Concealability Analysis for Current-state Opacity Enforcement via Editing Functions

Kun Peng, Yufeng Chen,Carla Seatzu, Zhiwu Li, and Alessandro Giua

October 2024

Abstract

This paper focuses on the problem of enforcing current-state opacity of a discrete event system via editing functions. In more detail, the observation exposed to an intruder is modified, either erasing or inserting some observations, so as to guarantee that the intruder is not able to discover the predefined secret. The notion of concealability, which formalizes the possibility of maintaining the secret hidden, is introduced starting from defining some illegal states on a particular structure called joint observer. An algorithm for the analysis of concealability is proposed. Finally, an online procedure to make the system opaque is proposed by selecting an editing function.

Published as:
K. Peng, Y. Chen, C. Seatzu, Z. Li, and A. Giua, "Concealability Analysis for Current-State Opacity Enforcement via Editing Functions," *IEEE Transactions on Automatic Control*, vol. 69, no. 10, pp. 7231–7238, October, 2024.
DOI: 10.1109/TAC.2024.3400848

This work was supported in part the Fundamental Research Funds for the Central Universities under Grant JB210402, and the Science and Technology Development Fund, MSAR, under Grant 0029/2023/RIA1, in part by PRIN project MOTOWN: Motown: Smart Production Planning and Control for Manufacturing of Electric Vehicle Powertrain in industry 4.0 Environment, Call 2022 Prot. 20228XEHRE, in part by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.(*Corresponding authors: Yufeng Chen.*)

Kun Peng and Yufeng Chen are with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (e-mails: wc_water2013@163.com; chyf01@163.com).

Zhiwu Li is with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau SAR, China (e-mail: zwli@must.edu.mo).

Carla Seatzu and Alessandro Giua are with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy (e-mail: carla.seatzu@unica.it; giua@unica.it).

1 Introduction

A rich variety of problems related to the security of discrete event systems (DESs) can be formulated in terms of opacity. A system is said to be opaque with respect to some secret information if an unauthorized observer, called *intruder*, can never infer the secret information by observing the system.

In a DES, secret information is normally described by a set of states or by a language. Accordingly, opacity problems can be classfied into language-based opacity [1] and state-based opacity, where the two notions are shown to be equivalent [22, 23]. The type of state opacity includes initial-state opacity [2], current-state opacity [3], K-delayed-state opacity [4], and infinite-state opacity [5]. In this paper, we consider current-state opacity for deterministic finite state automata (DFA).

For a DFA that is not current-state opaque, two main strategies have been proposed in the literature to enforce opacity. The first one, reported in [7, 8, 9, 21], uses supervisory control to disable those evolutions that disclose the secret. The second one, explores approaches based on editing output observations to mislead the intruder.

In [10, 11, 12], the output observations can only be edited by inserting fake events, while in [13, 14] and in our work, observations can be modified through the insertion or erasure of events. A detailed comparison between the developed results in this research and those in [13, 14] will be presented in this section.

Other related works touch upon opacity enforcement through editing functions with different focuses. In [15] abstraction approaches are used to derive opacity equivalent reduced-complexity models. In [16] a modular opacity enforcement which extends the results in [14] is proposed. The selection of dynamic masks for opacity enforcing is addressed in [17, 18].

In our approach for opacity enforcement, we assume that an *operator* is capable of editing the observation visible to the intruder. In particular, only the events in a given set E_{ins} may be inserted in the intruder observation, and only those events in a given set E_{era} may be erased.

In this work, a joint observer, which has been proposed to solve problems of state estimation under attack [19, 20], is adopted to enumerate all admissible and feasible editing behavior. The parallel composition of two appropriately modified observers, namely the operator observer and the intruder observer, is called a *joint observer*.

Concealability formalizes the ability of the operator to enforce opacity by editing output sequences. To analyze concealability, we trim the joint observer recursively by removing illegal states. It is proved that a secret set is concealable if and only if the corresponding trimmed joint observer is non-empty. Finally, we propose an online procedure to compute an editing function to enforce opacity.

This research has a close connection with [13, 14], yet there are several significant differences in observation editing. Unlike the studies in [13, 14], where the editing between two consecutive events depends on the past string and the upcoming event, the insertion decision in our method depends only on the past string. Furthermore, while the work in [13, 14] assumes that the intruder and operator have the same set of observable events, this paper considers a more general case where the intruder's observable event may be a subset of the operator's. We also assume that erasures and insertions are limited to specific observable events.

The proposed approach offers a computational advantage. Consider a plant with n states and m observable events. In [13, 14], the *All Edit Structure under Constraints* is used. It has up to $4^n m$ states and its construction has time complexity of $\mathcal{O}(4^{3n})$, while we use the *Trimmed Joint Observer* which contains up to 4^n states with time complexity of $\mathcal{O}(4^{2n})$.

2 Preliminaries

An alphabet E is a set of symbols. The set of all finite strings over E is denoted as E^* (including the empty string $\varepsilon \in E^*$). The concatenation of two strings $\sigma_1 = e_{1,1}e_{1,2}\cdots e_{1,n}$ and $\sigma_2 = e_{2,1}e_{2,2}\cdots e_{2,m}$ is $\sigma_1\sigma_2 = e_{1,1}e_{1,2}\cdots e_{1,n}e_{2,1}e_{2,2}\cdots e_{2,m}$. The definition of concatenation can be extended to two sets of strings $\mathcal{L}_1, \mathcal{L}_2 \subseteq E^*$ as $\mathcal{L}_1\mathcal{L}_2 = \{\sigma_1\sigma_2 | \sigma_1 \in \mathcal{L}_1 \wedge \sigma_2 \in \mathcal{L}_2\}$. A string σ is called a *prefix* of a string σ' if there exists a string σ'' such that $\sigma\sigma'' = \sigma'$. The prefix set of a string $\sigma \in E^*$ is denoted as $\bar{\sigma} = \{\sigma_1 \in E^* | (\exists \sigma_2 \in E^*)\sigma_1\sigma_2 = \sigma\}$. For a subset of an alphabet $E' \subseteq E$, a string $\sigma \in E^*$ and $e \in E$, the projection of σ under E' is defined as $P_{E'}(\varepsilon) = \varepsilon$, $P_{E'}(\sigma e) = P_{E'}(\sigma)e$ if $e \in E'$ and $P_{E'}(\sigma e) = P_{E'}(\sigma)$, otherwise.

A deterministic finite automaton (DFA) is a four-tuple $G = (Q, E, \delta, q_0)$, where Q is the set of states, E is the set of events, $\delta : Q \times E \to Q$ is the partial transition function such that $\delta(q, e) = q'$ means that the occurrence of e leads G from state q to q', and q_0 is the initial state. Function δ can be extended to $\delta(q, \varepsilon) = q$, $\delta(q, \sigma e) = \delta(\delta(q, \sigma), e)$, where $\sigma \in E^*$ and $e \in E$. The language generated by G is defined by $\mathcal{L}(G) = \{\sigma \in E^* | \delta(q_0, \sigma)!\}$, where ! means "is defined".

In order to make the presentation more intuitive, we may also denote $\delta \subseteq Q \times E \times Q$ as $(q, e, q') \in \delta$ if $\delta(q, e) = q'$.

In a partially observed DFA, $E_o \subseteq E$ denotes the set of events whose occurrences are observable by an external agent, while $E_{uo} = E \backslash E_o$ denotes the set of unobservable events. The execution of a string $\sigma \in \mathcal{L}(G) \subseteq E^*$ produces the observation $P_{E_o}(\sigma)$.

The *unobservable reach* of state $q \in Q$ is defined as $UR_{E_o}(q) = \{q' \in Q | (\exists \sigma \in E_{uo}^*)\delta(q, \sigma) = q'\}$. The unobservable reach of a set of states $Q' \subseteq Q$ is denoted by $UR_{E_o}(Q') = \bigcup_{q \in Q'} UR_{E_o}(q)$.

