Journal of Information Security and Applications 61 (2021) 102926

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Bounded opacity for timed systems

Ikhlass Ammar [a,b,*], Yamen El Touati [b,c,d], Moez Yeddes [b,e], John Mullins [f]

[a] Faculty of Sciences of Tunis (FST), University of Tunis El Manar, Tunisia
[b] OASIS Laboratory, National Engineering School of Tunis, University of Tunis El Manar, Tunisia
[c] Faculty of Computing and IT, Northern Border University, Kingdom of Saudi Arabia
[d] ISAMM, University of Manouba, Tunisia
[e] National Institute of Applied Sciences and Technology (INSAT), University of Carthage, Tunisia
[f] École Polytechnique de Montréal. Montréal (Québec), Canada

ARTICLE INFO

Keywords:
Real-time systems
Timed automata
Timed opacity
Timed bounded languages

ABSTRACT

In many security applications, system behaviors must be kept secret (opaque) to outside observers (intruders). Opacity was first studied for discrete event systems, and then it was extended to real-time systems. One of the challenges of real-time systems is the difficulty to guarantee their opacity against a potential attacker. In general, this property is undecidable for systems modeled by timed automata. A secret location, S, of a system is timed opaque to an intruder having partial observability of the system, if the intruder can never infer from the observation of any execution that the system has reached any secret location. In the present study, the static partial observability for systems modeled by nondeterministic timed automata is investigated. Thus, it focuses on systems where the timing of secret state reachability is bounded. The first contribution of this study is to define the bounded timed opacity property wherein, its complexity is proved. The second contribution is to consider systems where the secret should be kept hidden for a certain period referred to as the Δ-duration bounded opacity property. Also, a formal definition is proposed and its complexity is proved. In addition, the proposed properties are verified using timed bounded model checking. A case study "Exchange in the Cloud system" is modeled by timed automaton to verify the proposed properties using SpaceEx tool.

1. Introduction

The continuing development of computer systems and networks has included the advancement of cybersecurity. Various theories and notions have emerged to model, design and check that a given system is secure. The security problem is categorized into classes of security properties, namely, integrity properties, privacy properties and availability properties [1].

Opacity is a security property that is categorized under the privacy properties class, where information leaks from a system to an attacker are modeled and verified [2]. The opacity problem has recently been used in the context of Discrete-Event Systems (DES), where the system is modeled as a finite-state automaton and the intruder is modeled as an observer that knows the system's behavior, [3–6]. The parameters of opacity are a predicate that is given as either a subset of states or events of the system modeled by its Labeled Transition System (LTS), and an observation function that models the abilities of an unwanted party and returns the subset of observable events (the user can observe these events). The predicate is considered as a subset of secret locations or a set of sequences of events (i.e., traces). The system is considered opaque if no intruders are able to infer any information about the secret predicate. If the predicate is a subset of locations, the observer cannot confirm whether the system is in a secret location. In the literature, different variants of opacity exist using a subset of locations as a predicate, such as the Initial-State Opacity [7], the Current-State Opacity [3,8], Initial-and-Final State Opacity [9]. If the predicate is a set of events sequences, the language based opacity is verified when the observer cannot know if the timed traces are in the secret language. The observability of the system's behavior exists in four categories: static, dynamic, Orwellian and m-Orwellian [10].

Moreover, the opacity problem is studied in the context of real-time systems. In these systems and in critical systems, the timing of actions is crucial, as well as the timing of secret disclosure. For these systems, the dense-time has to be taken into consideration, which can easily lead to undecidable results [11]. The proof of undecidability is based on the undecidability of the language inclusion problem (universality problem) for nondeterministic timed automata. In this context, it is

* Corresponding author at: Faculty of Sciences of Tunis (FST), University of Tunis El Manar, Tunisia.
E-mail addresses: ikhlass_ammar@yahoo.fr (I. Ammar), yamen.eltouati@yahoo.fr (Y. El Touati), yeddes@gmail.com (M. Yeddes), john.mullins@polymtl.ca (J. Mullins).

https://doi.org/10.1016/j.jisa.2021.102926

Available online 14 July 2021

interesting to identify restrictions on timed automata that do not reduce considerably the expressive power of the real-time systems to maintain the decidability of the problem. Real-Time Automata (RTA) [12] is a subclass of decidable timed automata. In addition, the authors in [13, 14] define a new notion of timed opacity, namely, timed opacity w.r.t, the execution time D for Parametric Timed Automata (PTA).

In the last decade, a lot of work on various aspects of bounded time domains has been widely studied. The authors in [15,16] have proved the decidability of the bounded verification of inclusion problem for timed automata with a given upper bound using timed-bounded model checking. This work used Timed-Bounded Model Checking for MTL logic, LTL+past, and LTL. Timed bounded Model Checking focuses on design verification: given a model M and a formula ϕ, checking the existence of a counterexample (a behavior of M which violates ϕ) over α units of time.

According to the inclusion problem of bounded timed automata, we introduce a similar notion to general opacity, so called timed bounded opacity for nondeterministic timed automata. We consider that the secret S is bounded opaque before a given duration α, if the attacker can never disclose the secret S before the upper bound α. In addition, we define a similar notion for a K-step opacity for real-time systems bounded by a given duration, referred to as Δ-duration bounded opacity. This opacity property has been studied to verify whether the attacker could reveal the secret before a given duration Δ. These notions are introduced for a static projection. We prove that these new notions of opacity are decidable for timed automata. Then, the timed bounded model checking is used to verify the timed bounded opacity and Δ-duration bounded opacity. Many real time systems present some features that need to be kept hidden during a given duration such as the English Online Auction Systems [17], Patent System, Exchange in the Cloud, etc. In this context, we are interested particularly to model the Exchange in the Cloud system using timed automaton and verify the proposed timed opacity properties.

The remainder of the paper is organized as follows. Section 2 reviews and introduces the basic notation and definitions related to timed languages and automata. Section 3 presents the definition of bounded timed opacity and its decidability. Section 4 presents the definition of Δ-duration bounded opacity and its decidability. A verification of the proposed properties using timed bounded model checking is detailed in Section 5. Section 6 outlines our design of a case study and the verification of timed opacity properties using SpaceEx. Section 7 surveys the related work of opacity associated with discrete event systems and real-time systems. Finally, Section 8 highlights the conclusion.

2. Background and general notation

$\mathbb{N}, \mathbb{R}, \mathbb{R}_+, \mathbb{R}_+^*$ represent respectively the set of natural numbers, real numbers, non negative real numbers and positive real numbers. B^* is the set of finite sequences of elements in B. B^∞ is the set of infinite sequences of elements in B.

$\overline{a}$ represents the vector where all elements are equal to $a \in \mathbb{R}$. $]a,b]$, $[a,b]$, $[a,b[$, $]a,b[$ represent respectively the left-open, closed, right-open intervals such that $]a,b] = \{x, x > a \text{ and } x \leq b\}$, $[a,b] = \{x, x \geq a, x \leq b\}$, $[a,b[= \{x, x \geq a \text{ and } x < b\}$, $]a,b[= \{x, x < a \text{ and } x < b\}$ where $a < b$. $floor(a) \in \mathbb{N}$ is a function that maps $a \in \mathbb{R}_+$ to the greatest integer less than or equal to a. The supremum of set $B \subset \mathbb{R}$ is denoted by $Sup(B) = a$ where $\forall \epsilon > 0$, $\exists b \in B$ with $a > b - \epsilon$ and $b \leq a$, $\forall b \in B$.

The set of conjunctions of constraints on X, denoted by $C(X)$ is a set of convex constraints on X in the form $x \sim c$ with $x \in X$, $\sim \in \{\leq, =, >, <, \geq\}$ and $c \in \mathbb{R}_+$. A clock valuation is a mapping $v : X \longrightarrow \mathbb{R}_+$. Given $d \in \mathbb{R}_+$, we let $v + d$ denote the valuation $(v + d)(x) = v(x) + d$. Given $X' \subseteq X$, the reset function $v[X' \longrightarrow 0]$ is defined by $v[X' \longrightarrow 0] = 0$ if $x \in X'$, otherwise $v[X' \longrightarrow 0] = v(x)$.

Definition 1 (*Timed Automaton [18]*)**.** A timed automaton A is a tuple $A = (L, l_0, X, \Sigma, I, T)$ 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, $I \in C(X)^L$ is a function that associates an invariant with each location and T is a finite set of transitions $T \subseteq L \times C(X) \times \Sigma \times 2^X \times L$. Given $e \in T$ an edge ; $e = (l, g, a, r, l')$, where g is the guard, a is an action and r is the reset set.

Definition 2 (*Timed Automata with Final Locations*)**.** Let $A_f = (L, l_0, X, \Sigma, I, F, T)$ be a timed automaton with final locations where $F \subseteq L$ is the set of final locations.

Definition 3 (*Timed Transition Systems (TTS)*)**.** [19] Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1). The timed transition system $G_A = (Q, q_0, \Sigma, \rightarrow)$ associated with A where: Q is the set of states $Q \subseteq L \times \mathbb{R}_+^X$, q_0 is the initial state $q_0 = (l_0, \bar{0})$, $\rightarrow \subseteq (Q \times (\Sigma \cup \mathbb{R}_+) \times Q)$ is the transition relation.

There are two kinds of transition relations ($\rightarrow$) in a timed automaton. **Delay transition relation**, [20]. It lets time elapse so that the value of all clocks increases accordingly. It is defined as follows: for (l, v) and a real-valued time increment $t \geq 0$, $(l, v) \xrightarrow{t} (l, v + t)$, if for all $0 \leq t' \leq t$, $v + t'$ satisfies the invariant $I(l)$. **Discrete transition relation**, [20]. It can be defined as a single symbolic transition $e = (l, g, a, r, l') \in T$ where for (l, v) and v satisfies the guard g, $(l, v) \xrightarrow{a} (l', v')$ and $v' = v[r \rightarrow 0]$.

A state of A is a pair $q = (l, v) \in L \times \mathbb{R}_+^X$. A *run* ρ is a possibly infinite execution, of the timed automaton that can be represented as follows: $\rho = q_0 \xrightarrow[d_0]{a_0} q_1 \cdots \xrightarrow[d_p]{a_p} q_{p+1} \cdots$ [18] where d_0 is interpreted by the duration between the execution time of a_0 and zero, $0 < p$, d_p is interpreted by the duration between the execution time of a_p and the execution time of a_{p-1}. For more simplicity, we use the following representation of run: $\rho = q_0 \xrightarrow{(a_0,d_0)} q_1 \cdots \xrightarrow{(a_p,d_p)} q_{p+1} \cdots$. For a finite run $\rho_f = q_0 \xrightarrow{(a_0,d_0)} \cdots \xrightarrow{(a_p,d_p)} q_{p+1}$, we note $Last(\rho_f) = l_{p+1}$. The set of runs over A is denoted by $Runs(A)$. The *trace* is presented by $tr(\rho) = (a_0, \gamma_0) \cdots (a_p, \gamma_p) \cdots$ where $\gamma_i = \sum_{j=0}^{i} d_j$, $\forall i \geq 0$. A *timed word* u is accepted by A, if $\exists \rho \in Runs(A)$ such that $tr(\rho) = u$. We say that the timed word u is carried by accepting the run ρ. The empty timed word is denoted by ϵ. The *timed language* $TL(A)$ accepted by A is defined as the set of timed words (traces) accepted by A: $TL(A) = \{u = (a_0, \gamma_0) \cdots (a_p, \gamma_p) \cdots, \exists \rho \in Runs(A) \text{ such that } tr(\rho) = u\}$.

The function *last locations* $LastLocs : (\Sigma \times \mathbb{R}_+)^* \rightarrow 2^L$ is defined as follows: $LastLocs(u) = \{l \in L, \exists \rho_f \in Runs(A) \text{ such that } tr(\rho_f) = u \text{ and } l = Last(\rho_f)\}$. The reason why this set is not always reduced to a singleton, is the non-determinism of the automaton i.e. if A is deterministic, then the last locations is a singleton. Let $V \subseteq TL(A)$ be a subset of timed languages, the last operation can be extended to be applied as follows: $LastLocs(V) = \bigcup_{u \in V} LastLocs(u)$.

$\ulcorner u \urcorner = \gamma_p$ represents the *last time* of the finite timed word $u = (a_0, \gamma_0)(a_1, \gamma_1) \cdots (a_p, \gamma_p)$.

