Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 55-28 (2022) 135–141

You Don't Know What I Know: On Notion of High-Order Opacity in Discrete-Event Systems ⋆

Bohan Cui * **Xiang Yin** * **Shaoyuan Li** * **Alessandro Giua** **

* *Department of Automation, Shanghai Jiao Tong University,*
Shanghai, 200240, China
(E-mail: {bohan_cui, yinxiang, syli}@sjtu.edu.cn).
** *DIEE, University of Cagliari, Cagliari 09123, Italy.*
(E-mail: giua@unica.it).

Abstract: In this paper, we investigate a class of information-flow security properties called *opacity* in partial-observed discrete-event systems. Roughly speaking, a system is said to be opaque if the intruder, which is modeled by a passive observer, can never determine the "secret" of the system for sure. Most of the existing notions of opacity consider secrets related to the actual behaviors of the system. In this paper, we consider a new type of secret related to the *knowledge of the system user*. Specifically, we assume that the system user also only has partial observation of the system and has to reason the actual behavior of the system. We say a system is *high-order opaque* if the intruder can never determine that the system user knows some information of importance based on its own incomparable information. We provide the formal definition of high-order opacity. Two algorithms are provided for the verification of this new notion: one with doubly-exponential complexity for the worst case and the other with single-exponential complexity. Illustrative examples are provided for the new notion of high-order opacity.

Keywords: Opacity, Discrete Event Systems, Partial Observation.

1. INTRODUCTION

With the development of information and network technologies, smart devices with both computation and communication capabilities have been widely used in cyber-physical control systems. The large information transmission between devices, on the one hand, makes control systems more flexible and intelligent by enabling, for example, edge or cloud based control architectures. On the other hand, however, the large amount of communications also makes security issue much more severe. Therefore, security and privacy considerations have been becoming increasingly more important in the analysis and design of networked cyber-physical systems (CPS).

In this work, we consider an important class of information-flow security properties, called *opacity*, in the context of discrete-event systems (DES), a widely used formal model for describing high-level behaviors of CPS (Cassandras and Lafortune, 2021). Generally speaking, opacity captures whether or not some secret behaviors of the system can be revealed to an external intruder that can access partial information-flow of the system. A system is said to be opaque if the intruder can never infer the secret behaviors of the system based on its own observation. It has been shown that the notions of opacity subsume many existing security properties in the literature (Bryans et al., 2008). Due to its importance, verification and enforcement of opacity has drawn considerable attention in the past few years; see, e.g., Lafortune et al. (2018); Behinaein et al. (2019); Yin and Li (2020); Liu and Zamani (2021); Balun and Masopust (2021); Yin et al. (2021); Wintenberg et al. (2022).

Since opacity is an information-flow security property, one of the key ingredients in its definition is what is the "secret" the system wants to hide against the intruder. The most general formulation of opacity is language-based opacity, where secret is modeled as a set of secret strings; see, e.g., Lin (2011). In specific applications, however, secret strings usually have concrete meanings, e.g., visited a critical location at some instant. Therefore, state-based secret have been more widely adopted in the verification and synthesis of opacity. State-based opacity includes, e.g., initial-state opacity (Saboori and Hadjicostis, 2013; Lai et al., 2021), current-state opacity (Tong et al., 2017), infinite-step opacity (Saboori and Hadjicostis, 2011; Yin and Lafortune, 2017) and K-step opacity (Yin et al., 2019; Ma et al., 2021).

All of the aforementioned notions of opacity consider the secret of system as some *actual behaviors of importance*, e.g., is visiting or has visited a secret state. In some applications, however, the secret of the system may be *the knowledge of the user* about the current status of the system. To motivate the "knowledge security" issue, let us consider the following scenario. Suppose that there are two investors in the market. Each of them knows the

⋆ This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61803259).

10.1016/j.ifacol.2022.10.335

trading strategy of the other but does not the information available to the other. Suppose that one investor wants to investigate some undisclosed information about a stock in order to decide to buy it or not. At the same time, he does not want the other investor to know the fact that he has obtained sufficient information for trading; otherwise, the other investor may take the advantage of the information that the first investor will buy the stock. This example suggests that, in some scenarios, knowing something of importance is also closely related to the security consideration.

In this paper, we investigate the verification of opacity from a new perspective by considering the secret of the system as the knowledge of the system user. Specifically, we model the user and the intruder as two observers with different information-flow. The objective of the user is to gain sufficient knowledge about the system, e.g., for the purpose of decision-making. Here we consider a specific type of knowledge objective, which is to distinguish certain pairs of states (Wang et al., 2007; Sears and Rudie, 2014). On the other hand, the intruder may use its own information to infer whether or not the user has sufficient knowledge. We formulate the knowledge-security of the user in terms of the notion of *high-order opacity*. Specifically, the system user is said to be high-order opaque if the intruder can never determine for sure whenever the user knows something of importance. We present two algorithms for verifying high-order opacity. The first algorithm is based on the construction of two successive observers, which yields a doubly-exponential complexity. The second algorithm is based on estimating, from the intruder's point of view, all state pairs that cannot be distinguished by the user; the complexity of the second approach is only single-exponential in the size of the plant.

2. PRELIMINARIES

2.1 System Model

Let Σ be a finite set of events. A string is a finite sequence of events and Σ^* denotes the set of all strings over Σ including the empty string ϵ. For any string $s \in \Sigma^*$, $|s|$ denotes the length of s with $|\epsilon| = 0$. A language $L \subseteq \Sigma^*$ is a set of strings and we denote by $\bar{L}$ the prefix-closure of language L, i.e., $\bar{L} = \{s \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } sw \in L\}$.

