Automatic Verification of Simulatability in Security Protocols

Tadashi Araragi
NTT Communication Science Laboratories,
2-4 Hikaridai Seika-cho Soraku-gun,
Kyoto, Japan 619-0237
araragi@cslab.kecl.ntt.co.jp

Olivier Pereira*
Université catholique de Louvain
Place du Levant, 3 B-1348
Louvain-la-Neuve, Belgium
olivier.pereira@uclouvain.be

Abstract

This paper investigates the problem of the automatic verification of the computational indistinguishability of systems in the simulation-based security setting, which allows proving the composable security of cryptographic protocols whose security relies on computational hardness assumptions. We use task-structured Probabilistic I/O Automata (task-PIOA) as our modeling framework. In this context, proofs of indistinguishability between real and ideal systems are typically divided into steps involving either proofs of perfect indistinguishability or proofs of computational indistinguishability. Our method automates the proof of perfect indistinguishability for a class of simple protocols, which is, by far, the most error-prone and time-consuming part of those security proofs. We proceed by transforming the targeted real and ideal probabilistic systems into nondeterministic ones, and check the bisimulation between the obtained systems by a partition refinement algorithm. We prove the correctness of our transformation. Our method has also been implemented in a symbolic way and we showed its usefulness by applying it to a practical protocol for oblivious transfer.

keywords: security protocols, formal verification, simulation, computational security, task-PIOA

1 Introduction

The wide adoption of complex cryptographic protocols in open environments stimulates the elaboration of more and more sophisticated models for the security analysis of such protocols [1, 5, 7, 13]. In turn, the complexity of the security proofs obtained in these models has recently attracted much effort to mechanize such proofs [2, 4, 11, 15].

In this paper, we outline a new technique for obtaining tool-supported security proofs in the task-PIOA framework of Canetti et al. [7] that permits the modeling of both probabilistic and nondeterministic choices, resulting in security notions incomparable to those obtained in pure probabilistic models (as shown in [8]).

A central aspect of the security proofs in all of the above models consists in proving the indistinguishability of two systems by any (computationally-bounded) environment. In the task-PIOA framework, this indistinguishability notion is captured by a series of implementation relations, $\leq_0$, $\leq_\delta$, and $\leq_{neg,pt}$, that capture perfect, statistical, and computational indistinguishability, respectively, [7, 10].

To establish that two task-PIOAs, A and B, are $\leq_{neg,pt}$-related,[1] proofs are split into several steps (as in game-based proofs [3, 14]): to show that $A \leq_{neg,pt} B$, one proves that $A_1 \leq_{neg,pt} \cdots \leq_{neg,pt} A_n$, where $A_1 = A$ and $A_n = B$, and exploits the transitivity of the implementation relation. Families A_i and A_{i+1} are defined so that they are either perfectly indistinguishable (that is, they are $\leq_0$-related) or only differ by a small detail corresponding to a computational assumption. As observed in [7], for instance, the proofs of the steps corresponding to perfect implementation relation $\leq_0$, based on establishing simulation relations, are especially page consuming (around 30 pages for a simple protocol in [6]) and error-prone. We propose here a technique that allows automation of these proof steps, based on a partition refinement algorithm that is generally used for showing bisimulation relations in process algebra (the other steps typically require more imagination, but can be easily managed by hand).

2 Preliminaries

2.1 Task-PIOA

Task-PIOA is a framework of I/O automata that enables sophisticated discussion on the composition of systems and

*O. Pereira is a Research Associate of the Belgian Fund for Scientific Research – F.R.S.-FNRS.

[1] Formally, this relation involves task-PIOA families, indexed by security parameter k. We omit this distinction here.

the nondeterminism of the behavior. A probabilistic I/O automaton ($PIOA$) A is a tuple $< Q, \overline{q}, I, O, H, \Delta >$, where (i) Q is a set of states, with $start\ state\ \overline{q} \in Q$; (ii) I, O, and H are sets of $input$, $output$, and $internal\ actions$, respectively; (iii) $\Delta \subset Q \times (I \cup O \cup H) \times Disc(Q)$ is a $transition\ relation$, where $Disc(Q)$ is the set of discrete probabilistic distributions on Q. A transition $\langle\ q,\ a,\ \mu\ \rangle \in \Delta$ means that action a is $enabled$ in state q and has a probabilistic transition to the next states specified by μ. Actions in $I \cup O$ are called $external$ actions and those in $O \cup H$ are called $locally\ controlled$ actions. We denote $I \cup O \cup H$ by $Act(A)$.