The observer [6] of a DFA $G = (Q, E, \delta, q_0)$ with a set of observable events E_o, is a DFA $\mathcal{O}(G, E_o) = (C, E_o, \delta_{obs}, c_0)$ with a set of states $C \subseteq 2^Q$, transition function $\delta_{obs} : C \times E_o^* \to C$ such that $\delta_{obs}(c, \omega) = c'$ with $c' = \{q' \in Q | (\exists q \in c)(\exists \sigma \in E^*)\delta(q, \sigma) = q' \wedge P_{E_o}(\sigma) = \omega\}$, and initial state $c_0 = UR_{E_o}(q_0)$ coinciding with the unobservable reach of the initial state. Note that $\mathcal{L}(\mathcal{O}(G, E_o)) = P_{E_o}(\mathcal{L}(G))$, i.e., the observer generates the observed language of G. Furthermore, given an observation $\omega \in P_{E_o}(\mathcal{L}(G))$, the possible states at which the system could be, called *consistent states*, is $\mathcal{S}(\omega) = \delta_{obs}(c_0, \omega)$.

For two automata $G_1 = (Q_1, E_1, \delta_1, q_{01})$ and $G_2 = (Q_2, E_2, \delta_2, q_{02})$, the parallel composition of G_1 and G_2 is $G_{12} = G_1 || G_2 = (Q_1 \times Q_2, E_1 \cup E_2, \delta_{12}, (q_{01}, q_{02}))$, where a state of G_{12} is a pair $(q_1, q_2) \in Q_1 \times Q_2$ with $q_1 \in Q_1$ and $q_2 \in Q_2$. The transition function δ_{12} satisfies

$$\delta_{12}((q_1, q_2), e) = \begin{cases} (\delta_1(q_1, e), \delta_2(q_2, e)) & \text{if } e \in E_1 \cap E_2 \\ (\delta_1(q_1, e), q_2) & \text{if } e \in E_1 \backslash E_2 \\ (q_1, \delta_2(q_2, e)) & \text{if } e \in E_2 \backslash E_1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

3 Problem Statement

We consider a plant represented by a DFA with alphabet E. Let $E_{int} \subseteq E$ be the set of events observed by an intruder. For the current-state opacity, the secret information is represented by a subset of states, namely a secret set $Q_S \subseteq Q$. We define the non-secret set as $Q_{NS} = Q \backslash Q_S$.

Definition 1 (Current-state opacity) Let $G = (Q, E, \delta, q_0)$ be a DFA, $E_{int} \subseteq E$ be the set of events that can be observed by an intruder, and $Q_S \subseteq Q$ be a secret set. G is said to be current-state opaque with respect to Q_S if for all sequences $\sigma \in \mathcal{L}(G)$ with $\delta(q_0, \sigma) = q \in Q_S$, there exists another sequence $\sigma' \in \mathcal{L}(G)$ such that $\delta(q_0, \sigma') = q' \notin Q_S$ and $P_{E_{int}}(\sigma) = P_{E_{int}}(\sigma')$.

In plain words, a DFA G is current-state opaque with respect to a secret set Q_S if for all sequences that lead to a secret state, there exists an observation equivalent sequence that leads to a non-secret state. In such a case, an intruder that observes the evolution of the system can never establish if G is in a secret state. It has been remarked in [3] that G is current-state opaque with respect to Q_S if and only if the observer $\mathcal{O}(G, E_{int}) = (C_{int}, E_{int}, \delta_{int}, c_{0,int})$ verifies the predicate $(\forall c \in C_{int}) c \not\subseteq Q_S$, since this means that for all observations $\omega \in P_{E_{int}}(\mathcal{L}(G))$, the set of consistent states satisfies $\mathcal{S}(\omega) \not\subseteq Q_S$. For a plant G that violates the definition of current-state opacity, we define the language that reveals the secret as $\mathcal{L}_l(G) = \{\omega \in P_{E_{int}}(\mathcal{L}(G)) | (\forall \sigma \in \mathcal{L}(G)) P_{E_{int}}(\sigma) = \omega \Rightarrow \delta(q_0, \sigma) \in Q_S\}$.

3.1 Problem setting and editing function

Given a plant that is not current-state opaque, we assume that an operator can edit the observation it generates, thus misleading the intruder so as to enforce opacity. This setting is summarized in Fig. 1, where $E_{ope} \subseteq E$ is a set of observable events by the operator, $E_{era} \subseteq E_{int}$ (resp. $E_{ins} \subseteq E_{int}$) is a set of events that can be erased (resp. inserted), and $E_e = E_{ope} \cup E_+ \cup E_-$ be the editing alphabet with $E_+ = \{e_+ | e \in E_{ins}\}$ and $E_- = \{e_- | e \in E_{era}\}$. Typically, we assume that $E_{int} \subseteq E_{ope}$.

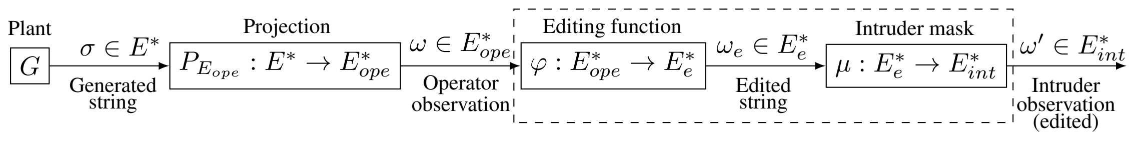


Figure 1: Description of the proposed setting for opacity enforcement via editing function.

Note that the presented approach can also be applied to non-deterministic finite automata since the construction of the current-state observer for a DFA and an NFA follows the same procedure.

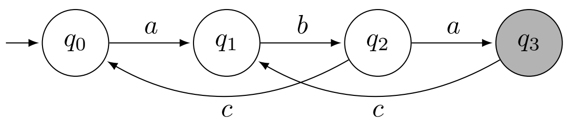


Figure 2: A DFA model of a simple manufacturing system.

Given a string $\omega \in P_{E_{ope}}(\mathcal{L}(G)) \subseteq E_{ope}^*$ produced by the plant, the operator can edit it to obtain a string $\omega_e \in E_e^*$ by inserting in ω an arbitrary number of events in E_+ and by arbitrarily changing any event $e \in E_{era}$ with the corresponding event $e_- \in E_-$.

Definition 2 (Editing function) An editing function is a mapping $\varphi : E_{ope}^* \to E_e^*$ such that for $\omega \in E_{ope}^*$ and $e \in E_{ope}$:

$$\varphi(\varepsilon) \in E_+^*$$

$$\varphi(\omega e) \in \begin{cases} \varphi(\omega) e E_+^* & \text{if } e \notin E_{era}; \\ \varphi(\omega)\{e, e_-\}E_+^* & \text{if } e \in E_{era}. \end{cases}$$

Note that an edited string is differently perceived by the operator and the intruder. Based on the fact that the operator knows that string $\omega_e = \varphi(\omega) \in E_e^*$ corresponds to observation $\omega = \varphi^{-1}(\omega_e)$, we define the operator mask as follows.

Definition 3 (Operator mask) An operator mask $\mu_{ope} : E_e^* \to E_{ope}^*$ is recursively defined as follows: $\mu_{ope}(\varepsilon) = \varepsilon$ and for $\omega_e \in E_e^*$, $e' \in E_e$,

$$\mu_{ope}(\omega_e e') = \begin{cases} \mu_{ope}(\omega_e) e, & \text{if } e' = e \in E_{ope} \vee e' = e_- \in E_-; \\ \mu_{ope}(\omega_e), & \text{if } e' \in E_+. \end{cases}$$

The intruder perceives a string $\omega_e \in E_e^*$ as if all the events in E_+ correspond to those generated by the plant while it ignores the events in $(E_{ope} \backslash E_{int}) \cup E_-$. Hence, ω_e corresponds to an edited string through the following mask.

