INFORMATION FLOW ANALYSIS FOR PROBABILISTIC TIMED AUTOMATA

Ruggero Lanotte[1], Andrea Maggiolo–Schettini[2] and Angelo Troina [2]
[1]*Dipartimento di Scienze della Cultura, Politiche e dell'Informazione – Università dell'Insubria*
[2]*Dipartimento di Informatica – Università di Pisa*

Abstract In multilevel systems it is important to avoid unwanted indirect information flow from higher levels to lower levels, namely the so called *covert channels*. Initial studies of information flow analysis were performed by abstracting away from time and probability. Recently, work has been done in order to consider also aspects either of time or of probability, but not both. In this paper we propose a general framework, based on Probabilistic Timed Automata, where both *probabilistic* and *timing covert channels* can be studied. As an application, we study a system with covert channels that we are able to discover by our techniques.

1. Introduction

In a multilevel system every agent is confined in a bounded security level; information can flow from a certain agent to another agent only if the level of the former is lower than the level of the latter. Access rules can be imposed by the system in order to control direct unwanted transmission from higher levels to lower levels; however, it could be possible to transmit information indirectly by using system side effects. Usually, this kind of indirect transmissions, called *covert channels*, do not violate the access rules imposed by the system.

The existence of covert channels has led to the more general approach of *information flow security*, which aims at controlling the way information may flow among different entities. The idea is to try to directly control the whole flow of information, rather than only the direct communication among agents. In [10] the authors introduce the notion of *non-interference*, stating, intuitively, that the low level agents should not be able to deduce anything about the activity of the high level agents. By imposing some information flow rules, it is possible to control direct and indirect leakages, as both of them give rise to unwanted information flows.

In the literature, there are many different definitions of security based on the information flow idea, and each one formulated in some system model (see, e.g., [1, 5, 7–11, 16, 19]). Most of the properties considered are based

on analysis of information flow that does not take into consideration aspects of time or probability, and therefore are not useful to check the existence of probabilistic or timing covert channels. To overcome this, a significant work has been done in order to extend the study by considering either time (see, e.g., [5, 7, 9]) or probability (see, e.g., [1, 11, 19]), but, to the best of our knowledge, not both.

In this paper we propose a general framework where both probabilistic and timing covert channels can be studied. For the description of systems we choose the model of Probabilistic Timed Automata. Timed Automata have been introduced by Alur and Dill [3], extensions with probability have been proposed e.g. in [2, 6, 13, 14]. We introduce a particular class of Probabilistic Timed Automata (PTA) well-suited for the analysis of information flow security properties.

The framework of PTA allows one to specify timed systems showing a probabilistic behavior in an intuitive and succinct way. Therefore, within the framework of PTA, where time and probabilities are taken into consideration, the modeler can describe, in the same specification, different aspects of a system and analyze on a single model real-time properties, performance and reliability properties (by using classical model checking techniques) and finally information flow security properties able to detect both probabilistic and timing covert channels.

2. Probabilistic Timed Automata

Let us assume a set X of positive real variables called *clocks*. A *valuation* over X is a mapping $v : X \to \mathbb{R}^{\geq 0}$ assigning real values to clocks. For a valuation v and a time value $t \in \mathbb{R}^{\geq 0}$, let $v + t$ denote the valuation such that $(v + t)(x) = v(x) + t$, for each clock $x \in X$.

The set of *constraints* over X, denoted $\Phi(X)$, is defined by the following grammar: $\phi ::= x \sim c \mid \phi \wedge \phi \mid \neg\phi \mid \phi \vee \phi \mid true$, where ϕ ranges over $\Phi(X)$, $x \in X$, $c \in \mathbb{Q}$ and $\sim \in \{<, \leq, =, \neq,$
$>, \geq\}$.

We write $v \models \phi$ when *the valuation v satisfies the constraint ϕ*. Formally, $v \models x \sim c$ iff $v(x) \sim c$, $v \models \phi_1 \wedge \phi_2$ iff $v \models \phi_1$ and $v \models \phi_2$, $v \models \neg\phi$ iff $v \not\models \phi$, $v \models \phi_1 \vee \phi_2$ iff $v \models \phi_1$ or $v \models \phi_2$, and $v \models true$.

Let $B \subseteq X$; with $v[B]$ we denote the valuation resulting after resetting all clocks in B. More precisely, $v[B](x) = 0$ if $x \in B$, $v[B](x) = v(x)$, otherwise. Finally, with $\mathbf{0}$ we denote the valuation with all clocks reset to 0, namely $\mathbf{0}(x) = 0$ for all $x \in X$.

DEFINITION 1 *A Probabilistic Timed Automaton (PTA) is a sixtuple $A = (\Sigma, X, Q, q_0, \delta, \pi)$, where:*

- *Σ is a finite alphabet of actions.*

- *X is a finite set of positive real variables called clocks.*
- *Q is a finite set of states and $q_0 \in Q$ is the initial state.*
- *$\delta \subseteq Q \times \Sigma \cup \{\tau\} \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions. The symbol τ represents the silent or internal move. For a state q, we denote with $start(q)$ the set of transitions with q as source state, i.e. the set $\{(q_1, a, \phi, B, q_2) \in \delta \mid q_1 = q\}$.*
- *$\pi : \delta \rightarrow]0, 1]$ is a probability function such that $\pi(e)$ is the probability of performing the transition e. We require that $\sum_{e \in start(q)} \pi(e) = 1$.*

A *configuration* of A is a pair (q, v), where $q \in Q$ is a state of A, and v is a valuation over X. The initial configuration of A is represented by $(q_0, \mathbf{0})$ and the set of all the configurations of A is denoted with $\mathcal{S}_A$.