We consider a DES modeled by a deterministic finite-state automaton (DFA)

$$G = (X, \Sigma, \delta, x_0),$$

where X is a finite set of states, Σ is a finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, where for any $x, x' \in X$, $\sigma \in \Sigma$, $x' = \delta(x, \sigma)$ means that there exists a transition from state x to state x' via event σ and $x_0 \in X$ is the initial state. The transition function is also extended to $\delta : X \times \Sigma^* \to X$ recursively by: (i) for any $x \in X$, $\delta(x, \epsilon) = x$ and (ii) for any $x \in X, s \in \Sigma^*, \sigma \in \Sigma$, we have $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$. The set of all strings generated by G starting from state $x \in X$ is defined as $\mathcal{L}(G, x) = \{s \in \Sigma^* : \delta(x, s)!\}$, where "!" means "is defined". The set of all strings generated by G is defined as $\mathcal{L}(G) := \mathcal{L}(G, x_0)$. For any $s \in \mathcal{L}(G)$, we write $\delta(x_0, s)$ simply as $\delta(s)$. For the sake of simplicity, we assume that system G is live, i.e., for any $x \in X$, there exists $\sigma \in \Sigma$ such that $\delta(x, \sigma)!$.

Remark 2.1. Here we consider DES with deterministic transitions with a unique initial state. This is without loss of generality in the partial-observation setting because it is well-known that one can always use unobservable events to mimic non-determinism.

Let $\Sigma' \subseteq \Sigma$ be a subset of events. The natural projection from Σ to Σ' is a mapping $P_{\Sigma'} : \Sigma^* \to \Sigma'^*$ defined recursively by:

$$P_{\Sigma'}(\epsilon) = \epsilon \text{ and } P_{\Sigma'}(s\sigma) = \begin{cases} P_{\Sigma'}(s)\sigma & \text{if } \sigma \in \Sigma' \\ P_{\Sigma'}(s) & \text{if } \sigma \notin \Sigma' \end{cases}.$$

The natural projection is also extended to $P_{\Sigma'} : 2^{\Sigma^*} \to 2^{\Sigma'^*}$by: for any $L \subseteq \Sigma^*$, $P_{\Sigma'}(L) = \{P_{\Sigma'}(s) \in \Sigma'^* : s \in L\}$.

2.2 Intruder Model and Current-State Opacity

In the context of information-flow security analysis, the intruder is usually modeled as a passive observer that (i) knows the model of the system; and (ii) can observe partial behaviors generated by the system. Formally, we assume that the event set is partitioned as

$$\Sigma = \Sigma_a \dot{\cup} \Sigma_{ua},$$

where Σ_a and Σ_{ua} are the sets of observable events and unobservable events of the intruder, respectively. For the sake of simplicity, we use notation $P_a : \Sigma^* \to \Sigma_a^*$ to denote the natural projection from Σ to Σ_a. Then upon the occurrence of string $s \in \mathcal{L}(G)$, the intruder observes $P_a(s) \in \Sigma_a^*$.

For any observation $\alpha \in P_a(\mathcal{L}(G))$, the intruder can *estimate* the current-state of the system based on α and the system model G. Formally, the *current-state estimate* of the intruder upon the observation of α is defined by

$$\hat{X}_a(\alpha) = \{\delta(s) \in X : \exists s \in \mathcal{L}(G) \text{ s.t. } P_a(s) = \alpha\}.$$

In the context of standard state-based opacity analysis, it is assumed that the system has a "secret" modeled by a set of secret states $X_S \subseteq X$. We denote by $X_{NS} = X \setminus X_S$ the set of non-secret states. Then the system is said to be *current-state opaque* if the intruder can never determine for sure that the system is currently at a secret state. In other words, for any string that leads the system to a secret state, there should exist at least one string that leads the system to a non-secret state such that they have the same observation from the intruder's point of view. We review the definition of current-state opacity as follows.

Definition 1. (Current-State Opacity). Given system G, a set of secret states X_S and a set of intruder's observable events $\Sigma_a \subseteq \Sigma$, system G is said to be current-state opaque (w.r.t. Σ_a and X_S) if

$$\begin{aligned}&(\forall s \in \mathcal{L}(G) : \delta(s) \in X_S) \\ &(\exists t \in \mathcal{L}(G) : \delta(t) \in X_{NS})[P_a(s) = P_a(t)]\end{aligned} \quad (1)$$

or equivalently,

$$(\forall s \in \mathcal{L}(G))[\hat{X}_a(P_a(s)) \not\subseteq X_S]. \quad (2)$$

3. NOTION OF HIGH-ORDER OPACITY

In the standard formulation of current-state opacity, the secret is modeled as the actual behavior of the system.

In some applications, the user may want to hide its *knowledge about the system*. To this end, in this section, we introduce the notion of *high-order opacity*. We first present a motivating example and then provide the formal definition.