Definition 4 (Intruder mask) An intruder mask $\mu_{int} : E_e^* \to E_{int}^*$ is recursively defined as follows: $\mu_{int}(\varepsilon) = \varepsilon$ and for $\omega_e \in E_e^*$, $e' \in E_e$,

$$\mu_{int}(\omega_e e') = \begin{cases} \mu_{int}(\omega_e) e, & \text{if } e' = e \in E_{int} \vee e' = e_+ \in E_+; \\ \mu_{int}(\omega_e), & \text{if } e' \in E_- \cup (E_{ope} \backslash E_{int}). \end{cases}$$

Note that the edited intruder observation may be $\omega' = \mu_{int}(\omega_e) \notin P_{E_{int}}(\mathcal{L}(G))$, i.e., it corresponds to none of the possible observations of the intruder. In such a case the editing function is not stealthy, i.e., the intruder is aware that the observation has been edited.

Example 1 A manufacturing system can be modeled as the DFA shown in Fig. 2, where $Q_S = \{q_3\}$ (in grey), $E_{ope} = \{a, b, c\}$, and $E_{int} = \{a, c\}$. It consists of a buffer with two slots. In state q_0 the buffer is empty, in state q_i (for $i = 1, 2$) only slot i is occupied, and in state q_3 the buffer is full. Items enter the system from slot 1 (event a) and leave the system from slot 2 (event c). An item in slot 1 can be moved to slot 2 (event b).

As an example, if sequence $\sigma = abacbc$ occurs, the observations of the operator and the intruder are $P_{E_{ope}}(\sigma) = abacbc$ and $P_{E_{int}}(\sigma) = aacc$, respectively.

Now, assume that $E_{ins} = E_{era} = \{c\}$ and $\varphi(abacbc) = abc_+acbc_-$. This means that the operator edits the observation by inserting a fake c after the occurrence of ab and erasing the observation produced by the second occurrence of c. The observation of the intruder will be $\mu_{int}(abc_+acbc_-) = acac$.

3.2 Joint observer

To better clarify the set of consistent states computed by an operator and an intruder for a given edited word ω_e, we use two observers on the editing alphabet E_e, i.e., the edited operator observer and the edited intruder observer.

Definition 5 (Edited operator observer) Let $\mathcal{O}(G, E_{ope}) = (C_{ope}, E_{ope}, \hat{\delta}_{ope}, c_{0,ope})$ be the observer of the operator. The edited operator observer is $\Omega_{ope}(G) = (C_{ope}, E_e, \delta_{ope}, c_{0,ope})$ where the transition function $\delta_{ope} : C_{ope} \times E_e^* \to C_{ope}$ satisfies: for $c \in C_{ope}$ and $e' \in E_e$,

$$\delta_{ope}(c, e') = \begin{cases} \hat{\delta}_{ope}(c, e), & \text{if } (e' = e \in E_{ope} \vee e' = e_- \in E_-) \\ & \quad \wedge \hat{\delta}_{ope}(c, e)!; \\ c, & \text{if } e' = e_+ \in E_+; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

The edited operator observer can be computed by Algorithm 1. For the current-state observer of the operator, line 2 adds self-loops labelled by all inserted events e_+ at each state, and line 4 adds an event e_- in parallel with each e.

Algorithm 1: Computation of the edited operator observer

```
Input: A DFA $G = (Q, E, \delta, q_0)$ and the current-state observer of an operator
       $\mathcal{O}(G, E_{ope}) = (C_{ope}, E_{ope}, \hat{\delta}_{ope}, c_{0,ope})$.
Output: The edited operator observer $\Omega_{ope}(G) = (C_{ope}, E_e, \delta_{ope}, c_{0,ope})$.
1 Initialization: $\delta_{ope} \leftarrow \hat{\delta}_{ope}$.
2 $\delta_{ope} \leftarrow \delta_{ope} \cup \{(c, e_+, c) | c \in C_{ope} \wedge e_+ \in E_+\}$.
3 foreach $(c, e, c') \in \hat{\delta}_{ope}$ with $e \in E_{era}$ do
4     $\delta_{ope} \leftarrow \delta_{ope} \cup \{(c, e_-, c')\}$.
5 Output $\Omega_{ope}(G) = (C_{ope}, E_e, \delta_{ope}, c_{0,ope})$.
```

Proposition 1 Given an edited operator observer $\Omega_{ope}(G) = (C_{ope}, E_e, \delta_{ope}, c_{0,ope})$,

(a) there exists an observation $\omega \in P_{E_{ope}}(\mathcal{L}(G))$ generated by the plant and an editing function φ such that $\varphi(\omega) = \omega_e$ if and only if $\omega_e \in \mathcal{L}(\Omega_{ope}(G))$;

(b) for all strings $\omega_e \in \mathcal{L}(\Omega_{ope}(G))$, the current-state estimation of the operator with the observation $\omega = \mu_{ope}(\omega_e)$ is $\mathcal{S}_{ope}(\omega) = \delta_{ope}(c_{0,ope}, \omega_e)$.

Proposition 1, as well as Propositions 2 and 3 in the following, are proved in the Appendix.

In simple words, the language of the edited operator observer consists of all words that can be obtained by editing a system observation ω. In this observer, $\varphi(\omega)$ yields the sequence consistent with ω.

Definition 6 (Edited intruder observer) Let $\mathcal{O}(G, E_{int}) = (C_{int}, E_{int}, \hat{\delta}_{int}, c_{0,int})$ be the observer of an intruder. The edited intruder observer is $\Omega_{int}(G) = (C_{int} \cup \{c_d\}, E_e, \delta_{int}, c_{0,int})$, where the transition function is $\delta_{int} : C_{int} \times E_e^* \to C_{int} \cup \{c_d\}$ such that for $c \in C_{int}$ and $e' \in E_e$:

$$\delta_{int}(c, e') = \begin{cases} \hat{\delta}_{int}(c, e), & \text{if } (e' = e \in E_{int} \vee e' = e_+ \in E_+) \\ & \quad \wedge \hat{\delta}_{int}(c, e)!; \\ c_d, & \text{if } (e' = e \in E_{int} \vee e' = e_+ \in E_+) \\ & \quad \wedge \hat{\delta}_{int}(c, e) \text{ is not defined} \\ c, & \text{if } e' \in E_-; \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

while $\delta_{int}(c_d, e')$ is undefined for all $e' \in E_e$.

An edited intruder observer can be computed by Algorithm 2. For the current-state observer of the intruder, line 2 adds self-loops labelled by all erased events e_-, and line 4 adds an event e_+ in parallel with event e. For an event e that is not enabled at a state c, line 7 adds an arc from c to c_d labelled by e. If $e \in E_{ins}$, line 9 adds an arc with the same direction labelled by e_+. As mentioned, the intruder mask μ_{int} maps an edited sequence to an observation of the intruder. For an edited sequence $\sigma_e \in E_e^*$, if $\mu_{int}(\sigma_e) \notin P_{E_{int}}(\mathcal{L}(G))$, the intruder realizes that this observation has been corrupted, and ω_e leads the edited intruder observer to c_d.

