paccept(\mathcal{T}, \rho) := \Pr[\beta \leftarrow tdist(\mathcal{T}, \rho) : \beta \text{ contains } accept],$$

where $\beta \leftarrow tdist(\mathcal{T}, \rho)$ means β is drawn randomly from $tdist(\mathcal{T}, \rho)$.

From now on, we assume that every environment has *accept* as an output. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable task-PIOAs, $\epsilon, b \in \mathsf{R}^{\geq 0}$, and $b_1, b_2 \in \mathsf{N}$. Then we define $\mathcal{T}_1 \leq_{\epsilon,b,b_1,b_2} \mathcal{T}_2$ as follows: Given any b-time-bounded environment $\mathcal{E}$ for both $\mathcal{T}_1$ and $\mathcal{T}_2$, and any b_1-bounded task schedule ρ_1 for $\mathcal{T}_1 \| \mathcal{E}$, there is a b_2-bounded task schedule ρ_2 for $\mathcal{T}_2 \| \mathcal{E}$ such that $|Paccept(\mathcal{T}_1 \| \mathcal{E}, \rho_1) - Paccept(\mathcal{T}_2 \| \mathcal{E}, \rho_2)| \leq \epsilon$. In other words, from the perspective of a b-time-bounded environment, $\mathcal{T}_1$ and $\mathcal{T}_2$ "look almost the same" in the sense that $\mathcal{T}_2$ can use at most b_2 steps to emulate at most b_1 steps of $\mathcal{T}_1$. The relation $\leq_{\epsilon,b,b_1,b_2}$ is transitive and preserved under composition and hiding, with certain adjustments to errors and time bounds.

We extend the relation $\leq_{\epsilon,b,b_1,b_2}$ to task-PIOA families in the obvious way: Let $\overline{\mathcal{T}}_1 = \{(\mathcal{T}_1)_k\}_{k \in \mathsf{N}}$ and $\overline{\mathcal{T}}_2 = \{(\mathcal{T}_2)_k\}_{k \in \mathsf{N}}$ be (pointwise) *comparable* task-PIOA families, $\epsilon, b : \mathsf{N} \to \mathsf{R}^{\geq 0}$, and $b_1, b_2 : \mathsf{N} \to \mathsf{N}$. Then we say that $\overline{\mathcal{T}}_1 \leq_{\epsilon,b,b_1,b_2} \overline{\mathcal{T}}_2$ provided that $(\mathcal{T}_1)_k \leq_{\epsilon(k),b(k),b_1(k),b_2(k)} (\mathcal{T}_2)_k$ for every k.

Restricting attention to negligible error and polynomial time bounds, we obtain a generic version of approximate implementation, $\leq_{neg,pt}$; we will used this relation throughout our analysis to express computational security properties. A function $\epsilon : \mathsf{N} \to \mathsf{R}^{\geq 0}$ is said to be *negligible* if, for every constant $c \in \mathsf{R}^{\geq 0}$, there exists $k_0 \in \mathsf{N}$ such that $\epsilon(k) < \frac{1}{k^c}$ for all $k \geq k_0$. (In other words, ϵ diminishes more quickly than the reciprocal of any polynomial.) We say that $\overline{\mathcal{T}}_1 \leq_{neg,pt} \overline{\mathcal{T}}_2$ if, for all polynomials p and p_1, there is a polynomial p_2 and a negligible function ϵ such that $\overline{\mathcal{T}}_1 \leq_{\epsilon,p,p_1,p_2} \overline{\mathcal{T}}_2$. We show that $\leq_{neg,pt}$ is transitive and preserved under composition; for composition, we need to assume polynomial time bounds for one of the task-PIOA families.