The parallel composition of PIOAs is based on the synchronization of shared actions. Two PIOAs, $A_1 = < Q_1, \overline{q}_1, I_1, O_1, H_1, \Delta_1 >$, and $A_2 = < Q_2, \overline{q}_2, I_2, O_2, H_2, \Delta_2 >$, are said to be $compatible$ if $Act(A_1) \cap H_2 = Act(A_2) \cap H_1 = O_1 \cap O_2 = \phi$. In that case, we define composition $A_1 \| A_2$ as $< Q_1 \times Q_2, \langle\ \overline{q}_1, \overline{q}_2\ \rangle, (I_1 \cup I_2) \setminus (O_1 \cup O_2), (O_1 \cup O_2), (H_1 \cup H_2), \Delta >$, where Δ is $\{\ \langle\ \langle\ q_1, q_2 \rangle, a, \mu_1 \times \mu_1\ \rangle \mid a \in A_1 \cap A_2, \langle\ q_1, a, \mu_1\ \rangle \in \Delta_1, \langle\ q_2, a, \mu_2\ \rangle \in \Delta_2\ \} \cup \{\ \langle\ \langle\ q_1, q_2\ \rangle, a, \mu_1 \times \delta(q_1)\ \rangle \mid a \in A_1 \setminus A_2, \langle\ q_1, a, \mu_1\ \rangle \in \Delta_1\ \} \cup \{\ \langle\ \langle\ q_1, q_2\ \rangle, a, \delta(q_1) \times \mu_2\ \rangle \mid a \in A_2 \setminus A_1, \langle\ q_2, a, \mu_2\ \rangle \in \Delta_2\ \}$. $\delta(q)$ is the distribution where q has probability 1.

To discuss the probabilistic properties of task-PIOAs, a partition of the locally controlled actions of PIOAs into *tasks* is introduced, those tasks serving as units of scheduling for resolving the nondeterminism. Formally a *task-PIOA* is a pair $\langle P, R \rangle$ where P is a PIOA and R is an equivalence relation on the locally controlled actions of P. Each equivalence class of R is called a task. We assume that task-partitions satisfy the following condition: Action determinism: for every state q and every task T of a task-PIOA, there is at most one action $a \in T$ enabled in q.

The composition of two task-PIOAs is given by the composition of their underlying PIOAs and by the union of their task partitions.

The action determinism condition enables using *task schedulers* to resolve the nondeterminism. A task scheduler ρ is a finite or infinite sequence of tasks $T_1 T_2 T_3 \cdots$. This induces a probabilistic execution. Tasks $T_1 T_2 T_3 \cdots$ are applied from the start state $\overline{q}$ in this order, and we obtain a series of probabilistic transitions called $probabilistic\ execution\ generated\ by\ \rho$ and denoted by $\epsilon_{A,\rho}$ or ϵ_ρ. By the action determinism, when each task T_i is applied, at most one action in T_i is enabled in each possible current state. If there is, we apply the action, and otherwise we do nothing and move to the application of next task T_{i+1}. In this way, the nondeterminism is resolved when a scheduler is given. When a probabilistic execution ϵ (= ϵ_ρ) and a scheduler ρ' are given, $apply(\epsilon, \rho')$ is defined as $\epsilon_{\rho\rho'}$. A probabilistic execution generated by a finite scheduler is called $finite$. A finite probabilistic execution can be seen as a distribution on finite executions. For a finite probabilistic execution ϵ, let $exec_1 \cdots exec_n$ be the executions with non-zero probability by ϵ. Then $lstate(\epsilon)$ is the distribution on the set of the last states of $exec_1 \cdots exec_n$ that assigns probability $\epsilon(exec_i)$ to the last state of $exec_i$ for each i. A $trace\ distribution$ for a probabilistic execution ϵ, denoted by $tdist(\epsilon)$, is the result of abstracting away internal actions and each transient state in ϵ.

