Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 58-1 (2024) 24–29

Timed Opacity Verification for Switching Output Automata ⋆

T. Liu C. Seatzu A. Giua

DIEE, University of Cagliari, Cagliari 09123, Italy,
(e-mail: t.liu@studenti.unica.it; carla.seatzu@unica.it; giua@unica.it).

Abstract: In this paper, we consider timed opacity verification for switching output automata (SOA). We define a subset of global states of the SOA as vulnerable and associate to each of them a secret dwell interval. A SOA is opaque if an intruder, based on the observation of the outputs over time, cannot determine whether the current global state is vulnerable and the corresponding dwell time belongs to the secret dwell interval. We define a secret-dependent evolution automaton as the logical abstraction of the SOA's evolution when incorporating timed secrets. We show how an observer, i.e., the deterministic finite automaton equivalent to the secret-dependent evolution automaton, can be used to verify timed opacity.

Copyright © 2024 The Authors. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

Keywords: Timed opacity verification, switching output automata, observer.

1. INTRODUCTION

Opacity describes the security and privacy characteristics of a system. Several opacity properties have been thoroughly studied in the context of Discrete Event Systems (DES): see Bryans *et al.* (2005), Saboori *et al.* (2013), Falcone *et al.* (2015) and Tong *et al.* (2017).

Timed opacity is an extended notion of opacity which has been described as the inability of an external observer to accurately infer the system's secret behaviors based on timing information by Cassez (2009). Timed bounded opacity extends the concept of timed opacity by adding a global time upper bound: if an attacker is unable to disclose the secret before a given time upper bound, then the system is timed bounded opaque (Ammar *et al.*, 2021; André *et al.*, 2023). More generally, in a timed setting the secret itself may be a time-varying property, which depends, say, on the current state and on the current value of the timers associated to the system. In this paper we consider such a notion of secret.

Up to now, most research on timed opacity has been based on timed automata (TAs). Cassez (2009) first introduced the timed opacity problem for timed automata. Ammar *et al.* (2021) primarily address the issue of static partial observability within nondeterministic timed automata models, proposing the attribute of bounded timed opacity and proving its complexity. This work enhances the understanding of systems' security and privacy traits, especially in contexts where timing is a critical factor. André *et al.* (2023) investigate the expiring timed opacity problem in timed automata, analyzing a range of time bounds that ensure system opacity and demonstrating when these bounds can be effectively computed for timed automata.

In this paper, we consider a particular model called Switching Output Automaton (SOA) defined by Liu *et al.* (2023). This is an intuitive formalism, well suited to describe man-made systems where the output can take discrete values, or even piecewise-continuous values which can be quantized. Timing information is intricately intertwined with its evolution process and is subject to observation. We focus on whether a SOA is timed opaque.

In our previous work (Liu *et al.* (2023)), we assumed the secret is a subset of discrete states of a SOA. We say an observation is stable if the current output has remained constant for a period greater than or equal to the minimum dwell time. When defining opacity we considered that the intruder has no way of exploiting the information obtained from a nonstable observation because in a very short time it becomes outdated. Therefore we were interested in estimating the set of states consistent with any stable observation. In this work, we consider a more general setting, where the secret set is defined in terms of the current global state – i.e., current discrete state and output – but also on the current dwell time in such a global state. An SOA is not timed opaque if for some evolutions the intruder can detect that the current global state and dwell time belong to the secret.

In this paper, we provide the following contributions.

First, we formally define a new notion of *secret* and of *timed opacity* for an SOA. The secret in this context is time-dependent. A global state of the system is considered secret only when its dwell time belongs to a defined time interval.

Second, we define a *secret-dependent evolution automaton* and provide an algorithm for its computation. This automaton provides a purely logical abstraction of the evolution of the SOA: the dwell time in each global state

⋆ This work was partially supported by the China Scholarship Council, the project Research on Interdisciplinary Issues in Mechatronics Integration (YJSJ24001) funded by the Fundamental Research Funds for the Central Universities and the Innovation Fund of Xidian University, and the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

2405-8963 Copyright © 2024 The Authors. This is an open access article under the CC BY-NC-ND license.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2024.07.005

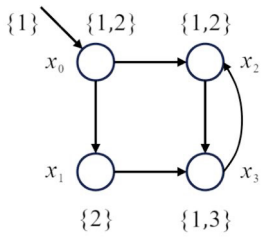

Fig. 1. A switching output automaton.

is quantized and belongs to a finite set of intervals. This abstraction, which depends on the particular secret, is guaranteed to keep all the info that is needed to solve the opacity verification problem.

Third, we construct an observer which provides the set of states consistent with a logical observation of the secret-dependent evolution automaton. Finally, we show how to verify timed opacity using the proposed observer.

The paper is structured as follows. We recall preliminaries in Section 2. The problem is addressed in Section 3. Section 4 discusses the secret-dependent evolution automaton and its equivalent observer. Timed opacity is verified in Section 5. Section 6 concludes the paper.

2. PRELIMINARIES

Definition 1. A *switching output automaton* is defined as $G = (X, Y, B, h, x_0, y_0)$, where

- X is a finite set of states;
- Y is an *output alphabet*;
- $B \subseteq X \times X$ is a set of arcs (or edges) ;
- $h : X \to 2^Y$ is the *output function*;
- $x_0 \in X$ is the initial state;
- $y_0 \in h(x_0)$ is the initial output symbol.