Given two finite timed words $u, v \in (\Sigma \times \mathbb{R}_+)^*$, we denote by $u.v$ (or uv) the concatenation of u and v defined in the usual way. We say that $u \in (\Sigma \times \mathbb{R}_+)^*$ is a prefix of $v \in (\Sigma \times \mathbb{R}_+)^*$, denoted by $u \preceq v$, if there exists $x \in (\Sigma \times \mathbb{R}_+)^*$ such that $u \cdot x = v$. We note that $|u|$ stands for the length of the timed word u.

The *timed language* $TL(A_f)$ accepted by A_f is defined as follows $TL(A_f) = \{u = (a_0, \gamma_0) \cdots (a_p, \gamma_p), \exists \rho \in Runs(A_f) \text{ such that } tr(\rho) = u \text{ and } LastLocs(u) \in F\}$.

The *timed bounded language* $TL(A, \alpha)$ accepted by A, is defined as the set of timed words that are bounded by the duration α and accepted by A: $TL(A, \alpha) = \{u \in TL(A), \ulcorner u \urcorner \leq \alpha\}$. In the same way the timed bounded language accepted by A_f is defined as follows $TL(A_f, \alpha) = \{u \in TL(A_f), \ulcorner u \urcorner \leq \alpha\}$.

Given $L' \subseteq L$, the *timed bounded language with last locations* $TL_{L'}(A, \alpha)$ is defined as follows: $TL_{L'}(A, \alpha) = \{u \in TL(A, \alpha), LastLocs(u) \cap L' \neq \emptyset\}$.

The interface between a system and an observer is specified by a subset of observable actions $\Sigma_o \subseteq \Sigma$ and unobservable actions $\Sigma_u = \Sigma \setminus \Sigma_o$. The behavior visible by an observer is then defined by its projection denoted P_{Σ_o} that removes from a sequence in $(\Sigma \times \mathbb{R}_+)^*$ all events that are in $(\Sigma_u \times \mathbb{R}_+)$. Formally, $P_{\Sigma_o} : (\Sigma \times \mathbb{R}_+)^* \to (\Sigma_o \times \mathbb{R}_+)^*$ is defined as follows: where $P_{\Sigma_o}(\epsilon) = (\epsilon)$ and

$$\begin{cases} P_{\Sigma_o}(v.(a,\gamma)) = P_{\Sigma_o}(v) & \text{if } a \in \Sigma_u \\ P_{\Sigma_o}(v.(a,\gamma)) = P_{\Sigma_o}(v).(a,\gamma) & \text{otherwise} \end{cases} \quad (1)$$

where $v \in (\Sigma \times \mathbb{R}_+)^*, a \in \Sigma, \gamma \in \mathbb{R}_+$ and ϵ is the empty string. The latter projection can be extended to a set of timed words $P_{\Sigma_o}(V)$ where $V \in TL(A)$.

$[v] = \{u \in TL(A), P_{\Sigma_o}(u) = P_{\Sigma_o}(v)\}$ represents the set of timed words generated by A which have the same projection as v. It can be extended to be applied on a set of timed words $[V]$ where $V \subseteq TL(A)$.

Given a timed word $u = (a_0, \gamma_0)\ (a_1, \gamma_1) \cdots\ (a_p, \gamma_p)$, the untimed projection, $\pi(u) \in \Sigma^*$ is presented as follows: $\pi(u) = a_0\ a_1\ a_2 \cdots\ a_p$.

Moreover, we define that *u precedes v by one observable action*, denoted by $u \preceq_1 v$ if there exists $x \in (\Sigma \times \mathbb{R}_+)^*$ such that $ux = v$ and $|P_{\Sigma_o}(x)| = 1$.

In the paper, we consider that the timed automaton satisfies the non-Zeno property. By definition, a timed automaton model is said to be non-Zeno if all execution paths are non-Zeno. A Zeno path has an infinite number of discrete actions occurring in finite time (i.e. infinitely many actions take place in finite time) [21,22].

Opacity in the context of dense-time was introduced by [11]. The timed opacity is undecidable for real-time systems, where the opacity is an important privacy property. The timing of actions is crucial, as well as the timing of secret states disclosure.

Definition 4 (*Timed Opacity, [11]*)**.** Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) and G_A its TTS (Definition 3), with $\Sigma_o \subseteq \Sigma$ as the set of observable actions and $S \subseteq L$ as the set of secret locations.

The secret S is said to be timed opaque under the projection map P_{Σ_o} if:

$\forall u \in TL_S(A), \exists v \in TL_{L\setminus S}(A)$ such that $P_{\Sigma_o}(u) = P_{\Sigma_o}(v)$.

In the next section, we formally define a new approach of opacity called the timed bounded opacity, which is restricted to bounded dense-time systems. We consider that an attacker can see a truncated form of the generated timed sequences through an interface which inhibits the observation of some events. Thus, the attacker has partial knowledge of the events generated by A that corresponds to the projection of the generated sequence over the observable alphabet $\Sigma_o \subseteq \Sigma$. The attacker has the power to see only observable actions and their occurrence times. Moreover, we limit all timed words by a given upper bound α.

3. Timed bounded opacity

Timed bounded opacity is similar to the timed opacity property but is limited by an upper bound α. This property is satisfied if the attacker is not able to disclose the secret before the upper bound α. The timed bounded opacity is defined under the static projection as follows:

Definition 5 (*Timed Bounded Opacity*)**.** Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) and G_A its TTS (Definition 3), with $\Sigma_o \subseteq \Sigma$ as the set of observable actions, $S \subseteq L$ as the set of secret locations and $\alpha \in \mathbb{R}_+^*$

The secret S is said to be timed bounded opaque under the projection map P_{Σ_o} if:

$\forall u \in TL_S(A, \alpha), \exists v \in TL_{L\setminus S}(A, \alpha)$ such that $P_{\Sigma_o}(u) = P_{\Sigma_o}(v)$

Intuitively, we say that a secret is timed bounded opaque under a projection P_{Σ_o} if for every sequence u that ends in secret locations and is bounded by α (i.e. $u \in TL_S(A, \alpha)$), there exists another sequence v, that has the same projection as u but does not end in any secret location. That is:

$\forall u \in P_{\Sigma_o}(TL_S(A, \alpha)),\ LastLocs([u]) \not\subseteq S$ [11]

Equivalently, we say that a secret is timed bounded opaque under a projection P_{Σ_o} if the timed bounded language that ends in secret locations $TL_S(A, \alpha)$ is included in a timed bounded language that ends in non-secret locations $TL_{L\setminus S}(A, \alpha)$ under a projection P_{Σ_o}. A more formal lemma is given as follows:

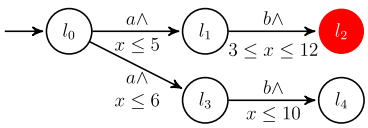


Fig. 1. Example of timed bounded opacity.

Lemma 3.1. *The secret S is timed bounded opaque under the projection map P_{Σ_o} if:*

$P_{\Sigma_o}(TL_S(A, \alpha)) \subseteq P_{\Sigma_o}(TL_{L\setminus S}(A, \alpha))$

Example 3.1. Let A be a timed automaton as shown in Fig. 1, and let $L = \{l_0, l_1, l_2, l_3, l_4\}$ be its set of locations, $X = \{x\}$ the set of clocks, $S = \{l_2\}$ and $\Sigma_o = \Sigma = \{a, b\}$. Thus, P_{Σ_o} corresponds to the identity function.

It is clear that if $\alpha = 10$, S is timed bounded opaque. For each timed word that is bounded by α and ends in a secret location, there is an equivalent timed word (in terms of projection P_{Σ_o}) that ends in a non-secret location. This is primarily due to the nondeterminism of this automaton. For instance, the intruder cannot conclude the timed word $(a, 2)\ (b, 9)$ carried by accepting runs $\rho = q_0 \xrightarrow{(a,2)} q_1 \xrightarrow{(b,7)} q_2$ ending in location l_2 and $\rho' = q_0 \xrightarrow{(a,2)} q_3 \xrightarrow{(b,7)} q_4$ ending in location l_4. However, if we consider $\alpha = 11$, the secret S is not timed bounded opaque. For instance, the timed word $(a, 2)\ (b, 11)$ carried by accepting run $\rho = q_0 \xrightarrow{(a,2)} q_1 \xrightarrow{(b,9)} q_2$ ending in location l_2 does not have an equivalent timed word ending in a non-secret location.

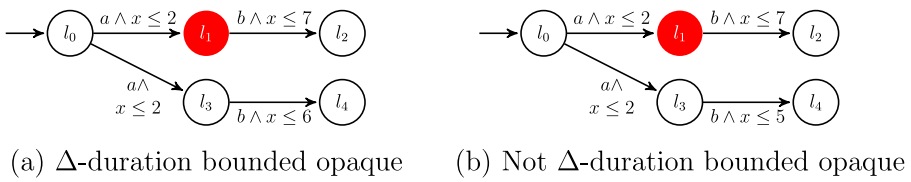
In general, the timed opacity is undecidable for timed automata. In the next theorem, we prove the decidability of timed bounded opacity for timed automata bounded by α.

Theorem 3.2. *The timed bounded opacity is decidable for non-Zeno timed automata and 2-EXPSPACE-Complete.*

Proof. The timed bounded opacity can be reduced to time-bounded language inclusion using Lemma 3.1. that is proved decidable and 2-EXPSPACE-complete in [15]. Therefore, the timed bounded opacity is 2-EXPSPACE-complete for non-Zeno timed automata.

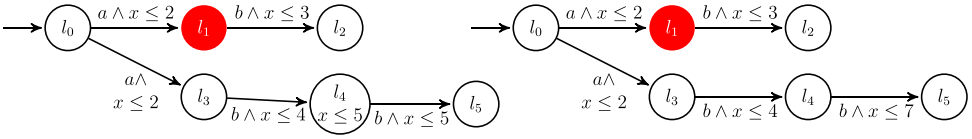
The timed bounded opaque property answers the following question: Can the attacker disclose in real-time secrets that occur before α? In addition, we can consider another interesting question in the same timed bounded context: Can the attacker disclose, in a duration Δ (called Δ window), secrets that occur before α?

In the next section, we propose a new approach for opacity of the dense-time systems. This approach is called Δ-duration bounded opacity. Moreover, this approach is similar to K-step weak opacity [23]. Intuitively, these approaches take into account the opacity of the secret in the past. In K-step opacity, the knowledge of the secret becomes worthless after the observation of a given number of actions. However, in Δ-duration bounded opacity, the knowledge of the secret becomes worthless after a given duration Δ.



(a) Δ-duration bounded opaque (b) Not Δ-duration bounded opaque

Fig. 2. Δ-duration bounded opacity and projection.



(a) Not Δ-duration bounded opaque (b) Δ-duration bounded opaque

Fig. 3. Δ-duration bounded opacity and the secret disclosure because of the lack of actions.

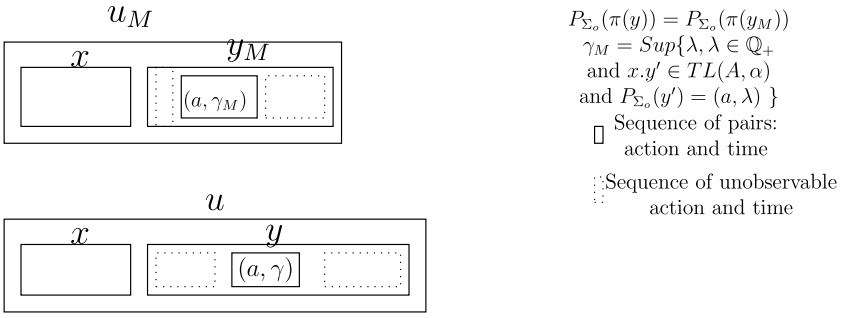


Fig. 4. Explanation of the function *LastSup*.

4. Δ-duration bounded opacity

First, we consider that the attacker must be unable to disclose the secret before the upper bound α, which corresponds to the property of timed bounded opacity in Section 3. Second, the secret must be kept hidden for at least a given duration Δ. To better understand the verification of Δ-duration bounded opacity, Figs. 2 and 3 outline four examples of timed automata and the result of Δ-duration verification, where $\alpha = 100$, $\Delta = 4$ and $S = \{l_1\}$.