There is a discrete *transition step* from a configuration $s_i = (q_i, v_i)$ to a configuration $s_j = (q_j, v_j)$ through action $a \in \Sigma \cup \{\tau\}$, written $s_i \xrightarrow{a} s_j$, if there is a transition $e = (q_i, a, \phi, B, q_j) \in \delta$ such that $v_i \models \phi$, and $v_j = v_i[B]$.

There is a continuous *timed step* from a configuration $s_i = (q_i, v_i)$ to a configuration $s_j = (q_j, v_j)$ through time $t \in \mathbb{R}^{>0}$, written $s_i \xrightarrow{t} s_j$, if $q_j = q_i$ and $v_j = (v_i + t)$.

Given a configuration $s = (q_i, v_i)$, with $Adm(s) = \{(q_i, a, \phi, B, q) \in \delta \mid v_i \models \phi\}$ we represent the set of transitions that an automaton could execute from configuration s, and we say that a transition in $Adm(s)$ is *enabled* in s. Given two configurations $s_i = (q_i, v_i)$, $s_j = (q_j, v_j)$ and given $a \in \Sigma \cup \{\tau\}$ we represent with $Adm(s_i, a, s_j) = \{(q_i, a, \phi, B, q_j) \in \delta \mid v_i \models \phi \wedge v_j = v_i[B]\}$ the set of transitions that lead from configuration s_i to configuration s_j through a transition step labeled with a. A configuration $s = (q_i, v_i)$ is called *terminal* iff $Adm(s') = \emptyset$ for all $s' = (q_i, v_i + t)$ where $t \in \mathbb{R}^{\geq 0}$; we denote with S_T the set of terminal configurations.

For configurations $s_i = (q_i, v_i)$, $s_j = (q_j, v_j)$ and $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$, we define with $P(s_i, \alpha, s_j)$ the probability of reaching configuration s_j from configuration s_i through a step $s_i \xrightarrow{\alpha} s_j$ labeled with α. Formally $P(s_i, \alpha, s_j) = \frac{\sum_{e \in Adm(s_i, \alpha, s_j)} \pi(e)}{\sum_{e \in Adm(s_i)} \pi(e)}$ if $\alpha \in \Sigma \cup \{\tau\}$ and $P(s_i, \alpha, s_j) = 1$ if $\alpha \in \mathbb{R}^{>0}$.

The probability of executing a transition step from a configuration s is chosen according to the values returned by the function π among all the transitions enabled in s, while we set to 1 the probability of executing a timed step labeled with $t \in \mathbb{R}^{>0}$. Intuitively, an automaton chooses non-deterministically whether to execute a transition step (selected probabilistically among all the transitions enabled in s) or to let time pass performing a timed step.

An *execution fragment* starting from s_0 is a finite sequence of timed and transition steps $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots \xrightarrow{\alpha_k} s_k$. With $ExecFrag$

we denote the set of execution fragments and with $ExecFrag(s)$ the set of execution fragments starting from s. We define $last(\sigma) = s_k$ and $|\sigma| = k$. The execution fragment σ is called *maximal* iff $last(\sigma) \in S_T$. For any $j < k$, with σ^j we define the sequence of steps $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_j} s_j$. If $|\sigma| = 0$ we put $P(\sigma) = 1$, else, if $|\sigma| = k \geq 1$, we define $P(\sigma) = P(s_0, \alpha_1, s_1) \cdot \ldots \cdot P(s_{k-1}, \alpha_k, s_k)$.

An *execution* is either a maximal execution fragment or an infinite sequence $s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$. We denote with $Exec$ the set of executions and with $Exec(s)$ the set of executions starting from s. Finally, let $\sigma \uparrow$ denote the set of executions σ' such that $\sigma \leq_{prefix} \sigma'$, where *prefix* is the usual prefix relation over sequences.

Executions and execution fragments of a PTA arise by resolving both the nondeterministic and the probabilistic choices [13]. To resolve the nondeterministic choices of a PTA, we introduce now *schedulers* of PTAs.

A *scheduler* of a PTA $A = (\Sigma, X, Q, q_0, \delta, \pi)$ is a partial function F from $Exec$ to $\mathbb{R}^{>0}$. For a scheduler F of a PTA A we define $ExecFrag^F$ (resp. $Exec^F$) as the set of execution fragments (resp. executions) $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \ldots$ such that $\alpha_i \in \mathbb{R}^{>0}$ iff $F(\sigma^{i-1}) = \alpha_i$, for any $0 < i < |\sigma|$. We note that, if $F(\sigma)$ is not defined, then a discrete step is chosen for σ. Namely, $\alpha_i \notin \mathbb{R}^{>0}$ iff $F(\sigma^{i-1})$ undefined. A scheduler should also respect the *nonZeno* condition of divergent times. Formally we have that for any infinite sequence $\sigma = s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} \ldots$ in $Exec^F$ the sum $\sum_{\alpha_i \in \mathbb{R}^{>0}} \alpha_i$ diverges.

