Journal of Discrete Event Dynamical Systems manuscript No.
(will be inserted by the editor)

A General Language-Based Framework for Specifying and Verifying Notions of Opacity

Andrew Wintenberg · Matthew Blischke · Stéphane Lafortune · Necmiye Ozay

Received: date / Accepted: date

Abstract Opacity is an information flow property that captures the notion of *plausible deniability* in dynamic systems, that is whether an intruder can deduce that "secret" behavior has occurred. In this paper we provide a general framework of opacity to unify the many existing notions of opacity that exist for discrete event systems. We use this framework to discuss language-based and state-based notions of opacity over automata. We present several methods for language-based opacity verification, and a general approach to transform state-based notions into language-based ones. We demonstrate this approach for current-state and initial-state opacity, unifying existing results. We then investigate the notions of K-step opacity. We provide a language-based view of K-step opacity encompassing two existing notions and two new ones. We then analyze the corresponding language-based verification methods both formally and with numerical examples. In each case, the proposed methods offer significant reductions in runtime and space complexity.

Keywords Opacity · Verification · Language-Based Opacity · K-step Opacity

Research supported in part by US NSF under grants CNS-1738103, CNS-1801342, and ECCS-1553873.

A. Wintenberg
E-mail: awintenb@umich.edu
M. Blischke
E-mail: matblisc@umich.edu
S. Lafortune
E-mail: stephane@umich.edu
N. Ozay
E-mail: necmiye@umich.edu
Department of EECS, University of Michigan,
1301 Beal Avenue, Ann Arbor, MI 48109-2122, USA

1 Introduction

As modern systems become increasingly connected, information flow has become critical to their correct operation. These systems have entered many areas of life in the form of autonomous vehicles, the smart grid, location-based services, and medical monitoring, to name but a few areas. The increasing amount of physical and human interaction with these systems raises concerns over security and privacy. Transmission of information across networks possesses an inherent risk of revealing private information to an outside observer called the *intruder*, potentially with malicious intent. Formal modeling of information flow properties has been proposed as a way to understand and manage these risks in networked dynamic systems. Notions like non-interference [13] and anonymity [20] were developed in the computer science community for this purpose.

More recently, the notion of *opacity* [19] was proposed as a general information flow property capturing the notion of "plausible deniability": opacity holds if an intruder cannot deduce sensitive information from their observations of a system's behavior. Opacity was further developed for a variety of Discrete Event System (DES) models, including transition systems [4] [1], finite state automata [21], Petri nets [3] [26], timed automata [6], modular automata [18], and more. Within these models, many notions of opacity have been proposed to capture different forms of private or secret information. Of particular importance are language-based opacity [17], current-state opacity [21], initial-state opacity [22], and the related notions of K-step and infinite step opacity [23,24]. In addition to the type of private information, the capabilities of the intruder are also integral to notions of opacity. While many works in DES consider a single intruder with static observations of observable events, more complex observation schemes have also been considered, such as decentralized observers in [29] or dynamic observers in [7]. Opacity is an expressive notion of security. Many existing security properties, including non-interference and anonymity, can be formulated as opacity [14]. Additionally, opacity has been utilized in practical applications, like the enforcement of privacy in language-based services [32]. For a thorough review of works in opacity in the context of DES, as of 2016, please see [15].

Although a variety of notions of opacity have been proposed, they may not directly capture the desired notion of privacy or security in a given networked system. One approach to analyzing specific notions of opacity is to transform them into existing notions where existing methods can be applied. While some transformations between the various forms of opacity over automata have been studied (for example between current-state, initial-state, language-based [29]), it is unclear if other notions like K-step opacity are comparable or how to handle new notions. The first contribution of this paper is to develop a systematic approach for specifying and analyzing various notions of opacity. This is accomplished with a general definition of opacity extending the notion developed for transition systems [4]. We use this framework to model language-based opacity over automata and present several methods for verifi-

cation thereof. Then we develop a general transformation between state-based and language-based notions of opacity. Using this, state-based notions of opacity can be described by constructing automata to specify secret behavior and verified using language-based methods. This approach is first demonstrated on the simple notions of current-state and initial-state opacity. The resulting verification methods resemble the existing standard approaches for verification of these forms of opacity.

The second contribution of this paper is to apply the proposed framework and verification methods to the less well-understood notions of K-step and infinite step opacity. Whereas current-state opacity only considers an intruder's current state estimate, K-step and infinite step opacity may involve the intruder *smoothing* their estimates, i.e., improving estimates of the past with current information. While it may appear that these notions are incomparable [33], we provide a unified view of two prominent existing notions of K-step opacity along with two new ones that emerge using our framework. These notions are then transformed into language-based and hence current-state opacity. Furthermore, the resulting language-based verification methods offers considerable advantages over existing methods. We demonstrate this both formally and with numerical examples.

The remaining sections of this paper are organized as follows. Section 2 presents a general behavioral definition of opacity. Section 3 reviews finite automata and discusses language-based and state-based opacity over them along with methods for verification. Section 4 applies these concepts to verifying current-state and initial-state opacity. Section 5 defines K-step and infinite step opacity in relation to existing notions. Section 6 presents methods for verification of K-step opacity while Section 7 discusses the complexity of these methods. Section 8 discusses verification of infinite step opacity. Section 9 presents numerical results comparing verification methods for K-step opacity. Finally, Section 10 concludes the paper.

2 A general framework for opacity

In this section we present a general framework of opacity to formalize the intuition of a system having "plausible deniability". In order to unify the different notions of opacity that exist for a variety of system models, we discuss systems in terms of their *behavior*, taking the approach of [27]. Consider a system under observation by an intruder. We denote the set of possible behaviors or runs of the system as R. For example, R may be the set of solutions to a differential equation modeling a continuous-time system or R may be the language of an automaton modeling a discrete event system. The intruder makes observations of this behavior in the space O through an observation map $\Theta : R \to O$. Opacity describes the inability of the intruder to discern a class of secret runs $R_{\mathrm{S}} \subseteq R$ from a class of nonsecret runs $R_{\mathrm{NS}} \subseteq R$. This inability can either be total or partial. In the following definitions we extend Θ to sets in the standard way, i.e., $\Theta(R) = \{\Theta(r) \mid r \in R\}$.

Definition 1 We say that $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is *totally opaque* to $\Theta : R \to O$ if

$$\Theta(R_{\mathrm{S}}) \subseteq \Theta(R_{\mathrm{NS}})\,. \tag{1}$$

Definition 2 We say that $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is *partially opaque* to $\Theta : R \to O$ if

$$\Theta(R_{\mathrm{S}}) \cap \Theta(R_{\mathrm{NS}}) \neq \emptyset\,. \tag{2}$$

When the behavior is taken to be the runs of a transition system, total opacity corresponds to the notion of opacity given in [4].

Specific notions of opacity correspond to different specifications of the secret and nonsecret runs of the system and capabilities of the intruder. In this work, we focus on total opacity as it relates to desirable notions of privacy and security, e.g., all secrets are hidden. Alternatively, partial opacity can express notions like *diagnosability* [17], e.g., faults can be detected. While the secret and nonsecret behavior can be arbitrary sets, we often consider them to be complements. That is to say that nonsecret behavior means behavior which is not secret $R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}$. In this case observe the following.

Observation 1 If $R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}$ then $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is totally opaque if and only if (R, R_{NS}) is totally opaque. This is because

$$\Theta(R_{\mathrm{S}}) \subseteq \Theta(R_{\mathrm{NS}}) \;\Leftrightarrow\; \Theta(R) = \Theta(R_{\mathrm{S}}) \cup \Theta(R_{\mathrm{NS}}) \subseteq \Theta(R_{\mathrm{NS}})\,. \tag{3}$$

So under this condition, it suffices to consider only R_{NS} and R. ◊

2.1 Joint & separate opacity

More complex notions of privacy can involve multiple classes of possibly overlapping secret behaviors. Consider a set of pairs of classes of secret and nonsecret behaviors $\{R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)\}_{i\in I}$ over an index set I. We consider two forms of opacity over these pairs with respect to an observation map Θ.

Definition 3 We say that $\{R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)\}_{i\in I}$ is *jointly opaque* to Θ if

$$\left(\bigcup_{i\in I} R_{\mathrm{S}}(i), \bigcap_{i\in I} R_{\mathrm{NS}}(i)\right) \text{ is totally opaque.} \tag{4}$$

Joint opacity considers all secrets uniformly. It requires that a run in one secret class can be explained by a run that is nonsecret in every class.

Definition 4 We say that $\{R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)\}_{i\in I}$ is *separately opaque* to Θ if

$$\forall i \in I,\ (R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)) \text{ is totally opaque.} \tag{5}$$

Separate opacity considers all secrets individually. It requires that a run in one secret class can be explained by a run that is nonsecret in that class, but perhaps secret in another class.

Observation 2 When $|I| = 1$, joint and separate opacity reduce to total opacity. When $|I| \geq 1$, joint opacity implies separate opacity. For $I' \subseteq I$, joint (separate) opacity of $\{R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)\}_{i\in I}$ implies joint (separate) opacity of $\{R_{\mathrm{S}}(i), R_{\mathrm{NS}}(i)\}_{i\in I'}$, respectively. ◊

3 Opacity over automata

Automata are a widely used model in discrete event systems. There are many existing notions of opacity for automata which capture different privacy and security properties. We can express these notions in the framework presented in Section 2 as total opacity with appropriate choices of secret and nonsecret behavior and of the intruder. When secret and nonsecret behaviors are given as languages marked by automata, we refer to this as *language-based opacity*. More generally, when secret and nonsecret behaviors are defined in terms of the automaton's events and properties of the states we refer to this as *state-based opacity*. For example, many state-based notions involve visits to states designated as secret or nonsecret. It is known that some state-based notions of opacity like current-state and initial-state opacity can be efficiently transformed into language-based opacity as in [29]. In this section, we first review automata theory then discuss language-based opacity in the framework of Section 2 along with corresponding methods for verification. We then develop a general transformation from state-based to language-based notions of behavior. With this transformation, we describe how state-based opacity can be verified using language-based methods.

3.1 Automata Review

Given a finite set of events E, we denote the set of finite strings over E as E^* including the empty string ϵ. For a string $s \in E^*$, we denote the length of s as $|s|$ and write $s = s_0 \cdots s_{|s|-1}$. A language $L \subseteq E^*$ is a subset of strings. A nondetermistic finite automaton (NFA) is defined by a tuple $G = (Q, E, f, Q_0, Q_m)$ with a finite set of states Q, events E, transition function $f : Q \times E \to 2^Q$, initial states Q_0 and marked states Q_m. A deterministic finite automaton (DFA) is an NFA G such that $|Q_0| = 1$ and for all $q \in Q$ and $e \in E$ it holds that $|f(q, e)| \leq 1$. Unless stated otherwise, the term automaton will refer to an NFA. We also extend f to the domain $Q \times E^*$ in the standard way by

$$f(q, \epsilon) = q, \quad f(q, se) = f(f(q, s), e)\,. \tag{6}$$

For arbitrary sets $Q'_0, Q'_m \subseteq Q$, we define the language of G starting in Q'_0 and marked by Q'_m as

$$\mathcal{L}_{Q'_m}(G, Q'_0) = \{s \in E^* \mid \exists q_0 \in Q'_0\ \exists q_m \in Q'_m\ q_m \in f(q_0, s)\}\,. \tag{7}$$

Then the language generated by G is defined $\mathcal{L}(G) = \mathcal{L}_Q(G, Q_0)$ and the language marked by G is defined $\mathcal{L}_m(G) = \mathcal{L}_{Q_m}(G, Q_0)$. We call a language marked by an automaton a *regular language*.

We now present several constructions using automata. For more details on these constructions, see [5]. Given a deterministic automaton G we construct the complement automaton G^c by inverting the marking of G, adding a marked "dead" state, and completing any missing transitions in G to this dead state. It then holds that $\mathcal{L}_m(G^c) = E^* \setminus \mathcal{L}_m(G)$.

Given an automaton G we also construct the reversal G^R of G by swapping the initial and marked states and reversing all transitions, i.e., $q' \in f(q,e)$ if only if in the reverse $q \in f^R(q',e)$. Note the reversal of a deterministic automaton may be nondeterministic. The result G^R marks and generates the reversals of the languages of G where the reversal L^R of a language L is defined

$$L^R = \{s_n s_{n-1} \cdots s_0 \mid s_0 \cdots s_n \in L\}\,. \tag{8}$$

Given a nondeterministic automaton G we construct the determinization $\det(G) = (\overline{Q}, E, \overline{f}, \overline{Q}_0, \overline{Q}_m)$ where $\overline{Q} = 2^Q$, $\overline{Q}_0 = \{Q_0\}$, $\overline{Q}_m = \{\overline{q} \in \overline{Q} \mid \overline{q} \cap Q_m \neq \emptyset\}$ and

$$\forall \overline{q} \in Q\ \overline{e} \in E,\ \overline{f}(\overline{q}, e) = \bigcup_{q \in \overline{q}} f(q,e)\,. \tag{9}$$

The result $\det(G)$ marks and generates the same languages as G and is deterministic. We refer to this as the power set construction. Given automata $G = (Q_G, E, f_G, Q_{G,0}, Q_{G,m})$ and $H = (Q_H, E, f_H, Q_{H,0}, Q_{H,m})$, we construct the product automaton $G \times H = (Q_G \times Q_H, E, f_\times, Q_{G,0} \times Q_{H,0}, Q_{G,m} \times Q_{H,m})$ where $f_\times((q_G, q_H), e) = f_G(q_G, e) \times f_H(q_H, e)$. The result $G \times H$ then marks and generates the intersection of the languages of G and H.

In this work, we also consider nondeterministic automata with ϵ-transitions where transitions may also be labeled with the empty string ϵ. The above constructions can also be performed these automata with slight modification. For example see [5].

Using the framework from Section 2, we consider systems whose behavior $R \subseteq E^*$ can be represented as regular languages over the events E. In this case, we can model the system as a finite automaton. We consider observation maps $\Theta : R \to O$ that map system behavior to a string of observations in a finite set Γ with $O = \Gamma^*$. Furthermore, this map should preserve regularity. A class of such maps simply replace each event in E with an event in Γ or the empty string.

Definition 5 A *static mask* over $R \subseteq E^*$ is a mapping $\Theta : R \to \Gamma^*$ that satisfies

1. $\Theta(\epsilon) = \epsilon$,
2. $\forall s = s_0 \cdots s_n \in R,\ \Theta(s) = \Theta(s_0) \cdots \Theta(s_n)$.

Any function $\Theta : E \to \Gamma \cup \{\epsilon\}$ can be uniquely made into a static mask over $R \subseteq E^*$ by concatenation.

For example, given a set of observable events $E_o \subseteq E$, the natural projection P_{E_o} is a static mask defined by $P_{E_o}(e) = e$ for $e \in E_o$ and $P_{E_o}(e) = \epsilon$ for $e \notin E_o$. Given an automaton G, we can construct an automaton that marks $\Theta(\mathcal{L}_m(G))$ which in a slight abuse of notation we denote as $\Theta(G)$. This is done by simply replacing the events labeling transitions in G with their observation through Θ. Note, this may result in an automaton with ϵ-transitions. More general regularity-preserving observation maps and similar constructions are described in [17] but are not considered here.

3.2 Language-based opacity

In the context of language-based opacity, we model the behavior of a system as a regular language $R \subseteq E^*$ (not necessarily prefix-closed) marked by a finite automaton G. The secret and nonsecret behaviors of this system are given as regular sublanguages $R_{\mathrm{S}}, R_{\mathrm{NS}} \subseteq R$. We consider observation maps given by a static mask $\Theta : R \to O$ with $O = \Gamma^*$.

Definition 6 Given an automaton G, languages $R_{\mathrm{S}}, R_{\mathrm{NS}} \subseteq \mathcal{L}_m(G)$, and a static mask Θ, we say G is *language-based opaque* with respect to Θ, R_{S}, and R_{NS} if $\Theta(R_{\mathrm{S}}) \subseteq \Theta(R_{\mathrm{NS}})$, or equivalently $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is totally opaque to Θ.

