

Tamper-Tolerant Current-State Opacity in Cyber-Physical Systems Under Sensor Attacks

Abstract—This paper investigates the security of cyber-physical systems (CPSs) by focusing on the enforcement of opacity under sensor attacks. We consider a closed-loop system in which an attacker can manipulate sensor readings to mislead the supervisor, potentially causing erroneous control actions and compromising the system’s current-state opacity. To formally capture the system’s security under such conditions, we introduce the notion of tamper-tolerant current-state opacity (TCSO), and provide a verification framework. Based on this, we formulate the tamper-tolerant current-state opacity enforcement problem (TCSOEP) from a defensive perspective. To address this problem, we propose synthesis methods to compute supervisor S_r^k and supervisor S_r^b , which enforce TCSO under unbounded and bounded attack scenarios, respectively.

Index Terms—Discrete-event systems, Sensor attacks, Opacity, Supervisory control

I. INTRODUCTION

Cyber-physical systems (CPSs) integrate computational components with physical processes in a tightly coupled manner and are widely adopted in safety-critical domains such as smart grids, autonomous vehicles, industrial automation, and medical devices. The interaction between cyber and physical layers enables real-time monitoring, control, and decision making. However, the growing reliance on communication networks and software functionalities exposes CPSs to cyber attacks that can compromise the confidentiality, integrity, and availability of the system. In particular, attacks targeting the disclosure of sensitive internal behaviors have attracted increasing attention in recent years [1].

In this paper, we model CPSs using the discrete-event systems (DESs) framework, which provides a rigorous formalism for analyzing information-flow security properties. Within this framework, opacity has emerged as a fundamental property for preventing secret information from being inferred by an external observer through system outputs. A system is considered opaque if an attacker cannot conclusively deduce its secret from the available observations. As security requirements continue to increase in safety-critical applications, opacity has become an active area of research, providing a basis for developing supervisory control strategies that enhance the resilience of CPSs against malicious intrusions. Several notions of opacity have been proposed, including current-state opacity [2]–[4], initial-state opacity [5], k -step and infinite-step opacity [6], [7]. A variety of verification techniques have been developed for the formal verification of opacity properties [8]–[10]. When a system is verified to be non-opaque, a main challenge is how to enforce it through appropriate. Existing approaches can be broadly classified into two categories. The first, commonly referred to as supervisory control of opacity, restricts system behavior to eliminate secret-revealing

executions [11]–[14]. The second alters the information flow of the system to mislead the intruder [15], [16].

While significant progress has been made in enforcing opacity in DESs, much of the earlier research has focused on scenarios where intruders act as observers without the ability to interfere with the system’s behavior. In contrast, recent studies have shown that supervisors can be vulnerable to active attacks, particularly through compromised communication channels between the plant and the supervisor, such as sensor deception attacks. In these attacks, intruders can insert, delete, or replace critical events in the communication channel, misleading the supervisor into making incorrect control decisions and potentially driving the plant into unsafe states [17]–[23].

A recently studied class of sensor attacks has been investigated, in which the attacker’s objective is redefined: instead of directly forcing the plant into unsafe physical states, the attacker aims to compromise system properties such as opacity, detectability, and diagnosability. For example, the work in [24] investigates attacks on initial-state opacity by synthesizing both strong and weak attack strategies designed to deceive the supervisor into revealing the system’s initial secret. Similarly, the authors of [25] adopt an adversarial perspective to synthesize attack strategies that can compromise the system’s strong and weak detectability. Additionally, the work in [26] investigates the least-cost state estimation under sensor attacks by assigning a positive integer cost to each type of attack. The authors analyze the effect of total cost constraints on state estimation and propose a least-cost tamper-tolerant state estimation method of polynomial complexity. Furthermore, the work in [27] leverages this cost-based framework to provide verification methods for fault diagnosis and tampering detection under unconstrained and cost-constrained sensor attacks.

This work studies the enforcement of opacity in DESs modeled as deterministic finite automata, considering both bounded and unbounded active sensor attacks. The attacker considered here can actively manipulate sensor observations by inserting, deleting, or replacing events to infer whether the plant is currently in a secret state. For bounded attacks, the total number of active manipulations is limited to k , reflecting practical resource constraints such as time or energy. In both bounded and unbounded scenarios, the attacker is assumed to have full knowledge of the plant and supervisor models and observes events through the same communication channel as the supervisor. To systematically enforce opacity in DESs under sensor deception attacks, we introduce the notion of tamper-tolerant current-state opacity (TCSO) and develop verification methods for determining whether a system satisfies TCSO. The primary objective of this work is to synthesize a supervisor that guarantees TCSO under both bounded and unbounded attack scenarios. The main contributions are sum-

marized as follows:

- 1) We propose the notion of tamper-tolerant current-state opacity and provide a method to verify whether a system satisfies TCSO, enabling systematic security analysis under sensor deception attacks.
- 2) To accurately capture the interaction between the plant, the supervisor, and an attacker, we construct an extended system model that integrates plant dynamics, supervisory control and sensor attacks.
- 3) Based on this model, we propose synthesis methods to obtain both the supervisor S_r and the supervisor S_r^k , ensuring that the system satisfies TCSO under unbounded and bounded attacks, respectively, while preserving as much permissible behavior as possible.

This paper is structured as follows. Section II introduces the basic concepts and notation. Section III formalizes the system setting with an attack model and defines the TCSO along with the tamper-tolerant current-state opacity enforcement problem (TCSOEP). Section IV develops a method to verify whether a system satisfies TCSO. Section V proposes synthesis procedures for constructing both supervisor S_r and supervisor S_r^k , ensuring TCSO under unbounded and bounded attacks. Finally, Section VI concludes the paper and outlines future research directions.

II. PRELIMINARIES

A deterministic finite-state automaton is defined as $G = (X, \Sigma, \delta, x_0)$, where X is a finite set of states, Σ is a finite set of events, $\delta : X \times \Sigma \rightarrow X$ is a (partial) transition function, and $x_0 \in X$ is the initial state. The transition function δ can be extended to $\delta : X \times \Sigma^* \rightarrow X$, where Σ^* denotes the Kleene closure of Σ , i.e., the set of all finite sequences of events over Σ , including the empty sequence ε . The language generated by G is defined as $L(G) = \{s \in \Sigma^* \mid \delta(x_0, s) \in X\}$. For strings $u, v \in \Sigma^*$, the concatenation of u and v is defined as a string uv . The concatenation of two strings $u, v \in \Sigma^*$ is a new string $w = uv \in \Sigma^*$ composed of the sequence of events in u followed by the sequence of events in v . A string $u \in \Sigma^*$ is called a prefix of a string $s \in \Sigma^*$ if there exists another string $v \in \Sigma^*$ such that $s = uv$. The prefix closure of a language $L \subseteq \Sigma^*$ is the set of all prefixes of strings in L , formally given by $\bar{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^* : st \in L\}$. The concatenation of two languages $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Sigma^*$ is a new language $L_1 L_2 = \{ss' \mid s \in L_1 \wedge s' \in L_2\}$, which consists of all strings obtained by concatenating a string from L_1 with a string from L_2 .

