Task-Structured Probabilistic I/O Automata

Ran Canetti[*,‡], Ling Cheung[†], Dilsun Kaynar[‡], Moses Liskov[§], Nancy Lynch[‡], Olivier Pereira[¶], Roberto Segala[||]

[*]IBM T.J. Watson Research Center, USA, [†]Radboud University of Nijmegen, the Netherlands, [‡]MIT CSAIL, USA,
[§]The College of William and Mary, USA, [¶]Université Catholique de Louvain, Belgium, [||]Università di Verona, Italy

***Abstract*— Modeling frameworks such as Probabilistic I/O Automata (PIOA) and Markov Decision Processes permit both probabilistic and nondeterministic choices. In order to use such frameworks to express claims about probabilities of events, one needs mechanisms for resolving the nondeterministic choices. For PIOAs, nondeterministic choices have traditionally been resolved by schedulers that have perfect information about the past execution. However, such schedulers are too powerful for certain settings, such as cryptographic protocol analysis, where information must sometimes be hidden.**

Here, we propose a new, less powerful nondeterminism-resolution mechanism for PIOAs, consisting of *tasks* and *local schedulers*. Tasks are equivalence classes of system actions that are scheduled by oblivious, global task sequences. Local schedulers resolve nondeterminism within system components, based on local information only. The resulting task-PIOA framework yields simple notions of external behavior and implementation, and supports simple compositionality results. We also define a new kind of simulation relation, and show it to be sound for proving implementation. We illustrate the potential of the task-PIOA framework by outlining its use in verifying an Oblivious Transfer protocol.

I. Introduction

The *Probabilistic I/O Automata (PIOA)* modeling framework [Seg95], [SL95] is a simple combination of I/O Automata [LT89] and Markov Decision Processes (MDP) [Put94]. As demonstrated in [LSS94], [SV99], [PSL00], PIOAs are well suited for modeling and analyzing distributed algorithms that use randomness as a computational primitive. In this setting, distributed processes use random choices to break symmetry, in solving problems such as choice coordination [R82] and consensus [B83], [AH90]. Each process is modeled as an automaton with random transitions, and an entire protocol is modeled as the parallel composition of process automata and automata representing communication channels.

This modeling paradigm combines nondeterministic and probabilistic choices in a natural way. Nondeterminism is used here for modeling uncertainties in the timing of events in highly unpredictable distributed environments. It is also used for modeling distributed algorithms at high levels of abstraction, leaving many details unspecified. This in turn facilitates algorithm verification, because results proved about nondeterministic algorithms apply automatically to an entire family of algorithms, obtained by resolving the nondeterministic choices in particular ways.

In order to formulate and prove probabilistic properties of distributed algorithms, one needs mechanisms for resolving the nondeterministic choices. In the randomized distributed setting, the most common mechanism is a *perfect-information* event scheduler, which has access to local state and history of all system components and has unlimited computation power. Thus, probabilistic properties of distributed algorithms are typically asserted with respect to worst-case, adversarial schedulers who can choose the next event based on complete knowledge of the past (e.g., [PSL00]).

One would expect that a similar modeling paradigm, including both probabilistic and nondeterministic choices, would be similarly useful for modeling *cryptographic protocols*. These are special kinds of distributed algorithms, designed to protect sensitive data when it is transmitted over unreliable channels. Their correctness typically relies on computational assumptions, which say that certain problems cannot be solved by an adversarial entity with bounded computation resources [Gol01]. However, a major problem with this extension is that the perfect-information scheduler mechanism used for distributed algorithms is too powerful for use in the cryptographic setting. A scheduler that could see all information about the past would, in particular, see "secret" information hidden in the states of non-corrupted protocol participants, and be able to "divulge" this information to corrupted participants, e.g., by encoding it in the order in which it schedules events.

In this paper, we present *task-PIOAs*, an adaptation of PIOAs, that has new, less powerful mechanisms for resolving nondeterminism. Task-PIOAs are suitable for modeling and analyzing cryptographic protocols; they may also be useful for other kinds of distributed systems in which the perfect information assumption is unrealistically strong.

Task-PIOAs: A *task-PIOA* is simply a PIOA augmented with a partition of non-input actions into equivalence classes called *tasks*. A task is typically a set of related actions, for example, all the actions of a cryptographic protocol that send a round 1 message. Tasks are units of scheduling; they are scheduled by simple oblivious, global *task schedule* sequences. We define notions of *external behavior* and *implementation* for task-PIOAs, based on the trace distribution semantics proposed by Segala [Seg95]. We define parallel composition in the obvious way and show that our implementation relation is compositional.

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).

We also define a new type of *simulation relation*, which incorporates tasks, and prove that it is sound for proving implementation relationships between task-PIOAs. This new relation differs from simulation relations studied earlier [SL95], [LSV03], in that it relates probability measures rather than states. In many cases, including our work on cryptographic protocols (see below), tasks alone suffice for resolving nondeterminism. However, for extra expressive power, we define a second mechanism, *local schedulers*, which can be used to resolve nondeterminism within system components, based on local information only. This mechanism is based on earlier work in [CLSV04].

Cryptographic protocols: In [CC$^+$06a], we applied the task-PIOA framework to analyze an Oblivious Transfer (OT) protocol of Goldreich, et al. [GMW87]. That analysis required defining extra structure for task-PIOAs, in order to express issues involving computational limitations. Thus, we defined notions such as *time-bounded task-PIOAs*, and *approximate implementation with respect to time-bounded environments*. Details are beyond the scope of this paper, but we outline our approach in Section IV.

Adversarial scheduling: The standard scheduling mechanism in the cryptographic community is an *adversarial scheduler*, namely, a resource-bounded algorithmic entity that determines the next move adaptively, based on its own view of the computation so far. This is weaker than the *perfect-information scheduler* used for distributed algorithms, which have access to local state and history of all components and have unlimited computation power. Our task schedule sequences are essentially *oblivious schedulers*, which fix the entire schedule of tasks, nondeterministically, in advance. This formulation does not directly capture the adaptivity of adversarial schedulers.

Our solution is to separate scheduling concerns into two parts. We model the adaptive adversarial scheduler as a system component, for example, a message delivery service that can eavesdrop on the communications and control the order of message delivery. Such a system component 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. On the other hand, basic scheduling choices are resolved by a task schedule sequence, chosen nondeterministically in advance. These tasks are equivalence classes of actions, independent of actual choices that are determined during the execution. We believe this separation is conceptually meaningful: The high-level adversarial scheduler is responsible for choices that are essential in security analysis, such as the ordering of message deliveries. The low-level schedule of tasks resolves inessential choices. For example, in the OT protocol, both the transmitter and receiver make random choices, but it is inconsequential which does so first.

Related work: The literature contains numerous models that combine nondeterministic and probabilistic choices (see [SdV04] for a survey). However, few tackle the issue of partial-information scheduling, as we do. Exceptions include [CH05], which models local-oblivious scheduling, and [dA99], which uses partitions on the state space to obtain partial-information schedules. The latter is essentially within the framework of *partially observable MDPs (POMDPs)*, originally studied in the context of reinforcement learning [KLA98]. All of these accounts neglect partial information aspects of (parameterized) actions, therefore are not suitable in a cryptographic setting. A version of local schedulers was introduced in [CLSV04].

