HAL
open science

The Dark Side of Timed Opacity

Franck Cassez

To cite this version:

Franck Cassez. The Dark Side of Timed Opacity. Proc. of the 3rd International Conference on Information Security and Assurance (ISA'09), Jun 2009, Seoul, Korea, South Korea. pp.21–30. inria-00493635

HAL Id: inria-00493635

https://inria.hal.science/inria-00493635v1

Submitted on 21 Jun 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The Dark Side of Timed Opacity

Franck Cassez*

National ICT Australia & CNRS
The University of New South Wales
Sydney, Australia
franck.cassez@cnrs.irccyn.fr
Franck.Cassez@nicta.com.au
http://www.irccyn.fr/franck

Abstract. In this paper we extend the notion of opacity, defined for discrete-event systems, to dense-time systems. We define the timed opacity problem for timed automata and study its algorithmic status. We show that for the very restrictive class of Event Recording Timed Automata, the opacity problem is already undecidable leaving no hope for an algorithmic solution to the opacity problem in dense-time.

1 Introduction

Security issues have become increasingly important in the last decade with the development of the Internet. Various notions and theories have emerged to model, design and check that a given system is secure. These theories provide sound foundations for the analysis of security policies. For instance, many security policies like access control, channel-control can be formulated within the framework of transitive or intransitive *non-interference* (see [1] for an excellent introduction to the subject).

Opacity. Non-interference cannot capture every type of security policies. Thus extension and generalization have been proposed to address relevant practical problems. *Opacity* was introduced in [2,3] to model leaks of information from a system to an attacker. The framework of opacity is the following: a (model of the) system A is given which generates sequences of actions over an alphabet Σ. A *secret* S is a subset of the sequences of actions generated by A. An *attacker* can only see the generated sequences through an interface which prevents the observation of some of the events: he has only a partial knowledge of the events generated by A and sees the projection of a sequence over an alphabet $\Sigma_o \subseteq \Sigma$. The secret S is opaque w.r.t. A and Σ_o, if for every observation w the attacker can make, he can never infer that this observation was produced by a secret sequence in S.

Assume the system A can generate the sequences of events $\{ab, cb\}$ and the attacker can only see $\Sigma_o = \{b\}$. Let $S = \{ab\}$ be the secret. Then S is opaque

* Author supported by a Marie Curie International Outgoing Fellowship within the 7th European Community Framework Programme.

w.r.t. A and Σ_o. Indeed the only observation the attacker can make is the sequence "b", but then, he cannot know whether this observation is the projection of ab or cb and thus cannot know the secret.

The opacity problem consists in checking whether a secret is opaque for a model of a system. Anonymity and non-interference can be reduced to opacity using a proper encoding [2]. If the system A is given by a finite transition system over an alphabet Σ, the secret S is a regular language included in Σ^* and the observation of an attacker is defined by a projection over $\Sigma_o \subseteq \Sigma$, the opacity problem can decided [2]. The previous introductory example can be modeled by the finite automaton $\mathcal{A}$ of Figure 1.

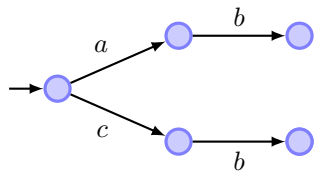

Fig. 1. The Automaton $\mathcal{A}$

Taking Time into Account. In the above mentioned framework, the attacker can only observe sequences of events but not the point in *time* at which they occur. But *time* could be an important information for an attacker. Indeed, assume the system generates $\{ab, cb\}$ as before, but when it generates ab, then "b" occurs at (global) time 1, and when it generates cb then "b" occurs at (global) time 2 (for instance, the system is slower when doing the "c" action than doing the "b"). Using this timing information, and yet observing only "b", an attacker can know whether an "a" or a "c" has occurred: if "b" occurs at time 1, the system generated ab, otherwise if "b" occurs at time 2, it generated cb. If the secret is ab as before, then it is no more opaque for an attacker who has a clock.