This definition corresponds to the notion of strong opacity in [17] and language-based opacity in [29]. As $\Theta(R_{\mathrm{S}})$ and $\Theta(R_{\mathrm{NS}})$ are regular, language-based opacity is equivalent to a regular language containment. For many existing notions of opacity, the nonsecret behavior is simply behavior that is not secret, i.e. $R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}$. In this case, it is convenient to define $R_{\mathrm{NS}} = R \cap L_{\mathrm{NS}}$ using a regular nonsecret specification language L_{NS} over E. The language L_{NS} is specified by an automaton H_{NS} such that $L_{\mathrm{NS}} = \mathcal{L}_m(H_{\mathrm{NS}})$ and so $R_{\mathrm{NS}} = \mathcal{L}_m(G \times H_{\mathrm{NS}})$. Using Observation 1, we see that language-based opacity of G is equivalent to the regular language containment

$$\Theta(R) \subseteq \Theta(R_{\mathrm{NS}}), \text{ where } \Theta(R) = \mathcal{L}_m(\Theta(G)),\ \Theta(R_{\mathrm{NS}}) = \Theta(G \times H_{\mathrm{NS}})\,. \quad (10)$$

3.3 Verification of language-based opacity

By expressing language-based opacity as the well-studied problem of regular language containment, we can leverage existing techniques to verify opacity. We present three methods to check this language containment.

As input, the following methods take an automaton $G = (Q, E, f, Q_0, Q_m)$ modeling the system, a nonsecret specification automaton $H_{\mathrm{NS}} = (Q_{\mathrm{NS}}, E, f_{\mathrm{NS}}, Q_{\mathrm{NS},0}, Q_{\mathrm{NS},m})$, and a static mask $\Theta : R \to \Gamma^*$ where $R = \mathcal{L}_m(G)$. These methods verify the total opacity of $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ to Θ where $R_{\mathrm{NS}} = \mathcal{L}_m(G \times H_{\mathrm{NS}})$ and $R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}$. This is done by verifying the equivalent containment of equation (10).

Approach 1 (Forward Comparison) A standard approach for verifying language containment utilizes the following equivalence:

$$\Theta(R) \subseteq \Theta(R_{\mathrm{NS}}) \ \Leftrightarrow\ \Theta(R) \cap \Theta(R_{\mathrm{NS}})^c = \emptyset\,. \quad (11)$$

We construct $G_{FC} = \Theta(G) \times \det(\Theta(G \times H_{\mathrm{NS}}))^c$ so that $\mathcal{L}_m(G_{FC}) = \Theta(R) \cap \Theta(R_{\mathrm{NS}})^c$. Note determinization is required to construct the complement as $\Theta(G \times H_{\mathrm{NS}})$ is nondeterministic in general. Hence $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is totally opaque if and only if G_{FC} marks the empty language. We then verify opacity by ensuring G_{FC} contains no reachable, marked state. $\Diamond$

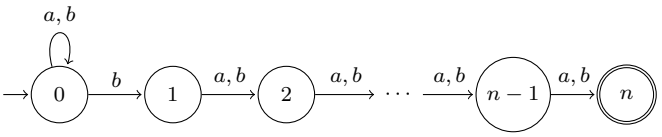

Fig. 1 An automaton G_n with $n+1$ states. The forward determinization of $\det(G_n)$ has 2^n+1 states while the reverse determinization $\det(G_n^R)$ has only $n+1$ states.

Approach 2 (Reverse Comparison) Instead of directly checking the language containment, note that containment of languages is equivalent to containment of the reversed languages, therefore:

$$\Theta(R) \subseteq \Theta(R_{\mathrm{NS}}) \;\Leftrightarrow\; \Theta(R)^R \subseteq \Theta(R_{\mathrm{NS}})^R \,. \tag{12}$$

Similar to the forward comparison method, we can construct $G_{RC} = \Theta(G)^R \times \det(\Theta(G \times H_{\mathrm{NS}})^R)^c$ so that $\mathcal{L}_m(G_{RC}) = \Theta(R)^R \cap (\Theta(R_{\mathrm{NS}})^R)^c$. We then verify opacity by ensuring G_{RC} contains no reachable, marked state. For some forms of opacity, reverse comparison significantly outperforms forward comparison. This is possible because there are automata whose determinizations are exponentially larger than the determinizations of their reverses. For example consider the automaton depicted in Figure 1. ◊

We can simplify the verification procedure by making assumptions on the structure of H_{NS}. Suppose that H_{NS} is a *complete automaton*, i.e., $\mathcal{L}(H_{\mathrm{NS}}) = E^*$. Note that a given H_{NS} can be made to satisfy this by adding at most one state without affecting its marked language. In this case $G \times H_{\mathrm{NS}}$ will encode both R and R_{NS} with different sets of marked states. With this observation, we can construct a deterministic finite automaton $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS}}))$ called the *secret observer* which marks nonsecret observations. With this automaton we can verify opacity using the following result.

Proposition 1 Suppose that $\mathcal{L}(H_{\mathrm{NS}}) = E^*$. Using the power set construction, define $G_{SO} = (\overline{Q}, \Gamma, \overline{f}, \{\overline{q}_0\}, \overline{Q}_m)$ where $\overline{Q} \subseteq 2^Q$ and $\overline{q}_0 = Q_0$ so that $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS}}))$. Then $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is totally opaque to Θ if and only if for all $\gamma \in \mathcal{L}(G_{SO})$ it holds for $\overline{q} = \overline{f}(\overline{q}_0, \gamma)$ that

$$\overline{q} \cap (Q_m \times Q_{\mathrm{NS}}) = \emptyset \;\vee\; \overline{q} \cap (Q_m \times Q_{\mathrm{NS},m}) \neq \emptyset \,. \tag{13}$$

Proof First note as $\mathcal{L}(H_{NS}) = E^*$, it holds $\mathcal{L}(G \times H_{NS}) = \mathcal{L}(G) \supseteq R$. Hence

$$\mathcal{L}(G_{SO}) = \mathcal{L}(\Theta(G \times H_{\mathrm{NS}})) = \Theta(\mathcal{L}(G \times H_{NS})) \supseteq \Theta(R) \,. \tag{14}$$

For $\gamma \in \mathcal{L}(G_{SO})$ let $\overline{q} = \overline{f}(\overline{q}_0, \gamma)$. By the construction of G_{SO}, note that

$$\exists q \in \overline{q} \cap Q_m \times Q_{\mathrm{NS}} \;\Leftrightarrow\; \exists r \in \mathcal{L}_m(G) \cap \mathcal{L}(H_{\mathrm{NS}}) = R \;\wedge\; \Theta(r) = \gamma \,. \tag{15}$$

Likewise, note that

$$\exists q \in \overline{q} \cap (Q_m \times Q_{\mathrm{NS},m}) \;\Leftrightarrow\; \exists r \in \mathcal{L}_m(G) \cap \mathcal{L}_m(H_{\mathrm{NS}}) = R_{\mathrm{NS}} \wedge \Theta(r) = \gamma \,. \tag{16}$$

Hence the state $\overline{q}$ satisfies the conditions in (13) if and only if $\gamma \in \Theta(R_{\mathrm{NS}})$ or $\gamma \notin \Theta(R)$. Combining these facts with equation (14) yields the result. □

We use this result in the following approach.

Approach 3 (Secret Observer) Given $\mathcal{L}(H_{\mathrm{NS}}) = E^*$, construct the secret observer $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS}}))$. Using Proposition 1 we verify opacity by checking that every reachable state of G_{SO} satisfies the conditions in (13). $\Diamond$

In each of these approaches, we verify opacity by constructing an automaton G_{FC}, G_{RC}, or G_{SO} and checking if each of its reachable states satisfies a given property. As these are the largest automata constructed in these approaches, we quantify the complexity of these approaches in terms of the number of states in these automata. We can improve these methods by incrementally constructing the reachable part of these automata and terminate if a violating state is found.

Remark 1 It is well-known that checking the containment of languages represented by nondeterministic automata, as required here, is PSPACE-complete [25]. We can use this to establish the known result that verification of language-based opacity in this setting is also PSPACE-complete. This provides a lower bound on the complexity of the proposed approaches for general automata. Still in practice, we may choose one approach over another based upon the specific structure of H_{NS} for a fixed notion of opacity. For example consider the following.

When the secret observer method is applicable, i.e., $\mathcal{L}(H_{\mathrm{NS}}) = E^*$, the complexity of the secret observer method is always no worse than the complexity of the forward comparison method. This is because both approaches require the construction of the automaton $\det(\Theta(G \times H_{\mathrm{NS}}))$, while this is all that is required for the secret observer method. So for a given H_{NS} satisfying $\mathcal{L}(H_{\mathrm{NS}}) = E^*$, we do not consider the forward comparison method. It is possible that a lower complexity could be obtained by a different choice of H_{NS} with $\mathcal{L}(H_{\mathrm{NS}}) \neq E^*$. $\Diamond$

3.4 Transforming state-based behavior

We now discuss state-based notions of opacity in the framework of Section 2. Whereas in language-based opacity secret and nonsecret behaviors are defined solely in terms of the events, in state-based opacity these behaviors are defined in terms of both events and properties of the states visited in the automaton. As many existing notions of state-based opacity implicitly assume prefix-closed behavior, we consider a system modeled by an automaton $\mathcal{A} = (X, \Sigma, \delta, X_0)$ without marked states in the context of state-based opacity. Here we use the convention that automata denoted by $\mathcal{A}$ are used to represent state-based behavior, while automata denoted by G and H more generally are used to represent languages. We can express state-based opacity in the framework of Section 2 by identifying the relevant behavior of the automaton.

We consider when the secret behavior is defined by labels A on the states assigned by a map $\ell : X \to A$. Viewing the events as inputs and state labels

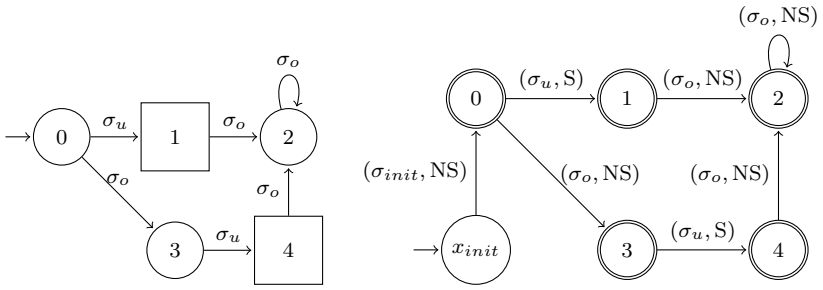

Fig. 2 On the left, an automaton $\mathcal{A}$ is depicted. The labeling function ℓ is defined by labeling square states as S and round states as NS (so $A = \{\S, \mathrm{NS}\}$). On the right, the automaton $G = T^{IO}(\mathcal{A}, \ell)$ is depicted.

as outputs, we can describe the runs of this system as input-output sequences. In this view, the system $(\mathcal{A}, \ell)$ is sometimes referred to as a *Moore machine* [5]. We write these runs as sequences of pairs of an input (event) and the resulting output (state label). To do this we introduce an artificial event σ_{init} representing the system turning on to be paired with the label of the initial state. For example consider the automaton $\mathcal{A}$ depicted in Figure 2. The run starting at state 0 labeled NS, transitioning with event σ_u to state 1 labeled S, then transitioning with event σ_o to state 2 labeled NS, would be represented as $r = (\sigma_{init}, \mathrm{NS})(\sigma_u, \mathrm{S})(\sigma_o, \mathrm{NS})$. In this way, we see the state-based behavior of the system can be written as a regular language marked by an automaton. This automaton is constructed by augmenting events with state labels and introducing an artificial initial state.

Definition 7 Given $\mathcal{A} = (X, \Sigma, \delta, X_0)$ and $\ell : X \to A$, we define the *label-transform* of $\mathcal{A}$ by $T^{IO}(\mathcal{A}, \ell) = (Q, E, f, Q_0, Q_m)$, where $Q = X \cup \{x_{init}\}$, $E = (\Sigma \cup \{\sigma_{init}\}) \times A$, $Q_0 = \{x_{init}\}$, $Q_m = X$, and nonempty transitions defined by

$$\begin{aligned} \forall a \in A, \quad & f(x_{init}, (\sigma_{init}, a)) = \{x_0 \in X_0 \mid \ell(x_0) = a\}, \\ \forall x \in X,\ \forall \sigma \in \Sigma,\ \forall a \in A, \quad & f(x, (\sigma, a)) = \{x' \in \delta(x, \sigma) \mid \ell(x') = a\}. \end{aligned} \tag{17}$$

An example of this transformation is depicted in Figure 2. Note that this transformation adds only a single state to $\mathcal{A}$ and allows us to make the following definition.

Definition 8 We define the set of *input-output sequences of* $\mathcal{A}$ *under* ℓ as $\mathcal{L}^{IO}(\mathcal{A}, \ell) = \mathcal{L}_m(T^{IO}(\mathcal{A}, \ell))$.

We then consider the behavior of $\mathcal{A}$ under ℓ to be $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$. In this way, we can specify and verify state-based notions of opacity over one automaton as language-based notions over another.

3.5 Specification and verification of state-based opacity

We can express existing state-based notions of opacity over an automaton $\mathcal{A}$ with state labeling map ℓ as total opacity over the input-output behavior $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$ with respect to some secret and nonsecret behaviors $R_{\mathrm{S}}, R_{NS} \subseteq R$ and a static mask $\Theta : R \to \Gamma^*$. We assume that the secret and nonsecret runs are specified as in the language-based setting.

Assumption 1 There exists a nonsecret specification automaton H_{NS} such that

$$L_{\mathrm{NS}} = \mathcal{L}_m(H_{\mathrm{NS}}), \quad R_{\mathrm{NS}} = R \cap L_{\mathrm{NS}}, \quad R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}\,. \tag{18}$$

Such specification automata H_{NS} for current-state and initial-state opacity are presented in Section 4.

Remark 2 Nonsecret behavior could also be specified with a temporal logic formula ϕ_{NS} with appropriate semantics. From ϕ_{NS}, the finite automaton H_{NS} marking runs that satisfy ϕ_{NS} could be synthesized. In this way, opacity can be viewed as a temporal logic hyperproperty [8]. ◊

Additionally, the state-based notions we consider model observation as projection of strings with respect to a set of observable events $\Sigma_o \subseteq \Sigma$. As such we only consider the observation map over the input-output sequences induced by this projection. By convention we will consider σ_{init} to be observable, i.e., the intruder observes when the system turns on. In this case we make the following assumption

Assumption 2 The intruder observes only occurrences of observable events $\Sigma_o \subseteq \Sigma$. This induced observation map is then a static mask defined by $\Theta : R \to \Gamma^*$ where $\Gamma = \Sigma_o \cup \{\sigma_{init}\}$ and $\Theta((\sigma, a)) = \sigma$ if $\sigma \in \Sigma_o \cup \{\sigma_{init}\}$ and $\Theta((\sigma, a)) = \epsilon$ otherwise.

Remark 3 Although not done here, one could also consider partially observable state outputs by defining an appropriate static mask over the event and state labels. ◊

Under these assumptions, notions of state-based opacity are specified by a nonsecret specification automaton H_{NS} and set of observable events Σ_o. *We will consider this setting in the remainder of this work.* We can then apply any of the language-based approaches of Section 3.3 to $G = T^{IO}(\mathcal{A}, \ell)$, H_{NS}, and the observation map Θ induced by Σ_o to verify the total opacity of $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ to Θ. This procedure is summarized in Figure 3. Due to the structure of G resulting from the transformation T^{IO}, the secret observer method has a simple interpretation.