Our general approach to cryptographic protocol verification was directly inspired by the Interactive Turing Machine (ITM) framework of [Can01]. There, participants in a protocol are modeled as ITMs and messages as bit strings written on input and output tapes. ITMs are purely probabilistic, and scheduling nondeterminism is resolved using predefined rules. In principle, this framework could be used to analyze cryptographic protocols rigorously, including computational complexity issues. However, complete analysis of protocols in terms of Turing machines is impractical, because it involves too many low-level machine details. Indeed, in the computational cryptography community, protocols are typically described using an informal high-level language, and proof sketches are given in terms of the informal protocol descriptions. We aim to provide a framework in which proofs in the ITM style can be carried out formally, at a high level of abstraction. Also, we aim to exploit the benefits of nondeterminism to a greater extent than the ITM approach.

Several other researchers have added features for computational cryptographic analysis to conventional abstract concurrency modeling frameworks such as process algebras and restricted forms of PIOAs [LMMS98], [PW00], [PW01], [MMS03]. These approaches again use less nondeterminism than we do: individual system components are purely probabilistic, and scheduling is determined by predefined rules. For example, in [LMMS98], a uniform distribution is imposed on the set of possible reductions for each term. In [MMS03], internal reductions are prioritized over external communications and several independence properties are assumed. In [PW01], scheduling is based on a distributed scheme wherein each system component schedules the next one, based on its own local information. None of the prior work separates high-level and low-level nondeterminism resolution, as we do.

Roadmap: Section II defines task-PIOAs, task schedules, composition, and implementation, and presents a compositionality result. Section III presents our simulation relation and its soundness theorem. Section IV summarizes our OT protocol case study. Section V discusses local schedulers, and concluding discussions follow in Section VI. Further details appear in [CC$^+$06b].

II. TASK-PIOAS

A. Basic PIOAs

We assume our reader is comfortable with basic notions of probability theory, such as σ-fields and (discrete) probability measures. A summary is provided in [CC$^+$06b].

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 and pairwise disjoint sets of actions, referred to as *input, output and internal actions*, respectively; and (iii) $D \subseteq$

$(Q \times (I \cup O \cup H) \times \mathsf{Disc}(Q))$ is a *transition relation*, where $\mathsf{Disc}(Q)$ 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 state $q \in Q$ and input action $a \in I$, a is enabled in q.
- *Transition determinism:* For every $q \in Q$ and $a \in A$, there is at most one $\mu \in \mathsf{Disc}(Q)$ such that $(q, a, \mu) \in D$. If there is exactly one such μ, it is denoted by $\mu_{q,a}$, and we write $\mathsf{tran}_{q,a}$ for the transition $(q, a, \mu_{q,a})$.

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, then 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 $q_{i+1} \in \mathsf{supp}(\mu)$, where $\mathsf{supp}(\mu)$ denotes the support of μ. We write $\mathsf{fstate}(\alpha)$ for q_0, and, if α is finite, we write $\mathsf{lstate}(\alpha)$ for its last state. We use $\mathsf{Frags}(\mathcal{P})$ (resp., $\mathsf{Frags}^*(\mathcal{P})$) to denote the set of all (resp., all finite) execution fragments of $\mathcal{P}$. An *execution* of $\mathcal{P}$ is an execution fragment beginning from the start state $\bar{q}$. $\mathsf{Execs}(\mathcal{P})$ (resp., $\mathsf{Execs}^*(\mathcal{P})$) denotes the set of all (resp., finite) executions of $\mathcal{P}$.

The *trace* of an execution fragment α, written $\mathsf{trace}(\alpha)$, is the restriction of α to the set of external actions of $\mathcal{P}$. The symbol $\leq$ denotes the prefix relation on sequences, which applies in particular to execution fragments and traces.

Nondeterministic choices in $\mathcal{P}$ are resolved using a *scheduler*, which is a function $\sigma : \mathsf{Frags}^*(\mathcal{P}) \longrightarrow \mathsf{SubDisc}(D)$ such that $(q, a, \mu) \in \mathsf{supp}(\sigma(\alpha))$ implies $q = \mathsf{lstate}(\alpha)$. Here $\mathsf{SubDisc}(D)$ denotes the set of discrete sub-probability measures on D—that is, the measure of the entire space D is required to be ≤ 1. Thus, σ decides (probabilistically) which transition (if any) to take after each finite execution fragment α. A scheduler σ and a finite execution fragment α *generate* a measure $\epsilon_{\sigma,\alpha}$ on the σ-field $\mathcal{F}_{\mathcal{P}}$ generated by cones of execution fragments, where each cone $C_{\alpha'}$ is the set of execution fragments that have α' as a prefix. The measure of a cone, $\epsilon_{\sigma,\alpha}(C_{\alpha'})$, is defined recursively, as:

1) 0, if $\alpha' \not\leq \alpha$ and $\alpha \not\leq \alpha'$.
2) 1, if $\alpha' \leq \alpha$.
3) $\epsilon_{\sigma,\alpha}(C_{\alpha''})\mu_{\sigma(\alpha'')}(a, q)$, if α' is of the form $\alpha'' \; a \; q$ and $\alpha \leq \alpha''$. Here, $\mu_{\sigma(\alpha'')}(a, q)$ is defined to be $\sigma(\alpha'')(\mathsf{tran}_{\mathsf{lstate}(\alpha''),a})\mu_{\mathsf{lstate}(\alpha''),a}(q)$, that is, the probability that $\sigma(\alpha'')$ chooses a transition labeled by a and that the new state is q.

Standard measure theoretic arguments ensure that $\epsilon_{\sigma,\alpha}$ is well-defined. We call the state $\mathsf{fstate}(\alpha)$ the *first state* of $\epsilon_{\sigma,\alpha}$ and denote it by $\mathsf{fstate}(\epsilon_{\sigma,\alpha})$. If α consists of the start state $\bar{q}$ only, we call $\epsilon_{\sigma,\alpha}$ a *probabilistic execution* of $\mathcal{P}$.

Let μ be a discrete probability measure over $\mathsf{Frags}^*(\mathcal{P})$. We denote by $\epsilon_{\sigma,\mu}$ the measure $\sum_\alpha \mu(\alpha)\epsilon_{\sigma,\alpha}$ and we say that $\epsilon_{\sigma,\mu}$ is *generated by* σ and μ. We call the measure $\epsilon_{\sigma,\mu}$ a *generalized probabilistic execution fragment* of $\mathcal{P}$. If every execution fragment in $\mathsf{supp}(\mu)$ consists of a single state, then we call $\epsilon_{\sigma,\mu}$ a *probabilistic execution fragment* of $\mathcal{P}$.

We note that the trace function is a measurable function from $\mathcal{F}_{\mathcal{P}}$ to the σ-field generated by cones of traces. Thus, given a probability measure ϵ on $\mathcal{F}_{\mathcal{P}}$, we define the *trace distribution* of ϵ, denoted $\mathsf{tdist}(\epsilon)$, to be the image measure of ϵ under trace. We denote by $\mathsf{tdists}(\mathcal{P})$ the set of trace distributions of (probabilistic executions of) $\mathcal{P}$.