2.2 Implementation relations

The indistinguishablity of two systems of task-PIOAs is formulated as follows. We denote the set of trace distributions generated from task-PIOA A by $tdists(A)$. An $environment\ E$ for A is defined as a task-PIOA compatible with A such that $A \parallel E$ has no input action and E has a special output action $accept$. Task-PIOAs A_1 and A_2 are said to be $comparable$ when they have the same external actions. The statement that any environment cannot distinguish two comparable task-PIOAs A_1 and A_2 is formulated as follows. For any environment E for A_1 and A_2, $tdists(\ A_1 \parallel E\ ) \subset tdists(\ A_1 \parallel E\ )$. In other words, for any environment E for A_1 and A_2, for any task scheduler ρ_1 of $A_1 \parallel E$, there is a task scheduler ρ_2 of $A_2 \parallel E$ such that $tdist(\epsilon_{A_1 \| E, \rho_1}) = tdist(\epsilon_{A_2 \| E, \rho_2})$. In this case, we say that $A_1\ implements\ A_2$, and this relation is denoted by $A_1 \leq_0 A_2$.

An approximate implementation relation is also introduced in the task-PIOA framework in order to describe computational indistinguishability. This relation is written as $A_1 \leq_{neg,pt} A_2$ and informally means that any environment with only polynomial time computational power has a negligible chance to distinguish a trace distribution of A_1 from those of A_2. We do not define the $\leq_{neg,pt}$ in more details here as it is only instrumental in this paper, a detailed discussion is available in [7].

2.3 Known results

We list some known results related to our method. In the following, A_1, A_2, and A_3 are task-PIOAs.

Theorem 1 *[7] (1) If $A_1 \leq_{neg,pt} A_2$ and $A_2 \leq_{neg,pt} A_3$, then $A_1 \leq_{neg,pt} A_3$.*
(2) If $A_1 \leq_{neg,pt} A_2$ and A_3 is compatible with A_1 and A_2 respectively, then $A_1 \parallel A_3 \leq_{neg,pt} A_2 \parallel A_3$.

With this result, when we prove a security requirement $Real \leq_{neg,pt} Ideal$ in a task-PIOA framework for a real system $Real$ and an ideal system $Ideal$, we decompose this implementation relation in an alternative sequence of implementations by introducing intermediate task-PIOAs Int_i: $Real \leq_0 Int_1 \leq_{neg,pt} Int_2 \cdots \leq_{neg,pt} \cdots Int_k \leq_0 Ideal$,

where $\leq_{neg,pt}$ parts correspond to the computational assumptions used in the Real protocols. Then we prove these implementation relations individually.

In order to establish the $\leq_0$ relation between two task-PIOAs A_1 and A_2 a simulation relation has been proposed, which enables our automatic verification. Let R be a relation on probabilistic executions. Then the expansion of R, denoted by $\mathcal{E}(R)$, is defined as follows. $\epsilon_1' \; \mathcal{E}(R) \; \epsilon_2'$ if and only if there are probabilistic executions $\{\epsilon_1^i\}$, $\{\epsilon_2^i\}$, distributions $\{p_1^i\}$, $\{p_1^j\}$, that is, p_k^i>0 and $\sum_i p_k^i$ = 1, a class of positive values $\{w_{i,j}\}$ such that (i) ϵ_1' = $\sum_i p_1^i \epsilon_1^i$, ϵ_2' = $\sum_i p_2^i \epsilon_2^i$ (ii) if $\epsilon_1^i R \epsilon_2^j$ then $w_{i,j} > 0$, otherwise, $w_{i,j}$ = 0 (iii) p_1^i = $\sum_j w_{i,j}$ and p_2^j = $\sum_i w_{i,j}$. Now, we define the $simulation$ relation between two task-PIOAs. Let A_1 and A_2 be task-PIOAs and R be a relation on the probabilistic executions of A_1 and A_2. R is a simulation from A_1 to A_2 if the following conditions are satisfied. Let ϵ_1 and ϵ_2 be the probabilistic executions of A_1 and A_2, respectively: (1) if $\epsilon_1 R \epsilon_2$, then tdist(ϵ_1) = tdist(ϵ_2); (2) there exists mapping c: $R_1^* \times R_1 \rightarrow R_2^*$ exists such that (i) Start condition: $\delta(\overline{q}_1) R \; \delta(\overline{q}_2)$; (ii) Step condition: if $\epsilon_1 R \epsilon_2$, $\epsilon_1 = \epsilon_{\rho_1}$, and T is a task of A_1, then $apply(\epsilon_1, T) \; \mathcal{E}(R) \; apply(\epsilon_2, c(\rho_1, T))$.