Taking into account the ability of the attacker to measure time gives a more accurate and realistic model of the system. Whether it is dense-time of discrete-time is important as it has been shown that the expressiveness of dense-time models (timed automata) is strictly larger than discrete-time ones [4]. In this paper we use timed automata [4] to model dense-time systems and study the opacity problem in this setting.

Related Work. Formalizing security policies is an important issue and has been investigated a lot in the recent years. In these frameworks, a finite or discrete state model of the system is known, and the security policy is specified as a property of this model: see [1,5] for non-interference and [2,3] for opacity.

Taking into account timing aspects in the model has been investigated for (strong non-deterministic) non-interference (SNNI) in [6,7] where it is shown that checking SNNI is decidable for deterministic timed automata. To our knowledge, the opacity problem for timed systems has not been studied and this is the contribution of this paper.

We show that, although it is important to take into account in the model of the system the capability of an attacker to measure time, the problem of checking whether such a model is opaque is undecidable for a very restricted class of timed systems (Event Recording Automata). This leaves no hope for an algorithmic solution to this problem.

Organisation of the Paper. Section 2 recalls the basics of timed languages and timed automata. Section 3 contains a formal definition of the opacity problem for dense-time systems. Section 4 is the main part of the paper and we prove that opacity is undecidable for timed automata. Finally, Section 5 gives a summary of the contribution of the paper.

2 Preliminaries

2.1 Notations

In the sequel Σ is a finite alphabet. We let τ be the *unobservable* action, and let $\Sigma_\tau = \Sigma \cup \{\tau\}$. $\mathbb{R}$ is the set of real numbers and $\mathbb{R}_{\geq 0}$ is the set of non-negative reals numbers. $\mathbb{N}$ the set of natural numbers, $\mathbb{Z}$ the set of integers and $\mathbb{B} = \{\text{TRUE}, \text{FALSE}\}$ is the set of boolean values.

Let X be a finite set of *clocks*. We let $\mathcal{C}(X)$ be the set of *convex constraints* on X, i.e., the set of conjunctions of constraints of the form $x \bowtie c$ with $c \in \mathbb{Z}$ and $\bowtie \in \{\leq, <, =, >, \geq\}$. A *clock valuation* is a mapping $v : X \to \mathbb{R}_{\geq 0}$. We let $\mathbb{R}_{\geq 0}^X$ be the set of clock valuations over X. We let $\mathbf{0}_X$ be the *zero* valuation where all the clocks in X are set to 0 (we use $\mathbf{0}$ when X is clear from the context). Given $\delta \in \mathbb{R}$, we let $v+\delta$ denote the valuation $(v+\delta)(x) = v(x)+\delta$. Given a constraint $g \in \mathcal{C}(X)$ and a valuation v, we write $v \models g$ if g is satisfied by the valuation v. Given a set $R \subseteq X$ and a valuation v of the clocks in X, $v[R]$ is the valuation $v[R](x) = v(x)$ if $x \notin R$ and $v[R](x) = 0$ otherwise.

The set of finite words over an alphabet Σ is Σ^* which contains the empty word ε. If $w = a_1 \cdots a_n$, $|w| = n$ is the *length* of w, and we write $w[i], 1 \leq i \leq |w|$ to denote the ith letter a_i of w. A *language* L is any subset of Σ^*. A finite *timed word* over Σ is a word in $(\Sigma \times \mathbb{R}_{\geq 0})^*$ i.e., over the (infinite) alphabet $\Sigma \times \mathbb{R}_{\geq 0}$. Thus a timed word is a pair $w = (\sigma, t)$ with $\sigma \in \Sigma^*$, $t \in \mathbb{R}_{\geq 0}^*$ with $|\sigma| = |t|$ and with the convention that event $\sigma[i]$ occurs at global time $t[i]$. Hence we require the $t[i]$'s to form an *increasing sequence*.

