Verifying Statistical Zero Knowledge with Approximate Implementations[⋆]

Ling Cheung[1], Sayan Mitra[1], and Olivier Pereira[2]

[1] Computer Science and AI Laboratory,
Massachusetts Inst. of Technology,
32 Vassar Street, Cambridge, MA 02139, USA
{lcheung, mitras}@csail.mit.edu
[2] Laboratoire de Microélectronique
Université catholique de Louvain
1, Place de l'Université B-1348 Louvain-la-Neuve, Belgium
olivier.pereira@uclouvain.be

Abstract. Statistical zero-knowledge (SZK) properties play an important role in designing cryptographic protocols that enforce honest behavior while maintaining privacy. This paper presents a novel approach for verifying SZK properties, using recently developed techniques based on approximate simulation relations. We formulate statistical indistinguishability as an implementation relation in the Task-PIOA framework, which allows us to express computational restrictions. The implementation relation is then proven using approximate simulation relations. This technique separates proof obligations into two categories: those requiring probabilistic reasoning, as well as those that do not. The latter is a good candidate for mechanization. We illustrate the general method by verifying the SZK property of the well-known identification protocol proposed by Girault, Poupard and Stern.

[⋆] Supported by the MURI project:DARPA/AFOSR MURI F49620-02-1-0325 grant.

1 Introduction

From the early 1980's, it appeared in more and more contexts that proving security is about establishing that the behaviors of systems are indistinguishable. This idea probably appeared first in the security community through the work of Goguen and Meseguer introducing noninterference [1], and in the cryptography community through the work of Goldwasser and Micali introducing semantic security [2].

In cryptography, three flavors of indistinguishability have traditionally been considered [3]. Suppose $X = \{X_k\}_{k \in \mathbb{N}}$ and $Y = \{Y_k\}_{k \in \mathbb{N}}$ are two families of random variables indexed by k, which we refer to as the security parameter. These two families are (i) *perfectly* indistinguishable if they have the same distribution for each k, (ii) *statistically* indistinguishable if the statistical distance between X_k and Y_k decreases faster than any inverse polynomial in k, and (iii) *computationally* indistinguishable if, for every probabilistic polynomial time-bounded[1] distinguisher family $D = \{D_k\}_{k \in \mathbb{N}}$, the probability that D_k distinguishes X_k from Y_k decreases faster than any inverse polynomial in k.

Various techniques have been used to establish perfect indistinguishability, including (bi-)simulation relations, invariant assertion, etc. Computational indistinguishability is traditionally proved by reduction: we first make a fundamental assumption that two (well-known) systems are computationally indistinguishable, and then prove that the existence of a polynomial-time distinguisher for the two protocol systems implies the existence of a distinguisher for the well-known systems, which violates the fundamental assumption.

Recently, much work has been done in the distributed system and security communities to develop reasoning techniques for statistical distance between system executions [4,5,6,7,8,9,10]. These techniques are proposed in the context of noninterference, therefore are not immediately applicable in the analysis of cryptographic protocols. For example, they typically do not consider the notion of polynomial time-bounded computation (an exception being the model of [6]). We shall return to these works in the related work section.

In this paper, we show that statistical indistinguishability properties can be formulated very naturally in the Task-PIOA framework [11,12], and they can be verified using the approximate simulation relation techniques developed in [13,14], which provide formal soundness proofs for these techniques. Here soundness means the existence of an approximate simulation guarantees that every trace distribution in the first system can be matched by some trace distribution in the second system in such a way that the trace distributions are "close" with respect to some metric.

We exemplify our approach through the analysis of a classical identification protocol proposed by Girault, Poupard and Stern [15]. Our analysis establishes that the GPS protocol is statistical zero-knowledge; that is, there is a probabilistic polynomial-time simulator that produces a protocol transcript that is

[1] This means there is a polynomial p such that, for every k, the running time of D_k is bounded by $p(k)$.

statistically indistinguishable from the one resulting from a real execution of the protocol. The crucial point here is, the simulator must achieve this without knowing the private information of the party who proves his identity.

This case study is an important first step towards applying approximate simulation techniques to the verification of statistical security of cryptographic protocols. We believe this is an area of great potential, because simulation relation techniques are well-suited for hierarchical verification of large systems with nondeterministic behavior. Many applications of these techniques in mechanical verification of systems have been published (see, e.g., [16,17] for some recent non-security related case studies in the I/O Automaton framework). Our GPS case study demonstrates that nondeterminism can be used to simplify specifications, without increasing the proof complexity[2]. Our new simulation-based verification technique yields one set of proof obligations which require probabilistic reasoning and a separate set of obligations that do not involve probabilities. The latter type of obligations can be checked using the currently available TIOA Toolsuite and its interface to the PVS theorem prover [18]. Eventually, we hope to (partially) mechanize or even automate the verification of statistical indistinguishability properties for cryptographic protocols.

Related Work. Probabilistic observational equivalence or bisimilarity can only establish perfect indistinguishability or absolute non-interference. But in reality such properties are hard to achieve, if not impossible to achieve. This has been noted both in the security community (for example in [10]), and also in the literature related to verification of timed and probabilistic systems [19]. Based on these observations, there has been intense research in the recent years towards developing metric-based generalizations of probabilistic bisimulations and observational equivalences. Jou and Smolka [20] first introduced the idea of formalizing similarity of observed behavior by using metrics. Approximation metrics for probabilistic systems in the context of *Labelled Markov Processes (LMP)* have been extensively investigated and many fundamental results have been obtained by Desharnais, Gupta, Jagadeesan and Panangaden [21,22,23] and by van Breugel, Mislove, Ouaknine, and Worrell [24,25,26,27,28]. Our notion of approximate implementation, introduced in [13] and developed further in [14], differs from the previous approaches in at least one of the following ways: (a) the task-PIOA model allows both nondeterministic and probabilistic choices, and (b) the implementation relation in our framework is based on trace distributions and not bisimilarity of states. Approximate implementation is derived from a metric over probability distributions of observed behavior, and thus, we do not require the state spaces of the underlying automata nor the common space of Input/Output actions to be metric spaces.

Approximate implementation relations have been applied to verify information flow and confidentiality. See, for example, [9,10,8,4] and the references therein for an overview of this body of work. In order to prove that there is no

[2] Since Task-PIOA uses oblivious, task-based scheduling, nondeterministic specifications can in fact be given to cryptographic protocols.

information flow between two objects A and B, it suffices to show that B cannot distinguish a pair of behaviors of the system that differ only in A's behavior. Typically, variants of weak probabilistic bisimulation are used to verify this type of indistinguishability properties. A general notion of information flow which subsumes the computational case, and the associated verifiaction machinary has been developed in [6]. Also, an approximate notion of bisimilarity is introduced in [10] as a means to prove quantitative bounds on the information flow through systems. However, these two papers do not allow internal nondeterminism[3], as we do here as a means of abstraction and simplification.

The approximate simulation function based verification technique proposed in this paper is particularly suitable for SZK properties because in order to perform the verification task we have the freedom to choose the simulator S as well as the simulation function relating S with P and V. To the best of our knowledge, approximate implementations have not been applied to verify SZK properties prior to this work.

2 Background

Task-structured Probabilistic I/O Automaton (Task-PIOA) [29] is a modeling framework for distributed systems which allows both probabilistic and nondeterministic state transitions. It has a task-based scheduling mechanism which is less powerful that than traditional perfect-information scheduling. This mechanism together with suitable restrictions on the computing power of task-PIOAs and schedulers have been applied to verify cryptographic protocols.

Given a set X, we use $X_\perp$ to denote $X \cup \{\perp\}$ and $\mathsf{Disc}(X)$ to denote the set of discrete probability measures on X. If μ is a discrete probability measure on X, the *support* of μ, written as $\mathsf{supp}(\mu)$, is the set of elements of X that have non-zero measure. The task-PIOA model used in this paper is slightly more general than the one in [30], because we allow the starting configuration of an automaton to be any distribution over states and not just a Dirac mass.

Definition 1. *A* task-structured probabilistic I/O automaton $\mathcal{A}$ *is a 7-tuple* $(Q, \mu_0, I, O, H, D, \mathcal{R})$ *where:*

1. Q *is a countable set of* states*;*
2. $\mu_0 \in \mathsf{Disc}(Q)$ *is the* starting distribution *on states;*
3. I*,* O *and* H *are countable and pairwise disjoint sets of actions, referred to as* input, output, *and* internal actions*, respectively. The set* $A := I \cup O \cup H$ *is called the set of* actions *of* $\mathcal{A}$*. If* $I = \emptyset$*, then* $\mathcal{A}$ *is said to be* closed.
4. $D \subseteq (Q \times A \times \mathsf{Disc}(Q))$ *is a* transition relation*. If* $(q, a, \mu) \in D$*, we write* $q \xrightarrow{a} \mu$*. If* $q \xrightarrow{a} \mu$ *and* $q' \in \mathsf{supp}(\mu)$*, we write* $q \xrightarrow{a} q'$*. An action* a *is* enabled *in a state* q *if there exists* q'*, such that* $q \xrightarrow{a} q'$*.*

[3] By internal nondeterminism we mean nondeterministic choices that are not related to inputs.