Theorem 1. *Suppose $\overline{\mathcal{T}}_1$, $\overline{\mathcal{T}}_2$ and $\overline{\mathcal{T}}_3$ are three comparable task-PIOA families such that $\overline{\mathcal{T}}_1 \leq_{neg,pt} \overline{\mathcal{T}}_2$ and $\overline{\mathcal{T}}_2 \leq_{neg,pt} \overline{\mathcal{T}}_3$. Then $\overline{\mathcal{T}}_1 \leq_{neg,pt} \overline{\mathcal{T}}_3$.*

Theorem 2. *Suppose $\overline{\mathcal{T}}_1$, $\overline{\mathcal{T}}_2$ are comparable families of task-PIOAs such that $\overline{\mathcal{T}}_1 \leq_{neg,pt} \overline{\mathcal{T}}_2$, and suppose $\overline{\mathcal{T}}_3$ is a polynomial time-bounded task-PIOA family, compatible with both $\overline{\mathcal{T}}_1$ and $\overline{\mathcal{T}}_2$. Then $\overline{\mathcal{T}}_1 \| \overline{\mathcal{T}}_3 \leq_{neg,pt} \overline{\mathcal{T}}_2 \| \overline{\mathcal{T}}_3$.*

Simulation relations: In order to use simulation relations in a setting with time bounds, we impose an additional assumption on the length of matching task schedules: Given $c \in \mathsf{N}$, a simulation R is said to be *c-bounded* if $|corr(\rho_1, T)| \leq c$ for all ρ_1 and T. We have the following theorem:

Theorem 3. *Let $\overline{\mathcal{T}}_1$ and $\overline{\mathcal{T}}_2$ be comparable task-PIOA families, $c \in \mathsf{N}$. Suppose that for every polynomial p, every k, and every $p(k)$-bounded environment $\mathcal{E}_k$ for $(\overline{\mathcal{T}}_1)_k$ and $(\overline{\mathcal{T}}_2)_k$, there exists a c-bounded simulation R_k from $(\overline{\mathcal{T}}_1)_k \| \mathcal{E}_k$ to $(\overline{\mathcal{T}}_2)_k \| \mathcal{E}_k$. Then $\overline{\mathcal{T}}_1 \leq_{neg,pt} \overline{\mathcal{T}}_2$.*

Proof. In [CCK$^+$06b], soundness of simulation is proved as follows: Given a simulation R between closed task-PIOAs $\mathcal{T}_1$ and $\mathcal{T}_2$, and a task schedule $\rho_1 = T_1, T_2, \ldots$ for $\mathcal{T}_1$, we construct a task schedule ρ_2 for $\mathcal{T}_2$ by concatenating sequences returned by *corr*: $\rho_2 := corr(\lambda, T_1) \ldots corr(T_1 \ldots T_n, T_{n+1}) \ldots$ (λ denotes the empty sequence.) We then prove that $tdist(\rho_1) = tdist(\rho_2)$. Note that, if R is c-bounded, then the length of ρ_2 is at most $c \cdot |\rho_1|$.

Now let polynomials p and p_1 be given as in the definition of $\leq_{neg,pt}$. Let p_2 be $c \cdot p_1$ and ϵ be the constant-0 function. Using the proof outlined above, it is easy to check that p_2 and ϵ satisfy the requirements for $\leq_{neg,pt}$. $\square$

5 Hard-Core Predicates

In this section, we reformulate the standard definition of hard-core predicates using our approximate implementation relation $\leq_{neg,pt}$. This is an important step towards a fully formalized computational analysis of the OT protocol, since the security of OT relies on properties of hard-core predicates.[7] For the rest of the paper, we fix a family $\overline{D} = \{D_k\}_{k\in\mathsf{N}}$ of finite domains and a family $\overline{Tdp} = \{Tdp_k\}_{k\in\mathsf{N}}$ of sets of trap-door permutations such that D_k is the domain of every $f \in Tdp_k$.

