Attack-Resilient Supervisory Control of Discrete Event Systems under Dynamic-Event-Protection Mechanisms

Bohan Cui, Alessandro Giua and Xiang Yin*†

April, 2026

Abstract

We investigate the problem of synthesizing safe supervisors for discrete-event systems under actuator attacks, where an adversary can partially override control commands at vulnerable states. We introduce a novel dynamic-event-protection mechanism, where the system can defend itself from attacks by taking defense actions when it meets certain required safety levels. The system employs two policies: a safety-enhancement policy that dynamically manipulates protecting events to increase the safety level, and a state-defense policy that determines whether to defend against attacks when sufficient safety levels are accumulated. Our goal is to synthesize a attack-resilient supervisor, along with compatible safety-enhancement and state-defense policies, to ensure the closed-loop system remains safe under any possible attacks on vulnerable states. We provide a sound and complete approach for synthesizing the supervisor and policies by formulating the problem as a safety game played on a multilayered duplication structure of the original system. We illustrate the proposed approach by running examples.

Published as:
B. Cui, A. Giua, and X. Yin, "Attack-Resilient Supervisory Control of Discrete Event Systems Under Dynamic-Event-Protection Mechanisms" *IEEE Control Systems Letters* (Volume: 9), pp. 1189-1194, 2025.
DOI: 10.1109/LCSYS.2025.3578986

*This work was supported by the NSFC Grants 62173226, 61803259.

†Bohan Cui and Xiang Yin are with the School of Automation and Sensing, Shanghai Jiao Tong University, Shanghai 200240, China. {bohan_cui, yinxiang}@sjtu.edu.cn. Alessandro Giua is with the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari 09123, Italy. giua@unica.it.

1 Introduction

Supervisory control of discrete-event systems (DES) is a widely used formal approach for controller synthesis in industrial control systems and engineering cyber-physical systems [2]. In modern engineering systems, supervisory control systems are often implemented via communication networks, making them vulnerable to attacks. For example, an attacker may manipulate sensor readings or actuator commands, causing incorrect control logic to be executed and leading the system to a dangerous state. In recent years, supervisory control of DES under attacks has garnered significant attention in the literature; see, e.g., [1, 3, 4, 6, 12, 13, 15, 17, 18].

Existing works on attack modeling and defense in the context of supervisory control of DES can date back to [14], which considers supervisor synthesis under actuator enablement attacks. In [16], both the sensor and actuator attacks are considered by modeling as nondeterministic finite state transducers. In [1], a generic attack detection and mitigation strategy was proposed, where four types of attacks are taken into consideration. In addition to investigating different types of attacks, research on attacks in DES has also been conducted from multiple perspectives, such as synthesizing attack strategies from the attacker's point of view [5, 10], and synthesizing the supervisor that is robustly safe under any possible attacks [11].

In many applications, a system can spend additional effort to enhance its safety level and protect itself from external attacks. For example, in networked systems, by expanding server capacity or enhancing network infrastructure, the system can absorb and mitigate the overwhelming traffic generated by attackers during a DDoS attack. In the context of DES, recent works [7–9] have explored the *secret protection problem*, where the system's secret is assumed to be stored in specific states. To secure the secret, it is necessary to enhance the system's security level. These studies focus on synthesizing optimal secret protection strategies that prevent intruders from accessing the secret while minimizing the associated cost.

In this paper, we investigate the attack-resilient supervisor synthesis problem for DES. Specifically, we focus on the actuator attack setting, where an adversary can partially override control decisions at vulnerable states. Inspired by recent works on secret protection [7–9], we consider a scenario where the system can increase its safety level to safeguard control decisions. The supervisory control system is equipped with two policies: a *safety-enhancement policy* that dynamically manipulates protecting events to raise the safety level, and a *state-defense policy* that determines whether to defend against attacks at vulnerable states when sufficient safety levels are accumulated. Once a defense action is taken, the safety level is reset to zero, and the system needs to rebuild it. We refer to this setting as the *dynamic-event-protection mechanism*, as the safety level changes dynamically based on the system's actions and decisions. Our objective is to design a supervisor, along with a safety-enhancement policy and a state-defense policy, ensuring overall safety against any possible attacks launched at vulnerable states.