3.1 Motivating Example

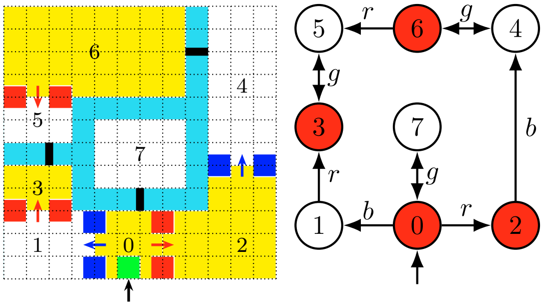
System Model: Suppose that there is a robot moving in a workspace with rivers, bridges and checkpoints as shown in Figure 1(a). We assume that the robot can cross the bridges (denoted by black lines) via both directions, but the checkpoints (denoted by red and blue blocks) are only one-way whose directions are specified by arrows in the figure. The mobility of the robot can be modeled as DFA G shown in Figure 1(a), where states correspond to regions in the workspace and events b, r and g corresponds to "passing a blue checkpoint", "passing a red checkpoint" and "crossing the bridge", respectively.

User Model: Suppose that there is a central station that wants to communicate with the robot, e.g, to send commands. We assume that communication signals are only available in service regions and the central station cannot communicate with the robot in no-service regions, which are marked by yellow in Figure 1(a) and are represented by red states $\{0, 2, 3, 6\}$ in the DFA model. Furthermore, we assume that the central station has sensors placed at each bridge and each red checkpoint, i.e., it can observe the occurrences of events r and g. In order to make sure that each command sent can be received by the robot *for sure*, the central station will send command to the robot only when it knows for sure that the robot is currently at white regions.

Intruder Model: At the same time, we assume that there is an intruder having sensors placed at each bridge and each blue checkpoint, i.e., it can observe the occurrences of events b and g. By knowing the strategy of the central station, the intruder may try to hack to the communication channel between the central station and the robot. Therefore, it will successfully intercept transmitted command when it knows for sure that the central station knows for sure that the robot is at a service region.

Analysis: In order to establish communications with the central station, the robot may choose path $0 \xrightarrow{g} 7$, which is the shortest path to reach a service region. Furthermore, the central station will observe event g and upon which it knows for sure that the robot is indeed in a service region. Hence, it will send a command to the robot. However, at the same time, the intruder will also observe event g and upon which it knows for sure that the central station will send a command. Therefore, the command will be intercepted by the intruder, which makes the system not secure.

To establish a secure communication, the robot can choose path $0 \xrightarrow{r} 2 \xrightarrow{b} 4 \xrightarrow{g} 6 \xrightarrow{r} 5$ to go to service region 5. Along this path, the central station will observe rgr and it knows for sure that the robot is in a service region. However, the intruder will observe bg and it may think that the robot may have chosen path $0 \xrightarrow{b} 1 \xrightarrow{r} 3 \xrightarrow{g} 5$. If it is the case, then the central station will observe rg and it cannot distinguish if the robot is at state 5 or state 6. Therefore, along this path, the central station will know for sure that the robot is in a service region, while the intruder does not know that the central station knows that. This means that the communication in this scenario is secure. In what follows, we will formulate such a scenario of "*the intruder does not know that the user knows something*" using the notion of high-order opacity.



(a) Workspace of the robot. (b) DFA model of the robot.

Fig. 1. A motivating example with $\Sigma_o = \{r, g\}$ and $\Sigma_a = \{b, g\}$.

3.2 Knowledge of the User

As we discussed in the above motivating example, the user of the system may also have its own observation and based on which it can obtain certain knowledge about the current status of the system. To formalize the issue of *knowledge security*, we further assume that the event set is also partitioned as

$$\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo},$$

where Σ_o is the set of events that can be observed by the user and Σ_{uo} is the set of events that cannot be observed by the user. Similarly, we denote by $P_o : \Sigma^* \to \Sigma_o^*$ the natural projection from Σ to Σ_o. Also, for any observation of the user $\alpha \in P_o(\mathcal{L}(G))$, we denote by $\hat{X}_o(\alpha)$ the current-state estimate of the user upon the observation of α. Note that, there is no relationship between the observation of the intruder and the observation of the user, i.e., Σ_a and Σ_o can be incomparable, because the intruder and the user may have different sensors monitoring the system.

For the purpose of decision-making such as control or diagnosis, the user usually wants to determine whether or not it knows some fact based on its partial observation. In general, the knowledge of the user can be defined as a predicate on its observations

$$\texttt{Know} : P_o(\mathcal{L}(G)) \to \{\texttt{true}, \texttt{false}\}$$

such that "$\texttt{Know}(\alpha) = \texttt{true}$" means that the user knows some fact based on observation $\alpha \in P_o(\mathcal{L}(G))$.

In general, by observing $\alpha \in P_o(\mathcal{L}(G))$, the user's complete knowledge about the system is $P_o^{-1}(\alpha) \cap \mathcal{L}(G)$, which is the set of all possible strings consistent with the observation. In this work, we consider a more concrete type of knowledge of the user called *distinguishability*. Specifically, we assume that the user is interested in distinguishing certain pairs of states (a.k.a. the disambiguation task)

$$T_{\text{spec}} \subseteq X \times X.$$

Specifically, the user wants to distinguish between each pair of states $(x, x') \in T_{\text{spec}}$ and if so, we say that it has sufficient knowledge w.r.t. the disambiguation task T_{spec}.

Then using T_{spec}, we can specify the knowledge predicate $\texttt{Know} : P_o(\mathcal{L}(G)) \to \{\texttt{true}, \texttt{false}\}$ by: for any $\alpha \in P_o(\mathcal{L}(G))$, we have $\texttt{Know}(\alpha) = \texttt{true}$ iff

$$(\forall s, t \in \mathcal{L}(G))[P_o(s) = P_o(t) = \alpha \Rightarrow (\delta(s), \delta(t)) \notin T_{\text{spec}}] \quad (3)$$

or equivalently,

$$(\hat{X}_o(\alpha) \times \hat{X}_o(\alpha)) \cap T_{\text{spec}} = \emptyset. \quad (4)$$

Intuitively, we have $\texttt{Know}(\alpha) = \texttt{false}$ whenever there are two different strings both have the same observation α but lead to two different states whose pair is in T_{spec}.