$TW^*(\Sigma)$ is the set of finite timed words over Σ and again ε is the empty timed word. A *timed language* is any subset of $TW^*(\Sigma)$. Given a timed language K, we let $Unt(K) = \{\sigma \in \Sigma^* \mid \exists t \in (\mathbb{R}_{\geq 0})^* : (\sigma, t) \in K\}$. Given $\Sigma' \subseteq \Sigma$, the *projection* of $(\sigma_1, t_1)(\sigma_2, t_2) \cdots (\sigma_n, t_n)$ is the timed word that comprises of the letters $\sigma_i \in \Sigma'$. For example $\boldsymbol{\pi}_{\{a,b\}}((a,1)(c,2.34)(\tau,2.986)(b,3.146)(c,4.16))$ is $(a,1)(b,3.146)$. $\boldsymbol{\pi}_{\Sigma'}(K) = \{\boldsymbol{\pi}_{\Sigma'}(w) \mid w \in K\}$. Let $K \subseteq (\Sigma')^*$. The *inverse projection* $\boldsymbol{\pi}_{\Sigma}^{-1}(K)$ is defined by: $\boldsymbol{\pi}_{\Sigma}^{-1}(K) = \{w \in TW^*(\Sigma) \mid \boldsymbol{\pi}_{\Sigma'}(w) \in K\}$.

2.2 Timed Automata

Timed Automata (TA) were introduced in [8] to model real-time systems using dense-time. The fundamental results about timed automata can be found in [4].

Definition 1 (Timed Automaton). *A* Timed Automaton A *is a tuple* $(L, l_0, X, \Sigma_\tau, E, F)$ *where:*

– L *is a finite set of* locations*;*
– l_0 *is the* initial location*;*
– X *is a finite set of* clocks*;*
– Σ *is a finite set of* actions*;*
– $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\tau \times 2^X \times L$ *is a finite set of* edges*; in an edge* (ℓ, g, a, r, ℓ')*,* g *is the* guard*, and* r *the* reset *set;*
– $F \subseteq L$ *is the set of* final *locations.* ■

A *state* of A is a pair $(\ell, v) \in L \times \mathbb{R}^X_{\geq 0}$. A *run* ϱ of A from (ℓ_0, v_0) is a sequence of the form:

$$\varrho = (\ell_0, v_0) \xrightarrow{\delta_0} (\ell_0, v_0 + \delta_0) \xrightarrow{a_0} (\ell_1, v_1) \cdots \quad \cdots \xrightarrow{a_{n-1}} (\ell_n, v_n) \xrightarrow{\delta_n} (\ell_n, v_n + \delta_n)$$

s.t. for every $i \geq 0$ there is some edge $(\ell_i, g_i, a_i, r_i, \ell_{i+1}) \in E$ and: (i) $v_i + \delta_i \models g_i$, (ii) $v_{i+1} = (v_i + \delta_i)[r_i]$. The set of finite runs from $s = (\ell, v)$ is denoted $\mathit{Runs}(s, A)$ and we let $\mathit{Runs}(A) = \mathit{Runs}((l_0, \mathbf{0}), A)$. We let $\mathit{last}(\varrho) = (\ell_n, v_n + \delta_n)$. The *trace*, $\mathit{tr}(\varrho)$, of the finite run ϱ is the timed word $(\sigma_1, t_1)(\sigma_2, t_2) \cdots (\sigma_n, t_n)$ with $\sigma_i = a_{i-1}, 1 \leq i \leq n$ and $t_i = \sum_{k=0}^{i-1} \delta_k, 1 \leq i \leq n$. For $V \subseteq \mathit{Runs}(A)$, we let $\mathit{Tr}(V) = \{\mathit{tr}(\varrho) \mid \varrho \in V\}$, which is the set of traces of the runs in V. Let $\mathit{Tr}(A) = \mathit{Tr}(\mathit{Runs}(A))$ be the set of traces *generated* by A (note that this language is prefix-closed).

A finite timed word w is *accepted* by A if it is the trace of a run ϱ of A that ends in an F-location i.e., $\mathit{last}(\varrho) \in F \times \mathbb{R}^X_{\geq 0}$. $\mathcal{L}(A)$ is the set of traces of finite timed words accepted by A.