The main contributions of this paper are summarized as follows. First, we present a formal model for the above described dynamic-event-protection mechanism. Then we define the new structure of safety level automaton (SLA), a multilayered structure that captures the evolution of safety level by accounting for the effects of the safety-enhancement and state-defense policies. Finally, we show that the attack-resilient supervisor synthesis problem can be effectively solved by unfolding the SLA to incorporate the supervisor's decisions and all possible attacks. Compared with the works on secret protection [7–9], our work investigates a dynamic setting where the safety level can decrease due to defense actions. Moreover, here we need to consider the joint effects of the supervisor and the protection policies, which does not exist in the secret protection problem.

2 Preliminaries

2.1 System Model

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

We consider a DES modeled by a deterministic finite-state automaton (DFA) $G = (X, \Sigma, \delta, x_0)$, where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function such that $\delta(x, \sigma) = x'$ denotes the transition labeled by σ from state x to state x', and $x_0 \in X$ is the unique initial state. The transition function is also extended to $\delta : X \times \Sigma^* \to X$ recursively by: (i) for any $x \in X$, $\delta(x, \epsilon) = x$; and (ii) for any $x \in X, s \in \Sigma^*, \sigma \in \Sigma$, we have $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$. The set of all strings generated by G starting from state $x \in X$ is defined as $\mathcal{L}(G, x) = \{s \in \Sigma^* : \delta(x, s)!\}$, where "!" means "is defined". The language generated by G is defined as $\mathcal{L}(G) := \mathcal{L}(G, x_0)$. For simplicity, we write $\delta(x_0, s)$ as $\delta(s)$ for any $s \in \mathcal{L}(G)$.

2.2 Standard Supervisory Control Theory

In the standard framework of supervisory control theory, the event set is partitioned as

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc},$$

where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. A supervisor is a mechanism that dynamically disables controllable events to enforce some specifications. We denote $\Gamma = \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. Then a supervisor is a function $S : \mathcal{L}(G) \to \Gamma$. We use notation S/G to represent the controlled system and the language generated by S/G, denoted by $\mathcal{L}(S/G)$ is defined recursively by:

(i) $\epsilon \in \mathcal{L}(S/G)$;

(ii) $[s \in \mathcal{L}(S/G) \wedge s\sigma \in \mathcal{L}(G) \wedge \sigma \in S(s)] \Leftrightarrow [s\sigma \in \mathcal{L}(S/G)]$.

The safety of the system is modeled by a prefix-closed sub-language $K \subseteq \mathcal{L}(G)$. The standard safe control problem is to design a supervisor S such that $\mathcal{L}(S/G) \subseteq K$ and as permissive as possible. Without loss of generality, we assume that K is generated by $H = (X_H, \Sigma, \delta_H, x_{H,0})$, which is a *sub-automaton* of G [2]. Then for any $s \in \mathcal{L}(G)$, we have $s \in K$ if it does not pass through any states in $X \setminus X_H$.

3 Attack-Protection Supervisory Control

In this section, we formally present the attack-protection model that we investigate in this paper.