Assuming the basic notions of probability theory (see e.g. [12]) we define the probability space on the executions starting in a given configuration $s \in \mathcal{S}_A$ as follows. Given a scheduler F, let $Exec^F(s)$ be the set of executions starting in s, $ExecFrag^F(s)$ be the set of execution fragments starting in s, and $\Sigma^F_{Field}(s)$ be the smallest sigma field on $Exec^F(s)$ that contains the basic cylinders $\sigma \uparrow$, where $\sigma \in ExecFrag^F(s)$. The probability measure $Prob^F$ is the unique measure on $\Sigma^F_{Field}(s)$ such that $Prob^F(\sigma \uparrow) = P(\sigma)$.

In the following, A is a PTA, F is a scheduler for A, $\hat{\alpha}$ stands for α if $\alpha \in \Sigma \cup \mathbb{R}^{>0}$ and for ε (the empty string) if $\alpha = \tau$, $s \in \mathcal{S}_A$ and $\mathcal{C} \subseteq \mathcal{S}_A$.

Consider now $Exec^F(\tau^*\hat{\alpha}, \mathcal{C})$, the set of executions that lead to a configuration in $\mathcal{C}$ via a sequence in $\tau^*\hat{\alpha}$. We define $Exec^F(s, \tau^*\hat{\alpha}, \mathcal{C}) = Exec(\tau^*\hat{\alpha}, \mathcal{C}) \cap Exec^F(s)$. Finally, given a scheduler F, we define the probability $Prob^F(s, \tau^*\hat{\alpha}, \mathcal{C}) = Prob^F(Exec^F(s, \tau^*\hat{\alpha}, \mathcal{C}))$ as:

$$\begin{cases} 1 & if\ \alpha = \tau \wedge s \in \mathcal{C} \\ \sum_{q \in \mathcal{S}_A} Prob^F(s, \tau, q) \cdot Prob^F(q, \tau^*, \mathcal{C}) & if\ \alpha = \tau \wedge s \notin \mathcal{C} \\ \sum_{q \in \mathcal{S}_A} Prob^F(s, \tau, q) \cdot Prob^F(q, \tau^*\alpha, \mathcal{C}) + Prob(s, \alpha, \mathcal{C}) & if\ \alpha \neq \tau \end{cases}$$

Weak bisimulation

The *bisimulation* of a system by another system is based on the idea of mu-

tual step-by-step simulation. Intuitively, two systems A and A' are bisimilar, if whenever one of the two systems executes a certain action and reaches a configuration s, the other system is able to simulate this single step by executing the same action and reaching a configuration s' which is again bisimilar to s. A *weak bisimulation* is a bisimulation which does not take into account τ (internal) moves. Hence, whenever a system simulates an action of the other system, it can also execute some internal τ actions before and after the execution of that action. A *branching bisimulation* is, instead, a weak bisimulation where τ moves are allowed only before the execution of the action to simulate.

To abstract away from τ moves, Milner [18] introduces the notion of observable step, which consists of a single *visible* action α preceded and followed by an arbitrary number (including zero) of internal moves. Such moves are described by a *weak* transition relation $\stackrel{\alpha}{\Longrightarrow} = (\stackrel{\tau}{\longrightarrow})^* \stackrel{\alpha}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^*$, where $\longrightarrow$ is the classical strong relation, and $\stackrel{\tau}{\Longrightarrow} = (\stackrel{\tau}{\longrightarrow})^*$. It is worth noting that with such a definition a weak internal transition $\stackrel{\tau}{\Longrightarrow}$ is possible even without performing any internal action.

For the definition of weak bisimulation in the fully probabilistic setting, Baier and Hermanns [4] replace Milner's weak internal transitions $s \stackrel{\tau}{\Longrightarrow} s'$ by the probability $Prob(s, \tau^*, s')$ of reaching configuration s' from s via internal moves. Similarly, for visible actions α, Baier and Hermanns define $\stackrel{\alpha}{\Longrightarrow}$ by means of the probability $Prob(s, \tau^*\alpha, s')$.

DEFINITION 2 *Let $A = (\Sigma, X, Q, q_0, \delta, \pi)$ be a probabilistic timed automaton. A* weak bisimulation *on A is an equivalence relation $\mathcal{R}$ on $\mathcal{S}_A$ such that, for all $(s, s') \in \mathcal{R}$, $\mathcal{C} \in \mathcal{S}_A/\mathcal{R}$ and schedulers F, there exists a scheduler F' such that $Prob_A^F(s, \tau^*\alpha, \mathcal{C}) = Prob_A^{F'}(s', \tau^*\alpha, \mathcal{C})$ for every $\alpha \in \Sigma \cup \{\tau\} \cup \mathbb{R}^{>0}$, and vice versa.*
Two configurations s, s' are called weakly bisimilar *on A (denoted $s \approx_A s'$) iff $(s, s') \in \mathcal{R}$ for some weak bisimulation $\mathcal{R}$.*

DEFINITION 3 *Two probabilistic timed automata $A = (\Sigma, X, Q, q_0, \delta, \pi)$ and $A' = (\Sigma', X', Q', q_0', \delta', \pi')$ such that $Q \cap Q' = \emptyset$ and $X \cap X' = \emptyset$ are called weak bisimilar (denoted by $A \approx A'$) if, given the probabilistic timed automaton $\hat{A} = (\Sigma \cup \Sigma', X \cup X', Q \cup Q', q_0, \delta \cup \delta', \hat{\pi})$, with $\hat{\pi}(e) = \pi(e)$ if $e \in \delta$ and $\hat{\pi}(e) = \pi'(e)$ otherwise, it holds $(q_0, \mathbf{0}) \approx_{\hat{A}} (q_0', \mathbf{0})$, where the valuation $\mathbf{0}$ is defined over all clocks of the set $X \cup X'$.*

In [15] we have given an algorithm that resorts to the theory of regions of timed automata [3] in order to decide weak bisimulation. Along the line of [15] we derive the following proposition.