Example 1.
The automaton $\mathcal{B}$ of Figure 2 is a simple timed automaton with one clock x. It indicates that an "a" or a "c" can occur at time 0 and be followed either by a "b" at time 1 or a "b" at time 2. In this example we do not use strict constraints like $1 < x < 2$. Thus automaton $\mathcal{B}$ can generate the following runs:

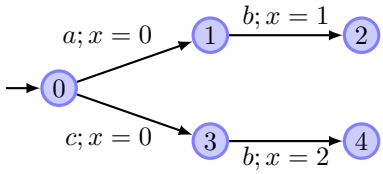

Fig. 2. The Automaton $\mathcal{B}$

$$(0, x = 0) \xrightarrow{a} (1, x = 0) \xrightarrow{1} (1, x = 1) \xrightarrow{b} (2, x = 1) \xrightarrow{\delta} (2, x = 1 + \delta)$$

or

$$(0, x = 0) \xrightarrow{c} (3, x = 0) \xrightarrow{2} (3, x = 2) \xrightarrow{b} (4, x = 2) \xrightarrow{\delta} (4, x = 2 + \delta)$$

with $\delta \geq 0$. The set of timed words generated by automaton $\mathcal{B}$ consists of two timed words and $\mathcal{L}(\mathcal{B}) = \mathit{Tr}(\mathcal{B}) = \{(a, 0)(b, 1), (c, 0)(b, 2)\}$. ◇

A τ-edge in a timed automaton is an edge (ℓ, g, a, r, ℓ') with $a = \tau$. A timed automaton $(L, l_0, X, \Sigma_\tau, E, F)$ is *deterministic* if: (i) it does not contain any τ-edge and (ii) whenever two edges $(\ell, g_1, a, r_1, \ell_1)$ and $(\ell, g_1, a, r_2, \ell_2)$ are in E, then $g_1 \wedge g_2$ is equivalent to FALSE. A is an *Event Recording Automaton* if: it is

deterministic[1] and (iii) each clock $x_a \in X$ is paired with an event $a \in \Sigma$, and (iv) if $(\ell, g, a, r, \ell') \in E$, then $r = \{x_a\}$.

We use the following classes of TA in the sequel:

- the most general class [9] of TA with τ-edges given by Definition 1 is denoted τNTA;
- NTA is the sub-class of τNTA which consists of non-deterministic TA with no τ-edges;
- DTA is the sub-class of NTA which consists of deterministic TA;
- ERA is the sub-class of DTA which consists of Event Recording Automata (ERA, see [10]); ERA are TA where each clock is associated with an event and when this event occurs the corresponding clock is reset.

These classes of TA can be ordered according to the class of timed languages they accept. A class C is more expressive than C' if every timed language accepted by a TA of C' can be accepted by a TA of C. This defines a pre-order $C' \sqsubseteq C$ (reads "C is more expressive than C'") on classes of TA. The expressive power of the different classes is strictly increasing in the following order (see [10,9]):

$$\text{ERA} \sqsubset \text{DTA} \sqsubset \text{NTA} \sqsubset \tau\text{NTA} \tag{1}$$

One of the key result in the seminal paper of Alur and Dill [4] is that the *universality problem* for NTA, i.e., checking whether a NTA A accepts all timed words, is undecidable (notice that it is decidable for DTA [4]).

2.3 Product of Timed Automata

In the sequel we need to use the *product* of two TA to reduce the opacity problem to the L-opacity problem (section 4.2).

Definition 2 (Product of TA). *Let $A_i = (L_i, l_0^i, X_i, \Sigma_\tau^i, E_i, F_i)$, $i \in \{1, 2\}$, be two TA such that $X_1 \cap X_2 = \emptyset$. The* product *of A_1 and A_2 is the timed automaton $A_1 \times A_2 = (L, l_0, X, \Sigma_\tau, E, F)$ given by:*

