How to Model Bounded Computation in Long-Lived Systems

Ran Canetti[1], Ling Cheung[2], Dilsun Kaynar[3], Nancy Lynch[2], and Olivier Pereira[4]*

[1] IBM TJ Watson Research Center
[2] Massachusetts Institute of Technology
[3] Carnegie Mellon University
[4] Université catholique de Louvain

Abstract. In most interesting cases, the security of cryptographic protocols relies on the assumption that adversarial entities have limited computational power, and it is generally accepted that security degrades progressively over time. However, some cryptographic services (e.g., time-stamping services or digital archives) are long-lived in nature; that is, their lifetime need not be bounded by a polynomial. In such cases, it is impossible to guarantee security in the traditional sense: even information theoretically secure protocols can fail if the attacker is given sufficient run time.
This work proposes a new paradigm for long-lived computation, where computational restrictions are stated in terms of space and processing rates. In this setting, entities may live for an unbounded amount of real time, subject to the condition that only a polynomial amount of work can be done per unit real time. Moreover, the space used by these entities is allocated dynamically and must be polynomially bounded. We propose a key notion of approximate implementation, which is an adaptation of computational indistinguishability to the long-lived setting. We show that approximate implementation is preserved under polynomial parallel composition, and under exponential sequential composition. This provides core foundations for an exciting new area, namely, the analysis of long-lived cryptographic systems.

1 Introduction

Nearly all the systems defined and analyzed in cryptographic protocol research are *short-lived*. In these systems, protocol parties can execute only a bounded number of steps, after which the protocol concludes. Depending on the particular model, adversarial entities may perform certain pre- and/or post-computations. While the adversary may be unbounded in these additional phases, it must be bounded during protocol execution. It is typical that security degrades substantially (namely, polynomially) in the length of protocol execution.

* Contact author, `olivier.pereira@uclouvain.be`

In this paper, we address the problem of analyzing *long-lived* cryptographic services. In a long-lived system, protocol parties may be active for an unbounded amount of real time, subject to the condition that only a polynomial amount of work can be done per unit real time. Here the adversary's interaction with the system becomes unbounded, and the adversary may perform an unbounded number of computation steps during protocol execution. This renders traditional security notions insufficient: long-lived systems are not likely to be secure in the traditional sense, simply because the adversary has now unbounded run time. For example, given sufficient time, the adversary can launch a brute force attack on an encryption service by testing every key of the right length. Thus, we need a new notion of security that is independent of (or only mildly dependent on) the lifetime of the system.

As it turns out, the move to long-lived systems requires some non-trivial departures from standard cryptographic modeling. First and foremost, entities with unbounded run time cannot be modeled simply as *probabilistic polynomial time (PPT)* Turing machines. We resolve this issue by introducing real time explicitly in the Task-PIOA model [1] and by imposing computational restrictions in terms of *rates* (i.e., number of computation steps per unit real time).

Another interesting challenge is the restriction on space. In the traditional setting, space restrictions are trivial because a polynomially bounded entity can only access a polynomially bounded amount of space. In the long-lived setting, the issue of space warrants new considerations. For instance, we would like to model dynamic allocation of space, as new entities become invoked and old entities are terminated. This is achieved using the designation of variables: all variables of a dormant entity (either not yet invoked or already killed off) must be set to a special null value. Together with restrictions on the number of variables and the number of active entities, we are able to express the core idea of bounded space, even for systems consisting of an unbounded number of entities.[1]

Having appropriate restrictions on space and computation rates, we define an approximate implementation relation for long-lived systems. This can be thought of as a version of the traditional notion of *computational indistinguishability*, which is essential in almost all security definitions. Roughly speaking, two systems (*real* and *ideal*) are computationally indistinguishable if their behaviors appear the same to a computationally bounded environment. This definition is difficult to replicate in the long-lived setting, because an environment with exponential run time can distinguish the two systems by launching brute force attacks. Our solution is to weaken the environment, so that brute force attacks can be tolerated to some degree.[2] We do so by letting the environment begin its observation from some arbitrary time t and restricting our attention to a polynomial length execution starting from t. Quantifying over all possible t, we

[1] An obvious example is a sequential composition (in the temporal sense) of an unbounded number of entities, each of which uses a bounded amount of space.

[2] In Section 2, we will see a protocol that is designed to withstand complete failures of old cryptographic services.

capture the intuition that two long-lived systems are equivalent if an observer with bounded rate and space cannot make distinctions "fast enough".

Finally, we show that approximate implementation is preserved under polynomial parallel composition and under exponential sequential composition. The latter highlights the power of our model: we can formulate and prove properties of an exponential number of entities in a meaningful way.

Related Work In the past decades, the cryptography and concurrency communities have developed rigorous frameworks for modeling protocols, formulating security properties, and proving correctness (e.g., [6, 2, 10, 1, 5]). These models, however, concentrate on short-lived systems where system lifetime is comparable to the computational complexity of individual entities and to the level of security provided. In [9], Müller-Quade and Unruh study *long-term security* of cryptographic protocols. They consider adversaries that try to derive information from the protocol transcript *after* protocol conclusion. This work can be seen as orthogonal to ours, because it does not consider long-lived protocol execution as we do here. In particular, the adversary of [9] has polynomially bounded interactions with the protocol parties.

2 Digital Timestamping: A Motivating Example

A digital archive is an example of the type of long-lived systems we consider in this paper. Documents are typically stored along with some metadata, such as creation time and access logs. In such a system, it is desirable to preserve the integrity of information even when cryptographic primitives used in the past become vulnerable to newly-discovered attacks. In [3, 4], Haber and Kamat address the problem of content integrity in long-term digital archives, and describe a protocol based on extending the lifetime of existing timestamp certificates.

A digital timestamping scheme takes as input a document d at a specific time t, and produces a certificate c that can be used later to verify the existence of d at time t. The security requirement here is that timestamp certificates are difficult to forge. If these certificates are generated using a single digital signature scheme, one must expect the possibility that, at some point in the future, new technology makes it feasible to forge signatures for that particular scheme. In that case, the credibility of timestamps is lost. The protocol of Haber and Kamat solves this problem by obtaining a new digital timestamp c' for the pair (d, c) using a new signature scheme. If the old signature scheme is broken after the renewal of the certificate, then the new certificate c' still provides evidence that d existed at the time t stated in the original certificate c. Note that time is used in an essential way in this protocol. An adequate model of this protocol would need to take into consideration the lifetime of each cryptographic primitives. It is also necessary to express timing constraints, such as the fact that a certificate cannot be renewed if it has already expired.

The protocol of Haber and Kamat can be modeled quite naturally using the framework presented in this paper. We consider signature services that are updated on a regular basis; that is, we divide the time line into a sequence of intervals and each interval is associated with a unique signature service. A dispatcher component accepts various timestamp requests (e.g., issue, validate, and renew). Upon each timestamp request, the dispatcher sends signing/verification request(s) to the appropriate active signature service. The result is then forwarded to the source of the original timestamp request. The goal here is to show that the composition of the dispatcher and the signature services is indistinguishable from an "ideal" system, which consists of the same dispatcher composed with ideal signature functionalities.[3] This is precisely the scenario we address in our sequential composition theorem (cf. Section 7). The details of our modeling are beyond the scope of the present paper.

3 Task-PIOAs

We review the basics of the task-PIOA framework [1], which has a partial-information scheduling mechanism based on tasks. A task is a set of related actions (e.g., actions representing the same activity but with different parameters). We view tasks as basic units of events, both for real time scheduling and for imposing computational bounds (cf. Sections 4 and 5).

Notations Given a set S, let $\mathsf{Disc}(S)$ denote the set of discrete probability measures on S. For $s \in S$, the *Dirac* measure on s is the discrete probability measure that assigns probability 1 to s. It is denoted $\delta(s)$.

Let V be a set of variables. Each $v \in V$ is associated with a *(static) type* $\mathsf{type}(v)$, which is the set of all possible values of v. We assume that $\mathsf{type}(v)$ is countable and contains the special symbol $\perp$. A *valuation* s for V is a function mapping every $v \in V$ to a value in $\mathsf{type}(v)$. The set of all valuations for V is denoted $\mathit{val}(V)$. Given $V' \subseteq V$, a valuation s' for V' is sometimes referred to as a *partial valuation* for V. Observe that s' induces a (full) valuation $\iota_V(s')$ for V, by assigning $\perp$ to every $v \notin V'$. Finally, we define an equivalence relation $\equiv_{V'}$ on $\mathit{val}(V)$: $s_1 \equiv_{V'} s_2$ if and only if $s_1.v = s_2.v$ for all $v \in V'$.

PIOA We define a *probabilistic input/output automaton (PIOA)* to be a tuple $\mathcal{A} = \langle V, S, s^{\mathsf{init}}, I, O, H, \Delta \rangle$, where:
(i) V is a set of *state variables* and $S \subseteq \mathit{val}(V)$ is a set of *states*;
(ii) $s^{\mathsf{init}} \in S$ is the *initial* state;
(iii) I, O and H are countable and pairwise disjoint sets of actions, referred to as *input, output and hidden actions*, respectively;
(iv) $\Delta \subseteq S \times (I \cup O \cup H) \times \mathsf{Disc}(S)$ is a *transition relation.*
The set $\mathit{Act} := I \cup O \cup H$ is 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 $O \cup H$. Any sequence of external actions is called a *trace.* We write

[3] Unlike a real signature service, an ideal signature functionality does not allow any forgeries.