Theorem 1 *The pair (R_S, R_{NS}) is totally opaque to Θ if every non-initial state of $G_{SO} = \det(\Theta(G \times H_{NS}))$ is marked.*

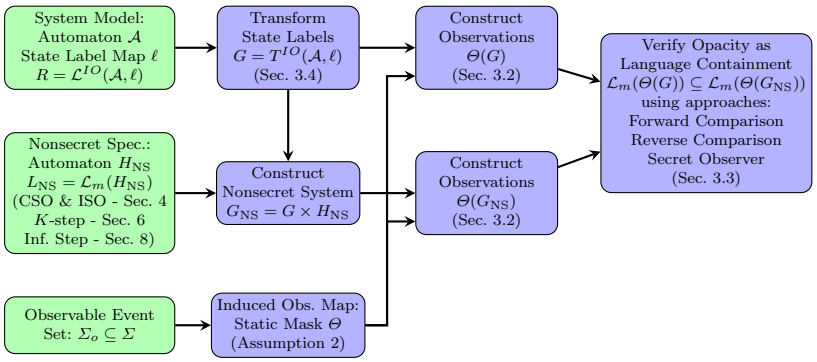

Fig. 3 The proposed method for verifying state-based opacity by transforming to language-based opacity.

Proof By construction, every non-initial state of G is marked. Hence the conditions in (13) hold exactly when a secret observer state is the initial state or contains pair of states marked in $G \times H_{\mathrm{NS}}$, i.e., the secret observer state is marked. So by Proposition 1, total opacity holds if every non-initial state of G_{SO} is marked. □

4 Current-state and initial-state opacity

Several existing notions of opacity used in discrete event systems define secret behavior in terms of secret states of automata. By viewing the secrecy of a state as a state output, we can express these notions as language-based opacity over the label-transform of the automaton. To verify these notions, we can then apply any of the language-based methods. In this section, we consider current-state opacity (CSO) and initial-state opacity (ISO). Although it is known that these notions can be transformed into language-based opacity [29], we include this discussion to demonstrate our transformation and provide insight into application to more complex state-based notions of opacity. In a sense, this work generalizes and systematizes the transformations of [29].

4.1 Labeling secret states

Consider an automaton $\mathcal{A} = (X, \Sigma, \delta, X_0)$ with a subset of states $X_{\mathrm{S}} \subseteq X$ designated as secret and observable events $\Sigma_o \subseteq \Sigma$. We also define the nonsecret states as $X_{\mathrm{NS}} = X \setminus X_{\mathrm{S}}$. This property of the states can be represented by labeling secret states with S and other nonsecret states as NS. So we define the set of labels $A = \{\mathrm{S}, \mathrm{NS}\}$ and labeling map $\ell : X \to A$ by

$$\ell(x) = \begin{cases} S, & x \in X_{\mathrm{S}} \\ \mathrm{NS}, & x \in X_{\mathrm{NS}} \end{cases} \tag{19}$$

A visit to a secret state in $\mathcal{A}$ following an event $\sigma \in \Sigma$ corresponds to the input-output pair $e = (\sigma, \mathrm{S})$. Likewise starting in a secret state in $\mathcal{A}$ corresponds to the pair $e = (\sigma_{init}, \mathrm{S})$. Using this observation, we define the set of secret and nonsecret input-output pairs as

$$E_{\mathrm{S}} = (\Sigma \cup \{\sigma_{init}\}) \times \{S\} \qquad E_{\mathrm{NS}} = (\Sigma \cup \{\sigma_{init}\}) \times \{\mathrm{NS}\}\,. \tag{20}$$

These sets can be used to specify the secret and nonsecret behavior in terms of the input-output sequences $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$ for CSO and ISO.

4.2 Current-state opacity (CSO)

First we consider current-state opacity. Current state opacity describes the inability of an intruder to deduce that the current state of the system is secret. It can be defined as follows.

Definition 9 (Current-State Opacity [12]) An automaton $\mathcal{A} = (X, \Sigma, \delta, X_0)$ is said to be *current-state opaque* with respect to the secret states $X_{\mathrm{S}} \subseteq X$ and observable events $\Sigma_o \subseteq \Sigma$ if

$$\begin{aligned} &\forall x_0 \in X_0\ \forall s \in \mathcal{L}(\mathcal{A})\ s.t.\ \exists x_{\mathrm{S}} \in \delta(x_0, s) \cap X_{\mathrm{S}}, \\ &\exists x_0' \in X_0\ \exists s' \in \mathcal{L}(\mathcal{A}),\ P_{\Sigma_o}(s) = P_{\Sigma_o}(s')\ \wedge \exists x_{\mathrm{NS}} \in \delta(x_0', s') \cap X \setminus X_{\mathrm{S}}\,. \end{aligned} \tag{21}$$

In words, runs of $\mathcal{A}$ ending with a visit to a secret state should look like a run ending with a visit to a nonsecret state. In terms of input-output sequences, this definition divides the behavior $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$ into secret and nonsecret behavior $R_{\mathrm{S}}, R_{\mathrm{NS}} \subseteq R$ defined by

$$L_{\mathrm{NS}} = E^* E_{\mathrm{NS}}, \quad R_{\mathrm{NS}} = R \cap L_{\mathrm{NS}}, \quad R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}\,, \tag{22}$$

where E_{NS} is defined in equation (20). We can use the nonsecret specification automaton H_{NS} depicted in Figure 4 with $\mathcal{L}_m(H_{\mathrm{NS}}) = L_{\mathrm{NS}}$ so that Assumption 1 is satisfied. Then using the observation map $\Theta : R \to \Gamma^*$ induced by the observable events Σ_o as defined in Assumption 2, we can see that $\mathcal{A}$ is current-state opaque if and only if $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ is totally opaque with respect to Θ. Hence we can use the language-based methods for verification.

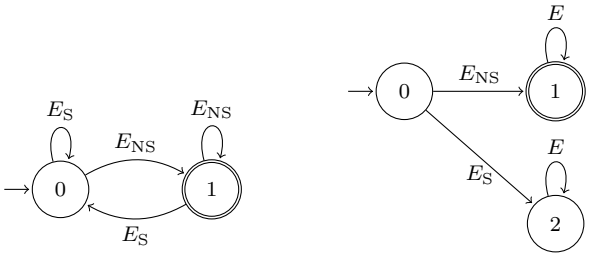

Fig. 4 The nonsecret specification automata H_{NS} for CSO (left) and ISO (right).

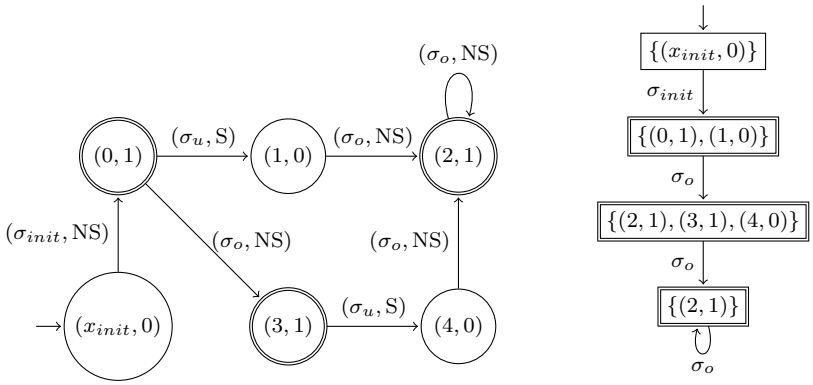

Fig. 5 The product $G \times H_{\text{NS}}$ (left) for $G = T^{IO}(\mathcal{A}, \ell)$ where $\mathcal{A}$ is from from Figure 2 and the nonsecret specification automaton H_{NS} for CSO from Figure 4 and the corresponding secret observer G_{SO} (right).

To do this we first construct $G = T^{IO}(\mathcal{A}, \ell)$. As $\mathcal{L}(H_{\text{NS}}) = E^*$, using Theorem 1 we can verify CSO of $\mathcal{A}$ by checking if every non-initial state of the secret observer $G_{SO} = \det(\Theta(G \times H_{\text{NS}}))$ is marked where $G = T^{IO}(\mathcal{A}, \ell)$. As an example of this method, we verify the current-state opacity of $\mathcal{A}$ from Figure 2 using its transformation $G = T^{IO}(\mathcal{A}, \ell)$. Assuming $\Sigma_o = \{\sigma_o\}$, we construct $G \times H_{\text{NS}}$ and $G_{SO} = \det(\Theta(G \times H_{\text{NS}}))$ which are depicted in Figure 5. As every non-initial state of G_{SO} is marked, we deduce $\mathcal{A}$ is CSO.

Remark 4 The construction $G = T^{IO}(\mathcal{A}, \ell)$ essentially moves the state label information from the states of $\mathcal{A}$ to the events of G. In the product $G \times H_{\text{NS}}$, these labels are then moved from the events back to the states in the form of state markings. As a result $G \times H_{\text{NS}}$ is the same as the original automaton $\mathcal{A}$ where nonsecret states are marked and there are new initial states resulting from x_{init} in G. In this way the secret observer method is comparable to the standard method for verifying current-state opacity [21] which checks if each state of the observer of $\mathcal{A}$ contains a nonsecret state. While our approach may seem convoluted for verifying CSO, the purpose of our discussion and of the above example are to demonstrate how our approach can be used to verify state-based notions of opacity in general. ◊

4.3 Initial-state opacity(ISO)

Next, we discuss the notion of initial-state opacity. Initial-state opacity describes the inability of an intruder to deduce that the initial-state of a run was secret. It can be defined as follows.

Definition 10 (Initial-State Opacity [29]) The automaton $\mathcal{A} = (X, \Sigma, \delta, X_0)$ is said to be *initial-state opaque* with respect to secret states $X_{\text{S}} \subseteq X_0$ and

observable events $\Sigma_o \subseteq \Sigma$ if

$$\begin{aligned} &\forall x_0 \in X_{\mathrm{S}}\ \forall s \in \mathcal{L}(\mathcal{A})\ s.t.\ \exists x \in \delta(x_0, s), \\ &\exists x_0' \in X_{\mathrm{NS}}\ \exists s' \in \mathcal{L}(\mathcal{A}),\ P_{\Sigma_o}(s) = P_{\Sigma_o}(s')\ \wedge\ \exists x' \in \delta(x_0', s')\,. \end{aligned} \tag{23}$$

Similar to the discussion of current-state opacity, we see that the initial-state opacity of $\mathcal{A}$ is equivalent to the total opacity of $(R_{\mathrm{S}}, R_{\mathrm{NS}})$ to the observation map Θ induced by Σ_o where

$$L_{\mathrm{NS}} = E_{\mathrm{NS}}E^*, \quad R_{\mathrm{NS}} = R \cap L_{\mathrm{NS}}, \quad R_{\mathrm{S}} = R \setminus R_{\mathrm{NS}}\,. \tag{24}$$

We can construct H_{NS} as in Figure 4 so that $\mathcal{L}_m(H_{\mathrm{NS}}) = L_{\mathrm{NS}}$ and $\mathcal{L}(H_{\mathrm{NS}}) = E^*$. Applying the secret observer method in this case is similar to transforming initial-state opacity to current-state opacity as in [29] and using the standard approach to verify current-state opacity. Furthermore, we can take advantage of the specific structure of H_{NS} to obtain more efficient verification methods. Namely, applying the reverse comparison method is similar to verifying ISO using the reversed initial-state estimator of [29] which is significantly more efficient than the inital method proposed in [22].

5 K-step & infinite step opacity

While current-state opacity captures the notion of hiding current secrets, K-step and infinite step opacity capture the notion of hiding past secrets. In this section, we define state-based notions of K-step and infinite step opacity over automata. We then show how these relate to the existing notions.

5.1 State-based K-step opacity

Consider a system as described in Section 4.1 consisting of an automaton $\mathcal{A} = (X, \Sigma, \delta, X_0)$ and map $\ell : X \to A$ labeling secret states with behavior $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$. We are given a subset of observable events $\Sigma_o \subseteq \Sigma$ inducing the observation map Θ. K-step opacity concerns visits to these secret states during the last K observations made by the intruder. We use the term *observation epoch* to refer to the system's behavior between observations. More specifically, the epoch starts when an observation is made and ends right before another observation is made or at the end of the run. We consider two types of secret behavior that can be exhibited in an observation epoch. In the first type, which we call **type 1**, *at least one secret state is visited*. In the second type, which we call **type 2**, *only secret states are visited*.

In order to describe these observation epochs in terms of the input-output pairs $E = (\Sigma \cup \{\sigma_{init}\}) \times A$, we define the sets of observable and unobservable input-output pairs by

$$E_o = \{e \in E \mid \Theta(e) \neq \epsilon\}, \quad E_{uo} = E \setminus E_o\,. \tag{25}$$

Here observability relates to the concepts of *silent transitions* from [14]. An unobservable pair $e \in E_{uo}$ is silent in that $\Theta(e) = \epsilon$, while an observable pair $e \in E_o$ is not silent as $\Theta(e) \neq \epsilon$. As we consider Θ induced by the projection of observable events Σ_o, it holds that $E_o = (\Sigma_o \cup \{\sigma_{init}\}) \times A$. In order to describe the secrecy of an observation epoch, we use the previous definition of the sets of secret and nonsecret input-output pairs $E_{\mathrm{S}}, E_{\mathrm{NS}}$ as in equation (20). With this we make the following definition.

Definition 11 The set of *observation epochs* is defined to be $L_{\mathrm{ep}} = E_o E_{uo}^*$. The sets of *observation epochs exhibiting type 1 or type 2 secrets*, respectively, are defined by

$$L_{\mathrm{ep,S},1} = L_{epoch} \cap (E^* \setminus E_{\mathrm{NS}}^*), \quad L_{\mathrm{ep,S},2} = L_{\mathrm{ep}} \cap E_{\mathrm{S}}^* \,. \tag{26}$$

Likewise the sets of type 1 and type 2 nonsecret epochs are defined by

$$\begin{aligned} L_{\mathrm{ep,NS},1} &= L_{\mathrm{ep}} \setminus L_{\mathrm{ep,S},1} = L_{\mathrm{ep}} \cap E_{\mathrm{NS}}^*, \\ L_{\mathrm{ep,NS},2} &= L_{\mathrm{ep}} \setminus L_{\mathrm{ep,S},2} = L_{\mathrm{ep}} \cap (E^* \setminus E_{\mathrm{S}}^*) \,. \end{aligned} \tag{27}$$

Because every run in R starts with the input-output pair (σ_{init}, a) for some $a \in A$ and $(\sigma_{init}, a) \in E_o$ by definition, it holds that $R \subseteq E_o E^* = L_{\mathrm{ep}}^+$. This means any run $r \in R$ can uniquely be written as a concatenation of observation epochs, i.e., $\exists M > 0$, $r = r_{\mathrm{ep},0} \cdots r_{\mathrm{ep},M-1}$ with $r_{\mathrm{ep},i} \in L_{\mathrm{ep}}$ for all $i < M$. We refer to the epoch $r_{\mathrm{ep},M-k-1}$ as the epoch k^{th} from the end or as k epochs ago. For K-step opacity, we define different classes of secret and nonsecret behavior for each epoch in the past, up to K epochs ago. For $k \leq K$ and type $j \in \{1, 2\}$ secrets, we define

$$L_{\mathrm{S},j}(k) = L_{\mathrm{ep}}^* L_{\mathrm{ep,S},j} L_{\mathrm{ep}}^k \,, \tag{28}$$

$$L_{\mathrm{NS},j}(k) = L_{\mathrm{ep}}^+ \setminus L_{\mathrm{S},j}(k) = (L_{\mathrm{ep}}^* L_{\mathrm{ep,NS},j} L_{\mathrm{ep}}^k) \cup \bigcup_{i=1}^{k} L_{\mathrm{ep}}^i \,. \tag{29}$$

We refer to $L_{\mathrm{S},j}(k)$ and $L_{\mathrm{NS},j}(k)$ as the *k-delayed secret and nonsecret behavior* specifications, respectively. Note that a run consisting of fewer than $k+1$ observation epochs is by definition not an element of $L_{\mathrm{S},j}(k)$ as a secret could not have occurred $k+1$ epochs ago. The k-delayed secret and nonsecret behavior of R with type $j \in \{1, 2\}$ secrets are then defined

$$R_{\mathrm{S},j}(k) = R \cap L_{\mathrm{S},j}(k), \qquad R_{\mathrm{NS},j}(k) = R \setminus R_{\mathrm{S},j}(k) = R \cap L_{\mathrm{NS},j}(k) \,. \tag{30}$$

By considering these secrets jointly, we can model an intruder deducing *if* a secret occurred within K epochs ago (or when a secret ever occurred in the case where $K = \infty$).