PROPOSITION 4 *It is decidable to check whether two configurations or two probabilistic timed automata are weak bisimilar.*

Auxiliary Operators for Probabilistic Timed Automata

We assume two probabilistic timed automata $A_1 = (\Sigma, X_1, Q_1, r_0, \delta_1, \pi_1)$ and $A_2 = (\Sigma, X_2, Q_2, u_0, \delta_2, \pi_2)$ with $Q_1 \cap Q_2 = \emptyset$ and $X_1 \cap X_2 = \emptyset$. We also assume a set $L \subseteq \Sigma$ of synchronization actions. Finally, given a transition $e = (q, a, \phi, B, q') \in \delta_i$, with $\pi_{i_a}(e)$ we denote the normalized probability of executing transition e with respect to all other transitions starting from q and labelled with a, i.e. $\pi_{i_a}(e) = \frac{\pi_i(e)}{\sum_{e' \in start_i^a(q)} \pi_i(e')}$, where $start_i^a(q)$ denotes the set of transitions in δ_i with q as source state and a as labelling action, i.e. the set $\{(q_1, a', \phi, B, q_2) \in \delta_i \mid q_1 = q \wedge a' = a\}$.

DEFINITION 5 *The* parallel composition *of two PTA A_1 and A_2, with respect to the synchronization set L and the advancing speed parameter $p \in]0,1[$, is defined as $A_1 ||_L^p A_2 = (\Sigma, X, Q, (r_0, u_0), \delta, \pi)$. The set Q of states of $A_1 ||_L^p A_2$ is given by the cartesian product $Q_1 \times Q_2$ of the states of the two automata A_1 and A_2, while the set of clocks X is given by the union $X_1 \cup X_2$. Given a state (r, u) of $A_1 ||_L^p A_2$, δ and π are obtained by the following rules:*

- *If from state r the automaton A_1 has a transition $e_1 = (r, a, \phi, B, r')$ with action $a \notin L$ and probability $\pi_1(e_1) = p'$, then $A_1 ||_L^p A_2$ has a transition $e = ((r, u), a, \phi, B, (r', u)) \in \delta$ with probability $\pi'(e) = p \cdot p'$.*
- *If from state u the automaton A_2 has a transition $e_2 = (u, a, \phi, B, u')$ with action $a \notin L$ and probability $\pi_2(e_2) = p'$, then $A_1 ||_L^p A_2$ has a transition $e = ((r, u), a, \phi, B, (r, u')) \in \delta$ with probability $\pi'(e) = (1 - p) \cdot p'$.*
- *If from state r the automaton A_1 has a transition $e_1 = (r, a, \phi_1, B_1, r')$ with action $a \in L$ and probabilities $\pi_1(e_1) = p'$ and $\pi_{1_a}(e_1) = \hat{p}'$, and from state u the automaton A_2 has a transition $e_2(u, a, \phi_2, B_2, u')$ with probabilities $\pi_2(e_2) = p''$ and $\pi_{2_a}(e_2) = \hat{p}''$, A_1 and A_2 can synchronize and therefore $A_1 ||_L^p A_2$ has a transition $e = ((r, q), \tau, \phi_1 \wedge \phi_2, B_1 \cup B_2, (r', q')) \in \delta$ with probability $\pi'(e) = p \cdot p' \cdot \hat{p}'' + (1 - p) \cdot p'' \cdot \hat{p}'$.*
- *For all $e = (q, a, \phi, B, q') \in \delta$, $\pi(e) = \frac{\pi'(e)}{\sum_{e' \in \delta \cap start(q)} \pi'(e')}$.*

Given such a definition of parallel composition, whenever A_1 and A_2 synchronize they give rise to an internal action τ. Note that, chosen a transition e_1 (e_2) with label $a \in L$ of automaton A_1 (A_2) the transition e_2 (e_1) of A_2 (A_1) that synchronizes with e_1 (e_2) is chosen according to the probability $\pi_{2_a}(e_2)$ ($\pi_{1_a}(e_1)$) normalized with respect to all the other transitions labelled with a. Besides, according to Definition 1, it holds that $\sum_{e \in start(q)} \pi(e) \in \{0, 1\}$ for each state q of $A_1 ||_L^p A_2$. This is done due to the last rule, that uses the auxiliary structure π' to compute the normalized probabilities in π.

PROPOSITION 6 *Given PTA A_1 and A_2, $A_1||^p_L A_2$ is a PTA for all $p \in]0,1[$ and $L \subseteq \Sigma$.*

We now assume $A = (\Sigma, X, Q, q_0, \delta, \pi)$ and $L \subseteq \Sigma$.