- $L = L_1 \times L_2$,
- $l_0 = (l_0^1, l_0^2)$,
- $\Sigma = \Sigma^1 \cup \Sigma^2$,
- $X = X_1 \cup X_2$,
- $E \subseteq L \times \mathcal{C}(X) \times \Sigma \times 2^X \times L$ *and* $((\ell_1, \ell_2), g_{1,2}, \sigma, r, (\ell'_1, \ell'_2)) \in E$ *if:*
  - *either* $\sigma \in \Sigma_1 \cap \Sigma_2$, *and (i)* $(\ell_k, g_k, \sigma, r_k, \ell'_k) \in E_k$ *for* $k = 1$ *and* $k = 2$; *(ii)* $g_{1,2} = g_1 \wedge g_2$ *and (iii)* $r = r_1 \cup r_2$;
  - *or for* $k = 1$ *or* $k = 2$, $\sigma \in (\Sigma_k \setminus \Sigma_{3-k}) \cup \{\tau\}$, *and (i)* $(\ell_k, g_k, \sigma, r_k, \ell'_k) \in E_k$; *(ii)* $g_{1,2} = g_k$ *et (iii)* $r = r_k$;
- $F = F_1 \times F_2$. ■

[1] In their original paper [10], Alur *et al.* do not require that ERA be deterministic. Considering deterministic ERA is not a restriction since (non-deterministic ERA) are determinizable [10].

A_1 and A_2 have no clocks in common, ans it is a well-known fact that:

Lemma 1. $\mathcal{L}(A_1 \times A_2) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

Thus given two automata A_1 and A_2 that have no clocks in common, it is possible to construct the product $A_1 \times A_2$ which accepts exactly the intersection of the two timed languages $\mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

3 Timed Opacity

In this section we give a formal definition of opacity for timed automata along with a small example.

Assume we are given a timed automaton $(L,\ l_0, X,\ \Sigma_\tau,\ E, F)$ and a *secret* $S \subseteq TW^*(\Sigma)$. Let $\Sigma_o \subseteq \Sigma$ be a set of *observable* events. Opacity is a property which ensures that, if the system generates a timed word w, and an external observer, the *attacker*, can only see $\boldsymbol{\pi}_{\Sigma_o}(w)$, he can never infer whether the original timed word w generated by A is in S or not. Let $\boldsymbol{\pi}$ be the projection over Σ_o.

Two words $w, w' \in TW^*(\Sigma)$ are $\boldsymbol{\pi}$-*equivalent*, denoted $w \equiv w'$ if $\boldsymbol{\pi}(w) = \boldsymbol{\pi}(w')$. Given $w \in \boldsymbol{\pi}(Tr(A))$, we let $[w]$ be the set of timed words of A the projection of which is w. Thus $[w] = \boldsymbol{\pi}^{-1}(w) \cap Tr(A)$ i.e., $[w]$ is the set of timed words generated by A which give the same trace when projected on Σ_o. We can define formally what is opacity for timed automata. The relational operator $\not\subseteq$ is used to denote non inclusion: for two sets A and B, $A \not\subseteq B$ stands for "A is not included in B".

Definition 3 (Opacity). *The secret S is* opaque *w.r.t. A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \boldsymbol{\pi}(Tr(A))$, $[w] \not\subseteq S$.* ■

The *opacity problem* is the following:

Given a TA A, a secret S and $\Sigma_o \subseteq \Sigma$, is S opaque w.r.t. A and Σ_o ?

Definition 3 defines opacity as a non-inclusion property. If $[w] \not\subseteq S$ for $w \in \boldsymbol{\pi}(Tr(A))$, it simply means that the observation w can be produced by at least two timed words w_1 and w_2 in A and at least one of them is not in S. Thus we cannot conclude by observing w that a word of S generated the observation w.

Example 2. Assume $\Sigma = \{a, b, c\}$ and $\Sigma_o = \{b\}$. Let $S = \{ab\}$ be the secret and $\mathcal{A}$ be the finite automaton[2] given in Figure 1 (all locations are accepting) in the introduction. S is opaque w.r.t. the automaton $\mathcal{A}$ and Σ_o because, if an attacker sees a "b", he cannot infer what the preceding letter was and cannot know the secret ab: $\boldsymbol{\pi}^{-1}(b) \cap Tr(\mathcal{A}) = \{ab, cb\} \not\subseteq S$.