Definition 12 For $K \in \mathbb{N} \cup \{\infty\}$, we say the system $\mathcal{A}$ with secrets labeled by ℓ is *jointly K-step opaque with type j secrets* if $\{(R_{\mathrm{S},j}(k), R_{\mathrm{NS},j}(k))\}_{k=0}^{K}$ as defined in (30) is jointly opaque.

By considering these secrets separately, we can model an intruder deducing *when* a secret occurred within K epochs ago (or when a secret ever occurred in the case that $K = \infty$).

Definition 13 For $K \in \mathbb{N} \cup \{\infty\}$, we say the system $\mathcal{A}$ with ℓ is *separately K-step opaque with type j secrets* if $\{(R_{\mathrm{S},j}(k), R_{\mathrm{NS},j}(k))\}_{k=0}^{K}$ as defined in (30) is separately opaque.

For $K = \infty$ we refer to these definitions as *infinite step opacity*. While separate K-step opacity involves $R_{\mathrm{NS},j}(k)$ and hence $L_{\mathrm{NS},j}(k)$ for $k \leq K$, joint opacity only involves their intersections. For convenience we define for $K \in \mathbb{N}$

$$L_{\mathrm{NS},j}^{joint}(K) = \bigcap_{k=0}^{K} L_{\mathrm{NS},j}(k) = L_{\mathrm{ep}}^{*} L_{\mathrm{ep,NS},j}^{K+1} \cup \bigcup_{k=1}^{K} L_{\mathrm{ep,NS},j}^{k}\,, \tag{31}$$

so that $\bigcap_{k=0}^{K} R_{\mathrm{NS},j}(k) = R \cap L_{\mathrm{NS},j}^{joint}(K)$. In the joint sense, a run is secret if it consists entirely of nonsecret epochs or its last nonsecret epoch was at least $K+1$ epochs ago.

By comparing the nonsecret specification languages, we can relate the different notions of K-step opacity for $K \in \mathbb{N} \cup \{\infty\}$. Because $L_{\mathrm{ep,NS},1} \subseteq L_{\mathrm{ep,NS},2}$, it holds that $L_{\mathrm{NS},1}(K) \subseteq L_{\mathrm{NS},2}(K)$ and thus $R_{\mathrm{NS},1}(K) \subseteq R_{\mathrm{NS},2}(K)$. Hence joint and separate K-step opacity with type 1 secrets imply joint and separate K-step opacity with type 2 secrets, respectively. Additionally using Observation 2, we see that joint K-step opacity with type $j \in \{1,2\}$ secrets implies separate K-step opacity with type j secrets. These implications are depicted in Figure 6. This figure also depicts the relation to the existing notions of K-step opacity derived in the next section. Furthermore for $K \leq K'$, joint and separate K'-step opacity with type $j \in \{1,2\}$ secrets implies joint and separate K-step opacity with type j secrets.

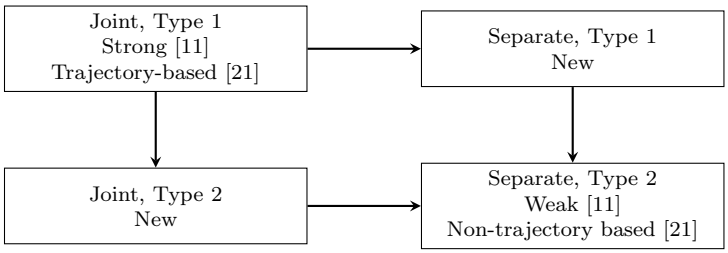

Fig. 6 Types of K-step opacity. Arrows indicate logical implication. For example, joint type 1 K-step opacity implies separate type 1 K-step opacity.

5.2 Relation to existing notions of K-step opacity

Now we show how these definitions relate to the existing notions of K-step opacity (for finite K). These notions were originally defined over deterministic

finite automata, so for consistency we derive these relations in this setting. Consider a deterministic automaton $\mathcal{A} = (X, \Sigma, \delta, \{x_0\})$ and interpret δ as a partial function $\delta : X \times \Sigma \to X$. Consider a set of secret states $X_{\mathrm{S}} \subseteq X$ and nonsecret states $X_{\mathrm{NS}} = X \setminus X_{\mathrm{S}}$ as well as a set of observable events $\Sigma_o \subseteq \Sigma$.

The first form of K-step opacity was developed in [21], and was later referred to as non-trajectory-based K-step opacity in [23] and weak K-step opacity in [12].

Definition 14 (K-step Weak Opacity [12]) The automaton $\mathcal{A}$ is *weakly K-step opaque* with respect to X_{S} and Σ_o if

$$\begin{aligned}&(\forall uv \in \mathcal{L}(\mathcal{A}) \ s.t. \ |P_{\Sigma_o}(v)| \leq K \ \wedge \delta(x_0, u) \in X_{\mathrm{S}})\\ &(\exists u'v' \in \mathcal{L}(\mathcal{A}))\\ &(P_{\Sigma_o}(uv) = P_{\Sigma_o}(u'v') \ \wedge \ P_{\Sigma_o}(u) = P_{\Sigma_o}(u') \ \wedge \ \delta(x_0, u') \in X_{\mathrm{NS}})\,.\end{aligned}$$

The second version we consider is referred to as trajectory-based K-step opacity in [23] and strong K-step opacity in [12].

Definition 15 (K-step Strong Opacity [12]) The automaton $\mathcal{A}$ is *strongly K-step opaque* with respect to X_{S} and Σ_o if

$$\begin{aligned}&(\forall t \in \mathcal{L}(\mathcal{A}))\\ &(\exists t' \in \mathcal{L}(\mathcal{A}), \ \forall u', v' \ s.t. \ t' = u'v')\\ &(P_{\Sigma_o}(t) = P_{\Sigma_o}(t') \ \wedge \ (|P_{\Sigma_o}(v')| \leq K \ \Rightarrow \ \delta(x_0, u') \in X_{\mathrm{NS}})\end{aligned}$$

Weak K-step opacity describes the inability of the intruder to deduce an exact time of a visit to a secret state within the last K observations. Strong K-step opacity describes the inability of the intruder to deduce there was a visit to a secret state within the last K observations. With this intuition we can relate weak to separate and strong to joint opacity.

Theorem 2 *Consider a deterministic automaton $\mathcal{A}$ with labeling map ℓ defined by the secret states X_S and observable events Σ_o. Then*

1. *Weak K-step opacity of $\mathcal{A}$ is equivalent to separate K-step opacity with type 2 secrets of $\mathcal{A}$.*
2. *Strong K-step opacity of $\mathcal{A}$ is equivalent to joint K-step opacity with type 1 secrets of $\mathcal{A}$.*

Proof Because the automaton $\mathcal{A}$ is deterministic, there is a unique sequence of states associated with each string in $\mathcal{L}(\mathcal{A})$. This defines a bijection $h : R \to \mathcal{L}(\mathcal{A})$ where $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$ such that

$$\forall r \in R, \quad P^I(r) = \sigma_{init} \cdot h(r), \quad \Theta(r) = \sigma_{init} \cdot P_{\Sigma_o}(h(r))\,. \tag{32}$$

Then note that we can write for $k \leq K$

$$\begin{aligned}h(R_{\mathrm{NS},1}(k)) = \{t \in \mathcal{L}(\mathcal{A}) \mid \ \forall i \leq |t|, \ |P_{\Sigma_o}(t_i \cdots t_{|t|-1})| = k \ \Rightarrow \\ \delta(x_0, t_0 \cdots t_{i-1}) \in X_{\mathrm{NS}}\}\end{aligned} \tag{33}$$

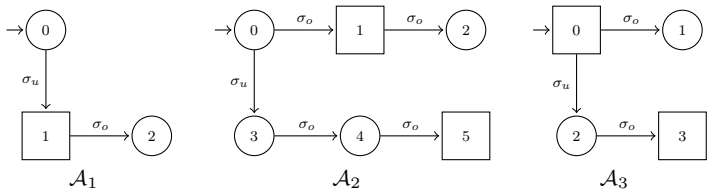

Fig. 7 Automata demonstrating the differences in the various notions of K-step opacity. Here square states denote secret states. The observable event set is $\Sigma_o = \{\sigma_o\}$.

$$h(R_{\mathrm{NS},2}(k)) = \{t \in \mathcal{L}(\mathcal{A}) \mid \ |P_{\Sigma_o}(t)| < k \vee \exists i \leq |t| \ |P_{\Sigma_o}(t_i \cdots t_{|t|-1})| = k \wedge \delta(x_0, t_0 \cdots t_{i-1}) \in X_{\mathrm{NS}})\}\,. \quad (34)$$

Suppose $\mathcal{A}$ is weakly K-step opaque and let $k \leq K$. Consider a run of $\mathcal{A}$ given by $r \in R$. If $|\Theta(r)| < k$ then by definition $r \in R_{\mathrm{NS},2}(k)$. Otherwise consider $t = h(r)$ so $|P_{\Sigma_o}(t)| \geq k$. Let $i \leq |t|$ be such that $|P_{\Sigma_o}(t_i \cdots t_{|t|-1})| = k$ and define $u = t_0 \cdots t_{i-1}$ and $v = t_i \cdots t_{|t|-1}$. By weak opacity of $\mathcal{A}$, there must exist $t' = u'v'$ such that $P_{\Sigma_o}(t) = P_{\Sigma_o}(t')$, $|P_{\Sigma_o}(v')| = k$, and $\delta(x_0, u') \in X_{\mathrm{NS}}$. Thus for $r' = h^{-1}(t')$ it holds that $r' \in R_{\mathrm{NS},2}(k)$ and $\Theta(r) = \Theta(r')$. Hence $\mathcal{A}$ is separately K-step opaque with type 2 secrets. The proof of the converse is similar.

Now we consider strong K-step opacity. Suppose that $\mathcal{A}$ is strongly K-step opaque. Consider a run of $\mathcal{A}$ given by $r \in R$ and define $t = h(r)$. By strong K-step opacity of $\mathcal{A}$, there exists $t' \in \mathcal{L}(\mathcal{A})$ with $P_{\Sigma_o}(t) = P_{\Sigma_o}(t')$ where for every $i' \leq |t'|$ such that $|P_{\Sigma_o}(t'_{i'} \cdots t'_{|t'|-1})| \leq K$ it holds that $\delta(x_0, t'_0 \cdots t'_{i'-1}) \in X_{\mathrm{NS}}$. Thus for $r' = h^{-1}(t')$ it holds that $r' \in R_{\mathrm{NS},1}(k)$ for all $k \leq K$ and $\Theta(r') = \Theta(r)$. Thus $\mathcal{A}$ is jointly K-step opaque with type 1 secrets. The proof of the converse is similar. □

The other notions of joint opacity with type 2 secrets and separate opacity with type 1 secrets, to our knowledge, have not been previously proposed. The differences between the proposed notions of K-step opacity stem from how secrets interact with unobservable behavior. To demonstrate how these notions differ, consider the automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ from Figure 7 and secret states X_{S} given by the square states and observable event set $\Sigma_o = \{\sigma_o\}$. In $\mathcal{A}_1$ for example, there are no type 2 secret epochs possible as a visit to secret state 1 must be preceded by a visit to nonsecret state 0 in the same epoch. Hence $\mathcal{A}_1$ is jointly and separately 1-step opaque with type 2 secrets. We can verify the various notions of 1-step opacity for all of these automata as depicted in Table 1.

To paraphrase, joint K-step opacity with type 1 secrets reflects the inability of the intruder to deduce *if* there was a period between observations where a *single* secret state was visited, while joint K-step opacity with type 2 secrets reflects the inability of the intruder to deduce *if* there was a period between observations where *only* secret states were visited. Likewise, separate K-step opacity with type 1 secrets reflects the inability of the intruder to deduce *when* there was a period between observations where a *single* secret state was

1-Step Opacity Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Separate Type 2	Yes	Yes	Yes
Separate Type 1	No	Yes	No
Joint Type 2	Yes	No	No
Joint Type 1	No	No	No

Table 1 The results of verifying joint and separate 1-step opacity with type 1 and type 2 secrets for the automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ from Figure 7.

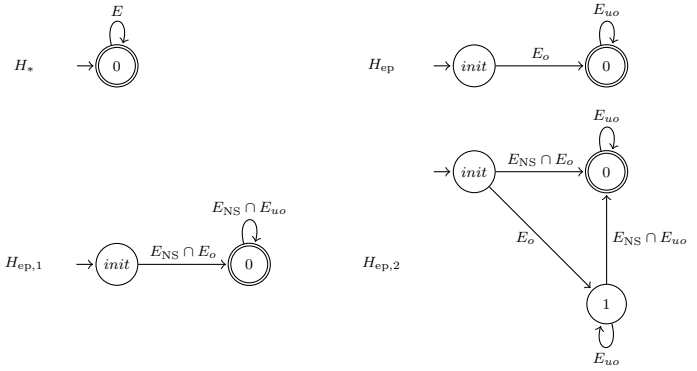

Fig. 8 Automata used to construct nonsecret specification automata for K-step opacity defined over input-output pairs E categorized into nonsecret pairs E_{NS} as defined in equation (20), and observable and unobservable pairs E_o, E_{uo} as defined in equation (25).

visited, while separate K-step opacity with type 2 secrets reflects the inability of the intruder to deduce *when* there was a period between observations where *only* secret states were visited. So we see for automata without unobservable events, type 1 and type 2 secrets are equivalent and these new notions of joint and separate reduce to the the existing notions of strong and weak. While these new notions may only reflect differences in the modeling of unobservable events in some sense, they demonstrate how the proposed approach can be used to formulate precise notions of opacity appropriate for a given problem.

6 Verification methods for finite K-step opacity

In this section we will present methods for verification of K-step opacity for finite K. First, we construct automata specifying nonsecret behavior. Then we show how to use these automata to verify joint K-step opacity and separate K-step opacity.

6.1 Nonsecret specification automata

In order to use language-based methods to verify K-step opacity, we must first construct automata that mark the corresponding nonsecret specification

languages. To do this, we will use the automata depicted in Figure 8 as building blocks. These automata are defined in terms of the input-output pairs E categorized into nonsecret pairs E_{NS} as defined in equation (20), and observable and unobservable pairs E_o, E_{uo} as defined in equation (25). Note that $\mathcal{L}_m(H_*) = E^*$, $\mathcal{L}_m(H_{\mathrm{ep}}) = L_{\mathrm{ep}}$, $\mathcal{L}_m(H_{\mathrm{ep},1}) = L_{\mathrm{ep,NS},1}$, and $\mathcal{L}_m(H_{\mathrm{ep},2}) = L_{\mathrm{ep,NS},2}$ [1]. To efficiently represent the nonsecret specifications languages, we note that

$$\mathcal{L}_{\mathrm{NS},j}(k{+}1) = L_{\mathrm{NS},j}(k)L_{\mathrm{ep}} \cup L_{\mathrm{ep}}, \quad \mathcal{L}^{joint}_{\mathrm{NS},j}(k{+}1) = L^{joint}_{\mathrm{NS},j}(k)L_{\mathrm{ep},NS,j} \cup L_{\mathrm{ep},NS,j}\,. \tag{35}$$

So by appropriately defining the initial states and concatenating the automata from Figure 8, we can construct automata that specify the nonsecret runs. While the standard concatenation construction adds an epsilon-transition between the marked states of one automaton and the initial states of the next, we can reduce the resulting number of states by merging these states as in the following construction.

