On the Role of Scheduling in Simulation-Based Security*

Ran Canetti[1,2], Ling Cheung[2], Nancy Lynch[2], and Olivier Pereira[3]

[1] IBM TJ Watson Research Center
[2] Massachusetts Institute of Technology
[3] Université catholique de Louvain, olivier.pereira@uclouvain.be

Abstract. In a series of papers, Küsters et al. investigated the relationships between various notions of simulation-based security. Two main factors, the placement of a "master process" and the existence of "forwarder processes", were found to affect the relationship between different definitions. In this extended abstract, we add a new dimension to the analysis of simulation-based security, namely, the scheduling of concurrent processes. We show that, when we move from sequential scheduling (as used in previous studies) to task-based nondeterministic scheduling, the same syntactic definition of security gives rise to incomparable semantic notions of security. Under task-based scheduling, the hierarchy based on placement of "master process" is no longer relevant, because no such designation is necessary to obtain meaningful runs of a system. On the other hand, the existence of "forwarder processes" remains an important factor.

1 Introduction

In simulation-based security, the behavior of a multi-party protocol ρ executing in the "real world", often in the presence of an adversary *Adv*, is compared against the behavior of a simulator *Sim* interacting with an ideal process ϕ (also called a functionality). Intuitively, if the behaviors of these two systems are indistinguishable, then ρ is at least as secure as ϕ.

This notion of simulation traces back to the early works of Micali et al. on *zero-knowledge proof systems* [GMR85] and *secure function evaluation* [GMW87]. Much progress was made during the 1990's [GL90,Bea91,MR91,PW94,Can95], leading to the general definitions of *reactive simulatability (RSIM)* [PW01] and *universally composable (UC) security* [Can01]. Many related definitions also appeared, including *black-box (BB) simulatability* [PW01] and *strong simulatability (SS)* [DKM+04].

* Canetti was supported by NSF CyberTrust Grant #0430450. Cheung was supported by NSF Award #CCR-0326227. Lynch was supported 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 was supported by the Belgian National Fund for Scientific Research (F.R.S.-FNRS).

Using informal notations of parallel composition $\|$ and indistinguishability $\approx$, some major variants of simulation-based security can be formulated as follows:

- *Reactive Simulatability*: $\rho \leq_{RSIM} \phi$ iff $\forall Adv\ \forall Env\ \exists Sim : \rho\|Adv\|Env \approx \phi\|Sim\|Env$.
- *UC Security*: $\rho \leq_{UC} \phi$ iff $\forall Adv\ \exists Sim\ \forall Env : \rho\|Adv\|Env \approx \phi\|Sim\|Env$.
- *Black-Box Simulatability*: $\rho \leq_{BB} \phi$ iff $\exists Sim\ \forall Adv\ \forall Env : \rho\|Adv\|Env \approx \phi\|Adv\|Sim\|Env$.
- *Strong Simulatability*: $\rho \leq_{SS} \phi$ iff $\exists Sim\ \forall Env : \rho\|Env \approx \phi\|Sim\|Env$.

Note that the indistinguishability condition is stated relative to an environment process *Env*, which takes on the role of a "distinguisher". Moreover, in the last three definitions, the simulator must be specified before the environment. This essentially guarantees composability of the security definition, because the simulation must be successful regardless of the environment in which the protocol ρ is executed.

The relationships between some of these variants can be deduced simply by examining their logical structures. For example, strong simulatability implies black-box simulatability, which in turn implies both reactive simulatability and UC security. The other implications are less obvious and are investigated by Datta et al. in [DKM+04,DKMR05], which present a hierarchy of simulation-based security definitions using two criteria: (i) the identity of the master process (which may be the environment, adversary or simulator) and (ii) the definability of a *forwarder* process that is able to forward an unbounded number of messages. In particular, it is shown that strong simulatability is equivalent to UC security if and only if forwarder processes are definable.

The notion of a master process used in the first criterion is common in modeling frameworks with *sequential activation*: given a system of machines/processes executing in parallel, at most one machine is active at any given point in time, and, when the active machine produces a message, the intended recipient is the next active machine. A designated *master process* is triggered if for whatever reason the chain of activation is broken.

Sequential activation is implemented in many frameworks, including the Interactive Turing Machine (ITM) model in [Can01,Küs06], the Reactive System (RS) model[1] of [PW01,BPW04] and the Sequential Probabilistic Process Calculus (SPPC) of [DKM+04,DKMR05]. Since machines are activated via message delivery, one need not specify a separate scheduler to resolve nondeterminism (as is typical in traditional models of concurrency).

Although less common, non-sequential activation is also found in cryptography-oriented frameworks, including the Probabilistic Polynomial-time Process Calculus (PPC) of [LMMS98,MMS03,DKM+04,MRST06] and the Task-PIOA model of [CCK+06b,CCK+06c]. Here, nondeterminism is resolved using schedulers, which are state-dependent functions (or Markov chains) in PPC and oblivi-

[1] In general, the high-level scheduling in RS need not be sequential: messages are not delivered immediately; instead, they are stored in buffers that may be triggered by a component other than the sender. However, sequential scheduling is typically implemented in actual cryptographic protocol analysis [BPW03].

ous task sequences in Task-PIOA. We believe these scheduling mechanisms are more natural for the modeling of cryptographic protocols, as they capture the fact that the ordering of events is often difficult to predict in large distributed systems, and cannot be fully controlled even by adversarial components.

Since scheduling is an integral part of the semantics of concurrent processes, it is natural to ask whether the same definition of security would have different meanings when we move between sequential and non-sequential scheduling. It was a folklore belief that the two types of scheduling are semantically equivalent, because sequential scheduling can be emulated in a non-sequential framework, and vice versa. In this paper, we show that such claims are misleading, because there exist protocols for which the same security relation holds under one type of scheduling but not under the other. This shows scheduling is in fact an important aspect in simulation-based security.