Lemma 1 ([6]) *Let R be a simulation relation from A_1 to A_2. If $\epsilon_1 \mathcal{E}(R) \epsilon_2$, $\epsilon_1 = \epsilon_{\rho_1}$ and T is a task of A_1, then $apply(\epsilon_1, T) \; \mathcal{E}(R) \; apply(\epsilon_2, c(\rho_1, T))$*

Note that the premise is not $\epsilon_1 R \epsilon_2$ but $\epsilon_1 \mathcal{E}(R) \epsilon_2$. This lemma is used for the proof of our correctness theorem in Section 3.4. The existence of a simulation relation of the type defined above is a sufficient condition for the $\leq_0$ relation.

Theorem 2 ([7]) *Let A_1 and A_2 be two comparable task-PIOA. Suppose that, for any environment E for A_1 and A_2, there is a simulation relation R from $A_1 \| E$ to $A_2 \| E$. Then $A_1 \leq_0 A_2$.*

Besides, the existence of a simulation relation between two task-PIOAs is also a sufficient condition for the $\leq_{neg,pt}$-relation, as soon as there exists a task mapping c that satisfies condition (2) of the simulation relation definition and never maps any task and sequence of tasks on more than a constant number of tasks.

Theorem 3 ([7]) *Let A_1 and A_2 be two comparable task-PIOA. Suppose that, for any environment E for A_1 and A_2, there is a simulation relation from $A_1 \| E$ to $A_2 \| E$ using task mapping c, and $c(\rho_1, T)$ is b-bounded for every ρ_1 and T for some $b \in \mathbb{N}$. Then $A_1 \leq_{neg,pt} A_2$.*

3 Verification Procedure

3.1 Basic idea of verification

The $\leq_0$-relation of two task-PIOAs is typically proved by a exhibiting simulation relation R that relates the two compared automata. However, verifying the simulation of probabilistic systems is generally difficult and intractable [9].

To cope with this difficulty, we make some assumptions about the systems we expect to compare based on observations from previous analyzes. The most central point is that all sophisticated computational behaviors are isolated in the proof steps in which computational assumptions are exploited. As a result, the steps where the $\leq_0$ relation must be proved only relate variables that are distributed based on very simple distributions: they are either uniformly distributed in some set or fixed to some value that corresponds to the Dirac distribution.

Based on this assumption, we introduce units of actions as a sequence of invisible actions followed by a single observable action. Then to verify the coincidence of trace distributions, we check the simulation of the units of actions based not only on observable actions but also on the set of internally executed random value generation actions in each unit by a verification method for nondeterministic bisimulation. The uniformity of the randomness guarantees the soundness of this procedure.

3.2 Verification setting

Class of targeting protocols: Our method of proving relation $\leq_0$ is currently restricted to some class of protocols defined as follows. Each protocol has a finite number of variables called $state \; variables$ to specify its state. There are four kinds of actions: sending a message, receiving a message, random value generation, and calculation. The precondition of each action is a logical conjunction of predicates of the form $x_i = \perp$ or $x_i \neq \perp$, where x_i is a state variable. $x_i = \perp$ means that x_i has no value. The postcondition of each action is a parallel substitution of the values to state variables. We assume that the state variable initially has no value and is instantiated once during the protocol execution. It is simply achieved by including $x \neq \perp$ in the precondition, when the action includes the substitution of x in its postcondition. The random value generation generates a uniformly distributed value, and the calculation is nondeterministic action. Therefore , only the probabilistic behavior of the protocol is uniform, except the actions of the composed environment. We call the protocols that satisfy the above conditions $value \; independent$ protocols. We believe some security protocols satisfy these assumptions.