Remark 3.1. Although here we consider a specific type of user knowledge captured by disambiguation task T_{spec}, this task is general enough for many practical requirements. For example, it can captures "whether or not the user knows its current location precisely" by setting $T_{\text{spec}} = \{(x, x') \in X \times X : x \neq x'\}$. In general, how to define T_{spec} is problem dependent and in this work, we will work on the generic knowledge requirement T_{spec}.

3.3 High-Order Opacity

Before formally introducing the definition of high-order opacity, we summarize the capabilities of the intruder considered (a.k.a. intruder model) as follows

- A1 It knows the DFA model G of the system;
- A2 It can observe the occurrence of each event in Σ_a generated online;
- A3 It knows that the user can observe the occurrence of each event in Σ_o online but itself cannot observe the occurrences of events $\Sigma_o \setminus \Sigma_a$ directly.

The basic idea of high-order opacity still follows the essence of opacity, which is plausible deniability for the secret behavior. However, here "secret" is captured by the knowledge predicate $\texttt{Know}$ rather than secret states X_S. We require that the intruder should never be able to determine for sure that the user knows something. This leads to the following definition.

Definition 2. (High-Order Opacity). Given system G, a disambiguation task $T_{\text{spec}} \subseteq X \times X$, a set of intruder's observable events $\Sigma_a \subseteq \Sigma$ and a set of user's observable events $\Sigma_o \subseteq \Sigma$, system G is said to be high-order opaque (w.r.t. T_{spec}, Σ_a and Σ_o) if

$$\begin{aligned} &(\forall s \in \mathcal{L}(G) : \texttt{Know}(P_o(s)) = \texttt{true}) \quad (5)\\ &(\exists t \in \mathcal{L}(G) : \texttt{Know}(P_o(t)) = \texttt{false})[P_a(s) = P_a(t)]. \end{aligned}$$

The above definition can be understood as follows. Let $s \in \mathcal{L}(G)$ be an actual string generated by the system. Upon the occurrence of s, the user observes $P_o(s)$ while the intruder observes $P_a(s)$. The user's knowledge is completely determined by $P_o(s)$ but the intruder does not know the user's knowledge perfectly. From the intruder's point of view, the user may have observed any string in $P_o(P_a^{-1}(P_a(s)) \cap \mathcal{L}(G))$, and if for any observation α in this language, we have $\texttt{Know}(\alpha) = \texttt{true}$, then the intruder knows for sure that predicate $\texttt{Know}$ holds true for the user. Therefore, the condition in Equation (5) can also be expressed equivalently by

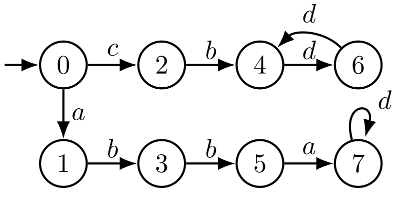


Fig. 2. System G with $\Sigma_o = \{b, d\}$ and $\Sigma_a = \{a, b\}$.

$$\begin{aligned} &(\forall \alpha \in P_a(\mathcal{L}(G)))(\exists \beta \in P_o(P_a^{-1}(\alpha) \cap \mathcal{L}(G)))\\ &[\texttt{Know}(\beta) = \texttt{false}]. \quad (6) \end{aligned}$$

Remark 3.2. High-order opacity can also be considered as an instance of *language-based opacity* Lin (2011). Specifically, we can define $L_S = \{s \in \mathcal{L}(G) : \texttt{Know}(P_o(s)) = \texttt{true}\}$ as the secret language. Then high-order opacity becomes language-based opacity w.r.t. secret language L_S and projection P_a. Therefore, the key in checking high-order opacity is to identify L_S effectively.

We illustrate the notion of high-order opacity by the following example.

Example 1. Let us consider system G shown in Figure 2, where the user can observe $\Sigma_o = \{b, d\}$ and the intruder can observe $\Sigma_a = \{a, b\}$. The disambiguation task of the user is simply to determine the current-state of the system, i.e., $T_{\text{spec}} = \{(x, x') \in X \times X : x \neq x'\}$. Clearly this system is high-order opaque. To see this, we note that the user can determine its current state only after it observes the first occurrence of event d. For string $s = abbad^n$ such that $\texttt{Know}(bbd^n) = \texttt{true}$, the intruder observes $P_a(s) = abba$ and it may think that what actually happens is string $s' = abba$ such that $\texttt{Know}(P_o(abba)) = \texttt{false}$ because the user cannot distinguish between states 5 and 7 by observing bb. Similarly, for $t = cbd^{2n}$ such that $\texttt{Know}(bd^{2n}) = \texttt{true}$, the intruder observes $P_a(t) = b$ and it may think that what actually happens is string $t' = cb$ such that $\texttt{Know}(P_o(cb)) = \texttt{false}$ because the user cannot distinguish between states 3 and 4 by observing b. In other word, the intruder can never know that the observer knows the current state of the system G.