In our first example (Section 3.1), we give two protocols, one of which implements an input/ouput correlation explicitly while the other one does not. Under sequential scheduling, these two protocols are equivalent with respect to UC security, because the input message determines which machine is activated next and hence which output is produced. This shows sequential scheduling can "create" correlations that are not present in machine specifications. In contrast, oblivious task-based scheduling does not allow the possibility to forge correlations between dynamically chosen values, because a scheduler is a sequence of tasks that are chosen nondeterministically in advance. The same two protocols are therefore UC-inequivalent under oblivious scheduling.

In our second example (Section 3.2), we show that sequential scheduling can give the distinguisher environment additional power, because it can control the ordering of events elsewhere in the system by timing its own messages. (This holds even if the environment is *not* the master scheduler, because the master scheduler kicks in only if the activation chain is broken.) This allows the environment to distinguish two protocols that are UC-equivalent under oblivious scheduling.

Finally, we observe that the "forwarder" criterion of [DKM$^+$04,DKMR05] remains meaningful and important when we move from sequential to non-sequential activation. (The "master process" criterion is no longer meaningful, because there is no designation of master processes in a non-sequential framework.) We prove that strong simulatability is equivalent to UC security in the Task-PIOA framework. The proof rests upon the fact that task-PIOA specifications do *not* impose length restrictions on task schedules, and hence forwarder processes are definable as task-PIOAs.

Roadmap In Section 2, we briefly review the Task-PIOA framework and our general modeling paradigm for cryptographic protocol analysis. Then, in Section 3, we use two examples to show that sequential and non-sequential activation schemes give rise to incomparable notions of security. In Section 4, we prove that forwarders are definable in Task-PIOA, and that strong simulatability is equivalent to UC security.

2 Security Modeling with Task-PIOAs

Our basic framework is that of task-PIOAs [CCK+06b], which provides a partial-information scheduling mechanism suitable for cryptographic protocol analysis.

PIOAs A *probabilistic I/O automaton (PIOA)* $\mathcal{A}$ is a tuple $\langle Q, \bar{q}, I, O, H, \Delta \rangle$, 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; (iii) $\Delta \subseteq Q \times (I \cup O \cup H) \times \mathsf{Disc}(Q)$ 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 $\langle q, a, \mu \rangle \in \Delta$ for some μ. The set $Act := I \cup O \cup H$ is called the *action alphabet* of $\mathcal{A}$. If $I = \emptyset$, then $\mathcal{A}$ is said to be *closed*. The set of *external* actions of $\mathcal{A}$ is $I \cup O$ and the set of *locally controlled* actions is $O \cup H$. We assume that $\mathcal{A}$ satisfies the following conditions.

- **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 $\mu \in \mathsf{Disc}(Q)$ such that $\langle q, a, \mu \rangle \in \Delta$.

Parallel composition for PIOAs is based on synchronization of shared actions. Two PIOAs $\mathcal{A}_i$, $i \in \{1, 2\}$, are said to be *compatible* if $Act_i \cap H_j = O_i \cap O_j = \emptyset$ whenever $i \neq j$. In that case, we define their *composition* $\mathcal{A}_1 \| \mathcal{A}_2$ to be $\langle Q_1 \times Q_2, \langle \bar{q}_1, \bar{q}_2 \rangle, (I_1 \cup I_2) \setminus (O_1 \cup O_2), O_1 \cup O_2, H_1 \cup H_2, \Delta \rangle$, where Δ is the set of triples $\langle \langle q_1, q_2 \rangle, a, \mu_1 \times \mu_2 \rangle$ such that (i) a is enabled in some q_i and (ii) for every i, if $a \in A_i$ then $\langle q_i, a, \mu_i \rangle \in \Delta_i$, otherwise μ_i assigns probability 1 to q_i. A *hiding* operator is also available: given $\mathcal{A} = \langle Q, \bar{q}, I, O, H, \Delta \rangle$ and $S \subseteq O$, $\mathsf{hide}(\mathcal{A}, S)$ is the tuple $\langle Q, \bar{q}, I, O', H', \Delta \rangle$, where $O' = O \setminus S$ and $H' = H \cup S$. This prevents synchronizations of actions in S with any other PIOA.

Task-PIOAs To resolve nondeterminism, we make use of the notion of tasks introduced in [CCK+06b]. Formally, a *task-PIOA* is a pair $(\mathcal{A}, \mathcal{R})$ such that (i) $\mathcal{A}$ is a PIOA and (ii) $\mathcal{R}$ is a partition of the locally-controlled actions. With slight abuse of notation, we use $\mathcal{A}$ to refer to both the task-PIOA and the underling PIOA. The equivalence classes in $\mathcal{R}$ are referred to as *tasks*. Unless otherwise stated, we will use terminologies inherited from the PIOA setting.

The following axiom is imposed on task-PIOAs.

- **Action Determinism:** For every state $q \in Q$ and every task $T \in \mathcal{R}$, 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.

Given compatible task-PIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we define their *composition* to be $\langle \mathcal{A}_1 \| \mathcal{A}_2, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$. The hiding operator for PIOAs also extends in the obvious way: given a set S of output actions, $hide(\langle \mathcal{A}, \mathcal{R} \rangle, S)$ is simply $\langle hide(\mathcal{A}, S), \mathcal{R} \rangle$.

Finally, a *task schedule* for a closed task-PIOA $\langle \mathcal{A}, \mathcal{R} \rangle$ is a finite or infinite sequence $\rho = T_1.T_2.T_3 \ldots$ of tasks in $\mathcal{R}$. This induces a well-defined (probabilistic) execution of $\mathcal{A}$ as follows: (i) from the start state $\bar{q}$, we apply the first task T_1; (ii) due to action- and transition-determinism, T_1 specifies at most one transition from $\bar{q}$; (iii) if such transition exists, it is taken, otherwise nothing happens; (iv) repeat with remaining T_i's.

Example: Adaptive Adversary For cryptographic applications, we adopt the following modeling paradigm.