In Fig. 2.(a), the secret S is Δ-duration bounded opaque. For each timed word that transits through the secret location before the expiry of the Δ window, there exists an equivalent timed word with an identical projection, which does not pass through the secret location. It is noted that the timed word $(a, 2)(b, 6.5)$ transits through the secret location and there is no equivalent timed word under projection P_{Σ_o} that avoids the secret location. This timed word does not affect the fact that the secret S is Δ-duration bounded opaque since it is outside of the Δ window $(6.5 - 2 > \Delta)$. Thus, the attacker cannot infer that the system passes through any secret location before the Δ expiry. In Fig. 2.(b), the secret S is not Δ-duration bounded opaque due to the existence of a timed word (for example $(a, 2)(b, 5.5)$) that passes through the secret location, and there is no equivalent timed word under projection P_{Σ_o} that avoids the secret location. Hence, the attacker is aware of the system passing through the secret location before the expiry of the Δ window.

In Fig. 3.(a), the secret S is not Δ-duration bounded opaque, despite having an equivalent timed word under projection P_{Σ_o} that avoids the secret location, for each timed word that passes through the secret location before the Δ-duration. Hence, the attacker can disclose that the system passes through the secret location within the Δ-duration limit. In the case of $(a, 1.5)(b, 3)$ timed word, the Δ window stopwatch starts computation at $t = 1.5$ and expires at $t = 5.5$. During the interval $]5, 5.5]$, the attacker confirms that the system passes through the secret location l_2. This is because the occurrence of a second b action (for example, $(a, 1.5)(b, 3)(\mathbf{b}, \mathbf{5})$) is inevitable if the system does not pass through the secret location. It was observed that the second b action always occurs before the Δ window expires. In Fig. 3.(b), the secret S is Δ-duration bounded opaque. Thus, the attacker cannot infer that the system passes through the secret location within the Δ window. Compared to the timed automaton in 3.(a), a second action b may occur after the expiration of the Δ window (for example, $(a, 1.5)(b, 3)(b, 7)$). These examples suggest that the intruder can infer the secret by detecting the absence of specific actions in the non-secret path before the expiration of the Δ window.

All discussed examples have helped define the Δ-duration bounded opacity. Before defining this property, some required definitions are introduced below.

Definition 6 (*K-delay Respect*)**.** Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) and $K \in \mathbb{R}_+$ be a constant value. Given $u, v \in TL(A)$ where $u \preceq v$. We say that u respects v by K-delay (K-**delay respect**) if the difference between the last time of u and v is less than or equal to K, denoted by $u \vDash_K v$. More formally,

$u \vDash_K v$ if $u, v \in TL(A)$ and $u \preceq v$ and $\ulcorner v \urcorner - \ulcorner u \urcorner \leq K$

Example 4.1. Let $u = (a, 2)\ (b, 3)\ (b, 6)$ and $v = (a, 2)\ (b, 3)\ (b, 6)\ (a, 10)\ (b, 12)\ (a, 14)$ be two timed words. In the case of $K = 10$, $\ulcorner v \urcorner - \ulcorner u \urcorner = 14 - 6 = 8 \leq K = 10$. Thus, u respects v by 10-delay. However, in the case of $K = 5$, $\ulcorner v \urcorner - \ulcorner u \urcorner = 8 > K = 5$ and the property of K-delay respect is not satisfied.

Definition 7 (*LastSup Timed Word*)**.** Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1), Σ_o be the set of observable actions and $\alpha \in \mathbb{R}_+^*$ be a constant value. Given $u \in TL(A, \alpha)$ is a timed word. We say that u_M is LastSup timed word of u if the last time of u is supremum relative to the last transition of this timed word accepted by the timed

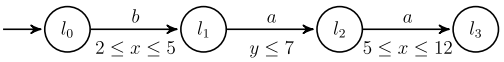


Fig. 5. LastSup timed word example.

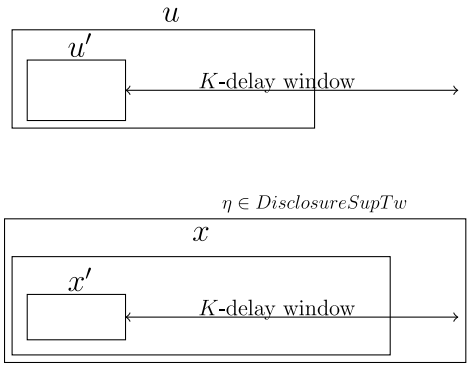


Fig. 6. Explanation of the function $DisclosureSupTW$.

automaton A, denoted by $LastSup(u) \in (\Sigma \times \mathbb{R}_+)^*$. More formally,

$$LastSup(u) = u_M \text{ if } \begin{cases} x \preceq_1 u \text{ where } u = x \cdot y \text{ and } P_{\Sigma_o}(y) = (a, \gamma) \\ x \preceq_1 u_M \text{ where } u_M = x \cdot y_M \text{ and} \\ P_{\Sigma_o}(y_M) = (a, \gamma_M) \\ \gamma_M = Sup\{\lambda, \lambda \in \mathbb{R}_+ \text{ and} \\ \quad x \cdot y' \in TL(A, \alpha) \text{ and } P_{\Sigma_o}(y') = (a, \lambda)\} \end{cases} \tag{2}$$

Fig. 4 explains the function $LastSup$. We note that the timed word generated by $LastSup$ cannot be an element of the timed language accepted by the timed automaton, if the corresponding constraint to the last action is defined by a strict inequality.

Example 4.2. We consider $u = (b, 3)\ (a, 6)\ (a, 10)$ a timed word accepted by the timed automaton shown in Fig. 5. The LastSup timed word of u is $u_M = LastSup(u) = (b, 3)\ (a, 6)\ (a, 12)$. The supremum time to execute the transition $e_2 = (l_2, a, g_2, r_2, l_3)$ when $g_2 : 5 \le x \le 12$ is 12.

Definition 8 (*DisclosureSupTW*). Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton, $K \in \mathbb{R}_+$ be a constant value, $x', u' \in TL(A, \alpha)$ and $x, u \in TL(A, \alpha + K)$ where $u' \vDash_K u$ and $x' \vDash_K x$. The operator $DisclosureSupTW(u, u', x, x', K)$ is defined as follows:

$DisclosureSupTW(u, u', x, x', K) =$

$$\begin{cases} \emptyset & \text{if } \exists y \in TL(A, \alpha + K), \\ & \quad u \preceq_1 y \text{ and } u' \vDash_K y \\ \{LastSup(\chi), & \\ \quad \chi \in TL(A, \alpha + K) \text{ and} & \\ \quad x \preceq_1 \chi \text{ and } \ulcorner LastSup(\chi) \urcorner - \ulcorner x' \urcorner < K\} & \text{Otherwise} \end{cases} \tag{3}$$

Fig. 6 explains the $DisclosureSupTW$ operator that allows us to find the timed words that enable the intruder to infer the secret within a K window limit. It returns a set of timed words or an empty set. If there exists a timed word y longer than u by one observable action and u' respects y by K-delay, the $DisclosureSupTW$ operator returns the empty set. Otherwise, for each timed word χ longer than x by one observable action and the duration between the last time of timed words χ and x' is less than K, then $DisclosureSupTW$ returns the set of $LastSup$ timed words.

Example 4.3. Fig. 7 shows the timed automaton $A = (L, l_0, X, \Sigma, I, T)$, where $X = \{x, y\}$, $\Sigma = \Sigma_o = \{a, b\}$, and $K = 4$. Let $x, x', u, u' \in TL(A)$ be timed words accepted by A, where:

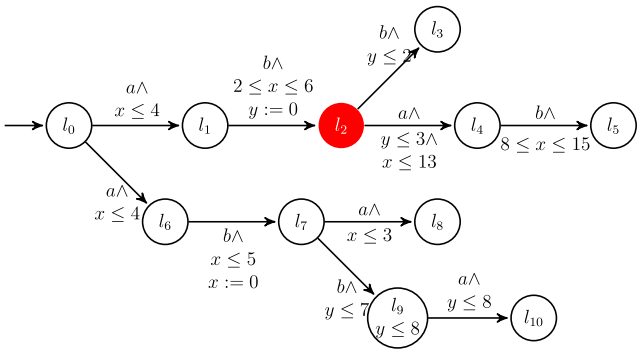


Fig. 7. Timed automaton example.

- Case 1:
  $u = (a, 3)\ (b, 5)\ (b, 7) = u'.(b, 7)$ is carried by accepting run $\rho_u = q_0 \xrightarrow{(a,3)} q_1 \xrightarrow{(b,2)} q_2 \xrightarrow{(b,2)} q_3$ and $u' = (a, 3)\ (b, 5)$.
  $x = (a, 3)\ (b, 5)\ (b, 7) = x'.(b, 7)$ is carried by accepting run $\rho_x = q_0 \xrightarrow{(a,3)} q_6 \xrightarrow{(b,2)} q_7 \xrightarrow{(b,2)} q_9$ and $x' = (a, 3)\ (b, 5)$.
  In this case, there exists a timed word χ longer than x by one observable action and x' respects $LastSup(\chi)$ by 4-delay.
  Then, $DisclosureSupTW(u, u', x, x', K) = \{(a, 3)(b, 5)(b, 7), (a, 8)\}$.
- Case 2:
  $u = (a, 3)\ (b, 5)\ (a, 6) = u'.(a, 6)$ is carried by accepting run $\rho_u = q_0 \xrightarrow{(a,3)} q_1 \xrightarrow{(b,2)} q_2 \xrightarrow{(a,1)} q_4$ and $u' = (a, 3)\ (b, 5)$.
  $x = (a, 3)\ (b, 5)\ (a, 6) = x'.(a, 6)$ is carried by accepting run $\rho_x = q_0 \xrightarrow{(a,3)} q_6 \xrightarrow{(b,2)} q_7 \xrightarrow{(a,1)} q_8$ and $x' = (a, 3)\ (b, 5)$.
  In this case, there exists $y = (a, 3)\ (b, 5)\ (a, 6)(b, 8) = x.(b, 8)$ longer than u by one observable action and u' respects y by 4-delay.
  Then, $DisclosureSupTW(u, u', x, x', K) = \emptyset$

In the following discussion, we present the formal definition of Δ-duration bounded opacity under the static projection.

Definition 9 (*Δ-Duration Bounded Opacity*). Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) and G_A its TTS (Definition 3), where $\Sigma_o \subseteq \Sigma$ is the set of observable actions, $S \subseteq L$ is the set of secret locations and $\alpha, \Delta \in \mathbb{R}_+^*$ are constant values.

The secret S is said to be Δ-duration bounded opaque under the projection map P_{Σ_o} if

$$\begin{cases} \forall u \in TL(A, \alpha + \Delta),\ \forall u' \in TL_S(A, \alpha) \text{ such that } u' \vDash_\Delta u\ , \\ \exists x \in TL(A, \alpha + \Delta),\ \exists x' \in TL_{L \setminus S}(A, \alpha) \text{ such that } x' \vDash_\Delta x \text{ and} \\ P_{\Sigma_o}(u) = P_{\Sigma_o}(x) \text{ and } P_{\Sigma_o}(u') = P_{\Sigma_o}(x') \text{ and} \\ DisclosureSupTW(u, u', x, x', \Delta) = \emptyset \end{cases} \tag{4}$$

Intuitively, we say that the secret is Δ-duration bounded opaque under the projection map P_{Σ_o} if the intruder is not able to infer that the system passes through a secret location before Δ window expires. Fig. 8 illustrates how to verify Δ-duration bounded opacity. The verification requires to check the opacity property prior to expiration of Δ window. In the previous step, $DisclosureSupTW$ remains empty. After the end of the step, the determination of $DisclosureSupTW$ set helps verify whether the secret is disclosed. If $DisclosureSupTW$ is not an empty set, then the system is not Δ-duration bounded opaque as there exist timed words that allow the intruder to confirm that the system has passed through a secret location within a Δ window limit. Otherwise, the system is Δ-duration bounded opaque. More details will be explained in Example 4.4.

Example 4.4. We consider the following cases relative to Fig. 7 where $S = \{l_2\}$:

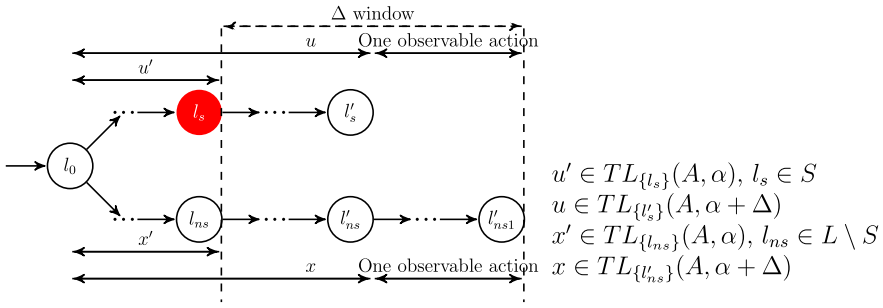


Fig. 8. Δ-duration bounded opacity intuitive presentation.