Tasks: Sending and receiving actions have a form send(msg_id, $dest\_id$, x_{i_1}, x_{i_2}, $\cdots$), where msg_id is a message id distinguished from other type of messages and $dest\_id$ is a destination id. If msg_id is its own id, then the action is a receiving one, and otherwise it is sending. x_j is a state variable and its current value appears at the occurrence in the action. In the task-PIOA presentation, sending actions with the same msg_id constitute one task as well as receiving actions. Each random value generation and calculation individually constitutes a task. Sending messages from/to an environment or an adversary are called $external$ tasks, and the other are $internal$ tasks. We assume the number of tasks is finite.

3.3 Transformation of probabilistic systems to nondeterministic systems for verification

When a probabilistic system A of a value independent protocol is given as a task-PIOA, we transform A into pure nondeterministic system A^{ab}.

First, we abstract away the individual random values appearing in actions as parameters. That is, we reduce action $task\_name(v_1, \ldots, v_k)$ to $task\_name$. We also abstract away the post condition. When the action includes a substitution x:= v in the post condition, it is replaced by x:= T. For a value independent protocol, each reduced action of A corresponds to a unique task. With this correspondence, we classify the reduced actions into external and internal actions. Let Int be the set of internal reduced actions of A and Ext be the set of external reduced actions of A. A *unit of actions* is defined as a pair of a finite sequence of internal reduced actions and an external action: an element of $Int^* \times Ext$. Let ($[in_1 \ldots in_k]$, ob) be a unit of actions denoted by ut. From ut, we create the action of A^{ab} as follows and denote it by $det\_action(ut)$.

1. The action name of $det\_action(ut)$ is (ob, $dist_{rd_1} \times \ldots \times dist_{rd_m}$), where $rd_1, \ldots, rd_m$ are the actions that generate the uniformly distributed random values included in $\{in_1, \ldots, in_k\}$, and $dist_{rd_i}$ is the distribution of rd_i.
2. The precondition of $det\_action(ut)$ is the merger of the preconditions of $in_1, \ldots, in_k$, and o in this order. That is, if a precondition of in_j or o is satisfied by a postcondition of a proceeding $in_{j'}$ ($1 \leq j' \leq j$ or $1 \leq j' \leq k$), then the precondition is not included in the merging.
3. The postcondition of $det\_action(ut)$ is simply the union of the postconditions of $in_1, \ldots, in_k$ and ob.

Then the set of actions of A^{ab} is defined as $\{det\_action(ut) \mid ut \in Int^* \times Ext\}$.

For the state of A^{ab}, we retain the identical state variables as A, but their domains are changed to $\{\top, \bot\}$ from the full range. We denote the result of the abstraction of state s by s^{ab}.

Informally, this transformation abstracts away individual random values in A and only keeps the information of what distributions caused by random value generation actions ($\{rd_1, \ldots, rd_m\}$) are brought in each unit of actions. This information determines the probabilistic behavior of the unit of actions because the randomness is uniform. Therefore, the action name of $det\_action(ut)$ represents the observable probabilistic behavior of ut.

By our assumptions, the sets of A^{ab}'s actions and its states are finite. Originally, the number of units of actions is exponential for In, but operation det_action greatly reduces the number because the resulting actions are equivalent if their action names, preconditions, and postconditions are individually identical. On the other hand, there are different actions with the same action name.

We introduce some notations. An action of A^{ab} is a class of the sequences of reduced actions. As stated before, a reduced action corresponds to a task. Hence, when a finite scheduler ρ whose last task is external and the others are internal is given, we can consider ρ a unit of actions and denote the associated action of A^{ab} by $[\rho]$. A scheduler whose last task is external and the others are internal is called a $unit\ scheduler$. When an action of A^{ab} act is given, it has name of the form (ob, $dist_{rd_1} \times \ldots \times dist_{rd_m}$). We denote ob by $r.ext$ and $dist_{rd_1} \times \ldots \times dist_{rd_m}$ by $r.dist$.

3.4 Correctness of transformation

Theorem 2 states that the existence of a simulation relation between two task-PIOAs implies that they implement each other. Besides, in the similar setting of Lynch et al. [12], the other direction of this statement is proved, that is, the implementation relation implies the existence of a simulation relation between two PIOAs. This result has not been transposed to the task-PIOA setting, but we prove it to be true for at least a proper restrictions on the targeting class of task-PIOAs. Then, we state the correctness of our transformation as the existence of simulation in original task-PIOAs is equivalent to the existence of simulation in the transformed nondeterministic systems.