DEFINITION 7 *The* restriction *of a probabilistic timed automaton A with respect to the set of actions L is given by $A \setminus L = (\Sigma, X, Q, q_0, \delta', \pi')$ where $\delta' = \delta \setminus \{(q, a, \phi, B, q') \mid a \in L\}$ and $\pi'(e) = \frac{\pi(e)}{\sum_{e' \in \delta' \cap start(q)} \pi(e')}$ for all $e = (q, a, \phi, B, q') \in \delta'$.*

DEFINITION 8 *The* hiding *of a probabilistic timed automaton A with respect to the set of actions L is given by $A/L = (\Sigma, X, Q, q_0, \delta', \pi)$ where each transition $e = (q, a, \phi, B, q')$ with $a \in L$ is replaced by the transition $e' = (q, \tau, \phi, B, q')$, where $\pi(e') = \pi(e)$.*

PROPOSITION 9 *Given a PTA A, $A \setminus L$ and A/L are PTA for all $L \subseteq \Sigma$.*

3. Security Properties

A multilevel system interacts with agents confined in different levels of clearance. In order to analyze the information flow between parties with different levels of confidentiality, the set of visible actions is partitioned into high level actions and low level actions. Formally, we assume the set of possible actions $\Sigma = \Sigma_H \cup \Sigma_L$, with $\Sigma_H \cap \Sigma_L = \emptyset$. In the following, with $l, l' \ldots$ and $h, h', \ldots$ we denote actions of Σ_L and Σ_H respectively. With Γ_H and Γ_L we denote the set of high level agents and low level agents. Formally, an automaton $A = (\Sigma', X, Q, q_0, \delta, \pi)$ is in Γ_H (Γ_L) if $\Sigma' \subseteq \Sigma_H$ ($\Sigma' \subseteq \Sigma_L$). For simplicity, we specify only two-level systems. More levels can by dealt with by iteratively grouping them in two clusters.

A low level agent is able to observe the execution of all the steps labelled with actions in Σ_L and all the timed steps. The basic idea of non-interference is that the high level does not interfere with the low level if the effects of high level communications are not visible by a low level agent. Finally, an important assumption when dealing with non-interference analysis is that a system is considered to be *secure* (no information flow can occur) if there is no interaction with high level agents (if high level actions are prevented).

Probabilistic Timed Non-interference

We say that a probabilistic timed automaton A satisfies the *Probabilistic Timed Non-interference* property (*PTNI*) if high level agents are not able to interfere with the observable behavior of the system from the low level point of view. Formally *PTNI* can be formulated as follows.

DEFINITION 10 *A PTA A is PTNI secure ($A \in PTNI$) $\Leftrightarrow A/\Sigma_H \approx A \setminus \Sigma_H$.*

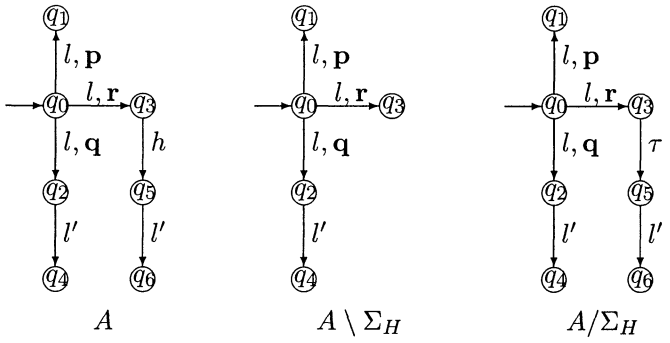

Figure 1. A probabilistic covert channel.

$A \setminus \Sigma_H$ represents the isolated system, where all high level actions are prevented. Such a system is considered secure due to the notion of non-interference. If the observational behavior of the isolated system is equal to the behavior of A/Σ_H, representing the system which communicates with high level agents in an invisible manner for the low agents point of view, PTA A is *PTNI* secure. This property, defined in an environment where both probability and time are studied, is able to catch information flow that may occur either due to the probabilistic behavior of the system or due to the time when actions occur.

PROPOSITION 11 *It is decidable to check whether a PTA $A \in$ PTNI.*

Proof. By the decidability of our weak bisimulation (see Proposition 4) and by the computable definitions of the operators of hiding and restriction. ∎

EXAMPLE 12 *In Figure 1 we show a case of probabilistic information flow. Abstracting away from probability, the system A could be considered secure (in a purely nondeterministic setting, in both A/Σ_H and $A \setminus \Sigma_H$ a low level agent can observe the action l or the sequence ll' without further information about the execution of h). In a probabilistic framework, given $\mathbf{p} + \mathbf{r} + \mathbf{q} = 1$, action h interferes with the probability of observing either a single l or the sequence ll'. Formally, in $A \setminus \Sigma_H$, a low level agent observes either the single l with probability $\mathbf{p} + \mathbf{r}$ or the sequence ll' with probability $\mathbf{q}$. However, in A/Σ_H the event l is observed with probability $\mathbf{p}$ and the sequence ll' with probability $\mathbf{r} + \mathbf{q}$. As a consequence, $A/\Sigma_H \not\approx A \setminus \Sigma_H$, so that the* PTNI *property reveals the probabilistic covert channel.*

EXAMPLE 13 *In Figure 2 we show a case of timing information flow. Abstracting away from time, the system A could be considered secure (in a purely untimed nondeterministic setting, in both A/Σ_H and $A \setminus \Sigma_H$ a low level agent*

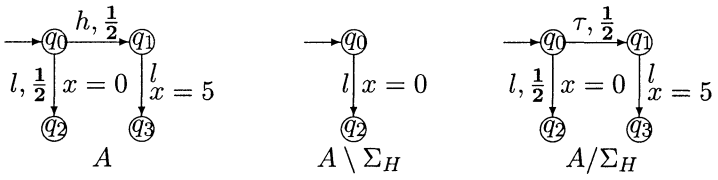

Figure 2. A timing covert channel.