- Case 1: $\alpha = 5$ and $\Delta = 4$, the secret S is not Δ-duration bounded opaque because the definition is not verified in the following example:
  $u = (a,3)\ (b,5)\ (b,7) = u'.(b,7) \in TL(A,9)$ is carried by accepting run $\rho_u = q_0 \xrightarrow{(a,3)} q_1 \xrightarrow{(b,2)} q_2 \xrightarrow{(b,2)} q_3$ and $u' = (a,3)\ (b,5) \in TL_S(A,5)$.
  There exists x' carried by accepting run $\rho_{x'} = q_0 \xrightarrow{(a,3)} q_6 \xrightarrow{(b,2)} q_7$,
  There exists x carried by accepting run $\rho_x = q_0 \xrightarrow{(a,3)} q_6 \xrightarrow{(b,2)} q_7 \xrightarrow{(b,2)} q_9$ and
  $DisclosureSupTW(u, u', x, x', 4) = \{(a,3)\ (b,5)\ (b,7)\ (a,8)\} \neq \emptyset$ traces the run $\rho = q_0 \xrightarrow{(a,3)} q_6 \xrightarrow{(b,2)} q_7 \xrightarrow{(b,2)} q_9 \xrightarrow{(a,1)} q_{10}$.
- Case 2: $\alpha = 5$ and $\Delta = 2$, the secret S is Δ-duration bounded opaque.
  All timed words $u' \in TL_S(A,5)$ and $u \in TL(A,7)$, such that $u' \vDash_\Delta u$, there exist x and x' that have the same projection as u and u' respectively, $x' \in TL_{L\setminus S}(A,5)$ and $x \in TL(A,7)$ and $DisclosureSupTW(u, u', x, x', 2) = \emptyset$.

The property of Δ-duration bounded opacity is a general case of the timed bounded opacity. If the secret is Δ-duration bounded opaque, then the secret is timed bounded opaque.

To prove the decidability of Δ-duration bounded opacity, we needed to transform a timed automaton accepting all timed words passing through secret locations (respectively non secret locations) and respecting the duration Δ to an equivalent timed automaton with final locations. These final locations correspond to the transformed secret locations (respectively non secret locations) and their reached locations during the duration Δ.

4.1. Transformation of timed automaton

Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) where $T = \{e_0, e_1, \dots, e_k\}$ is the set of transitions with $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2}) \in T$, $0 \le i \le k$ and $L = \{l_0, l_1, \dots, l_p\}$ is the set of locations. $S \subseteq L$ is a subset of locations and $\alpha, \Delta \in \mathbb{R}^*_+$ are constant values. TL is the timed bounded language accepted by A where each timed word passes through a location in S and respects the duration Δ. $TL = \{u, \forall u' \in TL_S(A,\alpha),\ u' \vDash_\Delta u\}$.

The purpose is to construct a timed automaton A'_f where $TL = TL(A'_f, \alpha + \Delta)$. New clocks are introduced in each transition ending in a secret location. These clocks are used to distinguish whether the current location respects Δ duration or not. The aforementioned locations will be fragmented according to the new introduced clocks as explained in the following.

Before that, some functions used in the transformation process are defined as follows:

- The function $Input(l', A)$ returns the subset of transitions ending in l'. $Input(l', A) = \{e \in T, e = (l, a, g, r, l')\}$.
- The function $Succs(l, A)$ returns all successor locations starting from l. By extension, $Succs(S, A)$ is defined by $Succs(S, A) = \cup_{l \in S} Succs(l, A)$.
- The function $Clocks(e, A)$ returns the set of clocks which is relevant in the transition $e = (l, a, g, r, l')$. $Clocks(e, A) = \{t \in X,\ t \text{ in } g\}$. By extension $Clocks(e, A)$ is defined by $Clocks(E, A) = \cup_{e \in E} Clocks(e, A)$ where $E \subseteq T$ is a subset of transitions.
- The function $Comb(Cl)$ returns a table including all possible combinations of $|Cl|$-bits where 0-bit and 1-bit correspond respectively to $t_{i_2,j} > \Delta$ and $t_{i_2,j} \le \Delta$ (Cl is a subset of clocks). The Function $Comb(Cl, k)$ returns the kth combination (k is the index of the combination) of the clocks which is relevant in $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2})$. For example, if $Cl = \{t_{1,1}, t_{1,2}\}$, all combinations returned by $Comb(Cl)$ are $1 - (00)$, $2 - (01)$, $3 - (10)$, $4 - (11)$, the $Comb(Cl, 2) = (10)$ that corresponds to $t_{1,1} \le \Delta$ and $t_{1,2} > \Delta$.

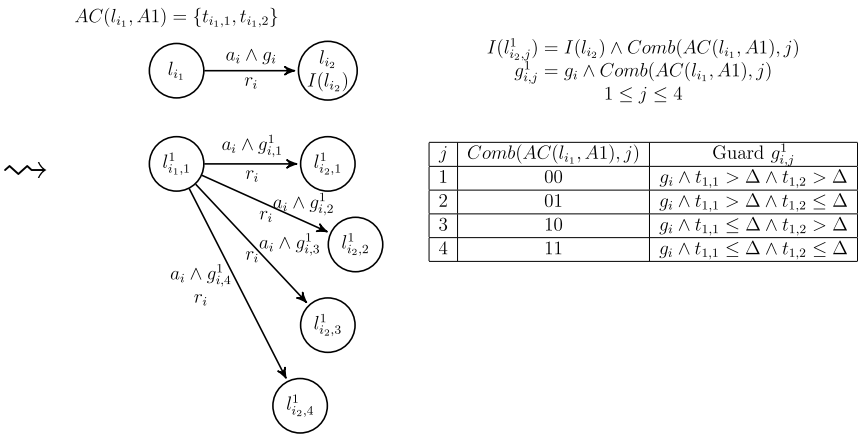
We construct an equivalent timed automaton with final locations $A'_f = (L', l_0, X', \Sigma, I', F, T') = Trans(A, S)$. This construction consists of the following steps:

- In the first step, we construct the timed automaton $A1 = (L, l_0, X', \Sigma, I, T1)$ where
  - $T1 = \{e'_0, e'_1, \dots, e'_k\}$ is the set of transitions.
  - $X' = X \cup \{t_{i_2,j},\ l_{i_2} \in S \text{ and } 1 \le j \le |Input(l_{i_2}, A1)|\}$. We note that the new added clock in transition is defined by $NewCl(e_i) = Clocks(e_i, A1) \setminus X$ where $e_i \in Input(l_{i_2}, A1)$.
  - $\forall e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2}) \in T$ and $\forall 1 \le j \le |Input(l_{i_2}, A1)|$, we construct $e'_i = (l_{i_1}, a_i, g'_i, r'_i, l_{i_2}) \in T1$ where

$$g'_i = \begin{cases} g_i \wedge t_{i_2,j} \le \alpha & \text{if } l_{i_2} \in S \text{ and } t_{i_2,j} = NewCl(e_i) \\ g_i & \text{otherwise} \end{cases}$$

$$\text{and } r'_i = \begin{cases} r_i \cup \{t_{i_2,j}\} & \text{if } l_{i_2} \in S \text{ and } t_{i_2,j} = NewCl(e_i) \\ r_i & \text{otherwise} \end{cases}$$

  The function $AC(l_{i_2}, A1)$ returns only the new added clocks in $\{t_{i_2,1}, t_{i_2,2}, \cdots, t_{i_2,j}\}$ that are relevant in all transitions $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2})$ ending in l_{i_2}. $NC(l_{i_2}, A1) = Clocks(Input(l_{i_2}, A1), A1) \setminus X$.
- In this step, we construct a timed automaton $A2 = (L2, l_0, X', \Sigma, I2, F, T2)$ from $A1 = (L, l_0, X', \Sigma, I, F, T1)$ and S. Each location in $A2$ is labeled by $l^m_{i,j}$ where i, j and m refer respectively to the location l_i in $A1$, the combination index of new clocks in $NC(l_i, A1)$ and a counter identifying a set of new clocks added to the invariant of l_i. Each transition in $A2$ is labeled by $e^m_{i,j}$ where i, j and m refer respectively to the transition $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2})$ in $A1$, the combination index of new clocks in $NC(l_{i_1}, A1)$ and a counter identifying a set of new clocks added to the invariant of l_{i_1}. For each transition $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2}) \in T1$, we construct all equivalent transitions in $T2$ as follows:
  - If $l_{i_1} \notin S \cup Succs(S, A1)$ and $l_{i_2} \notin Succs(S, A1)$ then we create the location $l^1_{i_2,1} \in L2$ (homologous to $l_{i_2} \in L$) and the transition $e^1_{i,1} = (l^1_{i_1,1}, a_i, g_i, r_i, l^1_{i_2,1}) \in T2$.
  - If $l_{i_1} \in S$ then we check whether the location l_{i_2} has a set of homologous locations $\{l^m_{i_2,1}, l^m_{i_2,2}, \dots, l^m_{i_2,j}, \dots, l^m_{i_2,2^{nc}}\}$ having the invariant $I(l_{i_2}) \wedge Comb(NC(l_{i_1}, A1), j)$ where $1 \le$



j	$Comb(AC(l_{i_1}, A1), j)$	Guard $g^1_{i,j}$
1	00	$g_i \wedge t_{1,1} > \Delta \wedge t_{1,2} > \Delta$
2	01	$g_i \wedge t_{1,1} > \Delta \wedge t_{1,2} \leq \Delta$
3	10	$g_i \wedge t_{1,1} \leq \Delta \wedge t_{1,2} > \Delta$
4	11	$g_i \wedge t_{1,1} \leq \Delta \wedge t_{1,2} \leq \Delta$

Fig. 9. Example of transition transformation.

$j \leq 2^{nc}$ and $nc = |NC(l_{i_1}, A1)|$. If the aforementioned set exists then we create 2^{nc} transitions $e^1_{i,j} = (l^1_{i_1,1}, a_i, g^1_{i,j}, r_i, l^m_{i_2,j})$. Otherwise, we create 2^{nc} locations $l^{m+1}_{i_2,j}$ with invariant $I(l^{m+1}_{i_2,j}) = I(l_{i_2}) \wedge Comb(NC(l_{i_1}, A1), j)$ and 2^{nc} transitions $e^1_{i,j} = (l^1_{i_1,1}, a_i, g^1_{i,j}, r_i, l^{m+1}_{i_2,j})$ where $g^1_{i,j} = g_i \wedge Comb(NC(l_{i_1}, A1), j)$. For example as shown in Fig. 9, if new clocks in the location $l_{i_1} \in S$ (corresponds to $l^1_{i_1,1} \in L2$) are $NC(l_{i_1}, A1) = \{t_{i_1,1}, t_{i_1,2}\}$ then from the location $l^1_{i_1,1}$ there are four transitions $\{e^1_{i,1}, e^1_{i,2}, e^1_{i,3}, e^1_{i,4}\}$ where $\forall 1 \leq j \leq 2^2$, $e^1_{i,j} = (l_{i_1}, a_i, g^1_{i,j}, r_i, l^1_{i_2,j}) \in T2$, $g^1_{i,j} = g_i \wedge Comb(NC(l_{i_1}, A1), j)$ and the invariant of $l^1_{i_2,j}$ is $I(l^1_{i_2,j}) = I(l_{i_2}) \wedge Comb(NC(l_{i_1}, A1), j)$.