5. *$\mathcal{R}$ is an equivalence relation on $O \cup H$. Equivalence classes of $\mathcal{R}$ are called* tasks. *Task T is* enabled *in a state q if some action $a \in T$ is enabled in q.*

In addition, $\mathcal{A}$ satisfies:

(a) For every $q \in Q$ and $a \in I$, a is enabled in q.
(b) For every $q \in Q$ and $a \in A$, there is at most one $\mu \in \mathsf{Disc}(Q)$ such that $(q, a, \mu) \in D$.
(c) For every $q \in Q$ and $T \in R$, at most one $a \in T$ is enabled in q.

The input enabling assumption (a) is standard in the I/O Automaton literature. Assumptions (b) and (c) enable us to resolve nondeterminism by specifying a sequence of tasks.

An *execution fragment* of $\mathcal{A}$ is a finite or infinite sequence $\alpha = q_0\, a_1\, q_1\, a_2\, \ldots$ of alternating states and actions, such that (i) if α is finite, then it ends with a state; and (ii) for every non-final i, there is a transition (q_i, a_{i+1}, q_{i+1}). We write $\alpha.fstate$ for q_0, and, if α is finite, we write $\alpha.lstate$ for its last state. We use $\mathsf{Frags}_{\mathcal{A}}$ (resp., $\mathsf{Frags}^*_{\mathcal{A}}$) to denote the set of all (resp., all finite) execution fragments of $\mathcal{A}$. An *execution* of $\mathcal{A}$ is an execution fragment beginning from some state in $\mathsf{supp}(\mu_0)$. $\mathsf{Execs}_{\mathcal{A}}$ (resp., $\mathsf{Execs}^*_{\mathcal{A}}$) denotes the set of all (resp., finite) executions of $\mathcal{A}$. The *trace* of an execution fragment α, written $\mathsf{trace}(\alpha)$, is the restriction of α to the set of input/output actions of $\mathcal{A}$. We say that β is a *trace* of $\mathcal{A}$ if there is an execution α of $\mathcal{A}$ with $\mathsf{trace}(\alpha) = \beta$. $\mathsf{Traces}_{\mathcal{A}}$ (resp., $\mathsf{Traces}^*_{\mathcal{A}}$) denotes the set of all (resp., finite) traces of $\mathcal{A}$.

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

$$\langle Q_1 \times Q_2, \langle \bar{q}_1, \bar{q}_2 \rangle, (I_1 \cup I_2) \setminus (O_1 \cup O_2), O_1 \cup O_2, H_1 \cup H_2, \Delta, \mathcal{R}_1 \cup \mathcal{R}_2 \rangle,$$

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 (i.e., μ_i is the Dirac measure on q_i, denoted $\mathsf{Dirac}(q_i)$).

2.1 Trace Distributions

In studying statistical indistinguishability of protocols modeled at task-PIOAs, we are interested in analyzing the probability distributions over the set of traces of automata. The distributions over traces are defined in terms of *probabilistic executions*, which are probability distributions over executions. Depending on how nondeterminism is resolved by a *scheduler*, a task-PIOA may give rise to many probabilistic executions. The theory of probabilistic executions of task-PIOAs with a general class of history dependent schedulers has been developed in [30]. However, in cryptographic applications, we restrict our attention to *static* (or *oblivious*), schedulers that do not depend on dynamic information generated during execution. This is so that the scheduler cannot channel dynamically generated information via the ordering of events. We refer to [11] for further discussions regarding oblivious schedulers. Formally, *task schedule* for $\mathcal{A}$ is any finite or infinite sequence $\sigma = T_1 T_2 \dots$ of tasks in R. In this paper we are going to be concerned with finite task schedules only. By virtue of Assumptions (b) and (c) for task-PIOAs, a task schedule can be used to generate a unique probabilistic execution of the task-PIOA $\mathcal{A}$. One can do this by repeatedly scheduling tasks, each of which determines at most one transition of $\mathcal{A}$. This is captured in the following operation that "applies" a task schedule to a task-PIOA.

Definition 2. *Let $\mathcal{A}$ be an action-deterministic task-PIOA. Given $\mu \in \mathsf{Disc}(\mathsf{Frags}^*_{\mathcal{A}})$ and a task schedule σ, $\mathsf{apply}(\mu, \sigma)$ is the probability measure on $\mathsf{Frags}_{\mathcal{A}}$ defined recursively by:*

1. $\mathsf{apply}(\mu, \lambda) := \mu$. *($\lambda$ denotes the empty sequence.)*
2. *For $T \in R$, $\mathsf{apply}(\mu, T)$ is defined as follows. For every $\alpha \in \mathsf{Frags}^*_{\mathcal{A}}$, $\mathsf{apply}(\mu, T)(\alpha) := p_1(\alpha) + p_2(\alpha)$, where:*
   - *$p_1(\alpha) = \mu(\alpha')\eta(q)$ if α is of the form $\alpha' a\, q$, where $a \in T$ and $\alpha'.lstate \xrightarrow{a} \eta$; $p_1(\alpha) = 0$ otherwise.*
   - *$p_2(\alpha) = \mu(\alpha)$ if T is not enabled in $\alpha.lstate$; $p_2(\alpha) = 0$ otherwise.*
3. *For σ of the form $\sigma' T$, $T \in R$, $\mathsf{apply}(\mu, \sigma) := \mathsf{apply}(\mathsf{apply}(\mu, \sigma'), T)$.*

In Case (2) above, p_1 represents the probability that α is executed when applying task T at the end of α'. Because of transition-determinism and action-determinism, the transition $(\alpha'.lstate, a, \eta)$ is unique, and so p_1 is well-defined. The term p_2 represents the original probability $\mu(\alpha)$, which is relevant if T is not enabled after α.

Given any task schedule σ, $\mathsf{apply}(\bar{\nu}, \sigma)$ is a probability distribution over $\mathsf{Exec}_{\mathcal{A}}$. Given a probability measure μ on $\mathsf{Execs}_{\mathcal{A}}$, we define the *trace distribution* of μ, denoted $\mathsf{tdist}(\mu)$ as follows:

$$tdist(\mu)(\beta) = \sum_{\alpha : trace(\alpha) = \beta} \mu(\alpha) - \sum_{a \in E} \sum_{\alpha : trace(\alpha) = \beta a} \mu(\alpha) \quad (1)$$

We write $\mathsf{tdist}(\mu, \sigma)$ as shorthand for $\mathsf{tdist}(\mathsf{apply}(\mu, \sigma))$, the trace distribution obtained by applying task schedule σ starting from the measure μ on execution fragments. We write $\mathsf{tdist}(\sigma)$ for $\mathsf{tdist}(\mathsf{apply}(\bar{\nu}, \sigma))$. A *trace distribution* of $\mathcal{A}$ is any $\mathsf{tdist}(\sigma)$. We use $\mathsf{tdists}(\mathcal{A})$ to denote the set $\{\mathsf{tdist}(\sigma) : \sigma \text{ is a task schedule for } \mathcal{A}\}$ of all trace distributions of $\mathcal{A}$.

We note that the current definition of trace distributions is equivalent to the one in [14], where $\mathsf{tdist}(\mu)$ is simply the image measure under the trace function. The current definition is more natural for statistically indistinguishability statements.

2.2 Time Bounds and Task-PIOA families

Since our target applications are cryptographic protocols, we need to express computational restrictions on task-PIOAs. To do so, we use the definition of time bounds for task-PIOAs in [11,29]. Roughly speaking, a task-PIOA $\mathcal{A}$ has p-bounded description, with $p \in \mathbb{N}$, if $\mathcal{A}$ has a bit-string representation bounded by p and the transition relation is computable by a probabilistic Turing machine with runtime bounded by p. A complete definition can be found in Appendix A.

We note that p limits the size of action names, the amount of available memory and the number of Turing machine steps taken at each transition of $\mathcal{A}$. It, however, does *not* limit the number of transitions that are taken in a particular run. This latter type of limitations is captured by bounds on the length of task schedules. For $q \in \mathbb{N}$, we write $\mathsf{tdists}(\mathcal{A}, q)$ for the set of trace distributions induced by task schedules of length at most q.

A *task-PIOA family* $\overline{\mathcal{A}}$ is an indexed set $\{\mathcal{A}_k\}_{k\in\mathbb{N}}$ of task-PIOAs. The index k is commonly referred to as the *security parameter*. We say that $\overline{\mathcal{A}}$ has *p-bounded description* for some $p : \mathbb{N} \to \mathbb{N}$ just in case: for all k, $\mathcal{A}_k$ has $p(k)$-bounded description. If p is a polynomial, then we say that $\overline{\mathcal{A}}$ has *polynomially-bounded description.*

For task-PIOA families, the notions of compatibility and parallel composition are defined pointwise.

3 Statistical Implementation

In this section, we define a statistical implementation relation for task-PIOA families, capturing the idea that the trace distributions produced by the two families are statistically close. We begin by defining δ-approximate implementation for individual task-PIOAs, and then we generalize to task-PIOA families, with the additional requirement that δ is a negligible function. Formally, a function $\delta : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is said to be *negligible* if, for every constant $c \in \mathbb{N}$, there exists $k_0 \in \mathbb{N}$ such that $\delta(k) < \frac{1}{k^c}$ for all $k \geq k_0$. (That is, δ diminishes more quickly than the reciprocal of any polynomial.)