For probabilistic execution ϵ, let $lstate(\epsilon)^{ab}$ be s^{ab} for a $s \in supp(lstate(\epsilon))$.

Lemma 2 *For value independent protocols, $lstate(\epsilon)^{ab}$ is well defined. That is, it is independent from the choice of s.*

This, which is one of the characteristics of value independent protocols, is easily verified.

Theorem 4 *Let A_1, A_2 be the probabilistic systems in a task-PIOA for value independent protocols. Then the following holds. There is a simulation from A_1^{ab} to A_2^{ab} if and only if there is a simulation from A_1 to A_2.*

Proof: ($\Rightarrow$). We construct a simulation relation from A_1 to A_2 by a given simulation relation R^{ab} from A_1^{ab} to A_2^{ab}. We define the relation R as $\epsilon_1 R\ \epsilon_2$ if $lstate(\epsilon_1)^{ab} R^{ab} lstate(\epsilon_2)^{ab}$. Now, we show that R is a simulation relation. For the start condition, $lstate(\delta(\overline{q_i}))^{ab}$ = $\overline{q_i}^{ab}$ and $(\overline{q_1})^{ab} R^{ab} (\overline{q_2})^{ab}$. Then $\delta(\overline{q_1}) R \delta(\overline{q_2})$ holds. For the step condition, we slightly modify the step condition as follows.

Modified step condition: Let ρ_1, ρ_2 be finite schedulers whose last task is external, $\epsilon_{\rho_1} R \epsilon_{\rho_2}$ and $tdist(\epsilon_{\rho_1})$ = $tdist(\epsilon_{\rho_2})$. Then for any unit scheduler ρ_1', there is a unit scheduler ρ_2' such that $\epsilon_{\rho_1\rho_1'} \mathcal{E}(R) \epsilon_{\rho_2\rho_2'}$ and $tdist(\epsilon_{\rho_1\rho_1'})$ = $tdist(\epsilon_{\rho_2\rho_2'})$.

Let s_i = $lstate(\epsilon_i)^{ab}$. By the assumption $\epsilon_{\rho_1} R \epsilon_{\rho_2}$, $s_1 R^{ab} s_2$. When there is a transition $s_1 \stackrel{[\rho_1']}{\rightarrow} s_1'$, there is a unit scheduler ρ_2' such that $s_2 \stackrel{[\rho_2']}{\rightarrow} s_2'$, $s_1' R^{ab} s_2'$, $[\rho_1'].ext$ = $[\rho_2'].ext$ and $[\rho_1'].dist$ = $[\rho_2'].dist$. Because $lstate(\epsilon_{\rho_i\rho_i'})^{ab}$ = s_i', $s_1'\ R\ s_2'$, and then $s_1'\ \mathcal{E}(R)\ s_2'$. By $[\rho_1'].ext$ = $[\rho_2'].ext$ and $[\rho_1'].dist$ = $[\rho_2'].dist$, $tdist(\epsilon_{\rho_1\rho_1'})$ = $tdist(\epsilon_{\rho_2\rho_1'})$, as desired.

($\Leftarrow$) By the assumption, there is a simulation relation from A_1 to A_2. We denote this relation by R. Now we define a simulation from A_1^{ab} to A_2^{ab}, denoted by R^{ab}, as $s_1 R s_2$ if there are probabilistic executions ϵ_1 of A_1 and ϵ_2 of A_2 such that s_1 = $(lstate(\epsilon_1))^{ab}$, s_2 = $(lstate(\epsilon_2))^{ab}$, $\epsilon_1 \mathcal{E}(R) \epsilon_2$ and $tdist(\epsilon_1)$ = $tdist(\epsilon_2)$. We show that R^{ab} is a simulation relation from A_1^{ab} to A_2^{ab}.