Definition 2.1: 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 $(q_i, a, \mu_i) \in D_i$, otherwise $\mu_i = \delta(q_i)$. This definition can be extended to any finite number of PIOAs rather than just two.

B. Task-PIOAs

We now augment the PIOA framework with task partitions, our main mechanism for resolving nondeterminism.

Definition 2.2: A *task-PIOA* is a pair $\mathcal{T} = (\mathcal{P}, R)$ where (i) $\mathcal{P} = (Q, \bar{q}, I, O, H, D)$ is a PIOA (satisfying transition determinism) and (ii) 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 *enabled* in a state q if some $a \in T$ is enabled in q.

Unless otherwise stated, technical notions for task-PIOAs are inherited from those for PIOAs. Exceptions include the notions of probabilistic executions and trace distributions. For now, we impose the following action-determinism assumption, which implies that tasks alone are enough to resolve all nondeterministic choices. We will remove this assumption when we introduce local schedulers, in Section V.

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

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

Definition 2.3: Let $\mathcal{T} = (\mathcal{P}, R)$ be an action-deterministic task-PIOA where $\mathcal{P} = (Q, \bar{q}, I, O, H, D)$. Given $\mu \in \mathsf{Disc}(\mathsf{Frags}^*(\mathcal{P}))$ and a task schedule ρ, $\mathsf{apply}(\mu, \rho)$ is the probability measure on $\mathsf{Frags}(\mathcal{P})$ defined recursively by:

1) $\mathsf{apply}(\mu, \lambda) := \mu$. ($\lambda$ denotes the empty sequence.)
2) For $T \in R$, $\mathsf{apply}(\mu, T)$ is defined as follows. For every $\alpha \in \mathsf{Frags}^*(\mathcal{P})$, $\mathsf{apply}(\mu, T)(\alpha) := p_1 + p_2$, where:
   - $p_1 = \mu(\alpha')\eta(q)$ if α is of the form $\alpha' \; a \; q$, where $a \in T$ and $(\mathsf{lstate}(\alpha'), a, \eta) \in D$; $p_1 = 0$ otherwise.
   - $p_2 = \mu(\alpha)$ if T is not enabled in $\mathsf{lstate}(\alpha)$; $p_2 = 0$ otherwise.
3) For ρ of the form $\rho'\ T$, $T \in R$, $\mathsf{apply}(\mu, \rho) := \mathsf{apply}(\mathsf{apply}(\mu, \rho'), T)$.
4) For ρ infinite, $\mathsf{apply}(\mu, \rho) := \lim_{i \to \infty}(\mathsf{apply}(\mu, \rho_i))$, where ρ_i denotes the length-i prefix of ρ.

In Case (2) above, p_1 represents the probability that α is executed when applying task T at the end of α'. Because of

transition-determinism and action-determinism, the transition $(\mathsf{lstate}(\alpha'), a, \eta)$ is unique, and so p_1 is well-defined. The term p_2 represents the original probability $\mu(\alpha)$, which is relevant if T is not enabled after α. It is routine to check that the limit in Case (4) is well-defined. The other two cases are straightforward.

Next, we show that $\mathsf{apply}(\mu, \rho)$ is a generalized probabilistic execution fragment generated by μ and a scheduler for $\mathcal{P}$, in the usual sense. Thus, a task schedule for a task-PIOA is a special case of a scheduler for the underlying PIOA.

Theorem 2.4: Let $\mathcal{T} = (\mathcal{P}, R)$ be an action-deterministic task-PIOA. For each measure μ on $\mathsf{Frags}^*(\mathcal{P})$ and task schedule ρ, there is scheduler σ for $\mathcal{P}$ such that $\mathsf{apply}(\mu, \rho)$ is the generalized probabilistic execution fragment $\epsilon_{\sigma,\mu}$.

Any such $\mathsf{apply}(\mu, \rho)$ is said to be a *generalized probabilistic execution fragment* of $\mathcal{T}$. *Probabilistic execution fragments* and *probabilistic executions* are then defined by making the same restrictions as for basic PIOAs. We write $\mathsf{tdist}(\mu, \rho)$ as shorthand for $\mathsf{tdist}(\mathsf{apply}(\mu, \rho))$, the trace distribution obtained by applying task schedule ρ starting from the measure μ on execution fragments. We write $\mathsf{tdist}(\rho)$ for $\mathsf{tdist}(\mathsf{apply}(\delta(\bar{q}), \rho))$ the trace distribution obtained by applying ρ from the unique start state. (Recall that the Dirac measure for an element x, $\delta(x)$, is the discrete probability measure that assigns probability 1 to $\{x\}$.) A *trace distribution* of $\mathcal{T}$ is any $\mathsf{tdist}(\rho)$. We use $\mathsf{tdists}(\mathcal{T})$ to denote the set $\{\mathsf{tdist}(\rho) : \rho \text{ is a task schedule for } \mathcal{T}\}$. Finally, we define composition of task-PIOAs:

Definition 2.5: Two task-PIOAs $\mathcal{T}_i = (\mathcal{P}_i, R_i), i \in \{1, 2\}$, are said to be *compatible* provided the underlying PIOAs are compatible. In this case, we define their *composition* $\mathcal{T}_1 \| \mathcal{T}_2$ to be the task-PIOA $(\mathcal{P}_1 \| \mathcal{P}_2, R_1 \cup R_2)$.

It is easy to see that $\mathcal{T}_1 \| \mathcal{T}_2$ is in fact a task-PIOA. In particular, since compatibility ensures disjoint sets of locally-controlled actions, $R_1 \cup R_2$ is an equivalence relation on the locally-controlled actions of $\mathcal{T}_1 \| \mathcal{T}_2$. It is also easy to see that action determinism is preserved under composition. Note that, when two task-PIOAs are composed, no new mechanisms are required to schedule actions of the two components—the tasks alone are enough.

C. Implementation

We now define the notion of external behavior for a task-PIOA and the induced implementation relation between task-PIOAs. Unlike previous definitions of external behavior, the one we use here is not simply a set of trace distributions. Rather, it is a mapping that specifies, for every possible "environment" $\mathcal{E}$ for the given task-PIOA $\mathcal{T}$, the set of trace distributions that can arise when $\mathcal{T}$ is composed with $\mathcal{E}$.

Definition 2.6: Let $\mathcal{T}$ be any task-PIOA and $\mathcal{E}$ be an action-deterministic task-PIOA. We say that $\mathcal{E}$ is an *environment* for $\mathcal{T}$ if (i) $\mathcal{E}$ is compatible with $\mathcal{T}$ and (ii) the composition $\mathcal{T} \| \mathcal{E}$ is closed. Note that $\mathcal{E}$ may have output actions that are not in the signature of $\mathcal{T}$.

Definition 2.7: The *external behavior* of $\mathcal{T}$, denoted by $\mathsf{extbeh}(\mathcal{T})$, is the total function that maps each environment $\mathcal{E}$ to the set of trace distributions $\mathsf{tdists}(\mathcal{T} \| \mathcal{E})$.