Definition 16 Let $H^i = (Q^i, E, f^i, Q^i_0, Q^i_m)$ for $i \in \{1,2\}$ be such that $Q^2_0 \cap Q^2_m = \emptyset$. Let $Q^{\cup} = Q^1 \sqcup Q^2 \setminus Q^2_0$, $Q^{\cup}_0 = Q^1_0 \cup Q^1_m$, and $Q^{\cup}_m = Q^2_m$. Here $\sqcup$ denotes the disjoint union. We define the *concatenated automaton* $H^1 \cup H^2 = (Q^{\cup}, E, f^{\cup}, Q^{\cup}_0, Q^{\cup}_m)$ where for all $\sigma \in E$,

$$\begin{aligned} \forall q^1 \in Q^1 \setminus Q^1_m,\ & f^{\cup}(q^1,\sigma) = f^1(q^1,\sigma) \\ \forall q^2 \in Q^2 \setminus Q^2_0,\ & f^{\cup}(q^2,\sigma) = f^2(q^2,\sigma) \\ \forall q^1 \in Q^1_m,\ & f^{\cup}(q^1,\sigma) = f^1(q^1,\sigma) \cup \bigcup_{q^2 \in Q^2_0} f^2(q^2,\sigma)\,. \end{aligned}$$

This construction merges the marked states of H^1 with the initial states of H^2. Note that $\mathcal{L}_m(H^1 \cup H^2) = (\mathcal{L}_m(H^1) \cup \mathcal{L}_{mm}(H^1)) \cdot \mathcal{L}_m(H^2)$, where $\mathcal{L}_{mm}(H^1) = \mathcal{L}_m(H^1, Q^1_m)$ is the marked language of H^1 starting at the marked states of H^1.

Based on the relation in (35), we use this $\cup$ to construct specification automata.

Definition 17 We define the *nonsecret specification automata for K-step opacity* iteratively as follows. Let $H_{\mathrm{NS},j}(0) = H^{joint}_{\mathrm{NS},j}(0) = H_* \cup H_{\mathrm{ep},j}$ and for $k \geq 0$ define

$$H_{\mathrm{NS},j}(k+1) = H_{\mathrm{NS},j}(k) \cup H_{\mathrm{ep}}, \qquad H^{joint}_{\mathrm{NS},j}(k+1) = H^{joint}_{\mathrm{NS},j}(k) \cup H_{\mathrm{ep},j}\,. \tag{36}$$

The following result relates these nonsecret specification automata to the K-delayed nonsecret behavior defining K-step opacity.

Proposition 2 For every $K \in \mathbb{N}$ it holds that

$$\begin{aligned} L^+_{\mathrm{ep}} \cap \mathcal{L}_m(H_{\mathrm{NS},j}(K)) &= L_{\mathrm{NS},j}(K), \\ L^+_{\mathrm{ep}} \cap \mathcal{L}_m(H^{joint}_{\mathrm{NS},j}(K)) &= L^{joint}_{\mathrm{NS},j}(K)\,. \end{aligned} \tag{37}$$

[1] While $H_{\mathrm{ep},2}$ could be designed to be deterministic, our nondeterministic $H_{\mathrm{ep},2}$ offers reduced complexity.

Proof We show this for $H_{\mathrm{NS},2}(K)$. The proofs for the other cases are similar. We claim that for all $k \leq K$ that $\mathcal{L}_{mm}(H_{\mathrm{NS},2}(k)) = E_{uo}^*$ and

$$\mathcal{L}_m(H_{\mathrm{NS},2}(k)) = E^* L_{\mathrm{ep,NS},2} L_{\mathrm{ep}}^k \cup E_{uo}^* \bigcup_{i=1}^{k} L_{\mathrm{ep}}^i \,. \tag{38}$$

Note that this condition holds for $k = 0$ as

$$\mathcal{L}_m(H_{\mathrm{NS},2}(0)) = E^* L_{\mathrm{ep,NS},2}, \qquad \mathcal{L}_{mm}(H_{\mathrm{NS},2}(0)) = E_{uo}^* \,. \tag{39}$$

Now assume that condition (38) holds for some $k < K$. Then by definition of $\dot{\cup}$ we have

$$\begin{aligned}
\mathcal{L}_m(H_{\mathrm{NS},2}(k+1)) &= (\mathcal{L}_m(H_{\mathrm{NS},2}(k)) \cup \mathcal{L}_{mm}(H_{\mathrm{NS},2}(k)))\mathcal{L}_m(H_{\mathrm{ep}}) \\
&= (E^* L_{\mathrm{ep,NS},2} L_{\mathrm{ep}}^k \cup E_{uo}^* \cdot \bigcup_{i=1}^{k} L_{\mathrm{ep}}^i \cup E_{uo}^*) L_{\mathrm{ep}} \\
&= E^* L_{\mathrm{ep,NS},2} L_{\mathrm{ep}}^{k+1} \cup E_{uo}^* \cdot \bigcup_{i=1}^{k+1} L_{\mathrm{ep}}^i
\end{aligned}$$

Hence by induction, condition (38) holds for all $k \leq K$. Then note because $L_{\mathrm{ep}} = E_o E_{uo}^*$ that

$$\begin{aligned}
L_{\mathrm{ep}}^+ \cap \mathcal{L}_m(H_{\mathrm{NS},2}(k)) &= L_{\mathrm{ep}}^* L_{\mathrm{ep,NS},2} L_{\mathrm{ep}}^k \cup \bigcup_{i=1}^{k} L_{\mathrm{ep}}^i \\
&= L_{\mathrm{NS},2}(k) \,.
\end{aligned}$$

□

We can then use the automata $H_{\mathrm{NS},j}(K)$ and $H_{\mathrm{NS},j}^{joint}(K)$ in specifying K-step opacity. As before, consider an automaton $\mathcal{A}$ with secret states labeled by ℓ with behavior given by the input-output pairs $R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$. Then in terms of equation (30),

$$R \cap L_{\mathrm{NS},j}(K) = R_{\mathrm{NS},j}(K), \qquad R \cap L_{\mathrm{NS},j}^{joint}(K) = \bigcap_{k=0}^{K} R_{\mathrm{NS},j}(k) \,. \tag{40}$$

So $H_{\mathrm{NS},j}(k)$ for $k \leq K$ can be used as nonsecret specification automata for verification of separate K-step opacity with type j secrets. Likewise, $H_{\mathrm{NS},j}^{joint}(K)$ can be used for joint K-step opacity with type j secrets. In any case, it holds that $\mathcal{L}(H_{\mathrm{NS},j}(K)) = \mathcal{L}(H_{\mathrm{NS},j}^{joint}(K)) = E^*$ so we will be able to apply the secret observer method later on.

Remark 5 By expanding the recursive definitions of $H = H_{\mathrm{NS},j}(K)$ or $H = H_{\mathrm{NS},j}^{joint}(K)$, we can write H in the form $\dot{\bigcup}_{i=0}^{K+1} H^i$. To avoid ambiguity due to redundant state names, we refer to the state q of H^i by (q, i) when embedded in H. ◊

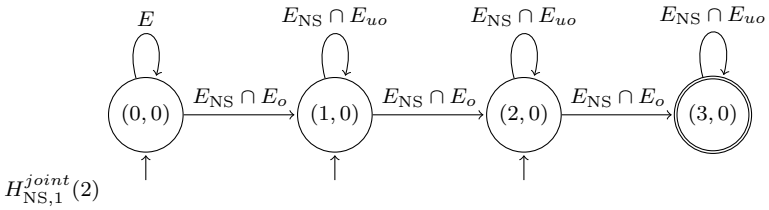

Fig. 9 The nonsecret specification automaton $H^{joint}_{\mathrm{NS},1}(2)$ for 2-step joint opacity with type 1 secrets.

6.2 Verification of joint K-step opacity

Using the nonsecret specification automaton $H^{joint}_{\mathrm{NS},j}(K)$ for $K \in \mathbb{N}$, we can verify joint K-step opacity with type j secrets as follows.

Approach 4 (Joint K-step opacity verification) Given $\mathcal{A}$, ℓ, Σ_o, and $K < \infty$, construct the label-transform $G = T^{IO}(\mathcal{A}, \ell)$, the nonsecret specification automaton $H^{joint}_{\mathrm{NS},j}(K)$, and the static mask Θ induced by Σ_o. We can then apply any of the language-based methods from Section 3.3 to $G, H^{joint}_{\mathrm{NS},j}(K)$, and Θ to verify the joint K-step opacity with type j secrets of $\mathcal{A}$. $\Diamond$

For example we depict $H^{joint}_{\mathrm{NS},1}(2)$ in Figure 9. Recall this automaton is constructed by concatenating H_* and three copies of $H_{\mathrm{ep},1}$. We apply the secret observer method to the automaton $\mathcal{A}$ from Figure 2 using its label transform $G = T^{IO}(\mathcal{A}, \ell)$ also depicted in 2. The construction of $G_{SO} = \det(\Theta(G \times H^{joint}_{\mathrm{NS},1}(2)$ is depicted in Figure 10. We see that the string $\sigma_{init}\sigma_o\sigma_o$ is not marked in G_{SO}. Hence by the secret observer method, $\mathcal{A}$ is not jointly 2-step opaque with type 1 secrets. Upon observing $\sigma_o\sigma_o$ we can deduce that $\mathcal{A}$ traversed the states $0, 1, 2, 2$ or $0, 3, 4, 2$ which both pass through secret states.

6.3 Verification of separate K-step opacity

Verification of separate K-step opacity is less straightforward than joint opacity. Using the definition of separate opacity, we could do this by verifying the total opacity of $(R, R_{\mathrm{NS},j}(k))$ for each $k \leq K$ using the language-based methods. Alternatively, we can combine these into a single test as in the joint case and avoid determinizing multiple automata by using two different approaches taking advantage of the structure of the problem.

By construction, $H_{\mathrm{NS},j}(k)$ is embedded within $H_{\mathrm{NS},j}(K)$ as a subautomaton for $k \leq K$. So we can use $H_{\mathrm{NS},j}(K)$ to specify the nonsecret runs $R_{\mathrm{NS},j}(k)$ for $k \leq K$ for separate K-step opacity. As in Remark 5, we can write $H_{\mathrm{NS},j}(k) = \bigcup_{i=0}^{k+1} H^i$ where $H^0 = H_*$, $H^1 = H_{\mathrm{ep,NS},j}$, and $H^i = H_{\mathrm{ep}}$ for $i \geq 2$. Recall using the convention of Remark 5, the marked states of $H_{\mathrm{NS},j}(k)$ are simply the marked states of H^k denoted by $Q^{k+1}_{\mathrm{NS},m}$ embedded into $H_{\mathrm{NS},j}(k)$ as $Q^{k+1}_{\mathrm{NS},m} \times \{k+1\}$. Hence it holds that $\mathcal{L}_{Q^{k+1}_{\mathrm{NS},m} \times \{k+1\}}(H_{\mathrm{NS},j}(K)) = \mathcal{L}_m(H_{\mathrm{NS},j}(k))$. This yields the following approach.

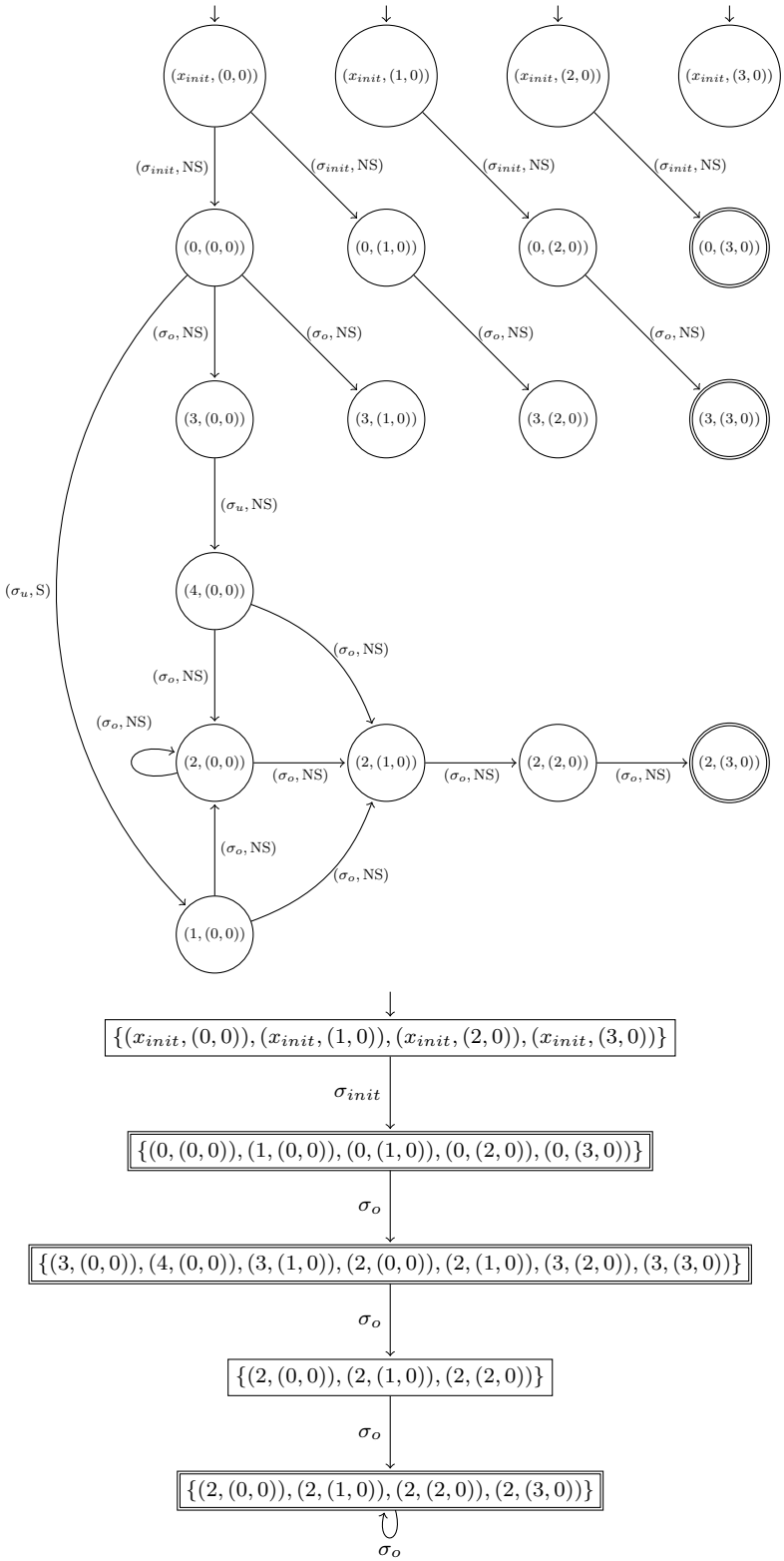

Fig. 10 The product(top) of G from Figure 2 with the nonsecret specification $H^{joint}_{\mathrm{NS},1}(2)$ and the corresponding secret observer G_{SO} (bottom).