Implementation relations are defined on task-PIOAs with the same external interface. More precisely, $\mathcal{A}_1$ and $\mathcal{A}_2$ are said to be *comparable* if $I_1 = I_2$ and $O_1 = O_2$. We define as follows δ-approximate implementation with respect to some metric d on trace distributions.

Definition 3. *Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are comparable, closed task-PIOAs and d is a metric on their space of trace distributions. For $\delta > 0$, $\mathcal{A}_1$ is said to δ-implement $\mathcal{A}_2$ with respect to d, written as $\mathcal{A}_1 \leq_{d,\delta} \mathcal{A}_2$, if for every $\eta_1 \in \mathsf{tdists}(\mathcal{A}_1)$ there exists $\eta_2 \in \mathsf{tdists}(\mathcal{A}_2)$ such that $d(\eta_1, \eta_2) \leq \delta$.*

In the rest of this paper, we focus on the following metric $\mathbf{d}$ and we abbreviate $\leq_{\mathbf{d},\delta}$ as $\leq_\delta$.

Definition 4. *Let $\mathcal{A}$ be a closed task-PIOA. The* statistical difference *over trace distributions of $\mathcal{A}$ is the function $\mathbf{d} : \mathsf{Disc}(\mathsf{Traces}_{\mathcal{A}}) \times \mathsf{Disc}(\mathsf{Traces}_{\mathcal{A}}) \to \mathbb{R}_{\geq 0}$ defined by:*

$$\mathbf{d}(\eta_1, \eta_2) \triangleq \sum_{\beta \in \mathsf{Traces}_A} |\eta_1(\beta) - \eta_2(\beta)| \, .$$

Notice, in Definition 3, we do not impose any restrictions on schedule lengths. A variation with schedule length bounds can be obtained easily as follows: for $q_1, q_2 \in \mathbb{N}$, we say that $\mathcal{A}_1 \leq_{q_1,q_2,\delta} \mathcal{A}_2$ if, for every $\eta_1 \in \mathsf{tdists}(\mathcal{A}_1, q_1)$ there exists $\eta_2 \in \mathsf{tdists}(\mathcal{A}_2, q_2)$ such that $\mathsf{d}(\eta_1, \eta_2) \leq \delta$.

Next we define statistical implementation for task-PIOA families with bounded schedule lengths. We do not impose time bounds on the description of these families, because in zero knowledge protocols the prover may be unbounded. Nonetheless, we impose time bounds on schedule lengths, so that the other entities (e.g., verifier and simulator) will be polynomially bounded, as long as they are shown to have polynomially bounded description.

Definition 5. *Let $\overline{\mathcal{A}}_1$ and $\overline{\mathcal{A}}_2$ be pointwise comparable task-PIOA families. We say that $\overline{\mathcal{A}}_1$* statistically implements *$\overline{\mathcal{A}}_2$ if*

$$\forall q_1 \ \exists q_2, \delta \ \forall k \quad (\overline{\mathcal{A}}_1)_k \leq_{q_1(k), q_2(k), \delta(k)} (\overline{\mathcal{A}}_2)_k,$$

where q_1 and q_2 are polynomials and δ is a negligible function.

3.1 Approximate Simulation Relations

We now define a notion of approximate simulation relation, which is a slight variation of the definition given in [13]. We consider here only finite task schedules and we do not make use of the expansion operation. These changes simplify the proof of the soundness theorem (Theorem 1).

Definition 6. *Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are two comparable closed task-PIOAs, ϵ is a nonnegative constant, and ϕ is a function $\mathsf{Disc}(\mathsf{Frags}^*_{\mathcal{A}_1}) \times \mathsf{Disc}(\mathsf{Frags}^*_{\mathcal{A}_2}) \to \mathbb{R}_{\geq 0} \cup \{\infty\}$. The function ϕ is an (ϵ, δ)-*approximate simulation *from $\mathcal{A}_1$ to $\mathcal{A}_2$ if exists a function $\mathsf{c} : R_1^* \times R_1 \to R_2^*$ such that the following properties hold:*

1. **Start condition:** *$\phi(\mu_{10}, \mu_{20}) \leq \epsilon$.*
2. **Step condition:** *If $\phi(\mu_1, \mu_2) \leq \epsilon$, $T \in R_1, \sigma \in R_1^*$ and μ_1 is consistent with σ, and μ_2 is consistent with $\mathsf{full}(\mathsf{c})(\sigma)$, then $\phi(\mu_1', \mu_2') \leq \epsilon$, where $\mu_1' = \mathsf{apply}(\mu_1, T)$ and $\mu_2' = \mathsf{apply}(\mu_2, c(\sigma, T))$.*
3. **Trace condition:** *There exists $\delta > 0$ such that if $\phi(\mu_1, \mu_2) \leq \epsilon$ then $\mathbf{d}_u(\mathsf{tdist}(\mu_1), \mathsf{tdist}(\mu_2)) \leq \delta$.*

Theorem 1. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two closed comparable task-PIOAs. If there exists a (ϵ, δ)-approximate simulation function from $\mathcal{A}_1$ to $\mathcal{A}_2$ then $\mathcal{A}_1 \leq_\delta \mathcal{A}_2$.*

Proof. Let ϕ be the assumed (ϵ, δ)-expanded approximate simulation function from $\mathcal{A}_1$ to $\mathcal{A}_2$. Let μ_1 be the probabilistic execution of $\mathcal{A}_1$ generated by the starting distribution μ_{10} and a finite task schedule $T_1, T_2, \ldots$. For each $i > 0$, we define σ_i to be $\mathsf{c}(T_1 \ldots T_{i-1}, T_i)$. Let μ_2 be the probabilistic execution of $\mathcal{A}_2$ generated by $\bar{\nu}_2$ and the concatenation $\sigma_1, \sigma_2, \ldots$. It suffices to show that: $\mathbf{d}_u(\mathsf{tdist}(\mu_1), \mathsf{tdist}(\mu_2)) \leq \delta$.

For each $j \geq 0$, we define

$$\mu_{1,j} \triangleq \mathsf{apply}(\bar{\nu}_1, T_1, \ldots, T_j) \text{ and } \mu_{2,j} \triangleq \mathsf{apply}(\bar{\nu}_2, \sigma_1, \ldots, \sigma_j)\,.$$

For $i \in \{1, 2\}$ and for each $j \geq 0$, $\mu_{i,j} \leq \mu_{i,j+1}$. Observe that for every $j \geq 0$, $\mu_{1,j+1} = \mathsf{apply}(\mu_{1,j}, T_{j+1})$ and also that $\mu_{2,j+1} = \mathsf{apply}(\mu_{2,j}, \sigma_{j+1})$.

We prove by induction that for all $j \geq 0$, $\hat{\phi}(\mu_{1,j}, \mu_{2,j}) \leq \epsilon$. For $j = 0$, $\mu_{1,0} = \bar{\nu}_1$ and $\mu_{2,0} = \bar{\nu}_2$. By the start condition of the simulation function, $\phi(\mu_{1,0}, \mu_{2,0}) \leq \epsilon$.

For the inductive step, we assume that $\phi(\mu_{1,j}, \mu_{1,j}) \leq \epsilon$ and show that $\phi(\mu_{1,j+1}, \mu_{2,j+1}) \leq \epsilon$. First of all, note that $\mu_{1,j+1} = \mathsf{apply}(\mu_{1,j}, T_{j+1})$ and $\mu_{2,j+1} = \mathsf{apply}(\mu_{2,j}, \mathsf{c}(\sigma_j T_{j+1}))$. From the induction hypothesis $\phi(\mu_{1,j}, \mu_{2,j}) \leq \epsilon$, therefore from the step condition of Definition 6 it follows that $\phi(\mu_{1,j+1}, \mu_{2,j+1}) \leq \epsilon$.

From the previous part and the trace condition of Definition 6, for each $j \geq 0$, $\mathbf{d}_u(\mathsf{tdist}\,\mu_{1,j}, \mathsf{tdist}\,\mu_{2,j}) \leq \delta$. We conclude that $\mathbf{d}_u(\mathsf{tdist}(\mu_1), \mathsf{tdist}(\mu_2)) \leq \delta$. □

In [13,14] it was shown that a more general type of simulation functions, called *expanded approximate simulations* also provide a sound way of proving δ-implementation. This stronger notion was in fact necessary in the Oblivious Transfer case study [11], where matching events may occur at different points during execution of the two systems. We do not make use of this in our GPS case study.

4 GPS Protocol

GPS is an interactive zero-knowledge identification scheme consisting of a number of repetitions of an elementary round. Its parameters are as follows.

- l: number of repetitions of the elementary round.
- $\mathcal{G}$: a generic multiplicative group.
- g: a fixed element of $\mathcal{G}$.
- S: integer upperbound for secret key.
- A, B: additional integer bounds. (We define $\Phi = (B - 1)(S - 1)$.)
- $\kappa \in [0, S - 1]$: the prover's secret key.
- $I = g^\kappa$: the public key.