- If $l_{i_1} \in Succs(S, A1)$, and there exists a set of locations $\{l^{m_1}_{i_1,1}, l^{m_1}_{i_1,2}, \ldots, l^{m_1}_{i_1,j_1}, \ldots, l^{m_1}_{i_1,2^{nc_1}}\} \in L2$ homologous to l_{i_1} then we check if the location l_{i_2} has a set of homologous locations $\{l^{m_2}_{i_2,1}, l^{m_2}_{i_2,2}, \ldots, l^{m_2}_{i_2,j_2}, \ldots, l^{m_2}_{i_2,2^{nc}}\}$ having the invariant $I(l_{i_2}) \wedge Comb(ac_{i_1}, j_2)$ where $ac_{i_1} = NC(l_{i_1}, A1) \cup NC(l^{m_1}_{i_1,1}, A2)$, $1 \leq j_2 \leq 2^{nc}$ and $nc = |ac_{i_1}|$. If the aforementioned set exists then for each location $l^{m_1}_{i_1,j_1} \in L2$ we create 2^{nc} transitions $e^{m_1}_{i,j} = (l^{m_1}_{i_1,j_1}, a_i, g^{m_1}_{i,j}, r_i, l^{m_2}_{i_2,j_2})$ where $g^{m_1}_{i,j} = g_i \wedge Comb(ac_{i_1}, j_2)$ and $1 \leq j \leq 2^{nc+nc_1}$. Otherwise, we create 2^{nc} locations $l^{m_2+1}_{i_2,j_2}$ with the invariant $I(l^{m_2+1}_{i_2,j_2}) = I(l_{i_2}) \wedge Comb(ac_{i_1}, j_2)$ and 2^{nc+nc_1} transitions $e^{m_1}_{i,j} = (l^{m_1}_{i_1,j_1}, a_i, g^{m_1}_{i,j}, r_i, l^{m_2+1}_{i_2,j_2})$ where $g^{m_1}_{i,j} = g_i \wedge Comb(ac_{i_1}, j_2)$, $1 \leq j_1 \leq 2^{nc_1}$, $1 \leq j_2 \leq 2^{nc}$ and $1 \leq j \leq 2^{nc+nc_1}$.
- if $l_{i_1} \notin S \cup Succs(S, A1)$ and $l_{i_2} \in Succs(S, A1)$ then for each equivalent location $l^m_{i_2,j_2}$ we create the transition $e^1_{i,j} = (l^1_{i_1,1}, a_i, g_i, r_i, l^m_{i_2,j_2})$.

- In the final step, all locations and transitions which are not reachable from the initial location have to be removed to obtain the timed automaton with final locations $A'_f = (L', l_0, X', \Sigma, I', F, T')$ where $F = \{l^m_{i,j} \in L2, l_i \in Succs(S, A1)$ and $\forall t \in NC(l^m_{i,j}, A2), t \leq \Delta$ satisfies $I(l^m_{i,j})\} \cup \{l^1_{i,1}, l_i \in S\}$ is the set of final locations.

Theorem 4.1. *Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton, $S \subseteq L$ be a set of locations and $\alpha, \Delta \in \mathbb{R}_+$ be constant values. The transformed timed automaton $A'_f = (L', l_0, X', \Sigma, I', F, T') = Trans(A, S)$ accepting the language $TL(A'_f)$ is equivalent to A accepting the language $TL = \{u, \forall u' \in TL_S(A, \alpha)\ u' \vDash_\Delta u\}$.*

Proof. We consider that $A = (L, l_0, X, \Sigma, I, T)$ is a timed automaton, S is a subset of locations ($S = \{l_1\}$) and $\alpha, \Delta \in \mathbb{R}_+$ are constant values. We assume that we have one successor location from l_1, denoted by l_2, and $|NC(l_1, A)| = 1$. The first transformation from A to A_1 consists of adding the constraint $t_{1,1} \leq \alpha$ in the transition ending in l_1. By constructing the timed automaton A_2, the location $l^1_{1,1}$ homologous to l_1 has two output transitions. For the transition $e_1 = (l_1, a_1, g_1, r_1, l_2) \in T$, we construct two transitions $e^1_{1,1} = (l^1_{1,1}, a_1, g_1 \wedge t_{1,1} \leq \Delta, r_1, l^1_{2,1})$ and $e^1_{1,2} = (l^1_{1,1}, a_1, g_1 \wedge t_{1,1} > \Delta, r, l^1_{2,2})$ where all transition guards correspond to the guard g_1 of the transition e_1. Therefore, all timed words passing through the location l_1 before the duration α and ending in l_2 during Δ are accepted by A and also accepted by the timed automaton $A'_f = (L2, l_0, X \cup \{t_{1,1}\}, \Sigma, I2, F, T2)$ where $L2 = (L \setminus \{l_1, l_2\}) \cup \{l^1_{1,1}, l^1_{2,1}, l^1_{2,2}\}$, $I2 = (I \setminus \{I(l_2)\}) \cup \{I(l^1_{2,1}), I(l^1_{2,2})\}$, $T2 = (T \setminus \{e_1\}) \cup \{e^1_{1,1}, e^1_{1,2}\}$ and $F = \{l^1_{1,1}, l^1_{2,1}\}$.

Assuming that we have p successor locations from l_1, the previous case will be applied for each successor location. For a finite subset of locations S and for a subset of successor locations $Succs(S, A)$, we maintain the same results obtained with the original timed automaton.

The space complexity of the construction of the equivalent timed automaton is exponential. The construction Algorithm is presented in Appendix.

In the next section, we prove the complexity of Δ-duration bounded opacity.

4.2. Decidability of Δ-duration bounded opacity

We have shown, in the previous section, that the timed bounded opacity is decidable for timed automaton. In this section, we prove the decidability of Δ-duration bounded opacity as follows:

Theorem 4.2. *Let $A = (L, l_0, X, \Sigma, I, T)$ be a timed automaton (Definition 1) where $S \subseteq L$ is the set of secret locations and $\alpha, \Delta \in \mathbb{R}^*_+$ are constant values.*

The Δ-duration bounded opacity is decidable for timed automaton A and 2-EXPSPACE-complete.

Proof. The proof of Δ-duration bounded opacity decidability is based on the two following sufficient conditions:

1. The first condition (C1) states the first part of Δ-duration bounded opacity definition (Definition 9), that corresponds to the opacity property verification within the Δ window where $DisclosureSupTW(u, u', x, x', \Delta) = \emptyset$, as follows:

$$\begin{cases} \forall u \in TL(A, \alpha + \Delta),\ u' \in TL_S(A, \alpha) \text{ such that } u' \vDash_\Delta u\ , \\ \exists x \in TL(A, \alpha + \Delta),\ \exists x' \in TL_{L \setminus S}(A, \alpha) \text{ such that} \\ x' \vDash_\Delta x,\ P_{\Sigma_o}(u) = P_{\Sigma_o}(x) \text{ and } P_{\Sigma_o}(u') = P_{\Sigma_o}(x') \end{cases} \quad \text{(C1)}$$

(5)

Let NS be a subset of non secret locations $NS = \{LastLocs(u) \in L \setminus S,\ \exists v \in TL_S(A, \alpha)$ and $P_{\Sigma_o}(u) = P_{\Sigma_o}(v)\}$. According to

Theorem 4.1, we construct two timed automata with final locations $A_f = (LA, l_0, XA, \Sigma, IA, FA, TA)$ and $B_f = (LB, l_0, XB, \Sigma, IB, FB, TB)$ equivalent to A with subsets of locations S and NS respectively. Therefore, $TL(A_f)$ and $TL(B_f)$ are two timed bounded languages accepted by A_f and B_f respectively. According to [15], the timed bounded inclusion problem for timed automata ($P_{\Sigma_o}(TL(A_f)) \subseteq P_{\Sigma_o}(TL(B_f))$) is decidable and 2-EXPSPACE-complete. Thus, the verification of the (C1) is 2-EXPSPACE-complete.

2. The condition (C2) states the second part of Definition 9, as follows:

$$DisclosureSupTW(u, u', x, x', \Delta) = \emptyset \qquad \text{(C2)} \tag{6}$$

In this condition, it is necessary to determine the set of locations that are reachable from non-secret locations before the expiry of Δ window. The determination of this set is based on the $DisclosureSupTW$ operator that returns accepted or not accepted timed words by the timed automaton A. Then, the obtained result will be split into two subsets: the first subset $TWMax$ including the accepted timed words by A and the second subset $TWSup$ including the not-accepted timed words by A. The corresponding last locations of each subset ($TWMax$, $TWSup$) are denoted by Loc_{TWMax} and Loc_{TWSup}, respectively. Formally,

$Loc_{TWMax} = \{LastLocs(\chi), \chi \in TWMax\}$

$Loc_{TWSup} = \{LastLocs(\chi), \exists \chi \in TL(A, \alpha + \Delta)$ and $LastSup(\chi) \in TWSup\}$

According to the Theorem of reachability in [18], the determination of previous subsets is decidable and PSPACE-complete.

Therefore, the verification of Δ-duration bounded opacity for the secret S in timed automaton A will be obtained by (C1), the verification of timed bounded opacity for the secret S' in timed automaton A', and (C2), the reachability of any locations in Loc_{TWMax} and Loc_{TWSup} subsets. Thus, the Δ-duration bounded opacity is decidable and 2-EXPSPACE-complete.

In the next section, we use timed bounded model checking to verify the proposed properties of timed bounded opacity.

5. Verification of properties using timed bounded model checking

We start by defining the necessary notions used in timed bounded model checking. Let $MSO(<,+1)$ be the monadic second-order logic over the structure $([0, floor(\alpha)+1[, <, 1)$ where $+1(x, y)$ holds iff $x+1 = y$. For simplicity $[0, floor(\alpha)+1[$ is denoted by $[0, N[$ where $N = floor(\alpha)+1 \in \mathbb{N}$. The monadic second-order logic includes the vocabulary first-order variables $x_1, x_2, \ldots$, the monadic predicate variables $X_1, X_2, \ldots$, and the binary relations $+1$ and $<$. The first-order quantifiers are $\forall x$ and $\exists x$. The second-order quantifiers are $\forall X$ and $\exists X$. An order-theoretic sublogics presented by the monadic second-order logic $MSO(<)$ uses all the second-order monadic formulas that omit the $+1$ relation. Therefore the grammar of MSO formulas is obtained as follows:

$\varphi ::= True \mid x < y \mid +1(x, y) \mid P(x) \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \forall x\varphi \mid \forall P\varphi$.

Let MP be a set of monadic predicates, and $\wp \subseteq MP$ is a finite set of monadic predicates. A flow (or signal) over $\wp$ is a function $f : [0, N[\rightarrow 2^{\wp}$ that is finitely variable. It corresponds to the interpretation of the monadic predicate P in $\wp$ ($P \in f(t)$, $t \in [0, N[$).

In this Section, we present the timed word $(a_0, \gamma_0) \cdots (a_p, \gamma_p)$ by $(\langle a_0 \cdots a_p \rangle, \langle \gamma_0 \cdots \gamma_p \rangle)$. Let $\wp = \Sigma$, if $0 \leq i \leq p$ and $\gamma_i < N$ then $f(\gamma_i) = \{a_i\}$ otherwise $f(\gamma) = \emptyset$.

Let φ be an $MSO(<,+1)$ formula and $\{x_1, x_2, \ldots, x_p\}$ be free first-order variables appearing in φ. The satisfaction relation $(f, \gamma) \Vdash \varphi(x)$ is defined by $\varphi \in f(\gamma)$.

Metric Temporal Logic MTL is an extension of LTL adding the indexing of the interval $\jmath$, that comprises the following temporal formulas:

$\theta ::= True \mid P \mid \theta_1 \wedge \theta_2 \mid \theta_1 \vee \theta_2 \mid \neg\theta \mid \Diamond_{\jmath}\theta \mid \Box_{\jmath}\theta \mid \theta_1 U_{\jmath} \theta_2$,

Table 1
From $MSO(<,+1)$ formula to $MSO(<)$ formula.

$MSO(<,+1)$ formula	$MSO(<)$ formula
$\forall x\ \psi(x)$	$\forall x(\psi(x) \wedge \psi(x+1) \wedge \psi(x+2) \wedge \cdots \wedge \psi(x+N-1))$
$x + k_1 < y + k_2$	$\begin{cases} x < y & \text{if } k_1 = k_2 \\ True & \text{if } k_1 < k_2 \\ False & \text{if } k_2 < k_1 \end{cases}$
$P(x+k)$	$P_k(x)$
$\forall P\psi$	$\forall P_0 \forall P_0 \cdots \forall P_{N-1}\psi$

where $\jmath \in [0, N[$, the backward analysis, denoted by $\Diamond_{\jmath}\theta$ and the forward analysis, denoted by $\Box_{\jmath}\theta$ are derived from $\theta_1 U_{\jmath} \theta_2$.