Approach 5 (Separate K-step opacity verification using secret observer) Given $\mathcal{A}$, ℓ, Σ_o, and $K < \infty$, construct the label-transform $G = T^{IO}(\mathcal{A}, \ell)$, the nonsecret specification automaton $H_{\text{NS},j}(K)$, and the static mask Θ induced by Σ_o. Recall that $\mathcal{A}$ is separate K-step opaque with type j secrets if the k-delayed behavior with type j secrets is opaque for each $k \leq K$. We can verify this by applying the secret observer method for each $k \leq K$ to G, $H_{\text{NS},j}(K)$, and Θ where we redefine the marked states of $H_{\text{NS},j}(K)$ to be $Q^{k+1}_{\text{NS},m} \times \{k+1\}$. Each of these tests involves analyzing the states of the same automaton $G_{SO} = \det(\Theta(G \times H_{\text{NS},j}(K)))$ under different notions of state markings. As a result, we must only determinize a single automaton to apply this approach. $\Diamond$

However, the idea of this approach is not applicable to the reverse comparison method as this would require considering multiple sets of initial states. Alternatively, we can avoid multiple determinizations by utilizing the fact that the intruder's knowledge of the system's behavior only increases as they make more observations. Informally, if the intruder deduces a secret happened within the last $K-1$ observations, after making another observation they can still deduce a secret happened within the last K observations. So if the intruder can always make more observations, it suffices to consider secrets that occurred exactly K observations ago for the purposes of verification. This is similar to the results of Lemma 2 in [21]. We will show under some conditions that it suffices to verify total opacity of $(R, R_{\text{NS},j}(K))$ to verify separate K-step opacity with type j secrets. Here we say that $\mathcal{A}$ is **observation extendable** with respect to Θ if for every $r \in R = \mathcal{L}^{IO}(\mathcal{A}, \ell)$, there exists $r_{suf} \in E^*_{uo} E_o$ so that $(r \cdot r_{suf}) \in R$, where E_{uo}, E_o are defined as in equation (25). With this we claim the following result.

Theorem 3 *If $\mathcal{A}$ is observation extendable, then $\mathcal{A}$ is separate K-step opaque with type j secrets if and only if $(R, R_{NS,j}(K))$ is totally opaque to Θ.*

Proof Suppose that $\mathcal{A}$ is separately K-step opaque with type j secrets. Let $r \in R$. By the separate opacity of $\mathcal{A}$, there exists a run $r' \in R_{\text{NS},j}(K) = R_{\text{NS}}$ with $\Theta(r) = \Theta(r')$. Hence $(R_{\text{S}}, R_{\text{NS},j}(K))$ is totally opaque to Θ.

Conversely, suppose that $(R, R_{\text{NS},j}(K))$ is totally opaque to Θ. Then let $r \in R$ and $k \in \{0, \cdots, K\}$. As R is observation extendable, there exists an extended run $r_{ext} = r \cdot r_{suf}$ so that $r_{ext} \in R$ and $|\Theta(r_{suf})| = K - k$. By hypothesis, there exists a run $r'_{ext} \in R_{\text{NS}} = R_{\text{NS},j}(K)$ with $\Theta(r'_{ext}) = \Theta(r_{ext})$. By defining r'_{suf} to be the last $K-k$ observation epochs of r'_{ext}, we can write $r'_{ext} = r' \cdot r'_{suf}$ with $|\Theta(r'_{suf})| = K - k$. Then we see that $r' \in R_{\text{NS},j}(k)$ and $\Theta(r') = \Theta(r)$. Hence $\mathcal{A}$ is separately K-step opaque with type j secrets. $\square$

So when the system is observation extendable, we can verify separate K-step opacity in the following way.

Approach 6 (Separate K-step opacity verification for observation extendable systems) Given $\mathcal{A}$, ℓ, Σ_o, and $K < \infty$ where $\mathcal{A}$ is observation extendable with respect to the static mask Θ induced by Σ_o, construct

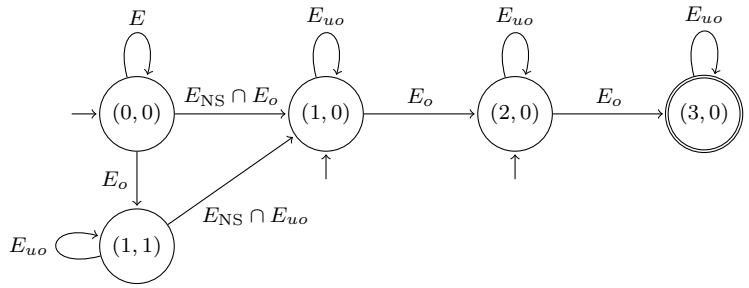

Fig. 11 The nonsecret specification automaton $H_{\mathrm{NS},2}(2)$ for separate 2-step opacity with type 2 secrets.

the label-transform $G = T^{IO}(\mathcal{A}, \ell)$ and the nonsecret specification automaton $H_{\mathrm{NS},j}(K)$ We can verify the separate K-step opacity with type j secrets of $\mathcal{A}$ by applying any of the language-based approaches to G, $H_{\mathrm{NS},j}(K)$, and Θ. $\Diamond$

Remark 6 While it may not be the case that $\mathcal{A}$ is observation extendable (for example if $\mathcal{A}$ is deadlocked), we can always modify $\mathcal{A}$ to be observation extendable while preserving K-step opacity. To do this we define a new automaton $\mathcal{A}_{ext}$ by adding an artificial observable event σ_{ext} as a self-loop for every state in $\mathcal{A}$. Then one can show that $\mathcal{A}_{ext}$ will be separately K-step opaque if and only if $\mathcal{A}$ is. Then by construction R_{ext} will be observation extendable, and so we can apply Approach 6 to $\mathcal{A}_{ext}$. $\Diamond$

Using Approach 6 we can verify separate K-step opacity using the reverse comparison or secret observer method. For example consider the system $\mathcal{A}$ from Figure 2 which is observation extendable and the nonsecret specification automaton $H_{\mathrm{NS},2}(2)$ which is depicted in Figure 11. The resulting secret observer $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS},2}(2)))$ for $G = T^{IO}(\mathcal{A}, \ell)$ is depicted in Figure 12. As every state except the initial state is marked, we see that $\mathcal{A}$ is separately 2-step opaque with type 2 secrets.

7 Complexity of K-step opacity verification

In this section, we analyze the complexity of the proposed methods for verifying K-step opacity for finite K for an automaton $\mathcal{A}$ with labeling map ℓ. These methods use the transformed automaton $G = T^{IO}(\mathcal{A}, \ell)$. First we analyze the secret observer using Approach 4 for joint opacity and Approach 5 for separate opacity. Then we analyze the reverse language comparison using Approach 6. Finally, we compare the secret observer methods to existing verifiers for K-step opacity known as the K-delayed state and trajectory estimators [11, 24]. For separate K-step we also compare with the two-way observer method [33] [16]. These results are summarized in Table 2 and Table 3. Note we allow the automaton $\mathcal{A}$ to be nondeterministic in general, but require $\mathcal{A}$ to be deterministic when comparing with existing methods as they only consider deterministic automata.

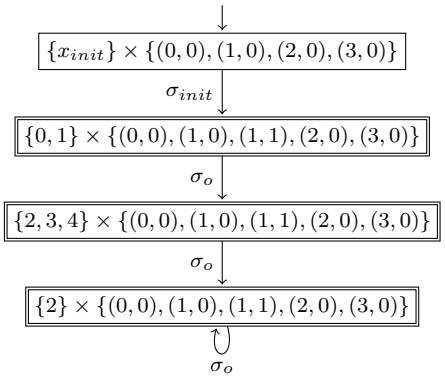

Fig. 12 The secret observer G_{SO} constructed for the automaton $\mathcal{A}$ from Figure 2 with the nonsecret specification $H_{\mathrm{NS},2}(2)$.

7.1 Secret observer complexity

Recall that applying the secret observer method in Approach 4, 5, or 6 to verify K-step opacity involves constructing the automaton $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS}}))$ for an appropriate choice of H_{NS}. We will bound the number of reachable states in this automaton to bound state complexity of these verification approaches. A naive upper bound for the number of states in the power set construction for determinization of an automaton with n states is simply 2^n. Using the known structure of H_{NS}, we can obtain a tighter bound for determinizing the automaton $\Theta(G \times H_{\mathrm{NS}})$. To do this, we will analyze which states of H_{NS} can be reached by runs that reach a fixed state of G in the following observation.

Observation 3 Consider two automata $G = (Q_G, E, f_G, Q_{G,0}, Q_{G,m})$ and $H_{\mathrm{NS}} = (Q_H, E, f_H, Q_{H,0}, Q_{H,m})$ with a static mask $\Theta : E^* \to \Gamma^*$. For convenience for $s \in E^*$ let $f_G(s) = \bigcup_{q_G \in Q_G} f_G(q_G, s)$ and $f_H(s) = \bigcup_{q_H \in Q_H} f_H(q_H, s)$. Suppose we are given sets $F \subseteq 2^{Q_H}$ and $C \subseteq \Gamma^*$ such that F is closed under union, $\emptyset \in F$, and for all $s \in \mathcal{L}(G \times H)$ such that $\Theta(s) \in C$ it holds that $f_H(s) \in F$. Then for every $\gamma \in C$ we can define the function $w_\gamma : Q_G \to F$ by

$$w_\gamma(q_G) = \bigcup_{\substack{s \in \Theta^{-1}(\gamma) \\ s.t.\ q_G \in f_G(s)}} f_H(s) \tag{41}$$

Then denote the automaton $\Theta(G \times H_{\mathrm{NS}})$ as

$$\Theta(G \times H_{\mathrm{NS}}) = (Q_{\Theta(G\times H)}, \Gamma \cup \{\epsilon\}, f_{\Theta(G\times H)}, Q_{\Theta(G\times H),0}, Q_{\Theta(G\times H),m}) \,. \tag{42}$$

For $\gamma \in C$ it holds that

$$f_{\Theta(G\times H)}(\gamma) = \bigcup_{s \in \Theta^{-1}(\gamma)} f_G(s) \times f_H(s) = \bigcup_{q_G \in Q_G} (\{q_G\} \times w_\gamma(q_G)) \,. \tag{43}$$

Hence the number of states in $\det(\Theta(G \times H_{\mathrm{NS}}))$ reached by a string in C is bounded by the number of functions from Q_G to F, of which there are $|F|^{|Q_G|}$. $\Diamond$

We can apply this observation to bound the complexity of the secret observer method. As $G_{SO} = \det(\Theta(G \times H_{\mathrm{NS}}))$ is deterministic, it has a single initial state reached by ϵ. So all states of G_{SO} other than the initial state are reached by $C = \Gamma^+$. Then to apply Observation 3, we must determine a set $F \supseteq \{f_{H_{\mathrm{NS}}}(s) \mid s \in C\}$ which is also closed under union and contains the empty set. We claim in verifying joint K-step opacity that

- for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},1}(K)$, we can choose $|F| = K + 3$ with

$$F = \{\emptyset\} \cup \{\{0, \cdots, k\} \times \{0\}\}_{k=0}^{K+1}, \tag{44}$$

- for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},2}(K)$, we can choose $|F| = 2(K+1)+1$ with

$$F = \{\emptyset\} \cup \{\{(0,0),(k+1,1)\} \cup (\{1, \cdots, k\} \times \{0,1\})\}_{k=0}^{K} \cup \{\{(0,0)\} \cup (\{1, \cdots, k+1\} \times \{0,1\})\}_{k=0}^{K}. \tag{45}$$

If we denote the number of states of the original automaton $\mathcal{A}$ as $n = |X|$, then the number of states of $G = T^{IO}(\mathcal{A}, \ell)$ is $n+1$, including the artificial initial state. Observation 3 then shows the number of states of G_{SO} other than the initial state is bounded by $|F|^n$. These bounds are given by $(K+3)^n$ for $H^{joint}_{\mathrm{NS},1}(K)$ and $(2K+3)^n$ for $H^{joint}_{\mathrm{NS},2}(K)$. For separate opacity, we use the naive power set bounds of $2^{n(K+2)}$ for $H_{\mathrm{NS},1}(K)$ and $2^{n(K+3)}$ for $H^{joint}_{\mathrm{NS},2}(K)$. These bounds are summarized in Table 2 and Table 3.

7.2 Reverse comparison complexity

We can use the same approach to analyze the reverse comparison method as in Approach 4 and Approach 6 to verify K-step opacity. These approaches require constructing the automaton $G_{RC} = \Theta(G)^R \times \det(\Theta(G \times H_{\mathrm{NS}})^R)$ for an appropriate choice of H_{NS}. By observing that $\Theta(G \times H_{\mathrm{NS}})^R = \Theta(G^R \times H^R_{\mathrm{NS}})$, we can use Observation 3, to bound the number of reachable states of $\det(\Theta(G^R \times H^R_{\mathrm{NS}})$. For the nonsecret specification automata H_{NS} use for K-step opacity, the reachable sets of H^R_{NS} are simpler than H_{NS}. Consider a string $s \in (L^+_{\mathrm{ep}})^R$ with $k = \max(0, K+1-|\Theta(s)|)$. Using the notation from Remark 5, we can see that H_{NS} must reach a state corresponding to H^k_{NS}. Consider the set $C_k = \Gamma^{K+1-k}$ with $1 \leq k \leq K$ and $C_0 = \Gamma^{K+1}\Gamma^*$. Then we determine a set $F_k \supset \{\delta_{H^R_{\mathrm{NS}}}(s) \mid s \in C_k\}$ that is closed under union and contains the empty set. We claim that

- for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},1}$ or $H_{\mathrm{NS}} = H_{\mathrm{NS},1}$ we can choose $|F_k| = 2$ with

$$F_k = \{\{(k,0)\}, \emptyset\}. \tag{46}$$

K	Forward ($n=4$)	Reverse ($n=4$)	Forward ($n=6$)	Reverse ($n=6$)
0	5	6	7	8
2	53	29	187	67
4	293	45	3007	147
8	2117	77	114487	275
16	16517	141	T/O	531

K	Forward ($n=4$)	Reverse ($n=4$)	Forward ($n=6$)	Reverse ($n=6$)
0	5	6	7	8
2	35	29	137	67
4	137	45	1547	147
8	749	77	36047	275
16	4949	141	1071767	531

Fig. 13 The number of states in the forward secret observer automata $G_{SO}(n)$ and reverse automata $G_{RC}(n)$ constructed from $G(n) = T^{IO}(\mathcal{A}(n), \ell_n)$. The bottom table uses $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},1}(K)$ and the top table uses $H_{\mathrm{NS}} = H_{\mathrm{NS},2}(K)$. Here T/O denotes a timeout where the automaton could not be constructed.

– for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},2}$ or $H_{\mathrm{NS}} = H_{\mathrm{NS},2}$ we can choose $|F_k| = 3$ with

$$F_k = \{\{(k,0)\}, \{(k,0),(k,1)\}, \emptyset\} \tag{47}$$

So by Observation 3 for C_k, the number of states of $\det(\Theta(G^R \times H^R_{\mathrm{NS}}))$ reached by a string $\gamma \in \Gamma^+$ with $k = \max(0, K+1-|\gamma|)$ is bounded by $|F_k|^{n+1}$ where $n = |X|$ is the number of states in the original automaton $\mathcal{A}$. Hence the number of states of $\det(\Theta(G^R \times H^R_{\mathrm{NS}}))$ is $O((K+1)2^n)$ for type 1 secrets and $O((K+1)3^n)$ for type 2 secrets. So then the number of states of $G_{RC} = \Theta(G^R) \times \Theta(\det(G \times H_{\mathrm{NS}}))^c$ is $O(n(K+1)2^n)$ for $H_{\mathrm{NS}} = H_{\mathrm{NS},1}(K), H^{joint}_{\mathrm{NS},1}(K)$ and $O(n(K+1)3^n)$ for $H_{\mathrm{NS}} = H_{\mathrm{NS},2}(K), H^{joint}_{\mathrm{NS},2}(K)$. These bounds are depicted in Table 2 and Table 3. From these bounds, we see that the reverse comparison method is distinguished by the factor K entering linearly into the bound. This may indicate for systems with a large number of states but small value of K, the reverse comparison method may be more efficient.