The parameters $\mathcal{G}$ and g are chosen by some fixed randomized algorithm which takes as input a random tape and a security parameter k. The parameters S, A and B are chosen so that SB/A is negligible in k. The security of GPS then relies on the intractability of computing base g discrete logarithms in $\mathcal{G}$ for exponents taken from the range $[0, S - 1]$.

Each elementary round of GPS is a three-round protocol between the prover and the verifier (Figure 1). In the first round, the prover chooses a number r uniformly at random from the range $[0, A - 1]$ and sends a commitment $x = g^r$ to the verifier. In the second round, the verifier chooses a challenge c uniformly at random from $[0, B - 1]$ and sends it to the prover. Upon receiving c, the prover checks that it is in the right range. In the last round, the prover sends to the verifier $y = r + c \cdot \kappa$, and the verifier checks whether $g^y = x \cdot I^c$ and $y \in [0, A + \Phi - 1]$.

In this paper, we focus on the case where both prover and verifier are honest. This implies all elementary rounds are independent from each other, therefore

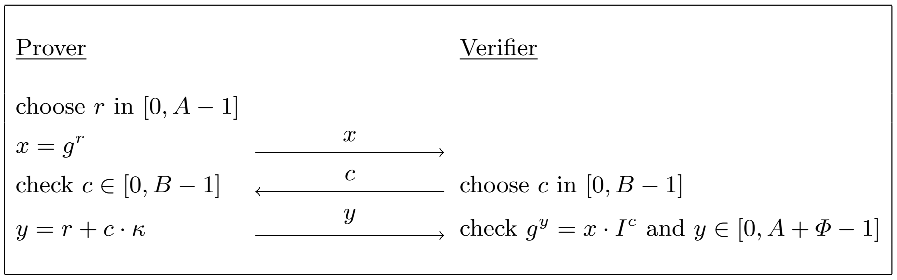

Fig. 1. GPS Elementary Round

a very simple simulator strategy is sufficient. For each elementary round, the simulator
(1) chooses $\langle c, y \rangle$ uniformly at random from $[0, B-1] \times [\Phi, A-1]$, and
(2) produces the transcript $\langle g^y/I^c, c, y \rangle$.
For statistically zero-knowledge, the simulator's transcript must be statistically indistinguishable from the real transcript produced by the prover-verifier pair.

4.1 Task-PIOA Modeling

Our model of the GPS protocol consists of three automata: Prover, Verifier and SIM. Since we are dealing with the case of honest prover and honest verifier, it is sufficient to analyze a single elementary round. (Though it is quite easy to adapt the specifications for multiple-round analysis, by adding history variables.) We treat the parameters $\mathcal{G}, g, A, B, S$ and Φ as globally known constants. The Prover automaton is parameterized with the secret key κ, and the Verifier and SIM are parameterized with the public key I.

The task-PIOA specification for Prover is given in Figure 2. It has two parameterized output actions, Commit(x) and Rely(y), corresponding respectively to the two messages from prover to verifier. It has one parameterized input action, Challenge(c), corresponding to the unique message from verifier to prover. There are three state variables: r_p, c_p and *commited*. The variable r_p is initialized with a random value from $[0, A-1]$ and is used to compute the commitment in the first round. The variable c_p stores the challenge received in the second round. The variable *commited* is a Boolean flag that is set to true after the first Commit action, so that Commit will be performed only once by Prover. Finally, Prover has two tasks, one for each parameterized output action.

The specification for Verifier is given in Figure 2. Just as Prover, it has three parameterized actions corresponding to the three protocol messages, respectively. Note that Commit(x) and Rely(y) are now inputs and Challenge(c) is an output. Verifier has an additional output action, Accept(b), where b is a Boolean value indicating whether the reply y from Prover is accepted. The three state variables: x_v, c_v and y_v contain, respectively, the contents of the three protocol messages.

```
Prover($\kappa : [0, S-1]$)
Variables:
 $r_p : [0, A-1]$, initially choose $uniform\ [0, A-1]$
 $c_p : [0, B-1]_\perp$, initially $\perp$
 $commited$ : Bool, initially false

Actions:
 output Commit($x : \mathcal{G}$), Reply($y : [0, A+\Phi-1]$)
 input Challenge($c : [0, B-1]$)

Transitions:

 Commit($x$)
 pre $\neg commited \wedge x = g^{r_p}$
 eff $commited$ := true

 Challenge($c$)
 eff $c_p := c$

 Reply($y$)
 pre $c_p \neq \perp \wedge y = r_p + c_p \cdot \kappa$
 eff $none$

Tasks:
 {Commit($x$)|$x \in \mathcal{G}$}
 {Reply($y$)|$y \in [0, A-1]$}
```

Fig. 2. Single Round Honest Prover with Secret Key κ.

We remark that c_v is chosen uniformly at random each time a commitment arrives. This is why we make sure that Prover performs Commit only once. Finally, Verifier also has two tasks, one for each parameterized output action.

```
Verifier($I$)
Variables:
 $x_v : \mathcal{G}_\perp$, initially $\perp$
 $c_v : [0, B-1]_\perp$, initially $\perp$
 $y_v : [0, A+\Phi-1]_\perp$, initially $\perp$

Actions:
 input Commit($x : \mathcal{G}$),
  Reply($y : [0, A+\Phi-1]$)
 output Challenge($c : [0, B-1]$),
  Accept($b$ : Bool)

Transitions:

 Commit($x$)
 eff $x_v := x$
  $c_v :=$ choose $uniform\ [0, B-1]$

 Challenge($c$)
 pre $x_v \neq \perp \wedge c = c_v$
 eff $none$

 Reply($y$)
 eff $y_v := y$

 Accept($b$)
 pre $y_v \neq \perp \wedge b = (g^{y_v} = x_v \cdot I^{c_v})$
 eff $none$

Tasks:
 {Challenge($c$)|$c \in [0, B-1]$}
 {Accept($b$)|$b \in$ Bool}
```

Fig. 3. Single Round Honest Verifier with Public Key $I = g^\kappa$.

We refer to the composition of Prover and Verifier as PV. Note that the initial distribution of Prover is $\mathsf{unif}([0, A-1]) \times \mathsf{Dirac}(\perp) \times \mathsf{Dirac}(\mathsf{false})$, because initially r_p is uniformly distributed over $[0, A-1]$. Moreover, all three state variables of Verifier have the value $\perp$ initially, therefore the initial distribution of PV is $\mathsf{unif}([0, A-1]) \times \mathsf{Dirac}(\perp) \times \mathsf{Dirac}(\mathsf{false}) \times \mathsf{Dirac}(\perp) \times \mathsf{Dirac}(\perp) \times \mathsf{Dirac}(\perp)$. We write μ_{10} for this distribution.

The specification for SIM is given in Figure 4. SIM has an internal action Try, which chooses $\langle c, y\rangle$ uniformly at random from $[0, B-1] \times [\Phi, A-1]$ and stores the values in the auxiliary variables c_a and y_a, respectively. We include the Try action in anticipation of the analysis of the cheating verifier case, where the simulator must choose $\langle c, y\rangle$ repeatedly, until an appropriate pair is found.

There are four parameterized output actions, playing the same roles as in Prover and Verifier. The commitment x is computed as g^{y_a}/I^{c_a}, and c_s is set to the same value as c_a after the Commit action. Similarly, y_s is set to the same value as y_a after the Reply action. The Boolean variable *flag* is used to makes sure that Challenge occurs *before* Reply. Finally, SIM has five tasks, one for each action. The initial distribution of SIM is $\mu_{20} := \mathsf{Dirac}(\bot) \times \mathsf{Dirac}(\mathsf{false}) \times \mathsf{Dirac}(\bot) \times \mathsf{Dirac}(\bot) \times \mathsf{Dirac}(\bot)$.

```
SIM($I$)
Variables:
  $x_s : \mathcal{G}_\bot$, initially $\bot$
  $c_s, c_a : [0, B-1]_\bot$, initially $\bot$
  $y_s, y_a : [\Phi, A-1]_\bot$, initially $\bot$
  $flag$ : Bool, initially false

Actions:
  internal Try
  output Commit($x : \mathcal{G}$), Challenge($c : [0, B-1]$),
   Reply($y : [0, A+\Phi-1]$), Accept($b$ : Bool)

Transitions:

  Try
  pre $c_a = \bot$
  eff $c_a$ := choose $uniform$ $[0, B-1]$
     $y_a$ := choose $uniform$ $[\Phi, A-1]$

Commit($x$)
pre  $c_a \neq \bot \wedge x = g^{y_a}/I^{c_a}$
eff  $c_s := c_a$
   $x_s := g^{y_a}/I^{c_a}$

Challenge($c$)
pre $c_s \neq \bot \wedge c = c_s$
eff $flag := true$

Reply($y$)
pre $flag \wedge y = y_a$
eff $y_s := y_a$

Accept($b$)
pre  $y_s \neq \bot \wedge b = (g^{y_s} = x_s \cdot I^{c_s})$
eff $none$

Tasks:
  $\{$Try$\}\{$Commit$(x) | x \in \mathcal{G}\}$
  $\{$Challenge$(c) | c \in [0, B-1]\}$
  $\{$Reply$(y) | y \in [0, A-1]\}$
  $\{$Accept$(b) | b \in$ Bool$\}$
```

Fig. 4. Simulator with Public Key $I = g^\kappa$.