can observe only the action l without further information about the execution of h). In a timed framework, given a clock $x \in \mathbb{R}^{\geq 0}$, the high action h interferes with the time of observing action l. Formally, in $A \setminus \Sigma_H$, a low level agent observes l executed immediately, while in A/Σ_H l could be either observed immediately or when the clock x reaches value 5. A low level agent, observing the event l when clock x has value 5 knows that action h has occurred. As a consequence, $A/\Sigma_H \not\approx A \setminus \Sigma_H$, so that the PTNI *property reveals the timing covert channel.*

Probabilistic Timed Non Deducibility on Composition

In [8] Focardi and Gorrieri promote the classification of a set of properties capturing the idea of information flow and non-interference. The *Non Deducibility on Composition* property (*NDC*) states that a system A in isolation has not to be altered when considering all the potential interactions of A with the high level agents of the external environment. We consider a notion of *NDC* called *Probabilistic Timed Non Deducibility on Composition* (*PTNDC*).

DEFINITION 14 *A PTA A is PTNDC secure ($A \in PTNDC$)* $\Leftrightarrow \forall \Pi \in \Gamma_H, \forall p \in]0,1[, \forall L \subseteq \Sigma_H \quad A/\Sigma_H \approx (A||_L^p \Pi) \setminus \Sigma_H$.

As we have seen, A/Σ_H represents the observable behavior of A from a low level agent point of view (i.e. the isolated system where all high level actions are hidden). System $(A||_L^p \Pi) \setminus \Sigma_H$ represents, instead, system A communicating with the high agent Π and then prevented by the execution of other high level actions. If the observational behavior of the isolated system is equal to the behavior of the system communicating with any high level agent, PTA A satisfies the *PTNDC* security property.

THEOREM 15 $A \in PTNDC \Rightarrow A \in PTNI$.

Proof. Consider $\Pi = (\emptyset, \emptyset, \{q\}, q, \emptyset, \pi) \in \Gamma_H$, i.e. an automaton representing a high level agent which does not perform any transition, and consider then the set $L = \emptyset$. If PTA A is *PTNDC*, then $\forall p \in]0,1[, \ A/\Sigma_H \approx (A||_L^p \Pi) \setminus \Sigma_H$. Now, by the definition of parallel composition, $(A||_L^p \Pi) = A$ and, therefore, $A/\Sigma_H \approx A \setminus \Sigma_H$, stating that $A \in PTNI$. ∎

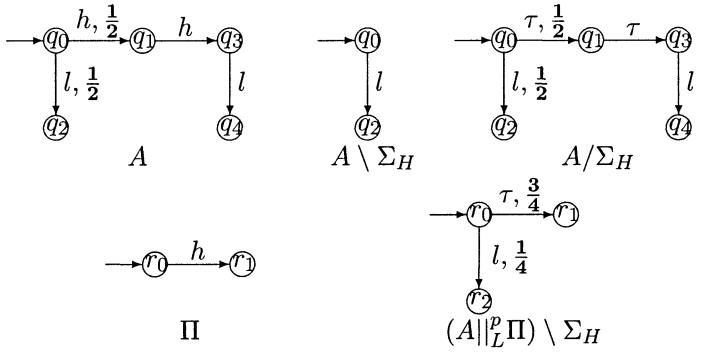

Figure 3. $A \in$ *PTNI*, but $A \notin$ *PTNDC*.

EXAMPLE 16 *Consider the PTA A of Figure 3. It is easy to see that A is* PTNI *secure, since $A/\Sigma_H \approx A \setminus \Sigma_H$. In both A/Σ_H and $A \setminus \Sigma_H$, a low level agent observes the single event l taken with probability* 1. *If we consider, instead, the high level agent Π of Figure 3, the set $L = \{h\}$ and $p = \frac{1}{2}$, we observe that $A/\Sigma_H \not\approx (A||_L^p \Pi) \setminus \Sigma_H$. In fact, A/Σ_H always performs action l with probability* 1, *while $(A||_L^p \Pi) \setminus \Sigma_H$ reaches a deadlock state r_1 and does not perform any visible action with probability $\frac{3}{4}$ (as it turns out after the parallel composition of A and Π). As a consequence, automaton A is not* PTNDC *secure.*

Theorem 15 and Example 16 show that the *PTNI* property is not able to detect some potential deadlock due to high level activities, exactly as put in evidence in [8]. For this reason we resort to the *PTNDC* property, which implies *PTNI*, in order to capture these finer undesirable behaviors.

It is worth noticing that, as it happens for the analogous properties defined in [1, 8, 9], the above definition of the *PTNDC* property is difficult to use in practice because of the universal quantification on the high level agents. Decidability of *PTNDC* depends, in fact, on the possibility of reducing all the high level automata in Γ_H to a finite case suitable for the particular automaton A we would like to study.

4. An Application

As an application we consider a network device, also studied in [9] in a timed framework, that manages, following a mutual exclusion policy, the access to a shared buffer. Assuming that the agents on the network are classified as low and high level agents, the device implements the *no-write-down no-read-up* policy [10]. Intuitively, the policy states that high level users can only read the buffer, while low level users can only write on it. Such a policy avoids direct

information flow from high level to low level, however malicious agents can exploit some covert channel in order to transmit information indirectly. For example, a low level user could get information about the high level activity by observing the amount of time the device is locked (non accessible by the low level) when high agents are reading, or by observing the frequencies with which high level agents make access on it. We would like to check whether some covert channel can be exploited by giving a specification of the network device, and then by checking the *PTNDC* property.