To demonstrate the advantage of the reverse language comparison, consider the following family of automata. For $n > 1$ define $\mathcal{A}(n) = (X_n, \Sigma_n, \delta_n, X_{n,0}, X_{n,m})$ where $X_n = \{0, \cdots, n-1\}$, $\Sigma_n = \{\sigma_0, \cdots, \sigma_{n-1}\}$, $\delta_n(i, \sigma_j) = (i+j) \bmod n$, $X_{n,0} = X_n \setminus \{0\}$, and $X_{n,m} = X_n$. Likewise define the labels $A = \{\mathrm{S}, \mathrm{NS}\}$ with $\ell_n(0) = \mathrm{S}$ and $\ell_n(i) = \mathrm{NS}$ for $i \neq 0$. We define all events to be observable $\Sigma_o = \Sigma_n$. After constructing $G(n) = T^{IO}(\mathcal{A}(n), \ell_n)$ for various n, we compute the number of states in the secret observer automaton $G_{SO}(n) = \det(\Theta(G(n) \times H_{\mathrm{NS}}))^c$ and in the reverse automaton $G_{RC}(n) = \Theta(G(n)^R) \times \det(\Theta(G(n)^R \times H^R_{\mathrm{NS}}))^c$ for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},1}(K)$ and $H_{\mathrm{NS}} = H_{\mathrm{NS},2}(K)$ across various values of K. These results are depicted in Figure 13. The number of states in the forward automata increases roughly exponentially with K while the number of states in the reverse automata increases linearly.

7.3 Comparison to K-delay State & trajectory estimators

We can compare our secret observer method with some existing methods for verification of K-step opacity. The first proposed verification methods for weak and strong K-step opacity are called the K-delay state estimator and K-delay trajectory estimator. These K-delay state estimators construct an automaton that estimates the possible states sequences over the last K observations from which one can deduce if weak opacity has been violated. The K-delay trajectory estimator augments this structure with a sequence of binary variables representing whether or not a secret state was visited between the observations to deduce if strong opacity has been violated. Their complexities are depicted in Table 2 and Table 3. By analyzing our Approach 5 for verifying weak K-step opacity, we see the resulting automaton estimates only the current state and whether a secret state has been visited during each of the last K observations. Likewise the automaton from our Approach 4 for strong K-step opacity estimates the current state and whether only secret states have been visited during each of the last K observations. As we can deduce whether or not secret states have been visited from the possible sequences of past states, we can view our secret observer automata as quotients of the K-delay estimators. In this way, the complexity of our proposed methods are at most that of the K-delay estimators for the respective forms of K-step opacity. We have provided a formal proof of this result in longer version of this paper [28].

To demonstrate this result, we construct a family of automata $\mathcal{A}(i)$ where the secret observer method has significantly reduced complexity compared to the delayed state/trajectory method for verification of strong/weak K-step opacity. For $i > 1$ define the deterministic automaton $\mathcal{A}(i) = (X_i, \Sigma_i, \delta_i, \{2\})$ where $X_i = \{1, \cdots, i\}$, $\Sigma_i = \{\sigma_1, \cdots, \sigma_i\}$, $\Sigma_o = \Sigma$, and the transition function defined by $\delta_i(j, \sigma_k) = k$. Consider the labeling map $\ell_i : X_i \to A$ where $A = \{\mathrm{S}, \mathrm{NS}\}$ defined by $\ell_i(1) = \mathrm{S}$ and $\ell_i(j) = \mathrm{NS}$ for $j \neq 1$. Note that $\mathcal{A}(i)$ recognizes a run along every state sequence in $\{2\} \cdot (X_i)^*$. Hence we see the K-delayed state observer states correspond to $\bigcup_{k=0}^{K} \{2\} \times (X_i)^k$, of which there are $\sum_{k=0}^{K} i^k = \frac{1-i^{K+1}}{1-i} = O(i^K)$ states. Let $G(i) = T^{IO}(\mathcal{A}(i), \ell_i)$. The secret observer $G_{SO}(i) = \det(\Theta(G(i) \times H_{\mathrm{NS},2}(K)))$ estimates the current state and the secrecy of the past $K+1$ epochs. We can verify that the number of states in $G_{SO}(i)$ is $O(i2^K)$. So we see that the secret observer method can be significantly less complex than the delayed state estimator for verification of weak K-step opacity. A similar result holds for strong K-step opacity.

8 Infinite step opacity

Now we consider K-step opacity for $K = \infty$, also called infinite step opacity. The results of Theorem 2 can be extended to the infinite step case. In particular our notion of separate infinite-step opacity with type 2 secrets corresponds to the existing notion of infinite step opacity as in [23, 33]. We will discuss how the previous verification methods for finite K can be adapted to this case.

Separate Type 2 (Weak)	
Algorithm	State Complexity
Secret Observer	$O(2^{n(K+3)})$
Reverse Comparison	$O(n(K+1)3^n)$
State Estimator [24]	$O((\lvert\Sigma_o\rvert+1)^K 2^n)$
Two-way Observer [33]	$O(\min(2^n, \lvert\Sigma_o\rvert^K)2^n)$

Table 2 State complexities of verification methods for separate K-step opacity with type 2 secrets (weak K-step opacity) of an automaton with n states. The discussion in Section 7.3 implies that the secret observer method has state complexity no worse than the K-delay state estimator.

Joint Type 1 (Strong)	
Algorithm	State Complexity
Secret Observer	$O((K+3)^n)$
Reverse Comparison	$O(K2^n)$
Trajectory Estimator [11]	$O((\lvert\Sigma_o\rvert+1)^K 2^n)$

Table 3 State complexities of verification methods for joint K-step opacity with type 1 secrets (strong K-step opacity) of an automaton with n states. The discussion in Section 7.3 implies that the secret observer method has state complexity no worse than the K-delay trajectory estimator.

Recall our definition of infinite step opacity involves an infinite number of nonsecret language specifications, i.e. the k-delayed nonsecret behavior $L_{\mathrm{NS},j}(k)$ for $k \in \mathbb{N}$ as defined in (30). Recall in the finite case we were able to reduce the multiple language comparison checks into a single check for verifying separate opacity. In Approach 5, we constructed one automaton that encompassed all of the nonsecret behavior, but this automaton would necessarily be infinite for $K = \infty$. In Approach 6, under the condition of observation extendability we showed it suffices to consider secret behavior occurring exactly K epochs ago, but there is no clear analog for this for $K = \infty$. Hence it appears that we cannot directly use our methods for verification of separate infinite step opacity. However we can use a result of [33] that states that infinite step opacity (separate opacity with type 2 secrets) is equivalent to K-step opacity for $K = 2^n$ where n denotes the number of states of the automaton in question. With this observation, we can verify separate infinite step opacity with type 2 secrets by verifying separate 2^n-step opacity. Alternatively, the two-way observer could be used to directly verify separate infinite step opacity [33].

We can more effectively apply our methods to joint infinite step opacity as this involves only one language comparison by definition. Note that we can define

$$L^{joint}_{\mathrm{NS},j}(\infty) = \bigcap_{i=0}^{\infty} L_{\mathrm{NS},j}(i) = L^{+}_{\mathrm{ep,NS},j} \tag{48}$$

As in the finite case, we can construct an automaton to specify this nonsecret behavior. Consider the automata depicted in Figure 14. To obtain a smaller complexity bound, we will apply the forward comparison method instead of the secret observer method. Recall in this case that we do not re-

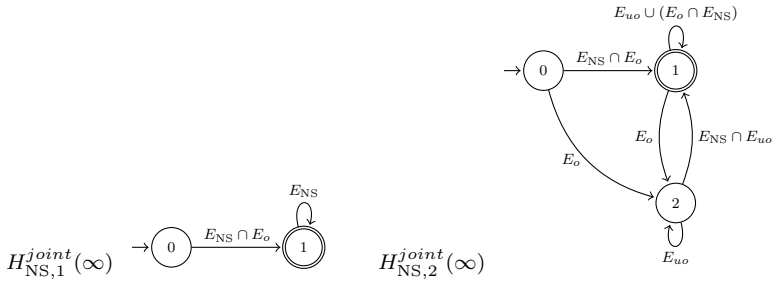

Fig. 14 The nonsecret specification automata for joint infinite step opacity.

quire $\mathcal{L}_m(H^{joint}_{\mathrm{NS},j}(\infty)) = E^*$. As before, we will analyze the complexity using Observation 3. Consider the set $C = \Gamma^+$. Then we determine a set $F \supset \{\delta_{H_{\mathrm{NS}}}(s) \mid s \in C_k\}$ that is closed under union and contains the empty set. We claim that

- for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},1}(\infty)$ we can choose $|F| = 2$ with

$$F = \{\emptyset, \{1\}\}\,. \tag{49}$$

- for $H_{\mathrm{NS}} = H^{joint}_{\mathrm{NS},2}(\infty)$ we can choose $|F| = 3$ with

$$F = \{\emptyset, \{2\}, \{1,2\}\} \tag{50}$$

So by Observation 3 for C, the number of states of $\det(\Theta(G \times H_{\mathrm{NS}}))$ reached by a string $\gamma \in \Gamma^+$ is bounded by $|F|^n$ with $n = |X|$ where $G = T^{IO}(\mathcal{A}, \ell)$. Then the number of states in the automaton $G_{FC} = \Theta(G) \times \det(\Theta(G \times H_{\mathrm{NS}}))^c$ other than the initial state is $O(n2^n)$ for type 1 secrets and $O(n3^n)$ for type 2 secrets. To the best of our knowledge, verification of joint infinite step opacity has not been reported in the literature previously.

9 Numerical examples

We evaluate the effectiveness of our verification methods for K-step opacity with numerical experiments. We compare the time and space complexity of the proposed methods with existing methods for verifying the existing notions of strong and weak K-step opacity. Recall these correspond to the notions of joint K-step opacity with type 1 secrets and separate K-step opacity with type 2 secrets, respectively. It should be noted while the existing methods were originally described for deterministic automata, there is a natural extension to the nondeterministic automata considered here. We compare the runtimes and number of states in the final verifier automata for an implementation of each method. In order to show how these methods scale with the size of the original system and the value of K, we verify the opacity of systems represented by randomly generated automata with secret states. We generate these automata in two ways. We present the runtimes and number of states in the verification

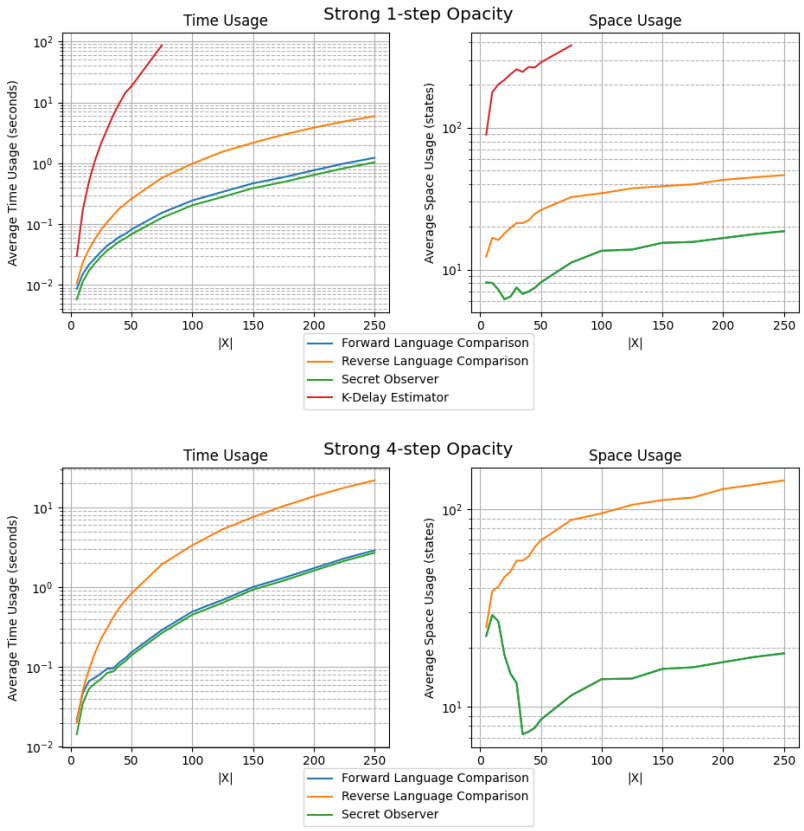

Fig. 15 Plots of average runtime (time usage) and the number of states in the verifier automata (space usage) versus the number of states in the random automata system model ($|X|$) for several methods for verifying strong K-step opacity.

automata averaged over 100 systems for fixed system sizes up to 250 states. These methods were implemented in the *MDESops* library [2].

9.1 First random generation approach

For the first experiment, we generate automata with a fixed number of states with a random number of outgoing transitions to random states. There are 18 events total with 6 observable events. All states are considered to be initial, and one state is labeled as secret.

For strong K-step opacity, we compare the proposed forward comparison, reverse comparison, and secret observer methods with the existing K-delay trajectory estimator. We consider both $K = 1$ and $K = 4$. The average results

[2] The library is available at `https://gitlab.eecs.umich.edu/M-DES-tools/desops/`.

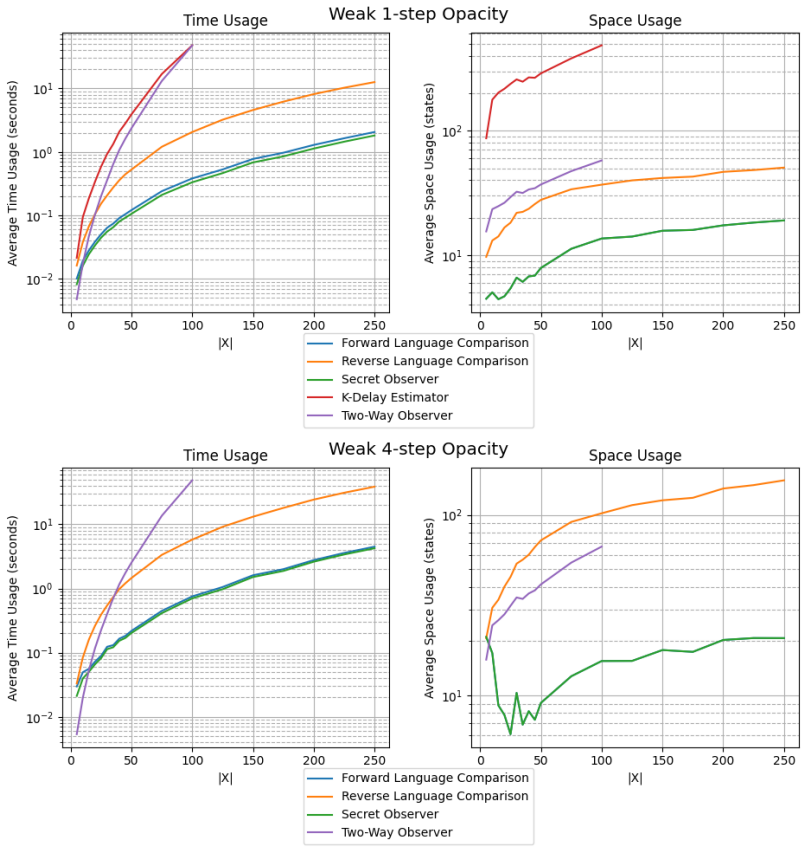

Fig. 16 Plots of average runtime (time usage) and the number of states in the verifier automata (space usage) versus the number of states in the random automata system model ($|X|$) for several methods for verifying weak K-step opacity.

over the randomly generated automata for verifying strong K-step opacity are depicted in Figure 15. Due to the long runtime of the K-delay trajectory estimator ($> 100s$), we do not evaluate this method for large automata in the $K = 1$ case and remove it entirely in the $K = 4$ case. In these examples, the forward comparison method performed nearly identically to the secret observer method, which is why it does not appear in the space usage plots. From these plots, we see that the proposed methods for verification perform significantly faster than the existing method. This supports the claim of Section 7.3, stating that the complexity of the secret observer method for verifying K-step opacity is less than that of the K-delay trajectory estimator. It is also interesting to note that the secret observer method outperforms the reverse language comparison for the small values of K investigated. This indicates the linear scaling with K in the complexity of this method is only significant for large values of K.