Thus, for each environment, we consider the set of trace distributions that arise from all task schedules. Note that these traces may include new output actions of $\mathcal{E}$, in addition to the external actions already present in $\mathcal{T}$. Our definition of implementation is influenced by common notions in the security literature (e.g., [LMMS98], [Can01], [PW01]). Namely, the implementation must "look like" the specification from the perspective of every possible environment. The precise notion of implementation is formulated in terms of inclusion of sets of trace distributions *for each environment automaton*. An advantage of this style of definition is that it yields simple compositionality results (Theorem 2.9).

Definition 2.8: Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be *comparable* action-deterministic task-PIOAs, that is, $I_1 = I_2$ and $O_1 = O_2$. We say that $\mathcal{T}_1$ *implements* $\mathcal{T}_2$, written $\mathcal{T}_1 \leq_0 \mathcal{T}_2$, if $\mathsf{extbeh}(\mathcal{T}_1)(\mathcal{E}) \subseteq \mathsf{extbeh}(\mathcal{T}_2)(\mathcal{E})$ for every environment $\mathcal{E}$ for both $\mathcal{T}_1$ and $\mathcal{T}_2$. In other words, we require $\mathsf{tdists}(\mathcal{T}_1 || \mathcal{E}) \subseteq \mathsf{tdists}(\mathcal{T}_2 || \mathcal{E})$ for every $\mathcal{E}$.

The subscript 0 in the relation symbol $\leq_0$ refers to the requirement that every trace distribution in $\mathsf{tdists}(\mathcal{T}_1 || \mathcal{E})$ must have an identical match in $\mathsf{tdists}(\mathcal{T}_2 || \mathcal{E})$. For security analysis, we also define another relation $\leq_{neg,pt}$, which allows "negligible" discrepancies between matching trace distributions [CC+06a].

D. Compositionality

Because external behavior and implementation are defined in terms of mappings from environments to sets of trace distributions, a compositionality result for $\leq_0$ follows easily:

Theorem 2.9: Let $\mathcal{T}_1$, $\mathcal{T}_2$ be comparable action-deterministic task-PIOAs such that $\mathcal{T}_1 \leq_0 \mathcal{T}_2$, and let $\mathcal{T}_3$ be an action-deterministic task-PIOA compatible with each of $\mathcal{T}_1$ and $\mathcal{T}_2$. Then $\mathcal{T}_1 \| \mathcal{T}_3 \leq_0 \mathcal{T}_2 \| \mathcal{T}_3$.

Proof. Let $\mathcal{T}_4 = (\mathcal{P}_4, R_4)$ be any environment (action-deterministic) task-PIOA for both $\mathcal{T}_1 \| \mathcal{T}_3$ and $\mathcal{T}_2 \| \mathcal{T}_3$. Fix any task schedule ρ_1 for $(\mathcal{T}_1 \| \mathcal{T}_3) \| \mathcal{T}_4$. Let τ be the trace distribution of $(\mathcal{T}_1 \| \mathcal{T}_3) \| \mathcal{T}_4$ generated by ρ_1. It suffices to show that τ is also generated by some task schedule ρ_2 for $(\mathcal{T}_2 \| \mathcal{T}_3) \| \mathcal{T}_4$. Note that ρ_1 is also a task schedule for $\mathcal{T}_1 \| (\mathcal{T}_3 \| \mathcal{T}_4)$, and that ρ_1 generates the same trace distribution τ in the composed task-PIOA $\mathcal{T}_1 \| (\mathcal{T}_3 \| \mathcal{T}_4)$.

Now, $\mathcal{T}_3 \| \mathcal{T}_4$ is an (action-deterministic) environment task-PIOA for each of $\mathcal{T}_1$ and $\mathcal{T}_2$. Since, by assumption, $\mathcal{T}_1 \leq_0 \mathcal{T}_2$, we infer the existence of a task schedule ρ_2 for $\mathcal{T}_2 \| (\mathcal{T}_3 \| \mathcal{T}_4)$ such that ρ_2 generates trace distribution τ in the task-PIOA $\mathcal{T}_2 \| (\mathcal{T}_3 \| \mathcal{T}_4)$. Since ρ_2 is also a task schedule for $(\mathcal{T}_2 \| \mathcal{T}_3) \| \mathcal{T}_4$ and ρ_2 generates τ, this suffices. □

III. Simulation Relations

We define a new simulation relation notion for closed, action-deterministic task-PIOAs, and show that it is sound for proving $\leq_0$. Our definition is based on three operations involving probability measures: flattening, lifting, and expansion. These have been previously defined, e.g., in [LSV03].

A. Flattening, lifting, and expansion

The *flattening* operation takes a discrete probability measure over probability measures and "flattens" it into a single probability measure. Formally, let η be a discrete probability measure on $\mathsf{Disc}(X)$. Then the flattening of η, denoted by $\mathsf{flatten}(\eta)$, is the discrete probability measure on X defined by $\mathsf{flatten}(\eta) = \sum_{\mu \in \mathsf{Disc}(X)} \eta(\mu)\mu$.

The *lifting* operation takes a relation R between two domains X and Y and "lifts" it to a relation between discrete

measures over X and Y. Informally speaking, a measure μ_1 on X is related to a measure μ_2 on Y if μ_2 can be obtained by "redistributing" the probability masses assigned by μ_1, in such a way that the relation R is respected. Formally, the *lifting* of R, denoted by $\mathcal{L}(R)$, is a relation from $\mathsf{Disc}(X)$ to $\mathsf{Disc}(Y)$ defined by: $\mu_1\ \mathcal{L}(R)\ \mu_2$ iff there exists a *weighting function* $w : X \times Y \to \mathbf{R}^{\geq 0}$ such that

1) For each $x \in X$ and $y \in Y$, $w(x,y) > 0$ implies $x\ R\ y$.
2) For each $x \in X$, $\sum_y w(x,y) = \mu_1(x)$.
3) For each $y \in Y$, $\sum_x w(x,y) = \mu_2(y)$.

Finally, the *expansion* operation takes a relation between discrete measures on two domains and returns a relation of the same kind that relates two measures whenever they can be decomposed into two $\mathcal{L}(R)$-related measures. Formally, let R be a relation from $\mathsf{Disc}(X)$ to $\mathsf{Disc}(Y)$. The *expansion* of R, written $\mathcal{E}(R)$, is a relation from $\mathsf{Disc}(X)$ to $\mathsf{Disc}(Y)$ defined by: $\mu_1\ \mathcal{E}(R)\ \mu_2$ iff there exist two discrete measures η_1 and η_2 on $\mathsf{Disc}(X)$ and $\mathsf{Disc}(Y)$, respectively, such that $\mu_1 = \mathsf{flatten}(\eta_1)$, $\mu_2 = \mathsf{flatten}(\eta_2)$, and $\eta_1\ \mathcal{L}(R)\ \eta_2$.

We use expansions directly in our definition of simulation. Informally, $\mu_1\ R\ \mu_2$ means that it is possible to simulate from μ_2 anything that can happen from μ_1. Furthermore, $\mu_1'\ \mathcal{E}(R)\ \mu_2'$ means that we can decompose μ_1' and μ_2' into pieces that can simulate each other, and so we can say that it is also possible to simulate from μ_2' anything that can happen from μ_1'. This intuition is at the base of the proof of our soundness result (cf. Theorem 3.5).