3.1 System Attack Mechanism

Attacker Model: We assume that attack can only be launched at some *vulnerable states* , denoted as

$$X_V \subseteq X.$$

That is, the attacker can only influence the system only upon reaching these states. We define the set of strings leading to vulnerable states as $\mathcal{L}_V(G) = \{s \in \mathcal{L}(G) : \delta(s) \in X_V\}$. Then let Ξ be a finite set of symbols representing possible attack actions. Similar to the supervisor, the *attacker* is therefore a function that dynamically issues attack actions but only at vulnerable states, i.e.,

$$A : \mathcal{L}_V(G) \to \Xi.$$

Attack Effect: The effect of the attacker on the supervisory control system is captured by a *fusion rule*

$$\oplus : \Gamma \times \Xi \to 2^{\Sigma}.$$

Specifically, given an original control decision $\gamma \in \Gamma$ and an attack decision $\xi \in \Xi$, the resulting overall control decision taken by the system is given by $\oplus(\gamma, \xi)$. Note that, our definition of attack actions Ξ and the fusion rule ξ are generic; the specific forms depend on the underlying attack setting. Hereafter, we focus on a specific type of attack called the *actuator-enablement* attack (AE-attack), where the attacker can enable some events that are originally disabled by the supervisor. In this setting, let $\Sigma_a \subseteq \Sigma_c$ be the set of events that can be enabled by the attacker. Then the action space of the attacker can be concretized by $\Xi = \{\xi \in 2^{\Sigma} : \xi \in \Sigma_a\}$ and the fusion rule becomes $\oplus(\gamma, \xi) = \gamma \cup \xi$.

3.2 System Protection Mechanism

In this paper, we consider a general system protection mechanism based on the *safety level* of the system. Specifically, the system can choose to protect itself from being attacked by the attacker based on two key factors:

- Whether it has accumulated a sufficient safety level, and
- Whether it decides to take a protection action once it has acquired enough safety level.

Formally, for each vulnerable state, we assign a *required safety level* by function

$$\ell : X_V \to \mathbb{N}^+,$$

which specifies the minimal safety level the system must maintain to launch a protection action against potential attacks. We denote by $\ell_m = \max_{x \in X_V} \ell(x)$ the maximal required safety level requirement among all vulnerable states.

Safety-Enhancement Policy: We assume that the safety level of the system can be enhanced upon the occurrence of certain events called *protecting events*, denoted as:

$$\Sigma_p \subseteq \Sigma.$$

These events model system behaviors that allow safety operations, such as network traffic monitoring or system backup, to be conducted simultaneously, thereby increasing the system's safety level. We also define $\Sigma_{up} = \Sigma \setminus \Sigma_p$. Note that, there is no relationship between Σ_p and Σ_c. The safety level is enhanced when the

occurred event is protected. For cost reasons, the system needs to dynamically protect events based on a *safety-enhancement policy*:

$$\pi : \mathcal{L}(G) \times \Sigma_p \to \{0, 1\},$$

where $\pi(s, \sigma) = 1$ means that $\sigma \in \Sigma_p \cap \Delta_{\mathcal{L}(G)}(s)$ is protected after $s \in \mathcal{L}(G)$ is generated. We also extend π to $\mathcal{L}(G) \times \Sigma \to \{0, 1\}$ by setting $\pi(s, \sigma) = 0$ for any $\sigma \in \Sigma_{up}$.

State-Defense Policy: Note that accumulating a sufficient safety level does not necessarily imply that the system is automatically defended. The system must actively take defense actions through a *state-defense policy*

$$D : \mathcal{L}_V(G) \to \{0, 1\},$$

where $D(s) = 1$ indicates that the system chooses to defend against an attack at the vulnerable state reached by $s \in \mathcal{L}(G)$, and $D(s) = 0$ means no defense is applied. We can extend D to $\mathcal{L}(G) \to \{0, 1\}$ by setting $D(s) = 0$ for any $s \notin \mathcal{L}_V(G)$. In this paper, we assume that when a state is defended, the next control decision cannot be attacked. This defense holds only until a new event occurs.

Safety Level Evolution: For the sake of simplicity, we assume that the safety level of the system evolves as follows:

- Whenever the system executes a protected event and does not reach a vulnerable state (meaning the safety accumulation will not be consumed in the next step), the safety level increases by one unit.
- Once a state protection action is taken, the safety level of the system is fully consumed, resetting to 0.