Algorithm 2: Computation of the edited intruder observer

```
Input: A DFA $G = (Q, E, \delta, q_0)$ and the current-state observer of an intruder
    $\mathcal{O}(G, E_{int}) = (C_{int}, E_{int}, \hat{\delta}_{int}, c_{0,int})$.
Output: The edited intruder observer $\Omega_{int}(G) = (C_{int} \cup \{c_d\}, E_e, \delta_{int}, c_{0,int})$.
1  Initialization: $\delta_{int} \leftarrow \hat{\delta}_{int}$.
2  $\delta_{int} \leftarrow \delta_{int} \cup \{(c, e_-, c) | c \in C_{int} \wedge e_- \in E_-\}$.
3  foreach $(c, e, c') \in \hat{\delta}_{int}$ with $e \in E_{ins}$ do
4      $\delta_{int} \leftarrow \delta_{int} \cup \{(c, e_+, c')\}$.
5  foreach $c \in C_{int}$ and $e \in E_{int}$ do
6      if $\hat{\delta}_{int}(c, e)$ is not defined then
7          $\delta_{int} \leftarrow \delta_{int} \cup \{(c, e, c_d)\}$.
8          if $e \in E_{ins}$ then
9              $\delta_{int} \leftarrow \delta_{int} \cup \{(c, e_+, c_d)\}$.
10 Output $\Omega_{int}(G) = (C_{int}, E_e, \delta_{int}, c_{0,int})$.
```

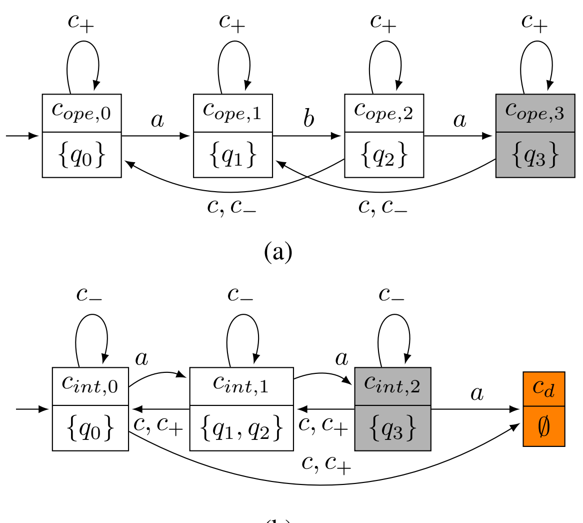


Figure 3: (a) Edited operator observer and (b) edited intruder observer of the DFA shown in Fig. 2.

Proposition 2 Given an edited intruder observer $\Omega_{int}(G) = (C_{int} \cup \{c_d\}, E_e, \delta_{int}, c_{0,int})$,

(a) its generated language $\mathcal{L}(\Omega_{int}(G))$ *is* $\{\omega_e \in E_e^* | \mu_{int}(\omega_e) \in P_{E_{int}}(\mathcal{L}(G))\}(\{\varepsilon\} \cup E_e)$ *;*

(b) for all strings $\omega_e \in \mathcal{L}(\Omega_{int}(G))$ and $\omega = \mu_{int}(\omega_e)$*:*

- *i)* if $\omega \in P_{E_{int}}(\mathcal{L}(G))$, the current-state estimation of the intruder with the observation $\omega = \mu_{int}(\omega_e)$ is $\mathcal{S}_{int}(\omega) = \delta_{int}(c_{0,int}, \omega_e)$*;*
- *ii) if* $\omega \notin P_{E_{int}}(\mathcal{L}(G))$, then $\delta_{int}(c_{0,int}, \omega_e) = c_d$.

In simple words, the edited intruder observer provides the state estimation of the intruder for all possible edited words.

Example 2 Consider the DFA G in Fig. 2, where $Q_S = \{q_3\}$, $E_{ope} = \{a, b, c\}$, $E_{int} = \{a, c\}$, and $E_{era} = E_{ins} = \{c\}$. The edited operator observer $\Omega_{ope}(G)$ and the edited intruder observer $\Omega_{int}(G)$ are visualized in Figs. 3(a) and (b), respectively. A state of the two observers is represented as a rectangle with two entries. The first indicates the name of the state and the second stands for the set of states of G corresponding to the current state of the observer. States $c_{ope,3}$ and $c_{int,2}$, representing that the observer estimation is contained in the secret set, are highlighted in grey. State c_d of the edited intruder observer is highlighted in orange.

Definition 7 (Joint observer) Let $G = (Q, E, \delta, q_0)$ be a DFA, $\Omega_{ope}(G) = (C_{ope}, E_e, \delta_{ope}, c_{0,ope})$ be its edited operator observer, and $\Omega_{int}(G) = (C_{int} \cup \{c_d\}, E_e, \delta_{int}, c_{0,int})$ be its edited intruder observer. A joint observer of G is defined as $\Omega_{joi}(G) = \Omega_{ope}(G)||\Omega_{int}(G) = (X, E_e, \delta_{joi}, x_0)$, where $X \subseteq C_{ope} \times (C_{int} \cup \{c_d\})$ and $x_0 = (c_{0,ope}, c_{0,int})$.

Proposition 3 Given a joint observer $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$, an observation $\omega \in P_{E_{ope}}(\mathcal{L}(G))$ by the operator, and an editing function φ such that $\varphi(\omega) = \omega_e$ and $\omega' = \mu_{int}(\omega_e)$, it holds:

$$\delta_{joi}(x_0, \omega_e) = \begin{cases} (\mathcal{S}_{ope}(\omega), \mathcal{S}_{int}(\omega')), & \text{if } \omega' \in P_{E_{int}}(\mathcal{L}(G)); \\ (\mathcal{S}_{ope}(\omega), c_d), & \text{otherwise.} \end{cases}$$

4 Concealability analysis and opacity enforcement

In this section, we introduce the notion of concealability of a secret set. Then, given a DFA and a concealable secret set, an online procedure to enforce opacity is proposed.

4.1 Concealability and admissible editing function

Definition 8 (Concealability) Let $G = (Q, E, \delta, q_0)$ be a DFA, $Q_S \subseteq Q$ be a set of secret states, E_{ope} and E_{int} be sets of events observable by an operator and an intruder, respectively. Q_S is said to be concealable if there exists an editing function φ such that for all sequences generated by the original plant $\sigma \in \mathcal{L}(G)$, it holds,

$$\mu_{int}(\varphi(P_{E_{ope}}(\sigma))) \in P_{E_{int}}(\mathcal{L}(G)), \tag{1}$$

$$P_{E_{int}}(\sigma) \in \mathcal{L}_l(G) \Rightarrow \mu_{int}(\varphi(P_{E_{ope}}(\sigma))) \notin \mathcal{L}_l(G). \tag{2}$$

Eq. (1) means that the intruder always observes a sequence that belongs to the language of the original plant. Eq. (2) implies that each sequence that can expose the secret can be edited into a sequence that does not expose the secret. In plain words, a secret set is said to be concealable if the observation can be stealthily edited to prevent the secret from being revealed.

To analyze the concealability, we partition the states of the joint observer. Consider a plant $G = (Q, E, \delta, q_0)$, a secret set $Q_S \subseteq Q$, and the joint observer $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$. The set of states of the joint observer is partitioned as $X = X_{nst} \cup X_{exp} \cup X_{pex} \cup X_{fex} \cup X_{reg}$, where

1. $X_{nst} = \{(c_{ope}, c_{int}) \in X | c_{int} = c_d\}$ is the set of states reached in $\Omega_{joi}(G)$ via a *non-stealthy* edited string ω_e. In such a case the intruder becomes aware that her/his observation has been corrupted.

2. $X_{exp} = \{(c_{ope}, c_{int}) \in X | c_{ope} \subseteq Q_S \wedge c_{int} \subseteq Q_S\}$ is the set of states reached in $\Omega_{joi}(G)$ via an *exposing* edited string ω_e which violates opacity. In such a case the intruder infers the secret.

3. $X_{pex} = \{(c_{ope}, c_{int}) \in X | c_{ope} \cap Q_S \neq \emptyset \wedge c_{int} \subseteq Q_S\}$ is the set of states reached in $\Omega_{joi}(G)$ via a *potentially exposing* edited string ω_e. In such a case the intruder believes that a secret state has been reached while the operator is uncertain: in his opinion the state could either be a secret state or a non-secret state.

4. $X_{fex} = \{(c_{ope}, c_{int}) \in X | c_{ope} \cap Q_S = \emptyset \wedge c_{int} \subseteq Q_S\}$ is the set of states reached in $\Omega_{joi}(G)$ via a *falsely exposing* string ω_e. In such a case the intruder is certain that a secret state has been reached while the operator is aware that the system is in a non-secret state.

5. $X_{reg} = X \backslash (X_{nst} \cup X_{exp} \cup X_{pex} \cup X_{fex})$ is the set of all other states called regular states.