In the traditional definition, a function $B : \bigcup_{k\in\mathsf{N}} D_k \rightarrow \{0,1\}$ is said to be a hard-core predicate for $\overline{Tdp}$ if, whenever f and z are chosen randomly from Tdp_k and D_k, respectively, the bit $B(f^{-1}(z))$ "appears random" to a probabilistic-polynomial-time observer, even if f and z are given to the observer as inputs. This captures the idea that $f^{-1}(z)$ cannot be computed efficiently from f and z. More precisely, a hard-core predicate is defined by the following slight reformulation of Definition 2.5.1 of [Gol01]:

Definition 1. *A* hard-core predicate *for $\overline{D}$ and $\overline{Tdp}$ is a predicate $B : \bigcup_{k\in\mathsf{N}} D_k \rightarrow \{0,1\}$ such that (i) B is polynomial-time computable; and (ii) for every probabilistic polynomial-time predicate $G = \{G_k\}_{k\in\mathsf{N}}$* [8]*, there is a negligible function ϵ such that, for all k,*

$$\left| \begin{array}{l} \Pr[\ f \leftarrow Tdp_k; \\ \quad z \leftarrow D_k; \\ \quad b \leftarrow B(f^{-1}(z)) : \\ \quad G_k(f,z,b) = 1 \quad] \end{array} - \begin{array}{l} \Pr[\ f \leftarrow Tdp_k; \\ \quad z \leftarrow D_k; \\ \quad b \leftarrow \{0,1\} : \\ \quad G_k(f,z,b) = 1\ ] \end{array} \right| \leq \epsilon(k).$$

Note that, when A is a finite set, the notation $x \leftarrow A$ means that x is selected randomly (according to the uniform distribution) from A.

Our new definition uses $\leq_{neg,pt}$ to express the idea that $B(f^{-1}(z))$ "appears random". We define two task-PIOA families, $\overline{SH}$ and $\overline{SHR}$. The former outputs random elements f and z from Tdp_k and D_k, and the bit $B(f^{-1}(z))$. The latter does the same except $B(f^{-1}(z))$ is replaced by a random element from $\{0,1\}$. Then B is said to be a hard-core predicate for $\overline{D}$ and $\overline{Tdp}$ if $\overline{SH} \leq_{neg,pt} \overline{SHR}$.

Definition 2. *B is said to be a* hard-core predicate *for $\overline{D}$ and $\overline{Tdp}$ if $\overline{SH} \leq_{neg,pt} \overline{SHR}$, where $\overline{SH}$ and $\overline{SHR}$ are defined as follows. For each $k \in \mathsf{N}$, let μ_k and μ'_k denote the uniform distributions on Tdp_k and D_k, respectively. Let μ'' be the uniform distribution on $\{0,1\}$.*

$\overline{SH}$ is defined to be $hide(\overline{Src_{tdp}} \| \overline{Src_{yval}} \| \overline{H}, \{rand_{yval}(*)\})$, where (i) $\overline{Src_{tdp}} = \{Src_{tdp}(Tdp_k, \mu_k)\}_{k\in\mathsf{N}}$; (ii) $\overline{Src_{yval}} = \{Src_{yval}(D_k, \mu'_k)\}_{k\in\mathsf{N}}$; and (iii) each H_k obtains f from $Src_{tdp}(Tdp_k, \mu_k)$ and y from $Src_{yval}(D_k, \mu'_k)$, and outputs $z := f(y)$ via action $rand_{zval}$ and $b := B(y)$ via action $rand_{bval}$ (cf. Figure 2). Since f is a permutation, this is equivalent to choosing z randomly and computing y as $f^{-1}(z)$.

[7] In [MMS03], an OT protocol using hard-core bits is analyzed in a process-algebraic setting. However, that work does not include a formalization of hard-core predicates.

[8] This is defined to be a family of predicates that can be evaluated by a family $(M_k)_k$ of probabilistic polynomial-time Turing machines.

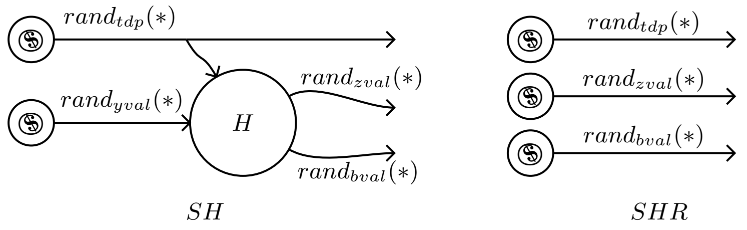

Fig. 2. SH and SHR

$\overline{SHR}$ is defined to be $\overline{Src_{tdp}} \| \overline{Src_{zval}} \| \overline{Src_{bval}}$, where (i) $\overline{Src_{tdp}}$ is as in $\overline{SH}$; (ii) $\overline{Src_{zval}} = \{Src_{zval}(D_k, \mu'_k)\}_{k \in \mathsf{N}}$; and (iii) $\overline{Src_{bval}} = \{Src_{bval}(\{0,1\}, \mu'')_k\}_{k \in \mathsf{N}}$.

