

A Two-Pronged Approach to Security of Discrete-Event Systems*

Shoma MATSUI · Karen RUDIE · Kai CAI

DOI:

Received: 13 June 2024 / Revised: 10 September 2024

©The Editorial Office of JSSC & Springer-Verlag GmbH Germany 2025

Abstract This paper investigates a security problem of simultaneously addressing two types of attacks: Eavesdropping and infiltration. The authors model the target system as a discrete-event system (DES) with subsets of concealable events and protectable events, in order to make the proposed methodology applicable to various practical systems and employ two existing works of DES security: Degree of opacity and state protection. Specifically, the authors consider that all protectable events are observable, and some observable events are concealable. In addition, protectable events cannot be protected once they are concealed. Given such a constraint, the goal is to figure out which events to conceal and which transitions to protect so that the prescribed requirements of degree of opacity and state protection are satisfied. In this work the authors decide which events to conceal as all transitions of a given event label are concealed or not concealed. The proposed problem formulation also requires a solution to only involve absolutely necessary protectable events in order for the system to avoid superfluous protection costs. The authors first examine a general version of our security problem with an intuitive algorithm to compute acceptable solutions, and then present a special version which results in a reduced computation time compared to the general version.

Keywords Degree of opacity, discrete-event systems, security, state protection.

1 Introduction

Due to the proliferation of possible attack surfaces in cyber-physical systems, it is increasingly crucial to protect the systems not only against malicious agents from external environments, but also against insiders who try to identify vulnerable behaviours of the systems by eavesdropping information the systems generate.

Shoma MATSUI · Karen RUDIE (Corresponding author)

Department of Electrical and Computer Engineering & the Ingenuity Labs Research Institute, Queen's University, Kingston K7L3N5, Canada. Email: s.matsui@queensu.ca; karen.rudie@queensu.ca.

Kai CAI

Department of Core Informatics, Osaka Metropolitan University, Osaka 545-0051, Japan.

Email: cai@omu.ac.jp.

*This work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, and JSPS KAKENHI under Grant Nos. 21H04875 and 22KK0155.

◇This paper was recommended for publication by Editor HUANG Minyi.

In this paper, we consider systems that possess two types of special states in addition to normal states: Secret states and critical states. Secret states represent components of the system at which external agents should not know if the system currently resides. Critical states, on the other hand, model the system's specific components which no one should be able to access freely. For these special states, we address two types of attacks: Eavesdropping and infiltration. Under eavesdropping attacks, adversaries are supposed to observe the behaviour of their target system and try to determine whether the system is currently at a secret state. On the other hand, infiltration attacks are more active than eavesdropping, i.e., attackers are trying to intrude into the system and access its critical states directly. We take a *two-pronged approach* to defend the system against such attacks: Obfuscation against eavesdropping and protection against infiltration. Specifically, we model our system as a discrete-event system (DES)^[1] to make our methodology applicable to various systems and to employ two existing security works of DES: *Degree of opacity*^[2] for obfuscation and *secure reachability*^[3] for protection.

There is considerable interest in the study of security among research groups in the DES community. For instance, the degree of opacity is an enhanced notion of opacity^[4–11], which is one of the widely investigated problems of privacy and secrecy in DES, to measure how much a secret state is indistinguishable with non-secret states for external agents. For an overview of opacity, readers are referred to the literatures [12, 13]. The notion of opacity that we use in this paper is *current-state opacity* (CSO) which was first introduced by Bryans, et al.^[4], and later extended to the representation using finite state automata^[5]. The central idea of CSO is a condition of a plant which holds if attackers can never be certain whether the current state of the plant is a secret state. Sharing the common assumption among opacity definitions that attackers have a full knowledge of the system structure, the work by Han, et al.^[14] introduced a stronger version of CSO which requires the system to generate strings which never reach secret states while having the same projection as strings that reach secret states.

While CSO can be verified by building an observer automaton of the system^[15], various methodologies to enforce CSO on non-CSO systems have been proposed in the last decade. The work by Wu and Lafortune^[16] introduced an insertion function which inserts additional observable events to the output from the system, and their methodology was extended to the decentralized and modular versions of CSO^[17]. Later on, the work by Mohajerani, et al.^[18] presented an approach of using an edit function that can not only insert additional events but also can delete events, to solve the enforcement problem of CSO in for modular systems. On the other hand, the work by Tong, et al.^[19] proposed an enforcement technique using supervisory control^[20] on the system by computing an *augmented I-observer* which is the parallel composition of the attacker's observation and the system's genuine output. Another approach to enforce CSO was proposed by Barcelos and Basilio^[21], which changes the order of observable events in strings generated by the system. Assuming that an attacker can observe the subset of observable events and that all controllable events which can be disabled by the controller are observable, Moulton, et al.^[22] showed that the computational complexity of synthesizing a supervisor which enforces CSO can be reduced by computing a series of subobservers, a relaxed variation of subautomaton. Another notable work is by Barcelos and Basilio^[23] which intro-

duced a notion of *utility behaviour* which should always be disclosed to the legitimate agents when that behaviour occurs in the system, while ensuring that CSO remains satisfied.

Unlike many works where opacity is a binary property, the work by Schonewille, et al.^[2] developed a quantitative definition of CSO which captures to what extent secret states in the system are opaque to external agents.

In contrast to the passive attack scheme of opacity, *sensor deception*^[24] and *actuator enablement*^[25] consider active attackers that try to tamper the system's behaviour by altering the system's observation and forcibly enabling events disabled by the controller. For sensor deception attack, the work by Meira-Góes, et al.^[26] employed a game-like structure between the system and the controller to figure out which observable events the attacker has to insert or delete against the controller's observation, so that the system reaches unsafe states. The subsequent work^[27] addressed a dual problem in which the goal is to obtain a robust supervisor against sensor deception attacks. For actuator enablement attack, the work by Lin, et al.^[28] presented a methodology to figure out which disabled transitions should be enabled by supposing that the attacker eavesdrops the control commands in addition to the subset of sensor information so that the attacker may be able to exploit the controller's decision in order to infer the occurrence of the attacker's unobservable events. The subsequent work by Lin, et al.^[29] relaxed the normality assumption^[28]. The work by Ma and Cai^[30] addressed the synthesis problem of a robust supervisor against actuator enablement attack in a further dire situation from the system's viewpoint where any controllable events can be attacked. While the aforementioned works independently deal with sensor deception and actuator enablement, the work by Meira-Góes, et al.^[31] presented a methodology to synthesize a robust supervisor against both types of attack by applying standard supervisory control with partial observation.

This paper also investigates active attackers who attempt to evade security mechanisms. Unlike much of the literature on sensor deception and actuator enablement, however, based on previous work^[3], we consider that the goal of active attackers is to reach special states called critical states in the system, instead of leading the system to its unsafe state. The main difference from other security notions in DES is that the systems are supposed to be equipped with security modules that can “protect” certain transitions in the system. Protection here is a somewhat general term that indicates a test which must be passed by external agents (including both legitimate ones and adversaries). One example of such protections is setting up a password to connect to a Wi-Fi access point. This central notion of protection over transitions in the system is analogous to transition disablement of standard supervisory control of DES in [20]. In this context, our goal is to ensure that every critical state is *protected*, i.e., every path from the initial state to special states contains at least the required number of protected transitions. The related works of state protection include a quantitative notion of *dynamic clearance level*^[32] to capture that the authorization given to agents to access critical states is transient, that is, agents must pass protected transitions again once a given number of consecutive unprotected events happen. Another relevant work of state protection was presented by Liu, et al.^[33], investigating a situation where a protectable event must not be protected once the prefix of a string contains a certain number of protected events. The work by Ma, et al.^[34] is also intriguing, as it assigns

a cost weight, a positive real number instead of a positive integer level, to each protectable event and addresses the problem of finding a (globally) optimal policy so that all critical states are protected.

The main contributions of this paper are threefold.

- 1) We propose a novel approach to simultaneously deal with passive attacks and active attack by combining the existing methodologies of degree of opacity and state protection against passive attacks and active attacks, respectively. Specifically, supposing that some events in the system are concealable and/or protectable, we formulate the problem of determining which events and transitions in the system should be concealed and/or protected, so that the target system satisfies prescribed quantitative requirements of degree of opacity, called an *obfuscation requirement*, and state protection, called a *protection requirement*. We also require the overall cost level of protection to be feasibly minimum, i.e., a solution to the problem should not yield unnecessarily costly (protectable) events, while both the obfuscation requirement and the protection requirement are satisfied. One key feature of our problem formulation is that concealed events cannot be protected, and vice versa. We believe that this limitation is realistic in practical implementations such as packet filtering and gateway authentication^[35], and it also motivates us to formally show that the system with full observation can be protected once we can determine how to protect the system under partial observation. While such a cybersecurity scheme involving two protection mechanisms is one of the potential applications of this work, another example could be hiring an external security company that checks identities of employees at a building entrance but is not authorized to enter specific office floors where the company's sensitive information is stored, so that the internal information is concealed to external entities while the access to the information is protected.
- 2) An effective algorithm to compute possible solutions is presented. Due unavoidably to the nature of partial observation in DES, our algorithm has exponential time complexity. It is also worth noting that there could be multiple solutions that satisfy all of our security requirements, because our problem essentially requires that two incomparable properties, namely degree of opacity and state protection, be satisfied at the same time, while imposing a trade-off between event concealment and transition protection.
- 3) Finally, we present a special version of our problem formulation which is solvable by two different algorithms with reduced computation complexity. By proving the monotonicity of degree of opacity in a special case, it is also shown that one of the two algorithms always results in a higher degree of opacity than the other one, while both produce solutions with the same overall level of protection cost.

The rest of this paper is organized as follows. We present some preliminary definitions, specifically DES with partial observation, degree of opacity, and state protection in Section 2. Next, Section 3 formulates a general version of the problem of finding which events to be concealed and which transitions to be protected, and presents an algorithm to compute acceptable

solutions. In Section 4, we formulate a special version of the problem in Section 3 by introducing additional constraints so that the overall computation time is reduced compared to the general version. Finally, Section 5 concludes this paper.

2 Preliminaries

2.1 Discrete-Event Systems and Partial Observation

One of the central notions employed in this paper is discrete-event system (DES) which illustrates systems as deterministic finite state automata, or simply automata. Specifically, we represent the target system as an automaton G defined as a 4-tuple:

$$G = (Q, \Sigma, \delta, q_0), \quad (1)$$

where Q is the set of states, Σ is the set of events, $\delta : Q \times \Sigma \rightarrow Q$ is the (partial) transition function, and $q_0 \in Q$ is the initial state. The automaton G is usually called a *plant* which is the system without our intervention. The transition function δ is extended to $\delta : Q \times \Sigma^* \rightarrow Q$ in the standard manner^[20], where Σ^* is the Kleene closure of Σ that comprises an empty string ε and all strings composed of the events in Σ . We call a set of strings *language*, and the language generated by G is denoted by $L(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q\}$. Henceforth, the subsets of secret states and critical states in a plant G are denoted by $Q_s \subseteq Q$ and $Q_c \subseteq Q$, respectively. Note that $Q_s \cap Q_c$ may not be empty, that is, there could be states that are both secret and critical, depending on how the target system is modelled as a DES.

To introduce the concept of obfuscation into our system model, we divide the set of events Σ into two (disjoint) subsets of observable events Σ_o and unobservable events Σ_{uo} , i.e., $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$. When the attacker is eavesdropping the system's behaviour, they cannot directly determine whether any events in Σ_{uo} have happened in the target system. In other words, when the attacker observes a specific string, there could be multiple possible strings that have actually been generated by the system, resulting in multiple possible states in which the eavesdropper thinks the system could reside.

In the DES framework, there is a useful tool called an *observer automaton*^[20] which enables us to emulate the attacker's conjecture of the system behaviour. Given a set of observable events $\Gamma \subseteq \Sigma_o$, an observer $H(\Gamma)$ of a plant G with respect to Γ is denoted by $Obs(G, \Gamma)$ which is also represented as a 4-tuple:

$$H(\Gamma) = Obs(G, \Gamma) = (\mathcal{A}, \Gamma, \delta_{H(\Gamma)}, A_0), \quad (2)$$

where $\mathcal{A} \subseteq 2^Q$ is the set of observer states, i.e., the set of subsets of states in Q , $\delta_{H(\Gamma)} : \mathcal{A} \times \Gamma \rightarrow \mathcal{A}$ is the partial transition function which is extended to $\delta_{H(\Gamma)} : \mathcal{A} \times \Gamma^* \rightarrow \mathcal{A}$ in the same manner as δ , and A_0 is the initial state of $H(\Gamma)$. The observer automaton can be constructed by replacing unobservable events with ε and then doing the standard conversion from a nondeterministic automaton to a deterministic one^[36].

Another useful notation of partial observation is *natural projection* $P : \Sigma^* \rightarrow \Gamma^*$ defined

recursively by

$$P(\varepsilon) := \varepsilon, \quad P(\sigma) := \begin{cases} \sigma, & \text{if } \sigma \in \Gamma, \\ \varepsilon, & \text{if } \sigma \notin \Gamma, \end{cases}$$

$$P(s\sigma) := P(s)P(\sigma) \text{ for } s \in \Sigma^* \text{ and } \sigma \in \Sigma.$$

In other words, the natural projection P removes unobservable events from a given string. The domain and codomain of P can be extended to languages, i.e., $P : 2^{\Sigma^*} \rightarrow 2^{\Gamma^*}$ which is given by $P(L) = \bigcup_{s \in L} P(s)$ for a language $L \subseteq \Sigma^*$. The inverse of a natural projection P is defined as $P^{-1} : \Gamma^* \rightarrow 2^{\Sigma^*}$ which returns all possible strings resulting in a given string by P . That is, given an observable string $s \in \Gamma^*$, we have that $P(s') = s$ for all strings $s' \in P^{-1}(s)$.