To conclude this section, we make a general remark about the semantics of Task-PIOA specifications. Namely, transitions are enabled whenever their preconditions are met, which implies that some transitions may be repeated. As depicted in Figure 5, the Challenge, Reply and Accept actions may be repeated. It is always possible to add boolean flags so that these actions occur at most once. We have chosen not to do so, in order to keep our specifications simpler. Moreover, this feature may be used to model duplicated messages.

5 Simulation Proof

Throughout this section we fix δ to be $\frac{\Phi+1}{A-1-\Phi}$. We will show that PV δ-implements SIM with respect to the **d** metric of Definition 4 , by establishing a suitable approximate implementation function ϕ. We will use the usual (.) notation to refer to state variable values. For example, we use $v.r_p$ to refer to the value of r_p at state $v \in Q_{\mathsf{PV}}$.

Notice that all of the parameters $\mathcal{G}, g, S, A, B, \Phi, \kappa$ and I depend on the security parameter k. Thus the automata $\mathsf{Prover}, \mathsf{Verifier}$, and Sim, as well as the error δ, are implicitly parameterized by k. Since we do not make any assumptions about k, the results in this section holds for any k.

We begin our development by identifying some simple invariant properties of PV and the SIM. These invariants are proved by induction on the length of the executions of the automata, and they rely on the atomic transition semantics of Task-PIOAs.

Lemma 1. *In all reachable states of* PV, $c_v \neq \bot \iff x_v \neq \bot \iff$ *commited*

Lemma 2. *In all reachable states of* SIM *the following conditions hold:*
1. $y_a \neq \bot \iff c_a \neq \bot$ *2.* $c_s \neq \bot \iff x_s \neq \bot$

Table 5 defines two sets of predicates for SIM and PV. These predicates together with the above invariants provide a concise way of describing sets of executions in which certain actions have taken place. For the purpose of exposition, we say that an execution is in phase i, if its last state is in the i^{th} predicate, but not in $i+1^{st}$. For example, if an execution α of PV has $\alpha.lstate \in \mathcal{V}_2$, then we can conclude that the variables c_v, x_v, and c_p have all been assigned some value (other than $\bot$); we say that α is in phase 2. Now, we define a notion of

Predicate	SIM variables	Predicate	PV variables
$\mathcal{S}_0$	$y_a \neq \bot$		
$\mathcal{S}_1$	$\mathcal{S}_0 \wedge c_s = c_a$	$\mathcal{V}_1$	$c_v \neq \bot$
$\mathcal{S}_2$	$\mathcal{S}_1 \wedge flag$	$\mathcal{V}_2$	$\mathcal{V}_1 \wedge c_p = c_v$
$\mathcal{S}_3$	$\mathcal{S}_2 \wedge y_s = y_a$	$\mathcal{V}_3$	$\mathcal{V}_2 \wedge y_v = r_p + \kappa c_v$

Table 1. Predicates defining execution rounds.

correspondence between the executions of PV and SIM.

Definition 7. *The relation* $\mathcal{R} \subseteq Q_{\mathsf{PV}} \times Q_{\mathsf{SIM}}$ *is defined as follows: For any* $v \in Q_{\mathsf{PV}}$ *and* $u \in Q_{\mathsf{SIM}}$, $v\mathcal{R}u$ *if and only if the following conditions hold: (i) For all* $i \in \{1,2,3\}$, $v \in \mathcal{V}_i \iff u \in \mathcal{S}_i$ *(ii)* $v.x_v = u.x_s$ *(iii)* $v.c_v = u.c_s$ *(iv)* $v.y_v = u.y_s$ *(v)* $v.r_p = u.y_a - \kappa u.c_s$ *(vi)* $v.c_v \neq \bot$.

The first condition in Definition 7 ensures that a pair of executions of PV and SIM are related by $\mathcal{R}$ only if they are in the same phase. Conditions (ii), (iii),(iv), and (v), forces the values sent through by the Commit, $\mathsf{Challenge}$, and Reply messages to be identical. The final condition simply prevents the initial state of PV, which is also a degenerate execution of PV, to be related to any execution of SIM.

Next we prove a key lemma which states that every (nondegenerate) execution of PV is related by $\mathcal{R}$ to a unique execution of SIM that also has the same trace. The proof of this lemma is free from probabilistic reasoning and follows from a standard inductive argument. Suppose some action a is enabled from the

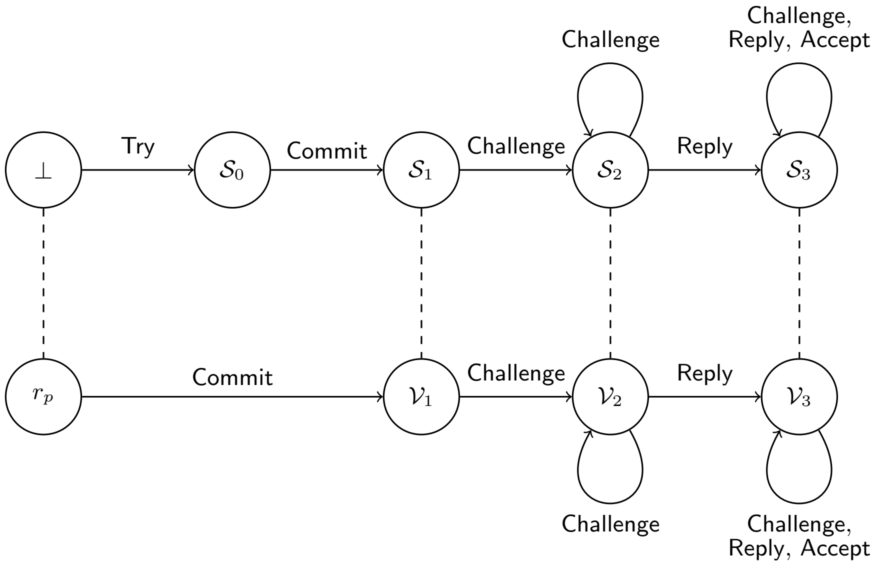

Fig. 5. PV and SIM

last state of any execution α_1 of PV and suppose there exists a unique execution α_2 that is related to α_1 and that has the same trace. We show that the extension of α_1 by a, can be matched by a unique execution of SIM that extend α_2 by a sequence of actions corresponding to a. This correspondence is illustrated in Figure 5. Further, we show that for each action in the sequence, there exists a unique transition of SIM which preserves the relation $\mathcal{R}$ on the last states of the extended executions.

Lemma 3. *For each execution α_1 of* PV *there exists at most one execution α_2 of* SIM *such that $trace(\alpha_1) = trace(\alpha_2)$ and $\alpha_1.lstate\ \mathcal{R}\ \alpha_2.lstate$.*

Proof. The proof is by an induction on the length of the execution α of PV. The base case hold vacuously because of Part (vi) of the definition of $\mathcal{R}$.

For the inductive case, suppose α_1 is an execution of PV and α_2 is the unique execution of SIM such that $trace(\alpha_1) = trace(\alpha_2)$ and $\alpha_1.lstate\mathcal{R}\alpha_2.lstate$. Suppose further that action a is enabled at $\alpha_1.lstate$, say v. Let $\alpha_1' = \alpha a v'$ and $\alpha_2.lstate = u$. We proceed by a case analysis on a:

– Case $a = \mathsf{Commit}(m)$, for some fixed $m \in G$. We show that α_2 can be uniquely extended by two transitions labeled by actions Try and Commit. Let $v'.c_v$ be a fixed element c of $[0, B-1]$, and $v'.r_p = v.r_p$ be a fixed element r of $[0, A-1]$ such that $g^r = m$. As $trace(\alpha_1') = trace(\alpha_1)\mathsf{Commit}$, we have to extend α_2 with an execution fragment with trace Commit.
Since $a = \mathsf{Commit}$ is enabled at v, from Lemma 1 $v \notin \mathcal{V}_1$. From Part (i) of the definition of $\mathcal{R}$, $u \notin \mathcal{S}_1$ and it follows from Lemma 2 that $c_s = \perp$.

Therefore, Try is the only action enabled at u. The Try action corresponds to a set of transitions based on different possible choices of c_a and y_a in the post-state; let u' be *one possible* post state. As $u' \in \mathcal{S}_0 \setminus \mathcal{S}_1$, the only possible enabled action at u' is Commit. In order for $\mathsf{Commit}(m)$ to be enabled at u', c_a must equal m. Let u'' be the uniquely determined post state of u' after Commit occurs. No other actions can occur at u'' without violating the trace condition. Further, $u''.y_a - \kappa c$ has to be equal to r in order for Part (v) of Definition 7 to be preserved. Thus the choice for $u'.y_a$ reduces to a single value, $r + \kappa c$. That is, there is a unique transition $u \to \mathsf{Try}u'$, for which $u' \overset{\mathsf{Commit}}{\to} (m)u''$ and $v'\mathcal{R}u''$. So, α_2 is extended by two successive uniquely defined transitions.