Therefore, given a safety-enhancement policy π and a state-defense policy D, for each string $s \in \mathcal{L}(G)$, we denote by $N_{\pi,D}(s)$ the safety level of the system upon string s. By assuming that the initial safety level of the system is 0, the safety level of the system is defined recursively by:

(i) $N_{\pi,D}(\epsilon) = 0$;

(ii) for any $s\sigma \in (\mathcal{L}(G) \setminus \mathcal{L}_V(G))$, we have

$$N_{\pi,D}(s\sigma) = N_{\pi,D}(s) + \pi(s, \sigma)$$

(iii) for any $s\sigma \in \mathcal{L}_V(G)$, we have

$$N_{\pi,D}(s\sigma) = \begin{cases} 0 & \text{if } D(s\sigma) = 1 \\ N_{\pi,D}(s) + \pi(s, \sigma) & \text{otherwise} \end{cases}$$

3.3 Resulting Supervisor and Problem Formulation

Note that, the system can take protection actions only when it meets the required safety level associated with the vulnerable state. Therefore, the safety-enhancement policy and the state-defense policy must be compatible.

Definition 1 (Compatible Policies)**.** We say a safety-enhancement policy π and a state-defense policy D are *compatible* if for any $s\sigma \in \mathcal{L}_V(G)$, we have

$$D(s\sigma) = 1 \quad \Rightarrow \quad N_{\pi,D}(s) + \pi(s, \sigma) \geq \ell(\delta(s\sigma)).$$

Hereafter, we will refer to the combination of a supervisor S, a safety-enhancement policy π, and a compatible state-defense policy D as a *policy pattern* (S, π, D). This policy pattern encapsulates all the decisions the system must make, including control actions, event protections, and state defenses. Given a *policy pattern* (S, π, D), attacker A and fusion rule $\oplus$, we denote by $S^{\pi,D}_{A,\oplus}$ as a resulting overall supervisor, which is defined by: for any $s \in \mathcal{L}(G)$, we have

$$S^{\pi,D}_{A,\oplus}(s) = \begin{cases} \oplus(S(s), A(s)) & \text{if } D(s){=}0 \text{ and } \delta(s){\in} X_V \\ S(s) & \text{otherwise} \end{cases}$$

We denote by $\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)$ the language generated by the attack-protection system. Then the problem we investigate is formulated as follows:

Problem 1. Given G with vulnerable states X_V and the required safety level $\ell : X_V \to \mathbb{N}^+$, specifications $K \subseteq \mathcal{L}(G)$, controllable events Σ_c, protecting events Σ_p, attack action set Ξ, and fusion rule $\oplus$, find a *policy pattern* (S, π, D) such that π and D are compatible, and for any possible attacker A, we have $\mathcal{L}(S^{\pi,D}_{A,\oplus}/G) \subseteq K$.

Note that we do not impose a permissiveness requirement in Problem 1 because there is a trade-off between permissiveness and the number of necessary protections for events. This trade-off is illustrated in the following example.

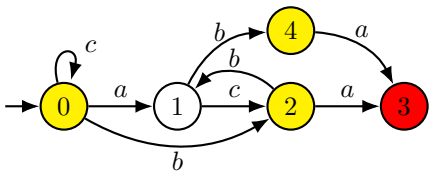


Figure 1: System G with $\Sigma_c = \{a, b\}$, $\Sigma_a = \{a\}$, $\Sigma_p = \{b, c\}$, $X_V = \{0, 2, 4\}$, $\ell(0) = \ell(2) = 1$, $\ell(4) = 2$ and $X_H = \{0, 1, 2, 4\}$.