We denote by $h(x) \subseteq Y$ the set of possible output symbols which may be produced when the current state is x. An arc $b = (x, x') \in B$ is considered to be directed from state x to state x'; in such a case x' is said to be a *direct successor* of state x and state x is said to be a direct predecessor of state x'. We denote by $\sigma(x) = \{x' \in X | \exists b = (x, x') \in B\}$ the set of direct successors of x.

An *alphabet* Y is a finite set of symbols and Y^* denotes the set of all (finite length) strings of symbols in Y, while $Y^+ = YY^*$.

The evolution of the state can be described by means of a state run

$$\xrightarrow{t_0} x^{(0)} \xrightarrow{t_1} x^{(1)} \xrightarrow{t_2} \cdots \xrightarrow{t_k} x^{(k)}$$

where for all $i = 0, \ldots, k$ it holds that $x^{(i)} \in X$ and for all $i = 0, \ldots, k-1$ it holds that $(x^{(i)}, x^{(i+1)}) \in B$. Such a run has length $k \geq 0$. It describes the evolution process of an automaton, which initially is in state $x^{(0)}$ at time t_0, and then at time t_i transitions from state $x^{(i-1)}$ to state $x^{(i)}$. While the automaton is in a given state its output may change. This is described by an output run associated to a state $x^{(i)}$:

$$\xrightarrow[\tau_0^{(i)}]{t_i} y_0^{(i)} \xrightarrow[\tau_1^{(i)}]{} y_1^{(i)} \xrightarrow[\tau_2^{(i)}]{} \cdots \xrightarrow[\tau_{h_i}^{(i)}]{} y_{h_i}^{(i)}$$

where $\forall j = 0, \ldots, h_i$, $y_j^{(i)} \in h(x^{(i)})$ and for all $j = 1, \ldots, h_i$, $y_{(j-1)}^{(i)} \neq y_{(j)}^{(i)}$. Such an output run has length $h_i \geq 0$. When state $x^{(i)}$ is reached at time $\tau_0^{(i)} = t_i$ the output takes value $y_0^{(i)}$ and changes from $y_{j-1}^{(i)}$ to $y_j^{(i)}$ at time $\tau_j^{(i)}$.

A state-output run can thus be represented as follow:

$$\begin{aligned}
&\xrightarrow[\tau_0^{(0)}=0]{t_0=0} \begin{pmatrix} x^{(0)} \\ y_0^{(0)} \end{pmatrix} \xrightarrow[\tau_1^{(0)}]{} \begin{pmatrix} x^{(0)} \\ y_1^{(0)} \end{pmatrix} \xrightarrow[\tau_2^{(0)}]{} \cdots \xrightarrow[\tau_{h_i}^{(0)}]{} \begin{pmatrix} x^{(0)} \\ y_{h_i}^{(0)} \end{pmatrix} \xrightarrow[\tau_0^{(1)}]{t_1} \\
&\quad \begin{pmatrix} x^{(1)} \\ y_0^{(1)} \end{pmatrix} \xrightarrow[\tau_1^{(1)}]{} \cdots \xrightarrow[\tau_0^{(i)}]{t_i} \begin{pmatrix} x^{(i)} \\ y_0^{(i)} \end{pmatrix} \xrightarrow[\tau_1^{(i)}]{} \cdots \xrightarrow[\tau_{h_i}^{(i)}]{} \begin{pmatrix} x^{(i)} \\ y_{h_i}^{(i)} \end{pmatrix} \\
&\quad \xrightarrow[\tau_0^{(i+1)}]{t_{i+1}} \begin{pmatrix} x^{(i+1)} \\ y_0^{(i+1)} \end{pmatrix} \xrightarrow[\tau_1^{(i+1)}]{} \cdots \xrightarrow[\tau_{h_i}^{(k)}]{} \begin{pmatrix} x^{(k)} \\ y_{h_i}^{(k)} \end{pmatrix}
\end{aligned}$$

where each state of the run $q = (x, y) \in X \times Y$ is a *global state* of the system: it consists of a pair whose first element is a discrete state $x \in X$ and whose second element is an output value $y \in h(x) \subseteq Y$.

An SOA has multiple state-output runs, which consists of three types of transitions.

Type 1 refers to a state change with no output change. It can be explained as follows. During the time interval $[\tau_{h_i}^{(i)}, t_{i+1})$, the system is in state $x^{(i)}$ and produces output $y_{h_i}^{(i)}$. At time instant t_{i+1}, the system transitions to state $x^{(i+1)}$, while maintaining the same output value, i.e., $y_{h_i}^{(i)} = y_0^{(i+1)}$.

Type 2 indicates a simultaneous state and output change. During the time interval $[\tau_{h_i}^{(i)}, t_{i+1})$, the system is in state $x^{(i)}$ and produces output $y_{h_i}^{(i)}$. At time t_{i+1}, the system transitions to state $x^{(i+1)}$ and generates a new output, which means that $y_{h_i}^{(i)} \neq y_0^{(i+1)}$.

Type 3 indicates an output change with no state change. In a run as described above, the system undergoes h_i changes in the output while the current state remains $x^{(i)}$ and the output changes are represented as $y_0^{(i)} \to y_1^{(i)} \to \cdots \to y_{h_i}^{(i)}$.