In the following we consider only a low level user and a high level user communicating with the network device. We assume that the low level user is always ready to write in the buffer, so we consider an agent that infinitely waits for a grant from the device and then writes in the buffer. In this manner we are considering a low level user that continuously monitors the activity of the device. We also assume that the entire procedure of receiving a grant in the network and writing in the buffer is executed in a time n. In Figure 4, we model the specification of a simple device (see the PTA B). Actions req_H, $read_H$, $grant_L$ and $write_L$ model respectively high level read requests, high level reads, low level write grants and low level writes. The set Σ_H of high level actions is $\{req_H, read_H\}$. The device B is always ready to accept an access request from the high level agent with probability $\frac{1}{2}$ and to grant a write access to the low level user with the same probability. Obviously, we always consider the device composed with a high level agent according to $||_L^p$ (we assume $p = \frac{1}{2}$ and $L = \{req_H, read_H\}$). On the one hand, when the device is composed with a high level agent that performs action req_H with probability 1, it synchronizes with the high agent accepting his request with probability $\frac{3}{4}$. On the other hand, if the high level agent does not perform req_H, the composed system performs action $grant_L$ with probability 1. As a consequence we can find out the following covert channels. Consider the high agent Π_1 of Figure 4, which executes a read request without performing the reading afterwards. System $(B||_L^p\Pi_1) \setminus \Sigma_H$ reaches a deadlock state that is not reached by B/Σ_H. In this way, the high level agent could transmit the bit 0 or 1 by alternatively blocking or not the device. Such a covert channel can be detected by the *PTNDC* property, in fact we have that $B/\Sigma_H \not\approx (B||_L^p\Pi_1) \setminus \Sigma_H$ so that $B \notin PTNDC$. Another interesting covert channel arises if one considers Π_2, which locks the buffer and executes a reading only after a time k. A low level user observing the behavior of $(B||_L^p\Pi_2) \setminus \Sigma_H$ does not receive any grant access for a time k when a req_H action is performed. In this way the high level agent could indirectly transmit value k to the low level user. We obviously have again that $B/\Sigma_H \not\approx (B||_L^p\Pi_2) \setminus \Sigma_H$.

The two covert channels introduced above could be avoided by introducing a timeout mechanism which releases the device if $read_H$ is not performed and by always releasing the device after a fixed amount of time has passed. In Fig-

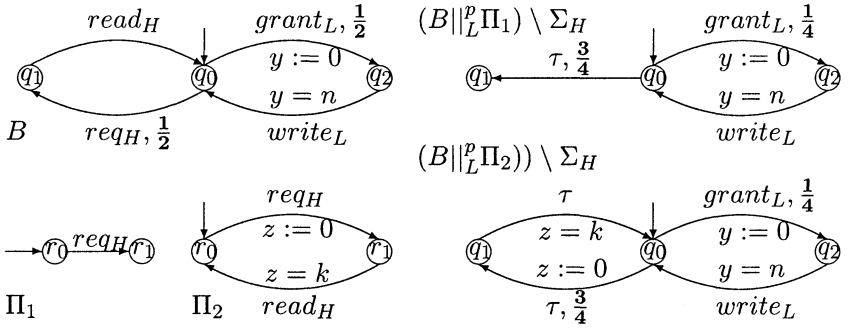

Figure 4. Device specification with timing covert channels.

ure 5 we show a device B' that accepts a high level request, and uses a clock x as timer and t as timeout. When it executes action req_H the timer is set to 0, action $read_H$ could be performed only when $x < t$, and when x reaches value t the device unlocks the buffer going back to q_0. When transitions starting from a given state have disjoint conditions we omit probabilities since their execution depends on the time configuration, rather than on the effective probability. The timing covert channels shown in the previous case could not be exploited anymore, however device B' is still insecure. In fact the device is unavailable for the fixed amount of time when a high level access is performed, and this is clearly observable by the low level user that has to wait the termination of the high level request before obtaining access to the buffer. This represents a typical situation where the unavailability of a shared resource can be encoded as 0 or 1 in order to transmit data. Such a situation is captured by the *PTNDC* property by considering again the automaton Π_2 and assuming $k < t$. In fact we have again that $B'/\Sigma_H \not\approx (B'||^p_L \Pi_2) \setminus \Sigma_H$.

The capacity of such a covert channel could be reduced, but not totally avoided, by considering a buffer that probabilistically locks himself without any high level request. In this manner the low level user could not be sure whether the buffer is really locked by the high user or not. In Figure 5, B'' represents a device that behaves in such a manner, locking himself with a probability $\mathbf{r}$. As we have said, this does not avoid entirely the covert channel, but the knowledge the low level user acquires is affected by some uncertainty. In fact, if the device is locked, the low level user could deduce that the high user locked the device with a certain probability while with probability $\mathbf{r}$ the device has locked himself for masking the higher user's activity.

We can completely hide the high level activity to the low level user by partitioning into two sessions the time in which users can access the buffer. During a *low session*, lasting a fixed amount of time n, only the low level user can access the buffer, then the device goes to the *high session*, where access is reserved,

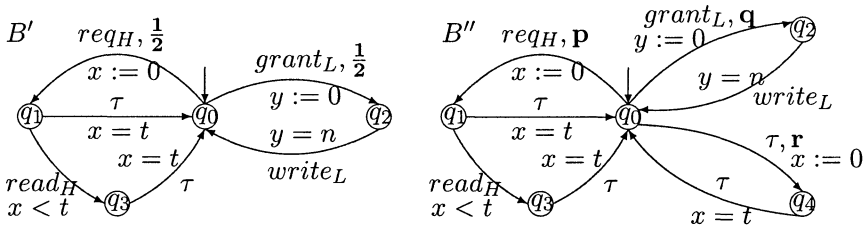

Figure 5. Improved device specifications.