B. Simulation relation definition

We need two more auxiliary definitions. The first expresses consistency between a probability measure over finite executions and a task schedule: informally, a measure ϵ over finite executions is said to be consistent with a task schedule ρ if it assigns non-zero probability only to those executions that are possible under the task schedule ρ. We use this condition in order to avoid useless proof obligations in our definition of simulation relation.

Definition 3.1: Let $\mathcal{T} = (\mathcal{P}, R)$ be a closed, action-deterministic task-PIOA, ϵ a discrete probability measure over finite executions of $\mathcal{P}$, and ρ a finite task schedule for $\mathcal{T}$. Then ϵ is *consistent with* ρ provided that $\mathsf{supp}(\epsilon) \subseteq \mathsf{supp}(\mathsf{apply}(\delta(\bar{q}), \rho))$, where $\bar{q}$ is the start state of $\mathcal{P}$.

For the second definition, suppose we have a mapping c that, given a finite task schedule ρ and a task T of a task-PIOA $\mathcal{T}_1$, yields a task schedule of another task-PIOA $\mathcal{T}_2$. The idea is that $\mathsf{c}(\rho, T)$ describes how $\mathcal{T}_2$ matches task T, given that it has already matched the task schedule ρ. Using c, we define a new function $\mathsf{full}(\mathsf{c})$ that, given a task schedule ρ, iterates c on all the elements of ρ, thus producing a "full" task schedule of $\mathcal{T}_2$ that matches all of ρ.

Definition 3.2: Let $\mathcal{T}_1 = (\mathcal{P}_1, R_1)$ and $\mathcal{T}_2 = (\mathcal{P}_2, R_2)$ be two task-PIOAs, and let $\mathsf{c} : ({R_1}^* \times R_1) \to {R_2}^*$ be a function that assigns a finite task schedule of $\mathcal{T}_2$ to each finite task schedule of $\mathcal{T}_1$ and task of $\mathcal{T}_1$. Define $\mathsf{full}(\mathsf{c}) : {R_1}^* \to {R_2}^*$ recursively as follows: $\mathsf{full}(\mathsf{c})(\lambda) := \lambda$, and $\mathsf{full}(\mathsf{c})(\rho T) := \mathsf{full}(\mathsf{c})(\rho) \frown \mathsf{c}(\rho, T)$ (the concatenation of $\mathsf{full}(\mathsf{c})(\rho)$ and $\mathsf{c}(\rho, T)$).

We can now define our new notion of simulation for task-PIOAs and establish its soundness with respect to the $\leq_0$ relation. Note that our simulation relations do not just relate states to states, but rather, probability measures on executions to probability measures on executions.[1] The use of measures on executions here rather than just executions is motivated by certain cases that arise in our OT protocol proof, e.g., cases where related random choices are made at different points in the low-level and high-level models (see Section III-D).

Definition 3.3: Let $\mathcal{T}_1 = (\mathcal{P}_1, R_1)$ and $\mathcal{T}_2 = (\mathcal{P}_2, R_2)$ be two comparable closed action-deterministic task-PIOAs. Let R be a relation from $\mathsf{Disc}(\mathsf{Execs}^*(\mathcal{P}_1))$ to $\mathsf{Disc}(\mathsf{Execs}^*(\mathcal{P}_2))$, such that, if $\epsilon_1\ R\ \epsilon_2$, then $\mathsf{tdist}(\epsilon_1) = \mathsf{tdist}(\epsilon_2)$. Then R is a simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$ if there exists $\mathsf{c} : ({R_1}^* \times R_1) \to {R_2}^*$ such that the following properties hold:

1) *Start condition:* $\delta(\bar{q}_1)\ R\ \delta(\bar{q}_2)$.
2) *Step condition:* If $\epsilon_1\ R\ \epsilon_2$, $\rho_1 \in {R_1}^*$, ϵ_1 is consistent with ρ_1, ϵ_2 is consistent with $\mathsf{full}(\mathsf{c})(\rho_1)$, and $T \in R_1$, then $\epsilon_1'\ \mathcal{E}(R)\ \epsilon_2'$ where $\epsilon_1' = \mathsf{apply}(\epsilon_1, T)$ and $\epsilon_2' = \mathsf{apply}(\epsilon_2, \mathsf{c}(\rho_1, T))$.

C. Soundness

Lemma 3.4: Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable closed action-deterministic task-PIOAs, R a simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$. Let ϵ_1 and ϵ_2 be discrete probability measures over finite executions of $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, such that $\epsilon_1\ \mathcal{E}(R)\ \epsilon_2$. Then $\mathsf{tdist}(\epsilon_1) = \mathsf{tdist}(\epsilon_2)$.

The following theorem says that, for closed task-PIOAs, the existence of a simulation relation implies inclusion of sets of trace distributions. Our main soundness result for (not necessarily closed) task-PIOAs then follows as a corollary.

Theorem 3.5: Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable closed action-deterministic task-PIOAs. If there exists a simulation relation from $\mathcal{T}_1$ to $\mathcal{T}_2$, then $\mathsf{tdists}(\mathcal{T}_1) \subseteq \mathsf{tdists}(\mathcal{T}_2)$.

Proof. Let R be the assumed simulation relation from $\mathcal{T}_1$ to $\mathcal{T}_2$. Let ϵ_1 be the probabilistic execution of $\mathcal{T}_1$ generated by $\bar{q}_1$ and a (finite or infinite) task schedule, $T_1, T_2, \cdots$. For each $i > 0$, define ρ_i to be $\mathsf{c}(T_1 \cdots T_{i-1}, T_i)$. Let ϵ_2 be the probabilistic execution generated by $\bar{q}_2$ and $\rho_1\rho_2\cdots$. We claim that $\mathsf{tdist}(\epsilon_1) = \mathsf{tdist}(\epsilon_2)$, which suffices.

For each $j \geq 0$, let $\epsilon_{1,j} = \mathsf{apply}(\bar{q}_1, T_1 \cdots T_j)$, and $\epsilon_{2,j} = \mathsf{apply}(\bar{q}_2, \rho_1 \cdots \rho_j)$. Then for each $j \geq 0$, $\epsilon_{1,j} \leq \epsilon_{1,j+1}$ and $\epsilon_{2,j} \leq \epsilon_{2,j+1}$; moreover, $\lim_{j\to\infty} \epsilon_{1,j} = \epsilon_1$ and $\lim_{j\to\infty} \epsilon_{2,j} = \epsilon_2$. Also, for every $j \geq 0$, $\mathsf{apply}(\epsilon_{1,j}, T_{j+1}) = \epsilon_{1,j+1}$ and $\mathsf{apply}(\epsilon_{2,j}, \rho_{j+1}) = \epsilon_{2,j+1}$. Observe that $\epsilon_{1,0} = \delta(\bar{q}_1)$ and $\epsilon_{2,0} = \delta(\bar{q}_2)$. The start condition for a simulation relation and a trivial expansion imply that $\epsilon_{1,0}\ \mathcal{E}(R)\ \epsilon_{2,0}$. Then by induction, using the definition of a simulation relation in proving the inductive step (this uses a series of lemmas; see [CC+06b] for details), we show that, for each $j \geq 0$, $\epsilon_{1,j}\ \mathcal{E}(R)\ \epsilon_{2,j}$. Then, by Lemma 3.4, for each $j \geq 0$, $\mathsf{tdist}(\epsilon_{1,j}) = \mathsf{tdist}(\epsilon_{2,j})$. Now, $\mathsf{tdist}(\epsilon_1) = \lim_{j\to\infty} \mathsf{tdist}(\epsilon_{1,j})$, and $\mathsf{tdist}(\epsilon_2) = \lim_{j\to\infty} \mathsf{tdist}(\epsilon_{2,j})$. Since for each $j \geq 0$, $\mathsf{tdist}(\epsilon_{1,j}) = \mathsf{tdist}(\epsilon_{2,j})$, we conclude that $\mathsf{tdist}(\epsilon_1) = \mathsf{tdist}(\epsilon_2)$, as needed. □