– Case $a = \mathsf{Challenge}(c)$, $c \in \mathbb{Z}$. We will show that α_2 can be uniquely extended by a transition corresponding to the $\mathsf{Challenge}$ action. Since $\mathsf{Challenge}$ is enabled at v, $v \in \mathcal{V}_1$. If in addition $v \in \mathcal{V}_2$, then from Part (i) of Definition 7 it follows that $u \in \mathcal{S}_2$. Therefore $u \overset{\mathsf{Challenge}(c)}{\to} u'$, where $u' = u$. Also, $v \in \mathcal{V}_2$ implies that $v' = v$, which suffices. Clearly, no other action can occur at u while satisfying the trace condition.

  Next, suppose $v \in \mathcal{V}_1 \setminus \mathcal{V}_2$. In this case $v \overset{\mathsf{Challenge}(c)}{\to} v'$ and all the components of v' equal those of v, except that $v'.c_p = c$ and $v.c_p = \perp$. From the induction hypothesis we know that $u \in \mathcal{S}_1 \setminus \mathcal{S}_2$. Therefore $u \overset{\mathsf{Challenge}}{\to} u'$, and all the components of u' equal those of u, except that $u'.flag = true$. It follows that $v'\mathcal{R}u'$.

– The proofs for the cases $a = \mathsf{Reply}$ and $a = \mathsf{Accept}$ are similar to the case for $a = \mathsf{Challenge}$; α_2 is uniquely extended by transitions corresponding to the Reply and Accept actions, respectively.

□

We define a function $\mathcal{F} : \mathsf{Execs}_{\mathsf{PV}} \to (\mathsf{Execs}_{\mathsf{SIM}})_{\perp}$, as follows: $\mathcal{F}(\alpha_1) \triangleq \alpha_2$, if α_2 is the unique execution of SIM for which $\alpha_1\ \mathcal{R}\ \alpha_2$, and $\mathcal{F}(\alpha_1) \triangleq \perp$, if no such α_2 exists. From the proof of Lemma 3 it is clear that $\mathcal{F}(\alpha_1) = \perp$ only when α is a degenerate execution corresponding to one of the starting states of PV. We note that for any non-degenerate execution α_1, Lemma 3 gives a procedure for inductively constructing $\mathcal{F}(\alpha_1)$.

The lemmas stated thus far assert facts about executions of PV and SIM and the relationship between them; their proofs do not require any probabilistic reasoning.

Definition 8. *We define three functions* $\phi, d_1, d_2 : \mathsf{Disc}(\mathsf{Frags}^*_{\mathsf{PV}}) \times \mathsf{Disc}(\mathsf{Frags}^*_{\mathsf{SIM}}) \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ *as follows:*

$$d_1(\mu_1, \mu_2) \triangleq \sum_{\alpha_1 \in \mathsf{Execs}_{\mathsf{PV}}, \mathcal{F}(\alpha_1) \neq \perp} |\mu_1(\alpha_1) - \mu_2(\mathcal{F}(\alpha_2))| \tag{2}$$

$$d_2(\mu_1, \mu_2) \triangleq \mathbf{d}(tdist(\mu_1), tdist(\mu_2)) \tag{3}$$

$$\phi(\mu_1, \mu_2) \triangleq \max_{i \in \{1,2\}} d_i(\mu_1, \mu_2) \tag{4}$$

In what follows, we develop the proof for Lemma 6 which asserts that ϕ is indeed an approximate simulation function for PV and SIM.

We begin by stating certain basic facts about the probabilistic executions of PV that can be generated by task schedules. Lemma 4 states that unless a Commit (Try resp.) task is scheduled the initial distribution for PV (SIM, resp.) does not change.

Lemma 4. *If σ is a task schedule for* PV *that does not contain* Commit *then* $\mathsf{apply}(\mu_{10}, \sigma) = \mu_{10}$. *Similarly, if σ is a task schedule for* SIM *that does not contain* Try *then* $\mathsf{apply}(\mu_{20}, \sigma) = \mu_{20}$.

The next lemma states that for any probabilistic execution μ_1 of PV that is obtained through the application of a task schedule, all the executions in the support of μ_1 are in the same phase.

Lemma 5. *Suppose σ is a task schedule for* PV *and* $\mu_1 = \mathsf{apply}(\mu_{10}, \sigma)$. *The following holds for all $i \in \{1, 2, 3\}$: if there exists $\alpha \in \mathsf{supp}(\mu_1)$, such that $\alpha.lstate \in \mathcal{V}_i$, then for all $\alpha \in \mathsf{supp}(\mu_1)$, $\alpha.lstate \in \mathcal{V}_i$.*

Lemma 6. *ϕ is an (δ, δ)-approximate simulation from* Comp *to* $\mathsf{Simulator}$.

Proof. First, we define the task correspondence function c as follows:

$$\mathsf{c}(\sigma, T) \triangleq \begin{cases} \{\mathsf{Try}\}\{\mathsf{Commit}\} & \text{if } T = \mathsf{Commit} \text{ and } \sigma \text{does } \textit{not} \text{ contain } \mathsf{Commit}, \\ \lambda & \text{if } T = \mathsf{Commit} \text{ and } \sigma \text{ contains } \mathsf{Commit}, \\ T & \text{otherwise.} \end{cases} \tag{5}$$

The proof has three parts corresponding to the three conditions in Definition 6.

Start condition. As μ_{10} and μ_{20} are supported at degenerate executions, $d_2(\mu_{10}, \mu_{20}) = 0$. It is easy to check using Parts (iii) and (vi) of the definition of $\mathcal{R}$ that $d_1(\mu_{10}, \mu_{20})$ is also 0.

Step condition. We assume μ_1 is consistent with some task schedule σ of PV, μ_2 is consistent with $\mathsf{full}(\mathsf{c})(\sigma)$, and that $\phi(\mu_1, \mu_2) \leq \delta$. It suffices to show that for each task T, $d_i(\mu_1', \mu_2') \leq \delta$, for $i = \{1, 2\}$, where $\mu_1' = \mathsf{apply}(\mu_1, T)$ and $\mu_2' = \mathsf{apply}(\mu_2, \mathsf{c}(\sigma, T))$. We proceed by a case analysis on T.

- Case $T = \mathsf{Commit}$. Consider the sub-case where σ contains Commit, then for every execution $\alpha_1 \in \mathsf{supp}(\mu_1)$, $\alpha_1.lstate = \textit{commited}$, that is, the Commit action is disabled. So, $\mu_1' = \mu_1$. Since $\mathsf{c}(\sigma, T) = \lambda$, $\mu_2' = \mu_2$, and from the induction hypothesis it follows that $\phi(\mu_1', \mu_2') \leq \delta$.

  If σ does not contain Commit, then from Lemma 4 $\mu_1 = \mu_{10}$ and $\mu_2 = \mu_{20}$.
  Part 1. We show that $d_1(\mu_1', \mu_2') \leq \delta$. From the definitions of the PV and

SIM automata, we know that

$$\mu_1'(\alpha) = \begin{cases} \frac{1}{AB} & \text{if } \alpha = v_0\mathsf{Commit}\ v_1, v_0 \text{ is an initial state of } \mathsf{PV}\text{, and} \\ & v_0 \overset{\mathsf{Commit}}{\rightarrow} v_1, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

$$\mu_2'(\alpha) = \begin{cases} \frac{1}{[A-1-\Phi]B} & \text{if } \alpha = u_0\mathsf{Try}\ u_1\mathsf{Commit}\ u_2, u_0 \text{ is the initial state of } \mathsf{SIM}, \\ & \text{and } u_0 \overset{\mathsf{Try}}{\rightarrow} u_1, u_1 \overset{\mathsf{Commit}}{\rightarrow} u_2, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Consider an execution α_1 of PV, such that $\mu_1'(\alpha_1) > 0$. From Lemma 3 we know that $\mathcal{F}(\alpha_1)$ is exactly of the form specified by the first condition in Equation (7). So, $\mu_1'(\alpha_1) - \mu_2'(\mathcal{F}(\alpha_2)) = \left|\frac{1}{AB} - \frac{1}{(A-1-\Phi)B}\right| = \frac{1+\Phi}{BA(A-1-\Phi)}$. Alternatively, if $\mu_1'(\alpha_1) = 0$, then $\mathcal{F}(\alpha_1)$ is not of the form specified by the first condition in Equation (7), and therefore $\mu_2'(\alpha_2) = 0$. From Equation (6) it follows that α_1 is not degenerate and by Lemma 3 there exists a unique $\alpha_2 = \mathcal{F}(\alpha_1)$. Thus, $d_1(\mu_1', \mu_2') = \frac{n(1+\Phi)}{BA(A-1-\Phi)}$, where n is the number of unique executions of PV that are of the form specified by Equation (7). The number of possible initial states v_0 of PV is A (r_p is chosen from $[0, A-1]$), and for each of these states the number of possible v_1 states is B (c_v is chosen from $[0, B-1]$ independently of r_p). It follows that $n = AB$, and $d_1(\mu_1', \mu_2') = \frac{1+\Phi}{(A-1-\Phi)} \leq \delta$.