2.2 Degree of Opacity

We measure to what extent secret states are obscured from adversaries by using the notions of degree of opacity. Although degree of opacity in the original work^[2] is defined by an arbitrary weighting function, in this paper we employ a specific example function in [2] based on the number of non-secret states indistinguishable with the secret states. That is, as the adversaries confuse more non-secret states with a secret state, the system is said to be more obscure.

Definition 2.1 (Degree of opacity^[2]) Consider a plant G , a set of secret states $Q_s \subseteq Q$, a subset of observable events $\Gamma \subseteq \Sigma_o$, and an observer $H(\Gamma)$ in (2). The degree of opacity for a secret state $q_s \in Q_s$ is given by

$$\Theta_{H(\Gamma)}(q_s) = \min_{A \in \mathcal{A} | q_s \in A} w(A), \quad (3)$$

where $w(A) = |A \setminus Q_s|$, i.e., the number of indistinguishable (non-secret) states in A .

Roughly speaking, in Definition 2.1 the degree of opacity of a secret state is defined by the number of non-secret states reachable by strings that look the same to an observer as strings leading to the secret state. We take the minimum of the number of confusing states, since the same secret state can appear in multiple states of the observer $H(\Gamma)$.

We also need a way to represent how we can achieve obfuscation in practical systems. In this paper, we suppose that some observable events in the system can be “concealed”, meaning that we can make some observable events unobservable to external entities. The subset of concealable events is denoted by $\Sigma_{co} \subseteq \Sigma_o$, and one of our main interests in this paper is to find $\Sigma'_{co} \subseteq \Sigma_{co}$ so that the system can achieve obfuscation against an eavesdropping attack. We will define such a condition in Subsection 3.1 in order to formulate our problem of achieving obfuscation and protection.

2.3 Secure Reachability

As a countermeasure against infiltration attacks, we employ the concept of secure reachability in the framework of *state protection*[†] for critical states in the plant. Specifically, protecting

[†]This notion in the original work was called *secret protection*, but in this paper we refer it as *state protection* to avoid confusion with the notion of secret states in opacity.

critical states means forcing all users (including malicious ones) to pass designated transitions before reaching critical states.

In the framework of state protection, the events in a plant G are partitioned into two types of events: Protectable events and unprotectable events. Protectable events model activities in the system to which protection technologies can be applied, e.g., authentication and authorization. We denote the (disjoint) subsets of protectable events and unprotectable events by $\Sigma_p \subseteq \Sigma$ and $\Sigma_{up} \subseteq \Sigma$ respectively. Namely, we have that $\Sigma = \Sigma_p \cup \Sigma_{up}$. Here, we assume that all protectable events are observable, namely $\Sigma_p \subseteq \Sigma_o$, modelling that a controller must observe events to apply protection technologies. As we mentioned in (1), we believe that this assumption is realistic, e.g., authentication gateways in cyber-physical systems must detect and identify communication packets in order to interfere the user activities and ask users to input their credentials. This assumption is also consistent with another limitation that concealed events cannot be protected.

Similar to the original work, our main tool of indicating which transitions in the plant should be protected is a *protection policy* defined as a mapping function from states to a subset of events. Here, it is important that we constrain ourselves to construct protection policies over the observer instead of the original plant, because we are simultaneously ensuring the obfuscation of secret states against the eavesdropping attacks. Moreover, this constraint effectively assures that in practical systems, we can still achieve the obfuscation of secret states even if the attackers further intrude into the security module of the system. In other words, we can keep our system from being completely compromised even if the attackers can somehow evade the protections. Mathematically, a protection policy over an observer is defined as follows.

Definition 2.2 (Protection policy over observer) Consider a plant G in (1), a subset of protectable events Σ_p , a subset of observable events $\Gamma \subseteq \Sigma_o$, and an observer $H(\Gamma)$ in (2). A protection policy over an observer $H(\Gamma)$ is a mapping function

$$\mathcal{P}_{H(\Gamma)} : Q \rightarrow 2^{\Sigma_p \cap \Gamma} \quad (4)$$

returning a set of protectable and observable events which should be protected at each state in $H(\Gamma)$.

Although the policy $\mathcal{P}_{H(\Gamma)}$ is our main tool against infiltration, it is also crucial to verify that the policy over observer is effective over the original plant G . In order for the effectiveness of $\mathcal{P}_{H(\Gamma)}$ for the plant G to be verified, we derive a protection policy over G from $\mathcal{P}_{H(\Gamma)}$ by a straightforward approach which protects (active) events specified by $\mathcal{P}_{H(\Gamma)}$ at all states in the observer state, that is, we drive a protection policy over G , denoted by \mathcal{P}_G , from $\mathcal{P}_{H(\Gamma)}$ by

$$(\forall q \in Q) \mathcal{P}_G(q) := \left(\bigcup_{A \in \{A' \in \mathcal{A} \mid q \in A'\}} \mathcal{P}_{H(\Gamma)}(A) \right) \cap \Xi(q), \quad (5)$$

where $\Xi : Q \rightarrow 2^\Sigma$ gives the set of active events at state q by $\Xi(q) = \{\sigma \in \Sigma \mid \delta(q, \sigma)\}$. For example, if a state q of G is included in a state A of $H(\Gamma)$ and $\mathcal{P}_{H(\Gamma)}$ specifies an event σ at A to protect, then σ at q will also be protected (if σ is active, or defined, at state q).

Using the protection policy over the plant in (5), we define in the following that a critical state is securely reachable with a specific number of transitions specified by a policy \mathcal{P}_G .

Definition 2.3 (Secure reachability^[37]) Consider a plant G in (1) and a protection policy \mathcal{P}_G in (5). Let $\Sigma_{\mathcal{P}_G} = \bigcup_{q \in Q} \mathcal{P}_G(q)$ be the subset of protectable events specified by \mathcal{P}_G . A critical state $q_c \in Q_c$ is said to be r -securely reachable with respect to \mathcal{P}_G if

$$(\forall s \in \Sigma^*) \delta(q_0, s) = q_c \implies s \in \underbrace{\Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^* \cdots \Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } r \text{ times}}. \quad (6)$$

That is, every string from the initial state to a critical state q_c contain at least r protectable events specified by $\Sigma_{\mathcal{P}_G}$.

Definition 2.3 emphasizes that critical states are still reachable after we apply protections to the specified transitions, which is one of the main differences of state protection from supervisory control.

3 Two-Pronged Security Problem

3.1 Problem Formulation

Before we formulate our problem of tackling eavesdropping and infiltration by employing degree of opacity and state protection, let us introduce two additional key notions, *protection cost levels* and *security requirements*, which the administrators usually need to take into account in practical systems.

First, as done in the previous work^[3], we consider that protectable events are grouped into n disjoint subsets, that is, $\Sigma_p = \dot{\bigcup}_{i=1}^n \Sigma_{cl,i}$. The cost level index i indicates how costly protectable events in $\Sigma_{cl,i}$ are to be actually protected in real systems. For instance, if the system can be equipped with three protection mechanisms: Passcode, IC card, and fingerprint authentication, then we have $n = 3$. In general, it is more costly to implement a protection mechanism using physical devices than passcodes, because physical devices require specific equipment to read the user identity while passcodes can be entered using keyboards, which are widely available peripherals. Moreover, the protection mechanisms with high security levels are usually more costly than those with low security levels. Observe that different protectable events could belong to the same cost level, e.g., both fingerprints and retina authentication require a biometrics scanner, depending on how they are implemented in practical systems. For simplicity, the cost levels are just indices and not quantitatively comparable. Namely, the cost of a single event in $\Sigma_{cl,i+1}$ is sufficiently larger than the total cost of all events in $\Sigma_{cl,i}$.

Furthermore, we consider that each of secret states and critical states has its own security requirement as to what extent it needs to be obscured/protected. Such requirements are defined as mapping functions $R_o : Q_s \rightarrow \mathbb{N}^+{}^\ddagger$ and $R_p : Q_c \rightarrow \mathbb{N}^+$ where R_o and R_p return a required (minimum) degree of opacity for a secret state and a required (minimum) number of protected transitions before reaching a critical state, respectively. We henceforth call R_o an *obfuscation requirement* and R_p a *protection requirement*.

[‡]We add superscript $+$ to explicitly indicate that \mathbb{N}^+ is a set of all positive integers that does not contain 0.

With the two types of requirements being presented, we define two crucial conditions which represent our goals of ensuring obfuscation and protection against attackers.

Definition 3.1 (Obfuscation) Consider a plant G and an obfuscation requirement R_o . Given a subset of concealed events $\Sigma'_{co} \subseteq \Sigma_{co}$, a plant G is said to be *obfuscated with respect to R_o and Σ'_{co}* if

$$(\forall q_s \in Q_s) \Theta_{H(\Sigma'_o)}(q_s) \geq R_o(q_s), \quad (7)$$

where $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$.

Definition 3.2 (Protection) Consider an automaton G and a protection requirement R_p . Given a protection policy \mathcal{P}_G , a plant G is said to be *protected with respect to R_p and \mathcal{P}_G* if every critical state $q_c \in Q_c$ is $R_p(q_c)$ -securely reachable with respect to \mathcal{P}_G , namely,

$$(\forall q_c \in Q_c)(\forall s \in \Sigma^*)\delta(q_0, s) = q_c \implies s \in \underbrace{\Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^* \cdots \Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R_p(q_c) \text{ times}}. \quad (8)$$

In words, the condition (7) requires that for every secret state in G , the degree of opacity is greater than or equal to the required value given by R_o . On the other hand, Definition 3.2 states that a plant G is protected if every critical state q_c is $R_p(q_c)$ -securely reachable, that is, there are at least $R_p(q_c)$ protectable events in every trajectory from the initial state to the critical state.

Since we constrain ourselves to the construction of protection policy $\mathcal{P}_{H(\Gamma)}$ in (4) over an observer $H(\Gamma)$ for a given subset of observable events $\Gamma \subseteq \Sigma_o$ instead of its original plant G , we also need another property with $\mathcal{P}_{H(\Gamma)}$ corresponding to Definition 3.2.

Definition 3.3 (Feasible protection) Consider a plant G , a subset of observable events $\Gamma \subseteq \Sigma_o$ its observer $H(\Gamma)$ in (2), a protection requirement R_p , and a protection policy $\mathcal{P}_{H(\Gamma)}$ over $H(\Gamma)$. Derive \mathcal{P}_G from $\mathcal{P}_{H(\Gamma)}$ by (5). A plant G is said to be *feasibly protected with respect to R_p and $\mathcal{P}_{H(\Gamma)}$* if every critical state $q_c \in Q_c$ in G is $R_p(q_c)$ -securely reachable with respect to \mathcal{P}_G .

Definition 3.3 is quite similar to Definition 3.2, and the only difference is that Definition 3.3 is defined for $\mathcal{P}_{H(\Gamma)}$ instead of \mathcal{P}_G which is indirectly given by (5). In other words, when we have a protection policy with partial observation, we say even under such a restricted protection policy a plant G can still achieve the protection goal of satisfying R_p .

We now have the properties of obfuscation and protection to formulate our goal of finding a pair of concealed event subset and protection policy that satisfy R_o and R_p simultaneously. However, it is also reasonable to question whether we can compare two different solutions and determine a “better” one for the same system. To determine what solution is “good” in this context, we define a binary relation of two solutions which takes both properties of obfuscation and protection into account rather than comparing that two properties independently, as a natural consideration to address eavesdropping and infiltration at the same time by finding solutions such that both R_o and R_p are satisfied. Although some solutions may end up being incomparable in this way, such solutions are still acceptable as long as the requirements of obfuscation and protection are satisfied.

While one might come up with various definitions of comparing two solutions to implement our methodology in practical systems, specifically in this paper, we define that a solution is better than another one if as a result the minimum degree of opacity is no lower and the maximum cost level is no higher than the other one. First for convenience, let us denote the minimum degree of opacity with respect to an observer $H(\Gamma)$ and the maximum cost level of a protection policy $\mathcal{P}_{H(\Gamma)}$ by

$$d_{H(\Gamma)} := \min_{q_s \in Q_s} \Theta_{H(\Gamma)}(q_s)$$

and

$$k_{\mathcal{P}_{H(\Gamma)}} := \max\{i \in [1, n] \mid \Sigma_{\mathcal{P}_{H(\Gamma)}} \cap \Sigma_{cl,i} \neq \emptyset\}, \quad (9)$$

respectively. Recall that protectable events are partitioned into n subsets of cost levels, which is why the maximum value of $\mathcal{P}_{H(\Gamma)}$ is n .

Given the aforementioned notations, we define a binary relation of two pairs of concealed event subset and protection policy as follows.