The satisfaction relation is defined based on the MTL formula θ as follows:

- $(f, \gamma) \Vdash True$.
- $(f, \gamma) \Vdash P$ iff $P \in f(\gamma)$.
- $(f, \gamma) \Vdash \theta_1 \wedge \theta_2$ iff $(f, \gamma) \Vdash \theta_1$ and $(f, \gamma) \Vdash \theta_2$.
- $(f, \gamma) \Vdash \theta_1 \vee \theta_2$ iff $(f, \gamma) \Vdash \theta_1$ or $(f, \gamma) \Vdash \theta_2$.
- $(f, \gamma) \Vdash \neg\theta$ iff $(f, \gamma) \nVdash \theta$.
- $(f, \gamma) \Vdash \Diamond_{\jmath}\theta$ iff $\exists t \in]\gamma, N[$ with $t > \gamma$, $t - \gamma \in \jmath$ and $(f, t) \Vdash \theta$.
- $(f, \gamma) \Vdash \Box_{\jmath}\theta$ iff $\forall t \in]\gamma, N[$ with $t > \gamma$, $t - \gamma \in \jmath$ and $(f, t) \Vdash \theta$.
- $(f, \gamma) \Vdash \theta_1 U_{\jmath} \theta_2$ iff $\exists t \in]\gamma, N[$ with $t > \gamma$, $t - \gamma \in \jmath$, $(f, t) \Vdash \theta_2$ and $\forall t' \in]\gamma, t[$, $(f, t') \Vdash \theta_1$.

MTL has two semantics: continuous and pointwise. In this paper, the first semantic is used. The Linear Temporal Logic with Past Operators, denoted by $LTL + Past$ is an extension of MTL logic. This logic simply augments LTL with the backwards operators $\overleftarrow{\Diamond}$ (sometimes in the past), $\overleftarrow{\Box}$ (always in the past), and S (since).

Some required transformations between metric logics are defined in the following sections.

5.1. Transformation from $MSO(<,+1)$ to $MSO(<)$

Let φ be an $MSO(<,+1)$ formula and $\wp \in MP$ be a predicate, we construct an $MSO(<)$ formula denoted by $\overline{\varphi}$. If the $MSO(<,+1)$ formula φ is satisfied over $[0, N[$ then the $MSO(<)$ formula $\overline{\varphi}$ is satisfied over $[0, 1[$.

Firstly, we associate a collection $P_0, \ldots, P_{N-1}$ of N fresh monadic predicates and $\overline{P} = \{P_i,\ P \in \wp,\ 0 \leq i \leq N-1\}$ where P_i represents $P \in \wp$ over the sub-interval $[i, i+1[$. Indeed, there is an obvious stacking bijection between the set of flows $f : [0, N[\rightarrow 2^{\wp}$ and the set of flows $\overline{f} : [0, 1[\rightarrow 2^{\overline{\wp}}$. For an $MSO(<,+1)$ formula φ with $\wp$ is the set of free monadic predicates, we define an $MSO(<)$ formula $\overline{\varphi}$ such that $f \Vdash \varphi$ iff $\overline{f} \Vdash \overline{\varphi}$. Table 1 shows how to transform an $MSO(<,+1)$ formula to $MSO(<)$.

5.2. Transformation from $MSO(<,+1)$ to LTL+past

In the same way as the transformation of $MSO(<,+1)$ to $MSO(<)$, the interval $[0, N[$ is decomposed into N vertically-stacked unit-length intervals, $sub(\theta)$ represents the set of sub formulas of θ and F_i^{ϱ} represents the atomic proposition of $\tilde{\theta}$ when $\varrho \in sub(\theta)$, $0 \leq i \leq N-1$ and $\tilde{\theta}$ is the LTL+past formula. Then, $\tilde{\theta}$ is hold at $[0, 1[$ for the monadic predicate F_i^{ϱ} when θ is hold at $[0, N[$.

Let $F = \{F_i^{\varrho},\ \varrho \in sub(\theta),\ 0 \leq i \leq N-1\}$ be the set of monadic predicates of $\tilde{\theta}$. Therefore, the flow $f : [0, \alpha[\rightarrow 2^{\wp}$ is extended to $\overline{f} : [0, 1[\rightarrow 2^{\overline{\wp}}$ which is extended to $\tilde{f} : [0, 1[\rightarrow 2^{F}$ such that $(f, t+i) \Vdash \varrho$ iff $(\tilde{f}, t) \Vdash F_i^{\varrho}$ where $\varrho \in sub(\theta)$, $0 \leq i \leq \alpha - 1$ and $t \in [0, 1[$.

In the following, the verification of timed bounded opacity properties will be presented.

5.3. Verification of opacity properties

The verification of timed bounded opacity is similar to the bounded verification of inclusion between two timed bounded languages as shown in Lemma 3.1. Therefore, for a given timed automaton $A = (L, l_0, X, \Sigma, I, T)$ and the set of secret locations S, we construct two timed automata $A_f = (L, l_0, X', \Sigma, I, F, T')$ and $B_f = (L, l_0, X', \Sigma, I, F', T')$ that have the same behavior as A and two sets of final locations $F = S$, $F' = L \setminus S$ with $X' = X \cup \{t\}$ (t is a universal clock) and T' is the set of transitions of A including the guard $t \leq \alpha$ in the transitions ending in a secret location. To verify the timed bounded opacity, we verify the inclusion between timed bounded languages $TL(A_f, \alpha)$ and $TL(B_f, \alpha)$ using timed bounded model checking based on the following steps:

- We consider that the bounded inclusion can be written into MTL formulas. Let $P = \Sigma U$ and $Q = \Sigma V$ be two sets of monadic predicates, θ_A and θ_B are two formulas over P for TA A_f respectively Q for TA B_f. Each $[0, N[$-timed word over Σ accepted by A_f or B_f can be extended to a $[0, N[$-flow over P satisfying A_f (respectively over Q satisfying B_f). Therefore, we obtain the following MTL formula: $\forall \Sigma \forall U(\theta_A(\Sigma, U) \rightarrow \exists V \theta_B(\Sigma, V))$
  equivalent to $\forall \Sigma \forall U \exists V(\neg\theta_A(\Sigma, U) \vee \theta_B(\Sigma, V))$.
- We apply the previous transformation: MTL formula to LTL+past formula and $MSO(<, +)$ to $MSO(<)$. This transformation requires an exponential memory space to store the LTL+past formula. The formula $(\neg\theta_A(\Sigma, U) \vee \theta_B(\Sigma, V))$ is transformed to an LTL+past formula ψ for the set of monadic predicates $R = \overline{\Sigma}\ \overline{U}\ \overline{V} W$ where there is a one-to-one stacking correspondence between the flows satisfying $(\neg\theta_A(\Sigma, U) \vee \theta_B(\Sigma, V))$ and $\exists W \psi(\overline{\Sigma}, \overline{U}, \overline{V}, W)$. Therefore the MTL formula is transformed to LTL+past formula as follows: $\forall \overline{\Sigma}\ \forall \overline{U}\ \exists \overline{V}\ \exists W\ \psi(\overline{\Sigma}, \overline{U}, \overline{V}, W)$.
- After the determination of LTL+past formula, we construct an untimed finite state automaton C where its transitions are labeled by subsets of $\overline{\Sigma}\ \overline{U}\ \overline{V} W$. The construction of untimed automaton C requires a second exponential memory space.
- The last step requires checking whether the automaton C is universal over $\overline{\Sigma}\ \overline{U}$. If the automaton C is universal, then the secret S is timed bounded opaque.

To verify the Δ-duration bounded opacity property, we construct, according to Theorem 4.1, two timed automata with final locations $A_f = (LA, l_0, XA, \Sigma, IA, FA, TA)$ and $B_f = (LB, l_0, XB, \Sigma, IB, FB, TB)$ equivalent to A with subsets of locations S and NS respectively. NS is a subset of non secret locations $NS = \{LastLocs(u) \in L \setminus S, \exists v \in TL_S(A, \alpha)$ and $P_{\Sigma_o}(u) = P_{\Sigma_o}(v)\}$. Therefore, the verification of Δ-duration bounded opacity follows the same steps as the verification of timed bounded opacity relative to $TL(A_f)$ and $TL(B_f)$ timed bounded languages. In this case, for each $[0, N[$-timed word over Σ accepted by A_f or B_f can be extended to a $[0, N[$-flow over P satisfying A_f (respectively Q satisfying B_f) where $N = floor(\alpha + \Delta) + 1$. If we obtain $TL(A_f) \nsubseteq TL(B_f)$, then the system is not Δ-duration bounded opaque, otherwise, we cannot conclude that the system is Δ-duration bounded opaque. In the second case, we define the set of locations $DiscLoc$ that is a subset of FB where $DiscLoc = \{LastLocs(v) \in FB, \forall x \in TL(A_f), \exists u \in TL(B_f)$ where $P_{\Sigma_o}(u) = P_{\Sigma_o}(x)$, $v \in TL(B_f)$, $u \preceq_1 v$ and $P_{\Sigma_o}(v) \notin P_{\Sigma_o}(TL(A_f))\}$. If any location in $DiscLoc$ is reachable then the system is not Δ-duration bounded opaque, otherwise the system is Δ-duration bounded opaque.

In the next section, we will verify the timed bounded opacity properties in a case study.

6. Case study

In this section, we consider Exchange in the Cloud system as a case study. The cloud is a term referring to information technology (IT), and software applications through a network connection, often by accessing data centers using wide area networks (WAN) or Internet connectivity. It is often divided into three categories: private, public, and hybrid, referring to who has access to the services or infrastructure. We consider that the private-cloud services are built by enterprises for use by their employees and partners only. Before requesting any service or product, the customer authenticates. Next, the customer requests the product through a cloud based service. The cloud service can provide the product from our storage location or from another cloud storage platform. In cases where the product is not housed in our storage location, the process of requesting from another cloud is private. The customer does not know that the main cloud is buying this product from another cloud platform.

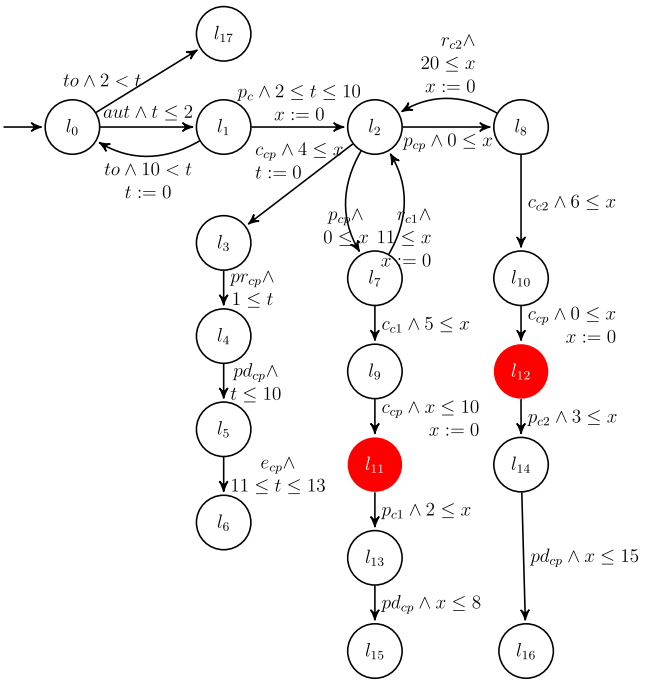


Fig. 10. Timed automaton of Exchange in the Cloud.

For the purpose of opacity analysis, we model a passive attacker (malicious client) that has a partial knowledge of the system and we assume that sending a product through another cloud is a secret information.

The timed automaton $A = (L, l_0, X, \Sigma, I, T)$ as shown in Fig. 10 models the system where:

- $L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_{10}, l_{11}, l_{12}, l_{13}, l_{14}, l_{15}, l_{16}, l_{17}\}$
- l_0 is the initial location
- $X = \{t, x\}$
- $\Sigma = \{aut, p_c, to, c_{cp}, p_{cp}, r_{c1}, r_{c2}, pr_{cp}, c_{c1}, c_{c2}, pd_{cp}, e_{cp}, p_{c1}, p_{c2}\}$

The meaning of all locations and actions in the modeling cloud are given in Tables 2 and 3, respectively. We assume that sending of a product by another cloud is secret information. We consider that $S = \{l_{11}, l_{12}\}$ is the set of secret locations, although, the attacker (malicious client) has a partial view based on unobservable actions $\Sigma_u = \{p_{cp}, r_{c1}, r_{c2}, c_{c1}, c_{c2}, p_{c1}, p_{c2}, pd_{cp}\}$ and a full knowledge of the time execution of the observable actions. In addition, the attacker cannot discover the secret before a duration α.