We assume that the time distance between the occurrence of any two such transitions must be greater than or equal to the minimum dwell time δ, i.e., $\forall i = 0, \ldots, k$, $\forall j = 0, \ldots, h_i - 1$: $\tau_{j+1}^{(i)} - \tau_j^{(i)} \geq \delta$ and $\forall i = 0, \ldots, k-1$: $\tau_0^{(i+1)} - \tau_{h_i}^{(i)} \geq \delta$. The minimum dwell time has been introduced to avoid Zeno phenomena, namely to avoid an infinite number of switches in the ouput and the discrete state in a time interval of finite length.

The observation of the system is a mapping $T \to Y$, where $T = \mathbb{R}_{\geq 0}$ represents the time that can be divided into a countably infinite number of time intervals as detailed above, in accordance with the state-output run. We define $y(t)$ as the output symbol produced by the system at time t. Function $y(t)$ is piecewise continuous and each value it takes has a duration at least equal to δ. Note that, by definition, the output of two adjacent time intervals is different. We use (y, τ) to denote that the output takes value y for a time interval of duration

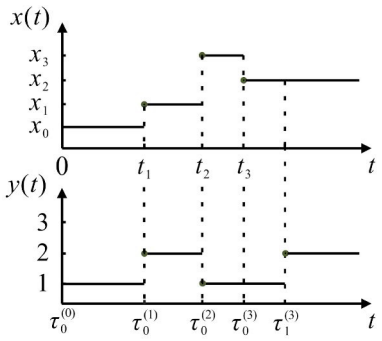

Fig. 2. A possible evolution of the switching output automaton in Fig. 1.

τ where $y \in Y$ and $\tau \in \mathbb{R}_{\geq 0}$. The output behavior of G is defined as $L(G) = \{\omega \in (Y \times \mathbb{R}_{\geq\delta})^+ | \forall i = 1,2,\ldots,n : \omega = (y_{[0,\tau_1)}, \tau_1)(y_{[\tau_1,\tau_2)}, \tau_2 - \tau_1)\cdots(y_{[\tau_{i-1},\tau_i)}, \tau_i - \tau_{i-1})$ and $y_{[\tau_{k-1},\tau_k)} \neq y_{[\tau_k,\tau_{k+1})}, k = 1,\ldots,i-1$ and $\tau_i - \tau_{i-1} \geq \delta\}$.

The state of the system is also a piecewise continuous function $x : T \to X$. The state $x(t)$ cannot be directly measured but an estimation of it can be computed based on the knowledge of the system model and the output $y(t)$. Clearly, since the same observation can in general be produced by different states, $x(t)$ cannot always be uniquely reconstructed.

Example 2. Let us consider the SOA $G = (X, Y, B, h, x_0, y_0)$ in Fig. 1, with states set $X = \{x_0, x_1, x_2, x_3\}$, output alphabet $Y = \{1,2,3\}$, arcs set $B = \{(x_0,x_1), (x_0,x_2), (x_1,x_3), (x_2,x_3), (x_3,x_2)\}$, output function defined by $h(x_0) = \{1,2\}, h(x_1) = \{2\}, h(x_2) = \{1,2\}, h(x_3) = \{1,3\}$, initial state x_0 and initial output symbol $y_0 = 1$. The set of direct successors of x_0, x_1, and x_2 are $\sigma(x_0) = \{x_1, x_2\}$, $\sigma(x_1) = \{x_3\}$, $\sigma(x_2) = \{x_3\}$, and $\sigma(x_3) = \{x_2\}$.

Fig. 2 shows the evolution of the state $x(t)$ and the output $y(t)$ corresponding to the following run:

$$\underset{\tau_0^{(0)}=0}{\overset{t_0=0}{\longrightarrow}} \begin{pmatrix} x_0 \\ 1 \end{pmatrix} \underset{\tau_0^{(1)}}{\overset{t_1}{\longrightarrow}} \begin{pmatrix} x_1 \\ 2 \end{pmatrix} \underset{\tau_0^{(2)}}{\overset{t_2}{\longrightarrow}} \begin{pmatrix} x_3 \\ 1 \end{pmatrix} \underset{\tau_0^{(3)}}{\overset{t_3}{\longrightarrow}} \begin{pmatrix} x_2 \\ 1 \end{pmatrix} \underset{\tau_1^{(3)}}{\longrightarrow} \begin{pmatrix} x_2 \\ 2 \end{pmatrix}.$$

We can find three types of transitions in the above run. At time t_1, the system transitions from state $(x_0, 1)$ to state $(x_1, 2)$, which corresponds to Type 2 transitions. The system moves from state $(x_3, 1)$ to state $(x_2, 1)$ at time t_3, a situation that is consistent with Type 1 transitions. The system moves from state $(x_2, 1)$ to state $(x_2, 2)$ at time $\tau_1^{(3)}$, a situation consistent with Type 3 transitions. ◇