Definition 3.4 (Policy comparison) Consider a plant G , a subset of observable events Σ_o , a subset of concealable events $\Sigma_{co} \subseteq \Sigma_o$, two subsets of concealed events $\Sigma_{co,1} \subseteq \Sigma_{co}$ and $\Sigma_{co,2} \subseteq \Sigma_{co}$, and two protection policies $\mathcal{P}_{H(\Sigma'_{o,1})}$ and $\mathcal{P}_{H(\Sigma'_{o,2})}$, where $\Sigma'_{o,1} = \Sigma_o \setminus \Sigma_{co,1}$ and $\Sigma'_{o,2} = \Sigma_o \setminus \Sigma_{co,2}$. We denote \leq as a binary relation on two pairs $(\Sigma_{co,1}, \mathcal{P}_{H(\Sigma'_{o,1})})$ and $(\Sigma_{co,2}, \mathcal{P}_{H(\Sigma'_{o,2})})$ that holds if

$$d_{H(\Sigma'_{o,1})} \leq d_{H(\Sigma'_{o,2})} \wedge k_{\mathcal{P}_{H(\Sigma'_{o,1})}} \geq k_{\mathcal{P}_{H(\Sigma'_{o,2})}}. \quad (10)$$

This means that if $(\Sigma_{co,1}, \mathcal{P}_{H(\Sigma'_{o,1})}) \leq (\Sigma_{co,2}, \mathcal{P}_{H(\Sigma'_{o,2})})$, then the second pair $(\Sigma_{co,2}, \mathcal{P}_{H(\Sigma'_{o,2})})$ is better than (or just as good as) the first pair $(\Sigma_{co,1}, \mathcal{P}_{H(\Sigma'_{o,1})})$.

By defining the binary relation in Definition 3.4, we are also able to state that one pair is *locally optimal* in terms of the degree of opacity and the protection cost level.

Definition 3.5 (Local optimality) Consider a plant G , a subset of observable events Σ_o , an obfuscation requirement R_o , a protection requirement R_p , and a pair of a concealed event subset and a protection policy $M_1 = (\Sigma_{co,1}, \mathcal{P}_{H(\Sigma'_{o,1})})$ where $\Sigma'_{o,1} = \Sigma_o \setminus \Sigma_{co,1}$. Assume that G is obfuscated with respect to R_o and $\Sigma_{co,1}$ and G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_{o,1})}$. The policy pair M_1 is locally optimal if for any other pair $M_2 = (\Sigma_{co,2}, \mathcal{P}_{H(\Sigma'_{o,2})})$ (where $\Sigma'_{o,2} = \Sigma_o \setminus \Sigma_{co,2}$) such that G is obfuscated with respect to R_o and $\Sigma_{co,2}$ and G is protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_{o,2})}$, it does not hold that $M_1 \leq M_2$.

With the aforementioned definitions, we are ready to formulate our problem of finding a policy pair such that both requirements R_o and R_p are satisfied.

Proposition 3.6 (Two-pronged state security problem) *Given a plant G , an obfuscation requirement R_o , a protection requirement R_p , a subset of observable events Σ_o , and a subset of concealable events $\Sigma_{co} \subseteq \Sigma_o$, find a pair of concealed event subset and protection policy $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ (where $\Sigma'_{co} \subseteq \Sigma_{co}$ and $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$) such that*

- A1. G is obfuscated with respect to R_o and Σ'_{co} ; and

A2. G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$; and

A3. $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ is locally optimal.

A solution to Problem 3.6 may not be unique, since there could be more than one pair such that properties A1–A3 hold, due to the existence of incomparable pairs with the binary relation in Definition 3.4. In other words, a set of solutions is not totally ordered but partially ordered according to the relation \leq . Moreover, one pair may be strictly better than another one due to

- 1) The degree of opacity being higher and the protection policy being the same; or
- 2) The protection policy resulting in lower cost but the degree of opacity being the same; or
- 3) Both the degree of opacity being higher and the protection cost being lower.

Example 3.7 Let us consider an example system in Figure 1. Suppose that the system contains one secret state q_4 and two critical states q_4 and q_7 , i.e., we have $Q_s = \{q_4\}$ and $Q_c = \{q_4, q_7\}$. For example in practical systems, q_4 could be a service for a finance department, in which any unauthorized access should be forbidden, and the employees in other departments should not be able to know that the financial service at q_4 is currently running. The critical state q_7 , on the other hand, is an internal service which all employees can access as long as they provide proof of their identity. Also, assume that we can conceal the events in $\Sigma_{co} = \{\sigma_1, \sigma_3, \sigma_4\}$ and let the set of protectable events be $\Sigma_p = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. The system does not have any unobservable events at first, i.e., $\Sigma_{uo} = \emptyset$. Let us simply partition the protectable events into three cost levels, namely, $\Sigma_{cl,1} = \{\sigma_1, \sigma_2\}$, $\Sigma_{cl,2} = \{\sigma_3\}$, and $\Sigma_{cl,3} = \{\sigma_4\}$. Finally, consider that we are required to satisfy an obfuscation requirement $R_o(q_4) = 1$ and a protection requirement $R_p(q_4) = 1, R_p(q_7) = 2$.

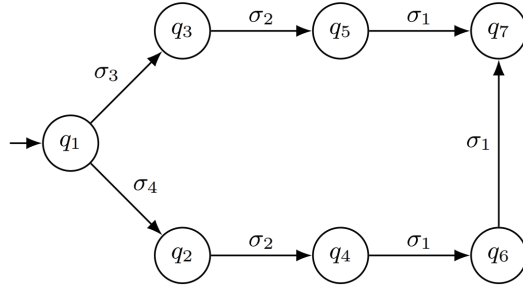


Figure 1 Example plant G_1

Our goal is to find a subset of concealable events $\Sigma'_{co} \subseteq \Sigma_{co}$ and a protection policy \mathcal{P}_H so that all the three properties in Problem 3.6 are satisfied. Indeed, we cannot satisfy property A1 if no events are concealed, namely, $\Sigma'_{co} = \emptyset$, since $H = \text{Obs}(G, \Sigma_o) = G$ in this case and $\Theta_{\Sigma_o \setminus \Sigma'_{co}}(q_4) = 0 < R_o(q_4)$. Similarly, we cannot satisfy property B2 if we conceal all events in Σ_{co} , namely $\Sigma'_{co} = \Sigma_{co}$, because \mathcal{P}_H can only protect observable (and not concealed) events, resulting in the critical state q_7 being unable to be 2-securely reachable.

Notably, we can have two incomparable solutions for this example system. First, let us pick $\Sigma'_{co} = \{\sigma_1\}$.

This results in an observer $H_1 = H(\{\sigma_2, \sigma_3, \sigma_4\})$ in Figure 2 and $d_{H_1} = 2$. The resulting protection policy is

$$\mathcal{P}_{H_1}(\{q_1\}) = \{\sigma_3, \sigma_4\}, \quad \mathcal{P}_{H_1}(\{q_2\}) = \{\sigma_2\}, \quad \mathcal{P}_{H_1}(\{q_3\}) = \{\sigma_2\},$$

which yields $k_{\mathcal{P}_{H_1}} = 3$ due to protecting σ_4 .

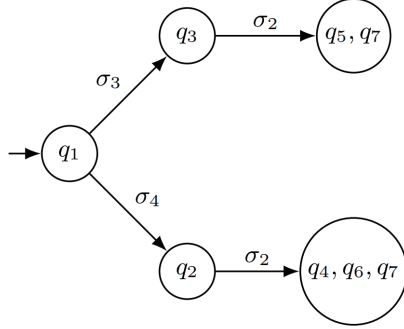


Figure 2 Observer $H_1 = H(\{\sigma_2, \sigma_3, \sigma_4\})$

Next, let $\Sigma'_{co} = \{\sigma_3, \sigma_4\}$ which results in an observer $H_2 = H(\{\sigma_1, \sigma_2\})$ in Figure 3 and yields

$$\mathcal{P}_{H_2}(\{q_1, q_2, q_3\}) = \{\sigma_2\}, \quad \mathcal{P}_{H_2}(\{q_4, q_5\}) = \{\sigma_1\}.$$

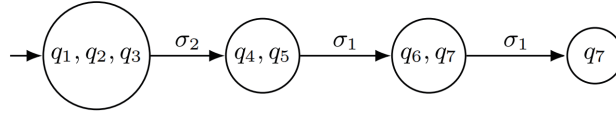


Figure 3 Observer $H_2 = H(\{\sigma_1, \sigma_2\})$

In this case, we have $d_{H_2} = 1$ and $k_{\mathcal{P}_{H_2}} = 1$. Observe that both solutions are incomparable according to Definition 3.4 because the former solution yields the higher (minimum) degree of opacity but the higher (maximum) cost level than the latter one, while both solutions satisfy all properties in Problem 3.6.

3.2 Policy Computation

The main idea to find a solution to Problem 3.6 is very simple. We first let all concealable-and-unprotectable events (i.e., $\Sigma_{co} \setminus \Sigma_p$) be unobservable, then compute an observer automaton of G , check whether there exists a protection policy that satisfies property A2, using the existing algorithm^[3] which returns a protection policy with minimum cost levels (if it exists). If yes, then we check if the degree of opacity for every secret state is greater than or equal to the

obfuscation requirement R_o . That is, we check if (7) holds with $\Sigma'_{co} = \Sigma_{co} \setminus \Sigma_p$. If the requirement R_o is satisfied, i.e., property A1 holds, then we add a pair of Σ'_{co} and $\mathcal{P}_{H(\Sigma_o \setminus \Sigma'_{co})}$ to the candidate set of solutions. We repeat this procedure for all combinations of concealable-and-protectable events, namely all subsets in $2^{\Sigma_{co} \cap \Sigma_p}$. After aggregating all candidate pairs, we remove ones which violate property A3 from the candidate set. In summary, we gather solution candidates that satisfy A1 and A2 for all combinations (including the empty subset) of concealable-and-protectable events, and then we evict candidates that violate A3.

We present our algorithm to find a solution to Problem 3.6 in Algorithm 1 and its sub-routines in Algorithm 2. Procedures MRCMC, SPEC, and RCMC are adopted as is from [37]. Algorithm 1 computes a pair of a concealed event subset and a protection policy for every combination of concealable-and-protectable events. In each iteration of computing a pair, if the algorithm detects either type of policy does not exist for the temporary subset of protectable events Σ_{co} (i.e., Line 13 and Line 18), then it skips that subset and continues to the next subset in $2^{\Sigma_o \cap \Sigma_p}$. Roughly speaking, Algorithm 1 examines all possible capabilities of concealing and protecting events, by calling MRCMC and CHECKDOO so that properties A1 and A2 in Problem 3.6 are satisfied. Once the set of solution candidates POL is formed, from Line 26, the policy pairs violating property A3 are removed from POL , and then Algorithm 1 finally returns POL which only contains the policy pairs satisfying all properties in Problem 3.6.

In particular for property A2 of Problem 3.6, we conservatively derive a protection requirement R'_p for each observer state from R_p in Line 11 by taking the maximum of the required number of protected transitions among states contained in the observer state. That is, letting $\mathcal{A}_c = \{A \in \mathcal{A} \mid A \cap Q_c \neq \emptyset\}$ be the set of observer states which contains any critical states, we derive a conservative protection requirement R'_p from R_p by

$$(\forall A_c \in \mathcal{A}_c) \ R'_p(A_c) = \max_{q_c \in A_c} R_p(q_c). \quad (11)$$

For example, say $R_p(q_{c,1}) = 1$ and $R_p(q_{c,2}) = 2$. If an observer has a state $A_c = \{q_1, q_2, q_{c,1}, q_{c,2}\}$ where q_1 and q_2 are not critical states, then we have $R'_p(A_c) = \max\{0, 0, 1, 2\} = 2$.

The worst-time complexity of Algorithm 1 is $O(2^{|\Sigma_{co} \cap \Sigma_p| + |Q|} + \binom{2^{|\Sigma_{co} \cap \Sigma_p|}}{2})$ which comes from Lines 2, 7, 26, and 27. The exponential portion $2^{|\Sigma_{co} \cap \Sigma_p| + |Q|}$ is due to the iteration between Lines 2 and 22 as it enumerates all subsets in the powerset of $\Sigma_{co} \cap \Sigma_p$ and constructs an observer of G having the set of states Q . The combination portion, denoted by a binomial coefficient, is due to the iteration between Lines 26 and 36 which compares every combination of two pairs in POL in the worst case, with the largest possible POL whose size is $2^{|\Sigma_{co} \cap \Sigma_p|}$.

To prove that Algorithm 1 produces correct solutions to Problem 3.6, i.e., all the properties of Problem 3.6 are satisfied, we first need to show that by computing a protection policy $\mathcal{P}_{H(\Gamma)}$ for an observer of G , the requirement R_p for the original plant G is also satisfied, especially for property A2.

Algorithm 1 Try all

This algorithm produces a set of pairs of concealed event subsets and protection policies which solve Problem 3.6. The algorithm first enumerates all subsets of concealable-and-protectable events to examine whether both the obfuscation requirement and the protection requirement are satisfied. After gathering solution candidates, pairs which are not locally optimal are removed from candidates

Require: $G = (\Sigma, \Sigma_o, \Sigma_p, \Sigma_{co}, Q_s, Q_c, R_o, R_p)$

Ensure: POL or NOT_FOUND

$POL = \emptyset, \quad \Sigma_{uo} = \Sigma \setminus \Sigma_o$

for $\Sigma'_{co} \in 2^{\Sigma_{co} \cap \Sigma_p}$ **do**

$\Sigma'_{uo} = \Sigma_{uo} \cup (\Sigma_{co} \setminus \Sigma_p) \cup \Sigma'_{co}$ \triangleright Make all concealable-and-unprotectable events and concealed events unobservable.