The objective of the operator is to enforce opacity while remaining stealthy. In such a case, the edited string ω_e should never be exposing or non-stealthy, i.e., a good editing function ensures that the states in X_{exp} and X_{nst} are never reached. We point out, however, that also potentially exposable states are dangerous since the belief of the intruder that the plant is in a secret state—albeit based on false information, namely, the edited observation—may in reality be correct. Based on this fact, it is interesting to explore editing functions that will never reach a set of *illegal states* defined as $X_{il} = X_{nst} \cup X_{exp} \cup X_{pex}$. Correspondingly, the set of *legal states* is $X_l = X \backslash X_{il}$. To prevent illegal states being reached, we define the admissibility of an editing function.

Definition 9 (Admissibility) Let $G = (Q, E, \delta, q_0)$ be a DFA, $Q_S \subseteq Q$ be the set of secret states, $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$ be its joint observer, and $X_l \subseteq X$ be a set of legal states (the corresponding set of illegal states is $X_{il} = X \backslash X_l$). An editing function φ is said to be admissible if for any possible observation of the operator $\omega \in P_{E_{ope}}(\mathcal{L}(G))$, the corresponding edited sequence $\varphi(\omega)$ satisfies $\delta_{joi}(x_0, \varphi(\omega)) = x \in X_l$.

In plain words, an editing function is admissible if it does not reveal the secret or the fact that the output has been edited.

Proposition 4 Let $G = (Q, E, \delta, q_0)$ be a DFA, $Q_S \subseteq Q$ be a secret set, E_{ope} and E_{int} be sets of events observable by the operator and the intruder, respectively, and $\Omega_{joi}(G)$ be the corresponding joint observer. Q_S is concealable if and only if there exists an admissible editing function.

Proof: (only if) According to Definition 8, if Q_S is concealable, there exists an editing function φ such that Eqs. (1) and (2) hold. For a sequence generated by the original plant $\sigma \in \mathcal{L}(G)$, the observation of the operator is $P_{E_{ope}}(\sigma)$ and the editing decision is $\varphi(P_{E_{ope}}(\sigma))$. According to Proposition 2, if $\mu_{int}(\varphi(P_{E_{ope}}(\sigma))) \in P_{E_{int}}(\mathcal{L}(G))$, then $\delta_{int}(c_{0,int}, \varphi(P_{E_{ope}}(\sigma))) \neq c_d$, implying $\delta_{joi}(x_0, \varphi(P_{E_{ope}}(\sigma))) \notin X_{nst}$.

If $P_{E_{int}}(\sigma) \in \mathcal{L}_l(G)$, then for all sequences $\sigma' \in \mathcal{L}(G)$, $P_{E_{int}}(\sigma) = P_{E_{int}}(\sigma')$ implies $\delta(q_0, \sigma') = q' \in Q_S$, and further $\delta(q_0, \sigma) = q \in Q_S$. Therefore, the set of states consistent with the observation $P_{E_{ope}}(\sigma)$ according to the operator satisfies $\mathcal{S}_{ope}(P_{E_{ope}}(\sigma)) \cap Q_S \neq \emptyset$ due to $q \in \mathcal{S}_{ope}(P_{E_{ope}}(\sigma))$. By Eq. (2), the editing decision $\varphi(P_{E_{ope}}(G))$ satisfies $\mu_{int}(\varphi(P_{E_{ope}}(\sigma))) \notin \mathcal{L}_l(G)$ and $\delta_{int}(c_{0,int}, \varphi(P_{E_{ope}}(\sigma))) \nsubseteq Q_S$ holds. For each $\sigma \in \mathcal{L}(G)$, the corresponding editing decision $\varphi(P_{E_{ope}}(\sigma))$ with $\delta_{joi}(x_0, \varphi(P_{E_{ope}}(\sigma))) = (\mathcal{S}_{ope}(\varphi(P_{E_{ope}}(\sigma))), \mathcal{S}_{int}(\varphi(P_{E_{ope}}(\sigma))))$ makes the fact that $\mathcal{S}_{ope}(\varphi(P_{E_{ope}}(\sigma))) \cap Q_S \neq \emptyset$ implies $\mathcal{S}_{int}(\varphi(P_{E_{ope}}(\sigma))) \nsubseteq Q_S$, indicating $\delta_{joi}(x_0, \varphi(P_{E_{ope}}(\sigma))) \notin X_{exp} \cup X_{pex}$. In other words, the editing function satisfies the predicate $(\forall \sigma \in \mathcal{L}(G)) \delta_{joi}(x_0, \varphi(P_{E_{ope}}(\sigma))) = x \in X_l$, i.e., φ is admissible.

(if) If there exists an admissible editing function φ, then for all operator observations $\omega \in P_{E_{ope}}(\mathcal{L}(G))$ with $\delta_{joi}(x_0, \varphi(\omega)) = x = (c_{ope}, c_{int}) \in X_l$, both $c_{int} \neq c_d$ and $c_{ope} \cap Q_S \neq \emptyset \Rightarrow c_{int} \nsubseteq Q_S$ hold. According to Propositions 2 and 3, if $c_{int} \neq c_d$, then $\mu_{int}(\varphi(\omega)) \in P_{E_{int}}(\mathcal{L}(G))$ holds, implying that Eq. (1) is satisfied. If $c_{ope} \cap Q_S \neq \emptyset$, there exists a sequence generated by the original plant $\sigma \in \mathcal{L}(G)$ such that $P_{E_{ope}}(\sigma) = \omega$ and $\delta(q_0, \sigma) \in Q_S$. According to Propositions 2 and 3, if $c_{int} \nsubseteq Q_S$, then there exists at least one sequence generated by the original plant $\sigma \in \mathcal{L}(G)$ with $P_{E_{int}}(\sigma) = \mu_{int}(\varphi(\omega))$, $\delta_{int}(q_0, \sigma) \notin Q_S$ holds, which implies $\mu_{int}(\varphi(\omega)) \notin \mathcal{L}_l(G)$. Thus, Eq. (2) is satisfied.

Our objective is to describe all admissible editing functions. In addition, we want to provide an algorithm to choose one of such admissible editing functions when the above set is not empty, which is the case when the secret is concealable. To this aim, we define a *trimmed joint observer*.

Definition 10 (Trimmed joint observer) Let $G = (Q, E, \delta, q_0)$ be a DFA and $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$ be its joint observer. A trimmed joint observer with respect to a set of illegal states $X_{il} \subseteq X$ is defined as $\Omega_{tj}(G) = (X', E_e, \delta_{tj}, x_0)$, where X' contains all states, from which, by a suitable editing action, the operator can prevent reaching an illegal state, and δ_{tj} is the transition relation satisfying

$$(\forall x, x' \in X')(\forall e \in E_e) \delta_{joi}(x, e) = x' \Rightarrow \delta_{tj}(x, e) = x'.$$

A trimmed joint observer can be computed by deleting all illegal states and all states from which an illegal state can be reached inevitably. Thus, the edited behavior computed in accordance with $\Omega_{tj}(G)$ can prevent any illegal state from being reached. It should be noted that a trimmed joint observer may contain a particular class of states, called *pre-emptive states*, which are defined as follows.

Definition 11 (Pre-emptive state) Let $G = (Q, E, \delta, q_0)$ be a DFA, $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$ be the joint observer, and $\Omega_{tj}(G) = (X', E_e, \delta_{tj}, x_0)$ be its trimmed joint observer. A state $x \in X'$ is said to be pre-emptive if

$$(\exists e \in E_{int}) \delta_{joi}(x, e) \notin X' \wedge (e \notin E_{era} \vee \delta_{joi}(x, e_-) \notin X')$$

holds. The set of pre-emptive states is denoted as X_p.

A state of the trimmed joint observer $x \in X'$ is pre-emptive if there exists an event whose occurrence at x (even if erased) leads the joint observer outside of X'. To guarantee that the evolution continues in X', an appropriate event should be inserted pre-emptively at x.