Example 1. Consider system G in Figure 1. The specification K is to avoid reaching state 3. The vulnerable states are 0, 2, and 4, with required safety levels $\ell(0) = \ell(2) = 1$ and $\ell(4) = 2$. To ensure safety, event a must be disabled at states 2 and 4. This requires $D(s) = 1$ for all $\delta(s) \in \{2, 4\}$, meaning the safety level must exceed 1 and 2 upon reaching states 2 and 4, respectively. To achieve this, event b must be consistently protected to increase the safety level, and event c should be protected at the beginning. If event a occurs and the system transitions to state 1, event b should be disabled, and event c should be protected. To be more specific, consider string bbb. After the first b, the system reaches state 2, which requires immediate defense ($D(b) = 1$). Therefore, we have $\pi(\epsilon, b) = 1$ to increase the safety level to 1 before the defense. Since the defense resets the safety level to 0. To ensure the required safety level $\ell(4) = 2$ when reaching state 4, the next two b's must also be protected.

While the above safety-enhancement policy results in a maximally permissive closed-loop behavior, it requires infinite protections of b, as states 1 and 2 form a loop involving events b and c. An alternative approach is to consistently disable event b, requiring only the protection of event c at state 1. This illustrates the trade-off between the permissiveness of the supervisor and the number of protections enforced by the safety-enhancement policy. In this paper, we focus on developing a supervisory control strategy that meets safety requirements while maintaining feasible system operation under all possible attacks. The analysis of optimality, including minimizing protections and maximizing permissiveness, is left for future work.

4 Synthesis Procedure

In this section, we present the synthesis procedure of the overall supervisor. We conduct this by first considering the influence of the safety-enhancement policy and the state-defense policy and then involving the decisions of the supervisor and the attacker.

4.1 Safety Level Automaton

First, we introduce the safety level automata (SLA), which captures the evolution of the safety level with respect to safety-enhancement policy π and state-defense policy D.

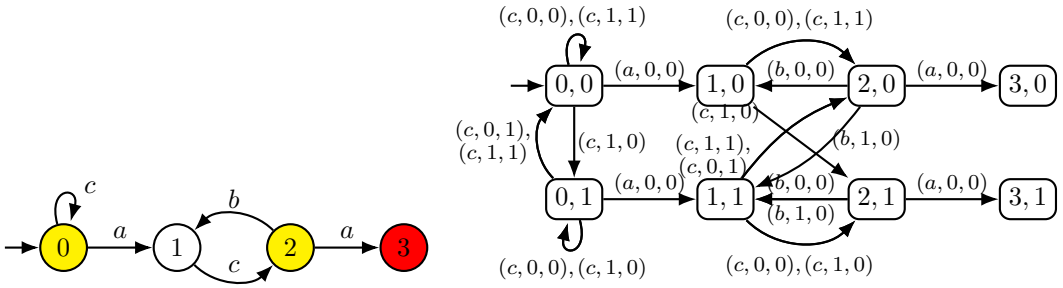
Definition 2 (Safety Level Automata)**.** Given system G and safety level requirement $\ell : X_V \to \mathbb{N}^+$ with maximal level ℓ_m, the safety level automaton (SLA) is defined by

$$L = (Q, \Sigma_L, f, q_0),$$

where

- $Q \subseteq X \times \{0, \ldots, \ell_m\}$ is the set of states;
- $\Sigma_L = \Sigma \times \{0, 1\} \times \{0, 1\}$ is the set of events;
- $f : Q \times \Sigma_L \to Q$ is the transition function defined by: for any $(x, n) \in Q$ and $(\sigma, \imath, \imath') \in \Sigma_L$, we have:
  - $f((x, n), (\sigma, 0, 0)) = (\delta(x, \sigma), n)$ for any $n \in \{0, \ldots, \ell_m\}$;
  - $f((x, n), (\sigma, 1, 0)) = (\delta(x, \sigma), n + 1)$ for any $n \in \{0, \ldots, \ell_m - 1\}$, $\sigma \in \Sigma_p$;
  - $f((x, \ell_m), (\sigma, 1, 0)) = (\delta(x, \sigma), \ell_m)$ for any $\sigma \in \Sigma_p$;
  - $f((x, n), (\sigma, 0, 1)) = (\delta(x, \sigma), 0)$ for any $\delta(x, \sigma) \in X_V$, $n \geq \ell(\delta(x, \sigma))$; and
  - $f((x, n), (\sigma, 1, 1)) = (\delta(x, \sigma), 0)$ for any $\delta(x, \sigma) \in X_V$, $\sigma \in \Sigma_p$, $n + 1 \geq \ell(\delta(x, \sigma))$;
- $q_0 = (x_0, 0)$ is the initial state.