The event set Σ is partitioned into two disjoint subsets: the observable event set Σ_o and the unobservable event set Σ_{uo} . A natural projection function $P : \Sigma^* \rightarrow \Sigma_o^*$ maps each sequence of events the corresponding sequence of observable events by removing all unobservable events. Formally, natural projection function $P : \Sigma^* \rightarrow \Sigma_o^*$ is recursively defined for $s \in \Sigma^*$ and $e \in \Sigma$ as:

$$P(\varepsilon) = \varepsilon, P(se) = \begin{cases} P(s)e, & \text{if } e \in \Sigma_o \\ P(s), & \text{if } e \in \Sigma_{uo}. \end{cases}$$

The inverse projection $P^{-1} : \Sigma_o^* \rightarrow 2^{\Sigma^*}$ is defined by $P^{-1}(t) = \{s \in \Sigma^* \mid P(s) = t\}$. The projection P and inverse projection P^{-1} can be extended to languages by applying them to all strings in the language. Formally, for $L \subseteq \Sigma^*$, $P(L) = \{t \in \Sigma_o^* \mid \exists s \in L : P(s) = t\}$, and for $L_o \subseteq \Sigma_o^*$, $P^{-1}(L_o) = \{s \in \Sigma^* \mid \exists t \in L_o : P(s) = t\}$.

For any state $x \in X$ in G , the set of events enabled at x is defined as $En_G(x) := \{e \in \Sigma \mid \delta(x, e)!\}$, where “!” indicates that the transition $\delta(x, e)$ is defined. For any automaton G , its accessible part, denoted by $Ac(G)$ is obtained by removing all states that are not reachable from the initial state x_0 along with their corresponding transitions.

In supervisory control theory, the set of events Σ is partitioned into controllable events Σ_c and uncontrollable events Σ_{uc} . A supervisor is defined as a function $S : P(L(G)) \rightarrow \Gamma$, where $\Gamma = \{\gamma \subseteq \Sigma : \Sigma_{uc} \subseteq \gamma\}$ is the set of control decisions. The closed-loop system of G under the supervision of S , denoted by S/G and hereafter referred to simply as the closed-loop system, generates the language $L(S/G)$ defined recursively as follows: (i) $\varepsilon \in L(S/G)$; (ii) for all $s \in L(S/G)$, if $s\sigma \in L(G)$ and $\sigma \in S(P(s))$, then $s\sigma \in L(S/G)$. Given a system G and a language $K \subseteq L(G)$, K is said to be controllable with respect to Σ_c and G if $(\forall s \in K)(\forall \sigma \in \Sigma_{uc})[s\sigma \in L(G) \Rightarrow s\sigma \in K]$.

Given a set of states $B \subseteq X$ in automaton $G = (X, \Sigma, \delta, x_0)$, the unobservable reach of B is defined as $UR(B) = \{x' \in X \mid \exists x \in B, \exists t \in \Sigma_{uo}^* : \delta(x, t) = x'\}$. The set of states that can be reached immediately from the state set $B \subseteq X$ upon the occurrence of an observable event $e \in \Sigma_o$ is defined as $Next(B, e) = \{x \in X \mid \exists x' \in B : \delta(x', e) = x\}$. The observer of automaton G is defined as $Obs(G) = (X_{obs}, \Sigma_o, \delta_{obs}, x_{0,obs}) = Ac(2^X, \Sigma_o, \delta_{obs}, UR(\{x_0\}))$, where $X_{obs} \subseteq 2^X$ is the set of observer states, $x_{0,obs} = UR(\{x_0\})$ is the initial observer state, and for any $x_{obs} \in 2^X$ and $\sigma \in \Sigma_o$, the transition function is given by $\delta_{obs}(x_{obs}, \sigma) = UR(Next(x_{obs}, \sigma))$.

The system G is associated with a set of secret states, denoted by $X_S \subseteq X$, and the set of non-secret states is $X_{NS} = X \setminus X_S$. A system is said to be opaque if, based on the observable sequences, an attacker cannot determine with certainty that system G is in a secret state. We now recall the definition of current-state opacity.

Definition 1. (Current-State Opacity) Let $G = (X, \Sigma, \delta, x_0)$ be a deterministic finite-state automaton with a set of observable events Σ_o , a set of secret states $X_S \subseteq X$, and a set of non-secret states $X_{NS} = X \setminus X_S$. Then, G is said to be current-state opaque if

$$(\forall s \in L(G) : \delta(x_0, s) \in X_S) \Rightarrow (\exists s' \in L(G)) \\ [\delta(x_0, s') \in X_{NS} \wedge P(s') = P(s)].$$

Given two deterministic finite-state automata $G_1 = (X_1, \Sigma_1, \delta_1, x_{1,0})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{2,0})$, the parallel composition of G_1 and G_2 is a deterministic finite-state automaton $G_1 \parallel G_2 = Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta_{1 \parallel 2}, (x_{1,0}, x_{2,0}))$, where the transition function $\delta_{1 \parallel 2}$ is defined for $(x_1, x_2) \in X_1 \times X_2$, and $\sigma \in \Sigma_1 \cup \Sigma_2$, as

$$\delta_{1\|2}((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)), & \text{if } \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)! \\ (\delta_1(x_1, \sigma), x_2), & \text{if } \delta_1(x_1, \sigma)! \wedge \sigma \notin \Sigma_2 \\ (x_1, \delta_2(x_2, \sigma)), & \text{if } \delta_2(x_2, \sigma)! \wedge \sigma \notin \Sigma_1 \\ \text{undefined}, & \text{otherwise} \end{cases}$$

III. PROBLEM STATEMENT

In this section, we introduce the closed-loop system under attacks and formalize the notion of tamper-tolerant current-state opacity under sensor attacks.

A. System Model Under Sensor Attacks

Fig. 1 illustrates the concept of sensor deception attacks within a supervisory control framework. In the absence of interference, the plant G generates events, and the corresponding event information is transmitted to the supervisor via sensors. Upon receiving this information, the supervisor updates its control decisions, which are subsequently applied to the plant G through actuators. However, due to the reliance on network communication, this architecture is vulnerable to attacked by attacker. An attacker may eavesdrop on sensor data and actively compromise sensors to modify event information received by the supervisor. These deceptive manipulations can mislead the supervisor's control decisions, potentially causing the closed-loop system S/G to behave in an undesired manner.

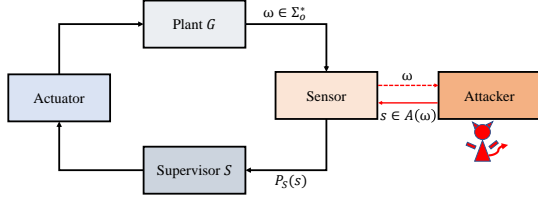


Fig. 1. The architecture of the closed-loop system under sensor attacks.

We assume that the attacker has full knowledge of the structures of both the plant G and the supervisor S , and is capable not only of eavesdropping but also of tampering with sensor readings, including insertion, deletion, and replacement of events within the communication channel. Such tampering is restricted to a subset of observable events, referred to as the vulnerable events, denoted by $\Sigma_a \subseteq \Sigma_o$. The set of possible tampering operations is defined as $\Sigma_T = \Sigma_I \cup \Sigma_D \cup \Sigma_R$, where $\Sigma_I = \{\alpha_\varepsilon \mid \alpha \in \Sigma_a\}$ denotes the insertion of event α into the channel by the attacker, $\Sigma_D = \{\varepsilon_\alpha \mid \alpha \in \Sigma_a\}$ denotes the deletion of event α from the communication channel, and $\Sigma_R = \{\beta_\alpha \mid \alpha, \beta \in \Sigma_a, \alpha \neq \beta\}$ denotes the replacement of event α by event β when $|\Sigma_a| \geq 2$, with $\Sigma_R = \emptyset$ if $|\Sigma_a| = 1$.