$\Sigma'_o = \Sigma_o \setminus \Sigma'_{uo}$

$\Sigma'_p = \Sigma_p \setminus \Sigma'_{co}$

\triangleright Make concealed protectable events unprotectable.

$\Sigma'_{up} = \Sigma \setminus \Sigma'_p$

$H(\Sigma'_o) = (\mathcal{A}, \Sigma'_o, -, -)$

$\mathcal{P}_{H(\Sigma'_o)}(A) = \emptyset$ for all $A \in \mathcal{A}$ \triangleright Initialize the protection policy for all observer states as an empty set.

$\mathcal{A}_c = \{A \in \mathcal{A} \mid A \cap Q_c \neq \emptyset\}$

for $A_c \in \mathcal{A}_c$ **do**

$R'_p = \max_{q_c \in A_c} R_p(q_c)$ \triangleright Using (11), compute a conservative protection requirement an observer state A_c containing critical states.

$\mathcal{P}' = \text{MRCMC}(H(\Sigma'_o), \Sigma'_p, A_c, R'_p)$ \triangleright Compute a protection policy such that A_c is R'_p -securely reachable using non-concealed protectable events in Σ'_p . The policy \mathcal{P}' is Null if A_c cannot be protected.

if \mathcal{P}' is Null **then**

Go to Line 2

end if

$\mathcal{P}_{H(\Sigma'_o)}(A) = \mathcal{P}_{H(\Sigma'_o)}(A) \cup \mathcal{P}'(A)$ for all $A \in \mathcal{A}$

end for

if $\text{CHECKDOO}(H(\Sigma'_o), Q_s, R_o)$ is False **then** \triangleright Check if the degree of opacity of every observer state containing any critical states is greater than or equal to the obfuscation requirement R_o . If not, CHECKDOO returns False.

Go to Line 2

end if

$POL = POL \cup \{(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})\}$

\triangleright Add a solution candidate to the set POL .

end for

if $POL = \emptyset$ **then**

return NOT_FOUND

\triangleright No solution candidates found.

end if

for $(\Sigma_{co,1}, \mathcal{P}_{H,1}) \in POL$ **do**

for $(\Sigma_{co,2}, \mathcal{P}_{H,2}) \in POL \setminus \{(\Sigma_{co,1}, \mathcal{P}_{H,1})\}$ **do**

if $(\Sigma_{co,1}, \mathcal{P}_{H,1}) \leq (\Sigma_{co,2}, \mathcal{P}_{H,2})$ **then**

$POL = POL \setminus \{(\Sigma_{co,1}, \mathcal{P}_{H,1})\}$ \triangleright The solution candidate is not locally optimal, so remove it from POL .

Go to Line 26

else if $(\Sigma_{co,2}, \mathcal{P}_{H,2}) \leq (\Sigma_{co,1}, \mathcal{P}_{H,1})$ **then**

$POL = POL \setminus \{(\Sigma_{co,2}, \mathcal{P}_{H,2})\}$ \triangleright The solution candidate is not locally optimal, so remove it from POL .

Go to Line 27

end if

end for

end for

return POL

Algorithm 2 Subroutines

```

procedure CHECKDOO( $H(\Gamma)$ ,  $Q_s$ ,  $R_o$ )      ▷ This checks if the degree of opacity of every secret state is
greater than or equal to the opacity requirement.
  for  $q_s \in Q_s$  do
    if  $\Theta_{H(\Gamma)}(q_s) < R_o(q_s)$  then
      return False
    end if
  end for
  return True
end procedure

procedure MRCMC( $G$ ,  $\Sigma_p$ ,  $q_c$ ,  $r$ )      ▷ This produces a protection policy for a plant
 $G$  which specifies at least  $r$  protectable events in every path from the initial state to a critical state  $q_c$  with
the minimum cost level due to RCMC.
   $G_1 = G = (Q, \rightarrow, \cdot)$ 
  for  $j \in [1, r]$  do
     $G_{K,j} = \text{SPEC}(G_j, q_c)$ 
     $M_j = \text{RCMC}(G_j, G_{K,j}, \Sigma_p)$ 
    if  $M_j \neq \text{Null}$  then
      Derive  $\mathcal{P}_j$  from  $M_j$ 
      if  $j < r$  then
        Form  $G_{j+1}$  from  $\mathcal{P}_j$ 
      end if
    else
      return Null
    end if
  end for
   $\mathcal{P}(q) := \bigcup_{j=1}^r \mathcal{P}_j(q)$  for all  $q \in Q$ 
  return  $\mathcal{P}$ 
end procedure

procedure SPEC( $G$ ,  $q_c$ ) ▷ This produces a specification automaton by removing a critical state  $q_c$  from  $G$ 
and transitions to and from  $q_c$ .
   $\delta_K(q) = \begin{cases} \text{undefined} & \text{if } q = q_c \vee \delta(q) = q_c \\ \delta(q) & \text{otherwise} \end{cases}$ 
   $G_K = (Q \setminus \{q_c\}, \Sigma, \delta_K, q_0)$ 
  return  $G_K$ 
end procedure

procedure RCMC( $G$ ,  $G_K$ ,  $\Sigma_p$ ) ▷ This produces a supervisor which specifies protectable events with the
minimum cost level.
   $K = L(G_K)$ 
  for  $k = 1, 2, \dots, n$  do
     $\Sigma_{p,k} = \bigcup_{j=1}^k \Sigma_{cl,j} \cap \Sigma_p$ 
    Compute a supervisor  $M$  s.t.  $L(M) = \sup \downarrow(K)$  w.r.t.  $\Sigma_{p,k}$ 
    if  $M$  is nonempty then
      return  $M$ 
    end if
  end for
  return Null
end procedure

```



Lemma 3.8 Consider a plant G , a subset of observable events $\Gamma \subseteq \Sigma_o$, an observer $H(\Gamma) = (\mathcal{A}, \Gamma, \delta_{H(\Gamma)}, A_0)$, and a protection requirement R_p for G . Given a protection policy $\mathcal{P}_{H(\Gamma)}$ for $H(\Gamma)$, if $H(\Gamma)$ is protected with respect to a conservative protection requirement R'_p in (11) and $\mathcal{P}_{H(\Gamma)}$, then by constructing a protection policy \mathcal{P}_G for G as in (5), the original plant G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Gamma)}$.

Proof Let $\mathcal{A}_c = \{A \in \mathcal{A} \mid A \cap Q_c \neq \emptyset\}$ be the subset of observer states which contain any critical states. Since the condition (8) holds for $H(\Gamma)$ and R'_p , we have that

$$(\forall A_c \in \mathcal{A}_c)(\forall s \in \Gamma^*) \delta_{H(\Gamma)}(A_0, s) = A_c \implies s \in \underbrace{\Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^*}_{\Sigma_{\mathcal{P}_{H(\Gamma)}} \text{ appears } R'_p(A_c) \text{ times}}. \quad (12)$$

Note that all events in the observer $H(\Gamma)$ are observable, so in (12) we consider all strings in Σ_o^* instead of Σ^* , and it is equivalent to

$$(\forall s \in \Gamma^*) \delta_{H(\Gamma)}(A_0, s) \in \mathcal{A}_c \implies s \in \underbrace{\Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^*}_{\Sigma_{\mathcal{P}_{H(\Gamma)}} \text{ appears } R'_p(\delta_{H(\Gamma)}(A_0, s)) \text{ times}}. \quad (13)$$

Expressions (12) and (13) mean that if an observable string s leads $H(\Gamma)$ to an observer state which contains any critical states of G , then s contains at least the number of protectable events required by R'_p .

By the definition of \mathcal{A}_c , we have that

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies \delta_{H(\Gamma)}(A_0, s) \in \mathcal{A}_c.$$

Thus, by syllogism, (13) implies that

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies s \in \underbrace{\Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_{H(\Gamma)}} \Gamma^*}_{\Sigma_{\mathcal{P}_{H(\Gamma)}} \text{ appears } R'_p(\delta_{H(\Gamma)}(A_0, s)) \text{ times}}. \quad (14)$$

Since we know that $\Sigma_{\mathcal{P}_{H(\Gamma)}} = \Sigma_{\mathcal{P}_G}$ from (5), it holds that by (14),

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies s \in \underbrace{\Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R'_p(\delta_{H(\Gamma)}(A_0, s)) \text{ times}}. \quad (15)$$

From (11), we naturally have the following property of the conservative requirement R'_p and the original requirement R_p .

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies R_p(q_c) \leq R'_p(\delta_{H(\Gamma)}(A_0, s)). \quad (16)$$

Since $\Sigma_{\mathcal{P}_G} = \Sigma_{\mathcal{P}_{H(\Gamma)}} \subseteq \Gamma$ by (4) and (5), for any $r, r' \in \mathbb{N}^+$ where $r \leq r'$, it holds that

$$\underbrace{\Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } r \text{ times}} \supseteq \underbrace{\Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } r' \text{ times}}.$$

In words, for any string $s \in \Gamma^*$, if s contains at least r' protected events, then s contains at least r protected events, and the converse is not true. Hence, from (15) and (16), we have that

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies s \in \underbrace{\Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^* \cdots \Gamma^* \Sigma_{\mathcal{P}_G} \Gamma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R_p(q_c) \text{ times}}. \quad (17)$$

Expressions (15) and (17) mean that if a string s in the observer H leading to a secret state q_c contains at least $R'_p(\delta_{H(\Gamma)}(A_0, s))$ protected events, then s contains at least $R_p(q_c)$ protected events because $R_p(q_c) \leq R'_p(\delta_{H(\Gamma)}(A_0, s))$.

Next, let $P : \Sigma^* \rightarrow \Gamma^*$ be a natural projection and $P^{-1} : \Gamma^* \rightarrow 2^{\Sigma^*}$ be the inverse of P . From (17), we have that

$$(\forall q_c \in Q_c)(\forall s \in \Gamma^*) q_c \in \delta_{H(\Gamma)}(A_0, s) \implies P^{-1}(s) \cap L(G) \subseteq \underbrace{\Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^* \cdots \Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R_p(q_c) \text{ times}}. \quad (18)$$

In words, since we know that a string s in $H(\Gamma)$ contains at least $R_p(q_c)$ protected events, the strings in G projected to s should also contain at least $R_p(q_c)$ protected events. Thus (18) is equivalent to

$$(\forall q_c \in Q_c)(\forall t \in \Sigma^*) q_c \in \delta_{H(\Gamma)}(A_0, P(t)) \implies t \in \underbrace{\Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^* \cdots \Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R_p(q_c) \text{ times}}. \quad (19)$$

Note that string t in (19) can contain unobservable events.


By the construction of observer $H(\Gamma)$, we have that

$$(\forall q_c \in Q_c)(\forall t \in \Sigma^*) \delta(q_0, t) = q_c \implies q_c \in \delta_{H(\Gamma)}(A_0, P(t)). \quad (20)$$

That is, if a critical state q_c is reachable in G by a string t , then an observer state containing q_c should be reachable by a projected string $P(t)$.

Thus, from (19) and (20) by syllogism, it holds that

$$(\forall q_c \in Q_c)(\forall t \in \Sigma^*) \delta(q_0, t) = q_c \implies t \in \underbrace{\Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^* \cdots \Sigma^* \Sigma_{\mathcal{P}_G} \Sigma^*}_{\Sigma_{\mathcal{P}_G} \text{ appears } R_p(q_c) \text{ times}}.$$

This is the same as (8), and recall that we derive \mathcal{P}_G from $\mathcal{P}_{H(\Gamma)}$ via (5). Therefore, as defined in Definition 3.3, if an observer $H(\Gamma)$ is protected with respect to R'_p and $\mathcal{P}_{H(\Gamma)}$, then the original plant G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Gamma)}$. 

The converse of Lemma 3.8 cannot be proved because the converse of (20) does not hold. This is the nature of the observer construction, that is, we cannot restore the original plant when we only know its observer, because some information of the structure of the plant is lost due to unobservable events.

Using Lemma 3.8, we will show that the solutions to Problem 3.6 produced by Algorithm 1 are correct. Namely, we will show that

- 1) If a solution to Problem 3.6 exists, then Algorithm 1 will not produce **NOT_FOUND**; and
- 2) When Algorithm 1 produces pairs of concealed events together with protection policies, each of those pairs is indeed a solution to Problem 3.6.

Theorem 3.9 *The pairs of concealed events and protection policies produced by Algorithm 1 are solution to Problem 3.6.*

Proof Algorithm 1 terminates since $2^{\Sigma_{co} \cap \Sigma_p}$, the set of observer states \mathcal{A}_c in every iteration, and POL are finite.

Assume that there is no solution to Problem 3.6. This means that for given G , R_o , and R_p , there does not exist any candidate that satisfies both properties A1 and A2 simultaneously. Note that property A3 holds as long as there is at least one pair such that both A1 and A2 hold.

If property A1 does not hold, then Procedure CHECKDOO at Line 18 of Algorithm 1 returns False and Algorithm 1 continues to the next iteration without adding a policy pair to POL . From Lemma 3.8, if G is not feasibly protected, then the observer $H(\Sigma'_o)$ of G is not protected. In that case, from Definition 3.3, there exists a state $A_c \in \mathcal{A}_c$ such that MRCMC in Line 12 returns Null, and thus Algorithm 1 continues to the next iteration without adding a policy pair to POL .

Hence, if no pair satisfies both A1 and A2 simultaneously, then POL remains empty, so Algorithm 1 returns NOT_FOUND at Line 24. Therefore, Algorithm 1 always returns NOT_FOUND if there exists no solution to Problem 3.6.