Part 2. We show that $d_2(\mu_1', \mu_2') \leq \delta$. Let $tdist(\mu_i') = \eta_i$, for $i \in \{1, 2\}$. From Equations (6) and (7) it follows that for every trace $\beta \in \mathsf{supp}(\eta_i')$, $\beta = \mathsf{Commit}(m)$ for some $m \in \mathcal{G}$. The value of m uniquely determines $|\eta_1(\beta) - \eta_2(\beta)|$. In what follows, we use the shorthand notation $\langle v_0, v_1\rangle$, to denote an execution $v_0\mathsf{Commit}(m)v_1$ of PV, and $\langle u_0, u_2\rangle$, to denote an execution $u_0\mathsf{Try}u_1\mathsf{Commit}(m)u_2$ of SIM. We rewrite $d_2(\mu_1', \mu_2') = \sum_{m\in\mathcal{G}} |\eta_1(\mathsf{Commit}(m)) - \eta_2(\mathsf{Commit}(m))|$ which can be written as:

$$\sum_{m\in\mathcal{G}} |\mu_1'(\{\langle v_0, v_1\rangle \mid g^{v_0.r_p} = m\}) - \mu_2'(\{\langle u_0, u_2\rangle \mid u_2.x_s = m\})|$$

$$= \sum_{m\in\mathcal{G}} \sum_{k\in[0,A-1], g^k = m} |\mu_1'(\{\langle v_0, v_1\rangle \mid v_0.r_p = k\}) - \mu_2'(\{\langle u_0, u_2\rangle \mid u_2.y_a - \kappa u_2.c_s = k\})|$$

$$\leq \sum_{k\in[0,A-1]} |\mu_1'(\{\langle v_0, v_1\rangle \mid v_0.r_p = k\}) - \mu_2'(\{\langle u_0, u_2\rangle \mid u_2.y_a - \kappa u_2.c_s = k\})|$$

Executions of PV that are of the form $\langle v_0, v_1\rangle$ and have the same value for $v_0.r_p$, differ only with respect to the value of $v_1.c_v$. Similarly, executions of SIM that are of the form $\langle u_0, u_2\rangle$ and have the same value for $u_2.x_s$, differ only with respect to the value of $u_2.c_a$. Rewriting terms as summation over possible values of c_v and c_a and using the triangle inequality, we get:

$$d_2(\mu_1', \mu_2') \leq \sum_{k\in[0,A-1]} \sum_{c\in[0,B-1]} |\mu_1'(\{\langle v_0, v_1\rangle \mid v_0.r_p = k, v_1.c_v = c\}) - \mu_2'(\{\langle u_0, u_2\rangle \mid u_2.x_s = k, u_2.c_a = c\})| .$$

Since $\mathcal{F}(\langle v_0, v_1\rangle) = \langle u_0, u_2\rangle$ for any fixed k and c.

$$d_2(\mu'_1, \mu'_2) \leq \sum_{\alpha, trace(\alpha)\in \mathsf{supp}(tdist(\mu'))} |\mu'_1(\alpha) - \mu'_2(\mathcal{F}(\alpha))| = d_1(\mu'_1, \mu'_2) \leq \delta.$$

– Case $T = \mathsf{Challenge}$. Suppose there exists an $\alpha_1 \in \mathsf{supp}(\mu_1)$, such that $\alpha_1.lstate \notin \mathcal{V}_1$, then by Lemma 5, for every $\alpha_1 \in \mathsf{supp}(\mu_1)$, $\alpha_1.lstate \notin \mathcal{V}_1$. That is, for all $\alpha_1 \in \mathsf{supp}(\mu'_1)$, the $\mathsf{Challenge}$ task is disabled at the last state of α_1. Further, for all such α_1's, $\mathcal{F}(\alpha_1).lstate \notin \mathcal{S}_1$ and the task $\mathsf{c}(\mathsf{Challenge}) = \mathsf{Challenge}$ is also disabled at $\mathcal{F}(\alpha_1).lstate$. So, in this case $\mathsf{Challenge}$ task does not alter the probabilistic executions; from the induction hypothesis we have $\phi(\mu'_1, \mu'_2) \leq \delta$.
On the other hand, for every $\alpha_1 \in \mathsf{supp}(\mu_1)$, $\alpha_1.lstate \in \mathcal{V}_1$, then for any such α_1, $\mathcal{F}(\alpha_1).lstate \in \mathcal{S}_1$.
Part 1. We show that $d_1(\mu'_1, \mu'_2) \leq \delta$. The task $\mathsf{Challenge}$ is enabled at the last state of α_1 and at that of the corresponding execution $\alpha_2 = \mathcal{F}(\alpha_1)$ of SIM. If $\alpha'_1 = \alpha_1 \mathsf{Challenge}\ v_1$ such that $\alpha'_1.lstate \overset{\mathsf{Challenge}}{\rightarrow} v_1$, then $\mu'_1(\alpha'_1) = \mu_1(\alpha_1)$ and $\mu_2(\mathcal{F}(\alpha'_1)) = \mu_2(\mathcal{F}(\alpha_1))$. And for any other α', $\mu'_1(\alpha') = \mu'_2(\mathcal{F}(\alpha') = 0$. Thus, for each $\alpha'_1 \in \mathsf{supp}(\mu'_1)$, $\mu'_1(\alpha'_1) - \mu'_2(\mathcal{F}(\alpha'_1)) = \mu_1(\alpha_1) - \mu_2(\mathcal{F}(\alpha_1))$, where $\alpha'_1 = \alpha_1 \mathsf{Challenge} v_1$. Taking summation over all $\alpha'_1 \in \mathsf{supp}(\mu'_1)$ and applying the induction hypothesis it follows that $d_1(\mu'_1, \mu'_2) \leq \delta$.
Part 2. We show that $d_2(\mu'_1, \mu'_2) \leq \delta$. First of all, consider a trace β'_1 in the support of $tdist(\mu'_1)$. From the previous part we know that $tdist(\mu'_1)(\beta'_1) = tdist(\mu_1)(\beta_1)$ and $tdist(\mu'_2)(\beta'_1) = tdist(\mu_2)(\beta_1)$, where $\beta'_1 = \beta_1 \mathsf{Challenge}(c)$, for some $c \in [0, B-1]$. Next, consider a trace β_1 for which $tdist(\mu'_1)(\beta'_1) = 0$. For any executions α_1 that have $trace(\alpha_1) = \beta'_1$, $\mu'_1(\alpha_1) = 0$. Also, from the previous part and the definition of $\mathcal{F}$ it follows that for the unique execution α_2 of SIM for which $\mathcal{F}(\alpha_2) = \alpha_1$, $\mu'_2(\alpha_2) = 0$, and therefore $tdist(\mu'_2)(\beta'_1) = 0$. Combining these two cases, we have $d_2(\mu'_1, \mu'_2) = \sum_{\beta' \in \mathsf{supp}\, \mu'_1} |tdist(\mu'_1)(\beta'_1) - tdist(\mu'_2)(\beta'_1)| = \sum_{\beta \in \mathsf{supp}\, \mu_1} |tdist(\mu_1)(\beta_1) - tdist(\mu_2)(\beta_1)| = d_2(\mu_1, \mu_2) \leq \delta$.

– The proof for the cases $T = \mathsf{Reply}$ (and $T = \mathsf{Accept}$), is very similar to the proof for $T = \mathsf{Challenge}$. If σ does not contain a sub-sequence of $\mathsf{Commit}, \mathsf{Challenge}$ ($\mathsf{Commit}, \mathsf{Challenge}, \mathsf{Reply}$, resp.), then $\mu'_1 = \mu'_2$. If σ contains such a sub-sequence then μ_1 and μ'_1 are related as follows: $\mu'_1(\alpha'_1) = \mu_1(\alpha)$ if α'_1 is the same as α followed by a Reply (an Accept, resp.) action, and otherwise $\mu'_1(\alpha'_1) = 0$. The distributions μ'_2 and μ_2 are analogously related. And therefore, the values of both the d_1 and d_2 metrics remain unchanged.

Trace condition. for any μ_1, μ_2, if $\phi(\mu_1, \mu_2) \leq \delta$ then from our definition of ϕ, $d_2(tdist(\mu_1), tdist(\mu_2)) \leq \delta$. □

The main result of this section follows from Lemma 6 and Theorem 1.

Theorem 2. *Let* $\overline{\mathsf{PV}}$ *denote the task-PIOA family containing, for each* k*, the automaton* PV *parameterized by* k*. Similarly for* $\overline{\mathsf{Sim}}$*. Then* $\overline{\mathsf{PV}}$ *statistically implements* $\overline{\mathsf{SIM}}$*.*

Proof. By Lemma 6 and Theorem 1, we know that PV is a δ-implementation of Sim. By assumption, SB/A is negligible in k. By definition,

$$\delta = \frac{\Phi + 1}{A - 1 - \Phi} = \frac{(S-1)(B-1)+1}{(A-1-(S-1)(B-1))}.$$

Therefore, δ is also negligible in k. It is now sufficient to note that the function full(c) in the proof of Lemma 6 increases the length of task schedules by at most 1. □

Note that the protocol transcript $\langle x, c, y \rangle$ can be extracted from the traces of PV and Sim (in particular, those traces containing all of Commit, Challenge and Reply). Therefore, we may infer from Theorem 2 that the GPS protocol specified by Prover and Verifier satisfies the statistical zero-knowledge property.