The plant G generates strings consisting of both observable and unobservable events. Each generated string is projected onto its corresponding observable sequence, which constitutes the attacker's observation. Based on this observation, the attacker may perform tampering operations to produce a set of all possible tampered sequences. For each observable event α generated by the plant, the attacker may perform at most

one tampering operation on α , that is, either a deletion or a replacement, but not both. In addition, between any two consecutive observable events received by the attacker, an arbitrary number of insertion operations may be performed. To capture all such possibilities, we define the attack map $A : \Sigma_o^* \rightarrow 2^{(\Sigma_T \cup \Sigma_o)^*}$ for $\omega \in \Sigma_o^*$ and $\alpha \in \Sigma_o$ as follows:

$$A(\varepsilon) = \{\varepsilon\},$$

$$A(\alpha) = \begin{cases} \{\alpha\}\Sigma_I^*, & \text{if } \alpha \in \Sigma_o \setminus \Sigma_a, \\ \{\alpha, \varepsilon_\alpha\}\Sigma_I^*, & \text{if } |\Sigma_a| = 1 \wedge \alpha \in \Sigma_a, \\ (\{\beta_\alpha \mid \beta \in \Sigma_a \setminus \{\alpha\}\} \cup \{\alpha, \varepsilon_\alpha\})\Sigma_I^*, & \text{if } |\Sigma_a| \geq 2 \wedge \alpha \in \Sigma_a, \end{cases}$$

$$A(\omega\alpha) = A(\omega)A(\alpha).$$

For each tampered sequence $s \in A(\omega)$ generated by the attacker, we consider the sequence actually observed by the supervisor under attacks. To formalize this, we define the projection function $P_S : (\Sigma_T \cup \Sigma_o)^* \rightarrow \Sigma_o^*$ for $s \in (\Sigma_T \cup \Sigma_o)^*$ and $e \in \Sigma_T \cup \Sigma_o$ which specifies the supervisor's observation under attacks:

$$P_S(\varepsilon) = \varepsilon,$$

$$P_S(e) = \begin{cases} e, & \text{if } e \in \Sigma_o, \\ \beta, & \text{if } e = \beta_\varepsilon \in \Sigma_I, \\ \varepsilon, & \text{if } e = \varepsilon_\alpha \in \Sigma_D, \\ \beta, & \text{if } e = \beta_\alpha \in \Sigma_R, \end{cases}$$

$$P_S(se) = P_S(s)P_S(e).$$

To illustrate the possible attack scenarios and their impact on the supervisor's observation, we consider the following example. Suppose $\Sigma = \{a, b, c, d\}$, $\Sigma_o = \{a, c, d\}$ and $\Sigma_a = \{a, c\}$. For a sequence $\omega = abdc \in L(G)$ generated by plant G , the attacker observes its projection onto the observable events through the communication channel, i.e., $P(\omega) = adc$. The corresponding set of all possible tampered sequences generated by the attacker is $A(P(\omega)) = A(adc) = \{a, \varepsilon_a, c_a\}\{\varepsilon, a_\varepsilon, c_\varepsilon\}\{d\}\{\varepsilon, a_\varepsilon, c_\varepsilon\}\{c, \varepsilon_c, a_c\}\{\varepsilon, a_\varepsilon, c_\varepsilon\}$. For a tampered sequence $s = c_a c_\varepsilon d \varepsilon c a_\varepsilon \in A(P(\omega))$, the corresponding observation of the supervisor is $P_S(s) = ccda$.

The supervisor S applies control by issuing a set of enabled events after each observation. If the subsequent observed event received by the supervisor—possibly after tampering by the attacker—does not belong to this enabled set, the supervisor immediately detects the presence of an attack. Equivalently, supervisor S only accepts observation sequences that are contained in its language $L(S)$. A strategic attacker will therefore avoid tampering operations that generate observation sequences the supervisor can instantly recognize as invalid. In supervisory control and information security, only those attackers that remain undetected by the supervisor pose a threat to the system's privacy.

Definition 2 (Stealthy Attacker). Given a plant G with a set of observable events Σ_o , a set of controllable events Σ_c and a set of vulnerable events Σ_a , a supervisor S , and an attacker, the attacker is said to be stealthy if, for every observation $\omega \in P(L(G))$, any tampered sequence $s \in A(\omega)$ generated by the attacker satisfies $P_S(s) \in L(S)$.

Within the aforementioned framework, we consider the closed-loop system under attacks, which integrates the dynamics of the plant, the supervisor, and the stealthy attacker.

Definition 3. Given a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$ with a set of observable events Σ_o , a set of controllable events Σ_c and a set of vulnerable events Σ_a , a supervisor $S = (Y, \Sigma, \delta_s, y_0)$, and a stealthy attacker, the closed-loop system under attacks is defined as an automaton $(S/G)_A = Ac(X \times Y, \Sigma \cup \Sigma_T, \delta_A, (x_0, y_0)) = (X_A, \Sigma_A, \delta_A, x_{0,A})$, where

- $X_A \subseteq X \times Y$ is the state set;
- $\Sigma_A = \Sigma \cup \Sigma_T$ is the event set;
- $x_{0,A} = (x_0, y_0)$ is the initial state;
- $\delta_A : (X \times Y) \times \Sigma_A \rightarrow (X \times Y)$ is the transition function, defined for $(x, y) \in X \times Y$ and $e \in \Sigma_A$ as follows:

$$\delta_A((x, y), e) = \begin{cases} (\delta(x, e), \delta_s(y, e)), & \text{if } e \in \Sigma_o \cap \text{En}_G(x) \cap \text{En}_S(y), \\ (\delta(x, e), y), & \text{if } e \in \Sigma_{uo} \cap \text{En}_G(x), \\ (\delta(x, \alpha), y), & \text{if } [e = \varepsilon_\alpha \in \Sigma_D] \wedge [\alpha \in \text{En}_G(x)], \\ (\delta(x, \alpha), \delta_s(y, \beta)), & \text{if } [e = \beta_\alpha \in \Sigma_R] \wedge [\alpha \in \text{En}_G(x)] \wedge [\beta \in \text{En}_S(y)], \\ (x, \delta_s(y, \beta)), & \text{if } [e = \beta_\varepsilon \in \Sigma_I] \wedge [\beta \in \text{En}_S(y)], \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

The system models the complete dynamics of the attacked system. Each path in $(S/G)_A$ corresponds to a sequence of events, consisting of the original events generated by the plant and possibly some events that have been tampered with by the attacker. The language generated by $(S/G)_A$, denoted by $L((S/G)_A)$, is the set of all sequences corresponding to its paths. For analysis, it is essential to recover the sequence of events originally produced by the plant from any path in $(S/G)_A$. In addition, between any two consecutive observable events received by the attacker, an arbitrary number of insertion operations may be performed. To this end, we define the projection function $P_G : (\Sigma_T \cup \Sigma)^* \rightarrow \Sigma^*$ for $s \in (\Sigma_T \cup \Sigma)^*$ and $e \in \Sigma_T \cup \Sigma_o$, which recovers the sequence of events generated by the plant G under attacks:

$$P_G(\varepsilon) = \varepsilon, \\ P_G(e) = \begin{cases} e, & \text{if } e \in \Sigma, \\ \varepsilon, & \text{if } e = \beta_\varepsilon \in \Sigma_I, \\ \alpha, & \text{if } e = \varepsilon_\alpha \in \Sigma_D, \\ \alpha, & \text{if } e = \beta_\alpha \in \Sigma_R, \end{cases} \\ P_G(se) = P_G(s)P_G(e).$$

Example 1. Consider the automaton $G = (X, \Sigma, \delta, x_0)$ and the supervisor $S = (Y, \Sigma, \delta_s, y_0)$ shown in Fig. 2,

with $\Sigma_o = \{a, c, d, e, f\}$, $\Sigma_c = \{a, c, d, e\}$, $\Sigma_{uo} = \{b\}$, $\Sigma_{uc} = \{b, f\}$, $\Sigma_a = \{d, e\}$, and $X_S = \{2\}$. The closed-loop system S/G and the closed-loop system under attacks $(S/G)_A$ can be constructed as shown in Fig. 3 and Fig. 4, respectively. For a sequence $s = bafce\varepsilon_d\varepsilon_e \in L((S/G)_A)$, the corresponding sequence generated by plant G under attacks is $P_G(s) = bafcde$.