3. PROBLEM STATEMENT

In this work, we investigate timed opacity for SOA modeled as $G = (X, Y, B, h, x_0, y_0)$. A global state of the automaton (as defined in Section 2) is a pair $q = (x, y) \in X \times Y$ such that $y \in h(x) \subseteq Y$. The set of global states is denoted Q. The number of global states is

$$n_Q = |Q| = \sum_{x \in X} |h(x)| \leq |X| \cdot |Y|.$$

We consider a problem of timed opacity where a set of *vulnerable* global states $Q_v = \{q_1, q_2, \ldots, q_s\} \subseteq Q$ is given. A mapping $\mathcal{I}_v$ associates with each state $q_i \in Q_v$ a *secret dwell interval* $\mathcal{I}_v(q_i) = [\delta_i', \delta_i'')$. We assume that the lower and upper bounds of each interval are multiple of the minimum dwell time δ, i.e.,

$$\delta_i' = k_i'\delta,\ \delta_i'' = k_i''\delta,\ k_i' \in \mathbb{N},\ k_i'' \in \mathbb{N}_+ \cup \{+\infty\} \text{ and } k_i' < k_i''.$$

Definition 3. We denote a *secret*

$$S = S(Q_v, \mathcal{I}_v) = \{(q,t) \in Q \times \mathbb{R} \mid q \in Q_v,\ t \in \mathcal{I}_v(q)\}$$

which depends on the set of vulnerable global states Q_v and secret dwell intervals $\mathcal{I}_v(q)$ where $q \in Q_v$. ◇

Definition 4. An SOA is opaque with respect to a timed secret S if the intruder is never able to determine if the pair (q,t) – where $q \in Q$ is the current global state and t is the current dwell time, i.e., the time spent in q – belongs to S. ◇

We plan to construct a new evolution automaton whose states depend on the particular secret, and then build an observer for the evolution automaton to verify timed opacity.

4. SECRET-DEPENDENT EVOLUTION AUTOMATON AND OBSERVER

To better describe the evolution of an SOA G with respect to a timed secret S, we construct a nondeterministic secret-dependent evolution automaton $G_e(S)$. According to Definition 3, we can partition the set of possible dwell times associated with a global state in suitable intervals and, correspondingly, in the secret-dependent evolution automaton we represent a time-driven evolution (no change of discrete state or output) as a sequence of *logical states*, each one associated to one of such intervals. Logical state $(x,y)_j$ denotes a condition in which the global state is (x,y) and its dwell time is in interval I_j. Here δ denotes the minimum dwell time.

- If $q = (x,y) \notin Q_v$, i.e., q is not vulnerable, intervals of interest are:
  $$I_0 = [0, \delta) \quad \text{and} \quad I_1 = [\delta, +\infty)$$
  corresponding to logical sequence
  $$(x,y)_0 \xrightarrow{\ \delta\ } (x,y)_1 \circlearrowleft \delta$$
- If $q = (x,y) \in Q_v$ is vulnerable with secret dwell interval $\mathcal{I}_v(q) = [k'\delta, k''\delta)$, let $k = k''$ if $k'' \in \mathbb{N}$ else $k = \max\{1, k'\}$. In such a case, intervals of interest are: $I_0 = [0,\delta), I_1 = [\delta, 2\delta), \ldots, I_{k-1} = [(k-1)\delta, k\delta)$ and $I_k = [k\delta, +\infty)$ corresponding to logical sequence
  $$(x,y)_0 \overset{\delta}{\to} (x,y)_1 \overset{\delta}{\to} \ldots \overset{\delta}{\to} (x,y)_{k-1} \overset{\delta}{\to} (x,y)_k \circlearrowleft \delta.$$

From logical state $(x,y)_0$ (i.e., until the minimum dwell time elapses) no event may occur, while from all other logical states changes of output and transitions to a different discrete state are possible. For a vulnerable global state (x,y), logical states $(x,y)_j$ with $k' \leq j < k$ (called *secret logical states*) correspond to conditions in which the SOA is in global state $q = (x,y)$ with a dwell time t such that $(q,t) \in S$.

We define a function $\mathcal{R} : Q \to \mathbb{N}_+$ to represent the maximum value of j for each global state $q \in Q$. Clearly, when q is not vulnerable, $\mathcal{R}(q) = 1$; while when q is vulnerable with secret dwell interval $\mathcal{I}_v(q) = [k'\delta, k''\delta)$, $\mathcal{R}(q) = k$ where $k = k''$ if $k'' \in \mathbb{N}$ else $k = \max\{1, k'\}$.

Definition 5. Given an SOA $G = (X, Y, B, h, x_0, y_0)$ and a secret set $S = \{(q_i, t) \in Q \times \mathbb{R} \mid q_i \in Q_v,\ t \in \mathcal{I}_v(q_i)\}$, its *secret-dependent evolution automaton* is a nondeterministic finite automaton $G_e(S) = (\bar{Q}, Y_e, \Delta, \bar{q}_0)$ where

- $\bar{Q} = \{(x, y)_j | q = (x, y) \in Q, j \in \{0, \cdots, \mathcal{R}(q)\}$ is a finite set of logical states;
- $Y_e = Y \cup \{\delta\}$ is the alphabet;
- $\Delta \subseteq \bar{Q} \times Y_e \cup \{\varepsilon\} \times \bar{Q}$ is the transition relation;
- $\bar{q}_0 = (x_0, y_0)_0$ is the initial state. ◇

Algorithm 1 Constructing the secret-dependent evolution automaton

Input: $G = (X, Y, B, h, x_0, y_0)$, a switching output automaton and $S = \{(q_i, t) \in Q \times \mathbb{R} \mid q_i \in Q_v,\ t \in \mathcal{I}_v(q_i)\}$, the secret set

Output: $G_e(S) = (\bar{Q}, Y_e, \Delta, \bar{q}_0)$, the secret-dependent evolution automaton