Conversely, if POL is nonempty, then all policy pairs in POL satisfy A1 and A2, and Algorithm 1 continues to Line 26. Between Lines 26 and 36, all pairs that violate A3 are removed from $A3$ by Lines 28 and 31. Therefore, all policy pairs in POL satisfy A1, A2, A3. ■

Example 3.10 Let us revisit the example system in Example 3.7. The power set $2^{\Sigma_{co} \cap \Sigma_p}$ includes both $\{\sigma_1\}$ and $\{\sigma_3, \sigma_4\}$, and the set of candidate solutions POL after Line 22 contains the solution pairs we presented in Example 3.7. In particular, property A3 holds since these two solutions are incomparable according to Definition 3.4.

4 Special Version of the Two-Pronged Security Problem

As seen in Section 3, finding solutions to Problem 3.6 requires us to examine all subsets in $2^{\Sigma_{co} \cap \Sigma_p}$, resulting in two exponential portions for the worst-time complexity. In this section, we try to remove the exponential portion of $2^{\Sigma_{co} \cap \Sigma_p}$ by imposing additional constraints on Problem 3.6. The main idea is to gradually reveal/hide protectable events one by one instead of enumerating all subsets of $2^{\Sigma_{co} \cap \Sigma_p}$.

4.1 Algorithm that Uses Full Concealment First

Let us define a couple of notions before formulating the special version of Problem 3.6. The first definition will be used to impose a total order on protectable events according to their cost levels. Specifically, we need protectable events to be totally ordered according to their cost levels so that we can deterministically pick a protectable event to be revealed or hidden. For example, let us say $\Sigma_p = \{\sigma_1, \sigma_2\}$. If two protectable events belong to the same cost level, e.g., $\Sigma_{cl,i} = \{\sigma_1, \sigma_2\}$, then there are three options of events to reveal/hide, i.e., $\{\sigma_1\}$, $\{\sigma_2\}$, or $\{\sigma_1, \sigma_2\}$, which we want to avoid.

Definition 4.1 (Event cost comparison) Consider two protectable events σ_1 and σ_2 . We denote $<$ as a binary relation on σ_1 and σ_2 that holds if $\sigma_1 \in \Sigma_{cl,i}$ and $\sigma_2 \in \Sigma_{cl,j}$ where $i, j \in [1, n]$ and $i < j$, i.e., $\sigma_1 < \sigma_2$ simply means that the event σ_2 is more costly than event σ_1 .

Next, we define a key condition which will replace property A3 of Problem 3.6. The main idea is to only focus on the protection cost levels, in order to prevent solutions from being incomparable and to determine the best pair of concealed events and a protection policy.

Definition 4.2 (Minimal feasible protection policy) Consider a plant G , a protection requirement R_p , a set of observable events $\Sigma_o \subseteq \Sigma$, a set of concealable events $\Sigma_{co} \subseteq \Sigma_o$, a subset of concealed events $\Sigma'_{co} \subseteq \Sigma_{co}$, and a protection policy $\mathcal{P}_{H(\Sigma'_o)} : Q \rightarrow 2^{\Sigma_p \cap \Sigma'_o}$ (where $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$) such that G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$. A protection policy $\mathcal{P}_{H(\Sigma'_o)}$ is said to be *minimally feasible* with respect to Σ'_{co} if for any other protection policy $\mathcal{P}'_{H(\Sigma'_o)} : Q \rightarrow 2^{\Sigma_p \cap \Sigma'_o}$ such that G is feasibly protected with respect to R_p and $\mathcal{P}'_{H(\Sigma'_o)}$, it holds that

$$k_{\mathcal{P}_{H(\Sigma'_o)}} \leq k_{\mathcal{P}'_{H(\Sigma'_o)}}. \quad (21)$$

That is, after concealing all events in Σ'_{co} , a policy $\mathcal{P}_{H(\Sigma'_o)}$ is minimally feasible if it does not yield unnecessarily costly events.

Using Definition 4.1 and Definition 4.2, we modify Problem 3.6 as follows.

Proposition 4.3 (Special two-pronged state security problem) Consider a plant G , a subset of observable events $\Sigma_o \subseteq \Sigma$, a subset of concealable events $\Sigma_{co} \subseteq \Sigma_o$, an obfuscation requirement R_o , and a protection requirement R_p . Assuming that Σ_p is totally ordered according to the binary relation $<$ in Definition 4.1, find a pair of a subset of concealed event and a protection policy $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ (where $\Sigma'_{co} \subseteq \Sigma_{co}$ and $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$) such that

- B1. G is obfuscated with respect to R_o and Σ'_{co} ; and
- B2. G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$; and
- B3. $\mathcal{P}_{H(\Sigma'_o)}$ is minimally feasible with respect to Σ'_{co} .

Similar to Problem 3.6, a solution to Problem 4.3 may not be unique, since property B3 does not impose any additional condition on the degree of opacity. It can be seen that Problem 4.3 is neither a restricted nor a relaxed version of Problem 3.6 but a “special” version, that is, we are not required to find a better solution in terms of the degree of opacity, but need to find the best solution for cost levels.

The central idea to find a solution to Problem 4.3 is similar to that of Problem 3.6, except we exploit that protectable events are totally ordered to deterministically find the best solution in terms of cost level according to property B3 without examining all subsets in the power set of $\Sigma_{co} \cap \Sigma_p$, while Problem 3.6 can result in incomparable solutions due to property A3.

Our algorithm to find a solution to Problem 4.3 is presented in Algorithm 3, sharing the same techniques of the same subroutines in Algorithm 2 with Algorithm 1. The main difference

between Algorithm 3 and Algorithm 1 is that Algorithm 3 begins with concealing all events in Σ_{co} and in each iteration, the current least costly (concealed-and-protectable) event is made observable on Line 17, instead of evaluating all subsets of $2^{\Sigma_{co} \cap \Sigma_p}$. Unlike Algorithm 1, thanks to MRCMC, it is not required to examine multiple candidate solutions in order to satisfy property B3.

Algorithm 3 Full concealment first

This algorithm produces a pair of a concealed event subset and a protection policy which solves Problem 4.3. The algorithm first conceals all concealable events in Σ_{co} , and then tests whether both the obfuscation requirement and the protection requirement are satisfied. If not, this algorithm makes the most costly event in $\Sigma_{co} \cap \Sigma_p$ observable, and then tests the requirements again. Once a pair which satisfies the requirements is found, then this algorithm returns that pair as a solution to Problem 4.3.

Require: $G = (\Sigma, \Sigma_o, \Sigma_{co}, \Sigma_p, Q_s, Q_c, R_o, R_p)$

Ensure: $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ or NOT_FOUND

```

 $\Sigma_{uo} = \Sigma \setminus \Sigma_o, \quad \Sigma'_{co} = \Sigma_{co},$ 
 $\Sigma'_{uo} = \Sigma_{uo} \cup \Sigma'_{co},$  ▷ Make all concealable events unobservable.
 $\Sigma'_o = \Sigma \setminus \Sigma'_{uo},$ 
 $\Sigma'_p = \Sigma_p \setminus \Sigma'_{co},$  ▷ Make concealed protectable events unprotectable.
 $\Sigma'_{up} = \Sigma \setminus \Sigma'_p,$ 
 $H(\Sigma'_o) = Obs(G, \Sigma'_o) = (\mathcal{A}, \Sigma'_o, \Sigma, \Sigma),$ 
 $\mathcal{P}_{H(\Sigma'_o)}(A) = \emptyset$  for all  $A \in \mathcal{A},$  ▷ Initialize the protection policy for all observer states as an empty set.
 $\mathcal{A}_c = \{A \in \mathcal{A} \mid A \cap Q_c \neq \emptyset\}$ 
for  $A_c \in \mathcal{A}_c$  do
   $R'_p = \max_{q_c \in A_c} R_p(q_c)$  ▷ Using (11), compute a conservative protection requirement for an observer
  state  $A_c$  containing critical states.
   $\mathcal{P}' = \text{MRCMC}(H(\Sigma'_o), \Sigma'_p, A, R'_p)$  ▷ Compute a protection policy such that  $A_c$  is  $R'_p$ -securely reachable
  using non-concealed protectable events in  $\Sigma'_p$ . The policy  $\mathcal{P}'$  is Null if  $A_c$  cannot be protected.
  if  $\mathcal{P}'$  is Null then
    if  $\Sigma'_{co} \cap \Sigma_p = \emptyset$  then
      return NOT_FOUND ▷ Terminate if there is no concealed event to reveal left.
    end if
     $\sigma_p = \min(\Sigma'_{co} \cap \Sigma_p, <),$ 
     $\Sigma'_{co} = \Sigma'_{co} \setminus \{\sigma_p\},$  ▷ Reveal the least costly concealed-and-protectable event.
     $\Sigma'_{uo} = \Sigma_{uo} \cup \Sigma'_{co},$  ▷ Make all concealed events unobservable.
     $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co},$ 
     $\Sigma'_p = \Sigma_p \setminus \Sigma_{co},$ 
     $\Sigma'_{up} = \Sigma \setminus \Sigma'_p.$ 
    Go to Line 6
  end if
   $\mathcal{P}_{H(\Sigma'_o)}(A) = \mathcal{P}_{H(\Sigma'_o)}(A) \cup \mathcal{P}'(A)$  for all  $A \in \mathcal{A}$ 
end for
if CHECKDOO( $H(\Sigma'_o), Q_s, R_o$ ) is False then ▷ Check if the degree of opacity
of every observer state containing any critical state is greater than or equal to the obfuscation requirement
 $R_o$ . If not, CHECKDOO returns False.
  return NOT_FOUND
end if
return  $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ 

```



The time complexity of Algorithm 3 in the worst case is $O(|\Sigma_{co} \cap \Sigma_p| \times 2^{|Q|})$. The exponential part $2^{|Q|}$ is due to the observer construction, and $|\Sigma_{co} \cap \Sigma_p|$ comes from the iteration between Lines 6 and 22 as the maximum number of the iterations is $|\Sigma_{co} \cap \Sigma_p|$. Although the complexity is still exponential, we conclude that Algorithm 3 is less computationally expensive than Algorithm 1.

To show that solutions produced by Algorithm 3 will solve Problem 4.3, we utilize Lemma 3.8 again.

Proposition 4.4 *A pair of a concealed event subset and a protection policy produced by Algorithm 3 is a solution to Problem 4.3.*

Proof Algorithm 3 terminates since $\Sigma_{co} \cap \Sigma_p$ is finite and all protectable events will eventually be removed from Σ'_{co} in Line 17.

If Algorithm 3 returns a pair, then it reaches Line 28. From MRCMC in Line 11 and the construction of $\mathcal{P}_{H(\Sigma'_o)}$ in Line 24, the observer $H(\Sigma'_o)$ is protected with respect to R'_p in Line 10 and $\mathcal{P}_{H(\Sigma'_o)}$. From Lemma 3.8 and Line 24, the original plant G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$ in Line 24. Thus, property B2 of Problem 4.3 is satisfied.

Next, since Algorithm 3 also returns a candidate subset of concealed event, CHECKDOO in Line 26 returns True. CHECKDOO ensures that the degree of opacity of every secret state q_s is greater than or equal to the requirement $R_o(q_s)$. Thus, by concealing all events in Σ'_{co} , (7) holds for R_o given to Algorithm 3. Therefore, property B1 of Problem 4.3 is satisfied by Σ'_{co} .

Finally, the cost level of protectable events specified by \mathcal{P}' from MRCMC on Line 11 is minimal (cf. [37]). That is, the condition in (21) holds for $\mathcal{P}_{H(\Sigma'_o)}$ in every iteration between Lines 9 and 25 of Algorithm 3. Thus, property B3 of Problem 4.3 is satisfied by $\mathcal{P}_{H(\Sigma'_o)}$.

Therefore, if Algorithm 3 returns a pair $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ where $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$, then Σ_{co} and $\mathcal{P}_{H(\Sigma'_o)}$ are a solution to Problem 4.3. \blacksquare

Observe that Algorithm 3 returns NOT_FOUND immediately after it confirms that CHECKDOO returns False. This is because the degree of opacity defined by the number of confusing (non-secret) states is monotonic in the special case where the set of observable events in each iteration is a subset of that in the previous iteration. In other words, if the degree of opacity with Σ'_{uo} is below the requirement R_o , then there is no chance for the degree of opacity to become larger than R_o , since Σ'_{uo} shrinks in each iteration. We show this property in the following proposition.