These two systems are represented in Fig. 2. There, the automata labeled with $\textcircled{\$}$ represent the random source automata. We claim that these two definitions of hard-core bits are equivalent:

Theorem 4. *B is a hard-core predicate for $\overline{D}$ and $\overline{Tdp}$ according to Definition 1 if and only if B is a hard-core predicate for $\overline{D}$ and $\overline{Tdp}$ according to Definition 2.*

To illustrate how our new definition of hard-core predicates can be exploited in analyzing protocols, we show that a hard-core predicate can be applied twice, and a probabilistic polynomial-time environment still cannot distinguish the outputs from random values. We use this fact in our OT proof, in a situation where the transmitter applies the hard-core predicate to both $f^{-1}(zval(0))$ and $f^{-1}(zval(1))$, where f is the chosen trap-door function.

We show, if B is a hard-core predicate, then no probabilistic polynomial-time environment can distinguish distribution $(f, z(0), z(1), B(f^{-1}(z(0))), B(f^{-1}(z(1))))$ from distribution $(f, z(0), z(1), b(0), b(1))$, where f is a randomly-chosen trap-door permutation, $z(0)$ and $z(1)$ are randomly-chosen elements of D_k, and $b(0)$ and $b(1)$ are randomly-chosen bits. We do this by defining two task-PIOA families, $\overline{SH2}$ and $\overline{SHR2}$, that produce the two distributions, and showing that $\overline{SH2} \leq_{neg,pt} \overline{SHR2}$. Task-PIOA family $\overline{SH2}$ is defined as

$$hide(\overline{Src_{tdp}} \| \overline{Src_{yval0}} \| \overline{Src_{yval1}} \| \overline{H0} \| \overline{H1}, \{rand(*)_{yval0}, rand(*)_{yval1}\}),$$

where $\overline{Src_{tdp}}$ is as in the definition of $\overline{SH}$, $\overline{Src_{yval0}}$ and $\overline{Src_{yval1}}$ are isomorphic to $\overline{Src_{yval}}$ in $\overline{SH}$, and $\overline{H0}$ and $\overline{H1}$ are two instances of $\overline{H}$ (with appropriate renaming of actions). Task-PIOA family $\overline{SHR2}$ is defined as

$$(\overline{Src_{tdp}} \| \overline{Src_{zval0}} \| \overline{Src_{zval1}} \| \overline{Src_{bval0}} \| \overline{Src_{bval1}}),$$

where $\overline{Src_{tdp}}$ is as in $\overline{SH2}$, $\overline{Src_{zval0}}$ and $\overline{Src_{zval1}}$ are isomorphic to $\overline{Src_{zval}}$ in $\overline{SHR}$, and $\overline{Src_{bval0}}$ and $\overline{Src_{bval1}}$ are isomorphic to $\overline{Src_{bval}}$ in $\overline{SHR}$.

Theorem 5. *If B is a hard-core predicate, then $\overline{SH2} \leq_{neg,pt} \overline{SHR2}$.*

Proof. Theorem 4 implies that $\overline{SH} \leq_{neg,pt} \overline{SHR}$. To prove that $\overline{SH2} \leq_{neg,pt} \overline{SHR2}$, we introduce a new task-PIOA family $\overline{Int}$, which is intermediate between $\overline{SH2}$ and $\overline{SHR2}$. $\overline{Int}$ is defined as

$$hide(\overline{Src_{tdp}} \| \overline{Src_{yval0}} \| \overline{H0} \| \overline{Src_{zval1}} \| \overline{Src_{bval1}}, \{rand(*)_{yval0}\}),$$

where $\overline{Src_{tdp}}$ is exactly as in $\overline{SH2}$ and $\overline{SHR2}$; $\overline{Src_{yval0}}$ and $\overline{H0}$ are as in $\overline{SH2}$; and $\overline{Src_{zval1}}$ and $\overline{Src_{bval1}}$ are as in $\overline{SHR2}$. Thus, $\overline{Int}$ generates $bval0$ using the hard-core predicate $\overline{B}$, as in $\overline{SH2}$, and generates $bval1$ randomly, as in $\overline{SHR2}$.