The timed automaton in Figure 2, page 4, generates the timed language $\mathcal{L}(\mathcal{B}) = \{(a, 0)(b, 1), (c, 0)(b, 2)\}$. Notice also that $Unt(\mathcal{L}(\mathcal{B})) = \mathcal{L}(\mathcal{A})$. Define the secret by the timed language $S' = \{(a, t_1)(b, t_2)\}, 0 \leq t_1 \leq t_2$. Assume now

[2] A finite automaton is a timed automaton with no clocks.

that an attacker can measure time with his own clock but can only observe $\Sigma_o = \{b\}$. Then S' is not opaque any more w.r.t. $\mathcal{B}$ and Σ_o: indeed, if the attacker sees "b" at time 1 he knows an "a" has occurred. Thus there is a timed word $w = (a,0)(b,1) \in Tr(\mathcal{B})$ s.t. $\boldsymbol{\pi}_{\Sigma_o}(w) = (b,1) \in \boldsymbol{\pi}_{\Sigma_o}(Tr(\mathcal{B}))$ and $\boldsymbol{\pi}^{-1}_{\Sigma_o}((b,1)) = \{w\} \subseteq S'$ which contradicts the requirement of Definition 3. $\diamondsuit$

In this sense timed opacity is an interesting generalisation of opacity as it takes into account the time-measuring capabilities of an external attacker which gives him additional observational power. Nevertheless, from an algorithmic viewpoint, timed opacity is hopeless as we will show in the next section.

4 Checking Timed Opacity

In this section we will prove that the simplest versions of the timed opacity problem are undecidable for timed automata.

We first define a version of timed opacity where the secret is a set of states an automaton can be in. Let $A = (L, l_0, X, \Sigma_\tau, E, F)$ be a timed automaton where F is a set of *secret* locations. Let $\Sigma_o \subseteq \Sigma$ and $\boldsymbol{\pi}$ be the associated projection. Given a word $w \in TW^*(\Sigma)$, we let $last(w) = \{last(\varrho) \mid \varrho \in Runs(A) \wedge tr(\varrho) = w\}$; $last(w)$ is the set of states A can be in after reading the timed word w. This extends trivially to sets of traces.

4.1 Checking Location-Based Opacity

We can now define a *location-based* version of timed opacity: Let $A = (L, l_0, X, \Sigma_\tau, E, F)$ be a timed automaton with secret locations F.

Definition 4 (L-Opacity). *F is* L-opaque *with respect to A and $\Sigma_o \subseteq \Sigma$ iff for each $w \in \boldsymbol{\pi}(Tr(A))$, $last([w]) \not\subseteq F$.* ■

The *L-opacity problem* asks the following:

> Given a TA A, a set of secret states F and $\Sigma_o \subseteq \Sigma$, is F L-opaque w.r.t. A and Σ_o?

A first negative result is easy to prove for L-opacity:

Theorem 1. *The L-opacity problem is undecidable for* NTA.

Proof. We reduce the *universality problem* for NTA, which is known to be undecidable [4], to the L-opacity problem for NTA.

Let $A = (L, l_0, X, \Sigma, E, F)$ be a non-deterministic TA, with accepting locations F. The universality problem consists in deciding whether $\mathcal{L}(A) = TW^*(\Sigma)$.

First thing we do is to complete A by adding edges from each location, with guard TRUE, and fresh target location *Bad* which is not in F. Then we can assume that $Tr(A) = TW^*(\Sigma)$. Note that it does not change $\mathcal{L}(A)$ as *Bad* is not accepting. $\mathcal{L}(A)$ is universal is equivalent to:

$$\forall w \in TW^*(\Sigma), last(w) \not\subseteq L \setminus F \tag{2}$$

Let $\Sigma_o = \Sigma$. $L \setminus F$ is L-opaque w.r.t. A and Σ amounts to:

$$\forall w \in \pi(Tr(A)), last([w]) \not\subseteq L \setminus F \tag{3}$$

but as $\pi(Tr(A)) = Tr(A) = TW^*(\Sigma)$ and $[w] = w$ (no τ-edges), equation (2) is equivalent to equation (3). As universality is undecidable for NTA, the result follows. □

Because τNTA includes NTA, it follows that the L-opacity problem is also undecidable for τNTA. To prove the previous result we did not need to take any particular strict subset of Σ. It turns out that, using this possibility to make the observable alphabet vary, we can prove that the L-opacity problem is also undecidable for DTA.