Proposition 4.5 *Consider a plant G and a set of secret states $Q_s \subseteq Q$. Let $\Sigma_{o,1} \subseteq \Sigma$ and $\Sigma_{o,2} \subseteq \Sigma$ be sets of observable events, and $H(\Sigma_{o,1}) = \text{Obs}(G, \Sigma_{o,1})$ and $H(\Sigma_{o,2}) = \text{Obs}(G, \Sigma_{o,2})$ be observers based on G , $\Sigma_{o,1}$, and $\Sigma_{o,2}$. It holds that for all $q_s \in Q_s$,*

$$\Sigma_{o,1} \subseteq \Sigma_{o,2} \implies \Theta_{H(\Sigma_{o,1})}(q_s) \geq \Theta_{H(\Sigma_{o,2})}(q_s).$$

Proof For $i \in \{1, 2\}$, let $P_i : \Sigma^* \rightarrow \Sigma_{o,i}^*$ be a natural projection and $P_i^{-1} : \Sigma_{o,i}^* \rightarrow 2^{\Sigma^*}$ be the inverse of P_i . From $\Sigma_{o,1} \subseteq \Sigma_{o,2}$, it holds that for all $s \in L(G)$,

$$P_1^{-1}(P_1(s)) \cap L(G) \supseteq P_2^{-1}(P_2(s)) \cap L(G). \quad (22)$$

In words, since P_1 erases more events than P_2 , P_1^{-1} produces more strings in $L(G)$ than P_2^{-1} . For convenience, let $B_i(s) = P_i^{-1}(P_i(s)) \cap L(G)$ for $i \in \{1, 2\}$, and $L_s(q) = \{s \in L(G) \mid \delta(q_0, s) = q\}$ for all $q \in Q$. From (22), for all $q_s \in Q_s$ and $s \in L_s(q_s)$, it holds that

$$\begin{aligned} \{q \in Q \setminus Q_s \mid (\exists s' \in L(G)) \delta(q_0, s') = q \wedge s' \in B_1(s)\} \\ \supseteq \{q \in Q \setminus Q_s \mid (\exists s' \in L(G)) \delta(q_0, s') = q \wedge s' \in B_2(s)\}. \end{aligned} \quad (23)$$

That is, for some string s that leads the plant G to a secret state, there are more strings in $L(G)$ that P_1 makes them look the same as s than P_2 does. This implies that there can be more non-secret states reached by a string $s' \in L(G)$ such that $P_1(s') = P_1(s)$ than that by a string $s'' \in L(G)$ such that $P_2(s'') = P_2(s)$. For convenience, let

$$E_i(q_s, s) = \{q \in Q \setminus Q_s \mid (\exists s' \in L(G)) \delta(q_0, s') = q \wedge s' \in B_i(s)\},$$

for all $q_s \in Q_s$, $s \in L_s(q_s)$, and $i \in \{1, 2\}$. From (23), for all $q_s \in Q_s$ and $s \in L_s(q_s)$, it holds that

$$|E_1(q_s, s)| \geq |E_2(q_s, s)|. \quad (24)$$

Thus, for all $q_s \in Q_s$, it holds that

$$\min_{s \in L_s(q_s)} |E_1(q_s, s)| \geq \min_{s \in L_s(q_s)} |E_2(q_s, s)|. \quad (25)$$

By the construction of $Obs(G, \Sigma_{o,i}) = (\mathcal{A}_i, \Sigma_{o,i}, \neg, -)$ for $i \in \{1, 2\}$, letting $\mathcal{A}'_i(q) = \{A \in \mathcal{A}_i \mid q \in A\}$ for $q \in Q$, it holds that for all $q_s \in Q_s$,

$$\min_{A_i \in \mathcal{A}'_i(q_s)} |A_i \setminus Q_s| = \min_{s \in L_s(q_s)} |E_i(q_s, s)|.$$

In words, this is because $E_i(q_s, s)$ represents how the construction of observers merges different states into one state, that is, states are merged if they can be reached by different strings in $L(G)$ which are projected to the same string. Hence, from (25), it holds that for all $q_s \in Q_s$,

$$\min_{A_1 \in \mathcal{A}'_1(q_s)} |A_1 \setminus Q_s| \geq \min_{A_2 \in \mathcal{A}'_2(q_s)} |A_2 \setminus Q_s|. \quad (26)$$

By Definition 2.1, for all $q_s \in Q_s$

$$\begin{aligned} \Theta_{H(\Sigma_{o,i})}(q_s) &= \min_{A_i \in \mathcal{A}_i \mid q_s \in A} |A_i \setminus Q_s| \\ &= \min_{A_i \in \mathcal{A}'_i(q_s)} |A_i \setminus Q_s|. \end{aligned}$$

Therefore, from (26), for all $q_s \in Q_s$,

$$\Sigma_{o,1} \subseteq \Sigma_{o,2} \implies \Theta_{H(\Sigma_{o,1})}(q_s) \geq \Theta_{H(\Sigma_{o,2})}(q_s). \quad (27)$$

The proof is **completed**. ■

Example 4.6 Let us consider an example plant in Figure 4. Suppose that G_2 contains two secret states q_6 and q_{10} , and critical states q_8 and q_{10} , that is, we have $Q_s = \{q_6, q_{10}\}$ and $Q_c = \{q_8, q_{10}\}$. Also, consider the set of concealable events $\Sigma_{co} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_5, \sigma_6, \sigma_7\}$, the set of protectable events $\Sigma_p = \{\sigma_1, \sigma_2, \sigma_5, \sigma_6, \sigma_7, \sigma_8\}$ which are partitioned into six levels $\Sigma_{cl,1} = \{\sigma_1\}$, $\Sigma_{cl,2} = \{\sigma_2\}$, $\Sigma_{cl,3} = \{\sigma_5\}$, $\Sigma_{cl,4} = \{\sigma_6\}$, $\Sigma_{cl,5} = \{\sigma_7\}$, $\Sigma_{cl,6} = \{\sigma_8\}$. In this example, there are no unobservable events (namely $\Sigma_{uo} = \emptyset$), and the requirements are given by $R_o(q_6) = 2$, $R_o(q_{10}) = 1$, $R_p(q_8) = 1$, and $R_p(q_{10}) = 2$.

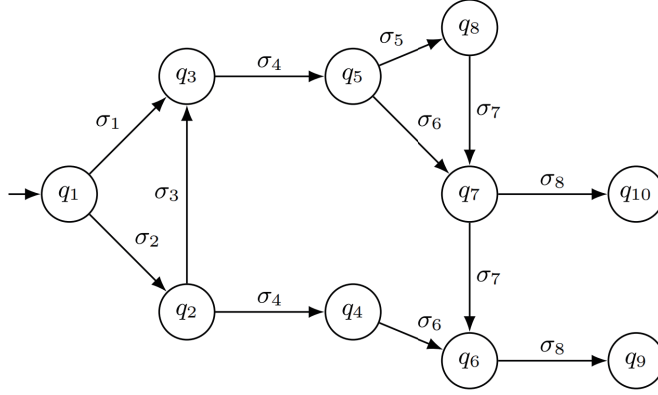


Figure 4 Example plant G_2

By running Algorithm 3, we obtain a solution $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ where

$$\begin{aligned} \Sigma'_{co} &= \{\sigma_3, \sigma_5, \sigma_6, \sigma_7\}, \quad \Sigma'_o = \Sigma_o \setminus \Sigma'_{co} = \{\sigma_1, \sigma_2, \sigma_4, \sigma_8\}, \quad \mathcal{P}_{H(\Sigma'_o)}(q_1) = \{\sigma_1, \sigma_2\}, \\ \mathcal{P}_{H(\Sigma'_o)}(\{q_5, q_6, q_7, q_8\}) &= \{\sigma_8\}, \quad \mathcal{P}_{H(\Sigma'_o)}(\{q_4, q_5, q_6, q_7, q_8\}) = \{\sigma_8\}. \end{aligned}$$

From (5), we can confirm that the original plant G_2 is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$, that is, q_8 and q_{10} are 1-securely reachable and 2-securely reachable, respectively. Observe that from $\mathcal{P}_{H(\Sigma'_o)}(\{q_5, q_6, q_7, q_8\}) = \{\sigma_8\}$, event σ_8 at state q_6 will be protected, even though protecting σ_8 at state q_6 serves no purpose since the only state reachable from q_6 by σ_8 is non-critical state q_9 . Such a policy, however, is still acceptable as all three conditions in Problem 4.3 are satisfied. That is, a policy which unnecessarily protects transitions is acceptable unless it increments the maximum cost level.

Figure 5 depicts the observer $H(\Sigma'_o)$ of G_2 computed in Algorithm 3 where $\Sigma'_o = \{\sigma_1, \sigma_2, \sigma_4, \sigma_8\}$. From states $\{q_5, q_6, q_7, q_8\}$ and $\{q_9, q_{10}\}$, the degrees of opacity of q_6 and q_{10} are 3 and 1, respectively. Since $\mathcal{P}_{H(\Sigma'_o)}$ specified event σ_8 , the maximum cost level invoked for this example is 6.

In fact, we can also employ Algorithm 1 to find a solution to Problem 4.3 by replacing condition (10) of the binary relation in Definition 3.4 with (21), so that Algorithm 1 does not compare, in Lines 28 and 31, policy pairs by the degree of opacity, but by the protection cost level.

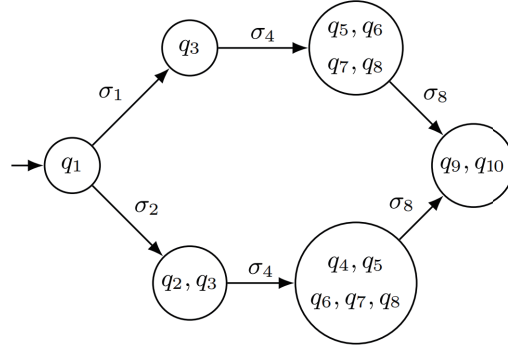


Figure 5 Observer $H(\Sigma'_o)$

Proposition 4.7 Denote \leq as a binary relation on two policy pairs $(-, \mathcal{P}_{H,1})$ and $(-, \mathcal{P}_{H,2})$ that holds if $k_{\mathcal{P}_{H,1}} \geq k_{\mathcal{P}_{H,2}}$. If Algorithm 1 returns a nonempty *POL*, namely not *NOT_FOUND*, then the policies of every pair in *POL* are solutions to Problem 4.3.

Proof Since properties A1 and A2 are the same as properties B1 and B2, respectively, by Theorem 3.9, the policies in *POL* satisfy B1 and B2. Moreover, since Algorithm 1 removes from *POL* all policy pairs that violate (21), the remaining pairs in *POL* satisfy B3. ■

Although adjusting Algorithm 1 in this way can yield a solution to Problem 4.3, the result inherits the same computational shortcomings as the original Algorithm 1. Thus, we presented Algorithm 3, which provides a computational improvement over Algorithm 1.

4.2 Algorithm that Uses Least Concealment First

In Algorithm 3, all concealable events are first labelled as unobservable events, and then it gradually reveals concealable-and-protectable events with the least cost level. It is, however, also natural to question whether an algorithm which first reveals all concealable-and-protectable events and then gradually hides them can produce the same result as Algorithm 3. To examine this question, we adjust Algorithm 3 and present the result as Algorithm 4.

The main differences from Algorithm 4 are that in Algorithm 4, the set of temporarily unobservable events Σ'_{uo} in Line 3 does not contain any protectable events, and the most costly protectable event will be hidden by Lines 19 and 21 in each iteration. The time complexity of Algorithm 4 in the worst case is the same as Algorithm 3, namely $O(|\Sigma_{co} \cap \Sigma_p| \times 2^{|Q|})$. As we did for Algorithm 3, we show that the policy pair produced by Algorithm 4 solves Problem 4.3.

Proposition 4.8 A pair of a concealed event set and a protection policy produced by Algorithm 4 is a solution to Problem 4.3.

Proof Proposition 4.8 can be proved in the same manner as Proposition 4.4. If Algorithm 4 returns a policy pair $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$, then Algorithm 4 does not terminate at Line 14. That is, there exists a pair $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ such that \mathcal{P}' in Line 12 is not Null and CHECKDoO in Line 18 is True. From MRCMC in Line 12 and the construction of $\mathcal{P}_{H(\Sigma'_o)}$ in Line 16, the observer $H(\Sigma'_o)$ is protected with respect to $\mathcal{P}_{H(\Sigma'_o)}$ and R'_p derived by (11) from R_p . By Lemma 3.8,

Algorithm 4 Least concealment first

This algorithm produces a pair of a concealed event subset and a protection policy which solves Problem 4.3. The algorithm first only conceals concealable-and-unprotectable events, and then tests whether both the obfuscation requirement and the protection requirement are satisfied. If not, this algorithm makes the most costly event in $\Sigma_{co} \cap \Sigma_p$ unobservable, and then tests the requirements again. Once a pair which satisfies the requirements is found, then this algorithm returns that pair as a solution to Problem 4.3.