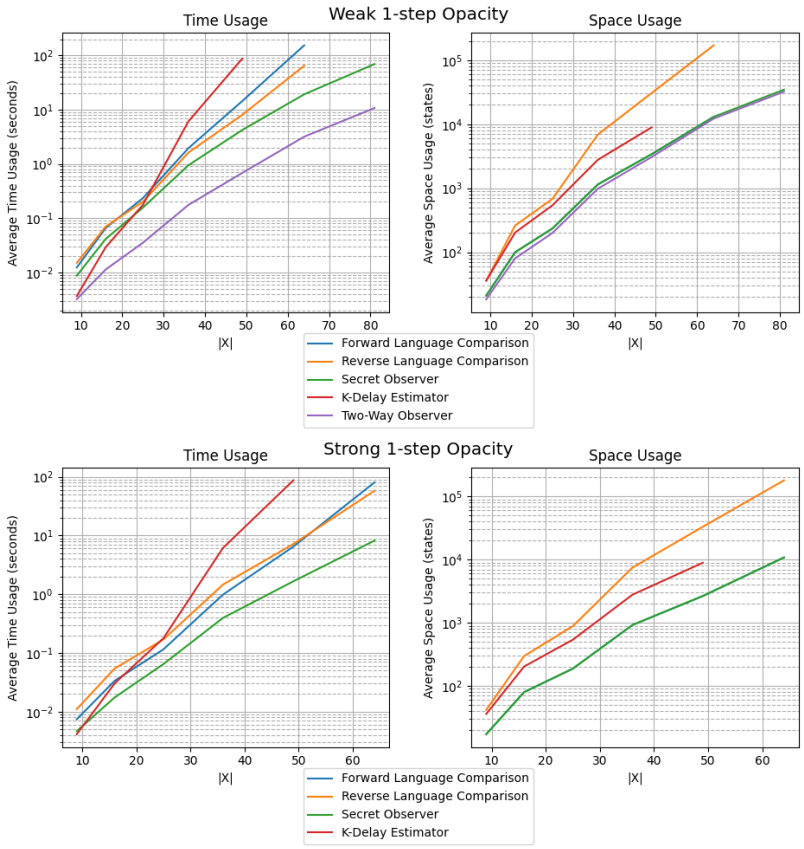

Fig. 17 Plots of average runtime (time usage) and the number of states in the verifier automata (space usage) versus the number of states in the system model ($|X|$) for several methods for verifying strong and weak 1-step opacity for the grid-based automata.

For weak K-step opacity, we compare the proposed forward comparison, reverse comparison, and secret observer methods with the existing K-delay state estimator and the two-way observer [33]. For the secret observer method, Approach 5 is used, while for the forward and reverse comparison methods, Approach 6 is used. As in the strong case, we consider both $K = 1$ and $K = 4$. The average results over the randomly generated automata for verifying weak K-step opacity are depicted in Figure 16. Due to the long runtime of the K-delay state estimator and two-way observer in some cases, we omit these results when necessary. As in the strong case, the forward comparison method performed nearly identically to the secret observer method. From these plots, we see that the proposed methods for verification outperform the existing K-delay state estimator in average runtime and size in all cases. While the runtime in applying the two-way observer is smaller for small-sized automata, the secret observer method outperforms it on the average in time and space

for larger automata (> 15 states). It should be noted that one property of this method for generating random automata is that for larger system sizes, nearly all of the automata generated were opaque for each notion of K-step opacity. We consider a more balanced and structured method for generation next.

9.2 Second random generation approach (grid-based)

In the second experiment, we generate automata as a square grid where states can transition to the 4 adjacent states. These transitions are then randomly removed or labeled with a random event. The number of observable events and secret states are scaled logarithmically with the system size. Again, all states are considered initial. The generation of these automata was tuned to provide a balance of automata that were opaque and not opaque across all system sizes.

We present results for verifying weak and strong 1-step opacity in Figure 17. These results show similar trends to the previous method for generating random automata. One notable difference is that the two-way observer method for verifying weak K-step opacity offers slightly improved performance over the proposed secret observer method.

10 Conclusion

We have presented several new results for the information-flow property of opacity in the context of discrete event systems. We presented a general framework of opacity to unify the many existing notions across a variety of system and intruder models. We used this framework to discuss notions of opacity over automata, both language-based and state-based. We provided several methods for verification of language-based opacity. We then developed a general approach for specifying state-based notions of opacity with automata and a transformation of these notions to language-based ones. Together, we used these results to describe existing notions of opacity like current-state opacity and initial-state opacity. We demonstrated how our approach unifies existing methods for opacity by showing the resulting language-based verification methods for these notions embody the existing verification methods. We further demonstrated the effectiveness of this approach in our investigation of K-step and infinite step opacity.

Using the intuition of K-step opacity with our approach, we derived a uniform view of four notions of K-step and infinite-step opacity. Two of these notions correspond to the existing notions of strong and weak K-step opacity, while the other two are new and meaningful notions. We developed appropriate specification automata for these notions, allowing verification with the language-based methods. We formally analyzed the complexity of these methods for K-step and infinite step opacity, showing these methods compare favorably in some instances to existing methods. In particular, we showed that the

proposed secret observer method outperforms the existing K-delay estimators for verifying strong and weak K-step opacity. Finally, we performed numerical experiments with randomly-generated automata to compare the verification methods. These results showed that the proposed verification methods offer increased performance over existing methods.

It would be interesting to apply our approach of specifying notions of opacity to capture more specific notions of privacy and security for real systems and evaluate the corresponding verification methods. These notions could capture time-dependent notions of privacy like K-step opacity or multiple notions of privacy arranged hierarchically. As we express opacity in a language-based way, any method for checking regular-language containment could be used for verification. For example, lattice-based methods as in [9] could be used for verification while avoiding the complexity of explicit determinization required by the methods presented here (note the general problem still remains PSPACE-complete). Additionally, it would be useful to extend the proposed framework to consider notions of opacity beyond the binary property considered here. For example, probabilistic opacity in a stochastic setting [34], approximate opacity for systems with numerical observations [35], or quantifying levels of opacity [2].

Finally, it would be interesting to use the proposed framework for opacity in the context of enforcement. Enforcement involves the synthesis of mechanisms to alter the system in order to guarantee opacity. As the framework expresses state-based notions of opacity in a language-based manner, existing language-based synthesis methods could be leveraged to enforce more general notions of opacity. For example, enforcement of opacity via supervisory control has been studied in [10] [33]. Additionally, enforcement via obfuscation as in [30, 31] appears to be readily implementable with this approach.

Acknowledgments

The authors would like to thank the reviewers for their useful and very detailed comments. They were most helpful in improving the paper for clarity and precision.

References

1. The Complexity of Diagnosability and Opacity Verification for Petri Nets | SpringerLink. URL `https://link-springer-com.proxy.lib.umich.edu/chapter/10.1007/978-3-319-57861-3_13`
2. Bérard, B., Mullins, J., Sassolas, M.: Quantifying Opacity. Mathematical Structures in Computer Science **25**(2), 361–403 (2015). DOI 10.1017/S0960129513000637. URL `http://arxiv.org/abs/1301.6799`. ArXiv: 1301.6799
3. Bryans, J., Koutny, M., Ryan, P.: Modelling Opacity Using Petri Nets. Electr. Notes Theor. Comput. Sci. **121**, 101–115 (2005). DOI 10.1016/j.entcs.2004.10.010
4. Bryans, J.W., Koutny, M., Mazaré, L., Ryan, P.Y.A.: Opacity generalised to transition systems. International Journal of Information Security **7**(6), 421–435 (2008). DOI 10.1007/s10207-008-0058-x. URL `https://doi.org/10.1007/s10207-008-0058-x`

5. Cassandras, C.G., Lafortune, S.: Introduction to discrete event systems, 2. ed edn. Springer, New York, NY (2008). OCLC: 255370614
6. Cassez, F.: The Dark Side of Timed Opacity. In: J.H. Park, H.H. Chen, M. Atiquzzaman, C. Lee, T.h. Kim, S.S. Yeo (eds.) Advances in Information Security and Assurance, Lecture Notes in Computer Science, pp. 21–30. Springer Berlin Heidelberg (2009)
7. Cassez, F., Dubreil, J., Marchand, H.: Dynamic Observers for the Synthesis of Opaque Systems. In: Z. Liu, A.P. Ravn (eds.) Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, pp. 352–367. Springer, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-04761-9_26
8. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal Logics for Hyperproperties. In: M. Abadi, S. Kremer (eds.) Principles of Security and Trust, Lecture Notes in Computer Science, pp. 265–284. Springer, Berlin, Heidelberg (2014). DOI 10.1007/978-3-642-54792-8_15
9. Doyen, L., Raskin, J.F.: Antichains for the Automata-Based Approach to Model-Checking. Logical Methods in Computer Science **5**(1), 5 (2009). DOI 10.2168/LMCS-5(1:5)2009. URL `http://arxiv.org/abs/0902.3958`. ArXiv: 0902.3958
10. Dubreil, J., Darondeau, P., Marchand, H.: Supervisory Control for Opacity. IEEE Transactions on Automatic Control **55**(5), 1089–1100 (2010). DOI 10.1109/TAC.2010.2042008. Conference Name: IEEE Transactions on Automatic Control
11. Falcone, Y., Marchand, H.: Runtime Enforcement of K-step Opacity. pp. 7271–7278 (2013). DOI 10.1109/CDC.2013.6761043
12. Falcone, Y., Marchand, H.: Enforcement and validation (at runtime) of various notions of opacity. Discrete Event Dynamic Systems **25**(4), 531–570 (2015). DOI 10.1007/s10626-014-0196-4. URL `https://doi.org/10.1007/s10626-014-0196-4`
13. Focardi, R., Gorrieri, R., Martinelli, F.: Non Interference for the Analysis of Cryptographic Protocols. In: U. Montanari, J.D.P. Rolim, E. Welzl (eds.) Automata, Languages and Programming, Lecture Notes in Computer Science, pp. 354–372. Springer, Berlin, Heidelberg (2000). DOI 10.1007/3-540-45022-X_31
14. Hadjicostis, C.N.: Introduction to Estimation and Inference in Discrete Event Systems. In: C.N. Hadjicostis (ed.) Estimation and Inference in Discrete Event Systems: A Model-Based Approach with Finite Automata, Communications and Control Engineering, pp. 1–14. Springer International Publishing, Cham (2020). DOI 10.1007/978-3-030-30821-6_1. URL `https://doi.org/10.1007/978-3-030-30821-6_1`
15. Jacob, R., Lesage, J.J., Faure, J.M.: Overview of discrete event systems opacity: Models, validation, and quantification. Annual Reviews in Control **41**, 135–146 (2016). DOI 10.1016/j.arcontrol.2016.04.015. URL `https://www.sciencedirect.com/science/article/pii/S1367578816300189`
16. Lan, H., Tong, Y., Guo, J., Giua, A.: Comments on "A new approach for the verification of infinite-step and K-step opacity using two-way observers" [Automatica 80 (2017) 162–171]. Automatica **122**, 109290 (2020). DOI 10.1016/j.automatica.2020.109290. URL `https://www.sciencedirect.com/science/article/pii/S0005109820304891`
17. Lin, F.: Opacity of discrete event systems and its applications. Automatica **47**(3), 496–503 (2011). DOI 10.1016/j.automatica.2011.01.002. URL `http://www.sciencedirect.com/science/article/pii/S0005109811000173`
18. Masopust, T., Yin, X.: Complexity of detectability, opacity and A-diagnosability for modular discrete event systems. Automatica **101**, 290–295 (2019). DOI 10.1016/j.automatica.2018.12.019. URL `https://linkinghub.elsevier.com/retrieve/pii/S0005109818306253`
19. Mazaré, L.: Using unification for opacity properties. In: In Proceedings of the Workshop on Issues in the Theory of Security (wits'04, pp. 165–176 (2004)
20. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for Web transactions. ACM Transactions on Information and System Security **1**(1), 66–92 (1998). DOI 10.1145/290163.290168. URL `https://doi.org/10.1145/290163.290168`
21. Saboori, A., Hadjicostis, C.N.: Notions of security and opacity in discrete event systems. In: 2007 46th IEEE Conference on Decision and Control, pp. 5056–5061 (2007). DOI 10.1109/CDC.2007.4434515. ISSN: 0191-2216
22. Saboori, A., Hadjicostis, C.N.: Verification of initial-state opacity in security applications of DES. In: 2008 9th International Workshop on Discrete Event Systems, pp. 328–333 (2008). DOI 10.1109/WODES.2008.4605967

23. Saboori, A., Hadjicostis, C.N.: Verification of infinite-step opacity and analysis of its complexity*. IFAC Proceedings Volumes **42**(5), 46–51 (2009). DOI https://doi.org/10.3182/20090610-3-IT-4004.00013. URL https://www.sciencedirect.com/science/article/pii/S1474667015355944
24. Saboori, A., Hadjicostis, C.N.: Verification of K-step opacity and analysis of its complexity. In: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, pp. 205–210 (2009). DOI 10.1109/CDC.2009.5400083. ISSN: 0191-2216
25. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time(Preliminary Report). In: Proceedings of the fifth annual ACM symposium on Theory of computing, STOC '73, pp. 1–9. Association for Computing Machinery, New York, NY, USA (1973). DOI 10.1145/800125.804029. URL https://doi.org/10.1145/800125.804029
26. Tong, Y., Li, Z., Seatzu, C., Giua, A.: Verification of State-Based Opacity Using Petri Nets. IEEE Transactions on Automatic Control **62**(6), 2823–2837 (2017). DOI 10.1109/TAC.2016.2620429. Conference Name: IEEE Transactions on Automatic Control
27. Willems, J.C.: The Behavioral Approach to Open and Interconnected Systems. IEEE Control Systems Magazine **27**(6), 46–99 (2007). DOI 10.1109/MCS.2007.906923. Conference Name: IEEE Control Systems Magazine
28. Wintenberg, A., Blischke, M., Lafortune, S., Ozay, N.: A General Language-Based Framework for Specifying and Verifying Notions of Opacity. arXiv:2103.10501 [cs] (2021). URL http://arxiv.org/abs/2103.10501. ArXiv: 2103.10501
29. Wu, Y.C., Lafortune, S.: Comparative analysis of related notions of opacity in centralized and coordinated architectures. Discrete Event Dynamic Systems **23**(3), 307–339 (2013). DOI 10.1007/s10626-012-0145-z. URL https://doi.org/10.1007/s10626-012-0145-z
30. Wu, Y.C., Lafortune, S.: Synthesis of insertion functions for enforcement of opacity security properties. Automatica **50**(5), 1336–1348 (2014). DOI 10.1016/j.automatica.2014.02.038. URL https://www.sciencedirect.com/science/article/pii/S0005109814000764
31. Wu, Y.C., Raman, V., Rawlings, B.C., Lafortune, S., Seshia, S.A.: Synthesis of Obfuscation Policies to Ensure Privacy and Utility. Journal of Automated Reasoning **60**(1), 107–131 (2018). DOI 10.1007/s10817-017-9420-x. URL https://doi.org/10.1007/s10817-017-9420-x
32. Wu, Y.C., Sankararaman, K.A., Lafortune, S.: Ensuring Privacy in Location-Based Services: An Approach Based on Opacity Enforcement. IFAC Proceedings Volumes **47**(2), 33–38 (2014). DOI 10.3182/20140514-3-FR-4046.00008. URL https://linkinghub.elsevier.com/retrieve/pii/S1474667015373778
33. Yin, X., Lafortune, S.: A new approach for the verification of infinite-step and K-step opacity using two-way observers. Automatica **80**, 162–171 (2017). DOI 10.1016/j.automatica.2017.02.037. URL http://www.sciencedirect.com/science/article/pii/S0005109817301115
34. Yin, X., Li, Z., Wang, W., Li, S.: Infinite-step opacity of stochastic discrete-event systems. In: 2017 11th Asian Control Conference (ASCC), pp. 102–107 (2017). DOI 10.1109/ASCC.2017.8287150
35. Yin, X., Zamani, M., Liu, S.: On Approximate Opacity of Cyber-Physical Systems. IEEE Transactions on Automatic Control pp. 1–1 (2020). DOI 10.1109/TAC.2020.2998733. Conference Name: IEEE Transactions on Automatic Control