(1) An adaptive adversary is modeled as a system component, for example, a message delivery service that can eavesdrop on network communications and control the order of message delivery. Thus, the adversary resolves the so-called *high-level* nondeterminism.
(2) *Low-level* nondeterminism is resolved by a task schedule chosen nondeterministically in advance. For example, in a typical protocol, many different parties make independent random choices, and it is inconsequential which of them does so first. A task schedule fixes a particular order in which the different coin tosses occur.

We illustrate this paradigm via an example. Consider a toy protocol in which a sender, S, and two receivers, R_0 and R_1, exchange messages via an adversary Adv (Figure 1). The sender S also chooses two random bits b and s independently. The first bit b is announced to the adversary Adv and the second bit s is kept secret until S receives an acknowledgment from either R_0 or R_1. If the acknowledgment from R_b arrives before the acknowledgment from R_{1-b}, S reveals s to Adv, otherwise s remains secret.

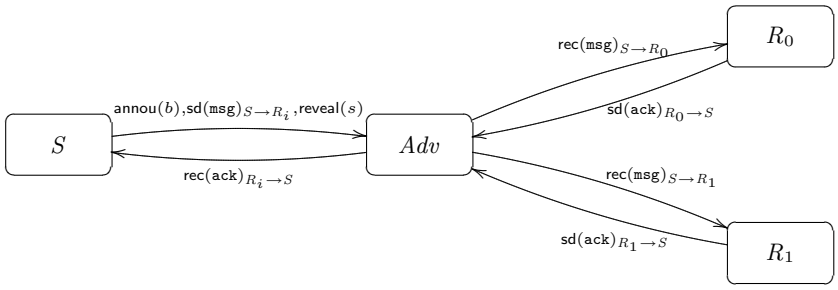

Fig. 1. A Toy Protocol

The adversary Adv delivers the messages from S to R_i whenever they are available, while the acknowledgments from R_i to S are buffered until S announces b. Then Adv delivers $\mathtt{ack}_b$ before $\mathtt{ack}_{1-b}$. Finally, a receiver R_i simply accepts the message from S and responds with an acknowledgment. A detailed description of these automata appears in [CCK$^+$06a].

In this protocol, the ordering between $\mathsf{rec}(\mathtt{ack})_{R_0\to S}$ and $\mathsf{rec}(\mathtt{ack})_{R_1\to S}$ is a good example of high-level nondeterminism. Once these acknowledgments are placed onto the network (via actions $\mathsf{sd}(\mathtt{ack})_{R_i\to S}$), the adversary controls their transit delays. In particular, the adversary described above waits until it learns the value of b, and then it delivers the acknowledgment from R_b. This ensures that S will reveal s if the task $\mathsf{reveal}(*)$ is scheduled subsequently. In fact, it is easy to check that the following task schedule allows Adv to learn s with

probability 1. This shows *Adv* is *adaptive*, since b is generated randomly during execution.

$\mathsf{choose}.\ \mathsf{annou}(*).\ \mathsf{sd}(\mathtt{msg})_{S\to R_0}.\ \mathsf{sd}(\mathtt{msg})_{S\to R_1}.\ \mathsf{rec}(\mathtt{msg})_{S\to R_0}.\ \mathsf{rec}(\mathtt{msg})_{S\to R_1}.$
$\mathsf{sd}(\mathtt{ack})_{R_0\to S}.\ \mathsf{sd}(\mathtt{ack})_{R_1\to S}.\ \mathsf{rec}(\mathtt{ack})_{R_0\to S}.\ \mathsf{rec}(\mathtt{ack})_{R_1\to S}.\ \mathsf{reveal}(*)$

We now turn to low-level nondeterminism. For instance, the ordering between $\mathsf{sd}(\mathtt{msg})_{S\to R_0}$ and $\mathsf{sd}(\mathtt{msg})_{S\to R_1}$ is inessential in the security analysis, provided they are both performed by S. Similarly, the ordering between $\mathsf{annou}(*)$ and $\mathsf{sd}(\mathtt{msg})_{S\to R_i}$ is also inessential. All of these are examples of low-level nondeterministic choices and represent implementation freedom in S. That is, an actual implementation of S may perform $\mathsf{annou}(*)$, $\mathsf{sd}(\mathtt{msg})_{S\to R_0}$ and $\mathsf{sd}(\mathtt{msg})_{S\to R_1}$ in any order. We capture all these possibilities in our formal semantics by quantifying over all possible task schedules.

Implementation The formal semantics of a closed task-PIOA is given in terms of the *trace distributions* induced by task schedules. As we described earlier, each task schedule induces a probabilistic run, from which a trace distribution is obtained by abstracting away state information. That is, a trace distribution contains only information about actions taken during the run.

For a possibly open task-PIOA $\mathcal{A}$, the semantics is given relative to closing environments: a task-PIOA *Env* is an *environment* for $\mathcal{A}$ if it is compatible with $\mathcal{A}$ and $\mathcal{A}\|Env$ is closed. The *external behavior* of $\mathcal{A}$ is then the mapping that takes each environment *Env* to the set of trace distributions of $\mathcal{A}\|Env$ (denoted $\mathsf{TrDists}(\mathcal{A}\|Env)$).

We also define an implementation relation between task-PIOAs with the same I/O interface, expressing the idea that every possible behavior of one automaton in a particular environment is also a possible behavior of another automaton in the same environment. Formally, $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to be *comparable* if $I_1 = I_2$ and $O_1 = O_2$. In that case, $\mathcal{A}_1$ is said to *implement* $\mathcal{A}_2$, denoted $\mathcal{A}_1 \leq_0 \mathcal{A}_2$, if $\mathsf{TrDists}(\mathcal{A}_1\|Env) \subseteq \mathsf{TrDists}(\mathcal{A}_2\|Env)$ for all environments *Env* for $\mathcal{A}_1$ and $\mathcal{A}_2$.