Remark 3.3. It is worth noting that high-order opacity only requires that *whenever* the user has the knowledge to distinguish T_{spec}, the intruder does not know this fact. This definition itself does not require that the user can always or will eventually have this knowledge. This is essentially a utility requirement, which is a different consideration apart from the security consideration. In other words, it is even possible that the intruder knows the secret information (distinguishing T_{spec}), but it does not know whether or not the user also knows this secret information. For instance, in Figure 2, if the intruder observes $abba \in P_a(\mathcal{L}(G))$, it can determine for sure that the system is currently at state 7. However, it can not determine whether the user knows the current state because $\hat{X}_o(P_o(abba)) = \{5, 7\}$, $\hat{X}_o(P_o(abbad)) = \{7\}$ and $P_a(abba) = P_a(abbad) = abba$. This scenario does not violate the requirement of high-order opacity according to the definition.

Remark 3.4. The new notion of high-order opacity subsumes the standard current-state opacity as defined in Definition 1. To see this, we can set $\Sigma_o = \Sigma$ and $T_{\text{spec}} = \{(x, x) \in X : x \in X_{NS}\}$ and then high-order opacity becomes current-state opacity. This is because T_{spec} can never be distinguished when the system is not at a secret state. Furthermore, since the user knows the current state perfectly, $\texttt{Know}(\alpha) = \texttt{true}$ is equivalent to the fact that the system is currently at a secret state. This reduction implies that high-order opacity is more general than current-state opacity. Since it is known that the verification of current-state opacity is PSPACE-complete (Balun and Masopust, 2020), we can conclude immediately that verifying high-order opacity is at least PSPACE-hard.

4. VERIFICATION OF HIGH-ORDER OPACITY

In this section, we present two algorithms for the verification of the proposed notion of high-order opacity; one is based on the construction of double-observer and the other is based on the construction of state-pair-observer.

4.1 Double-Observer Approach

Let $G = (X, \Sigma, \delta, x_0)$ be a DFA and $\Sigma' \subseteq \Sigma$ be a subset of observable event. Then the observer automaton of G w.r.t. Σ' is a new DFA

$$\text{Obs}_{\Sigma'}(G) = (Q, \Sigma, f, q_0),$$

where $Q \subseteq 2^X \setminus \emptyset$ is the set of states, $q_0 = \{\delta(w) \in X : w \in (\Sigma \setminus \Sigma')^*\}$ is the initial state, and $f : Q \times \Sigma \to Q$ is the deterministic transition function defined by: for any $q \in Q$ and $\sigma \in \Sigma$, we have

- if $\sigma \in \Sigma'$, then
$$f(q, \sigma) = \{\delta(x, \sigma w) \in X : x \in q, w \in (\Sigma \setminus \Sigma')^*\}$$
- if $\sigma \in \Sigma \setminus \Sigma'$ and there exists $x \in q$ such that $\delta(x, \sigma)!$, then $f(q, \sigma) = q$.

Intuitively, the observer automaton tracks all possible current states of system based on observations in Σ'^*. Specifically, for any $\alpha \in P_{\Sigma'}(\mathcal{L}(G))$, $f(q_0, \alpha)$ is the current-state estimate of α w.r.t. Σ'. For technical purposes, here we further add self-loops at each state for those feasible but unobservable events in $\Sigma \setminus \Sigma'$, which ensures that $\mathcal{L}(G) \subseteq \mathcal{L}(\text{Obs}_{\Sigma'}(G))$.

Note that the knowledge of the user is described by distinguishability which is based on the current-state estimation. Therefore, we can build the observer automaton w.r.t. Σ_o denoted by $\text{Obs}_{\Sigma_o}(G) = (Q_o, \Sigma, f_o, q_{o,0})$ to capture this issue. Specifically, since for any $\alpha \in P_o(\mathcal{L}(G))$, the knowledge predicate $\texttt{Know}$ holds true iff $(f_o(\alpha) \times f_o(\alpha)) \cap T_{\text{spec}} = \emptyset$. Therefore, we define the set of observer states for which the knowledge predicate holds true as

$$Q_{o,S} = \{q \in Q_o : (q \times q) \cap T_{\text{spec}} = \emptyset\}.$$

For high-order opacity, we need to further think from the intruder's point of view, i.e., how the intruder estimates the state estimate of the user. Therefore, we need to further build the observer automaton of $\text{Obs}_{\Sigma_o}(G)$ w.r.t. Σ_a, which is referred to as the *double-observer*, as follows

$$\text{Obs}_D(G) := \text{Obs}_{\Sigma_a}(\text{Obs}_{\Sigma_o}(G)) = (Q_D, \Sigma, f_D, q_{D,0})$$

Intuitively, the double-observer $\text{Obs}_D(G)$ tracks all the possible observer states in $\text{Obs}_o(G)$ based on another event set Σ_a. Also we can get that for any $s, t \in \mathcal{L}(G)$, we have $f_D(s) = f_D(t)$ if $P_a(s) = P_a(t)$ from the definition of the transition function of $\text{Obs}_D(G)$. In other word, the states of $\text{Obs}_D(G)$ are actually the current state estimate of observer $\text{Obs}_o(G)$ from the intruder's point of view. Using $\text{Obs}_D(G)$, we can easily check high-order opacity by the following theorem.

Theorem 1. System G is high-order opaque (w.r.t. T_{spec}, Σ_a and Σ_o) iff $\forall q \in Q_D : q \nsubseteq Q_{o,S}$.