1: Set $\bar{q}_0 = (x_0, y_0)_0$, $\bar{Q}_{new} = \{\bar{q}_0\}$, $\bar{Q} = \varnothing$
2: Set $Y_e = Y \cup \{\delta\}$
3: Set $\Delta = \varnothing$
4: **while** $\bar{Q}_{new} \neq \varnothing$ **do**
5: select a state $\bar{q} = (x, y)_j \in \bar{Q}_{new}$
6: **if** $j \in \{0, 1, \cdots, \mathcal{R}(\bar{q}) - 1\}$ **then**
7: $\bar{q}' = (x, y)_{j+1}$, $\Delta = \Delta \cup \{(\bar{q}, \delta, \bar{q}')\}$
8: **end if**
9: **if** $\bar{q}' \notin \bar{Q}_{new} \cup \bar{Q}$ **then**
10: $\bar{Q}_{new} = \bar{Q}_{new} \cup \{\bar{q}'\}$
11: **end if**
12: **if** $j = \mathcal{R}(\bar{q})$ **then**
13: $\Delta = \Delta \cup \{(\bar{q}, \delta, \bar{q})\}$
14: **end if**
15: **if** $j > 0$ **then**
16: **for all** $\bar{x} \in \sigma(x)$ **do**
17: **if** $y \in h(\bar{x})$ **then**
18: $\bar{q}' = (\bar{x}, y)_0$, $\Delta = \Delta \cup \{(\bar{q}, \varepsilon, \bar{q}')\}$
19: **if** $\bar{q}' \notin \bar{Q}_{new} \cup \bar{Q}$ **then**
20: $\bar{Q}_{new} = \bar{Q}_{new} \cup \{\bar{q}'\}$
21: **end if**
22: **end if**
23: **for all** $\bar{y} \in h(\bar{x}) \setminus \{y\}$ **do**
24: $\bar{q}' = (\bar{x}, \bar{y})_0$, $\Delta = \Delta \cup \{(\bar{q}, \bar{y}, \bar{q}')\}$
25: **if** $\bar{q}' \notin \bar{Q}_{new} \cup \bar{Q}$ **then**
26: $\bar{Q}_{new} = \bar{Q}_{new} \cup \{\bar{q}'\}$
27: **end if**
28: **end for**
29: **end for**
30: **for all** $\bar{y} \in h(x) \setminus \{y\}$ **do**
31: $\bar{q}' = (x, \bar{y})_0$, $\Delta = \Delta \cup \{(\bar{q}, \bar{y}, \bar{q}')\}$
32: **if** $\bar{q}' \notin \bar{Q}_{new} \cup \bar{Q}$ **then**
33: $\bar{Q}_{new} = \bar{Q}_{new} \cup \{\bar{q}'\}$
34: **end if**
35: **end for**
36: **end if**
37: $\bar{Q}_{new} = \bar{Q}_{new} \setminus \{\bar{q}\}$, $\bar{Q} = \bar{Q} \cup \{\bar{q}\}$
38: **end while**

In the evolution automaton a transition[1] denotes the logical event: a period of time equal to the minimum dwell time has elapsed. Such a transition, from a logical global state $(x, y)_j$ yields the logical global state $(x, y)_{j+1}$ if $j < \mathcal{R}(x, y)$ else it is a self-loop.

[1] In this definition δ is not a real number but a symbol. labeled δ

In addition to a δ-transition, from a logical state $(x, y)_j$ when $j > 0$ three types of transitions may occur: type 1 (state change with no output change), type 2 (simultaneous state and output change), type 3 (output change with no state change).

When in G a transition of type 1 occurs from a discrete state x to a discrete state $\bar{x} \neq x$, the output y does not change and an external observer cannot detect its occurrence. This also means that the system transitions from the global state (x, y) to the global state $(\bar{x}, y)$. The duration of stay in $(\bar{x}, y)$ starts from zero. In the evolution automaton we denote this by a transition labeled with the empty string[2] ε from a logical state $(x, y)_j$ to another logical state $(\bar{x}, y)_0$.

When a transition of type 2 or 3 occurs, the output changes from y to $\bar{y}$ and thus an external agent detects that a transition has occurred. However, due to the possibility of non-empty intersections in the output sets of different states, the output agent may not be able to detect whether a change in the state has also occurred.

A transition of type 2 describes the system changing from the global state (x, y) to the global state $(\bar{x}, \bar{y})$ where $x \neq \bar{x}$ and $y \neq \bar{y}$. The dwell time of $(\bar{x}, \bar{y})$ starts from zero. We denote this with a transition from $(x, y)_j$ to $(\bar{x}, \bar{y})_0$.

A transition of type 3 describes the system changing from the global state (x, y) to the global state $(x, \bar{y})$ where $y \neq \bar{y}$. The dwell time of $(x, \bar{y})$ starts from zero. We denote this with a transition from $(x, y)_j$ to $(x, \bar{y})_0$.

Algorithm 1 describes how a secret-dependent evolution automaton can be constructed. Starting from an initial state, new states are added considering, in the following order, the different situations.

- New states could be created because of the time elapsing. When j reaches its maximum value, no new states will be generated. (line 4 to line 14).
- New states could be created based on type 1 transitions (line 15 to line 22).
- New states could be created based on type 2 transitions (line 23 to line 29).
- New states could be created based on type 3 transitions (line 30 to line 38).