Require: $G = (\Sigma, \Sigma_o, \Sigma_p, Q_s, Q_c, R_o, R_p)$

Ensure: $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ or NOT_FOUND

```

1:  $\Sigma_{uo} = \Sigma \setminus \Sigma_o$ 
2:  $\Sigma'_{co} = \Sigma_{co} \setminus \Sigma_p$   $\triangleright$  Conceal concealable-and-unprotectable events be unobservable.
3:  $\Sigma'_{uo} = \Sigma_{uo} \cup \Sigma'_{co}$ 
4:  $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$ 
5:  $\Sigma'_p = \Sigma_p$ 
6:  $\Sigma'_{up} = \Sigma \setminus \Sigma'_p$ 
7:  $H(\Sigma'_o) = Obs(G, \Sigma'_o) = (\mathcal{A}, \Sigma'_o, \Sigma'_p)$ 
8:  $\mathcal{P}_{H(\Sigma'_o)}(A) = \emptyset$  for all  $A \in \mathcal{A}$   $\triangleright$  Initialize the protection policy for all observer states as an empty set.
9:  $\mathcal{A}_c = \{A \in \mathcal{A} \mid A \cap Q_c \neq \emptyset\}$ 
10: for  $A_c \in \mathcal{A}_c$  do
11:    $R'_p = \max_{q_c \in A_c} R_p(q_c)$   $\triangleright$  Using (11), compute a conservative protection requirement for an observer state  $A_c$  containing critical states.
12:    $\mathcal{P}' = \text{MRCMC}(H(\Sigma'_o), \Sigma'_p, A_c, R'_p)$   $\triangleright$  Compute a protection policy such that  $A_c$  is  $R'_p$ -securely reachable using non-concealed protectable events in  $\Sigma'_p$ . The policy  $\mathcal{P}'$  is Null if  $A_c$  cannot be protected.
13:   if  $\mathcal{P}'$  is Null then
14:     return NOT_FOUND
15:   end if
16:    $\mathcal{P}_{H(\Sigma'_o)}(A) = \mathcal{P}_{H(\Sigma'_o)}(A) \cup \mathcal{P}'(A)$  for all  $A \in \mathcal{A}$ 
17: end for
18: if CHECKDoO( $H(\Sigma'_o), Q_s, R_o$ ) is False then  $\triangleright$  Check if the degree of opacity of every observer state containing any critical state is greater than or equal to the obfuscation requirement  $R_o$ . If not, CHECKDoO returns False.
19:    $\sigma_p = \max(\Sigma_p \setminus \Sigma'_{co}, <)$ 
20:    $\Sigma'_{co} = \Sigma'_{co} \cup \{\sigma_p\}$   $\triangleright$  Hide the most costly concealable-and-protectable event.
21:    $\Sigma'_{uo} = \Sigma_{uo} \cup \Sigma'_{co}$   $\triangleright$  Make all concealed events unobservable.
22:    $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$ 
23:    $\Sigma'_p = \Sigma_p \setminus \Sigma'_{co}$ 
24:    $\Sigma'_{up} = \Sigma \setminus \Sigma'_p$ 
25:   Go to line 7
26: end if
27: return  $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ 

```

the plant G is feasibly protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$ in Line 16. Thus, property B2 of Problem 4.3 is satisfied.

Since CHECKDOO in Line 18 is True, by concealing all events in Σ'_{co} , property B1 of Problem 4.3 holds.

Finally, the construction of $\mathcal{P}_{H(\Sigma'_o)}$ in Line 16 which only merges the protection policies by MRCMC, the cost level of protectable events which are in the subsets returned by $\mathcal{P}_{H(\Sigma'_o)}$ for all states of $H(\Sigma'_o)$ is minimum. Thus, the condition in (21) holds for $\mathcal{P}_{H(\Sigma'_o)}$ in every iteration between Lines 10 and 17 of Algorithm 4. Hence, property B3 of Problem 4.3 is satisfied.

Therefore, if Algorithm 4 returns a pair, then it is a solution to Problem 4.3. \blacksquare

Similar to Algorithm 3, we can return NOT_FOUND immediately after MRCMC returns Null. This is because MRCMC returns Null for any subsets of $\Sigma_p (\subseteq \Sigma)$ if it returns Null with Σ_p . Specifically, for two subsets of protectable events $\Sigma_p \supset \Sigma'_p$ and two subsets of observable events $\Sigma_o \supset \Sigma'_o$, if $H(\Sigma_o)$ cannot be protected with respect to R'_p and Σ_p , then $H(\Sigma'_o)$ cannot be protected with respect to R'_p and Σ'_p , i.e., with fewer protectable events and fewer observable events available.

Proposition 4.9 *Consider a plant $G = (Q, \Sigma, _, _)$, a subset of critical states $Q_c \subseteq Q$, and a protection requirement R_p . Denote R'_p as a conservative protection requirement derived from R_p by (11). Given two subsets of protectable events $\Sigma_p \subseteq \Sigma_o$ and $\Sigma'_p \subset \Sigma'_o$ where $\Sigma'_p \subset \Sigma_p$ and $\Sigma'_o \subset \Sigma_o$, let $H(\Sigma_o) = (\mathcal{A}, \Sigma_o, \delta_{H(\Sigma_o)}, A_0)$, and $H(\Sigma'_o) = (\mathcal{A}', \Sigma'_o, \delta_{H(\Sigma'_o)}, A'_0)$. If there is no policy $\mathcal{P}_{H(\Sigma_o)} : \mathcal{A} \rightarrow 2^{\Sigma_p \cap \Sigma_o}$ such that $H(\Sigma_p)$ is protected with respect to R_p and $\mathcal{P}_{H(\Sigma_o)}$, then there is no policy $\mathcal{P}_{H(\Sigma'_o)} : \mathcal{A}' \rightarrow 2^{\Sigma'_p \cap \Sigma'_o}$ such that $H(\Sigma'_o)$ is protected with respect to R_p and $\mathcal{P}_{H(\Sigma'_o)}$.*

Proof By assumption, it holds that

$$(\exists q_c \in Q_c)(\exists s \in \Sigma_o^*) q_c \in \delta_{H(\Sigma_o)}(A_0, s) \wedge s \notin \underbrace{\Sigma_o^* \Sigma_p \Sigma_o^* \cdots \Sigma_o^* \Sigma_p \Sigma_o^*}_{\Sigma_p \text{ appears } R_p(q_c) \text{ times}}. \quad (28)$$

Letting $M : \Sigma_o^* \rightarrow \Sigma'_o{}^*$ be a natural projection, we have that

$$(\forall q_c \in Q_c)(\forall s \in \Sigma_o^*) q_c \in \delta_{H(\Sigma_o)}(A_0, s) \implies q_c \in \delta_{H(\Sigma'_o)}(A'_0, M(s)). \quad (29)$$

Also, letting $r \in \mathbb{N}^+$ be an arbitrary positive integer, from $\Sigma'_p \subset \Sigma_p$ and $\Sigma'_o \subset \Sigma_o$, it holds that

$$(\forall s \in \Sigma_o^*) s \notin \underbrace{\Sigma_o^* \Sigma_p \Sigma_o^* \cdots \Sigma_o^* \Sigma_p \Sigma_o^*}_{\Sigma_p \text{ appears } r \text{ times}} \implies M(s) \notin \underbrace{\Sigma'_o{}^* \Sigma'_p \Sigma'_o{}^* \cdots \Sigma'_o{}^* \Sigma'_p \Sigma'_o{}^*}_{\Sigma'_p \text{ appears } r \text{ times}}. \quad (30)$$

In words, if a string $s \in \Sigma_o^*$ does not contain r events in Σ_p , then $M(s) \in \Sigma'_o{}^*$ cannot contain r events in Σ'_p , since the number of events in Σ'_p contained in $M(s)$ is always less than or equal to the number of events in Σ_p contained in s . Therefore, from (29) and (30), the property (28) implies that

$$(\exists q_c \in Q_c)(\exists t \in \Sigma'_o{}^*) q_c \in \delta_{H(\Sigma'_o)}(A'_0, t) \wedge t \notin \underbrace{\Sigma'_o{}^* \Sigma'_p \Sigma'_o{}^* \cdots \Sigma'_o{}^* \Sigma'_p \Sigma'_o{}^*}_{\Sigma'_p \text{ appears } R_p(q_c) \text{ times}}.$$

The proof is **completed**. ■

Here, we can observe that the two similar Algorithm 3 and Algorithm 4 have the following relationship.

Theorem 4.10 *Algorithm 4 returns NOT_FOUND if and only if Algorithm 3 returns NOT_FOUND.*

Proof (If) Assume that Algorithm 3 returns NOT_FOUND at Line 14. This means that MRCMC returns Null with $\Sigma'_{co} \cap \Sigma_p = \emptyset$. Thus, in this case, Algorithm 4 also returns NOT_FOUND at Line 14 because it lets $\Sigma'_{co} = \Sigma_{co} \setminus \Sigma_p$ in Line 2.

Next, assume that Algorithm 3 returns NOT_FOUND in Line 27. Algorithm 3 lets $\Sigma'_o = \Sigma_o \setminus \Sigma'_{co}$ by Line 19, and calls MRCMC whenever it removes one least-costly protectable event from Σ'_{co} in Line 17. Thus, when Algorithm 3 reaches Line 25, the set Σ'_o is the first set such that MRCMC returns a non-empty protection policy \mathcal{P}' . That is, MRCMC returns Null for any $\Sigma''_o \subset \Sigma'_o$. Also, when Algorithm 3 returns NOT_FOUND with Σ'_o at Line 27, from Proposition 4.5, CHECKDOO returns False for any $\Sigma''_o \supset \Sigma'_o$. Therefore, since CHECKDOO in Line 18 of Algorithm 4 returns False for such Σ'_o and one protectable event is removed from Σ'_o , MRCMC in Line 12 returns Null and then Algorithm 4 returns NOT_FOUND.

(Only If) Assume that Algorithm 4 returns NOT_FOUND at Line 14 with $\Sigma'_{co} = \Sigma_{co} \setminus \Sigma_p$. In this case, since MRCMC returns Null for $\Sigma'_p = \Sigma_p$, by Proposition 4.9, MRCMC also returns Null for any $\Sigma'_p \subseteq \Sigma_p$. Thus, in Algorithm 3, since MRCMC on Line 11 returns Null for all $\Sigma'_p \subseteq \Sigma_p$, from Line 17, Σ'_{co} will eventually be equal to $\Sigma_{co} \setminus \Sigma_p$, and then Algorithm 3 returns NOT_FOUND at Line 14.

Next, assume that there exists $\Sigma'_p \subset \Sigma_p$ such that Algorithm 4 returns NOT_FOUND at Line 14. This implies that CHECKDOO returns False with any $\Sigma''_p \supset \Sigma'_p$. Denoting such Σ'_p by $\Sigma_{p,no}$, in Algorithm 3, from Proposition 4.9, MRCMC in Line 11 returns Null for all $\Sigma'_p \subseteq \Sigma_{p,no}$ and CHECKDOO in Line 26 returns False with any $\Sigma'_p \supset \Sigma_{p,no}$. Therefore, in this case, Algorithm 3 returns NOT_FOUND at Line 27. ■

In words, Proposition 4.10 states that if Algorithm 3 can find a solution to Problem 4.3, then Algorithm 4 can also do so, and vice versa.

Although a pair returned by Algorithm 4 solves Problem 4.3, a solution by Algorithm 3 always yields a higher (or equal) minimum degree of opacity compared to that of Algorithm 4, while they result in the same cost level.

Theorem 4.11 *Assume that Algorithm 3 and Algorithm 4 return $(\Sigma_{co,1}, \mathcal{P}_{H(\Sigma'_{o,1})})$ and $(\Sigma_{co,2}, \mathcal{P}_{H(\Sigma'_{o,2})})$, respectively, where $\Sigma'_{o,1} = \Sigma_o \setminus \Sigma_{co,1}$ and $\Sigma'_{o,2} = \Sigma_o \setminus \Sigma_{co,2}$. Both of the following properties hold:*

$$\text{P1. } (\forall q_s \in Q_s) \ \Theta_{H(\Sigma'_{o,1})}(q_s) \geq \Theta_{H(\Sigma'_{o,2})}(q_s);$$

$$\text{P2. } k_{\mathcal{P}_{H(\Sigma'_{o,1})}} = k_{\mathcal{P}_{H(\Sigma'_{o,2})}}.$$

Proof First, consider that there exist two subsets of unobservable events $\Sigma_{uo,1}$ and $\Sigma_{uo,2}$ such that $\Sigma_{co,1}$ is returned when $\Sigma'_{uo} = \Sigma_{uo,1}$ in Algorithm 3 and $\Sigma_{co,2}$ is returned when

$\Sigma'_{uo} = \Sigma_{uo,2}$ in Algorithm 4. From Lines 2 and 17 in Algorithm 3, Lines 3 and 21 in Algorithm 4, and Proposition 4.10, if there exists $\Sigma_{uo,2} \subseteq \Sigma_{uo}$ such that Algorithm 4 returns a pair, then it holds that $\Sigma_{uo,1} \supseteq \Sigma_{uo,2}$. Thus, from Lines 1 in Algorithm 3 and Lines 2 in Algorithm 4, we have that $\Sigma_{co,1} \supseteq \Sigma_{co,2}$. Therefore, since $\Sigma_o \setminus \Sigma_{co,1} \subseteq \Sigma_o \setminus \Sigma_{co,2}$, property P1 holds by Proposition 4.5.

Next, consider that there exists two subsets of protectable events $\Sigma_{p,1}$ and $\Sigma_{p,2}$ such that $\mathcal{P}_{H(\Sigma'_{o,1})}$ is returned when $\Sigma'_p = \Sigma_{p,1}$ in Algorithm 3 and $\mathcal{P}_{H(\Sigma'_{o,2})}$ is returned when $\Sigma'_p = \Sigma_{p,2}$ in Algorithm 4. From Lines 4 and 20 in Algorithm 3 and Lines 5 and 23 in Algorithm 4, by $\Sigma_{uo,1} \supseteq \Sigma_{uo,2}$, we have that $\Sigma_{p,1} \subseteq \Sigma_{p,2}$. Thus, since both $\mathcal{P}_{H,1}$ and $\mathcal{P}_{H,2}$ satisfy property B3 of Problem 4.3, it holds that $\Sigma_{\mathcal{P}_{H(\Sigma'_{o,1})}} = \Sigma_{\mathcal{P}_{H(\Sigma'_{o,2})}}$. Therefore, property P2 holds by (9). ■

According to Definition 3.4, we can conclude that Algorithm 3 always results in a better (or equally good) solution, compared to Algorithm 4.