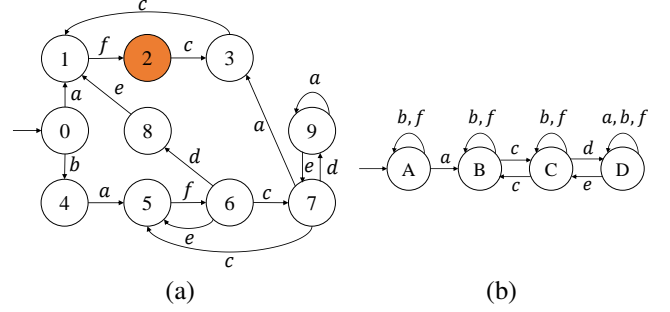


Fig. 2. (a) Plant G (b) Supervisor S .

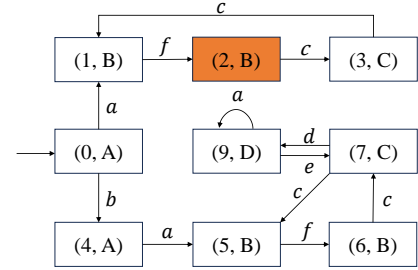


Fig. 3. The closed-loop system S/G .

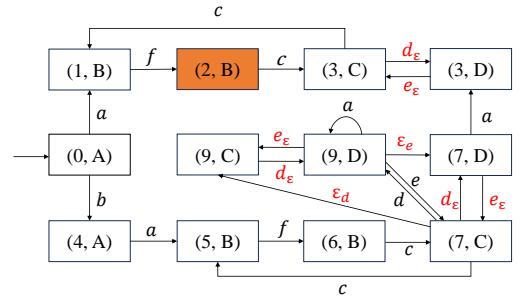


Fig. 4. The closed-loop system under attacks $(S/G)_A$.

B. Tamper-Tolerant Current-State Opacity

We consider a powerful attacker capable of tampering with sensor readings whose objective is to infer whether the plant G is currently in a secret state. If for any attack strategy the attacker might employ, the sequence observed by the attacker never allows the attacker to unambiguously infer that plant G is in a secret state, we say that the closed-loop system S/G satisfies TCSO.

Definition 4. (Tamper-Tolerant Current-State Opacity) Given a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$ with a set of observable events Σ_o , a set of controllable

events Σ_c , a set of vulnerable events Σ_a , a set of secret states $X_S \subseteq X$, and a set of non-secret states $X_{NS} = X \setminus X_S$, a supervisor $S = (Y, \Sigma, \delta_s, y_0)$, and a stealthy attacker, the closed-loop system S/G is said to be tamper-tolerant current-state opaque (TCSO) if

$$(\forall s \in L((S/G)_A) : \delta(x_0, P_G(s)) \in X_S) \Rightarrow (\exists s' \in L((S/G)_A) [\delta(x_0, P_G(s')) \in X_{NS} \wedge P(P_G(s')) = P(P_G(s))]).$$

Hereafter, we use ‘‘TCSO’’ to denote both ‘‘tamper-tolerant current-state opacity’’ and ‘‘tamper-tolerant current-state opaque’’, with the exact meaning clarified by the context. The following example illustrates that, under sensor attacks, the attacker can mislead the supervisor into issuing incorrect control commands, thereby causing the closed-loop system S/G to violate TCSO.

Example 2. Consider the closed-loop system under attacks $(S/G)_A$ in Example 1. For a string $s = bafcd_\epsilon ae_\epsilon cf \in L((S/G)_A)$, the corresponding sequence generated by plant G is $P_G(s) = bafcacf$, which leads the system G to the secret state 2. There is no other string in $L((S/G)_A)$ whose corresponding sequence generated by plant G reaches a non-secret state and has the same natural projection as $P_G(s)$. Therefore, the closed-loop system S/G is not TCSO.

Definition 5. (Tamper-Tolerant Current-State Opacity Enforcement Problem) Given a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$ with a set of observable events Σ_o , a set of controllable events Σ_c , a set of vulnerable events Σ_a , a set of secret states $X_S \subseteq X$, and a set of non-secret states $X_{NS} = X \setminus X_S$, together with a supervisor S and a stealthy attacker, the objective is to synthesize a maximally permissive supervisor S_r such that:

- 1) The closed-loop system S_r/G satisfies TCSO under sensor attacks;
- 2) For any other supervisor S'_r such that S'_r/G also satisfies TCSO under sensor attacks, it holds that $L(S_r/G) \not\subseteq L(S'_r/G)$.

IV. VERIFICATION OF TAMPER-TOLERANT CURRENT-STATE OPACITY

In this section, we present a verification method to determine whether the closed-loop system S/G satisfies TCSO under sensor attacks. The verification method is based on the construction of the observer automaton $Obs((S/G)_A)$ of the closed-loop system under attacks $(S/G)_A$. Given a set of states $B \subseteq X_A$ in $(S/G)_A = (X_A, \Sigma_A, \delta_A, x_{0,A})$, the unobservable reach of B is defined as $UR_A(B) = \{x'_A \in X_A \mid \exists x_A \in B, \exists t \in \Sigma_{uo}^* : \delta_A(x_A, t) = x'_A\}$. The set of states that can be reached immediately from the state set $B \subseteq X_A$ upon the occurrence of an event $e \in \Sigma_o \cup \Sigma_T$ is defined as $Next_A(B, e) = \{x_A \in X_A \mid \exists x'_A \in B : \delta_A(x'_A, e) = x_A\}$. The observer of $(S/G)_A$ is defined as $Obs((S/G)_A) = (X_A^{obs}, \Sigma_o \cup \Sigma_T, \delta_A^{obs}, x_{0,A}^{obs}) = Ac(2^{X_A}, \Sigma_o \cup \Sigma_T, \delta_A^{obs}, UR_A(\{x_{0,A}\}))$,

where $X_A^{obs} \subseteq 2^{X_A}$ is the set of observer states, $x_{0,A}^{obs}$ is the initial state, and for any $x_A^{obs} \in 2^{X_A}$ and $\sigma \in \Sigma_o \cup \Sigma_T$, the transition function is given by $\delta_A^{obs}(x_A^{obs}, \sigma) = UR_A(Next_A(x_A^{obs}, \sigma))$. We then identify a set of opacity-violating states X_{OV} in $Obs((S/G)_A)$. Formally, $X_{OV} = \{x_A^{obs} \in X_A^{obs} \mid \forall x_A \in x_A^{obs} : \psi_1(x_A) \in X_S\}$, where $\psi_1(x_A)$ denotes the first element of x_A . For convenience, the natural projection $P : \Sigma^* \rightarrow \Sigma_o^*$ is extended to $P_A : (\Sigma \cup \Sigma_T)^* \rightarrow (\Sigma_o \cup \Sigma_T)^*$ by treating all events in Σ_T as observable.

Theorem 1. Given a deterministic finite-state automaton G with a set of observable events Σ_o , a set of controllable events Σ_c , a set of vulnerable events Σ_a , a set of secret states $X_S \subseteq X$, and a set of secret states $X_{NS} = X \setminus X_S$, together with a supervisor S and a stealthy attacker, the system S/G satisfies TCSO if and only if $X_{OV} = \emptyset$.