The trimmed joint observer can be computed according to Algorithm 3, which can be intuitively explained as follows. A state set X_0 contains all legal states and the transition relation δ_{tj} is equal to δ_{joi}. Line 6 deletes all

```
Algorithm 3: Computation of a trimmed joint observer
  Input: A DFA $G = (Q, E, \delta, q_0)$, its joint observer $\Omega_{joi}(G) = (X, E_e, \delta_{joi}, x_0)$, and a set of illegal
         states $X_{il} \subseteq X$.
  Output: The trimmed joint observer $\Omega_{tj}(G) = (X', E_e, \delta_{tj}, x_0)$.
1 Initialization: $X_0 \leftarrow X \backslash X_{il}$ and $\delta_{tj} \leftarrow \delta_{joi}$.
2 $i \leftarrow 0$.
3 repeat
4     $i \leftarrow i + 1$.
5     $X_i \leftarrow X_{i-1}$
6     $\delta_{tj} \leftarrow \delta_{tj} \cap (X_i \times E_e \times X_i)$.
7     foreach $x \in X_i$ do
8         foreach $e \in E_{ope}$ do
9             if $\delta_{joi}(x, e) \notin X_i \wedge (e \notin E_{era} \vee \delta_{joi}(x, e_-) \notin X_i)$ then
10                $\delta_{tj} \leftarrow \delta_{tj} \backslash (\{x\} \times (E_{ope} \cup E_-) \times X_i)$.
11                if $(\forall x' \in X_i)(\nexists \omega \in E_+^*)\delta_{tj}(x, \omega) = x' \wedge ((\forall e \in E_{ope})\delta(x', e) \in X_i \vee \delta(x', e_-) \in X_i)$
                   then
12                    $X_i \leftarrow X_i \backslash \{x\}$.
13 until $X_i = X_{i-1}$;
14 $X' \leftarrow X_i \backslash \{x \in X_i | (\nexists \omega_e \in E_e^*)\delta_{tj}(x_0, \omega_e) = x\}$.
15 if $X' = \emptyset$ then
16     Output $\Omega_{tj}(G)$ is empty.
17 else
18     $\delta_{tj} \leftarrow \delta_{tj} \cap (X' \times E_e \times X')$.
19     Output $\Omega_{tj} = (X', E_e, \delta_{tj}, x_0)$.
```

the arcs involving at least one state that does not belong to X_i. If the firing of an unerasable event $e \in E_{ope} \backslash E_{era}$ leads to a state not in X_i, or both the firing of $e \in E_{era}$ and e_- can lead to a state beyond X_i, then an event should be inserted immediately to prevent the joint observer from reaching a state beyond X_i. Line 10 is used to delete all the arcs labeled with such events or the corresponding erased events at x. If x cannot reach a non-pre-emptive state by inserting events, it should be deleted by line 12. If all states are deleted, we have no way to enforce the current-state opacity of G. Lines 3–13 update the desired state set recursively until no state in X_i can reach an undesired state by firing an unerasable event. Finally, line 17 removes from X_i all the states that are not reachable and obtains X'. The trimmed joint observer $\Omega_{tj}(G)$ is finally obtained, with set of states being X'.

Lemma 1 If $\Omega_{tj}(G)$ is not empty, $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G)) = P_{E_{ope}}(\mathcal{L}(G))$ holds.

Proof: First we observe that, regardless of the fact that $\Omega_{tj}(G)$ is empty or not, $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G))) \subseteq P_{E_{ope}}(\mathcal{L}(G))$. By $\Omega_{joi}(G) = \Omega_{ope}(G)||\Omega_{int}(G)$, we have $\mathcal{L}(\Omega_{joi}(G)) \subseteq \mathcal{L}(\Omega_{ope}(G))$. According to Definition 10, $\mathcal{L}(\Omega_{tj}(G)) \subseteq \mathcal{L}(\Omega_{joi}(G)) \subseteq \mathcal{L}(\Omega_{ope}(G))$. Thanks to Definition 3, we have $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G))) \subseteq \mu_{ope}(\mathcal{L}(\Omega_{ope}(G))) = P_{E_{ope}}(\mathcal{L}(G))$.

We are left to prove that, if $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G))) \neq P_{E_{ope}}(\mathcal{L}(G))$, then $\Omega_{tj}(G)$ must be empty. Consider a sequence ω that belongs to $P_{E_{ope}}(\mathcal{L}(G)) \setminus \mu_{ope}(\mathcal{L}(\Omega_{tj}(G)))$. We show that there does not exist a prefix $\omega' \in \bar{\omega}$ such that $\omega' \in \mu_{ope}(P_{E_{ope}}(\mathcal{L}(G)))$. In fact, let ω' be the longest among such prefixes. Since ω' cannot be continued, any continuation $\omega' e \in P_{E_{ope}}(\mathcal{L}(G))$ would lead to a condition that reaches an illegal state inevitably. Hence the state reached in $\Omega_{joi}(G)$ via ω' should also be forbidden. Since this holds for all prefixes, including the empty word ε, then $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G))) = \emptyset$, which implies that $\Omega_{tj}(G)$ is empty.

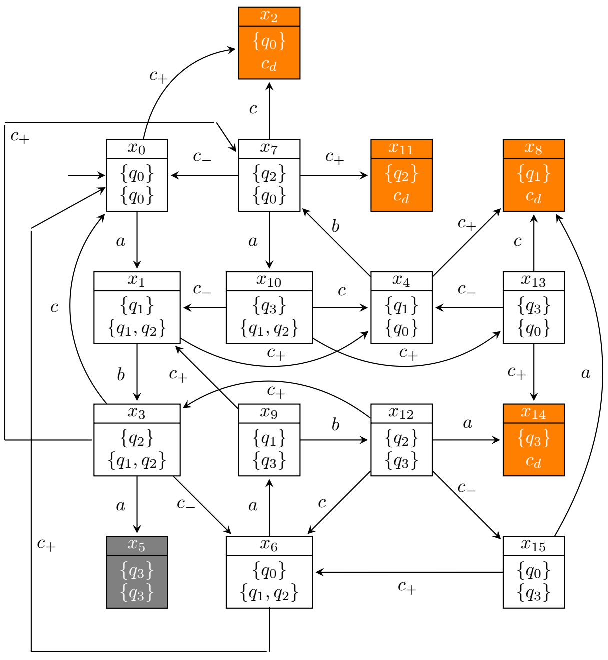
Theorem 1 Let $G = (Q, E, \delta, q_0)$ be a DFA, $Q_S \subseteq Q$ be a secret set, and $\Omega_{tj}(G)$ be its corresponding trimmed joint observer. Q_S is concealable if and only if $\Omega_{tj}(G)$ is not empty.

Proof: (only if) By contrapositive, suppose that $\Omega_{tj}(G)$ is empty. It is clear that there is no legal edited behavior and G cannot be edited to be current-state opaque.

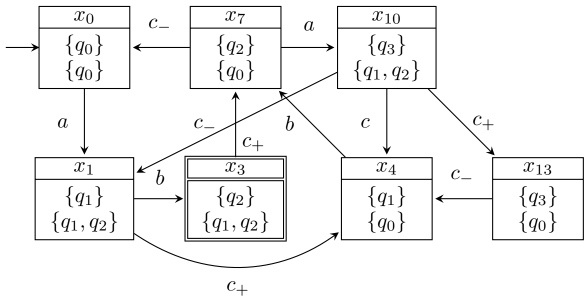
(if) According to Lemma 1, if $\Omega_{tj}(G)$ is not empty, $\mu_{ope}(\mathcal{L}(\Omega_{tj}(G))) = P_{E_{ope}}(\mathcal{L}(G))$ holds. In other words, for any sequence that is observed by an operator, there is always at least one edited sequence that can lead to a state in $\Omega_{tj}(G)$. As mentioned in the last section, reaching a falsely exposing state or a regular state does not reveal neither the secret nor the fact that the observation is being edited. Therefore, G can be edited to be current-state opaque.