Intuitively, the SLA L simulates system G under all possible safety-enhancement and state-defense policies. Specifically, every state $(x, n) \in Q$ indicates that the current state is x and the current safety level is n. For each event $(\sigma, \imath, \imath') \in \Sigma_L$, $\imath = 1$ means that σ is protected by the safety-enhancement policy while $\imath' = 1$ means that the next state is defended by state-defense policy after the occurrence of σ. This is why we have $f((x, n), (\sigma, 1, \imath'))!$ iff $\sigma \in \Sigma_p$, and $f((x, n), (\sigma, \imath, 1))!$ iff $\delta(x, \sigma) \in X_V$ and $n + \imath \geq \ell(\delta(x, \sigma))$.



(a) System model G.

(b) SLA L.

Figure 2: Running example G with $\Sigma_c = \{a, b\}$, $\Sigma_a = \{a\}$, $\Sigma_p = \{b, c\}$, $X_V = \{0, 2\}$, $\ell(0) = \ell(2) = 1$, and $X_H = \{0, 1, 2\}$

Example 2. Let us consider system G shown in Figure 2(a). The corresponding safety level automaton L is as shown in Figure 2(b). The initial state of L is $(0, 0)$. Starting from $(0, 0)$, transition $f((0, 0), (c, 1, 0)) = (0, 1)$ represents the fact that the safety level will increase from 0 to 1 when the protected event c occurs and $D(c) = 0$, while transition $f((0, 0), (c, 1, 1)) = (0, 0)$ represents that the safety level will remain 0 if (i) c is protected, and (ii) state 0 is protected upon the occurrence of c, i.e., $D(c) = 1$.

The relationship with strings in L with strings in the original system under π and D is captured as follows.

Definition 3 (Safety Strings)**.** Given the system G, safety-enhancement policy π and state-defense policy D, for any string $s = \sigma_1\sigma_2 \cdots \in \mathcal{L}(G)$, its corresponding safety string $s_{\texttt{safe}} = (\sigma_1, \imath_1, \imath'_1)(\sigma_2, \imath_2, \imath'_2) \ldots$ (w.r.t. π and D) is defined by: for any $(\sigma_i, \imath_i, \imath'_i), i = 1, 2, \ldots$, we have

- $\imath_i = 1$ if $\pi(\sigma_1 \ldots \sigma_{i-1}, \sigma_i) = 1$; and
- $\imath'_i = 1$ if $D(\sigma_1 \ldots \sigma_i) = 1$.

Then we have the following property for the SLA.

Proposition 1. Given system G with policies π and D, for any $s \in \mathcal{L}(G)$, we have $s_{\texttt{safe}} \in \mathcal{L}(L)$. Moreover, we have $f(s_{\texttt{safe}}) = (\delta(s), \min[N(s), \ell_m])$.

Proof. We prove this by induction of the length of s.

Induction Basis: Suppose that $|s| = 0$. Then we know that $s_{\texttt{safe}} = \epsilon$. It is clear that $f(s_{\texttt{safe}}) = q_0 = (x_0, 0) = (\delta(\epsilon), N(\epsilon))$. Therefore, the induction basis holds.