$s.v$ for the value of variable v in state s. An action a is *enabled* in a state s if $\langle s, a, \mu \rangle \in \Delta$ for some μ. We require that $\mathcal{A}$ satisfies the following conditions.

- **Input Enabling:** For every $s \in S$ and $a \in I$, a is enabled in s.
- **Transition Determinism:** For every $s \in S$ and $a \in A$, there is at most one $\mu \in \mathsf{Disc}(S)$ such that $\langle s, a, \mu \rangle \in \Delta$. We write $\Delta(s, a)$ for such μ, if it exists.
- **Initialization**: $s^{\mathsf{init}}.v = \bot$ for every $v \in V$.

Parallel composition for PIOAs is based on synchronization of shared actions. PIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to be *compatible* if $V_i \cap V_j = Act_i \cap H_j = O_i \cap O_j = \emptyset$ whenever $i \neq j$. In that case, we define their *composition* $\mathcal{A}_1 \| \mathcal{A}_2$ to be

$$\langle V_1 \cup V_2, S_1 \times S_2, \langle s_1^{\mathsf{init}}, s_2^{\mathsf{init}} \rangle, (I_1 \cup I_2) \setminus (O_1 \cup O_2), O_1 \cup O_2, H_1 \cup H_2, \Delta \rangle,$$

where Δ is the set of triples $\langle \langle s_1, s_2 \rangle, a, \mu_1 \times \mu_2 \rangle$ satisfying: (i) a is enabled in some s_i, and (ii) for every i, if $a \in A_i$, then $\langle s_i, a, \mu_i \rangle \in \Delta_i$, otherwise $\mu_i = \delta(s_i)$. It is easy to check that input enabling, transition determinism, and initialization are preserved under composition. Moreover, the definition of composition can be generalized to any finite number of components.

Task-PIOAs To resolve nondeterminism, we make use of the notion of tasks introduced in [7, 1]. Formally, a *task-PIOA* is a pair $\langle \mathcal{A}, \mathcal{R} \rangle$ where

(i) $\mathcal{A}$ is a PIOA with state variables V,
(ii) $\mathcal{R}$ is a partition of the locally-controlled actions of $\mathcal{A}$,

The equivalence classes in $\mathcal{R}$ are called *tasks*. For notational simplicity, we often omit $\mathcal{R}$ and refer to the task-PIOA $\mathcal{A}$. The following axioms are assumed.

- **Action Determinism:** For every state $s \in S$ and every task $T \in \mathcal{R}$, there is at most one action $a \in T$ that is enabled in s.
- **Output/Hidden Tasks:** The task partition $\mathcal{R}$ respects the output/hidden distinction. That is, no tasks contain both an output action and a hidden action. In that case, we may speak of *output* and *hidden* tasks.

Unless otherwise stated, terminologies are inherited from the PIOA setting. For instance, if some $a \in T$ is enabled in a state s, then T is said to be *enabled* in s.

Given compatible task-PIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we define their *composition* to be $\langle \mathcal{A}_1 \| \mathcal{A}_2, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle$ Note that $\mathcal{R}_1 \cup \mathcal{R}_2$ is an equivalence relation because compatibility requires disjoint sets of locally controlled actions. Moreover, it is easy to check that action determinism and output/hidden tasks are preserved under composition.

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 run of $\mathcal{A}$ as follows.

(i) From the start state s^{init}, we *apply* the first task T_1: due to action- and transition-determinism, T_1 specifies at most one transition from s^{init}; if such a transition exists, it is taken, otherwise nothing happens.
(ii) Repeat with remaining T_i's.

Such a run gives rise to a unique *probabilistic execution*, which is a probability distribution over execution paths in $\mathcal{A}$. A state s is said to be *reachable* under τ if a path ending in s has non-zero probability. Moreover, this induces a unique *trace distribution* $\mathsf{tdist}(\mathcal{A}, \tau)$, which is a probability distribution over the set of traces of $\mathcal{A}$. We refer to [1] for more details on these constructions.

4 Real Time Constraints

Recall that our goal is to model entities with unbounded life time but bounded processing rates. A natural approach is to introduce real time, so that computational restrictions can be stated in terms of the number of steps performed per unit real time. However, we must make sure that real time information does not appear as part of the logical state, because bounded entities cannot access real time to arbitrary precision. Fortunately, it is sufficient to include real time information in task schedules only. For components that make decision based on time, discrete approximations are used.

A *timed* task schedule τ for a closed task-PIOA $\langle \mathcal{A}, \mathcal{R} \rangle$ is a finite or infinite sequence $\langle T_1, t_1 \rangle, \langle T_2, t_2 \rangle, \ldots$ such that: $T_i \in \mathcal{R}$ and $t_i \in \mathbb{R}_{\geq 0}$ for every i, and $t_1, t_2, \ldots$ is non-decreasing. The *limit time*, denoted $\mathsf{ltime}(\tau)$, is defined as follows.

– If τ is empty, then $\mathsf{ltime}(\tau) := 0$.

– If $t_1, t_2, \ldots$ is bounded, then $\mathsf{ltime}(\tau) := \lim_{i \to \infty} t_i$, otherwise $\mathsf{ltime}(\tau) := \infty$.

Moreover, given an interval $I \subseteq \mathbb{R}_{\geq 0}$, we say that τ is a timed task schedule for I if $t_i \in I$ for every index i.

Following [8], we associate lower and upper real time bounds to each task. In addition, we impose a rate bound that limits the number of times a task may occur within an interval of time. A burst bound is also included to give more modeling flexibility. Formally, a *bound map* for a task-PIOA $\langle \mathcal{A}, \mathcal{R} \rangle$ is a tuple $\langle \mathsf{rate}, \mathsf{burst}, \mathsf{lb}, \mathsf{ub} \rangle$ such that: (i) $\mathsf{rate}, \mathsf{burst} : \mathcal{R} \to \mathbb{R}_{\geq 0}$, (ii) $\mathsf{lb} : \mathcal{R} \to \mathbb{R}_{\geq 0}$, (iii) $\mathsf{ub} : \mathcal{R} \to \mathbb{R}_{\geq 0}^{\infty}$, and (iv) for all $T \in \mathcal{R}$, $\mathsf{lb}(T) \leq \mathsf{ub}(T)$. To ensure that rate and ub can be satisfied simultaneously, we require $\mathsf{rate}(T) \geq 1/\,\mathsf{ub}(T)$ whenever $\mathsf{rate}(T) \neq 0$ and $\mathsf{ub}(T) \neq \infty$. From this point on, we assume that every task-PIOA is associated with a particular bound map.

Given a timed schedule τ and a task T, let $\mathsf{proj}_T(\tau)$ denote the result of removing all pairs $\langle T_i, t_i \rangle$ with $T_i \neq T$. We say that τ is *valid* for a bound map $\langle \mathsf{rate}, \mathsf{burst}, \mathsf{lb}, \mathsf{ub} \rangle$ if the following hold for every task T.

(i) If $\mathsf{lb}(T) > 0$, then consecutive appearances of T must be at least $\mathsf{lb}(T)$ apart. Formally: (a) if $\langle T, t \rangle$ is the first element of $\mathsf{proj}_T(\tau)$, then $t \geq \mathsf{lb}(T)$; (b) for every interval I of length $d < \mathsf{lb}(T)$, $\mathsf{proj}_T(\tau)$ contains at most one element $\langle T, t \rangle$ with $t \in I$.

(ii) If $\mathsf{ub}(T) \neq \infty$, then consecutive appearances of T must be at most $\mathsf{ub}(T)$ apart. Formally: for every interval $I \subseteq [0, \mathsf{ltime}(\tau)]$ of length $d > \mathsf{ub}(T)$, $\mathsf{proj}_T(\tau)$ contains at least one element $\langle T, t \rangle$ with $t \in I$.

(iii) For any interval I of length d, $\mathsf{proj}_T(\tau)$ contains at most $\mathsf{rate}(T) \cdot d + \mathsf{burst}(T)$ elements $\langle T, t \rangle$ with $t \in I$.

Note that every timed schedule τ projects to an untimed schedule ρ by removing all real time information t_i, thereby inducing a trace distribution of $\mathcal{A}$. The set of trace distributions induced by all valid timed schedules for $\mathcal{A}$ and $\langle \mathsf{rate}, \mathsf{burst}, \mathsf{lb}, \mathsf{ub} \rangle$ is denoted $\mathsf{TrDists}(\mathcal{A}, \mathsf{rate}, \mathsf{burst}, \mathsf{lb}, \mathsf{ub})$. Since the bound map is typically fixed, we often omit it and write $\mathsf{TrDists}(\mathcal{A})$.