Proof. According to the construction of $(S/G)_A$, if there exists a string $s \in L((S/G)_A)$ we have $\psi_1(\delta_A(x_{0,A}, s)) = \delta(x_0, P_G(s))$. Furthermore, for any string $s_o \in L(Obs((S/G)_A))$, let $F = \{s \in L((S/G)_A) \mid P_A(s) = s_o\}$. Then, by the construction of $Obs((S/G)_A)$, the state reached in $Obs((S/G)_A)$ after observing s_o satisfies $\delta_A^{obs}(x_{0,A}^{obs}, s_o) = \bigcup_{s \in F} \{\delta_A(x_{0,A}, s)\}$.

(\Leftarrow) Assume that $X_{OV} = \emptyset$. Then, by the definition of X_{OV} , for any string $s_o \in L(Obs((S/G)_A))$, we have $\bigcup_{s \in F} \{\psi_1(\delta_A(x_{0,A}, s))\} \not\subseteq X_S$, where $F = \{s \in L((S/G)_A) \mid P_A(s) = s_o\}$. This means that for any string $s \in L((S/G)_A)$ satisfying $\psi_1(\delta_A(x_{0,A}, s)) = \delta(x_0, P_G(s)) \in X_S$, there must exist another string $s' \in L((S/G)_A)$ such that $\psi_1(\delta_A(x_{0,A}, s')) = \delta(x_0, P_G(s')) \in X_{NS}$ and $P_A(s) = P_A(s')$. By the definitions of the projections P and P_G , it follows that $P_A(s) = P_A(s')$ implies $P(P_G(s)) = P(P_G(s'))$. Therefore, according to Definition 4, system S/G satisfies TCSO.

(\Rightarrow) Assume that S/G satisfies TCSO. According to Definition 4, for any string $s \in L((S/G)_A)$ satisfying $\delta(x_0, P_G(s)) = \psi_1(\delta_A(x_{0,A}, s)) \in X_S$, there exists another string $s' \in L((S/G)_A)$ such that $\delta(x_0, P_G(s')) = \psi_1(\delta_A(x_{0,A}, s')) \in X_{NS}$ and $P(P_G(s)) = P(P_G(s'))$. Therefore, for any string $s_o \in L(Obs((S/G)_A))$, let $F = \{s \in L((S/G)_A) \mid P_A(s) = s_o\}$. Then, we have $\bigcup_{s \in F} \{\psi_1(\delta_A(x_{0,A}, s))\} \not\subseteq X_S$. Since $X_{OV} = \{x_A^{obs} \in X_A^{obs} \mid \forall x_A \in x_A^{obs} : \psi_1(x_A) \in X_S\}$, we obtain $X_{OV} = \emptyset$. \square

Example 3. Consider the closed-loop system under attacks $(S/G)_A$ in Fig. 4. The corresponding observer $Obs((S/G)_A)$ can be constructed as depicted in Fig. 5. It is readily verified that $X_{OV} = \{(2, B)\} \neq \emptyset$. According to Theorem 1, we conclude that S/G is not TCSO.

V. ENFORCEMENT OF TAMPER-TOLERANT CURRENT-STATE OPACITY

In this section we construct the augmented attacked system from the closed-loop system under attacks $(S/G)_A$ to capture both the system dynamics and the attacker’s estimation. Based on this structure, we develop an algorithm to synthesize the

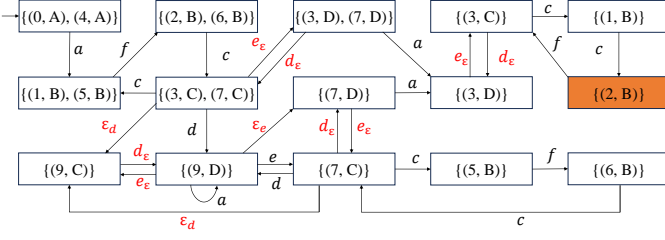


Fig. 5. The observer $Obs((S/G)_A)$ of the closed-loop system under attacks.

supervisor S_r that allows an unlimited number of attacks, and the supervisor S_r^k that limits the number of attacks to k .

A. Construction of Augmented Attacked System

We construct the augmented attacked system $(S/G)_A^{aug}$ by extending the closed-loop system under attacks $(S/G)_A$ to include the attacker's estimated state information, as follows.

Definition 6. Consider a closed-loop system under attacks $(S/G)_A = (X_A, \Sigma_A, \delta_A, x_{0,A})$, where $G = (X, \Sigma, \delta, x_0)$ is the plant and $S = (Y, \Sigma, \delta_s, y_0)$ is the supervisor. We define the augmented attacked system as $(S/G)_A^{aug} = (X_z, \Sigma_z, \delta_z, x_{0,z}) = Ac(X_A \times 2^X, \Sigma \cup \Sigma_T, \delta_z, x_{0,z})$, where:

- $X_z \subseteq X_A \times 2^X$ is the state set;
- $\Sigma_z = \Sigma \cup \Sigma_T$ is the event set;
- $\delta_z : (X_A \times 2^X) \times \Sigma_z \rightarrow (X_A \times 2^X)$ is the transition function defined, for $(x_A, R) = ((x, y), R) \in X_A \times 2^X$ and $e \in \Sigma \cup \Sigma_T$, as

$$\delta_z((x_A, R), e) = \begin{cases} (\delta_A(x_A, e), R'), & \text{if } \delta_A(x_A, e)!, \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Here, R' is defined as follows: if $P_G(e) \in \Sigma_o$, then $R' = UR_{En_S(\delta_s(y, P_S(e)))}(Next(R, P_G(e))) = \{x' \in X \mid \exists x \in Next(R, P_G(e)), \exists t \in (\Sigma_{uo} \cap En_S(\delta_s(y, P_S(e))))^* : \delta(x, t) = x'\}$; if $P_G(e) \in \Sigma_{uo}$, then $R' = R$.

- $x_{0,z} = (x_{0,A}, R_0) = ((x_0, y_0), R_0)$ is the initial state, with $R_0 = UR_{En_S(y_0)}(\{x_0\}) = \{x' \in X \mid \exists t \in (\Sigma_{uo} \cap En_S(y_0))^* : \delta(x_0, t) = x'\}$.

The augmented attacked system $(S/G)_A^{aug}$ extends the system $(S/G)_A$ by adding an extra component that tracks the attacker's estimated set of possible plant states based on its observations, allowing the identification of paths through which an attacker could compromise the TCSO property and enabling the synthesis of S_r .

The augmented attacked system $(S/G)_A^{aug}$ contains at most $|X| \times |Y| \times 2^{|X|}$ states. In the worst case scenario, we assume that all controllable events are vulnerable to attacks, which yields the following cardinality for the attack set Σ_T : $|\Sigma_T| = |\Sigma_I| + |\Sigma_D| + |\Sigma_R| = |\Sigma_o| + |\Sigma_o| + |\Sigma_o|^2$. Consequently, the augmented attacked system $(S/G)_A^{aug}$ contains at most $(|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) \cdot |X| \cdot |Y| \cdot 2^{|X|}$ transitions.

B. Synthesis of S_r

In this subsection, we synthesize a supervisor S_r such that the closed-loop system S_r/G satisfies TCSO. An algorithm is developed to compute supervisor S_r .

In augmented attacked system $(S/G)_A^{aug}$, there may exist states indicating that the attacker has successfully inferred that system G is in a secret state after sensor attacks. We denote the set of such undesired states by $X_{ur}((S/G)_A^{aug}) = \{x_z = (x_A, R) \in X_z \mid R \subseteq X_S\}$. Algorithm 1 presents the procedure for constructing the supervisor S_r .