The subscript 0 in $\leq_0$ refers to the requirement that every trace distribution in $\mathsf{TrDists}(\mathcal{A}_1||Env)$ must have an identical match in $\mathsf{TrDists}(\mathcal{A}_2||Env)$. For cryptographic protocol analysis, we define a variant based on the probability that *Env* produces a special "accept" output. An *approximate* implementation, denoted $\leq_{\mathsf{neg,pt}}$, is then defined for task-PIOA families, which allows "negligible" discrepancies between acceptance probabilities. More details on $\leq_{\mathsf{neg,pt}}$ can be found in Appendix A.

3 Separation between Sequential Scheduling and Oblivious Scheduling

We use two simple examples to illustrate the effect of different scheduling schemes on the semantics of security definitions. In Section 3.1, we exhibit two systems that are UC-equivalent under sequential scheduling, but they do not implement each other under oblivious scheduling. In Section 3.2, we present two systems in the opposite situation.

3.1 Sequential indistinguishability

Consider two variants of the system depicted in Figure 2. In the first variant, the environment interacts with two task-PIOAs A_0 and A_1, which simply answer requests. That is, task-PIOA A_i answers each $Hello_i$ input from the environment with the output action i. In the second variant, the environment interacts with two task-PIOAs B_0 and B_1, which behave like beacons. That is, each B_i spontaneously and persistently produces the output action i. These beacons also accept $Hello_i$ inputs, but they have no effect. The codes for A_i and B_i are given in Appendix C, Figure 4.

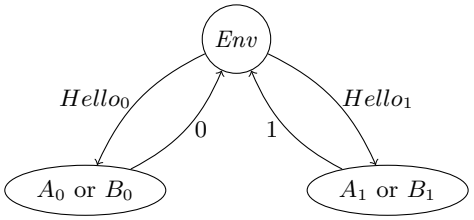

Fig. 2. Diagram for *Answer* and *Beacon*

We claim that, under sequential activation, no environment can distinguish the system $Answer := A_0 \| A_1$ from the system $Beacon := B_0 \| B_1$. Indeed, the only way to activate A_i (or B_i) is the $Hello_i$ action, after which both A_i and B_i will respond with i.

On the other hand, we observe that $Answer \not\leq_0 Beacon$, and $Beacon \not\leq_0 Answer$. Consider an environment $\mathcal{E}$ that has one single output task $Hello = \{Hello_0, Hello_1\}$, where the activation of $Hello_0$ or $Hello_1$ is decided through some internal coin flipping, performed as the unique action in the task $Flip = \{flip\}$ of $\mathcal{E}$. Consider now the task schedule $\rho := Flip.Hello.Out_0$, where Out_0 is the task $\{0\}$. In system $Answer \| \mathcal{E}$, the resulting trace distribution will be $Hello_0.0$ with probability $\frac{1}{2}$ and $Hello_1$ with probability $\frac{1}{2}$; hence 0 occurs with probability $\frac{1}{2}$. This trace distribution cannot be matched by any task schedule for $Beacon \| \mathcal{E}$, because task schedules are chosen nondeterministically in advance. More precisely, a task schedule for $Beacon \| \mathcal{E}$ either schedules Out_0 after $Flip.Hello$ or it does not. In the first case, 0 occurs with probability 1 and in the second with probability 0.

Furthermore, the trace distribution obtained from applying ρ to $Beacon \| \mathcal{E}$ cannot be matched by any task schedule for $Answer \| \mathcal{E}$, because no task schedule can ensure that 0 occurs with probability 1 in $Answer \| \mathcal{E}$.

This example can be easily extended to the security setting, where the input and output actions of A_i's and B_i's are treated as protocol inputs and outputs. We observe that $Answer \leq_{UC} Beacon$ when we have sequential scheduling, while this relation does not hold with task-based scheduling. Indeed, since

Answer and *Beacon* do not send any message on the network, the adversary cannot observe anything and the best we can do is to define the simulator as a copy of the adversary. Eventually, the UC security property comes down to the ability of the environment to distinguish *Answer* from *Beacon*.

To sum up, we see that a sequential scheduling scheme, like the ones considered in [Can01,BPW03,DKMR05,Küs06], may introduce constraints in the ordering of events that do not necessarily reflect actual network behavior (e.g., the relative timing between $Hello_i$ and i in $Beacon \| \mathcal{E}$). This may hide characteristics that could be exploited by an attacker in practice.

3.2 Task-based indistinguishability

In our second example, we consider two variants of the system depicted in Figure 3. The first variant, called *Secret*, consists of automata B_0, B_1 (as specified in Section 3.1) and *Box*. The *Box* automaton selects a random k-bit string x, and transmits it to the environment. Then, it waits for bit-valued messages from B_0 and B_1, which are stored in a k-bit buffer in order as they arrive. Eventually, if the buffer content coincides with x, *Box* performs an *ok* output action.

The second variant, $Secret_s$, is obtained from *Secret* by replacing *Box* with Box_s, which differs from *Box* in only one way: the *ok* action is never enabled. That is, Box_s does not perform any output actions except for sending x.

The *Box* and Box_s components are actually defined as task-PIOA families $\overline{Box} = \{Box_k\}_{k\in\mathbb{N}}$ and $\overline{Box_s} = \{(Box_s)_k\}_{k\in\mathbb{N}}$. The codes for Box_k and $(Box_s)_k$ are given in Appendix C, Figure 5. The index k determines the length of the secrete x.

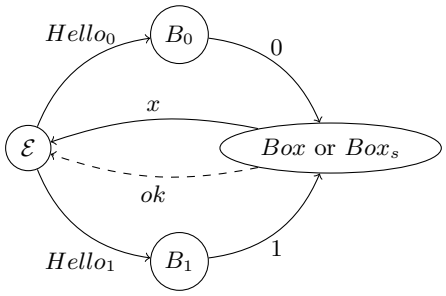

Fig. 3. Diagram for *Secret* and $Secret_s$

We first observe that the families $\overline{Box}$ and $\overline{Box_s}$ can be distinguished with high probability, regardless of the choice of scheduling scheme. In particular, consider an environment $\mathcal{E}$ which simply waits for x and issues 0's and 1's according to x.