The new bound map for $\mathcal{A}_1 \| \mathcal{A}_2$is obtained by taking the union of component bound maps: $\langle \mathsf{rate}_1 \cup \mathsf{rate}_2, \mathsf{burst}_1 \cup \mathsf{burst}_2, \mathsf{lb}_1 \cup \mathsf{lb}_2, \mathsf{ub}_1 \cup \mathsf{ub}_2 \rangle$. This is well defined since the task partition of $\mathcal{A}_1 \| \mathcal{A}_2$ is $\mathcal{R}_1 \cup \mathcal{R}_2$.

5 Complexity Bounds

Intuitively, we envision a large collection of task-PIOAs that runs for an unbounded amount of real time. While the number of task-PIOAs in this collection is large, only a bounded (maybe even constant) number of them will be active simultaneously at any given point in time. This holds for systems in which some task-PIOAs may run for a bounded amount of time and then die off, while some others are yet to be invoked. Each task-PIOA has bounded memory and bounded computation rates, therefore the overall collection of task-PIOAs should also satisfy these conditions.

We propose a notion of step bounds that captures these intuitions. Roughly speaking, step bounds limit the amount computation involved in executing a single action, as well as the amount of space that is allocated as a result of that action. Paired with the rate and burst bounds of Section 4, we obtain a notion of bounded space and bounded computation rates.

Step Bound We assume some standard bit string encoding for Turing machines and for the names of variables, actions, and tasks. We also assume that variable valuations are encoded in the obvious way, as a list of name/value pairs. Let $\mathcal{A}$ be a task-PIOA with variable set V. Given state s, let $\hat{s}$ denote the partial valuation obtained from s by removing all pairs of the form $\langle v, \bot \rangle$. We have $\iota_V(\hat{s}) = s$, therefore no information is lost by reducing s to $\hat{s}$. This key observation allows us to represent a "large" valuation s with a "condensed" partial valuation $\hat{s}$.

Let $p \in \mathbb{N}$ be given. We say that $\mathcal{A}$ has *step bound* p if the following hold.

(i) For every variable $v \in V$, $\mathsf{type}(v) \subseteq \{0,1\}^p$.

(ii) The name of every action, task, and variable of $\mathcal{A}$ has length at most p.

(iii) There exists a deterministic Turing machine M_{enable} satisfying: for every reachable state s, M_{enable} on input $\hat{s}$ outputs the list of tasks enabled in s.

(iv) There exists a probabilistic Turing machine $M_{\mathcal{R}}$ satisfying: for every reachable state s and task T, $M_{\mathcal{R}}$ on input $\langle \hat{s}, T \rangle$ decides whether T is enabled in s. If so, $M_{\mathcal{R}}$ computes and outputs a new partial valuation $\hat{s}'$, along with the unique $a \in T$ that is enabled in s. The distribution on $\iota_V(\hat{s}')$ coincides with $\Delta(s, a)$.

(v) There exists a probabilistic Turing machine M_I satisfying: for every reachable state s and action a, M_I on input $\langle \hat{s}, a \rangle$ decides whether a is an input action of $\mathcal{A}$. If so, M_I computes a new partial valuation $\hat{s}'$. The distribution on $\iota_V(\hat{s}')$ coincides with $\Delta(s, a)$.

(vi) The encoding of M_{enable} is at most p bits long, and M_{enable} terminates after at most p steps on every input. The same hold for $M_{\mathcal{R}}$ and M_I.

Thus, step bound p limits the size of action names, which often represent protocol messages. It also limits the number of tasks enabled from any reachable state (Condition (iii)) and the complexity of individual transitions (Conditions (iv) and (v)). However, p does not limit how quickly transitions take place. That will be restricted by the rate and burst bounds of Section 4.

Lemma 1 below guarantees that a task-PIOA with step bound p will never reach a state in which more than p variables have non-$\bot$ values. The proof follows a simple inductive argument. Lemma 2 says that, when we compose task-PIOAs in parallel, the complexity of the composite is proportional to the sum of the component complexities. The proof is similar to that of [1, Lemma 4.2].

Lemma 1. *Let $\mathcal{A}$ be a task-PIOA with step bound p. For every valid timed task schedule τ and every state s reachable under τ, there are at most p variables v such that $s.v \neq \bot$.*

Proof. By the definition of task-PIOAs, we have $s^{\mathsf{init}}.v = \bot$ for every v. For every other reachable state s', let s be the state immediately preceding s' on an execution path leading to s'. Thus s is reachable. If the transition from s to s' is locally controlled, we use the fact that $M_{\mathcal{R}}$ always terminates after at most p steps, therefore every possible output, including $\hat{s}'$, has length at most p. This implies $\hat{s}'$ is a partial valuation on at most p variables. If the transition from s to s' is an input, we follow the same argument with M_I. $\square$

Lemma 2. *Suppose we have a compatible set $\{\mathcal{A}_i | 1 \leq i \leq b\}$ of task-PIOAs, where each A_i has step bound $p_i \in \mathbb{N}$. The composition $\|_{i=1}^{b} \mathcal{A}_i$ has step bound $c_{\mathsf{comp}} \cdot \sum_{i=1}^{b} p_i$, where c_{comp} is a fixed constant.*

Turing Machine Simulation Given a closed (i.e., no input actions) task-PIOA $\mathcal{A}$ with step bound p, one can define a nondeterministic Turing machine $M_{\mathcal{A}}$ that simulates the execution of $\mathcal{A}$. The amount of work tape needed by $M_{\mathcal{A}}$ is polynomial in p. As a convention, we write s for the current state and s' for the next state after a transition. Recall that $\hat{s}$ denotes the partial valuation obtained from s by removing all pairs of the form $\langle v, \bot \rangle$. $M_{\mathcal{A}}$ maintains this partial valuation on its work tape. The following procedure is repeated indefinitely by $\mathcal{A}$.

(i) From state s, $M_{\mathcal{A}}$ gives to M_{enable} the partial valuation $\hat{s}$ currently stored on the work tape. At the initial state s^{init}, all variables have the default value $\bot$, thus $\hat{s}$ is the empty valuation. This corresponds to an initially empty work tape.

(ii) The run stops if M_{enable} outputs nothing. Otherwise, a task T is chosen nondeterministically from the output of M_{enable} and $\langle \hat{s}, T \rangle$ is given to $M_{\mathcal{R}}$.

(iii) $M_{\mathcal{R}}$ returns $\langle \hat{s}', a \rangle$. $M_{\mathcal{A}}$ checks every variable v appearing in $\hat{s}'$: if v appears in $\hat{s}$, then the value of v is updated on the work tape; otherwise, $M_{\mathcal{A}}$ allocates enough space to store the name of v and the value $\hat{s}'(v)$. Finally, $M_{\mathcal{A}}$ checks for all variables v appear in $\hat{s}$ but not $\hat{s}'$. The storage for those variables is freed.

Since the name and type of every variable are also bounded by p, we can infer from Lemma 1 that the space needed to represent a reachable state is polynomial in p (in fact, on the order of p^2). Moreover, the amount of work tape needed by M_{enable} and $M_{\mathcal{R}}$ is on the order of p, because these Turing machines execute at most p steps at each activation.[4] Therefore, the total amount of work tape needed by $M_{\mathcal{A}}$ is polynomial in p.

[4] Note that we are not concerned with M_I here, because $\mathcal{A}$ is assumed to be closed.

Overall Bound Putting the various bounds together, we have: for $p \in \mathbb{N}$, $\mathcal{A}$ is said to be *p-bounded* if (i) $\mathcal{A}$ has step bound p, (ii) and each of $\mathsf{rate}(T)$ and $\mathsf{burst}(t)$ is at most p for every task T.

Task-PIOA Families We now extend our definitions to task-PIOA families, indexed by a *security parameter* k. More precisely, a *task-PIOA family* $\bar{\mathcal{A}}$ is an indexed set $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$ of task-PIOAs. Given $p : \mathbb{N} \to \mathbb{N}$, we say that $\bar{\mathcal{A}}$ is *p-bounded* just in case: for all k, $\mathcal{A}_k$ is $p(k)$-bounded. If p is a polynomial, then we say that $\bar{\mathcal{A}}$ is *polynomially bounded.* Finally, the notions of compatibility and parallel composition for task-PIOA families are defined pointwise.

We remark that our notion of closed, polynomially bounded families is reminiscent of the traditional notion of PSPACE (which is equivalent to nondeterministic PSPACE). Our setting is slightly richer, because we also talk about rates of computation with respect to real time. Thus, we can distinguish machines that compute in large bursts from those that compute at a steady rate.

6 Long-Term Implementation Relation

Much of modern cryptography is based on the notion of computational indistinguishability. For instance, an encryption algorithm is (chosen-plaintext) secure if the ciphertexts of two distinct but equal-length messages are indistinguishable from each other, even if the plaintexts are generated by the distinguisher itself. The key assumption is that the distinguisher is computationally bounded, so that it cannot launch a brute force attack. In this section, we adapt this notion of indistinguishability to the long-lived setting.