Example 3 Consider the DFA G in Fig. 2. The joint observer is shown in Fig. 4(a). A state of the joint observer is represented as a rectangle, where the first entry denotes the name of the state. The second entry (a state set) is the estimation of the edited operator observer and the third (a state set) is the estimation of the edited intruder observer. The states in X_{nst} (X_{exp}) are highlighted in orange (grey). Now we illustrate the computation of the trimmed joint observer, as shown in Fig. 4(b).

Initially, it is $X_0 = X \backslash \{x_2, x_5, x_8, x_{11}, x_{14}\}$. The firing of unerasable event a at x_3, x_{12} and x_{15} leads to x_5, x_{14}, and x_8, respectively. By Lines 9–10, the arcs with label $E_{ope} \cup E_e$ at x_3, x_{12} and x_{15} are all deleted. After deleting these arcs, all states have at least one enabled transition. Therefore, no state is removed, i.e., $X_1 = X_0$, and the recursive process is ended. Finally, the four unreachable states x_6, x_9, x_{12} and x_{15} are removed, i.e., $X' = \{x_0, x_1, x_3, x_4, x_7, x_{10}, x_{13}\}$, and the pre-emptive state x_3, represented by a double rectangle, has only one output arc corresponding to $\delta_{tj}(x_3, c_+) = x_7$. Since $\Omega_{tj}(G)$ is not empty, Q_S is concealable.


x_0 $\{q_0\}$ $\{q_0\}$
x_1 $\{q_1\}$ $\{q_1,q_2\}$
x_2 $\{q_0\}$ c_d
x_3 $\{q_2\}$ $\{q_1,q_2\}$
x_4 $\{q_1\}$ $\{q_0\}$
x_5 $\{q_3\}$ $\{q_3\}$
x_6 $\{q_0\}$ $\{q_1,q_2\}$
x_7 $\{q_2\}$ $\{q_0\}$
x_8 $\{q_1\}$ c_d
x_9 $\{q_1\}$ $\{q_3\}$
x_{10} $\{q_3\}$ $\{q_1,q_2\}$
x_{11} $\{q_2\}$ c_d
x_{12} $\{q_2\}$ $\{q_3\}$
x_{13} $\{q_3\}$ $\{q_0\}$
x_{14} $\{q_3\}$ c_d
x_{15} $\{q_0\}$ $\{q_3\}$
a b c c_+ c_-

(a)


x_0 $\{q_0\}$ $\{q_0\}$
x_7 $\{q_2\}$ $\{q_0\}$
x_{10} $\{q_3\}$ $\{q_1,q_2\}$
x_1 $\{q_1\}$ $\{q_1,q_2\}$
x_3 $\{q_2\}$ $\{q_1,q_2\}$
x_4 $\{q_1\}$ $\{q_0\}$
x_{13} $\{q_3\}$ $\{q_0\}$
a b c c_+ c_-

(b)

Figure 4: (a) The joint observer and (b) the trimmed joint observer of the DFA in Fig. 2, where a double box denotes a pre-emptive state.

4.2 Complexity Analysis

Now we discuss the computational complexity of our approach. For a partially observable DFA with n states, the number of states of the current-state observer (for both operator and intruder) is at most 2^n and the computational complexity of the two observers is both in cases $\mathcal{O}(2^n)$. The number of states in the joint observer is at most 4^n and the computational complexity is $\mathcal{O}(4^n)$ since it is the parallel composition of the two observers. The trimmed joint observer is always smaller than the joint observer, i.e., it has at most 4^n states. By Algorithm 3, when computing a trimmed joint observer, we recursively delete the states that can reach illegal states inevitably. Since at least one state is removed at each execution of the *repeat-until* loop at line 3, both the *repeat-until* loop at line 3 and the *foreach* loop at line 7 are executed at most 4^n times. Therefore, the computational complexity of constructing a trimmed joint observer is thus $\mathcal{O}(4^{2n})$.

4.3 Online Opacity Enforcement via Editing Function

We now propose an online procedure to edit a sequence to prevent the intruder from inferring the secret. Let $G = (Q, E, \delta, q_0)$ be a DFA and $\Omega_{tj}(G) = (X', E_e, \delta_{tj}, x_0)$ be its trimmed joint observer. It has been discussed in the above section that the language of $\Omega_{tj}(G)$ is the legal edited behavior. To enforce G to be current-state opaque, an editing function φ can be decided by $\Omega_{tj}(G)$. The online edited behavior can follow φ, as illustrated in Procedure 1.

Procedure 1 Online procedure for opacity enforcement

```
1. Initialization: $\omega \leftarrow \varepsilon$. /* output sequence */
2. Initialization: $\omega_e \leftarrow \varepsilon$. /* edited sequence */
3. Initialization: $x_c \leftarrow x_0$. /* current state */
4. Select an inserted event sequence $\omega_+ \in E_+^*$ such that $\delta_{tj}(x_c, \omega_+) = x \notin X_p$.
5. $x_c \leftarrow x$.
6. $\omega_e \leftarrow \omega_e\omega_+$. /* $\varphi(\omega) = \omega_e$ */
7. Wait for an output $e$.
8. $\omega \leftarrow \omega e$.
9. Select $e' \in \{e, e_-\}$ such that $\delta_{tj}(x_c, e')!$ and let $\omega_e \leftarrow \omega_e e'$.
10. $x_c \leftarrow \delta_{tj}(x_c, e')$.
11. Go to step 4.
```

Initially, both the output sequence ω and the edited sequence ω_e are ε and the current state x_c is x_0. The operator needs to insert a (possibly empty) sequence of events in E_+ to reach a non-pre-emptive state. At a non-pre-emptive state, the operator waits for an observable event e to be generated by the plant and decides whether it should be erased or not. These steps are cyclicly repeated.

Example 4 Consider the DFA G in Fig. 2 and its trimmed joint observer $\Omega_{tc}(G)$ in Fig. 4(b). If sequence ab is observed by the operator, the joint observer reaches the pre-emptive state x_3, at which only c_+ is enabled. Thus, the operator must insert a fake event c and non-preemptive state x_7 is reached. The operator waits for a new event to be generated: if such an event is c, then it must be erased since c is not enabled at x_7 and the observer moves to state x_0.

5 Conclusions and Future Work

This paper proposes a joint observer-based approach to enforce a DFA to be current-state opaque by editing the output. A joint observer is the parallel composition of the edited operator observer and the edited intruder observer, which is used to enumerate all feasible edited behaviors. The illegal edit behavior is formalized by the illegal states of the joint observer, which reveal the secret or violate stealthiness. The generated language of the trimmed joint observer is the set of all edited words produced by admissible editing functions. Thus, it allows one to perform the verification of concealability. Finally, we use an online procedure to enforce a DFA to be current-state opaque.

Future work includes extending the proposed approach to the enforcement of initial state opacity. In addition, it is interesting to consider distributed or decentralized approaches to deal with large-scale systems.

6 Appendix

6.1 Proof of Proposition 1

(a) By construction, one can readily verify that all observations produced by the plant are strings in the language of the edited operator observer since $P_{E_{ope}}(\mathcal{L}(G)) = \mathcal{L}(\mathcal{O}(G, E_{ope})) \subseteq \mathcal{L}(\Omega_{ope}(G))$: these strings correspond to a *null editing function* that does not insert or erase any event. The language of the edited operator observer also contains additional strings since after each observation it may be possible to insert arbitrary strings of the events in E_+ (which are selflooped at each state) and the occurrence of an event $e \in E_{era}$ can be replaced by an erased event $e_- \in E_-$ (the two events are always in parallel). These actions exactly define the class of editing functions as described in Definition 2.

(b) Follows from the fact that $\mathcal{S}_{ope}(\omega) = \hat{\delta}_{ope}(c_{0,ope}, \omega) = \delta_{ope}(c_{0,ope}, \omega)$, i.e., the set of states consistent with observation ω can be determined from both observers (nominal and edited ones). An erased event satisfies $\mu_{ope}(e_-) = e$, and e_- is parallel with e. An inserted event satisfies $\mu_{ope}(e_+) = \varepsilon$, and e_+ is in a self-loop. Thus, $\hat{\delta}_{ope}(c_{0,ope}, \omega) = \delta_{ope}(c_{0,ope}, \omega_e)$.