Theorem 2. *The L-opacity problem is undecidable for* DTA.

Proof. Let $A = (L,\, l_0, X,\, \Sigma_\tau,\, E, F)$ be a τNTA. We show that F is L-opaque w.r.t. A and Σ if and only if F is L-opaque w.r.t. A' and Σ, where A' is a DTA. Assume the only non-determinism in A is on the τ-edges, i.e., A is deterministic for all the other actions in Σ. Then A has a finite number, say n, of τ-edges. Let $\{a_1, a_2, \cdots, a_n\}$ be fresh letters not in Σ. Order the τ-edges and replace the τ action in τ-edge k by a_k. This gives a DTA A' on the alphabet $\Sigma \cup \{a_1, \cdots, a_n\}$. It is easy to see that F is L-opaque w.r.t. A and Σ if and only if F is L-opaque w.r.t. A' and Σ. □

We now restrict our attention to ERA. It turns out that the L-opacity problem is also undecidable for ERA:

Theorem 3. *The L-opacity problem is undecidable for* ERA.

Proof. We reduce the L-opacity problem for DTA to an opacity problem on ERA. Let $A = (L,\, l_0, X,\, \Sigma, E, F)$ be a DTA. We let $\Sigma(X) = \{r_x, x \in X\}$ be a set of actions corresponding to each clock x in X. Also we define $X_a = \{x_a, a \in \Sigma\}$ to be a new set of clocks associated with actions in Σ.
Let (ℓ, g, a, R, ℓ') be an edge from E with $R = \{x_1,\, \cdots,\, x_k\}$. Consider the sequence of edges given in Figure 3. In this sequence, we reset the first clock (x_a does not appear in any guard of A) x_a and then use it to enforce the reset of the clocks in R within 0 time units. The set R_i is given by $\{x_i, \cdots, x_k\}$. The sequence of transition given on Figure 3 fires only if g is satisfied (and the new clocks in X_a are not constraining g) and then resets the clocks in R without any time elapsing. Thus when ℓ' is reached, the values of the clocks in X have the same values as the ones they would have if firing (ℓ, g, a, R, ℓ').

For an edge $e \in E$, denote $\kappa(e)$ the associated set of edges as given in Figure 3. Let $A' = (L \cup L', l_0, X \cup X_a, \Sigma \cup \Sigma(X), E', F)$ be the TA with $L' = L \times 2^X$ and E' comprises of all the edges $\kappa(e), e \in E$.

First A' is an ERA. Second F is opaque w.r.t. A and Σ if and only if F is opaque w.r.t. A' and Σ. This completes the proof. □

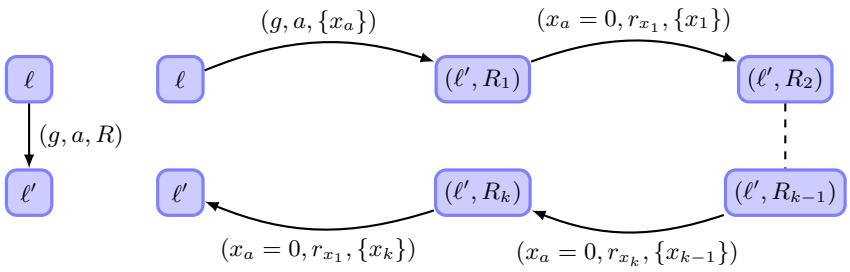

Fig. 3. Widget $\kappa(e)$ for Encoding DTA L-Opacity into ERA L-Opacity

4.2 Checking Opacity

The initial version of opacity of Definition 3 can be reduced to L-opacity for secret languages S generated by deterministic timed automata, even ERA. This opacity problem has two parameters: the model of the system and the secret language. It follows that:

Theorem 4. *The opacity problem is undecidable for systems given by* ERA *and secrets given by* ERA*.*

Proof. We reduce opacity to L-opacity. Let $A = (L, l_0, X, \Sigma, E, L)$ be an ERA and $\Sigma_o \subseteq \Sigma$. Assume the secret language $S \subseteq \mathcal{L}(A)$ is given by an ERA $A_S = (L_S, l_0^S, X_S, \Sigma, E_S, F_S)$. Define the product $A \times A_S$ and remind that the final locations are $F \times F_S$. Notice that the product of two ERAs is an ERA. By definition of $A \times A_S$, and Lemma 1, $\mathcal{L}(A \times A_S) = \mathcal{L}(A) \cap \mathcal{L}(A_S) = \mathcal{L}(A) \cap S$.

Consequently $w \in \mathcal{L}(A \times A_S)$ if and only if $w \in \mathit{Tr}(A) \cap S$. Thus $\mathit{last}([w]) \not\subseteq F \times F_S$ if and only if $[w] \not\subseteq S$. This completes the proof. □

Remark 1. In the untimed case, for finite transition system, non-interference can be reduced to an opacity problem [2]. It should not be difficult to extend the reduction to timed automata. Thus it is not so surprising that opacity is undecidable for NTA because checking non-interference for NTA is undecidable [6].

What is surprising is that this result holds for very restrictive classes of timed automata like ERA, which usually have very nice closure and decidability properties [10]. ♢

5 Conclusion

In this paper we have addressed the opacity problem for timed systems. It turns out that the opacity problem is undecidable for the very restrictive class of ERA. Notice that our result carries over to other reasonable models of dense-time systems like Time Petri Nets (TPN), because TPN and τNTA are equivalent w.r.t. timed language acceptance [11].

Our result is based on the undecidability of universality for NTA operating in dense-time. Considering a time domain like $\mathbb{N}$ (not dense) may render the opacity problem tractable.

Acknowledgements. The author would like to thank J. Dubreil and H. Marchand for introducing the opacity problem to him.

References

1. Rushby, J.: Noninterference, Transitivity and Channel-Control Security Policies. Technical report, SRI International (2005)
2. Mazaré, L.: Using unification for opacity properties. In: Proceedings of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'04), Barcelona (Spain) (2004) 165–176
3. Bryans, J., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to transition systems. In Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S.A., eds.: Formal Aspects in Security and Trust. Volume 3866 of Lecture Notes in Computer Science., Springer (2005) 81–95
4. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science (TCS) **126**(2) (1994) 183–235
5. Focardi, R., Gorrieri, R.: Classification of security properties (part I: Information flow). In Focardi, R., Gorrieri, R., eds.: Foundations of Security Analysis and Design I: FOSAD 2000 Tutorial Lectures. Volume 2171 of Lecture Notes in Computer Science., Heidelberg, Springer-Verlag (2001) 331–396
6. Gardey, G., Mullins, J., Roux, O.H.: Non-interference control synthesis for security timed automata. In: 3rd International Workshop on Security Issues in Concurrency (SecCo'05). Electronic Notes in Theoretical Computer Science, San Francisco, USA, Elsevier (2005)
7. Benattar, G., Cassez, F., Lime, D., Roux, O. H.: Synthesis of Non-Interferent Timed Systems. Submitted (2009)
8. Alur, R., Dill, D.: Automata for modeling real-time systems. In: Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90). Volume 443 of Lecture Notes in Computer Science., Springer (1990) 322–335
9. Bérard, B., Diekert, V., Gastin, P., Petit, A.: Characterization of the expressive power of silent transitions in timed automata. Fundamenta Informaticae **36**(2–3) (1998) 145–182
10. Alur, R., Fix, L., Henzinger, T.A.: Event clock automata: A determinizable class of timed automata. In: Proc. 6th International Conference on Computer Aided Verification (CAV'94). Volume 818 of Lecture Notes in Computer Science., Springer (1994) 1–13
11. Cassez, F., Roux, O.H.: Structural translation from time petri nets to timed automata. Journal of Software and Systems **79**(10) (2006) 1456–1468