For the start condition, by the start condition of R, $\delta(\overline{q}^1) R \delta(\overline{q}^2)$, and then $\delta(\overline{q}^1) \mathcal{E}(R) \delta(\overline{q}^2)$. On the other hand, $\overline{q_i}^{ab}$ = $lstate(\delta(\overline{q_i}))^{ab}$. Thus, $\overline{q_1}^{ab} R^{ab} \overline{q_2}^{ab}$. For the step condition, let $s_1 R^{ab} s_2$. That is, s_i = $(lstate(\epsilon_i))^{ab}$, $\epsilon_1 \mathcal{E}(R) \epsilon_1$, and $tdist(\epsilon_1)$ = $tdist(\epsilon_2)$.

We assume that $s_1 \stackrel{[\rho_1']}{\rightarrow} s_1'$. Let ρ_1' = $T_1^1 \cdots T_m^1$. By repeatedly using the results of Lemma 2, we have a finite scheduler ρ_1' and write ϵ_1' for $apply(\epsilon, \rho_1')$ and ϵ_2' for $apply(\epsilon, \rho_2')$, such that $\epsilon_1' \mathcal{E}(R)) \epsilon_2'$ and $tdist(\epsilon_1')$=$tdist(\epsilon_1')$. From $tdist(\epsilon_1)$ = $tdist(\epsilon_2)$ and $tdist(\epsilon_1')$=$tdist(\epsilon_1')$, $[\rho_1'].dist$ = $[\rho_2'].dist$. Moreover, because ρ_1' and ρ_2' are unit schedulers, $[\rho_1'].name$ = $[\rho_1'].name$. It is also easily shown that s_1' = $lstate(\epsilon_1')^{ab}$. Therefore, if we let s_2' = $lstate(\epsilon_2')^{ab}$, then s_2 has an observably identical transition to s_2' as $s_1 \stackrel{[\rho_1']}{\rightarrow} s_1'$ and $s_1' R^{ab} s_2'$ as desired.

3.5 Verification procedure

We prove the existence of a simulation from A_1^{ab} to A_2^{ab} by bisimulation. The basic idea is that $X \subset Y$ is implied by equation X=$X \cap Y$ for sets X and Y. We achieve this for transition systems by the following two system syntheses.

The first is the synchronization of two systems: $A \cap B$ for nondeterministic transition systems A and B. The set of the states of $A \cap B$ is the product of the sets of the states of A and B. The set of the actions of $A \cap B$ is the intersection of the sets of the actions of A and B. The transition of $A \cap B$ is the application of a common action to a pair of states: if $s_a \stackrel{a}{\rightarrow} s_a'$ in A and $s_b \stackrel{a}{\rightarrow} s_b'$ in B, then $(s_a, s_a) \stackrel{a}{\rightarrow} (s_a', s_b')$ in $A \cap B$. The second is the disjoint union of two systems: A+B. The set of the states of A+B is the disjoint union of the sets of the states of A and B. The set of actions is the union of the sets of the actions of A and B. The transition is obtained by individually applying the actions of A or B to their states: if $s_a \in A$, $s_a \stackrel{a}{\rightarrow} s_a'$ and $(s_a, A) \in A$+B, then $(s_a, A) \stackrel{a}{\rightarrow} (s_a', A)$, and the case of $(s_b, B) \in A$+B is similar.

With these syntheses, we obtain the following results.

Claim: Let $init_A$ and $init_B$ be the initial states of A and B, respectively. Then $(init_A, A)$ is bisimilar to $((init_A, init_B), A \cap B)$ in system A+$(A \cap B)$, if and only if there is a simulation from A to B.

Now, by Theorem 4 and the Claim, we prove $A_1 \leq_0 A_2$ by showing that $(\overline{q}_1^{ab}, A_1^{ab})$ is bisimilar to $((\overline{q}_1^{ab}, \overline{q}_2^{ab}), A_1^{ab} \cap A_2^{ab})$ in system A_1^{ab}+$(A_1^{ab} \cap A_2^{ab})$, using the partition refinement algorithm.

4 Implementation

We implemented our method in Java and represented the system state transitions and the set of states, "partitions," by Binary Decision Diagram (BDD). We applied the method to the example of oblivious transfer protocol in [6]. It successfully output the results in 13 minutes and 8 seconds on an Apple Power PC G5 2.5 GHz dual with 512 MB, creating 1,911 partitions and 1,497,102 BDD nodes.