As in [1], we define an implementation relation based on closing environments and acceptance probabilities. Let $\mathcal{A}$ be a closed task-PIOA with output action acc and task $\{\mathsf{acc}\}$. Let τ be a timed task schedule for $\mathcal{A}$. The *acceptance probability* of $\mathcal{A}$ under τ is: $\mathbf{P}_{\mathsf{acc}}(\mathcal{A}, \tau) := \Pr[\beta \text{ contains } \mathsf{acc} : \beta \leftarrow_{\mathsf{R}} \mathsf{tdist}(\mathcal{A}, \tau)]$; that is, the probability that a trace drawn from the distribution $\mathsf{tdist}(\mathcal{A}, \tau)$ contains the action acc. For $\mathcal{A}$ that is not necessarily closed, we include a closing environment. A task-PIOA Env is an *environment* for $\mathcal{A}$ if it is compatible with $\mathcal{A}$ and $\mathcal{A}\|\mathsf{Env}$ is closed. Throughout this paper, we assume that every environment has output action acc. In this way, we may speak of acceptance probabilities of $\mathcal{A}\|\mathsf{Env}$.

In the short-lived setting, we say that a system $\mathcal{A}_1$ implements another system $\mathcal{A}_2$ if every run of $\mathcal{A}_1$ can be "matched" by a run of $\mathcal{A}_2$ such that no probabilistic polynomial time environment can distinguish the two runs. As we discussed in the introduction, this type of definition is too strong for the long-lived setting, because we must allow environments with unbounded total run time (as long as they have bounded rate and space). For example, consider the timestamping protocol of [4, 3] (cf. Section 2). After running for a long period of real time, a distinguisher environment may be able to forge signatures from a much earlier time period. As a result, it can distinguish the real system from the ideal one in the traditional sense. However, the essence of the protocol is that such failures can in fact be tolerated, because the environment still cannot forge *recent* signatures.

This timestamping example suggests that we need a new notion of indistinguishability that "ignores" a large part of the execution history. That is, even when an attacker succeeds in breaking old cryptographic services, the system remains secure as long as the attacker cannot break the services used in recent history. Our new implementation relation aims to capture this intuition.

First we give some informal explanations. Let Env be an environment for both $\mathcal{A}_1$ and $\mathcal{A}_2$ and let $t \in \mathbb{R}_{\geq 0}$ be given. We set up two experiments as follows. In the first experiment, Env interacts with $\mathcal{A}_1$ according to some task schedule τ_1. In the second experiment, Env interacts with $\mathcal{A}_1$ according to τ_1 during the interval $[0, t]$. At time t, $\mathcal{A}_1$ is replaced by $\mathcal{A}_2$ and τ_1 is replaced by some task schedule τ_2 for $\mathsf{Env} \| \mathcal{A}_2$. Our implementation relation requires that, for every switching time t, the difference in acceptance probability in these two experiments must be small. In other words, the environment cannot tell whether $\mathcal{A}_1$ has been replaced by $\mathcal{A}_2$, regardless of the system history.

To make sure that such a substitution is well defined, we need to impose a number of *comparability* conditions on task-PIOAs $\mathcal{A}_1$ and $\mathcal{A}_2$.

(i) $\mathcal{A}_1$ and $\mathcal{A}_2$ must have the same external interface: $I_1 = I_2$ and $O_1 = O_2$. In this case, every environment E for $\mathcal{A}_1$ is also and environment for $\mathcal{A}_2$, provided E is compatible with $\mathcal{A}_2$.

(ii) There must be a *switch function* $f : S_{\mathcal{A}_1} \to S_{\mathcal{A}_2}$ that preserves initial state and reachability. When we replace $\mathcal{A}_1$ with $\mathcal{A}_2$ in the second experiment, we shall initialize $\mathcal{A}_2$ to the state $f(s_1)$, where s_1 is the final state of $\mathcal{A}_1$ at time t. Intuitively, f initializes $\mathcal{A}_2$ with enough information about the computation history between $\mathcal{A}_1$ and Env.

(iii) $\mathcal{A}_1$ and $\mathcal{A}_2$ have the same set of output tasks and the bound maps coincide on every output task.

(iv) Every hidden task of $\mathcal{A}_1$ either coincides with a hidden task of $\mathcal{A}_2$, or is disjoint from every hidden task of $\mathcal{A}_2$. In the first case, the bound maps must coincide.

Given $t \in \mathbb{R}_{>0}$ and comparable $\mathcal{A}_1$ and $\mathcal{A}_2$, we define $\mathsf{sub}(\mathcal{A}_1, \mathcal{A}_2, f, t)$ to be the automaton obtained by replacing $\mathcal{A}_1$ with $\mathcal{A}_2$ at time t, using the switch function f to transform states of $\mathcal{A}_1$ into states of $\mathcal{A}_2$. The precise definition of $\mathsf{sub}(\mathcal{A}_1, \mathcal{A}_2, f, t)$ is as follows.

– The variable space is $\{\mathit{flag}\} \cup V_{\mathcal{A}_1} \cup V_{\mathcal{A}_2}$, where *flag* is a fresh variable of type $\{\bot, \top\}$ and all other types are inherited from $V_{\mathcal{A}_1}$ and $V_{\mathcal{A}_2}$.
– The state space is $(\langle \mathit{flag}, \bot \rangle \times S_{\mathcal{A}_1} \times s_2^{\mathsf{init}}) \cup (\langle \mathit{flag}, \top \rangle \times s_1^{\mathsf{init}} \times S_{\mathcal{A}_2})$.
– The action alphabet is $\mathit{Act}_{\mathcal{A}_1} \cup \mathit{Act}_{\mathcal{A}_2} \cup \{\mathsf{switch}\}$, where switch is a fresh hidden action and the input/output/hidden classifications remain the same for $\mathit{Act}_{\mathcal{A}_1}$ and $\mathit{Act}_{\mathcal{A}_2}$.
– The task partition is $\mathcal{R}_{\mathcal{A}_1} \cup \mathcal{R}_{\mathcal{A}_2} \cup \{\{\mathsf{switch}\}\}$.
– The rate bound is $\mathsf{rate}_{\mathcal{A}_1} \cup \mathsf{rate}_{\mathcal{A}_2} \cup \{\langle \{\mathsf{switch}\}, \frac{1}{t} \rangle\}$.
– The burst bound is $\mathsf{burst}_{\mathcal{A}_1} \cup \mathsf{burst}_{\mathcal{A}_2} \cup \{\langle \{\mathsf{switch}\}, 0 \rangle\}$.
– The lower bound is $\mathsf{lb}_{\mathcal{A}_1} \cup \mathsf{lb}_{\mathcal{A}_2} \cup \{\langle \{\mathsf{switch}\}, t \rangle\}$.
– The upper bound is $\mathsf{ub}_{\mathcal{A}_1} \cup \mathsf{ub}_{\mathcal{A}_2} \cup \{\langle \{\mathsf{switch}\}, t \rangle\}$.
– The transition relation is $\Delta'_{\mathcal{A}_1} \cup \Delta'_{\mathcal{A}_2} \cup \Delta'_3$, where

- $\Delta'_{\mathcal{A}_1}$ is induced by $\Delta_{\mathcal{A}_1}$ and the injection $S_{\mathcal{A}_1} \to (\langle \mathit{flag}, \bot\rangle \times S_{\mathcal{A}_1} \times s_2^{\mathsf{init}})$;
- similarly for $\Delta'_{\mathcal{A}_2}$;
- $\Delta'_3 = \{\langle\langle \mathit{flag}, \bot\rangle \times s_1 \times s_2^{\mathsf{init}}\rangle, \mathsf{switch}, \delta(\langle\langle \mathit{flag}, \top\rangle \times s_1^{\mathsf{init}} \times f(s_1)\rangle)\rangle | s_1 \in S_{\mathcal{A}_1}\}$.

Note that the bound map is well defined due to Assumptions (iii) and (iv) above. Moreover, even though the task $\{\mathsf{switch}\}$ is scheduled more than once, the action switch occurs at most once, namely, at time t.

We are now ready to define our implementation relation formally.

Definition 1. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be comparable task-PIOAs. Let $p, q \in \mathbb{N}$ and $\epsilon \in \mathbb{R}_{\geq 0}$ be given. We say that $\mathcal{A}_1 \leq_{p,q,\epsilon} \mathcal{A}_2$ if there exists a switch function $f : S_{\mathcal{A}_1} \to S_{\mathcal{A}_2}$ such that the following holds for every $t \in \mathbb{R}_{>0}$: given a p-bounded environment Env and a valid timed schedule τ_1 for $\mathcal{A}_1 \| \mathsf{Env}$ for the interval $[0, t+q]$, there exists valid timed schedule τ_2 for $\mathsf{sub}(\mathcal{A}_1, \mathcal{A}_2, f, t) \| \mathsf{Env}$ for the interval $[0, t+q]$ such that $|\, \mathbf{P}_{\mathsf{acc}}(\mathcal{A}_1 \| \mathsf{Env}, \tau_1) - \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}_1, \mathcal{A}_2, f, t) \| \mathsf{Env}, \tau_2)| \leq \epsilon$.*

The relation $\leq_{p,q,\epsilon}$ can be extended to task-PIOA families as follows. Let $\bar{\mathcal{A}}_1 = \{(\mathcal{A}_1)_k\}_{k \in \mathbb{N}}$ and $\bar{\mathcal{A}}_2 = \{(\mathcal{A}_2)_k\}_{k \in \mathbb{N}}$ be (pointwise) comparable task-PIOA families. Given $\epsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ and $p, q : \mathbb{N} \to \mathbb{N}$, we say that $\bar{\mathcal{A}}_1 \leq_{p,q,\epsilon} \bar{\mathcal{A}}_2$ just in case $(\mathcal{A}_1)_k \leq_{p(k),q(k),\epsilon(k)} (\mathcal{A}_2)_k$ for every k.