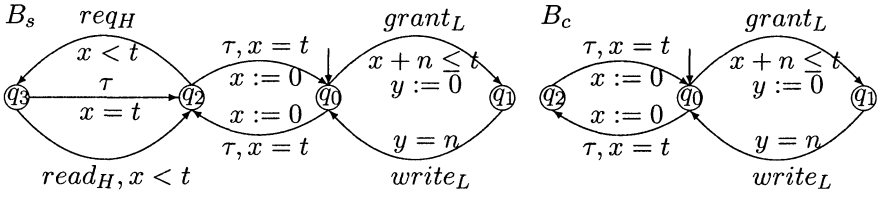

Figure 6. Secure Device.

for the same amount of time, to the high level user. This makes impossible for the low level user to discover something about the high level activity, since the same fixed amount of time is reserved to the high session even if the high user does nothing. In Figure 6 we specify a buffer B_s that behaves in such a manner: the buffer is reserved for a time t to the low level user and to the high level user alternatively. Automaton B_s is *PTNDC*, in fact, for every possible high level user Π, $(B_s||_L^p \Pi) \setminus \Sigma_H \approx B_c \approx B_s/\Sigma_H$. Intuitively, automaton B_c of Figure 6 represents the automaton resulting after the parallel composition between B_s and any high level user Π, and, therefore, B_c is weak bisimilar to B_s composed with any possible high level user Π. Finally, it easy to see that $B_c \approx B_s/\Sigma_H$.

5. Conclusions

The classical theory of non-interference must be extended to cope with real systems which may exhibit probabilistic and timing covert channels that are not captured by standard security models. In this paper we have developed a general framework where both probability and time are taken into account. By defining some information flow security property, we have shown how to detect with our model both probabilistic and timing covert channels.

We could easily give a definition of bisimulation requiring only that the difference of the probabilities in Definition 2 is less than a certain value and use it to give a measure of the security level of a system.

References

[1] A. Aldini, M. Bravetti, R. Gorrieri: *A Process-algebraic Approach for the Analysis of Probabilistic Non-interference*. Journal of Computer Security, to appear.

[2] R. Alur, C. Courcoubetis, D. L. Dill: *Verifying Automata Specifications of Probabilistic Real-Time Systems*. Real-Time: Theory in Practice, Springer LNCS 600, 28–44, 1992.

[3] R. Alur, D. L. Dill: *A Theory of Timed Automata*. Theoretical Computer Science 126:183–235, 1994.

[4] C. Baier, H. Hermanns: *Weak Bisimulation for Fully Probabilistic Processes*. Proc. of CAV'97, Springer LNCS 1254, 119–130, 1997.

[5] R. Barbuti, L. Tesei: *A Decidable Notion of Timed Non-interference*. Fundamenta Informaticae, 54(2–3):137–150, 2003.

[6] D. Beauquier: *On Probabilistic Timed Automata*. Theoretical Computer Science, 292:65–84, 2003.

[7] N. Evans, S. Schneider: *Analysing Time Dependent Security Properties in CSP Using PVS*. Proc. of Symp. on Research in Computer Security, Springer LNCS 1895, 222–237, 2000.

[8] R. Focardi, R. Gorrieri: *A Classification of Security Properties*. Journal of Computer Security, 3(1):5–33, 1995.

[9] R. Focardi, R. Gorrieri, F. Martinelli: *Information Flow Analysis in a Discrete-Time Process Algebra*. Proc. of 13th CSFW, IEEE CS Press, 170–184, 2000.

[10] J. A. Goguen, J. Meseguer: *Security Policy and Security Models*. Proc. of Symp. on Research in Security and Privacy, IEEE CS Press, 11–20, 1982.

[11] J. W. Gray III. *Toward a Mathematical Foundation for Information Flow Security*. Journal of Computer Security, 1:255–294, 1992.

[12] P. R. Halmos: *Measure Theory*. Springer-Verlag, 1950.

[13] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston: *Automatic Verification of Real-time Systems with Discrete Probability Distribution*. Theoretical Computer Science, 282:101–150, 2002.

[14] M. Kwiatkowska, R. Norman, J. Sproston: *Symbolic Model Checking of Probabilistic Timed Automata Using Backwards Reachability*. Tech. rep. CSR-03-10, University of Birmingham, 2003.

[15] R. Lanotte, A. Maggiolo-Schettini, A. Troina: *Weak Bisimulation for Probabilistic Timed Automata and Applications to Security*. Proc. of SEFM'03, IEEE CS Press, 34–43, 2003.

[16] D. McCullough: *Noninterference and the Composability of Security Properties*. Proc. of Symp. on Research in Security and Privacy, IEEE CS Press, 177–186, 1988.

[17] J. K. Millen: *Hookup Security for Synchronous Machines*. Proc. of 3rd CSFW, IEEE CS Press, 84–90, 1990.

[18] R. Milner: *Communication and Concurrency*. Prentice Hall, 1989.

[19] J. T. Wittbold, D. M. Johnson: *Information Flow in Nondeterministic Systems*. Proc. of Symp. on Research in Security and Privacy, IEEE CS Press, 144–161, 1990.