The secret-dependent evolution automaton $G_e(S)$ is a nondeterministic structure because it contains (unobservable) events labeled with the empty string ε or (undistinguishable) events exiting from the same state and sharing the same label. The evolution automaton describes all possible runs of an SOA G with a level of abstraction that depends on the secret set S.

Example 6. Consider the SOA shown in Fig. 1, we consider the set of vulnerable global states $Q_v = \{(x_2, 2)\}$, $\mathcal{I}_v((x_2, 2)) = [2\delta, 4\delta)$, the secret $S = \{((x_2, 2), t) | t \in [2\delta, 4\delta)\}$. For $(x_2, 2)$, the logical sequence is

$$(x_2, 2)_0 \xrightarrow{\delta} (x_2, 2)_1 \xrightarrow{\delta} (x_2, 2)_2 \xrightarrow{\delta} (x_2, 2)_3 \xrightarrow{\delta} (x_2, 2)_4 \circlearrowleft \delta.$$

The set of secret logical states is $\{(x_2, 2)_2, (x_2, 2)_3\}$. The secret-dependent evolution automaton $G_e(S)$ is shown in Fig. 3. It is readily apparent that $G_e(S)$ is nondeterministic: the logical global state $(x_0, 1)_1$ can activate the

[2] Here ε is used to denote the empty string.

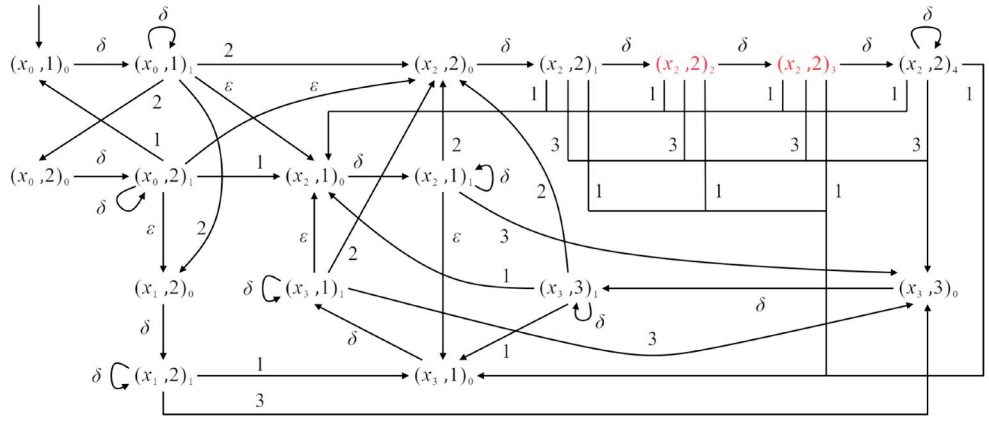

Fig. 3. The secret-dependent evolution automaton of the switching output automaton in Fig. 1.

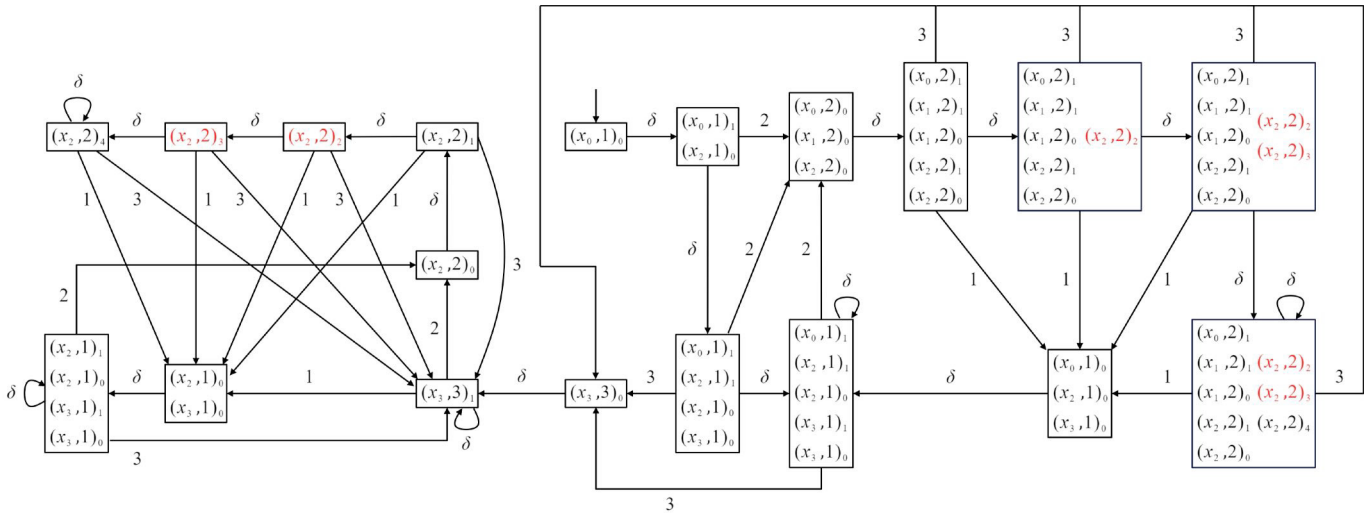

Fig. 4. The observer of the switching output automaton in Fig. 1.

transition labeled 2, leading to three different logical global states: $(x_0,2)_0$, $(x_1,2)_0$, and $(x_2,2)_0$. Additionally, it can activate the transition labeled ε to reach the logical global state $(x_2,1)_0$. ◇

Definition 7. Given an SOA $G = (X, Y, B, h, x_0, y_0)$, we define function $\psi : L(G) \to Y_e^*$ as follows: $\psi((y,t)) = \delta^k$ with $k = \lfloor t/\delta \rfloor$ and $\psi(\omega(y,t)) = \psi(\omega)\ y\delta^k$ with $k = \lfloor t/\delta \rfloor$ where $\lfloor\cdot\rfloor$ denotes the floor function and $\omega \in L(G)$.