Restricting our attention to negligible error and polynomial time bounds, we obtain the implementation relation $\leq_{\mathsf{neg,pt}}$. Formally, a function $\epsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is said to be *negligible* if, for every constant $c \in \mathbb{N}$, there exists $k_0 \in \mathbb{N}$ such that $\epsilon(k) < \frac{1}{k^c}$ for all $k \geq k_0$. (That is, ϵ diminishes more quickly than the reciprocal of any polynomial.) Given task-PIOA families $\bar{\mathcal{A}}_1$ and $\bar{\mathcal{A}}_2$, we say that $\bar{\mathcal{A}}_1 \leq_{\mathsf{neg,pt}} \bar{\mathcal{A}}_2$ if $\forall p, q\ \exists \epsilon\ \bar{\mathcal{A}}_1 \leq_{p,q,\epsilon} \bar{\mathcal{A}}_2$, where p, q are polynomials and ϵ is a negligible function.

7 Composition Theorems

In practice, cryptographic services are seldom used in isolation. More typically, different types of services operate in conjunction, interacting with each other and with multiple protocol participants. For example, a participant may submit a document to an encryption service to obtain a ciphertext, which is later submitted to a timestamping service. In such situations, it is important that the services are provably secure even in the context of composition.

In this section, we prove two types of composition theorems. The first concerns *parallel composition*, which is a combination of services that are active at the same time and may interact with each other. Given a polynomially bounded collection of real services such that each real service implement some ideal service, the parallel composition of the real services is guaranteed to implement the parallel composition of the ideal services.

Our second main theorem deals with *sequential composition*, that is, a combination of services that are active in succession. The interaction between two distinct services is much more limited in this setting, because one of them must have finished execution before the other one comes online. An example of such a collection is the signature services in the timestamping algorithm of [4, 3], where each service is replaced by the next one at regular intervals.

As in the parallel case, we prove that the sequential composition of real services implements the sequential composition of ideal services. Surprisingly, we can relax the polynomial restriction on the size of the composition to exponential.[5] This highlights a unique feature of our implementation relation: essentially, we walk down the real time line and, at any time point t, we consider a polynomial length interval around t.

Parallel Composition First, we show that the relation $\leq_{p,q,\epsilon}$ (cf. Definition 1) is preserved under polynomial composition, with some appropriate adjustment to the environment complexity bound and to the error in acceptance probability.

Theorem 1. *Let $b \in \mathbb{N}$ be given and, for each $1 \leq i \leq b$, let $\mathcal{A}_i^1$ and $\mathcal{A}_i^2$ be comparable task-PIOAs. Suppose there exists a non-decreasing function $r : \mathbb{N} \to \mathbb{N}$ such that, for all i, both $\mathcal{A}_i^1$ and $\mathcal{A}_i^2$ are $r(i)$-bounded. Suppose further that $\mathcal{A}_1^{\alpha_1}, \ldots, \mathcal{A}_b^{\alpha_b}$ are pairwise compatible for any combination of $\alpha_i \in \{1, 2\}$.*

Let $p, p', q \in \mathbb{N}$ and $\epsilon, \epsilon' \in \mathbb{R}_{\geq 0}$ be given, and assume the following.

(1) $p = c_{\mathsf{comp}} \cdot (b \cdot r(b) + p')$, where c_{comp} is the constant factor for composing task-PIOAs in parallel.

(2) $\epsilon' = b \cdot \epsilon$.

(3) For all i, $\mathcal{A}_i^1 \leq_{p,q,\epsilon} \mathcal{A}_i^2$.

Then we have $\|_{i=1}^b \mathcal{A}_i^1 \leq_{p',q,\epsilon'} \|_{i=1}^b \mathcal{A}_i^2$.

Proof. For each i, choose a function $f_i : S_{\mathcal{A}_i^1} \to S_{\mathcal{A}_i^2}$ according to the assumption that $\mathcal{A}_i^1 \leq_{p,q,\epsilon} \mathcal{A}_i^2$. Let f denote $f_1 \times \ldots \times f_b : \prod_{i=1}^b S_{\mathcal{A}_i^1} \to \prod_{i=1}^b S_{\mathcal{A}_i^2}$.

Let $t \in \mathbb{R}_{\geq 0}$ be given. Let Env be a p'-bounded environment and let τ_0 be a valid timed task schedule for $\|_{i=1}^b \mathcal{A}_i^1 \| \mathsf{Env}$ for the interval $[0, t+q]$. Let Env_1 denote $\|_{i=2}^b \mathcal{A}_i^1 \| \mathsf{Env}$, which is an environment for $\mathcal{A}_1^1$ and $\mathcal{A}_1^2$. Note that Env_1 is p-bounded. By assumption, we may choose a valid timed task schedule τ_1 for $\mathsf{sub}(\mathcal{A}_1^1, \mathcal{A}_1^2, f_1, t) \| \mathsf{Env}_1$ for the interval $[0, t+q]$ such that

$$| \mathbf{P}_{\mathsf{acc}}(\mathcal{A}_1^1 \| \mathsf{Env}_1, \tau_0) - \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}_1^1, \mathcal{A}_1^2, f_1, t) \| \mathsf{Env}_1, \tau_1)| \leq \epsilon.$$

Now let $\mathsf{Env}_2 := \mathsf{sub}(\mathcal{A}_1^1, \mathcal{A}_1^2, f_1, t) \| \mathcal{A}_3^1 \| \ldots \| \mathcal{A}_b^1 \| \mathsf{Env}$ and note that Env_2 is also p-bounded. Since $\mathcal{A}_2^1 \leq_{p,q,\epsilon} \mathcal{A}_2^2$. we may choose a valid timed task schedule τ_2 for $\mathsf{sub}(\mathcal{A}_2^1, \mathcal{A}_2^2, f_2, t) \| \mathsf{Env}_2$ for the interval $[0, t+q]$ such that

$$| \mathbf{P}_{\mathsf{acc}}(\mathcal{A}_2^1 \| \mathsf{Env}_2, \tau_1) - \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}_2^1, \mathcal{A}_2^2, f_2, t) \| \mathsf{Env}_2, \tau_2)| \leq \epsilon.$$

Repeating the same argument, we obtain a valid timed task schedule τ_b for $(\|_{i=1}^b \mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t)) \| \mathsf{Env}$ for the interval $[0, t+q]$ such that

$$\begin{aligned} | \mathbf{P}_{\mathsf{acc}}((\|_{i=1}^{b-1} \mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t)) \| \mathcal{A}_b^1 \| \mathsf{Env}, \tau_{b-1}) \\ - \mathbf{P}_{\mathsf{acc}}((\|_{i=1}^b \mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t)) \| \mathsf{Env}, \tau_b)| \leq \epsilon. \end{aligned}$$

[5] In our formulation, it is not meaningful to exceed an exponential number of components, because the length of component descriptions is limited by the security parameter.

Now H_i denote $\mathsf{sub}(\mathcal{A}_1^1, \mathcal{A}_1^2, f_1, t)\| \dots \|\mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t)\|\mathcal{A}_{i+1}^1\| \dots \|\mathcal{A}_b^1$ for $i \in \{1, \dots, b\}$. We have:

$$\begin{aligned}
&|\,\mathbf{P}_{\mathsf{acc}}(\|_{i=1}^b \mathcal{A}_i^1\|\mathsf{Env}, \tau_0) - \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\|_{i=1}^b \mathcal{A}_i^1, \|_{i=1}^b \mathcal{A}_i^2, f, t)\|\mathsf{Env}, \tau_b)| \\
&\le |\,\mathbf{P}_{\mathsf{acc}}(\|_{i=1}^b \mathcal{A}_i^1\|\mathsf{Env}, \tau_0) - \mathbf{P}_{\mathsf{acc}}(H_1\|\mathsf{Env}, \tau_1)| + \dots \\
&\quad + |\,\mathbf{P}_{\mathsf{acc}}(H_i\|\mathsf{Env}, \tau_i) - \mathbf{P}_{\mathsf{acc}}(H_{i+1}\|\mathsf{Env}, \tau_{i+1})| + \dots \\
&\quad + |\,\mathbf{P}_{\mathsf{acc}}(H_{b-1}\|\mathsf{Env}, \tau_{b-1}) - \mathbf{P}_{\mathsf{acc}}(H_b\|\mathsf{Env}, \tau_b)| \\
&\le b \cdot \epsilon = \epsilon'.
\end{aligned}$$

Finally, we obtain from τ_b a valid schedule τ_b' for $\mathsf{sub}(\|_{i=1}^b \mathcal{A}_i^1, \|_{i=1}^b \mathcal{A}_i^2, f, t)\|\mathsf{Env}$ for the interval $[0, t+q]$, by removing all $\{\mathsf{switch}_i\}$ tasks and inserting $\{switch\}$ tasks as dictated by the bound map. It is easy to see that

$$\mathbf{P}_{\mathsf{acc}}((\|_{i=1}^b \mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t))\|\mathsf{Env}, \tau_b) = \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\|_{i=1}^b \mathcal{A}_i^1, \|_{i=1}^b \mathcal{A}_i^2, f, t)\|\mathsf{Env}, \tau_b').$$

This completes the proof that $\|_{i=1}^b \mathcal{A}_i^1 \le_{p',q,\epsilon'} \|_{i=1}^b \mathcal{A}_i^2$. □

Using Theorem 1, it is not hard to prove that $\le_{\mathsf{neg,pt}}$ (cf. Section 6) is also preserved under polynomial composition. (Recall from Section 6 that $\le_{\mathsf{neg,pt}}$ is a relation on task-PIOA families based on $\le_{p,q,\epsilon}$.)