We performed the implementation of the cloud model on an Intel Core 2.81 GHz with 12 GB RAM using SpaceEx v0.8.406 as a tool of verification [24–27]. We note that the system is opaque (discrete opacity) under the projection map $P_{\Sigma_o}(\pi(u))$ because for all discrete words $aut p_c (p_{cp} r_{c1})^{\infty} p_{cp} c_{c1} c_{cp}$ and $aut p_c (p_{cp} r_{c2})^{\infty} p_{cp} c_{c2} c_{cp}$ ending in l_{11} and l_{12}, respectively, there is a discrete word $aut p_c c_{cp}$ ending in l_3 having the same projection.

Table 2
The meaning of all locations in the modeling cloud.

Actor	Locations		
	Meaning of customer	Meaning of principal cloud	Meaning of cloud 1 (C1)–cloud 2 (C2)
l_0	Authentication interface available	Waiting for purchase order	Waiting for purchase order by principal cloud
l_1	Authenticate and ready to buy a product	Waiting for purchase order	Waiting for purchase order by principal cloud
l_2	Waiting for cloud's response	Request received and consultation of the stock status	Waiting for purchase order by principal cloud
l_3	Confirmation received on the existence of the product	Product in preparation	Waiting for purchase order by principal cloud
l_4	Waiting for the product	Ready product	Waiting for purchase order by principal cloud
l_5	Product received	Request customer's feedback	Waiting for purchase order by principal cloud
l_6	–	Received email	Waiting for purchase order by principal cloud
l_7 (l_8)	Waiting for cloud's response	Waiting for cloud's C1 (C2) response	Purchase order received and consultation of the stock status
l_9 (l_{10})	Waiting for cloud's C1 (C2) response	Confirmation received on the existence of the product on other cloud	Waiting for the request to send the product
l_{11} (l_{12})	Waiting for the product	Waiting for the product	Request received by the cloud C1 (C2) to prepare the product
l_{13} (l_{14})	Waiting for the product	Product received by the cloud C1 (C2)	–
l_{15} (l_{16})	Product received by the cloud C1 (C2)	–	–

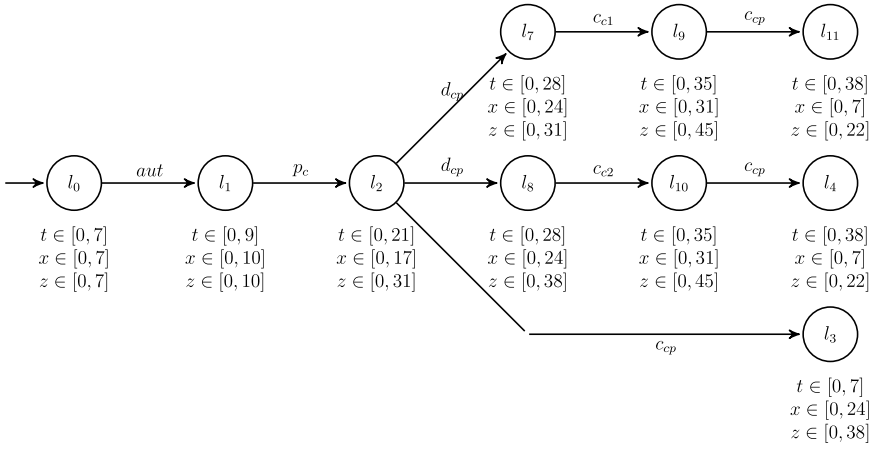


Fig. 11. The reachable values of bounded opacity verification when $\alpha = 15$.

Table 3
The meaning of all actions of modeling cloud.

Actions	Meaning of actions
aut	Authenticate the customer
p_c	The customer sends a purchase order
c_{cp}	The cloud sends the confirmation of the request of product
p_{cp}	The principal cloud sends a purchase order to another cloud
c_{c1}	C1 sends the confirmation of the request of product to principal cloud
c_{c2}	C2 sends the confirmation of the request of product to principal cloud
pr_{cp}	The main cloud prepares the product from the stock
p_{c1}	C1 sends the product to the main cloud
p_{c2}	C2 sends the product to the main cloud
pd_{cp}	The main cloud sends the product to the customer
e_{cp}	The customer sends an email to the main cloud
r_{c1}	C1 rejects the purchase sent
r_{c2}	C2 rejects the purchase sent
to	Time out

To verify the timed bounded opacity in this system, we should integrate a universal clock z and a guard $z \leq \alpha$ into transitions (l_9, g, c_{cp}, r, l_{11}) and (l_{10}, g, c_{cp}, r, l_{12}). we consider $\alpha = 15$, the system is timed bounded opaque. Thus, the customer cannot infer that the product is received by the main cloud or another cloud. Fig. 11 illustrates the reachable values of secret and non-secret locations wherein the local time horizon is equal to 5, the sampling time is 1 and the LGG support function is the algorithm of reachability.

Starting from a system that is timed bounded opaque where $\alpha = 15$, we check the Δ-duration bounded opacity in two cases. We consider $\Delta = 8$, the system is Δ-duration bounded opaque. However, if $\Delta = 11$, the system is not Δ-duration bounded opaque. The timed word $(aut, 2)(p_c, 10)(p_{cp}, 8)(c_{c2}, 10)(c_{cp}, 15)(pd_{c2}, 20)(pd_{cp}, 20)$ that passes through the secret location l_{12} does not have an equivalent timed word that passes through the non secret location l_3 under the projection map P_{Σ_o}.

7. Related works

In network communications and online services, much theoretical work has been performed on the topics of safety and privacy. In the literature, opacity is categorized into two families of formal definitions for Labeled transition systems: Falcone and Marchandin [23] rank the opacity property in three variants: Simple opacity, K-step strong opacity and K-step weak opacity. Opacity has been introduced for dense-time systems, called timed opacity, [11].

Another path was studied in [10], based on the control of observable events. The observation of the system can be classified into four categories: static, dynamic, Orwellian and m-Orwellian. The set of observable events is decided for the static case; but for the dynamic case, the sets of observable and unobservable events change across time.

These approaches suffer from many practical limitations. The authors, in [28], develop a monitor having the same observation abilities

of an intruder for testing the possibility of violating opacity at the time of execution. The opacity is formalized for dynamic security policies when an intruder can observe part of the system's behavior at a given moment [29]. Mullins and Yeddes, in [30,31] analyze the opacity with Orwellian observers and intransitive non-interference. They use a type of projection map referred to as Orwellian projection, where the set of observable events changes at runtime. They verify that the opacity for a regular secret is opaque with Orwellian observers. The authors, in [32], introduce Orwellian partial observability, where unobservable events are not revealed provided that downgrading events never occur for the trace in the future.

In recent years, time has become an important factor for the attack against secure systems. The opacity property was first extended to time settings, [11]. The language-based opacity problem is already undecidable for a very restrictive class of event recording timed automata (ERA). The authors, in [12], define the opacity properties of real-time automata (RTA), as a subclass of timed automata with a single clock without reset, using a transformation function into finite automata (FA) where the language generated by the RTA corresponds exactly to the language accepted by the FA. In this case, the language-opacity and initial-opacity are decidable by reduction to the inclusion problem of regular languages. In [13], the timed opacity problem is undecidable for parametric timed automata (PTA). However, it is decidable for a subclass of PTA, called L/U PTA, see [14] where timed opacity is defined for the execution times. The system is opaque w.r.t. a secret location l_s on the way to final location $l_f \in F$ for execution time D if D is the execution time between l_s and l_f, and between the corresponding non-secret locations to l_s and l_f.

Traditionally, the verification of real-time systems used the Difference Bound Matrices (DBMs) [33] that represent a region by a propositional variable. The aforementioned technique requires less memory space than other techniques used for real-time systems, such as Binary Decision Diagrams (BDDs) [34,35], DDDs [36], Clock Difference Diagrams (CDDs) [37], Rabbit and RED [38] which provides symbolic representations for the search space. Several timed extensions of temporal logic have been proposed for the verification of real-time systems. Propositional Temporal Logic PTL is introduced using operators in propositions qualified in terms of time through Timed Propositional Temporal Logic (TPTL) [39,40]. Temporal logic can be categorized in linear temporal logics, such as LTL and MTL, and branching logics, such as Computation Tree Logic (CTL) and Timed Computation Tree Logic (TCTL). Linear Temporal Logic (LTL) considers every execution as a single timeline [41,42]. Then, it extends propositional logic with temporal operators that allow it to predicate over previous and next states. LTL+past is an extension of LTL augmented with past operators [43]. In addition, Metric Temporal Logic (MTL) [44,45] is an extension of LTL using bounded versions of time operators. In general, the verification properties using the MTL model is undecidable when considering interval-based semantics. For that some restrictions are applied to MTL logic to obtain the decidability verification such as Metric Interval Temporal Logic (MITL) [42], MTL over finite words under pointwise semantics [46], some fragments of MTL as Safety-MTL [47] and CoFlat-MTL [48]. The branching logics consider multiple timeline at once. The model checking problem for TCTL over timed automata was introduced by Alur in [49]. It is an extension of CTL [50,51]. The authors in [52] proposed another algorithm for TCTL based on symbolic techniques and the author in [53] gives an algorithm for TCTL for timed Buchi automata implementation in a KRONOS model checking tool. The aforementioned tool is a full DBM-based model checker that supports forward and backward TCTL. Many tools have been developed for the purpose of model checking of real-time systems such as Hytech [54], UPPAAL [55], KRONOS [56], RED [57] and Rabbit [58]. There exist several methods to verify model checking, including explicit state [59], symbolic [60], SAT/SMT based [61], and bounded model checking (BMC) [62,63].

8. Conclusion

In this study, we described two new properties for timed opacity, called timed bounded opacity and Δ-duration bounded opacity using the static projection. The timed bounded opacity is a privacy property that formulates a system ability to hide all secrets that occur before the upper bound α from any outside observer who has reduced visibility of actions Σ_o. The main idea for checking the timed bounded opacity is the inclusion verification problem. Moreover, the Δ-duration bounded opacity was introduced to state the capability of keeping any secret hidden for a duration Δ. This property is similar to timed bounded opacity. Meanwhile, the Δ-duration bounded opacity is conceded by the secret that occurred before a certain time limit $(\alpha + \Delta)$. This property is similar to K-step opacity for discrete systems. The complexity of these opacity properties was investigated, and the properties were introduced under a simple projection. Therefore a verification of timed bounded opacity is proposed using timed bounded model checking based on the MTL logic metric and $MSO(<, +)$. Additionally, we modeled a real-time system called Exchange in the Cloud System by timed automata to verify the properties of timed bounded opacity that were defined in this work.

Future works will include the definition of timed bounded opacity under the dynamic and orwellian projections, as well as their verification algorithms. In addition, a new verification technique of these properties will be proposed. Furthermore, a new bound will be determined to make a system timed bounded opaque that is initially non opaque relative to another bound.