Things become more interesting when B_0 and B_1 are inserted between the environment and *Box* (respectively, Box_s). Formally, $\overline{Secret} = \{Secret_k\}_{k\in\mathbb{N}}$ is

defined by: $Secret_k = hide(B_0 \| B_1 \| Box_k, \{0,1\})$. Similarly for $\overline{Secret_s}$. Thus, the environment is separated from Box (respectively, Box_s) by B_0 and B_1, which produce bits 0 and 1 constantly, regardless of the environment's behavior.

We claim that an environment can still efficiently distinguish $\overline{Secret}$ from $\overline{Secret_s}$ when sequential scheduling is used. Indeed, if the environment performs the $Hello_0$ and $Hello_1$ actions according to x, the beacons B_0 and B_1 will be activated in the appropriate order, and hence Box_k will eventually perform the ok action with high probability. However, this last action never occurs in the $\overline{Secret_s}$ system since ok is never enabled.

On the other hand, we claim that the following relation holds: $\overline{Secret} \leq_{\mathsf{neg,pt}} \overline{Secret_s}$. This is because x is selected randomly at run time, while the ordering of 0's and 1's are determined by a task schedule chosen nondeterministically in advance. For any parameter k and any fixed task schedule ρ, the probability that x coincides with the ordering of the first k occurrences of Out_0 and Out_1 is at most 2^{-k}. Consequently, the ok action is enabled in Box_k with probability at most 2^{-k}. (Note that the relation $\overline{Secret} \leq_0 \overline{Secret_s}$ does not hold, because the probability of ok is nonzero.)

We can easily transpose this example to the security setting, by considering the $Hello_0$, $Hello_1$, ok and x actions as protocol inputs and outputs. Then $Secret \leq_{UC} Secret_s$ holds under oblivious scheduling, but not under sequential scheduling. This is the opposite situation compared to Section 3.1.

While in Section 3.1 we show that oblivious scheduling gives more distinguishing power to the environment, this example shows oblivious scheduling may also weaken the environment, by removing its ability to control the relative timing of certain events. The same weakening also applies to adversarial components, and we believe it yields a realistic model of distributed systems: the adversarial network should not have control over the ordering of *local* actions performed by non-corrupted components.

This example also highlights the fact that, in the Task-PIOA framework, task schedules are fixed *after* all components (environment, adversary, protocol parties, etc.) are in place, therefore components do *not* "know" the order in which activation will take place. This is clearly not the case under sequential scheduling. Again, it seems to be a realistic paradigm, because adversarial components operating in a large network are likely also subject to the effect of network noise, hence unable to predict precisely when different events may happen.

4 Equivalence between UC and SS

In this section, we prove that UC security and strong simulatability are equivalent in our framework. Essentially, we show that time-bounded task-PIOAs satisfy the forwarder axiom of [DKMR05].

Structures First we define the notion of structures in the spirit of [PW01]: a *structure* Π is a pair $\langle \mathcal{A}, EAct \rangle$, where $\mathcal{A}$ is a task-PIOA and $EAct$ is a subset of the external actions of $\mathcal{A}$, called the *environment actions*. The set of *adversary*

actions is defined to be $AAct := (I \cup O) \setminus EAct$. We also have: (i) $EI := EAct \cap I$ (*environment inputs*), (ii) $EO := EAct \cap O$ (*environment outputs*), (iii) $AI := AAct \cap I$ (*adversary inputs*) and (iv) $AO = AAct \cap O$ (*adversary inputs*).

Two structures Π_1 and Π_2 are said to be *comparable* if $EI_1 = EI_2$ and $EO_1 = EO_2$. They are *compatible* if $\mathcal{A}_1$ and $\mathcal{A}_2$ are compatible task-PIOAs and $Ext_1 \cap Ext_2 = EAct_1 \cap EAct_2$. That is, every shared action must be an environment action of both automata. Composition is straightforward: given compatible Π_1 and Π_2, their *composition* $\Pi_1 \| \Pi_2$ is the structure $\langle \mathcal{A}_1 \| \mathcal{A}_2, EAct_1 \cup EAct_2 \rangle$.

As the names suggest, an adversary interact with a protocol via adversary actions. Formally, a task-PIOA Adv is an *adversary* for the structure Π if: (i) Adv is compatible with $\mathcal{A}_\Pi$, (ii) $Act_{Adv} \cap Act_\Pi \subseteq AAct_\Pi$, and (iii) $AI_\Pi \subseteq O_{Adv}$. The last condition says that Adv provides all adversary inputs of Π.

Finally, we consider hiding for structures: given a structure $\langle \mathcal{A}, EAct \rangle$ and a set S of output actions of $\mathcal{A}$, we define $hide(\langle \mathcal{A}, EAct \rangle, S)$ to be the structure $\langle hide(\mathcal{A}, S), EAct \setminus S \rangle$. That is, the newly hidden actions can no longer be environment actions.

All of the definitions above can be formulated easily in the setting with time bounds, as well as for families of structures. Some details are provided in Appendix B.

Secure Emulation We have now enough machinery to formulate UC security and strong simulatability.