To see that $\overline{SH2} \leq_{neg,pt} \overline{Int}$, note that Definition 1 implies that

$$hide(\overline{Src_{tdp}} \| \overline{Src_{yval1}} \| \overline{H1}, \{rand(*)_{yval1}\}) \leq_{neg,pt} \overline{Src_{tdp}} \| \overline{Src_{zval1}} \| \overline{Src_{bval1}},$$

because these two systems are simple renamings of $\overline{SH}$ and $\overline{SHR}$. Now let $\overline{I}$ be the task-PIOA family $hide(\overline{Src_{yval0}} \| \overline{H0}, \{rand(*)_{yval0}\}$. It is easy to see, from the code for the two components of $\overline{I}$, that $\overline{I}$ is polynomial-time-bounded. Then by Theorem 2,

$$hide(\overline{Src_{tdp}} \| \overline{Src_{yval1}} \| \overline{H1}, \{rand(*)_{yval1}\}) \| \overline{I} \leq_{neg,pt} \overline{Src_{tdp}} \| \overline{Src_{zval1}} \| \overline{Src_{bval1}} \| \overline{I}.$$

Since the left-hand side of this relation is $\overline{SH2}$ and the right-hand side is $\overline{Int}$, this implies $\overline{SH2} \leq_{neg,pt} \overline{Int}$.

Similarly, $\overline{Int} \leq_{neg,pt} \overline{SHR2}$. Since $\overline{SH2} \leq_{neg,pt} \overline{Int}$ and $\overline{Int} \leq_{neg,pt} \overline{SHR2}$, transitivity of $\leq_{neg,pt}$ (Theorem 1) implies that $\overline{SH2} \leq_{neg,pt} \overline{SHR2}$. □

6 Computational Security

Here, we explain how we define the security of cryptographic protocols, illustrating with our OT example. Our method follows the general outline in [PW01,Can01], which in turn follows standard practice in the computational cryptography community. We first define a task-PIOA specifying the functionality the protocol is supposed to realize, then specify task-PIOAs describing the protocol, and finally, define what it means for a protocol to securely realize its specification.

The *functionality* task-PIOA represents a "trusted party" that receives protocol inputs and returns protocol outputs, at various locations. See the full version of [Can01] for many examples of classical cryptographic functionalities. The functionality we use for Oblivious Transfer is a task-PIOA *Funct* that behaves as follows. First, it waits for two bits (x_0, x_1) representing the inputs for the protocol's first party (the *transmitter*), and one bit i representing the input for the protocol's second party (the *receiver*). Then, it outputs the bit x_i to the receiver, and nothing to the transmitter.

Since the definitions of protocols are typically parameterized by a security parameter k, we define a protocol as a task-PIOA family $\overline{\pi} = \{\pi_k\}_{k \in \mathsf{N}}$, where π_k is the composition of the task-PIOAs specifying the roles of the different protocol participants for parameter k. Given families $\overline{D}$ and $\overline{Tdp}$, and a parameter k, the OT protocol we consider executes as follows. First the transmitter *Trans* selects a trapdoor permutation f from Tdp_k, together with its inverse f^{-1}, and sends f to the receiver *Rec*. Then, using its input bit i and two randomly selected elements (y_0, y_1) of D_k, *Rec* computes the pair $(z_0, z_1) = (f^{1-i}(y_0), f^i(y_1))$, and sends it in a second protocol message to *Trans*. Finally, using its input bits (x_0, x_1), *Trans* computes the pair $(b_0, b_1) = (x_0 \oplus B(f^{-1}(z_0)), x_1 \oplus B(f^{-1}(z_1)))$ and sends it to *Rec*, who can now recover x_i as $B(y_i) \oplus b_i$.