Corollary 3.6: Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two comparable action-deterministic task-PIOAs. Suppose that, for every environment $\mathcal{E}$ for both $\mathcal{T}_1$ and $\mathcal{T}_2$, there exists a simulation relation R from $\mathcal{T}_1 \| \mathcal{E}$ to $\mathcal{T}_2 \| \mathcal{E}$. Then $\mathcal{T}_1 \leq_0 \mathcal{T}_2$.

[1] It would be nice to simplify this definition so that it involves measures on states instead of measures on executions, but we don't yet know how to do this.

D. Example: $Trapdoor$ vs. $Rand$

The following example, from our OT proof, was a key motivation for generalizing prior notions of simulation relations. We consider two closed task-PIOAs, $Trapdoor$ and $Rand$. $Rand$ chooses a number randomly and outputs it. $Trapdoor$, on the other hand, first chooses a random number, then applies a known permutation f to the chosen number, and then outputs the result. (The name $Trapdoor$ refers to the type of permutation f that is used in the OT protocol.)

More precisely, $Rand$ has output actions $report(k)$, $k \in [n] = \{1, \ldots, n\}$ and internal action $choose$. It has tasks $Report = \{report(k) : k \in [n]\}$, and $Choose = \{choose\}$. Its state contains one variable $zval$, which assumes values in $[n] \cup \{\perp\}$, initially $\perp$. The $choose$ action is enabled when $zval = \perp$, and has the effect of setting $zval$ to a number in $[n]$, chosen uniformly at random. The $report(k)$ action is enabled when $zval = k$, and has no effect on the state (so it may happen repeatedly). See Figure 1.

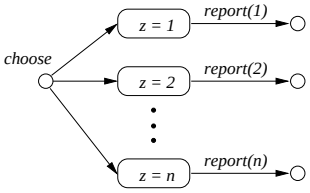

Fig. 1. Task-PIOA $Rand$

$Trapdoor$ has the same actions as $Rand$, plus internal action $compute$. It has the same tasks as $Rand$, plus the task $Compute = \{compute\}$. $Trapdoor$'s state contains two variables, y and z, each of which takes on values in $[n] \cup \{\perp\}$, initially $\perp$. The $choose$ action is enabled when $y = \perp$, and sets y to a number in $[n]$, chosen uniformly at random. The $compute$ action is enabled when $y \neq \perp$ and $z = \perp$, and sets $z := f(y)$. The $report(k)$ action behaves exactly as in $Rand$. See Figure 2.

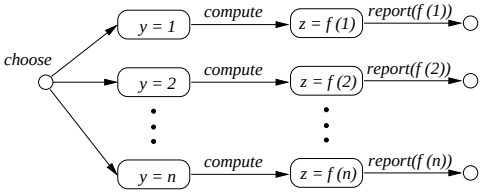

Fig. 2. Task-PIOA $Trapdoor$

We wanted to use a simulation relation to prove that $\mathsf{tdists}(Trapdoor) \subseteq \mathsf{tdists}(Random)$. In doing so, we decided that the steps that define z should correspond in the two automata, which meant that the $choose$ steps of $Trapdoor$, which define y, should map to no steps of $Rand$. Then, between the $choose$ and $compute$ in $Trapdoor$, a randomly-chosen value would appear in the y component of $Trapdoor$'s state, but no such value would appear in the corresponding state of $Rand$. Therefore, the simulation relation would have to relate a probability measure on states of $Trapdoor$ to a single state of $Rand$.

We were able to express this correspondence using a simulation relation of our new kind: If ϵ_1 and ϵ_2 are discrete measures over finite execution fragments of $Trapdoor$ and $Rand$, respectively, then we defined $(\epsilon_1, \epsilon_2) \in R$ exactly if the following conditions hold: (i) For every $s \in \mathsf{supp}(\mathsf{lstate}(\epsilon_1))$ and $u \in \mathsf{supp}(\mathsf{lstate}(\epsilon_2))$, $s.z = u.z$. (ii) For every $u \in \mathsf{supp}(\mathsf{lstate}(\epsilon_2))$, if $u.z = \perp$ then either $\mathsf{lstate}(\epsilon_1).y$ is everywhere undefined or else it is the uniform distribution on $[n]$. The task correspondence mapping c is defined by: $c(\rho, Choose) = \lambda$, $c(\rho, Compute) = Choose$, $c(\rho, Report) = Report$.

IV. Application to Security Protocols

In [CC$^+$06a], we use the task-PIOAs of this paper to model and analyze the Oblivious Transfer (OT) protocol of Goldreich et al. [GMW87]. In the OT problem, two input bits (x_0, x_1) are submitted to a Transmitter $Trans$ and a single input bit i to a Receiver Rec. After engaging in an OT protocol, Rec should output only the single bit x_i. Rec should not learn the other bit x_{1-i}, and $Trans$ should not learn i; moreover, an eavesdropping adversary should not, by observing the protocol messages, be able to learn anything about the inputs or the progress of the protocol. OT has been shown to be "complete" for multi-party secure computation, in the sense that, using OT as the only cryptographic primitive, one can construct protocols for securely realizing any functionality.

The protocol of [GMW87] uses trap-door permutations (and hard-core predicates) as an underlying cryptographic primitive. It uses three rounds of communication: First, $Trans$ chooses a random trap-door permutation f and sends it to Rec. Second, Rec chooses two random numbers (y_0, y_1) and sends (z_0, z_1) to $Trans$, where z_i for the input index i is $f(y_i)$ and $z_{1-i} = y_{1-i}$. Third, $Trans$ applies the same transformation to each of z_0 and z_1 and sends the results back as (b_0, b_1) Finally, Rec decodes and outputs the correct bit. The protocol uses cryptographic primitives and computational hardness in an essential way. Its security is inherently only computational, so its analysis requires modeling computational assumptions.

Our analysis follows the *trusted party* paradigm of [GMW87], with a formalization that is close in spirit to [PW00], [Can01]. We first define task-PIOAs representing the *real system (RS)* (the protocol) and the *ideal system (IS)* (the requirements). In RS, typical tasks include "choose random (y_0, y_1)", "send round 1 message", and "deliver round 1 message", as well as arbitrary tasks of incompletely-specified environment and adversary automata. Note that these tasks do not specify exactly what transition occurs; e.g., the send task does not specify the message contents—these are chosen by $Trans$, based on its own internal state.