Declaration of competing interest

I confirm that:

- All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.
- The Article I have submitted to the journal is original, has been written by the stated authors and has not been published elsewhere.
- The Images that I have submitted to the journal for review are original, was taken by the stated authors, and has not been published elsewhere.
- This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.
- The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

Appendix. Algorithm

Algorithm Appendix A.1 Transformation of Timed Automaton

input: timed automaton $A1 = (L, l_0, X', \Sigma, I, T1)$, constant value Δ
output: timed automaton $A2 = (L2, l_0, X', \Sigma, I2, T2)$
initialize $L2 := \{l^1_{0,1}\}$; $T2 := \emptyset$; $stack := \emptyset$;
for all $e = (l_0, a, g, r, l) \in T1$ **do**
  $stack.Push(e)$
end for
while $stack \neq \emptyset$ **do**
  $(l_{i_1}, a_i, g_i, r_i, l_{i_2}) := stack.POP$ /* Note $e_i = (l_{i_1}, a_i, g_i, r_i, l_{i_2}) \in T1$*/
  if $l_{i_1} \notin S \cup Succs(S, A1)$ **then**
    if $l_{i_2} \in Succs(S, A1)$ **then**
      $j := 0$
      for all $l^m_{i_2,j_2} \in L2$ **do**
        $j++$

14: $e^1_{i,j} := (l^1_{i_1,1}, a_i, g_i, r_i, l^m_{i_2,j_2})$ /*$l^1_{i_1,1} \in L2$ homologous to $l_{i_1} \in L^*$*/
15: $T2 := T2 \cup \{e^1_{i,j}\}$
16: **end for**
17: **else**
18: **if** $l_{i_2} \notin L2$ **then**
19: $stack.Push(e_i)$
20: $l^1_{i_2,1} := l_{i_2}$
21: $L2 := L2 \cup \{l^1_{i_2,1}\}$
22: **end if**
23: $T2 := T2 \cup \{e^1_{i,1} = (l^1_{i_1,1}, a_i, g_i, r_i, l^1_{i_2,1})\}$
24: **end if**
25: **else**
26: $j := 0$
27: **for all** $l^{m_1}_{i_1,j_1} \in L2$ and $I(l^{m_1}_{i_1,j_1}) = I(l_{i_1}) \wedge Comb(NC(l_{i_1}, A1), j_1)$ **do**
28: $ac_{i_1} := NC(l_{i_1}, A1) \cup NC(l^{m_1}_{i_1,j_1}, A2)$
29: **for** $(j_2 = 1, j_2 \leq 2^{|ac_{i_1}|}, j_2 ++)$ **do**
30: $j ++$
31: $g^{m_1}_{i,j} := g_i \wedge Comb(ac_{i_1}, j_2)$ /*$l^{m_1}_{i_1,j_1} \in L2$ homologous to l_{i_1}*/
32: **if** $l^{m_2}_{i_2,j_2} \in L2$ and $I(l^{m_2}_{i_2,j_2}) = I(l_{i_2}) \wedge Comb(ac_{i_1}, j_2)$ **then**
33: $e^{m_1}_{i,j} := (l^{m_1}_{i_1,j_1}, a_i, g^{m_1}_{i,j}, r_i, l^{m_2}_{i_2,j_2})$
34: **else**
35: **if** $l^{m_2}_{i_2,j_2} \notin L2$ **then**
36: $m_2 := 1$
37: **else**
38: $m_2 := m_2 + 1$
39: **end if**
40: $e^{m_1}_{i,j} := (l^{m_1}_{i_1,j_1}, a_i, g^{m_1}_{i,j}, r_i, l^{m_2}_{i_2,j_2})$
41: $stack.Push(e_i)$
42: $I(l^{m_2}_{i_2,j_2}) = I(l_{i_2}) \wedge Comb(ac_{i_1}, j_2)$
43: $L2 := L2 \cup \{l^{m_2}_{i_2,j_2}\}$
44: **end if**
45: $T2 := T2 \cup \{e^{m_1}_{i,j}\}$
46: **end for**
47: **end for**
48: **end if**
49: **for all** $e \in T1$ where $e = (l_{i_2}, a, g, r, l)$ **do**
50: $stack.Push(e)$
51: **end for**
52: **end while**
53: return $A2$

References

[1] Dubreil J. Monitoriting and supervisory control for opacity properties, vol. 1. University of Rennes; 2009.
[2] Mazare L. Using unification for opacity properties. In: Proceedings of WITS (Workshop on information technology and systems), vol. 4, 2004. p. 165–76.
[3] Saboori A, Hadjicostis CN. Notions of security and opacity in discrete event systems. In: Conference on decision and control. IEEE; 2007. p. 5056–61.
[4] Yin X, Lafortune S. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In: American control conference, 2015.
[5] Jacob R, Lesage JJ, Faure JM. Overview of discrete event systems opacity: Models, validation, and quantification. Annu Rev Control 2016;41:135–46.
[6] Bourouis A, Klai K, El Touati Y, Ben Hadj-Alouane N. Checking opacity of vulnerable critical systems on-the-fly. Int J Inf Technol Web Eng 2015;10:1–30.
[7] Saboori A, Hadjicostis CN. Verification of initial-state opacity in security applications of DES. Discrete Event Syst 2008;328–33.
[8] Cassez F, Dubreil J, Marchand H. Synthesis of opaque systems with static and dynamic masks. Form Methods Syst Des 2012;88–115.
[9] Wu Y, Lafortune S. Comparative analysis of related notions of opacity in centralized and coordinated architectures. Discrete Event Dyn Syst 2013;307–39.
[10] Bryans J, Koutny M, Ryan P. Modelling opacity using Petri nets. Electron Notes Theor Comput Sci 2005;121:101–15.
[11] Cassez F. The dark side of timed opacity. In: International conference on information security and assurance, vol. 5576, Springer; 2009. p. 21–30.
[12] Wang L, Zhan N, An J. The opacity of real-time automata. IEEE Trans Comput-Aided Des Integr Circuits Syst 2018;37:2845–56.
[13] Andre E. What's decidable about parametric timed automata? Int J Softw Tools Technol Transf 2019;203–19.
[14] Andre E, Sun J. Parametric timed model checking for guaranteeing timed opacity. Comput Sci 2019.
[15] Ouaknine J, Rabinovich A, Worrell J. Time-bounded verification. In: CONCUR 2009 - Concurrency theory, vol. 5710, 2009. p. 496–510.
[16] Baier C, Bertrand N, Bouyer P, Brihaye T. When are timed automata determinizable? In: Automata, languages and programming. Lecture notes in computer science, vol. 5556, Berlin: Springer; 2009. p. 43–54.
[17] Ammar I, El Taouti Y, Mullins J, Yeddes M. Verification of timed opacity applied to an English online auction system. In: International conferance Jeddah, Proceeding of academicsera 57th, Saudi Arabia. 2019.
[18] Alur R, Dill D. A theory of timed automata, In: Theoretical computer science, 2nd ed.. vol. 3, 1994. p. 183–235.
[19] Henzinger TA, Manna Z, Pnueli A. Timed transition systems. Computer science technical reports, 1992, p. 226–51.
[20] Alur R. Timed automata. In: International conference on computer aided verification, vol. 16333, 1994.
[21] Abadi M, Lamport L. An old-fashioned recipe for real time. In: Real-time: Theory in practice, REX workshop. LNCS, vol. 600, Springer-Verlag; 1991. p. 1–27.
[22] Alur R, Henzinger T. Modularity for timed and hybrid systems. In: Eighth conference on concurrency theory. LNCS, vol. 1243, 1997. p. 74–88.
[23] Falcone Y, Marchand H. Various notions of opacity verified and enforced at runtime. INRIA; 2010.
[24] Frehse G, Le Guernic C, Donze A, Cotton S, Ray R, Lebeltel O, et al. SpaceEx: Scalable verification of hybrid systems. In: Qadeer Shaz, Gopalakrishnan Ganesh, editors. 23rd international conference on computer aided verification. LNCS, Springer; 2011.
[25] Cotton S, Frehse G, Lebeltel O. The SpaceEx modeling language. SpaceEx tool, 2010.
[26] Frehse G. An introduction to SpaceEx v0.8. SpaceEx tool, 2010.
[27] Frehse G. A brief experimental comparison of the STC and LGG analysis algorithms in SpaceEx. SpaceEx tool, 2012.
[28] Falcone Y, Marchand H. Runtime enforcement of K-step opacity. In: IEEE conference of decision and control. 2nd ed. 2013. p. 7271–78.
[29] Gruska DP. Dynamics security policies and process opacity for timed process algebras. In: Perspectives of system informatics. 2016.
[30] Mullins J, Yeddes M. Opacity with Orwellian observers and intransitive non-interference. Computing Research Repository (CoRR); 2013.
[31] Mullins J, Yeddes M. Opacity with Orwellian observers and intransitive non-interference. Discrete Event Dyn Syst 2014;12:344–9.
[32] Mullins J. Enforcing opacity with Orwellian observation. In: Discrete event systems WODES. 2016. p. 306–12.
[33] Dill DL. Timing assumptions and verification of finite-state on current systems. In: Automatic verification methods for finite state systems, vol. 407, Springer-Verlag; 1989.
[34] Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput 1986;677–91.
[35] Burch J, Clarke E, Memillan K, Dill D, Hwang LJ. Symbolic model checking: 10^20 states and beyond. Inform and Comput 1990.
[36] Moeller J, Li chtenberg J, Andersen H, Hulgaard H. Model checking of timed systems using difference decision diagrams. In: Electronic notes in theoretical computer science. Elsevier Science; 2001. p. 23.
[37] Larsen KG, Weise C, Yi W, Pearson J. Clock difference diagrams. Technical Report 98/99, Sweden: DoCS, Uppsala University; 1998.
[38] Wang F. Efficient data structure for full symbolic verification of real-time software systems. In: Tools and algorithms for construction and analysis systems. 2000. p. 157–71.
[39] Alur R, Henzinger TA. A really temporal logic. J ACM 1994;41:181–204.
[40] Clarke E, Grumberg O, Long D. Model checking and abstraction. ACM Trans Program Lang Syst 1994;16:1512–42.
[41] Raskin JF. Automata and classical theories for deciding real-time (Ph.D. thesis), Namur, Belgium: University of Namur; 1999.
[42] Alur R, Henzinger T. Real-time logics: Complexity and expressiveness. Technical report Stanford. CA, USA; 1990.
[43] Pnueli A. The temporal logic of programs. In: FOCS'77. IEEE Comp. Soc. Press; 1977. p. 46–57.
[44] Koymans R, Vytopil J, Roever WP. Real-time programming and asynchronous message passing. In: Proceeding of the second annual ACM symposium on principales of distributed computing, New York, USA. 1983. p. 187–97.
[45] Koymans R. Specifying real-time properties with metric temporal logic. In: Real-time systems, vol. 2, 1990. p. 255–99.
[46] Ouaknine J, Worrell J. On the decidability of metric temporal logics. In: LICS. IEEE Computer Society; 2005. p. 188–97.
[47] Ouaknine J, Worrell J. Safety metric temporal logic is fully decidable. In: International conference on tools and algorithms for the construction and analysis of systems. Lecture notes in computer science, LNCS, vol. 3920, 2006. p. 411–25.

[48] Bouyer P, Markey N, Ouaknine J, Worrell J. The cost of punctuality. In: Proceedings of the 22nd annual symposium on logic in computer science. 2007. p. 109–18.
[49] Alur R, Courcoubetis C, Dill D. Model-checking for real-time systems. In: Proceeding of 5th annual symposium on logic in computer science. IEEE Computer Society Press; 1990. p. 414–25.
[50] Queille JP, Sifakis J. Specification and verification of concurrent systems in CESAR. In: SOP'82. LNCS, vol. 137, Springer; 1982. p. 337–51.
[51] Clarke EM, Emerson EA. Design and synthesis of synchronization skeletons using branching-time temporal logic. In: LOP'81. LNCS, vol. 131, Springer; 1982. p. 52–71.
[52] Henzinger TA, Nicollin X, Sifakis J, Yovine S. Symbolic model checking for real-time systems. Inform and Comput 1992;111:394–406.
[53] Tripakis S. The analysis of timed systems in practice (Ph.D. thesis), Grenoble, France: Universite Joseph Fourier; 1998.
[54] Henzinger TA, Ho PH, Wong-toi PH. Hytech: A model checker for hybrid systems. J Softw Tools Technol Transf 1997;1:460–3.
[55] Behrmann G, David A, Larsen KG. A tutorial on uppaal. In: Formal for the design of real-time systems. International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT; 2004, p. 200–36.
[56] Daws C, Olivero A, Tripakis S, Yovine S. The tools KRONOS. In: Hybrid systems III. 1996.
[57] Wang F. Model-checking distributed real-time systems with states, events and multiple fairness assumptions. In: Proceeding of the 10th international conference on algebric methodology and software technology. 2004. p. 553–68.
[58] Beyer D, Lewerentz C, Noack A. Rabbit: A tool for BDD-based verification of real-time systems. In: Proceeding of the 15th international conference on computer aided verification. 2003. p. 122–5.
[59] Hu A. Techniques for efficient formal verification using binary decision diagrams (Ph.D. dissertation), Stanford University; 1995.
[60] Burch JR, Clarke EM, McMillan KL. Symbolic model checking: 1020 states and beyond. Inform and Comput 1992;98:142–70.
[61] Delmas K, Delmas R, Pagetti C. Automatic architecture hardening using safety patterns. In: Computer safety, reliability, and security. Springer; 2015. p. 283–96.
[62] Biere A, Cimatti A, Clarke EM, Zhu Y. Symbolic model checking without BDDs. In: Proc. TACAS'99. 1999. p. 193–207.
[63] Audemard G, Cimatti A, Kornilowicz A, Sebastiani R. Bounded model checking for timed systems. In: Conference: Formal techniques for networked and distributed systems - FORTE 2002, 22nd IFIP WG 6.1 international conference Houston, Texas, USA, Proceedings. 2002.