Theorem 2 (Parallel Composition Theorem for $\le_{\mathsf{neg,pt}}$). *Let two sequences of task-PIOA families $\bar{\mathcal{A}}_1^1, \bar{\mathcal{A}}_2^1, \dots$ and $\bar{\mathcal{A}}_1^2, \bar{\mathcal{A}}_2^2, \dots$ be given, with $\bar{\mathcal{A}}_i^1$ comparable to $\bar{\mathcal{A}}_i^2$ for all i. Assume that $\bar{\mathcal{A}}_1^{\alpha_1}, \bar{\mathcal{A}}_2^{\alpha_2}, \dots$ are pairwise compatible for any combination of $\alpha_i \in \{1, 2\}$.*

Suppose there exist polynomials $r, s : \mathbb{N} \to \mathbb{N}$ such that, for all i, k, both $(\bar{\mathcal{A}}_i^1)_k$ and $(\bar{\mathcal{A}}_i^2)_k$ are bounded by $s(k) \cdot r(i)$. Assume that r is non-decreasing and

$$\forall p, q \ \exists \epsilon \ \forall i \ \bar{\mathcal{A}}_i^1 \le_{p,q,\epsilon} \bar{\mathcal{A}}_i^2, \tag{1}$$

where p, q are polynomials and ϵ is a negligible function. (This is a strengthening of the statement $\forall i \ \bar{\mathcal{A}}_i^1 \le_{\mathsf{neg,pt}} \bar{\mathcal{A}}_i^2$.) Let b be any polynomial. For each k, let $(\widehat{\mathcal{A}}^1)_k$ denote $(\bar{\mathcal{A}}_1^1)_k\| \dots \|(\bar{\mathcal{A}}_{b(k)}^1)_k$. Similarly for $(\widehat{\mathcal{A}}^2)_k$. Then we have $\widehat{\mathcal{A}}^1 \le_{\mathsf{neg,pt}} \widehat{\mathcal{A}}^2$.

Proof. By the definition of $\le_{\mathsf{neg,pt}}$, we need to prove the following: $\forall p', q \ \exists \epsilon' \ \widehat{\mathcal{A}}^1 \le_{p',q,\epsilon'} \widehat{\mathcal{A}}^2$, where p', q are polynomials and ϵ' is a negligible function. Let polynomials p' and q be given and define $p := c_{\mathsf{comp}} \cdot (b \cdot (r \circ b) + p')$, where c_{comp} is the constant factor for composing task-PIOAs in parallel. Now choose ϵ using p, q, and Assumption (1). Define $\epsilon' := b \cdot \epsilon$.

Let $k \in \mathbb{N}$ be given. We need to prove $(\widehat{\mathcal{A}}^1)_k \le_{p'(k),q(k),\epsilon'(k)} (\widehat{\mathcal{A}}^2)_k$. That is,

$$(\bar{\mathcal{A}}_1^1)_k\| \dots \|(\bar{\mathcal{A}}_{b(k)}^1)_k \le_{p'(k),q(k),\epsilon'(k)} (\bar{\mathcal{A}}_1^2)_k\| \dots \|(\bar{\mathcal{A}}_{b(k)}^2)_k.$$

For every i, we know that $(\bar{\mathcal{A}}_i^1)_k$ and $(\bar{\mathcal{A}}_i^2)_k$ are bounded by $(s(k) \cdot r)(i)$. Moreover, by the choice of ϵ, we have $(\bar{\mathcal{A}}_i^1)_k \le_{p(k),q(k),\epsilon(k)} (\bar{\mathcal{A}}_i^2)_k$ for all i. Therefore, we may apply Theorem 1 to conclude that $(\widehat{\mathcal{A}}^1)_k \le_{p'(k),q(k),\epsilon'(k)} (\widehat{\mathcal{A}}^2)_k$. This completes the proof. □

Sequential Composition We now treat the more interesting case, namely, unbounded sequential composition. The first challenge is to formalize the notion of sequentiality. On a syntactic level, all components in the collection are combined using the parallel composition operator. To capture the idea of successive invocation, we introduce some auxiliary notions.

Intuitively, we make a distinction between *active* and *dormant* entities. Active entities may perform actions and store information in memory. Dormant entities have no available memory and do not enable locally controlled actions.[6] A dormant entity behaves exactly like a "trivial" automaton, as in Definition 2 below. In Definition 3, we formalize the idea that an entity $\mathcal{A}$ may be invoked and terminated by some other entity $\mathcal{B}$. Then we show in Lemma 3 that $\mathcal{A}$ can be replaced by $\mathsf{trivial}(\mathcal{A})$ outside its active interval. Finally, we introduce sequentiality in Definition 4.

Definition 2. *Let $\mathcal{A}$ a task-PIOA. We define task-PIOA $\mathsf{trivial}(\mathcal{A})$ as follows: $\mathsf{trivial}(\mathcal{A})$ has the same action signature, task partition, and bound map as $\mathcal{A}$; $\mathsf{trivial}(\mathcal{A})$ has only one state, which does not enable any locally controlled actions.*

Definition 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be pairwise compatible task-PIOAs and let reals $t_1 \leq t_2$ be given. We say that $\mathcal{A}$ is* restricted to the interval $[t_1, t_2)$ by $\mathcal{B}$ *if:*

- *for any $t < t_1$, environment Env for $\mathcal{A}\|\mathcal{B}$, valid schedule τ for $\mathcal{A}\|\mathcal{B}\|\mathsf{Env}$ for $[0, t]$, and state s reachable under τ, no locally controlled actions of $\mathcal{A}$ is enabled in s, and $s.v = \bot$ for every variable v of $\mathcal{A}$.*
- *the same for all $t \geq t_2$.*

Lemma 3. *Suppose that $\mathcal{A}$ is restricted to the interval $[t_1, t_2)$ by $\mathcal{B}$. Let $t < t_1$, environment Env of $\mathcal{A}\|\mathcal{B}$, and valid task schedule τ for $\mathcal{A}\|\mathcal{B}\|\mathsf{Env}$ for the interval $[0, t]$ be given. Note that τ is also a valid task schedule for $\mathsf{trivial}(\mathcal{A})\|\mathcal{B}\|\mathsf{Env}$. Then $\mathbf{P}_{\mathsf{acc}}(\mathcal{A}\|\mathcal{B}\|\mathsf{Env}, \tau) = \mathbf{P}_{\mathsf{acc}}(\mathsf{trivial}(\mathcal{A})\|\mathcal{B}\|\mathsf{Env}, \tau)$.*

Proof. By Definition 3, the behavior of $\mathcal{A}$ before time t_1 is exactly the same as $\mathsf{trivial}(\mathcal{A})$. □

Definition 4 (Sequentiality). *Let $\mathcal{B}, \mathcal{A}_1, \mathcal{A}_2, \ldots$ be pairwise compatible task-PIOAs. We say that $\mathcal{A}_1, \mathcal{A}_2, \ldots$ are* sequential under $\mathcal{B}$ *if there exist reals $0 \leq t_1 < t_2 < \ldots$ such that: for all i, $\mathcal{A}_i$ is restricted to $[t_i, t_{i+1})$ by $\mathcal{B}$.*

For Lemma 4 and Theorem 3 below, let $\mathcal{A}_1^1, \mathcal{A}_2^1, \ldots$ and $\mathcal{A}_1^2, \mathcal{A}_2^2, \ldots$ be two sequences of task-PIOAs such that $\mathcal{A}_i^1$ and $\mathcal{A}_i^2$ are comparable for every i. Assume that $\mathcal{A}_1^{\alpha_1}, \mathcal{A}_2^{\alpha_2}, \ldots$ are pairwise compatible for any combination of $\alpha_i \in \{1, 2\}$. Also, let $L, \hat{p} \in \mathbb{N}$ be given and let $\mathcal{B}$ be a task-PIOA such that both $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ and $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$ are $\hat{p}$-bounded. Assume that both $\mathcal{A}_1^1, \ldots, \mathcal{A}_L^1$ and $\mathcal{A}_1^2, \ldots, \mathcal{A}_L^2$ are sequential under $\mathcal{B}$ for the same sequence of reals $t_1 < \ldots < t_{L+1}$.