This function serves to abstract the output behaviors of an SOA into logical observations of the secret-dependent evolution automaton.

Example 8. Consider the SOA in Fig. 1, we set the minimum dwell time $\delta = 1$ and output behavior $\omega = (1, 2.5)(2, 3)(3, 1.6)$. We compute the corresponding sequence $s = \psi(\omega) = \delta^2 2 \delta^3 3 \delta$.

An equivalent deterministic automaton, that we call *observer*, can be used to estimate the set of logical global states consistent with any observation $s = \psi(\omega) \in Y_e^*$. In the following we denote the *observer* of G as $G_{obs} = (Z, Y_e, \Delta_o, z_0)$ where

- $Z \subseteq 2^{\bar{Q}}$ is a finite state set;
- Y_e is the alphabet;
- $\Delta_o : Z \times Y_e \to Z$ is a partial transition function;
- $z_0 = \{(x_0, y_0)_0\}$ is the initial state. ◇

The function Δ_o can be extended from the domain $Z \times Y_e$ to the domain $Z \times Y_e^*$ in the routine recursive manner: $\Delta_o(z, y_e s) := \bigcup_{z' \in \Delta_o(z,y_e)} \Delta_o(z', s)$ where $y_e \in Y_e$, $s \in Y_e^*$ and $\Delta_o(z, \varepsilon) = z$.

Algorithm 2 Constructing the observer

Input: $G_e(S) = (\bar{Q}, Y_e, \Delta, \bar{q}_0)$, the evolution automaton
Output: $G_{obs} = (Z, Y_e, \Delta_o, z_0)$, the observer

Set $z_0 = D_\varepsilon(\bar{q}_0)$, $Z = \{z_0\}$, assign no tag to z_0
while states with no tag exists in Z **do**
  select a state $z \in Z$ with no tag
  for $y \in Y_e$ **do**
    $\alpha(z, y) = \bigcup_{\bar{q} \in z} D_y(\bar{q})$
    $\beta(z, y) = \bigcup_{\bar{q} \in \alpha(z,y)} D_\epsilon(\bar{q})$
    $\bar{z} = \beta(z, y)$
    if $\bar{z} \notin Z$ **then**
      $Z = Z \cup \{\bar{z}\}$, assign no tag to $\bar{z}$
    end if
    $\Delta_o(z, y) = \bar{z}$
  end for
  tag node z 'old'
end while
Remove all tags

Algorithm 2 describes the construction of the observer and makes use of the following notation.

- For all logical global states $\bar{q} \in \bar{Q}$ of $G_e(S)$ we define $D_\varepsilon(\bar{q}) = \left\{\bar{q}' \in \bar{Q} \mid (\bar{q}, \varepsilon, \bar{q}') \in \Delta^*\right\}$ as the set containing all logical global states reachable from $\bar{q}$ executing zero or more ε-transitions. Note that by definition $\bar{q} \in D^*_\varepsilon(\bar{q})$.
- For all logical global states $\bar{q} \in \bar{Q}$ of $G_e(S)$ and for all symbols $y \in Y_e$ we define $D_y(\bar{q}) = \{\bar{q}' \in \bar{Q} \mid (\bar{q}, y, \bar{q}') \in \Delta\}$ as the set containing all logical global states reachable from $\bar{q}$ executing exactly one observable y-transition.
- For all sets of logical global states $\bar{Q}_{sub} \subseteq \bar{Q}$ and for all symbols $y \in Y_e$, we define $\alpha(\bar{Q}_{sub}, y) = \bigcup_{\bar{q} \in \bar{Q}_{sub}} D_y(\bar{q})$ as the set of logical global states containing the logical global states reachable in $G_e(S)$ from a logical global state $\bar{q} \in \bar{Q}_{sub}$ executing exactly one observable y-transition. We define $\beta(\bar{Q}_{sub}, y) = \bigcup_{\bar{q} \in \alpha(\bar{Q}_{sub}, y)} D_\epsilon(\bar{q})$ as the set containing all logical global states reachable in $G_e(S)$ from a logical global state $\bar{q} \in \alpha(\bar{Q}_{sub}, y)$ executing zero or more ε-transitions.

We have developed a secret-dependent evolution automaton such that for any output behavior ω, we can construct a corresponding sequence s. String s yields observer state $z = \Delta_o(z_0, s)$ which represents the set of states of $G_e(S)$ consistent with logical observation s. In Algorithm 2, for each iteration we compute the set $\bar{z}$ which is the set of states consistent with the sequence $s \in Y_e^*$ starting from z_0. This will be the new state of the observer. The set $\bar{z}$ is calculated in two steps. First to compute $\alpha(z, y)$ where $z \subseteq \bar{Q}$ and $y \in Y_e$; then, compute $\beta(z, y)$, and $\bar{z} = \beta(z, y)$.