Algorithm 1 constructs a refined supervisor S_r to prevent secret disclosure under attacks. It begins by initializing the necessary sets, building the supervisor S from the observer $Obs(G)$, and identifying the states that could reveal the secret even in the absence of attacks. The augmented attacked system $(S/G)_A^{aug}$ is then computed, and the worklist U is populated with all secret-revealing states. All transitions in $(S/G)_A^{aug}$ that could lead to a secret disclosure are identified, and the corresponding supervisor transitions in S are collected in Δ . To ensure maximal permissiveness, redundant transitions in Δ are removed, and finally, the refined supervisor S_r is synthesized by removing the transitions¹ in Δ from S and computing its accessible part $Ac(S_r)$.

The computational complexity of Algorithm 1 is analyzed as follows. Constructing the observer $Obs(G)$ from the plant $G = (X, \Sigma, \delta, x_0)$ has a complexity of $O(|\Sigma_o| \cdot 2^{|X|})$. Building the supervisor S from $Obs(G)$ by adding self-loops for all uncontrollable events not enabled at each observer state has a complexity of $O(|\Sigma| \cdot 2^{|X|})$. Constructing the augmented attacked system $(S/G)_A^{aug}$ involves generating up to $|X| \cdot 2^{2|X|}$ states, each with at most $(|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|)$ outgoing transitions, resulting in a time complexity of $O((|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) |X| 2^{2|X|})$. The iterative computation of the vulnerable set of transitions Δ involves scanning all transitions of $(S/G)_A^{aug}$ and updating the sets U, V . Since each transition must be visited once and the total number of transitions is bounded by the product of the number of states and the maximum number of outgoing transitions per state, this step has the same order of complexity as constructing $(S/G)_A^{aug}$, namely, $O((|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) |X| 2^{2|X|})$. The size of the set Δ satisfies $|\Delta| \leq |\Sigma_o| \cdot 2^{|X|}$. Next, the algorithm needs to check every transition $(y, \alpha, y') \in \Delta$. For each transition t , it constructs the modified automaton $(S/G)_A^{aug}$. This step requires traversing all transitions of the augmented attacked system $(S/G)_A^{aug}$, resulting in a time complexity of $O(|\Sigma_o| \cdot 2^{|X|} (|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) |X| 2^{2|X|})$. Subsequently, the algorithm checks whether $X_{ur}((S/G)_A^{aug}) = \emptyset$, which has the same order of complexity as constructing $(S/G)_A^{aug}$. Since constructing the modified automaton $(S/G)_A^{aug}$ and checking whether $X_{ur}((S/G)_A^{aug}) = \emptyset$ must be repeated for each transition in the set Δ , the total time complexity of the refinement phase is $O(2|\Sigma_o|^2 (|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) |X| 2^{4|X|})$. Finally, obtaining the supervisor S_r by removing transitions in Δ and taking its accessible part has a complexity of $O(|\Sigma_o| \cdot 2^{|X|})$. Therefore, in the worst case, the time complexity of Algorithm 1 is $O((3+2|\Sigma_o|^2 2^{2|X|}) (|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|) |X| 2^{2|X|} + (|\Sigma| + |\Sigma_o|) \cdot 2^{|X|})$, where $|X|$ denotes the number of states of the plant G , $|\Sigma|$ denotes the number of all events of G , and

¹Removing a transition (y, e, y') deletes only the arc between states y and y' and the associated event e ; the states y and y' remain unchanged.

Fig. 6. The observer $Obs(G)$ of plant G .

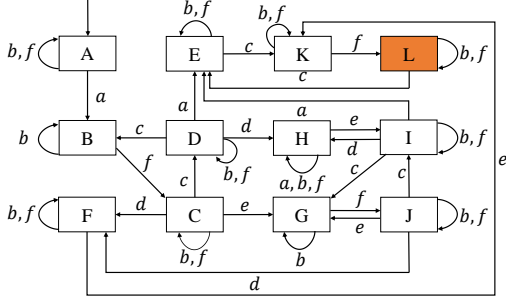


Fig. 7. The supervisor S derived from the observer $Obs(G)$.

Example 4. Consider the system G in Fig. 2, according to Definition 6 and Algorithm 1, we can construct the observer $Obs(G)$ of system G in Fig. 6, supervisor S in Fig. 7 and then construct the augmented attacked system $(S/G)_A^{aug}$ in Fig. 8. It is straightforward to observe that $X_{ur}((S/G)_A^{aug}) = \{((2, C), \{2\}), ((2, F), \{2\}), ((2, J), \{2\}), ((2, L), \{2\})\}$, which is highlighted in orange in Fig. 8. Algorithm 1 first computes a set of vulnerable transitions $\Delta = \{(C, e, G), (C, d, F), (E, c, K), (F, e, K), (I, c, K), (J, d, F), (J, e, G)\}$. Since re-enabling (C, d, F) or (J, d, F) while keeping other transitions in Δ disabled does not lead to a violation of TCSO, these transitions are excluded from Δ . The set Δ reduces to $\{(C, e, G), (E, c, K), (F, e, K), (I, c, K), (J, e, G)\}$. The supervisor S_r is constructed through Algorithm 1, as shown in Fig. 9.

C. Synthesis of S_r^k

During the synthesis of S_r , no limit is imposed on the total number of attacks. While this assumption allows us to explore the worst-case scenario, it is often unrealistic in practice, as attackers are constrained by factors such as time, cost, and detection risk. In this subsection, we consider a more realistic setting by introducing an upper bound k on the number of sensor attacks. If the attacker can only infer the current secret state after exceeding this bound, the system S/G is still regarded as having successfully preserved its secrecy. The objective in this subsection is to synthesize S_r^k such that system S_r/G satisfies TCSO under the constraint that the attacker can launch at most k sensor attacks.

To model the attack budget, a finite-state automaton is defined that tracks the number of active attacks launched. The attack-counting automaton I_k is defined as a tuple: $I_k = (Q_k, \Sigma_k, \delta_k, q_0)$, where

- $Q_k = \{0, 1, \dots, k\}$ is the state set, where each element is a natural number representing the number of attacks that have occurred so far, and k is the maximum number of attacks;
- $\Sigma_k = \Sigma \cup \Sigma_T$ is the event set;
- $\delta_k : Q_k \times \Sigma_k \rightarrow Q_k$ is the transition function defined for $q \in Q_k$ and $e \in \Sigma \cup \Sigma_T$ by

$$\delta_k(q, e) = \begin{cases} q, & \text{if } e \in \Sigma, \\ q + 1, & \text{if } e \in \Sigma_T \text{ and } q < k, \\ \text{undefined}, & \text{otherwise;} \end{cases}$$

- $q_0 = 0$ is the initial state, indicating that the attacker has not launched any attack yet.

The automaton I_k is shown in Fig. 10. To synthesize S_r^k , we construct the augmented attacked system with at most k sensor attacks $(S/G)_A^{K^{aug}}$ as the parallel composition of the augmented attacked system $(S/G)_A^{aug}$ and the automaton I_k , i.e., $(S/G)_A^{K^{aug}} = (X_z^k, \Sigma_z^k, \delta_z^k, x_{0,z}^k) = (S/G)_A^{aug} \parallel I_k$. In system $(S/G)_A^{K^{aug}}$, the attacker's capability is constrained by an upper bound k on the number of sensor attacks. We denote by $X_{ur}^k((S/G)_A^{K^{aug}}) = \{(x_A, R), q\} \in X_z^k \mid R \subseteq X_S\}$ the set of undesired states in which the attacker, operating within the allowed budget of at most k sensor deception attacks, has successfully inferred that G is in a secret state.