The above theorem immediately suggests Algorithm 1 for verifying high-order opacity. Specifically, we need to build the doubly-observer and check whether or not it contains a state such that the knowledge predicate holds true for each element in it. In the worst-case, the doubly-observer contains $2^{2^{|X|}}$ states and $|\Sigma|2^{2^{|X|}}$ transitions. Therefore, the overall complexity of Algorithm 1 is doubly exponential in the size of G.

Algorithm 1 High-Order-Opa-Dou-Obs

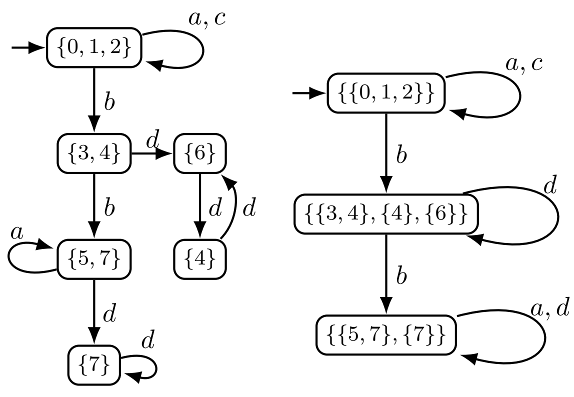
```
Input: $G$, $T_{\text{spec}}$, $\Sigma_a$, $\Sigma_o$
Output: High-order opaque or not
Build $\text{Obs}_o(G) = (Q_o, \Sigma, f_o, q_{o,0})$
Build $\text{Obs}_D(G) = (Q_D, \Sigma, f_D, q_{D,0})$
for all $p \in Q_D$ do
    if $p \subseteq Q_{o,S}$ then
        return $G$ is not high-order opaque
    end if
end for
return $G$ is high-order opaque
```

Although the worst-case complexity of Algorithm 1 is doubly-exponential, empirical studies show that, in many systems, the exponential state-space explosion in the observer construction does not really occur (Clavijo and Basilio, 2017). This is why we still choose to present this algorithm. We illustrate Algorithm 1 using the following example.

Example 2. Again, let us consider system G presented in Example 1. The observer $\text{Obs}_o(G)$ w.r.t. event set Σ_o is shown in Figure 3(a). Since we consider knowledge task $T_{\text{spec}} = \{(x, x') \in X \times X : x \neq x'\}$, for any string $s \in \mathcal{L}(G)$, we have $\texttt{Know}(P_o(s)) = \texttt{true}$ iff $f_o(q_{o,0}, s)$ is a singleton. Therefore, we have $Q_{o,S} = \{\{4\}, \{6\}, \{7\}\}$. Based on observer $\text{Obs}_o(G)$, we further build the double-observer $\text{Obs}_D(G)$ w.r.t. Σ_a as shown in Figure 3(b). For each state $q \in Q_D$ in it, we see that q always contains an element not in $Q_{o,S}$. Therefore, by Theorem 1, we conclude that G is high-order opaque. This conclusion is consistent with our analysis in Example 1.

4.2 State-Pair-Observer Approach

The double-observer approach uses the subset construction technique to capture information uncertainties for both the user and the intruder. However, since we use the subset construction twice, the double-observer is doubly-exponential in the size of the plant. Here, we note that, from the user's point of view, it is only interested in whether or not it can distinguish all pairs in T_{spec}. Therefore, from the intruder's point of view, it suffices to estimate the set of all state pairs the user cannot distinguish. This leads to the *state-pair-observer* defined as follows.



(a) $\text{Obs}_o(G)$ (b) $\text{Obs}_D(G)$

Fig. 3. Two observer automaton for G.

Definition 3. (State-Pair-Observer). Given DFA $G = (X, \Sigma, \delta, x_0)$, observable events of the user $\Sigma_o \subseteq \Sigma$ and observable events of the intruder $\Sigma_a \subseteq \Sigma$, the state-pair-observer is a new DFA

$$\text{Obs}_V(G) = (Q_V, \Sigma_a, f_V, q_{0,V}),$$

where

- $Q_V \subseteq 2^{X\times X} \setminus \emptyset$ is the set of states;
- Σ_a is the set of events;
- $f_V : Q_V \times \Sigma_a \to Q_V$ is the deterministic transition function defined by: for any $q \in Q_V, \sigma \in \Sigma_a$, we have
$$f_V(q, \sigma) = \left\{ (x'_1, x'_2) : \begin{array}{c} \exists (x_1, x_2) \in q, \exists w \in \Omega_{ua}(q, \sigma), \\ \exists w'_1, w'_2 \in \Sigma^* \text{ s.t.} \\ P_o(\sigma w) = P_o(w'_1) = P_o(w'_2) \text{ and} \\ (x'_1, x'_2) = (\delta(x_1, w'_1), \delta(x_2, w'_2)) \end{array} \right\},$$
where $\Omega_{ua}(q, \sigma)$ is the set of feasible unobservable strings defined by:
$$\Omega_{ua}(q, \sigma) = \{w \in \Sigma^*_{ua} : \exists (x, x) \in q \text{ s.t. } \delta(x, \sigma w)!\}.$$
- $q_{0,V}$ is the unique initial state defined by:
$$q_{0,V} = \left\{ (x'_1, x'_2) : \begin{array}{c} \exists w \in \Sigma^*_{ua} \cap \mathcal{L}(G), w'_1, w'_2 \in \mathcal{L}(G) \text{ s.t.} \\ P_o(w) = P_o(w'_1) = P_o(w'_2) \text{ and} \\ (x'_1, x'_2) = (\delta(w'_1), \delta(w'_2)) \end{array} \right\}$$