Example 9. Consider the SOA described in Fig. 1 whose secret-dependent evolution automaton is shown in Fig. 3. The observer is shown in Fig. 4.

5. VERIFYING CURRENT STATE OPACITY

We define a function $\mathcal{R}' : Q_v \to \mathbb{N}_+$ to represent the minimum secret value of j for each vulnerable global state $q_i \in Q_v$. Clearly, when q is vulnerable with secret dwell interval $\mathcal{I}_v(q) = [k'\delta, k''\delta)$, $\mathcal{R}'(q) = k'$. We define a function $\mathcal{R}'' : Q_v \to \mathbb{N}_+$ to represent the maximum secret value of j for each vulnerable global state $q_i \in Q_v$. Clearly, when q is vulnerable with secret dwell interval $\mathcal{I}_v(q) = [k'\delta, k''\delta)$, $\mathcal{R}''(q) = \mathcal{R}(q) - 1$ if $k'' \in \mathbb{N}$ else $\mathcal{R}''(q) = \mathcal{R}(q)$.

According to the definition of logical states, we define $\{q_{i\mathcal{R}'(q_i)}, \cdots, q_{i\mathcal{R}''(q_i)}\}$ as the set of secret logical states for the vulnerable global state $q_i \in Q_v$. To the secret set S, we associate a set composed of multiple sets of secret logical states for Q_v. Specifically, this is given by

$$S_v = \{\{q_{i\mathcal{R}'(q_i)}, \cdots, q_{i\mathcal{R}''(q_i)}\} | q_i \in Q_v\}.$$

Based on this, we can provide a necessary and sufficient condition to verify timed opacity for an SOA. This result is presented with only a sketch of the proof.

Proposition 10. Consider a SOA G and a secret set S. $G_e(S)$ is its secret-dependent evolution automaton and G_{obs} is its observer. Let Z be the set of states of the observer. The SOA G is timed opaque wrt S iff for all $z \in Z$, it holds that $z \notin S_v$.

Sketch of the Proof 1. ($\Rightarrow$) Assume that G is timed opaque with respect to S. The output behavior ω of the SOA can be abstracted as the sequence $s = \psi(\omega) \in Y_e^*$. The set of states consistent with s concides with a state of the observer, namely with a given $z \in Z$. Therefore, being G timed opaque with respect to S, for all states, it is $z \notin S_v$.

($\Leftarrow$) Assume that $\exists z \in Z$ such that $z \in S_v$. This means that there exists a string $s \in Y_e^*$ with which the state z is consistent. This implies that there exists at least one output behavior ω such that $\psi(\omega) = s$. Therefore, there exists at least one output behavior of G such that $(q, t) \in S$. Therefore, the system is not CSO. ◇

If we want to determine whether an SOA is CSO, we need to focus on the states of the observer. We have to verify whether at least one of them is a subset of S_v. If such is the case, the SOA is not opaque with respect to S.

Example 11. Consider the system in Fig.1 and its observer in Fig.4. Only $(x_2, 2)$ is a vulnerable state, hence $\mathcal{R}'((x_2, 2)) = 2$ and $\mathcal{R}''((x_2, 2)) = 3$. It is $S_v = \{(x_2, 2)_2, (x_2, 2)_3\}$. If there exists a state within the observer that is a subset of S_v, then the SOA is not timed opaque. There exists two states of the observer, $\{(x_2, 2)_2\}$ and $\{(x_2, 2)_3\}$, both of which are subsets of S_v. Thus, the system is not timed opaque. ◇

6. CONCLUSIONS AND FUTURE WORK

We have introduced a novel concept of secret and timed opacity based on SOA, and constructed a secret-dependent evolution automaton grounded in this new definition of secrecy. Furthermore, the timed opacity of the SOA has been verified through an observer. As a future work we plan to investigate fault diagnosis for SOA and study applications in the area of hydraulic networks and power systems.

REFERENCES

A. Saboori and C. N. Hadjicostis. Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, vol. 246, pp. 115-132, 2013.

E. André, E. Lefaucheux and D. Marinho. Expiring opacity problems in parametric timed automata. *Proc. of ICECCS 2023*, pp. 89-98, Toulouse, France, 2023.

F. Cassez. The dark side of timed opacity. *In: International Conference on Information Security and Assurance*, pp. 21–30, 2009.

I. Ammar, Y. E. Touati, M. Yeddes, J. Mullins. Bounded opacity for timed systems. *Journal of Information Security and Applications*, vol 61, pp. 2214-2126, 2021.

J. W. Bryans, M. Koutny, and P. Y. Ryan. Opacity using Petri nets. *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 101-115, 2005.

T. Liu, C. Seatzu and A. Giua. Verification of Current State Opacity using Switching Output Automata. *CoDIT 2023*, pp. 2665-2670, Rome, Italy, 2023.

Y. Falcone and H. Marchand. Enforcement and validation (at runtime) of various notions of opacity. *Discrete Event Dynamic Systems*, vol. 25, no. 4, pp. 531-570, 2015.

Y. Tong, Z. Li, C. Seatzu, and A. Giua. Verification of State-Based Opacity Using Petri Nets. *IEEE Trans. Automat. Contr.*, vol. 62, no. 6, pp. 2823-2837, Jun. 2017.