Algorithm 2 State Refinement for Augmented Attacked System with at most k sensor attacks $(S/G)_A^{K^{aug}}$.

Input: The augmented attacked system $(S/G)_A^{aug} = (X_z, \Sigma_z, \delta_z, x_{0,z})$.

Output: The refined augmented attacked system with at most k sensor attacks $(S/G)_A^{K^{aug}}$.

- 1: system $(S/G)_A^{K^{aug}} = (X_z^k, \Sigma_z^k, \delta_z^k, x_{0,z}^k) = (S/G)_A^{aug} \parallel I_k$;
- 2: Let $U = X_{ur}^k((S/G)_A^{K^{aug}})$;
- 3: Compute $X_p = \{x_z^k \in X_z^k \mid (\forall s \in (\Sigma \cup \Sigma_T)^*) \delta_z^k(x_z^k, s) \notin U\}$;
- 4: Refine $(S/G)_A^{K^{aug}}$ by removing all outgoing transitions of states in U , then deleting all states in X_p and taking its accessible part;
- 5: **for** each state $\dot{x} \in U$ **do**
- 6: **for** each transition (\dot{x}', e, \dot{x}) and (\dot{x}'', e, \dot{x}) in $(S/G)_A^{K^{aug}}$, where $\dot{x}' = (x_z', q')$, $\dot{x}'' = (x_z'', q'') \in X_z^k$, $q' < q''$ and $e \in \Sigma \cup \Sigma_T$ **do**
- 7: **if** for all $e \in En_{(S/G)_A^{K^{aug}}}(\dot{x}') \cup En_{(S/G)_A^{K^{aug}}}(\dot{x}'')$, it holds that $\delta_z^k(\dot{x}', e) = \delta_z^k(\dot{x}'', e)$ **then**
- 8: Merge state \dot{x} into state \dot{x}' by redirecting all incoming and outgoing transitions of \dot{x} to \dot{x}' and removing \dot{x} from $(S/G)_A^{K^{aug}}$;
- 9: $U = U \cup \{\dot{x}'\}$;
- 10: **end if**
- 11: $U = U \setminus \{\dot{x}\}$
- 12: **end for**
- 13: **end for**

To facilitate subsequent analysis, the augmented attacked system with at most k sensor attacks $(S/G)_A^{K^{aug}}$ is refined to remove irrelevant components while preserving all states and transitions related to potential attacks. Algorithm 2 presents the refinement procedure for $(S/G)_A^{K^{aug}}$. First, system $(S/G)_A^{K^{aug}}$ is constructed. Next, all states that do not affect the analysis of vulnerable transitions are removed, and the accessible part of the resulting automaton is taken. Then, states that share identical plant and supervisor components and have identical successor states for all transitions but differ only in their attack counter values are merged, while retaining the one with the smaller counter. After these refinements, the refined augmented attacked system with at most k sensor attacks $(S/G)_A^{K^{aug}}$ is obtained.

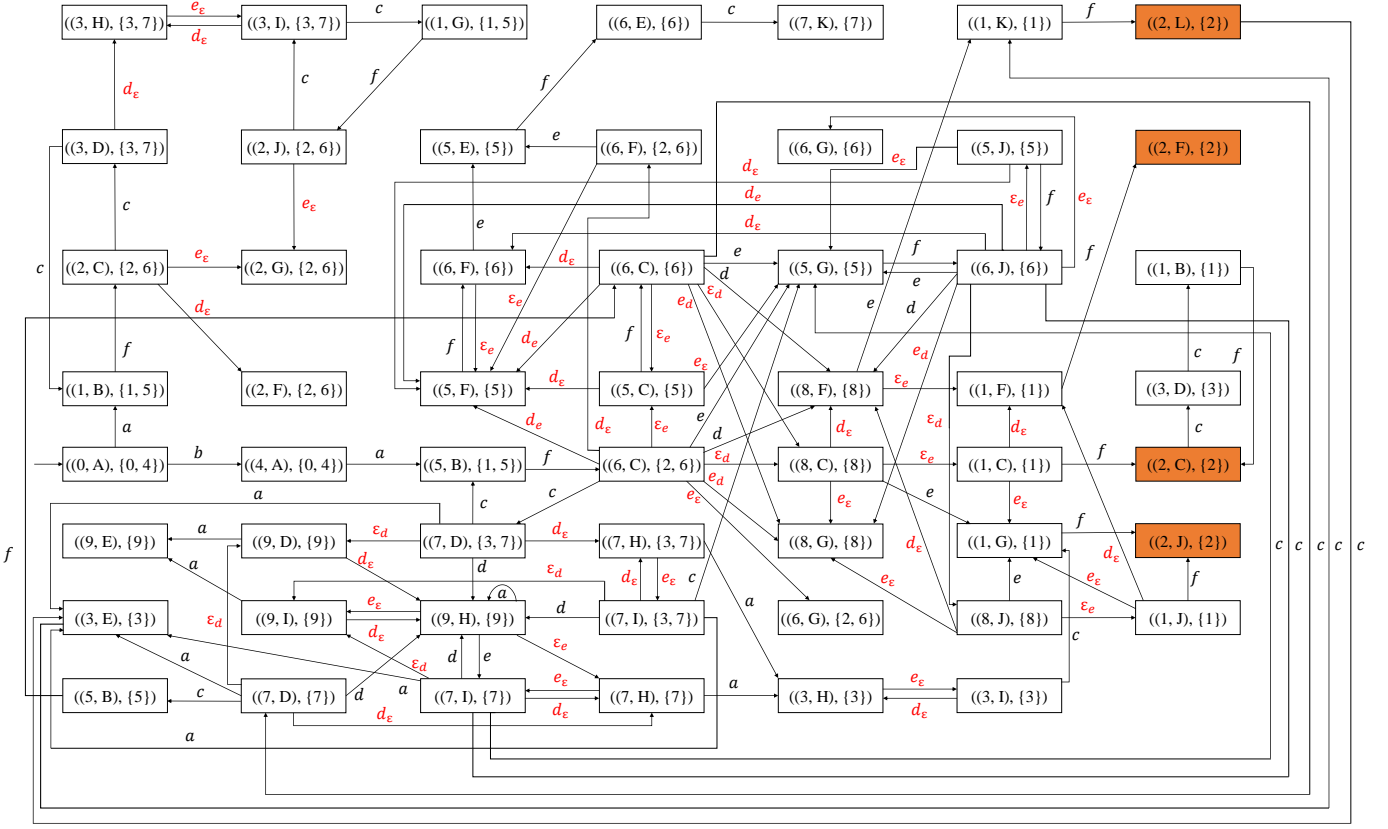


Fig. 8. The augmented attacked system $(S/G)_A^{aug}$.

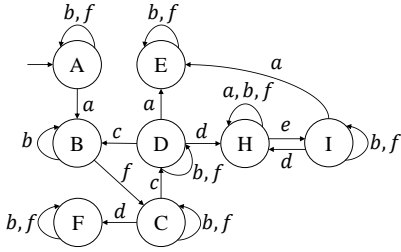


Fig. 9. The supervisor S_r .

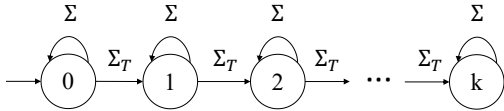


Fig. 10. The attacker limited to at most k attacks.