6.2 Proof of Proposition 2

(a) By Definitions 4 and 6, we have $\varepsilon \in \mathcal{L}(\Omega_{int}(G))$. For an edited sequence $\omega_e \in E_e^*$, if $\delta_{int}(c_{0,int}, \omega_e) = c \neq c_d$, then $\mu_{int}(\omega_e) \in P_{E_{int}}(\mathcal{L}(G))$ and for all edited events $e \in E_e$, $\delta_{int}(c, e)!$ holds. If $\delta_{int}(c_{0,int}, \omega_e) = c_d$, then $\mu_{int}(\omega_e) \notin P_{E_{int}}(\mathcal{L}(G))$ and $(\nexists e \in E_e)\delta_{int}(c_d, e)!$. Therefore, $\mathcal{L}(\Omega_{int}(G)) = \{\omega_e \in E_e^* | \mu_{int}(\omega_e) \in P_{E_{int}}(\mathcal{L}(G))\}(\{\varepsilon\} \cup E_e)$ holds..

(b) i) Follows from the fact that $\mathcal{S}_{int}(\omega) = \hat{\delta}_{int}(c_{0,int}, \omega) = \delta_{int}(c_{0,int}, \omega)$, i.e., the set of states consistent with observation ω can be determined from both observers (nominal and edited ones). An erased event satisfies $\mu_{int}(e_-) = \varepsilon$, and e_- is in a self-loop. An inserted event satisfies $\mu_{int}(e_+) = e$, and e_+ is parallel with e. Therefore, it holds $\hat{\delta}_{int}(c_{0,int}, \omega) = \hat{\delta}_{int}(c_{0,int}, \mu_{int}(\omega_e)) = \delta_{int}(c_{0,int}, \omega_e)$.

ii) As mentioned in (a), for an edited sequence $\omega_e \in \mathcal{L}(\Omega_{int}(G))$ with $\mu_{int}(\omega_e) \in P_{E_{int}}(\mathcal{L}(G))$ and an editing event $e \in E_e$ such that $\mu_{int}(\omega_e e) \notin P_{E_{int}}(\mathcal{L}(G))$, if $\hat{\delta}_{int}(c_{0,int}, \mu_{int}(\omega_e)) = c$, then we have $\delta_{int}(c_{0,int}, \omega_e e) = \delta_{int}(\delta_{int}(c_{0,int}, \omega_e), e) = \delta_{int}(c, e) = c_d$.

6.3 Proof of Proposition 3

According to Propositions 1 and 2, for an observation $\omega \in P_{E_{obs}}(\mathcal{L}(G))$ and an editing function φ such that $\varphi(\omega) = \omega_e$ and $\omega' = \mu_{int}(\omega_e) \in P_{E_{int}}(\mathcal{L}(G))$, we have $\mathcal{S}_{ope}(\omega) = \delta_{ope}(c_{0,ope}, \omega_e)$ and $\mathcal{S}_{int}(\omega) = \delta_{int}(c_{0,int}, \omega_e)$. By Definition 7, $\delta_{joi}((c_{0,ope}, c_{0,int}), \omega_e) = (\delta_{ope}(c_{0,ope}, \omega_e), \delta_{int}(c_{0,int}, \omega_e)) = (\mathcal{S}_{ope}(\omega), \mathcal{S}_{int}(\omega'))$. On the other hand, if $\omega' \notin P_{E_{int}}(\mathcal{L}(G))$, by Proposition 2, $\delta_{int}(c_{0,int}, \omega') = c_d$ holds. In this case, we have $\delta_{joi}((c_{0,ope}, c_{0,int}), \omega_e) = (\mathcal{S}_{obs}(\omega), c_d)$.

References

[1] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

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

[3] A. Saboori, and C. Hadjicostis, "Notion of security and opacity in discrete event systems," in *Proc. 46th IEEE Conf. Decis. Control*, 2007, pp. 5056–5061.

[4] X. Yin, and S. Lafortune, "A new approach for the verification of infinite-step and K-step opacity using two-way observers," *Automatica*, vol. 80, pp. 162–171, 2017.

[5] K. Zhang, X. Yin, and M. Zamani, "Opacity of nondeterministic transition systems: A (bi)simulation relation approach," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 5116–5123, 2019.

[6] C. G. Cassandras, and S. Lafortune, *Introduction to Discrete Event Systems (2nd)*, Springer, 2018.

[7] A. Saboori, and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies for secure discrete event systems," in *Proc. 47th IEEE Conf. Decis. Control*, Dec. 2008, pp. 889–894.

[8] A. Saboori, and C. N. Hadjicostis, “Opacity-enforcing supervisory strategies via state estimator constructions,” *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1153–1165, 2012.

[9] J. Dubreil, P. Darondeau, and H. Marchand, “Supervisory control for opacity,” *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1089–1100, 2010.

[10] Y.-C. Wu, and S. Lafortune, “Synthesis of insertion functions for enforcement of opacity security properties,” *Automatica*, vol. 50, no. 5, pp. 1336–1348, 2014.

[11] Y. Ji, Y.-C. Wu, and S. Lafortune, “Enforcement of opacity by public and private insertion functions,” *Automatica*, vol. 93, pp. 369–378, 2018.

[12] Y.-C. Wu, and S. Lafortune, “Synthesis of opacity-enforcing insertion functions that can be publicly known,“ in *Proc. 54th IEEE Conf. Decis. Control*, 2015, pp. 3506–3513.

[13] Y. Ji, and S. Lafortune, “Enforcing opacity by publicly known editing functions,” in *Proc. 56th IEEE Conf. Decis. Control*, 2017, pp. 4866–4871.

[14] Y. Ji, X. Yin, and S. Lafortune, “Opacity enforcement using nondeterministic publicly-known editing functions,” *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4369–4376, Oct. 2019.

[15] S. Mohajerani, Y. Ji, and S. Lafortune, “Efficient synthesis of edit functions for opacity enforcement using bisimulation-based abstractions,” in *Proc. 57th IEEE Conf. Decis. Control*, 2018, pp. 3573–3578.

[16] S. Mohajerani, Y. Ji, and S. Lafortune, “Compositional and abstraction-based approach for synthesis of editing function for opacity enforcement,” *IEEE Transactions on Automatic Control*, vol. 65, no. 8, pp. 3349–3364, Aug. 2019.

[17] F. Cassez, J. Dubreil, and H. Marchand, “Synthesis of opaque systems with static and dynamic masks,” *Formal Methods Syst. Design*, vol. 40, no. 1, pp. 88–115, 2012.

[18] X. Yin, and S. Li, “Synthesis of dynamic masks for infinite-step opacity,“ *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1429–1441, Apr. 2019.

[19] Q. Zhang, C. Seatzu, Z. Li, and A. Giua, “Joint state estimation under attack of discrete event systems,” *IEEE Access*, vol. 9, pp. 168068–168079, 2021.

[20] Q. Zhang, C. Seatzu, Z. W. Li, and A. Giua, “Selection of a stealthy and harmful attack function in discrete event systems,” *Scientific Reports,* vol. 12, 2022.

[21] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua, “Current-State Opacity Enforcement in Discrete Event Systems Under Incomparable Observations,” *Discrete Event Dynamic Systems*, vol. 28. no. 2, pp. 161–182, 2018.

[22] J. Balun and T. Masopust, “Comparing the notions of opacity for discrete-event systems,” *Discrete Event Dynamic Systems*, 31(4), 553–582, 2021.

[23] Y. C. Wu and S. Lafortune, “Comparative analysis of related notions of opacity in centralized and coordinated architectures,” *Discrete Event Dynamic Systems*, pp. 307–339, 2013.