Then we prove that RS implements IS. The proof consists of four cases, depending on which parties are corrupted.[2]. In the two cases where $Trans$ is corrupted, we can show that RS implements IS unconditionally, using $\leq_0$. In the cases where $Trans$ is not corrupted, we can show implementation only in a "computational" sense, namely, (i) for resource-bounded adversaries, (ii) up to negligible differences, and

[2]Actually, in [CC$^+$06a], we prove only one case—when only R is corrupted. We prove all four cases in [CC$^+$05], but using a less general definition of task-PIOAs than the one used here and in [CC$^+$06a], and with non-branching adversaries.

(iii) under computational hardness assumptions. Modeling these aspects requires additions to the task-PIOA framework of this paper, namely, defining a *time-bounded* version of task-PIOAs, and defining a variation, $\leq_{neg,pt}$, on the $\leq_0$ relation, which describes approximate implementation with respect to polynomial-time-bounded environments. Similar relations were defined in [LMMS98], [PW01]. Our simulation relations are also sound with respect to $\leq_{neg,pt}$. We also provide models for the cryptographic primitives (trap-door functions and hard-core predicates). Part of the specification for such primitives is that their behavior should look "approximately random" to outside observers; we formalize this in terms of $\leq_{neg,pt}$.

The correctness proofs proceed by levels of abstraction, relating each pair of models at successive levels using $\leq_{neg,pt}$. In the case where only Rec is corrupted, all but one of the relationships between levels are proved using simulation relations as defined in this paper (and so, they guarantee $\leq_0$). The only exception relates a level in which the cryptographic primitive is used, with a higher level in which the use of the primitive is replaced by a random choice. Showing this correspondence relies on our $\leq_{neg,pt}$-based definitions of the cryptographic primitive, and on composition results for time-bounded task-PIOAs. Since this type of reasoning is isolated to one correspondence, the methods of this paper in fact suffice to accomplish most of the work of verifying OT.

Each of our system models, at each level, includes an explicit adversary component automaton, which acts as a message delivery service that can eavesdrop on communications and control the order of message delivery. The behavior of this adversary is arbitrary, subject to general constraints on its capabilities. In our models, the adversary is the same at all levels, so our simulation relations relate the adversary states at consecutive levels directly, using the identity function. This treatment allows us to consider arbitrary adversaries without examining their structure in detail (they can do anything, but must do the same thing at all levels).

Certain patterns that arise in our simulation relation proofs led us to extend earlier definitions of simulation relations [SL95], [LSV03], by adding the expansion capability and by corresponding measures to measures: (i) We often correspond random choices at two levels of abstraction—for instance, when the adversary makes a random choice, from the same state, at both levels. We would like our simulation relation to relate the individual outcomes of the choices at the two levels, matching up the states in which the same result is obtained. Modeling this correspondence uses the expansion feature. (ii) The $Trapdoor$ vs. $Rand$ example described in Section III occurs in our OT proof. Here, the low-level system chooses a random y and then computes $z = f(y)$ using a trap-door permutation f. The higher level system simply chooses the value of z randomly, without using value y or permutation f. This correspondence relates measures to measures and uses expansion. (iii) In another case, a lower-level system chooses a random value y and then computes a new value by XOR'ing y with an input value. The higher level system just chooses a random value. However, XOR'ing any value with a random value yields the same result as just choosing a random value. This correspondence relates measures to measures and uses expansion.

V. Local Schedulers

With the action-determinism assumption, our task mechanism is enough to resolve all nondeterminism. However, action determinism limits expressive power. Now we remove this assumption and add a second mechanism for resolving the resulting additional nondeterminism, namely, a *local scheduler* for each component task-PIOA. A local scheduler for a given component can be used to resolve nondeterministic choices among actions in the same task, using only information about the past history of that component. Here, we define one type of local scheduler, which uses only the current state, and indicate how our results for the action-deterministic case carry over to this setting.

Our notion of local scheduler is simply a "sub-automaton": We say that task-PIOA $\mathcal{T}' = (\mathcal{P}', R')$ is a *sub-task-PIOA* of task-PIOA $\mathcal{T} = (\mathcal{P}, R)$ provided that all components are identical except that $D' \subseteq D$, where D and D' are the sets of discrete transitions of $\mathcal{P}$ and $\mathcal{P}'$, respectively. Thus, the only difference is that $\mathcal{T}'$ may have a smaller set of transitions. A *local scheduler* for a task-PIOA $\mathcal{T}$ is any action-deterministic sub-task-PIOA of $\mathcal{T}$. A *probabilistic system* is a pair $\mathcal{M} = (\mathcal{T}, \mathcal{S})$, where $\mathcal{T}$ is a task-PIOA and $\mathcal{S}$ is a set of local schedulers for $\mathcal{T}$. A *probabilistic execution* of a probabilistic system $\mathcal{M} = (\mathcal{T}, \mathcal{S})$ is defined to be any probabilistic execution of any task-PIOA $S \in \mathcal{S}$.

If $\mathcal{M}_1 = (\mathcal{T}_1, \mathcal{S}_1)$ and $\mathcal{M}_2 = (\mathcal{T}_2, \mathcal{S}_2)$ are two probabilistic systems, and $\mathcal{T}_1$ and $\mathcal{T}_2$ are compatible, then their *composition* $\mathcal{M}_1 \| \mathcal{M}_2$ is the probabilistic system $(\mathcal{T}_1 \| \mathcal{T}_2, \mathcal{S})$, where $\mathcal{S}$ is the set of local schedulers for $\mathcal{T}_1 \| \mathcal{T}_2$ of the form $S_1 \| S_2$, for some $S_1 \in \mathcal{S}_1$ and $S_2 \in \mathcal{S}_2$.

If $\mathcal{M} = (\mathcal{T}, \mathcal{S})$ is a probabilistic system, then an *environment* for $\mathcal{M}$ is any environment (action-deterministic task-PIOA) for $\mathcal{T}$. If $\mathcal{M} = (\mathcal{T}, \mathcal{S})$ is a probabilistic system, then the *external behavior* of $\mathcal{M}$, $\mathsf{extbeh}(\mathcal{M})$, is the total function that maps each environment task-PIOA $\mathcal{E}$ for $\mathcal{M}$ to the set $\bigcup_{S \in \mathcal{S}} \mathsf{tdists}(S \| \mathcal{E})$. Thus, for each environment, we consider the set of trace distributions that arise from two choices: of a local scheduler of $\mathcal{M}$ and of a global task schedule ρ.

If $\mathcal{M}_1 = (\mathcal{T}_1, \mathcal{S}_1)$ and $\mathcal{M}_2 = (\mathcal{T}_2, \mathcal{S}_2)$ are comparable probabilistic systems (i.e., $\mathcal{T}_1$ and $\mathcal{T}_2$ are comparable), then $\mathcal{M}_1$ *implements* $\mathcal{M}_2$, written $\mathcal{M}_1 \leq_0 \mathcal{M}_2$, provided that $\mathsf{extbeh}(\mathcal{M}_1)(\mathcal{E}) \subseteq \mathsf{extbeh}(\mathcal{M}_2)(\mathcal{E})$ for every environment (action-deterministic) task-PIOA $\mathcal{E}$ for both $\mathcal{M}_1$ and $\mathcal{M}_2$. We obtain a sufficient condition for implementation of probabilistic systems, in which each local scheduler for the low-level system always corresponds to the same local scheduler of the high-level system.