Intuitively, the state-pair-observer estimates from the intruder's point of view based on Σ_a. However, instead of estimating the state-estimate of the user as the case of double-observer, it estimates the set of state pairs that can not be distinguished by the user. Specifically, suppose that the current estimate of the intruder is $q \in 2^{X\times X}$, which means that it thinks that the user cannot distinguish state pair $(x_1, x_2) \in q$. Then by observing a new event $\sigma \in \Sigma_a$, the intruder needs to update its estimate by considering all strings with unobservable tail $w \in \Omega_{ua}(q, \sigma)$, where state pair $(x, x) \in q$ represents a possible actual state x of the system. Note that, for each actual string σw in the system, the user has its own observation according to Σ_o. Then for strings $w_1 \in \mathcal{L}(G, x_1)$ and $w_2 \in \mathcal{L}(G, x_2)$, if $P_o(w_1) = P_o(w_2) = P_o(\sigma w)$, then the user again cannot distinguish between state $\delta(x_1, w_1)$ and state $\delta(x_2, w_2)$.

The following result says that the proposed state-pair-observer indeed captures all pairs of states the user cannot distinguish based on the observation of the intruder.

Proposition 1. For any $s \in \mathcal{L}(G)$, the state reached by $P_a(s)$ in $\text{Obs}_V(G)$ satisfies the following:

$$f_V(q_{0,V}, P_a(s)) = \left\{ (\delta(w'_1), \delta(w'_2)) : \begin{array}{c} \exists t, w'_1, w'_2 \in \mathcal{L}(G), \\ \text{s.t. } P_a(s) = P_a(t) \text{ and} \\ P_o(t) = P_o(w'_1) = P_o(w'_2) \end{array} \right\}$$

Then for any observer state $q \in 2^{X\times X}$, we say q is a *secret revealing state* if any state pair $(x_1, x_2) \in q$ in it is not in T_{spec}, and we define

$$Q_{V,S} = \{q \in Q_V : q \cap T_{\text{spec}} = \emptyset\}$$

as the set of secret revealing states. Recall that each pair $(x_1, x_2) \in q$ is a pair of states the user *cannot* distinguish. Therefore, a secret revealing state means that the intruder knows for sure that the user *can* distinguish task T_{spec}; hence, the knowledge secret is revealed. This leads to the following main theorem.

Theorem 2. System G is high-order opaque (w.r.t. T_{spec}, Σ_a and Σ_o) iff $Q_{V,S} = \emptyset$.

The above theorem suggests immediately how to use state-pair-observer for verifying high-order opacity. The procedure is summarized in Algorithm 2, which is essentially a reachability search in $\text{Obs}_V(G)$. Note that there are at most $2^{|X|^2}$ states and $|\Sigma_a|2^{|X|^2}$ transitions in the state-pair-observer. Therefore, the worst-case complexity of Algorithm 2 is $O(|\Sigma_a|2^{|X|^2})$, which is single-exponential in the size of G. As we have discussed in Remark 3.4, verifying high-order complexity is at least PSPACE-hard since it subsumes the standard notion of current-state opacity. Then with this single-exponential upper-bound, we can further conclude that verifying high-order opacity is actually PSPACE-complete.

Algorithm 2 High-Order-Opa-Sta-Pai-Obs

```
Input: $G$, $T_{\text{spec}}$, $\Sigma_a$, $\Sigma_o$
Output: High-order opaque or not
Build $\text{Obs}_V(G) = (Q_V, \Sigma_a, f_V, q_{0,V})$
for all $q \in Q_V$ do
    if $q \in Q_{V,S}$ then
        return $G$ is not high-order opaque
    end if
end for
return $G$ is high-order opaque
```

Example 3. Let us still consider system G shown in Figure 2 with $T_{\text{spec}} = \{(x, x') \in X \times X : x \neq x'\}$. This state-pair-observer is shown in Figure 4. Initially, we have $P_a(\epsilon) = P_a(c) = \epsilon$. Note that we also have $P_o(\epsilon) = P_o(c) = \epsilon$ and $P_o^{-1}(\epsilon) \cap \mathcal{L}(G) = \{\epsilon, a, c\}$, i.e., the intruder thinks that user cannot distinguish states $\delta(\epsilon) = 0, \delta(a) = 1$ and $\delta(c) = 2$. Therefore, the 3×3 combinations of the state pairs give the initial-state in $\text{Obs}_V(G)$. Also, for example, consider string $s = cbdd$, where $P_a(s) = b$. We have $P_a^{-1}(P_a(s)) \cap \mathcal{L}(G) = \{cbd^n\}$. For strings cbd^n where $n \geq 1$, we have $P_o^{-1}(P_o(cbd^n)) \cap \mathcal{L}(G) = \{cbd^n\}$, i.e., the user can perfectly determine the state, and corresponding state pairs the user cannot distinguish are $(4, 4)$ and $(6, 6)$. For strings cb, we have $P_o^{-1}(P_o(cb)) \cap \mathcal{L}(G) = \{ab, cb\}$. Therefore, the state pairs the user cannot distinguish are $(3, 3), (4, 4), (3, 4)$ and $(4, 3)$. This is why we have

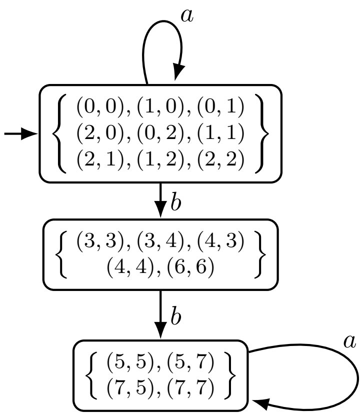

a
$\left\{\begin{matrix}(0,0),(1,0),(0,1)\\(2,0),(0,2),(1,1)\\(2,1),(1,2),(2,2)\end{matrix}\right\}$
b
$\left\{\begin{matrix}(3,3),(3,4),(4,3)\\(4,4),(6,6)\end{matrix}\right\}$
b
$\left\{\begin{matrix}(5,5),(5,7)\\(7,5),(7,7)\end{matrix}\right\}$
a

Fig. 4. State-pair-observer $Obs_V(G)$ for G.