Suppose there exist p, q, ϵ such that $\mathcal{A}_i^1 \leq_{p,q,\epsilon} \mathcal{A}_i^2$ for all i. Let $f_i : S_{\mathcal{A}_i^1} \to S_{\mathcal{A}_i^2}$ denote the function guaranteed by comparability, and define f as $f_1 \times \ldots \times f_L : \prod_{i=1}^{L} S_{\mathcal{A}_i^1} \to \prod_{i=1}^{L} S_{\mathcal{A}_i^2}$. Let $\mathsf{id}_{\mathcal{B}}$ denote the identity function on $S_{\mathcal{B}}$.

[6] For technical reasons, dormant entities must synchronize on input actions. However, since they must remain in a null state, all inputs are trivial loops.

Our goal is to prove that $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ implements $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$. To do so, we must consider a switch from $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ to $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$ at an arbitrary time t (cf. Section 6). Intuitively, since $\mathcal{B}$ invokes the $\mathcal{A}_i$'s sequentially, the substitution operator affects at most one index i. That is, the switching time t falls into at most one interval $[t_i, t_{i+1})$. This idea is expressed in Lemma 4 below.

Lemma 4. *Let $t \in \mathbb{R}_{\geq 0}$ be given and let* Env *be an environment for $\mathcal{S} := \mathsf{sub}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1), \mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2), \mathsf{id}_{\mathcal{B}} \times f, t)$. We have three cases.*

(i) $t < t_1$. Given any valid schedule τ for $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}$, there exists valid schedule τ' for $\mathcal{S}\|\mathsf{Env}$ with the same acceptance probability.

(ii) $t \geq t_{L+1}$. Given any valid schedule τ for $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}$, there exists valid schedule τ' for $\mathcal{S}\|\mathsf{Env}$ with the same acceptance probability.

(iii) There is a unique $\underline{i}$ such that $t \in [t_{\underline{i}}, t_{\underline{i}+1})$. Given any valid schedule τ for $\mathcal{B}\|(\|_{i=1}^{\underline{i}-1}\mathcal{A}_i^1)\|\mathsf{sub}(\mathcal{A}_{\underline{i}}^1, \mathcal{A}_{\underline{i}}^2, f_i, t)\|(\|_{i=\underline{i}+1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}$, there exists valid schedule τ' for $\mathcal{S}\|\mathsf{Env}$ with the same acceptance probability.

Proof. First we consider Case (i), where $\mathcal{S}$ is the result of switching before any of the $\mathcal{A}_i$ components becomes active. Let τ be a valid schedule for $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}$. By comparability, it holds for all i that $\mathcal{A}_i^1$ and $\mathcal{A}_i^2$ have the same set of output tasks and the bound maps coincide on output tasks. Therefore, we can obtain from τ a valid schedule for $\mathcal{S}\|\mathsf{Env}$ by inserting $\{\mathsf{switch}\}$ tasks and hidden tasks of $(\|_{i=1}^{L}\mathcal{A}_i^1)$. Let τ' denote the result of inserting a minimal number of such tasks.

By Definition 4, all of the $\mathcal{A}_i$ components are dormant (i.e., no locally controlled tasks are enabled and inputs are trivial loops) before time t. Therefore, $\mathcal{S}$ behaves exactly like $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$. Moreover, since all f_i's preserve initial state, all variables of $(\|_{i=1}^{L}\mathcal{A}_i^2)$ have value $\perp$ immediately after the first occurrence of $\{\mathsf{switch}\}$, at time t. Hence the behavior of $\mathcal{S}$ under τ' is the same as the behavior of $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$ under τ. Note that no actions actually take place when subsequent $\{\mathsf{switch}\}$ tasks are scheduled, because switch is never again enabled. Similarly for the additional hidden tasks of $(\|_{i=1}^{L}\mathcal{A}_i^1)$.

For Case (ii), we build τ' as in Case (i). Notice that the transition structure of $\mathcal{S}$ is completely isomorphic to that of $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ *before* the switch action. Moreover, by the sequentiality assumption, we know that all variables of $(\|_{i=1}^{L}\mathcal{A}_i^1)$ have value $\perp$ immediately before the switch. Since all f_i's preserve initial state, all variables of $(\|_{i=1}^{L}\mathcal{A}_i^2)$ must also have value $\perp$ immediately after the switch. Again by the sequentiality assumption, these variables must remain $\perp$ forever.

For Case (iii), we apply Cases (i) and (ii) repeatedly to obtain τ'' such that

$$\begin{aligned}&\mathbf{P}_{\mathsf{acc}}(\mathcal{B}\|(\|_{i=1}^{\underline{i}-1}\mathcal{A}_i^1)\|\mathsf{sub}(\mathcal{A}_{\underline{i}}^1, \mathcal{A}_{\underline{i}}^2, f_i, t)\|(\|_{i=\underline{i}+1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}, \tau)\\ &= \mathbf{P}_{\mathsf{acc}}(\mathcal{B}\|(\|_{i=1}^{L}\mathsf{sub}(\mathcal{A}_i^1, \mathcal{A}_i^2, f_i, t))\|\mathsf{Env}, \tau'').\end{aligned}$$

Then we obtain from τ'' a valid schedule τ' for $\mathcal{S}\|\mathsf{Env}$ by replacing each sequence of switch tasks (one for each index i) occurring at the same time with a single $\{\mathsf{switch}\}$ task. Clearly, this preserves the acceptance probability.

□

Now we are ready to show that $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ implements $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$ (Theorem 3). Notice that the error in acceptance probability increases by a factor of b, where b is the largest number of components that may be active in a time interval of length q. For example, if the life time of each component is $\frac{q}{2}$, then b is 3. This is the key difference between parallel and sequential composition: in the parallel case, the error increases with the total number of components (namely, L), and hence no more than a polynomial number of components can be handled. Here, L may be exponential, as long as b remains small. The proof of Theorem 3 involves a standard hybrid argument for active components, while dormant components are replaced by their $\mathsf{trivial}$ counterparts (cf. Definition 2).

Theorem 3. *Let $q \in \mathbb{N}$ be given and let b denote the largest number such that b consecutive t_i's fall into an interval of length q. (Such b must exist and is between 1 and L). Let $p' \in \mathbb{N}$ and $\epsilon' \in \mathbb{R}_{\geq 0}$ be given. Assume that $\epsilon' \geq b \cdot \epsilon$ and $p \geq c_{\mathsf{comp}} \cdot (\hat{p}+p')$, where c_{comp} is the constant factor for parallel composition and $\hat{p}$ is the bound for $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)$ and $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$. Then we have $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1) \leq_{p',q,\epsilon'} \mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2)$.*

Proof. Let $t \in \mathbb{R}_{\geq 0}$ be given. As in Lemma 4, we have three cases. We treat only the most interesting case: there is unique index $\underline{i}$ such that $t \in [t_{\underline{i}}, t_{\underline{i}+1})$.

Let Env be a p'-bounded environment and let τ_1 be a valid timed task schedule for $\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}$ for the interval $[0, t+q]$. We need to find τ_2' for $\mathsf{sub}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1), \mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2), \mathsf{id}_{\mathcal{B}} \times f, t)\|\mathsf{Env}$ such that

$$\begin{aligned}&|\,\mathbf{P}_{\mathsf{acc}}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}, \tau_1)\\ &\quad - \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1), \mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^2), \mathsf{id}_{\mathcal{B}} \times f, t)\|\mathsf{Env}, \tau_2')| \leq \epsilon.\end{aligned}$$

By Lemma 4, it suffices to find τ_2 for

$$\mathcal{B}\|(\|_{i=1}^{\underline{i}-1}\mathcal{A}_i^1)\|\mathsf{sub}(\mathcal{A}_{\underline{i}}^1, \mathcal{A}_{\underline{i}}^2, f_{\underline{i}}, t)\|(\|_{i=\underline{i}+1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}$$

with the same acceptance probability. Let Env_1 denote $\mathcal{B}\|(\|_{i=1}^{\underline{i}}\mathcal{A}_i^1)\|\mathsf{Env}$.