Definition 1 (UC-Security). *Suppose ρ and ϕ are comparable structure families. We say that ρ* UC-emulates *ϕ (denoted $\rho \leq_{UC} \phi$) if, for every polynomial time-bounded adversary family Adv for ρ, there is a polynomial time-bounded adversary family Sim for ϕ such that:*

$$hide(\rho || Adv, AAct_\rho) \leq_{\mathsf{neg,pt}} hide(\phi || Sim, AAct_\phi).$$

Definition 2 (Strong Simulatability). *Suppose ρ and ϕ are comparable structure families with $AAct_\rho \cap AAct_\phi = \emptyset$. We say that ρ* strongly simulates *ϕ (denoted $\rho \leq_{SS} \phi$) if there is a polynomial time-bounded adversary family Sim for ϕ such that:*

$$\rho \leq_{\mathsf{neg,pt}} hide(\phi || Sim, AAct_\phi).$$

Theorem 1. *Definitions 1 and 2 are equivalent.*

Proof (Sketch). Suppose ρ and ϕ are defined as in the hypotheses. Suppose first that $\rho \leq_{SS} \phi$, as in Definition 2. This means that there is a polynomial time-bounded adversary family Sim for ϕ such that $\rho \leq_{\mathsf{neg,pt}} \mathsf{hide}(\phi || Sim, AAct_\phi)$. We fix any polynomial time-bounded adversary family Adv for ρ, and define the polynomial time-bounded adversary family $Sim' = \mathsf{hide}(Sim || Adv, AAct_\rho)$ for ϕ. Now, using the composition and hiding properties of the $\leq_{\mathsf{neg,pt}}$ relation, we obtain that $\mathsf{hide}(\rho || Adv, AAct_\rho) \leq_{\mathsf{neg,pt}} \mathsf{hide}(\phi || Sim', AAct_\phi)$, which shows that $\rho \leq_{UC} \phi$, as required in Definition 1.

Now suppose that $\rho \leq_{UC} \phi$. First we define ρ' as $f(\rho)$, where f is some renaming function on $AAct_\rho$ so that $f(AAct_\rho)$ is completely fresh. It is easy to check

that $\rho' \leq_{UC} \phi$. Then we define a polynomial time-bounded adversary family Adv for ρ', which acts as a forwarder for all actions in $f(AAct_\rho)$. More precisely, each action $f(a) \in f(AAct_\rho)$ is forwarded by the action a. This adversary has only one task, $Forward = AAct_\rho$. Note that ρ and $\mathsf{hide}(\rho' \| Adv, f(AAct_\rho))$ are comparable. Finally, for every environment Env, we exhibit a 2-bounded simulation relation between $\rho \| Env$ and $\mathsf{hide}(\rho' \| Adv, f(AAct_\rho)) \| Env$ as follows: every task T of $\rho \| Env$ is mapped to the task sequence $f(T).Forward$.

This shows that $\rho \leq_{\mathsf{neg,pt}} \mathsf{hide}(\rho' \| Adv, f(AAct_\rho))$. Using that claim that $\rho' \leq_{UC} \phi$, we obtain a polynomial time-bounded adversary family Sim for ϕ with $\mathsf{hide}(\rho' \| Adv, f(AAct_\rho)) \leq_{\mathsf{neg,pt}} \mathsf{hide}(\phi || Sim, AAct_\phi)$. By transitivity of $\leq_{\mathsf{neg,pt}}$, we have $\rho \leq_{\mathsf{neg,pt}} \mathsf{hide}(\phi \| Sim, AAct_\phi)$.

□

Note that the forwarder adversary in the proof of Theorem 1 is unbounded, in the sense that it can forward as many messages as any environment can produce. Such unbounded forwarders are definable in the Task-PIOA framework because we do not place any *a priori* length restrictions on task schedules. These restrictions are handled in the definition of $\leq_{\mathsf{neg,pt}}$, where we quantify over all polynomial bounds on the length of task schedules (cf. Appendix A). Moreover, the schedule length bound of the ideal system may depend on the schedule length bound of the real system. Since the real system includes the environment, we may choose a large enough schedule length bound for the ideal system so that all messages from the environment will be forwarded. (In the proof of Theorem 1, every polynomial bound q on the length of task schedules for $\rho \| Env$ can be matched by the bound $2q$ for $\mathsf{hide}(\rho' \| Adv, f(AAct_\rho)) \| Env$, since the proposed simulation relation is 2-bounded.)

5 Conclusions

In this paper, we investigate whether the underlying treatment of concurrency affects the meaning of simulation-based security. We give an affirmative answer, based on two examples showing that UC security under sequential scheduling is incomparable with UC security under oblivious scheduling.

This extends the analysis of Datta et al. [DKM$^+$04,DKMR05] with a new dimension, namely, the scheduling of concurrent processes. In fact, our separation result is of a slightly different character: rather than proving one definition is stronger/weaker than another, we show that the *same* definition has different meanings. This separation applies not only to security definitions (involving adversary and simulator), but also to the underlying notion of indistinguishability.

Our results seemingly contradict the common understanding that sequential and non-sequential scheduling schemes can, to a large extent, emulate each other. This is not a real contradiction, because indistinguishability and security definitions are never given with a layer of emulation. For example, one can emulate oblivious scheduling in a sequential framework by adding a scheduler machine and specifying all other machines in such a way that activation

only takes place via the scheduler machine. However, this pattern of emulation is *not* adopted when security definitions are given in sequential frameworks (e.g., [Can01,BPW04,Küs06]). Therefore the existence of a scheduling emulation says little about how the meaning of a security definition changes with the underlying model of concurrency.

Aside from the issue of scheduling, we also consider the "forwarder" property of [DKM+04,DKMR05]. We observe that forwarders are definable in Task-PIOA and, as expected, strong simulatability is equivalent to UC security. This implies all three notions (i.e., strong simulatability, black-box simulatability and UC security) are equivalent in the Task-PIOA framework.

Unbounded forwarders are not definable if we impose *a priori* bounds on the length of task schedules. In that case, we claim that strong simulatability and UC security are no longer equivalent. Essentially, one can construct a protocol for which an unbounded simulator must be used in order to satisfy strong simulatability. We leave the details as future work.

References

[Bea91] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015, 2003. `http://eprint.iacr.org/`.

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

[Can95] R. Canetti. *Studies in Secure Multi-Party Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In Moni Naor, editor, *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society, 2001. Full version available on `http://eprint.iacr.org/2000/067`.

[CCK+05] 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. Cryptology ePrint Archive, Report 2005/452, 2005. `http://eprint.iacr.org/`.

[CCK+06a] 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-2006-060, CSAIL, MIT, Cambridge, MA, 2006. Submitted for journal publication. Most current version available at `http://theory.csail.mit.edu/~lcheung/papers/task-PIOA-TR.pdf`.