$f_V(q_{0,V}, P_a(s)) = \{(3,3),(4,4),(3,4),(4,3),(6,6)\}$. Note that, for each state $q \in Q_V$, q contains a state pair (x_1, x_2) such that $x_1 \neq x_2$, which means that $q \cap T_{\text{spec}} \neq \emptyset$. Intuitively, this means that the intruder thinks that the user may not be able to distinguish between different states x_1 and x_2. Since $Q_{V,S} = \emptyset$, according to Theorem 2, we conclude that the system G is high-order opaque, which is also consistent with our previous analysis.

5. CONCLUSION

In this paper, we investigated information-flow security properties of partially-observed DES from a new angle by considering secret as the user's knowledge of some fact of importance. We proposed the new notion of high-order opacity to capture the knowledge security requirement, i.e., the intruder can never know that the user knows something for sure. We showed that the new notion of high-order opacity subsumes the standard notion of current-state opacity. Effective algorithms were also provided for verifying this new notion.

This paper makes the first step towards the framework of knowledge security and there are several on-going and future directions. First, in this paper, we essentially consider the *current knowledge* of the user. In some applications, however, what the user wants to hide may be the fact that he *knew something at some previous instant.* This is analogous to the difference between current-state opacity and infinite/K-step opacity. Also, this paper only considers the verification of high-order opacity. When the system is verified to be non-opaque, one may further want to enforce high-order opacity using supervisory control (Dubreil et al., 2010) or insertion functions (Ji et al., 2019). These problems are still under investigation.

REFERENCES

Balun, J. and Masopust, T. (2020). On opacity verification for discrete-event systems. *IFAC-PapersOnLine*, 53(2), 2075–2080.

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

Behinaein, B., Lin, F., and Rudie, K. (2019). Optimal information release for mixed opacity in discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 16(4), 1960–1970.

Bryans, J.W., Koutny, M., Mazaré, L., and Ryan, P.Y. (2008). Opacity generalised to transition systems. *International J. Information Security*, 7(6), 421–435.

Cassandras, C.G. and Lafortune, S. (2021). *Introduction to discrete event systems*. Springer.

Clavijo, L.B. and Basilio, J.C. (2017). Empirical studies in the size of diagnosers and verifiers for diagnosability analysis. *Discrete Event Dynamic Sys.*, 27(4), 701–739.

Dubreil, J., Darondeau, P., and Marchand, H. (2010). Supervisory control for opacity. *IEEE Transactions on Automatic Control*, 55(5), 1089–1100.

Ji, Y., Yin, X., and Lafortune, S. (2019). Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, 108, 108476.

Lafortune, S., Lin, F., and Hadjicostis, C.N. (2018). On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45, 257–266.

Lai, A., Lahaye, S., and Li, Z. (2021). Initial-state detectability and initial-state opacity of unambiguous weighted automata. *Automatica*, 127, 109490.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3), 496–503.

Liu, S. and Zamani, M. (2021). Compositional synthesis of opacity-preserving finite abstractions for interconnected systems. *Automatica*, 131, 109745.

Ma, Z., Yin, X., and Li, Z. (2021). Verification and enforcement of strong infinite-and k-step opacity using state recognizers. *Automatica*, 133, 109838.

Saboori, A. and Hadjicostis, C.N. (2011). Verification of infinite-step opacity and complexity considerations. *IEEE Trans. Automatic Control*, 57(5), 1265–1269.

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

Sears, D. and Rudie, K. (2014). On computing indistinguishable states of nondeterministic finite automata with partially observable transitions. In *53Rd IEEE Conference on Decision and Control*, 6731–6736. IEEE.

Tong, Y., Li, Z., Seatzu, C., and Giua, A. (2017). Verification of state-based opacity using petri nets. *IEEE Transactions on Automatic Control*, 62(6), 2823–2837.

Wang, W., Lafortune, S., and Lin, F. (2007). An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions. *Systems & Control Letters*, 56(9-10), 656–661.

Wintenberg, A., Blischke, M., Lafortune, S., and Ozay, N. (2022). A general language-based framework for specifying and verifying notions of opacity. *Discrete Event Dynamic Systems*, 1–37.

Yin, X. and Li, S. (2020). Synthesis of dynamic masks for infinite-step opacity. *IEEE Transactions on Automatic Control*, 65(4), 1429–1441.

Yin, X., Zamani, M., and Liu, S. (2021). On approximate opacity of cyber-physical systems. *IEEE Transactions on Automatic Control*, 66(4), 1630–1645.

Yin, X. and Lafortune, S. (2017). A new approach for the verification of infinite-step and k-step opacity using two-way observers. *Automatica*, 80, 162–171.

Yin, X., Li, Z., Wang, W., and Li, S. (2019). Infinite-step opacity and k-step opacity of stochastic discrete-event systems. *Automatica*, 99, 266–274.