In order to define the security of a protocol with respect to a functionality, we must specify a particular class of adversaries. Depending on the context, adversaries may have different capabilities: they may have passive or active access to the network, may be able to corrupt parties (either statically or dynamically), may assume partial or full control of the parties, etc. Various scenarios are discussed in [Can01]. We specify a particular class of adversaries by defining appropriate restrictions on the signature and transition relation of adversary task-PIOAs. By composing an adversary task-PIOA Adv_k with a protocol task-PIOA π_k, we obtain what we call the *real system*.

For the OT protocol, we consider polynomial-time-bounded families of adversaries. The adversaries have passive access to protocol messages: they receive and deliver messages (possibly delaying, reordering, or losing messages), but do not compose messages of their own. They may corrupt parties only statically (at the start of execution). They are "honest-but-curious": they obtain access to all internal information of the corrupted parties, but the parties continue to follow the protocol definition. In this paper, we discuss only one corruption case, in which only the receiver is corrupted. In this case, the adversary gains access to the input and output of *Rec* (that is, i and x_i), and to its internal choices (that is, y_0 and y_1). However, as we said above, *Rec* continues to follow the protocol definition, so we model it as a component distinct from the adversary.

In order to prove that the protocol realizes the functionality, we show that, for every adversary family $\overline{\mathit{Adv}}$ of the considered class, there is another task-PIOA family $\overline{\mathit{Sim}}$, called a *simulator*, that can mimic the behavior of $\overline{\mathit{Adv}}$ by interacting with the functionality. This proves that the protocol does not reveal to the adversary any information it does not need to reveal—that is, anything not revealed by the functionality itself.

The quality of emulation is evaluated from the viewpoint of one last task-PIOA, the *environment*. Thus, security of the protocol says that no environment can efficiently decide if it is interacting with the real system, or with the composition of the functionality and the simulator (we call this composition the *ideal system*). This indistinguishability condition is formalized as follows:

Theorem 6. *Let $\overline{RS}$ be a real-system family, in which the family $\overline{\mathit{Adv}}$ of adversary automata is polynomial-time-bounded. Then there exists an ideal-system family $\overline{IS}$, in which the family $\overline{\mathit{Sim}}$ is polynomial-time-bounded, and such that $\overline{RS} \leq_{neg,pt} \overline{IS}$.*

In the statement of Theorem 6, quantification over environments is encapsulated within the definition of $\leq_{neg,pt}$: $\overline{RS} \leq_{neg,pt} \overline{IS}$ says, for every polynomial time-bounded environment family $\overline{\mathit{Env}}$ and every polynomial-bounded task schedule for $\overline{RS}\|\overline{\mathit{Env}}$, there is a polynomial-bounded task schedule for $\overline{IS}\|\overline{\mathit{Env}}$ such that the acceptance probabilities in these two systems differ by a negligible amount.

7 Levels-of-Abstraction Proofs

In order to prove that Theorem 6 holds for the OT protocol and the adversaries of Section 6, we show the existence of an ideal system family $\overline{IS}$ with $\overline{RS} \leq_{neg,pt} \overline{IS}$.

To that end, we build a structured simulator family $\overline{SSim}$ from any adversary family $\overline{Adv}$: for every index k, $SSim_k$ is the composition of Adv_k with an abstract version of π_k based on a task-PIOA $TR(D_k, Tdp_k)$. TR works as follows. First, it selects and sends a random element f of Tdp_k, as *Trans* would. Then, when Adv_k has delivered f, TR emulates *Rec*: it chooses a random pair (y_0, y_1) of elements of D_k, and sends the second protocol message, computed as the pair $(f^{1-i}(y_0), f^i(y_1))$. Next, TR generates the third protocol message using the bit x_i obtained from the *Funct*, which TR obtains because x_i is an output at the corrupted receiver. Namely, TR computes b_i as $B(y_i) \oplus x_i$ and b_{1-i} as a random bit. Observe that, if $\overline{Adv}$ is polynomial-time-bounded, then so is $\overline{SSim}$. Also, if we define $\overline{SIS}$ by $SIS_k = Funct \| SSim_k$, then $\overline{SIS}$ is an ideal system family.