6 Conclusion

In this paper we have presented a technique for verifying SZK properties in the task-PIOA framework [29], using approximate simulation relations. We formulate statistical indistinguishability as a new type of implementation relation for task-PIOAs, called statistical implementation. This is an extension of δ-approximate implementation to the setting of task-PIOA families (with polynomial bounds on schedule lengths). We also develop a general technique for proving statistical implementations, based on the notion of approximate simulation functions introduced in [13,14]. To illustrate the applicability of our methods, we have modeled the GPS identification protocol [15] in the task-PIOA framework. The formal model of GPS turns out to be very natural and contains internal nondeterminism.

A noteworthy feature of our approximate implementation based verification technique is that it allows us to separate probabilistic reasoning from non-probabilistic reasoning. As exemplified in the GPS case study, execution correspondence relations and invariant assertions are free of probabilistic reasoning. By replacing the probabilistic choices in the specifications with the corresponding nondeterministic choices, these lemmas can indeed be verified using the TIOA Toolkit and its interface to the PVS theorem prover [18].

Future Work. The analysis of the GPS protocol presented in this paper assumes an honest verifier that correctly follows the protocol. In the future, we plan on applying the same technique to verify the SZK property of the protocol against cheating verifiers. A cheating verifier Verifier$'$ does not choose the challenge c_v uniformly at random. Instead, it attempts to extract the secret key κ by recording the history of its interaction with the prover and generating the challenge based on some arbitrary function f of the history and the current commitment x_v. The simulator strategy SIM$'$ in this case would be to repeat the Try action a certain number of times, say L, and then execute the Commit action. And the challenging part of the simulation proof would be to find a suitable number L

for which the probability that the SIM' fails to match the commitment and the challenge messages is negligible.

We are interested in exploring applications of the approximate simulation based techniques in the verification of statistical security properties and concurrent zero-knowledge protocols. We also intend to incorporate specialized theorem prover strategies in the TIOA Toolkit that would enable us to partially automate these proofs.

References

1. Goguen, J.A., Meseguer, J.: Security policy and security models. In: Proceedings of the IEEE Symposium on Security and Privacy. (1982) 11–20
2. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker hiding all partial information. In: Proceedings of the 14th Annual ACM Symposium on the Theory of Computing, San Francisco California (1982) 365–377
3. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1) (1989) 186–208
4. Volpano, D., Smith, G.: Probabilistic noninterference in a concurrent language. Journal of Computer Security **7**(2,3) (1999) 231–253
5. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: 13th IEEE Computer Security Foundations Workshop — CSFW'00, Cambridge, UK, IEEE Computer Society Press (2000) 200–214
6. Backes, M., Pfitzmann, B.: Computational probabilistic non-interference. In: 7th European Symposium on Research in Computer Security. Number 2502 in LNCS, Springer-Verlag (2002)
7. Sabelfeld, A.: Confidentiality for multithreaded programs via bisimulation. In: Proceedings of Andrei Ershov 5th International Conference on Perspectives of System Informatics. Number 2890 in LNCS, Springer (2003)
8. Smith, G.: Probabilistic noninterference through weak probabilistic bisimulation. In: Proceedings of the 16th IEEE Computer Security Foundations Workshop, Pacific Grove, California, IEEE Computer Society Press (2003) 3–13
9. Aldini, A., Di Pierro, A.: A quantitative approach to noninterference for probabilistic systems. Electronic Notes in Theoretical Computer Science **99** (2004) 183–203
10. Di Pierro, A., Hankin, C., Wiklicky, H.: Approximate non-interference. Journal of Computer Security **12**(1) (2004) 37–81
11. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Using task-structured probabilistic I/O automata to analyze an oblivious transfer protocol. Cryptology ePrint Archive, Report 2005/452 (2005) `http://eprint.iacr.org/`.
12. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Task-structured Probabilistic I/O Automata. In: Proceedings of the 8th International Workshop on Discrete Event Systems – WODES'2006, IEEE (2006) 207–214
13. Mitra, S., Lynch, N.: Approximate simulations for task-structured probabilistic I/O automata. In: LICS workshop on Probabilistic Automata and Logics (PAuL06), Seattle, WA (2006)
14. Mitra, S., Lynch, N.: Approximate implementation relations for probabilistic I/O automata (2006) To appear in ENTCS.

15. Girault, M., Poupard, G., Stern, J.: On the fly authentication and signature schemes based on groups of unknown order. Journal of Cryptology **19**(4) (2006) 463—487
16. Umeno, S., Lynch, N.A.: Proving safety properties of an aircraft landing protocol using i/o automata and the pvs theorem prover: A case study. In: Formal Methods, 14th International Symposium on Formal Methods. Volume 4085 of Lecture Notes in Computer Science., Springer (2006) 64–80
17. Chockler, G., Lynch, N., Mitra, S., Tauber, J.: Proving atomicity: an assertional approach. In Fraigniaud, P., ed.: Proceedings of Nineteenth International Symposium on Distributed Computing (DISC'05). Volume 3724 of Lecture Notes in Computer Science., Cracow, Poland, Springer (2005) 152 – 168 Full version:`http://theory.lcs.mit.edu/~mitras/research/v30.pdf`.
18. Archer, M., Lim, H., Lynch, N., Mitra, S., Umeno, S.: Specifying and proving properties of timed I/O automata in the TIOA toolkit. In: In Fourth ACM-IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE'06), IEEE (2006)
19. Gupta, V., Jagadeesan, R., Panangaden, P.: Approximate reasoning for real-time probabilistic processes. The Quantitative Evaluation of Systems, First International Conference on (QEST'04) **00** (2004) 304–313
20. Jou, C.C., Smolka, S.A.: Equivalences, congruences and complete approximations for probabilistic processes. In: CONCUR 90. Number 458 in LNCS, Springer-Verlag (1990)
21. Desharnais, J., Jagadeesan, R., Gupta, V., Panangaden, P.: The metric analogue of weak bisimulation for probabilistic processes. In: Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS), Copenhagen, Denmark, 22-25 July 2002, IEEE Computer Society (2002) 413–422
22. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Approximating labelled markov processes. Inf. Comput. **184**(1) (2003) 160–200
23. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Metrics for labelled markov processes. Theor. Comput. Sci. **318**(3) (2004) 323–354
24. van Breugel, F., Worrell, J.: Towards quantitative verification of probabilistic transition systems. In: ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,, London, UK, Springer-Verlag (2001) 421–432
25. van Breugel, F., Mislove, M., Ouaknine, J., Worrell, J.B.: An intrinsic characterization of approximate probabilistic bisimilarity. In: Proceedings of FOSSACS 03. LNCS, Springer (2003)
26. van Breugel, F., Mislove, M.W., Ouaknine, J., Worrell., J.: Domain theory, testing and simulation for labelled markov processes. Theoretical Computer Science (2005)
27. Mislove, M.W., Ouaknine, J., Pavlovic, D., Worrell, J.: Duality for labelled markov processes. In: Proceedings of FOSSACS 04. Volume 2987 of LNCS., Springer (2004)
28. Mislove, M.W., Ouaknine, J., Worrell., J.: Axioms for probability and nondeterminism. ENTCS (2004)
29. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Time-bounded Task-PIOAs: A framework for analyzing security protocols. In Dolev, S., ed.: Proceedings the 20th International Symposium on Distributed Computing (DISC 2006). Volume 4167 of LNCS., Springer (2006) 238–253 Invited Paper.
30. Canetti, R., Cheung, L., Kaynar, D., Liskov, M., Lynch, N., Pereira, O., Segala, R.: Task-structured probabilistic I/O automata. Technical Report MIT-CSAIL-TR-2006-060, Massachusetts Institute of Technology, Cambridge, MA (2006)

A Task-PIOAs and Time Bounds

We assume a standard bit-string representation for various constituents of a task-PIOA, including states, actions, transitions and tasks. Let $p \in \mathbb{N}$ be given. A task-PIOA $\mathcal{A}$ is said to have *p-bounded description* just in case:

(i) the representation of every constituent of $\mathcal{A}$ has length at most p;
(ii) there is a Turing machine that decides whether a given bit string is the representation of some constituent of $\mathcal{A}$;
(iii) there is a Turing machine that, given a state and a task of $\mathcal{A}$, determines the next action;
(iv) there is a probabilistic Turing machine that, given a state and an action of $\mathcal{A}$, determines the next state of $\mathcal{A}$;
(v) all these Turing machines can be described using a bit string of length at most p, according to some standard encoding of Turing machines;
(vi) all these Turing machines return after at most p steps on every input.

Suppose we have a compatible set $\{\mathcal{A}_i : 1 \leq i \leq b\}$ of task-PIOAs, where each $\mathcal{A}_i$ has description bounded by some $p_i \in \mathbb{N}$. It is not hard to check that the composition $\|_{i=1}^{b}\mathcal{A}_i$ has description bounded by $c_{\mathbf{compose}} \cdot \sum_{i=1}^{b} p_i$, where $c_{\mathbf{compose}}$ is a fixed constant (the proof of this result in an immediate extension of the two task-PIOAs case described in [11, Lemma 4.2]).