Theorem 5.1: Let $\mathcal{M}_1 = (\mathcal{T}_1, \mathcal{S}_1)$ and $\mathcal{M}_2 = (\mathcal{T}_2, \mathcal{S}_2)$ be two comparable probabilistic systems. Suppose that f is a total function from $\mathcal{S}_1$ to $\mathcal{S}_2$ such that, for every $S_1 \in \mathcal{S}_1$, $S_1 \leq_0 f(S_1)$. Then $\mathcal{M}_1 \leq_0 \mathcal{M}_2$.

We also obtain a compositionality result for probabilistic systems. The proof is similar to that of Theorem 2.9, for the action-deterministic case.

Theorem 5.2: Let $\mathcal{M}_1$, $\mathcal{M}_2$ be comparable probabilistic systems such that $\mathcal{M}_1 \leq_0 \mathcal{M}_2$, and let $\mathcal{M}_3$ be a probabilistic system compatible with each of $\mathcal{M}_1$ and $\mathcal{M}_2$. Then $\mathcal{M}_1 \| \mathcal{M}_3 \leq_0 \mathcal{M}_2 \| \mathcal{M}_3$.

VI. Conclusions

We have extended the traditional PIOA model with a task mechanism, which provides a systematic way of resolving nondeterministic scheduling choices without using information about past history. We have provided basic machinery for using the resulting task-PIOA framework for verification, including a compositional trace-based semantics and a new kind of simulation relation. We have proposed extending the framework to allow additional nondeterminism, resolved by schedulers that use only local information. We have illustrated the utility of these tools with a case study involving analysis of an Oblivious Transfer cryptographic protocol.

Although our development was motivated by concerns of cryptographic protocol analysis, partial-information scheduling is interesting in other settings. For example, some distributed algorithms work with partial-information adversarial schedulers, althrough the problems they solve are provably unsolvable with perfect-information adversaries [Cha96], [Asp03]. Also, partial-information scheduling is realistic for modeling large distributed systems, in which basic scheduling decisions are made locally, and not by any centralized mechanism.

Many questions remain in our study of task-PIOAs: Our notion of implementation, $\leq_0$, is defined by considering all environments; can we characterize $\leq_0$ using a small subclass of environments? Can our simulation relation notion be simplified without sacrificing soundness or applicability? Also, our notion of local schedulers needs further development.

It remains to consider more applications of task-PIOAs, for cryptographic protocol analysis and for other applications. A next step in cryptographic protocol analysis is to formulate and prove protocol composition results like those of [PW00], [Can01] in terms of task-PIOAs. Finally, we would like to model perfect-information schedulers, as used for analyzing randomized distributed algorithms, using task-PIOAs.

Acknowledgments: We thank Frits Vaandrager for collaboration in early stages of this project, and Michael Backes, Anupam Datta, Ralf Kuesters, John Mitchell, Birgit Pfitzmann, and Andre Scedrov for technical discussions that helped us in clarifying our ideas and their connections to other work in analysis of cryptographic protocols. We thank Silvio Micali for impressing upon us the importance of adaptive adversarial schedulers in the cryptographic setting. We thank Sayan Mitra both for technical discussions and for help in producing the paper.

References

[Asp03] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2-3):165–175, 2003.

[AH90] J. Aspnes and M. Herlihy. Fast, randomized consensus using shared memory. *Journal of Algorithms*, 11(2), pages 441–461, September 1990.

[B83] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols.*Proc. 2nd ACM Symposium on Principles of Distributed Computing*, pages 2730, Montreal, Quebec, Canada, August 1983.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. *Proc.42nd IEEE Symposium on Foundations of Computing*, pages 136–145, 2001.

[CC+05] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala. Using probabilistic I/O automata to analyze an oblivious transfer protocol. Technical Report MIT-LCS-TR-1001a, MIT CSAIL, 2005.

[CC+06a] 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-2006-019, MIT CSAIL, 2006. Available at `http://hdl.handle.net/1721.1/31310`

[CC+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-2006-023, 2006.

[CLSV04] L. Cheung, N. Lynch, R. Segala and F. Vaandrager. Switched PIOA: Parallel Composition via Distributed Scheduling. To appear in TCS, Special Issue on FMCO 2004.

[CH05] L. Cheung and M. Hendriks. Causal dependencies in parallel composition of stochastic processes. Technical Report ICIS-R05020, Institute for Computing and Information Sciences, University of Nijmegen, 2005.

[Cha96] T.D. Chandra. Polylog randomized wait-free consensus. *Proc. 15th ACM Symposium on Principles of Distributed Computing*, pages 166–175, 1996.

[dA99] L. de Alfaro. The verification of probabilistic systems under memoryless partial-information policies is hard. *Proc. PROB-MIV 99*, pages 19–32, 1999.

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

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

[KLA98] L.P. Kaelbling, M.L. Littman, and A.R.Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[LMMS98] P. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Conference on Computer and Communications Security*, pages 112–121, 1998.

[LSS94] N.A. Lynch, I. Saias, and R. Segala. Proving time bounds for randomized distributed algorithms. *Proc. 13th ACM Symposium on Principles of Distributed Computing*, pages 314–323, 1994.

[LSV03] N.A. Lynch, R. Segala, and F.W. Vaandrager. Compositionality for probabilistic automata. *Proc. 14th International Conference on Concurrency Theory (CONCUR 2003)*, volume 2761 of *LNCS*, pages 208–221. Springer-Verlag, 2003.

[LT89] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.

[MMS03] P. Mateus, J.C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. *Proc. CONCUR 2003*, pages 323–345, 2003.

[PSL00] A. Pogosyants, R. Segala, and N.A. Lynch. Verification of the randomized consensus algorithm of Aspnes and Herlihy: a case study. *Distributed Computing*, 13(3):155–186, 2000.

[Put94] M.L. Puterman. *Markov Decision Process – Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[PW00] B. Pfitzman and M. Waidner. Composition and integrity preservation of secure reactive systems. *Proc. CCS 2000*, 2000.

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

[R82] M. Rabin. The choice coordination problem. *Acta Informatica*, 17:121–134, 1982.

[RMST04] A. Ramanathan, J.C. Mitchell, A. Scedrov, and V. Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. *Proc. FOSSACS 2004*, 2004.

[SdV04] A. Sokolova and E.P. de Vink. Probabilistic automata: system types, parallel composition and comparison. In *Validation of Stochastic Systems*, volume 2925 of *LNCS*, pages 1–43. Springer-Verlag, 2004.

[Seg95] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1995. Available as Technical Report MIT/LCS/TR-676.

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

[SV99] M.I.A. Stoelinga and F.W. Vaandrager. Root contention in IEEE 1394. *Proc. 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems*, volume 1601 of *LNCS*, pages 53–74. Springer-Verlag, 1999.