In showing that $\overline{RS} \leq_{neg,pt} \overline{SIS}$, we define two intermediate families of systems, $\overline{Int1}$ and $\overline{Int2}$, and decompose the proof into showing three subgoals: $\overline{RS} \leq_{neg,pt} \overline{Int1}$, $\overline{Int1} \leq_{neg,pt} \overline{Int2}$, and $\overline{Int2} \leq_{neg,pt} \overline{SIS}$. All arguments involving computational indistinguishability and other cryptographic issues are isolated to the middle level, namely, $\overline{Int1} \leq_{neg,pt} \overline{Int2}$.

The $Int1_k$ system is almost the same as SIS_k, except that TR is replaced by $TR1$, which differs from TR as follows. First, it has an extra input $in(x)_{Trans}$, obtaining the protocol input (x_0, x_1) intended for the transmitter. Second, it computes the third protocol message differently: the bit b_i is computed as in TR, but the bit b_{1-i} is computed using the hard-core predicate B, as $B(f^{-1}(z_{1-i})) \oplus x_{1-i}$. The $Int2_k$ system is defined to be the same as SIS_k except that it includes a random source automaton Src_{cval1} that chooses a random bit $cval1$, and TR is replaced by $TR2$, which is essentially the same as $TR1$ except that b_{1-i} is computed as $cval1 \oplus x_{1-i}$.

Our proofs that $\overline{RS} \leq_{neg,pt} \overline{Int1}$ and $\overline{Int2} \leq_{neg,pt} \overline{SIS}$ use simulation relations. To prove that $\overline{RS} \leq_{neg,pt} \overline{Int1}$, we show that, for every polynomial p, for every k, and for every $p(k)$-bounded environment Env_k for RS_k and $Int1_k$, there is a c-bounded simulation relation R_k from $RS_k \| Env_k$ to $Int1_k \| Env_k$, where c is a constant. Using Theorem 3, we obtain $\overline{RS} \leq_{neg,pt} \overline{Int1}$. Our proof of $\overline{Int2} \leq_{neg,pt} \overline{SIS}$ is similar, but with an interesting aspect. Namely, we show that computing the bit b_{1-i} as a random bit is equivalent to computing it as the XOR of a random bit and the input bit x_i.

Our proof that $\overline{Int1} \leq_{neg,pt} \overline{Int2}$ uses a computational argument, based on our definition of a hard-core predicate. The only difference between $\overline{Int1}$ and $\overline{Int2}$ is that a use of $B(f^{-1}(z_{1-i}))$ in $\overline{Int1}$ is replaced by a random bit in $\overline{Int2}$. This is precisely the difference between the $\overline{SR}$ and $\overline{SHR}$ systems discussed in Section 5. In order to exploit the fact that $\overline{SR} \leq_{neg,pt} \overline{SHR}$, we build an interface task-PIOA family $\overline{Ifc}$ which represents the common parts of $\overline{Int1}$ and $\overline{Int2}$. Then, we prove: (i) $\overline{Int1} \leq_{neg,pt} \overline{SH} \| \overline{Ifc} \| \overline{Adv}$ and $\overline{SHR} \| \overline{Ifc} \| \overline{Adv} \leq_{neg,pt} \overline{Int2}$ by exhibiting simple, constant-bounded simulation relations between these systems, and (ii) $\overline{SH} \| \overline{Ifc} \| \overline{Adv} \leq_{neg,pt} \overline{SHR} \| \overline{Ifc} \| \overline{Adv}$ by using our definition, Definition 2, of hard-core predicates, the fact that both $\overline{Ifc}$ and $\overline{Adv}$ are polynomial-time-bounded, and the composition property of $\leq_{neg,pt}$ (Theorem 2). Finally, using transitivity of $\leq_{neg,pt}$ (Theorem 1), we have $\overline{RS} \leq_{neg,pt} \overline{SIS}$, as needed.