By the choice of b and the fact that $t < t_{\underline{i}+1}$, we know that $t + q < t_{\underline{i}+b+1}$. Applying Lemma 3 repeatedly, we have

$$\begin{aligned}\mathbf{P}_{\mathsf{acc}}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}, \tau_1) &= \mathbf{P}_{\mathsf{acc}}((\|_{i=\underline{i}}^{L}\mathcal{A}_i^1)\|\mathsf{Env}_1, \tau_1)\\ &= \mathbf{P}_{\mathsf{acc}}((\|_{i=\underline{i}}^{\underline{i}+b}\mathcal{A}_i^1)\|\mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}_1, \tau_1).\end{aligned}$$

By comparability, $\mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^1)$ and $\mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^2)$ have the same external signature. Let τ_1' denote the result of removing all internal tasks of $\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^1$ from τ_1. Then

$$\mathbf{P}_{\mathsf{acc}}(\mathcal{B}\|(\|_{i=1}^{L}\mathcal{A}_i^1)\|\mathsf{Env}, \tau_1) = \mathbf{P}_{\mathsf{acc}}((\|_{i=\underline{i}}^{\underline{i}+b}\mathcal{A}_i^1)\|\mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}_1, \tau_1').$$

Let Env_2 denote $\mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L}\mathcal{A}_i^2)\|\mathsf{Env}_1$. Using a standard hybrid argument (cf. the proof of Theorem 1), we find valid schedule τ_b for

$$\mathsf{sub}((\|_{i=\underline{i}}^{\underline{i}+b}\mathcal{A}_i^1), (\|_{i=\underline{i}}^{\underline{i}+b}\mathcal{A}_i^2), f_{\underline{i}} \times \ldots \times f_{\underline{i}+b}, t)\|\mathsf{Env}_2$$

with the same acceptance probability. Using an argument similar to the proof of Lemma 4, we can find valid schedule τ_2 for $\mathsf{sub}(\mathcal{A}^1_{\underline{i}}, \mathcal{A}^2_{\underline{i}}, f_{\underline{i}}, t) \| (\|_{i=\underline{i}+1}^{\underline{i}+b} \mathcal{A}^2_i) \| \mathsf{Env}_2$ with the same acceptance probability. Expanding the definitions of Env_1 and Env_2 and applying Lemma 3 repeatedly, we have

$$\begin{aligned}
&\mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}^1_{\underline{i}}, \mathcal{A}^2_{\underline{i}}, f_{\underline{i}}, t) \| (\|_{i=\underline{i}+1}^{\underline{i}+b} \mathcal{A}^2_i) \| \mathsf{Env}_2, \tau_2) \\
&= \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}^1_{\underline{i}}, \mathcal{A}^2_{\underline{i}}, f_{\underline{i}}, t) \| (\|_{i=\underline{i}+1}^{\underline{i}+b} \mathcal{A}^2_i) \| \mathsf{trivial}(\|_{i=\underline{i}+b+1}^{L} \mathcal{A}^2_i) \| \mathsf{Env}_1, \tau_2) \\
&= \mathbf{P}_{\mathsf{acc}}(\mathsf{sub}(\mathcal{A}^1_{\underline{i}}, \mathcal{A}^2_{\underline{i}}, f_{\underline{i}}, t) \| (\|_{i=\underline{i}+1}^{L} \mathcal{A}^2_i) \| \mathsf{Env}_1, \tau_2) \\
&= \mathbf{P}_{\mathsf{acc}}(\mathcal{B} \| (\|_{i=1}^{\underline{i}-1} \mathcal{A}^1_i) \| \mathsf{sub}(\mathcal{A}^1_{\underline{i}}, \mathcal{A}^2_{\underline{i}}, f_{\underline{i}}, t) \| (\|_{i=\underline{i}+1}^{L} \mathcal{A}^2_i) \| \mathsf{Env}, \tau_2).
\end{aligned}$$

This completes the proof. □

Finally, we use Theorem 3 to obtain a sequential composition theorem for $\le_{\mathsf{neg,pt}}$.

Theorem 4 (Sequential Composition Theorem for $\le_{\mathsf{neg,pt}}$). *Let two sequences of task-PIOA families $\bar{\mathcal{A}}^1_1, \bar{\mathcal{A}}^1_2, \ldots$ and $\bar{\mathcal{A}}^2_1, \bar{\mathcal{A}}^2_2, \ldots$ be given, with $\bar{\mathcal{A}}^1_i$ comparable to $\bar{\mathcal{A}}^2_i$ for all i. Assume that $\bar{\mathcal{A}}_1^{\alpha_1}, \bar{\mathcal{A}}_2^{\alpha_2}, \ldots$ are pairwise compatible for any combination of $\alpha_i \in \{1, 2\}$. Let $L : \mathbb{N} \to \mathbb{N}$ be an exponential function and, for each k, let $(\widehat{\mathcal{A}}^1)_k$ denote $(\bar{\mathcal{A}}^1_1)_k \| \ldots \| (\bar{\mathcal{A}}^1_{L(k)})_k$. Similarly for $(\widehat{\mathcal{A}}^2)_k$.*

Let $\hat{p}$ be a polynomial and let $\bar{\mathcal{B}}$ be a task-PIOA families such that both $\mathcal{B} \| \widehat{\mathcal{A}}^1$ and $\mathcal{B} \| \widehat{\mathcal{A}}^2$ are $\hat{p}$-bounded. Suppose there exist a sequence of positive reals $t_1 < t_2 < \ldots$ such that, for each k, both $(\bar{\mathcal{A}}^1_1)_k, \ldots, (\bar{\mathcal{A}}^1_{L(k)})_k$ and $(\bar{\mathcal{A}}^2_1)_k, \ldots, (\bar{\mathcal{A}}^2_{L(k)})_k$ are sequential under $\mathcal{B}_k$ for the sequence $t_1 < \ldots < t_{L(k)+1}$. Assume there is a constant real number c such that consecutive t_i's are at least c apart.

Suppose that, for all polynomials p, q, there exists negligible function ϵ such that $\bar{\mathcal{A}}^1_i \le_{p,q,\epsilon} \bar{\mathcal{A}}^2_i$ for all i. Then we have $\mathcal{B} \| \widehat{\mathcal{A}}^1 \le_{\mathsf{neg,pt}} \mathcal{B} \| \widehat{\mathcal{A}}^2$.

Proof. Let polynomials p', q' be given and define $p := c_{\mathsf{comp}} \cdot (\hat{p} + p')$, where c_{comp} is the constant factor for composing task-PIOAs in parallel. Choose ϵ according to the assumption of the theorem. For each k, let $b(k)$ be the ceiling of $\frac{q'(k)}{c} + 1$. Since c is constant, b is a polynomial. Define $\epsilon' := b \cdot \epsilon$.

Let $k \in \mathbb{N}$ be given. Let $f_i : S_{(\bar{\mathcal{A}}^1_i)_k} \to S_{(\bar{\mathcal{A}}^2_i)_k}$ denote the function guaranteed by comparability, and define f as $f_1 \times \ldots \times f_L$. Let $\mathsf{id}_{\mathcal{B}_k}$ denote the identity function on $S_{\mathcal{B}_k}$. Now we can apply Theorem 3 to conclude that

$$\mathcal{B}_k \| (\bar{\mathcal{A}}^1_1)_k \| \ldots \| (\bar{\mathcal{A}}^1_{b(k)})_k \le_{p'(k), q(k), \epsilon'(k)} \mathcal{B}_k \| (\bar{\mathcal{A}}^2_1)_k \| \ldots \| (\bar{\mathcal{A}}^2_{b(k)})_k.$$

That is $(\mathcal{B} \| \widehat{\mathcal{A}}^1)_k \le_{p'(k), q(k), \epsilon'(k)} (\mathcal{B} \| \widehat{\mathcal{A}}^2)_k$. This completes the proof. □

8 Conclusion

This paper takes some important first steps towards a foundation for the analysis of long-lived cryptographic services. We augment the Task-PIOA model with real time information on task schedules. This allows us to express computational restrictions in terms of processing rates with respect to real time. As demonstrated by the Turing machine simulation of Section 5, this new complexity model is similar to the standard PSPACE model.

The long-term implementation relation $\leq_{\mathsf{neg,pt}}$ is largely inspired by the timestamping service example of Section 2. We capture the idea that, while an unbounded environment will eventually succeed in guessing a signing key, it may suffice to control the rate at which these successes occur. By virtue of the sequential composition theorem, it is sufficient to analyze each signature service in isolation, checking that the adversary cannot break the service too quickly.

In the future, we plan to study general security definitions based on long-term implementation, and to conduct formal analysis of practical long-lived protocols.

References

1. 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. Available at `http://eprint.iacr.org/2005/452/`.
2. 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`.
3. S. Haber. Long-lived digital integrity using short-lived hash functions. Technical report, HP Laboratories, May 2006.
4. S. Haber and P. Kamat. A content integrity service for long-term digital archives. In *Proceedings of the IS&T Archiving Conference*, 2006. Also published as Technical Memo HPL-2006-54, Trusted Systems Laboratory, HP Laboratories, Princeton.
5. 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.
6. P.D. Lincoln, J.C. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proceedings of ACM CCS-5*, 1998.
7. N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, September 1989.
8. M. Merritt, F. Modugno, and M.R. Tuttle. Time constrained automata. In *Proceedings of CONCUR 1991*, volume 527 of *LNCS*, pages 408–423, 1991.
9. J. Müller-Quade and D. Unruh. Long-term security and universal composability. In *Theory of Cryptography, Proceedings of TCC 2007*, volume 4392 of *LNCS*, pages 41–60. Springer-Verlag, March 2007. Preprint on IACR ePrint 2006/422.
10. 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.