The computational complexity of Algorithm 2 is analyzed as follows. The construction of the augmented attacked system $(S/G)_A^{Kaug}$ and the removal of all outgoing transitions from states in $X_{ur}^k((S/G)_A^{Kaug})$ each require a time complexity of $O((|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|)(k+1)|X||Y|2^{|X|})$. Afterward, removing unreachable states (i.e., taking the accessible part) involves a traversal of the entire state space, resulting in the same order of time complexity, $O((|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|)(k+1)|X||Y|2^{|X|})$.

The subsequent state-merging procedure compares states that share identical plant and supervisor components but differ in their attack counter values, and reassigns their transitions accordingly. This process has a worst-case time complexity of $O((|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|)(k+1)^2|X|^2|Y|2^{|X|})$. Hence, the overall worst-case time complexity of Algorithm 2 is $O((3 + (k+1)|X||Y|2^{|X|})(k+1)(|\Sigma| + |\Sigma_o|^2 + 2|\Sigma_o|)|X||Y|2^{|X|})$.

To synthesize the supervisor under the bounded-attack model, S_r^k , we can achieve this by a simple modification of Algorithm 1. After constructing the augmented attacked system $(S/G)_A^{aug}$ in Algorithm 1, Algorithm 2 is then executed on $(S/G)_A^{aug}$, resulting in the refined system $(S/G)_A^{Kaug}$. The set of undesired states $X_{ur}((S/G)_A^{aug})$ is replaced with $X_{ur}^k((S/G)_A^{Kaug})$, and all components associated with $(S/G)_A^{aug}$ are replaced with their counterparts in $(S/G)_A^{Kaug}$ when computing the vulnerable set of transitions Δ . Finally, the supervisor S_r^k is obtained by removing all transitions in the vulnerable set Δ from S and taking the accessible part.

Theorem 3. *Given a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$ with a set of secret states X_S and a stealthy attacker with at most k sensor attacks. The supervisor S_r^k computed by Algorithm 1 together with Algorithm 2 enforces tamper-tolerant current-state opacity under k attack limits.*

Proof. To begin, we show that the refinement operations in Algorithm 2 do not affect the construction of Δ in Algorithm 1. First, the refinement removes all outgoing transitions from $X_{ur}^k((S/G)_A^{Kaug})$ and eliminates all states that cannot

reach any state in $X_{ur}^k((S/G)_A^{K^{aug}})$, followed by taking the reachable part of $(S/G)_A^{K^{aug}}$. Any forward path that reaches a state in $X_{ur}^k((S/G)_A^{K^{aug}})$ ends at that state and does not rely on its outgoing transitions; likewise, states that cannot reach any state in $X_{ur}^k((S/G)_A^{K^{aug}})$ never participate in any path relevant to Δ . Therefore, these removals eliminate only irrelevant components and do not affect the construction of Δ in Algorithm 1. Next, states that share the same plant and supervisor components and have identical successors but differ only in the attack-counter value are merged, keeping the one with the smaller counter, thus producing the same set of transitions that would be classified into Δ . Hence, the computation of the transition set Δ in Algorithm 1 remains unaffected. The overall correctness argument follows the proof structure of Theorem 2. \square

VI. CONCLUSION AND FUTURE WORK

Example 5. Consider the system G in Fig. 2, our objective is to synthesize S_r^2 , capable of withstanding two sensor attacks.

REFERENCES

- [1] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—a survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1802–1831, 2017.
- [2] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.
- [3] A. Saboori and C. N. Hadjicostis, "Current-state opacity formulations in probabilistic finite automata," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 120–133, 2013.
- [4] R. J. Barcelos and J. C. Basilio, "Enforcing current-state opacity through shuffle and deletions of event observations," *Automatica*, vol. 133, p. 109836, 2021.
- [5] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of discrete event systems," *Information Sciences*, vol. 246, pp. 115–132, 2013.
- [6] X. Yin and S. Lafortune, "A new approach for the verification of infinite-step and k-step opacity using two-way observers," *Automatica*, vol. 80, pp. 162–171, 2017.
- [7] Z. Ma, X. Yin, and Z. Li, "Verification and enforcement of strong infinite-and k-step opacity using state recognizers," *Automatica*, vol. 133, p. 109838, 2021.
- [8] A. Saboori and C. N. Hadjicostis, "Verification of infinite-step opacity and complexity considerations," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1265–1269, 2011.
- [9] A. Saboori and C. N. Hadjicostis, "Verification of k -step opacity and analysis of its complexity," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 549–559, 2011.
- [10] I. Saadaoui, Z. Li, and N. Wu, "Current-state opacity modelling and verification in partially observed petri nets," *Automatica*, vol. 116, p. 108907, 2020.
- [11] A. Saboori and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies via state estimator constructions," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1155–1165, 2011.
- [12] P. Darondeau, H. Marchand, and L. Ricker, "Enforcing opacity of regular predicates on modal transition systems," *Discrete Event Dynamic Systems*, vol. 25, pp. 251–270, 2015.
- [13] Y. Falcone and H. Marchand, "Enforcement and validation (at runtime) of various notions of opacity," *Discrete Event Dynamic Systems*, vol. 25, pp. 531–570, 2015.
- [14] Y. Xie, X. Yin, and S. Li, "Opacity enforcing supervisory control using nondeterministic supervisors," *IEEE Transactions on Automatic Control*, vol. 67, no. 12, pp. 6567–6582, 2021.
- [15] Y. Ji, Y.-C. Wu, and S. Lafortune, "Enforcement of opacity by public and private insertion functions," *Automatica*, vol. 93, pp. 369–378, 2018.
- [16] Y. Ji, X. Yin, and S. Lafortune, "Enforcing opacity by insertion functions under multiple energy constraints," *Automatica*, vol. 108, p. 108476, 2019.
- [17] L. K. Carvalho, Y.-C. Wu, R. Kwong, and S. Lafortune, "Detection and mitigation of classes of attacks in supervisory control systems," *Automatica*, vol. 97, pp. 121–133, 2018.
- [18] R. Meira-Góes, H. Marchand, and S. Lafortune, "Dealing with sensor and actuator deception attacks in supervisory control," *Automatica*, vol. 147, p. 110736, 2023.
- [19] R. Meira-Góes, S. Lafortune, and H. Marchand, "Synthesis of supervisors robust against sensor deception attacks," *IEEE Transactions on Automatic Control*, vol. 66, no. 10, pp. 4990–4997, 2021.
- [20] R. Su, "On decidability of existence of nonblocking supervisors resilient to smart sensor attacks," *Automatica*, vol. 154, p. 111076, 2023.
- [21] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems," *Automatica*, vol. 121, p. 109172, 2020.
- [22] R. Su, "Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations," *Automatica*, vol. 94, pp. 35–44, 2018.
- [23] W. Duo, S. Wang, and M. Zhou, "Multi-step sensor attackability in cyber-physical systems," *IEEE Transactions on Automatic Control*, 2025.
- [24] J. Yao, S. Li, and X. Yin, "Sensor deception attacks against security in supervisory control systems," *Automatica*, vol. 159, p. 111330, 2024.
- [25] Q. Chen, R. Su, and Z. Li, "Synthesis of sensor attacks for tampering detectability of partially observed discrete-event systems," *IEEE Transactions on Automatic Control*, 2025.
- [26] Y. Li, C. N. Hadjicostis, N. Wu, and Z. Li, "Error-and tamper-tolerant state estimation for discrete event systems under cost constraints," *IEEE Transactions on Automatic Control*, vol. 68, no. 11, pp. 6743–6750, 2023.
- [27] Y. Li, C. N. Hadjicostis, N. Wu, and Z. Li, "Tamper-tolerant diagnosability analysis and tampering detectability in discrete event systems under cost constraints," *Automatica*, vol. 171, p. 111971, 2025.