8 Conclusions

We have introduced *time-bounded task-PIOAs* and *task-PIOA families*, building on the task-PIOA framework of [CCK$^+$06a,CCK$^+$06b]. We have adapted basic machinery, such as composition and hiding operations, and implementation and simulation relations, to time-bounded task-PIOAs. We have demonstrated the use of this framework in formulating and proving computational security properties for an Oblivious Transfer protocol [EGL85,GMW87]. Our proofs are decomposed into several stages, each showing an implementation relationship, $\leq_{neg,pt}$, between two systems. Most of these implementations are proved using simulation relations to match corresponding events and probabilities in the two systems. Others are proved using computational arguments involving reduction to the security of cryptographic primitives. Traditional reduction arguments for cryptographic primitives are reformulated in terms of $\leq_{neg,pt}$.

Our framework supports separation of scheduling concerns into two pieces: high-level scheduling, which is controlled by an algorithmic entity (e.g., the adversary component), and low-level scheduling, which is resolved nondeterministically by a task schedule. This separation allows inessential ordering of events to remain as nondeterministic choices in our system models, which increases the generality and reduces the complexity of the models and proofs.

We believe that the model and techniques presented here provide a suitable basis for analyzing a wide range of cryptographic protocols, including those that depend inherently on computational assumptions and achieve only computational security. Future plans include applying our methods to analyze more complex protocols, including protocols that use other cryptographic primitives and protocols that work against more powerful adversaries. We plan to establish general security protocol composition theorems in the style of [Can01,PW01] within our framework. We would like to formulate general patterns of adversarial behavior, in the style of [PW01], within our framework, and use this formulation to obtain general results about the security of the resulting systems.

References

[BCT04] G. Barthe, J. Cerderquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In *Automated Reasoning: Second International Joint Conference (IJCAR)*, 2004.

[Bla05] B. Blanchet. A computationally sound mechanized prover for security protocols. Cryptology ePrint Archive, Report 2005/401, 2005. `http://eprint.iacr.org/`.

[BPW04] M. Backes, B. Pfitzmann, and M. Waidner. Secure asynchronous reactive systems. Cryptology ePrint Archive, Report 2004/082, 2004. `http://eprint.iacr.org/`.

[BR04] M. Bellare and P. Rogaway. The game-playing technique and its application to triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd Annual Conference on Foundations of Computer Science (FOCS)*, 2001. Full version available at http://eprint.iacr.org/2000/067.

[CCK$^+$06a] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Task-structured probabilistic I/O automata. In *Proceedings of the 8th International Workshop on Discrete Event Systems (WODES)*, Ann Arbor, Michigan, July 2006.

[CCK$^+$06b] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Task-structured probabilistic I/O automata. Technical Report MIT-CSAIL-TR-2006-XXX, CSAIL, MIT, Cambridge, MA, 2006.

[CCK$^+$06c] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Using task-structured probabilistic I/O automata to analyze an oblivious transfer protocol. Technical Report MIT-CSAIL-TR-2006-047, CSAIL, MIT, Cambridge, MA, June 2006.

[DY83] D. Dolev and A.C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *CACM*, 28(6):637–647, 1985.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 19th Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[Gol01] O. Goldreich. *Foundations of Cryptography*, volume I Basic Tools. Cambridge Universirty Press, 2001.

[Hal05] S. Halevi. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181, 2005. `http://eprint.iacr.org/`.

[LMMS98] P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS-5)*, pages 112–121, 1998.

[MMS03] P. Mateus, J.C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time calculus. In *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR)*, volume 2761 of *LNCS*, pages 327–349, 2003.

[PW00] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[PW01] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy*, pages 184–200, 2001.

[RMST04] A. Ramanathan, J.C. Mitchell, A. Scedrov, and V. Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In *Proceedings of Foundations of Sotware Science and Computation Structires (FOSSACS)*, volume 2987 of *LNCS*, pages 468–483, 2004.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, May 1995. Also, MIT/LCS/TR-676.

[Sho04] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

[SL95] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, August 1995.