Example 4.12 Let us revisit the example plant G_2 in Example 4.6. By running Algorithm 4, we obtain a solution $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ where

$$\begin{aligned} \Sigma'_{co} &= \{\sigma_3, \sigma_6, \sigma_7\}, & \Sigma'_o &= \{\sigma_1, \sigma_2, \sigma_4, \sigma_5, \sigma_8\}, & \mathcal{P}_{H(\Sigma'_o)}(\{q_1\}) &= \{\sigma_1, \sigma_2\}, \\ \mathcal{P}_{H(\Sigma'_o)}(\{q_4, q_5, q_6, q_7\}) &= \{\sigma_8\}, & \mathcal{P}_{H(\Sigma'_o)}(\{q_5, q_6, q_7\}) &= \{\sigma_8\}, & \mathcal{P}_{H(\Sigma'_o)}(\{q_6, q_7, q_8\}) &= \{\sigma_8\}. \end{aligned}$$

By mapping back $\mathcal{P}_{H(\Sigma'_o)}$ to the original plant G_2 as in (5), we can confirm that the protection requirement R_p for both critical states q_8 and q_{10} is satisfied.

Figure 6 illustrates the observer $H(\Sigma'_o)$ of G_2 computed in Algorithm 4. The secret state q_6 has degree of opacity of 2 and q_{10} has degree of opacity of 1. The maximum cost level of this example solution is 6 since σ_8 is protected and σ_5 is not. By comparing the solution in Example 4.6, we can conclude that the two properties of Theorem 4.11 hold, that is, Algorithm 3 produced a better solution than Algorithm 4.

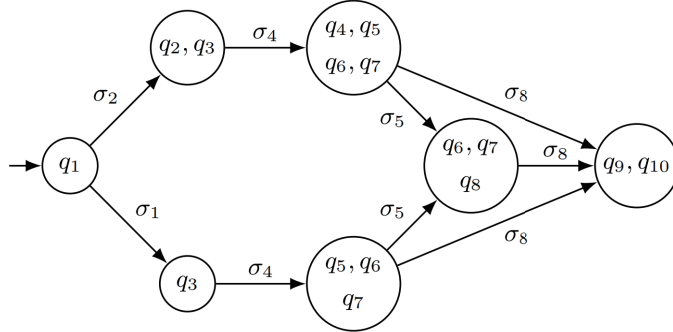


Figure 6 Observer $H(\Sigma'_o)$

Even if there exists a solution to Problem 4.3, the aforementioned three algorithms (Algorithms 1, 3, 4) may not be able to find a solution since Lemma 3.8 is only sufficient. However, from Proposition 4.7 and the following example, it can be seen that Algorithm 1 is more likely to find a solution than Algorithm 3 and Algorithm 4, as Algorithm 1 searches a larger space of concealable-and-protectable events than the other two.

Example 4.13 Consider an example plant G_3 in Figure 7. This plant has one secret state q_6 and two critical states q_6 and q_7 . The event set Σ is partitioned into the subsets of concealable events $\Sigma_{co} = \{\sigma_3, \sigma_4, \sigma_5\}$ and of protectable events $\Sigma_p = \{\sigma_1, \sigma_2, \dots, \sigma_5\}$. The protectable events are grouped into five levels as one event per one level in this example to meet the constraint of Problem 4.3, namely $\Sigma_{cl,1} = \{\sigma_1\}$, $\Sigma_{cl,2} = \{\sigma_2\}$, $\Sigma_{cl,3} = \{\sigma_3\}$, $\Sigma_{cl,4} = \{\sigma_4\}$, and $\Sigma_{cl,5} = \{\sigma_5\}$. Suppose that the obfuscation requirement and the protection requirement in this example are $R_o(q_6) = 2$ and $R_p(q_6) = 1, R_p(q_7) = 2$, respectively.

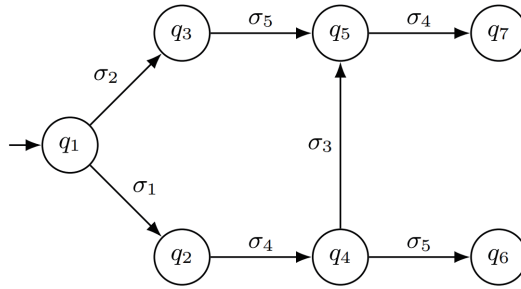


Figure 7 Example plant G_3

Algorithm 1 produces only one pair $(\Sigma'_{co}, \mathcal{P}_{H(\Sigma'_o)})$ where

$$\begin{aligned} \Sigma'_{co} &= \{\sigma_3, \sigma_5\}, \quad \Sigma'_o = \{\sigma_1, \sigma_2, \sigma_4\}, \quad \mathcal{P}_{H(\Sigma'_o)}(\{q_1\}) = \{\sigma_1, \sigma_2\}, \\ \mathcal{P}_{H(\Sigma'_o)}(\{q_3, q_5\}) &= \{\sigma_4\}, \quad \mathcal{P}_{H(\Sigma'_o)}(\{q_4, q_5, q_6\}) = \{\sigma_4\}. \end{aligned}$$

The observer $H(\Sigma'_o)$ of G_3 computed by Algorithm 1 is depicted in Figure 8. We can confirm that both requirements R_o and R_p are satisfied. We see here the trade-off between algorithm capability and computational resources: Whereas Algorithm 1 found a solution, both Algorithm 3 and Algorithm 4 return NOT_FOUND for this example, since neither examines $\Sigma'_{co} = \{\sigma_3, \sigma_5\}$, in the interest of reduced computation time.

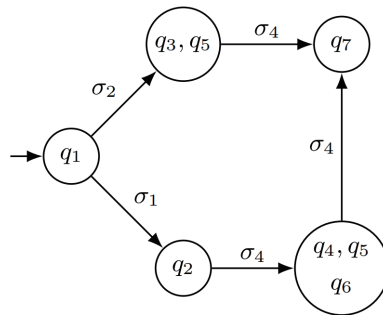


Figure 8 Observer $H(\Sigma'_o)$

5 Conclusion

We have studied a two-pronged approach to produce countermeasures against eavesdropping and infiltration by employing degree of opacity and state protection, respectively. General and special versions of two-pronged security problem are formulated. The general problem requires a solution (i.e., a pair of concealed event subset and protection policy) should not yield smaller minimum degree of opacity and larger maximum cost level than other solutions while accepting incomparable solutions. We have presented an algorithm to compute solutions to the general problem, illustrating it with example plants. The special version has also been introduced so as to reduce the computation time of the general version, by imposing additional constraints on a given plant and changing the way to compare two solutions to determine which one is better.

In the obfuscation part of the two-pronged security problem, we constrained the designer to concealable *events* instead of concealable *transitions*, i.e., every occurrence of a given event is either concealed or not concealed. Under this assumption, we proved that degree of opacity is monotonic. We did not need to make a similar restriction for the protection part of the security problem; rather, we allow specific transitions to be protected and do not require that all transitions associated with a particular event must either be protected or not protected. Our choice constrains us to enforce degree of opacity based on events instead of on transitions. In future work, we aim to extend our methodology by taking a transition-based approach for satisfying the obfuscation requirement instead of the current event-based approach, which would enable us to consider more flexible implementations in practical systems. Other possible directions of extension include the adoption of advanced notions of state protection such as dynamic clearance level^[32] and K -protection^[33].


Conflict of Interest

The authors declare no conflict of interest.

References

- [1] Ramadge P J and Wonham W M, Supervisory control of a class of discrete event processes, *Journal on Control and Optimization*, 1987, **25**(1): 206–230.
- [2] Schonewille B H, Moulton R H, and Rudie K, Enforcing degree of opacity with supervisory control, *Proceedings of the IEEE Conference on Decision and Control*, 2022, 5450–5457.
- [3] Matsui S and Cai K, Secret securing with multiple protections and minimum costs, *Proceedings of the IEEE Conference on Decision and Control*, 2019, 7635–7640.
- [4] Bryans J W, Koutny M, and Ryan P Y A, Modelling opacity using Petri nets, *Electronic Notes in Theoretical Computer Science*, 2005, **121**: 101–115.
- [5] Saboori A and Hadjicostis C N, Notions of security and opacity in discrete event systems, *Proceedings of the IEEE Conference on Decision and Control*, 2007, 5056–5061.

- [6] Saboori A and Hadjicostis C N, Verification of initial-state opacity in security applications of des, *Proceedings of the International Workshop on Discrete Event Systems*, 2008, 328–333.
- [7] Lin F, Opacity of discrete event systems and its applications, *Automatica*, 2011, **47**(3): 496–503.
- [8] Behinaein B, Lin F, and Rudie K, Optimal information release for mixed opacity in discrete-event systems, *IEEE Transactions on Automation Science and Engineering*, 2019, **16**(4): 1960–1970.
- [9] Yang J K, Deng W L, Qiu D W, et al., Opacity of networked discrete event systems, *Information Sciences*, 2021, **543**: 328–344.
- [10] Yang S and Yin X, Secure your intention: On notions of pre-opacity in discrete-event systems, *IEEE Transactions on Automatic Control*, 2023, **68**(8): 4754–4766.
- [11] Hou J Y, Yin X, and Li S Y, A framework for current-state opacity under dynamic information release mechanism, *Automatica*, 2022, **140**: 110238.
- [12] Jacob R, Lesage J J, and Faure J M, Overview of discrete event systems opacity: Models, validation, and quantification, *Annual Reviews in Control*, 2016, **41**: 135–146.
- [13] Guo Y, Jiang X N, Guo C, et al., Overview of opacity in discrete event systems, *IEEE Access*, 2020, **8**: 48731–48741.
- [14] Han X G, Zhang K Z, Zhang J H, et al., Strong current-state and initial-state opacity of discrete-event systems, *Automatica*, 2023, **148**: 110756.
- [15] Wu Y C and Lafortune S, Comparative analysis of related notions of opacity in centralized and coordinated architectures, *Discrete Event Dynamic Systems*, 2013, **23**(3): 307–339.
- [16] Wu Y C and Lafortune S, Synthesis of insertion functions for enforcement of opacity security properties, *Automatica*, 2014, **50**(5): 1336–1348.
- [17] Wu B, Dai J, and Lin H, Synthesis of insertion functions to enforce decentralized and joint opacity properties of discrete-event systems, *Proceedings of the American Control Conference*, 2018, 3026–3031.
- [18] Mohajerani S, Ji Y D, and Lafortune S, Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement, *IEEE Transactions on Automatic Control*, 2020, **65**(8): 3349–3364.
- [19] Tong Y, Li Z W, Seatzu C, et al., Current-state opacity enforcement in discrete event systems under incomparable observations, *Discrete Event Dynamic Systems*, 2018, **28**: 161–182.
- [20] Wonham W M and Cai K, *Supervisory Control of Discrete-Event Systems*, Springer, New York, 2019.
- [21] Barcelos R J and Basilio J C, Enforcing current-state opacity through shuffle and deletions of event observations, *Automatica*, 2021, **133**: 109836.
- [22] Moulton R H, Hamgini B B, Khouzani Z A, et al., Using subobservers to synthesize opacity-enforcing supervisors, *Discrete Event Dynamic Systems*, 2022, **32**(4): 611–640.
- [23] Barcelos R J and Basilio J C, Ensuring utility while enforcing current-state opacity, *IFAC-PapersOnLine*, 2023, **56**(2): 4595–4600.
- [24] Meira-Góes R, Kang E, Kwong R, et al., Stealthy deception attacks for cyber-physical systems, *Proceedings of the IEEE Conference on Decision and Control*, 2017, 4224–4230.
- [25] Carvalho L K, Wu Y C, Kwong R, et al., Detection and mitigation of classes of attacks in supervisory control systems, *Automatica*, 2018, **97**: 121–133.
- [26] Meira-Góes R, Kang E, Kwong R H, et al., Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems, *Automatica*, 2020, **121**: 109172.
- [27] Meira-Góes R, Lafortune S, and Marchand H, Synthesis of supervisors robust against sensor

- deception attacks, *IEEE Transactions on Automatic Control*, 2021, **66**(10): 4990–4997.
- [28] Lin L Y, Thuijsman S, Zhu Y T, et al., Synthesis of supremal successful normal actuator attackers on normal supervisors, *Proceedings of the American Control Conference*, 2019, 5614–5619.
 - [29] Lin L Y, Zhu Y T, and Su R, Synthesis of covert actuator attackers for free, *Discrete Event Dynamic Systems*, 2020, **30**: 561–577.
 - [30] Ma Z Y and Cai K, On resilient supervisory control against indefinite actuator attacks in discrete-event systems, *IEEE Control Systems Letters*, 2022, **6**: 2942–2947.
 - [31] Meira-Góes R, Marchand H, and Lafortune S, Dealing with sensor and actuator deception attacks in supervisory control, *Automatica*, 2023, **147**: 110736.
 - [32] Ma Z Y and Cai K, Optimal secret protection in discrete event systems with dynamic clearance levels, *IFAC-PapersOnLine*, 2023, **56**(2): 3579–3584.
 - [33] Liu R T, Duan W, Mangini A M, et al., K -protection of global secret in discrete event systems using supervisor control, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 2023, 2832–2837.
 - [34] Ma Z Y, Jiang J G, and Cai K, Secret protections with costs and disruptiveness in discrete-event systems using centralities, *IEEE Transactions on Automatic Control*, 2024, **69**(7): 4380–4395.
 - [35] Pascoe C E, The NIST Cybersecurity Framework (CSF) 2.0, 2024,  <https://doi.org/10.6028/nist.cswp.29>.
 - [36] Hopcroft J E, Motwani R, and Ullman J D, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachusetts, 2001.
 - [37] Matsui S, Cai K, and Rudie K, Distributed secret securing in discrete-event systems, *Proceedings of the American Control Conference*, 2024, 3202–3207.