[CCK+06b] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Task-structured Probabilistic I/O Automata. In *Proceedings of the 8th International Workshop on Discrete Event Systems – WODES'2006*, pages 207–214. IEEE, 2006.

[CCK+06c] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Time-bounded Task-PIOAs: A framework for analyzing security protocols. In S. Dolev, editor, *Proceedings the 20th International Symposium on Distributed Computing (DISC 2006)*, volume 14167 of *LNCS*, pages 238–253. Springer, 2006. Invited Paper.

[DKM+04] Anupam Datta, Ralf Kuesters, John C. Mitchell, Ajith Ramanathan, and Vitaly Shmatikov. Unifying equivalence-based definitions of protocol security. In *Proceedings of ACM SIGPLAN and IFIP WG 1.7 4th Workshop on Issues in the Theory of Security*, April 2004.

[DKMR05] Anupam Datta, Ralf Kuesters, John C. Mitchell, and Ajith Ramanathan. On the relationships between notions of simulation-based security. In J. Kilian, editor, *Proceedings of Theory of Cryptography Conference*, volume 3378 of *LNCS*, pages 476–494. Springer, Feb. 2005. Full version available on `http://eprint.iacr.org/2006/153`.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - Crypto '90*, pages 77–93, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science Volume 537.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85)*, pages 291–304, 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 218–229. ACM Press, 1987.

[Küs06] R. Küsters. Simulation-Based Security with Inexhaustible Interactive Turing Machines. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW-19 2006)*, pages 309–320. IEEE Computer Society, 2006.

[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, San Francisco, 1998.

[MMS03] P. Mateus, J.C. Mitchell, and A. Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time calculus. In R. Amadio and D. Lugiez, editors, *Proceedings of CONCUR 2003 - Concurrency Theory*, volume 2761 of *LNCS*, pages 327–349, Marseille, France, 2003. Springer.

[MR91] S. Micali and P. Rogaway. Secure computation. In Joan Feigenbaum, editor, *Advances in Cryptology - Crypto '91*, pages 392–404, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 576.

[MRST06] John Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Computer Science*, 353:118–164, 2006.

[PW94] Birgit Pfitzmann and Michael Waidner. A general framework for formal notions of "secure" system. Technical report, Hildesheimer Informatik-Berichte 11/94, Institut fr Informatik, Universitt Hildesheim., 1994.

[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, Oakland, CA, May 2001. IEEE Computer Society.

A Time-Bounded Task-PIOAs

In order to carry out computational analysis, we restrict our attention to those task-PIOAs whose operations can be represented by a collection of Turing machines with bounded runtime. This is the time-bounded Task-PIOA framework introduced in [CCK+05,CCK+06c].

We assume a standard bit-string representation for various parts of a task-PIOA, including states, actions, transitions and tasks. Let $\mathbb{R}^{\geq 0}$ denote the set of nonnegative reals and let $b \in \mathbb{R}^{\geq 0}$ be given. A task-PIOA $\mathcal{A}$ is said to be *b-bounded* just in case: (i) the bit-string representation of every automaton part has length at most b; (ii) there is a Turing machine that decides whether a given representation of a candidate automaton part is indeed an automaton part, and this machine runs in time at most b; (iii) there is a Turing machine that, given a state and a task of $\mathcal{A}$, determines the next action in time at most b; (iv) there is a probabilistic Turing machine that, given a state and an action of $\mathcal{A}$, determines the next state of $\mathcal{A}$ in time at most b. Furthermore, all 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 linear in the sum of the original bounds. Similarly, the hiding operator changes the time bound by a linear factor. Proofs for these claims can be found in [CCK+05].

Finally, we say that a task schedule ρ is *b-bounded* if $|\rho| \leq b$, that is, ρ is finite and contains at most b tasks.

Task-PIOA Families We define families of task-PIOAs indexed by a security parameter k: a *task-PIOA family* $\overline{\mathcal{A}}$ is an indexed set $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$ of task-PIOAs. The notion of time bound is also expressed in terms of the security parameter; namely, given $b : \mathbb{N} \to \mathbb{R}^{\geq 0}$, we say that $\mathcal{A}$ is *b-bounded* if every $\mathcal{A}_k$ is $b(k)$ -bounded.

The notions of compatibility and parallel composition are defined pointwise. Results for composition and hiding extends easily from those for time-bounded task-PIOAs. Again, detailed statements can be found in [CCK+05].

Approximate Implementation Our approximate implementation relation compares acceptance probabilities of an environment (in the style of [Can01]), as opposed to trace distributions or views (in the style of [BPW04]). Let $\mathcal{A}$ be a closed task-PIOA with a special output action acc and let ρ be a task schedule for $\mathcal{A}$. The *acceptance probability* with respect to $\mathcal{A}$ and ρ is defined to be:

$$\mathbf{P}_{\mathsf{acc}}(\mathcal{A}, \rho) := \Pr[\beta \leftarrow_{\mathsf{R}} \mathsf{tdist}(\mathcal{A}, \rho) : \beta \text{ contains } \mathsf{acc}],$$

where $\beta \leftarrow_{\mathsf{R}} \mathsf{tdist}(\mathcal{A}, \rho)$ means β is drawn randomly form the trace distribution induced by the task schedule ρ on task-PIOA $\mathcal{A}$.

We assume that every environment has acc as an output. Now let $\mathcal{A}_1$ and $\mathcal{A}_2$ be comparable task-PIOAs and let $\epsilon, p \in \mathbb{R}^{\geq 0}$ and $q_1, q_2 \in \mathbb{N}$ be given.