5 Discussion and Future work

For our verification method, we posed some restrictions on the targeting systems. Among them, the restriction on the preconditions of actions is strong and excludes a few important protocols such as mutual authentications where nonces are compared by the excluded condition: $x_1 = x_2$. To include such preconditions, we have to refine the classes of random values, which are currently $\{\top, \bot\}$, by these equations in preconditions. Such a method of introducing proper equivalence classes requires more careful analysis on the relations of classification and distribution. This important extension remains future work.

For verification, an alternative approach might directly employ the decision algorithm of probabilistic bisimulation

[9]. However, that algorithm can be intractable in proving weak bisimulation relations. Moreover, the number of actions in an original probabilistic system is huge if we do not abstract away the random values or introduce a coarse equivalence relation on them.

Our procedure automatizes the proof of perfect implementation relation $\leq_0$ for a class of security protocols. To make the proof of a security protocol under computational assumptions fully automatic, we have to introduce a systematic way of splitting the implementation of the original simulatability and designing simulators for implementation. The second simulator problem is an especially challenging future work.

Acknowledgments
We thank Tatsuaki Okamoto for arranging this research collaboration. We also appreciate Roberto Segala's helpful comments on our method.

References

[1] M. Backes, B. Pfitzmann, and M. Waidner, "Secure Asynchronous Reactive Systems," *Cryptology ePrint Archive, Report 2004/082,*, 2004, http://eprint.iacr.org/.

[2] G. Barthe, J. Cerderquist, and S. Tarento, "A Machine-Checked Formalization of the GM and the ROM," *Automated Reasoning: Second International Joint Conference, IJCAR 2004*, pp. 385–399, 2004.

[3] M. Bellare and P. Rogaway, "The game-playing technique and its application to triple encryption," Cryptology ePrint Archive, Report 2004/331, 2004.

[4] B. Blanchet, "A Computationally Sound Mechanized Prover for Security Protocols," *IEEE Symposium on Security and Privacy*, pp. 140-154, 2006.

[5] R. Canetti, "Universally Composable Security: A New Paradigm for Cryptographic Protocols," *FOCS 2001*, pp. 136-145, 2001.

[6] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, gUsing task-structured probabilistic I/O automata to analyze an oblivious transfer," *MIT CSAIL Technical Reports - MIT-CSAIL-TR-2006-047*, 2006.

[7] R. Canetti, L. Cheung, D. Kaynar, M. Liskov, N. Lynch, O. Pereira, and R. Segala, gAnalyzing Security Protocols Using Time-Bounded Task-PIOAs,h *Discrete Event Dynamic Systems*, vol. 18, num. 1, issn 0924-6703, pp. 111–159, Kluwer Academic Publishersm 2008.

[8] R. Canetti, L. Cheung, N. Lynch, and O. Pereira, "On the Role of Scheduling in Simulation-Based Security," *7th International Workshop on Issues in the Theory of Security (WITS'07)*, pp. 22–37, 2007.

[9] S. Cattani and R. Segala, "Decision Algorithms for Probabilistic Bisimulation," *CONCUR 2002*, pp. 371-385, 2002.

[10] L. Cheung, S. Mitra, and O. Pereira, "Verifying Statistical Zero Knowledge with Approximate Implementations," *Cryptology ePrint Archive, Report 2007/195*, 2007.

[11] K.-K. Raymond Choo, "Refuting Security Proofs for Tripartite Key Exchange with Model Checker in Planning Problem Setting," *19th IEEE Computer Security Foundations Workshop - CSFW 2006*, pp. 297–308, 2006.

[12] N. Lynch, R. Segala, and F. Vaandrager "Observing Branching Structure through Probabilistic Contexts," *SIAM Journal on Computing 2007*, Vol. 37, Issue 4, pp. 977–1013, 2007.

[13] J.C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague, "A Probabilistic Polynomial-time Process Calculus for the Analysis of Cryptographic Protocols," *Theoretical Computer Science 353*, pp. 118-164, 2006.

[14] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Report 2004/332, 2004.

[15] C. Sprenger, M. Backes, D. Basin, B. Pfitzmann, and M. Waidner, "Cryptographically Sound Theorem Proving," *19th IEEE Computer Security Foundations Workshop - CSFW 2006*, pp. 153-166, 2006.