(As a convention, we use variable p for automata time bounds and variable q for schedule length bounds.) We define $\mathcal{A}_1 \leq_{p,q_1,q_2,\epsilon} \mathcal{A}_2$ as follows: given any p-bounded environment Env for both $\mathcal{A}_1$ and $\mathcal{A}_2$ and any q_1-bounded task schedule ρ_1 for $\mathcal{A}_1 \| Env$, there is a q_2-bounded task schedule ρ_2 for $\mathcal{A}_2 \| Env$ such that

$$|\,\mathbf{P}_{\mathsf{acc}}(\mathcal{A}_1 \| Env, \rho_1) - \mathbf{P}_{\mathsf{acc}}(\mathcal{A}_2 \| Env, \rho_2)| \leq \epsilon.$$

In other words, from the perspective of a p-bounded environment, $\mathcal{A}_1$ and $\mathcal{A}_2$ "look almost the same" provided $\mathcal{A}_2$ can use q_2 many steps to emulate q_1 many steps of $\mathcal{A}_1$. We claim that $\leq_{p,q_1,q_2,\epsilon}$ is transitive and preserved under composition, with certain adjustments to errors and time bounds. Proofs can be found in [CCK+05].

The relation $\leq_{p,q_1,q_2,\epsilon}$ can be extended to task-PIOA families in the obvious way. Let $\overline{\mathcal{A}}_1 = \{(\mathcal{A}_1)_k\}_{k\in\mathbb{N}}$ and $\overline{\mathcal{A}}_2 = \{(\mathcal{A}_2)_k\}_{k\in\mathbb{N}}$ be *comparable* task-PIOA families, that is, they are pointwise comparable. Let $\epsilon, p : \mathbb{N} \to \mathbb{R}^{\geq 0}$ and $q_1, q_2 : \mathbb{N} \to \mathbb{N}$ be given. We say that $\overline{\mathcal{A}}_1 \leq_{p,q_1,q_2,\epsilon} \overline{\mathcal{A}}_2$ provided $(\mathcal{A}_1)_k \leq_{p(k),q_1(k),q_2(k),\epsilon(k)} (\mathcal{A}_2)_k$ for every k.

Restricting our attention to negligible error and polynomial time bounds, we obtain a generic version of approximate implementation, namely, $\leq_{\mathsf{neg,pt}}$. Formally, a function $\epsilon : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is said to be *negligible* if, for every constant $c \in \mathbb{R}^{\geq 0}$, there exists $k_0 \in \mathbb{N}$ 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 $\overline{\mathcal{A}}_1 \leq_{\mathsf{neg,pt}} \overline{\mathcal{A}}_2$ if: $\forall p\ \forall q_1\ \exists q_2\ \exists \epsilon\ \overline{\mathcal{A}}_1 \leq_{p,q_1,q_2,\epsilon} \overline{\mathcal{A}}_2$, where p, q_1, q_2 are polynomials and ϵ is a negligible function. Again, [CCK+05] contains proofs that $\leq_{\mathsf{neg,pt}}$ is transitive and preserved under composition and hiding.

B Time-Bounded Structures

Time-bounded structures are defined in a similar fashion as time-bounded task-PIOAs. First we need the notion of b-time recognizable set: given a set B of binary strings and $b \in \mathbb{R}^{\geq 0}$, we say that B is *b-time recognizable* if there is a probabilistic Turing machine M that

- decides, in time at most b, if a binary string a is in the set B and
- has a description with fewer than b bits according to some standard encoding.

If $\bar{B} = \{B_k\}_{k\in\mathbb{N}}$ is a family of sets of binary stings, we say that $\bar{B}$ is *polynomial time-recognizable* if there is a polynomial p such that every B_k is $p(k)$-time recognizable.

Now a structure $\Pi = (\mathcal{A}, EAct)$ is said to be *b-bounded* if $\mathcal{A}$ is b-bounded and the set $\langle EAct \rangle$ of the representations of actions in $EAct$ is b-time recognizable by some Turing machine M_{EAct}. For a family $\overline{\Pi}$ of structures and a function $b : \mathbb{N} \to \mathbb{R}^{\geq 0}$, we say that $\overline{\Pi}$ is *b-bounded* if Π_k is $b(k)$-bounded for every k. If $\overline{\Pi}$ is p-bounded for some polynomial p, then we say that $\overline{\Pi}$ is *polynomial time-bounded*.

We claim that, given a polynomial time-bounded family $\overline{\Pi}$ and a polynomial-time recognizable family $\bar{S}$ of sets of actions, the family $\mathsf{hide}(\overline{\Pi}, \bar{S})$ is again polynomial time-bounded. Moreover, the $\leq_{\mathsf{neg,pt}}$ relation is preserved by hiding.

C Task-PIOA Codes

Task-PIOA A_i and B_i

Signature
Input: $Hello_i$
Output: i

Tasks
$Out_i = \{i\}$

States
For A_i: $hello \in \{\bot, \top\}$, initially $\bot$
For B_i: none

Transitions:

$Hello_i$
Effect:
For A_i: $hello := \top$;
For B_i: $none$

i
Precondition:
For A_i: $hello = \top$; For B_i: $none$
Effect:
For A_i: $hello := \bot$; For B_i: $none$

Fig. 4. Code for Task-PIOAs $A(i)$ and $B(i)$

Task-PIOA Box_k and $(Box_s)_k$

Signature
Input: 0; 1
Output: $out(x), x \in \{0,1\}^k$; ok
Internal: $choose$

Tasks
$Out = \{out(*)\}$; $Ok = \{ok\}$

States
$xval \in \{0,1\}^k \cup \bot$, initially $\bot$
$input$, a buffer of at most k elements in $\{0,1\}$, initially empty

Transitions:

0
Effect:
if $input$ is not full then
add "0" to $input$

1
Effect:
if $input$ is not full then
add "1" to $input$

$choose$
Precondition:
$xval = \bot$
Effect:
$xval := random(\{0,1\}^k)$

$out(x)$
Precondition:
$x = xval \neq \bot$
$input \neq xval$
Effect:
none

ok
Precondition:
For Box_k: $input = xval \neq \bot$
For $(Box_s)_k$: $false$
Effect:
$none$

Fig. 5. Code for Task-PIOAs Box_k and $(Box_s)_k$