Induction Step: Now suppose that Proposition 1 holds for $s \in \mathcal{L}(G)$ wtih $|s| = k$, then for any $s\sigma \in \mathcal{L}(G)$:

- if $\pi(s, \sigma) = 0$ and $D(s\sigma) = 0$, then we have $s\sigma_{\texttt{safe}} = s_{\texttt{safe}}(\sigma, 0, 0)$ and $N(s\sigma) = N(s)$. According to the definition of f, we have $f(s\sigma_{\texttt{safe}}) = f(f(s_{\texttt{safe}}), (\sigma, 0, 0)) = (\delta(\delta(s), \sigma), \min[N(s), \ell_m]) = (\delta(s\sigma), \min[N(s\sigma), \ell_m])$;
- if $\pi(s, \sigma) = 1$ and $D(s\sigma) = 0$, then we have $s\sigma_{\texttt{safe}} = s_{\texttt{safe}}(\sigma, 1, 0)$ and $N(s\sigma) = N(s) + 1$. According to the definition of f, if $N(s) < \ell_m$, then we have $f(s\sigma_{\texttt{safe}}) = f(f(s_{\texttt{safe}}), (\sigma, 1, 0)) = (\delta(\delta(s), \sigma), N(s) + 1) = (\delta(s\sigma), \min[N(s\sigma), \ell_m])$; otherwise, we have $f(s\sigma_{\texttt{safe}}) = f(f(s_{\pi,D}), (\sigma, 1, 0)) = (\delta(\delta(s), \sigma), \ell_m) = (\delta(s\sigma), \min[N(s\sigma), \ell_m])$;
- if $\pi(s, \sigma) = 0$ and $D(s\sigma) = 1$, then we have $s\sigma_{\texttt{safe}} = s_{\texttt{safe}}(\sigma, 0, 1)$ and $N(s\sigma) = 0$. According to the definition of f, we have $f(s\sigma_{\texttt{safe}}) = f(f(s_{\texttt{safe}}), (\sigma, 0, 1)) = (\delta(\delta(s), \sigma), 0] = (\delta(s\sigma), \min[N(s\sigma), \ell_m])$;
- if $\pi(s, \sigma) = 1$ and $D(s\sigma) = 1$, then we have $s\sigma_{\texttt{safe}} = s_{\texttt{safe}}(\sigma, 1, 1)$ and $N(s\sigma) = 0$. According to the definition of f, we have $f(s\sigma_{\texttt{safe}}) = f(f(s_{\texttt{safe}}), (\sigma, 1, 1)) = (\delta(\delta(s), \sigma), 0] = (\delta(s\sigma), \min[N(s\sigma), \ell_m])$.

The proof is now complete. □

4.2 Synthesis of The Supervisor

Note that the SLA alone is not sufficient for supervisor synthesis, as it does not account for the potential effect of attacks. To incorporate the impact of attacks, we need to further classify language $\mathcal{L}_V(G)$ into two parts. When $D(s) = 0$, it indicates that the state $\delta(s)$ is not defended. In this case, upon issuing a control decision γ, the supervisor has to consider the union of all possible overall control decisions $\bigcup_{\xi \in \Xi} \oplus(\gamma, \xi)$. Note that since this paper focuses on AE-attacks, it suffices to treat Σ_a as uncontrollable events in this scenario, i.e., $\bigcup_{\xi \in \Xi} \oplus(\gamma, \xi) = \gamma \cup \Sigma_a$. On the other hand, when $D(s) = 1$, meaning the state $\delta(s)$ is defended, the supervisor can still consider the set of controllable events as Σ_c. Therefore, to capture the behavior of the closed-loop system under possible attacks, we need to further unfold the SLA by incorporating the effect of the attacker.

To this end, we first define a new set of symbols capturing admissible decisions of the safety-enhancement policy:

$$\Gamma^\pi = \{\gamma^\pi \in 2^{\Sigma\times\{0,1\}} : \Sigma^\pi_{uc} \subseteq \gamma^\pi\},$$

where $\Sigma^\pi_{uc} = (\Sigma_p \times \{1\}) \cup (\Sigma_{up} \times \{0\})$. Specifically, for any $s \in \mathcal{L}(G)$, given an admissible decision $\gamma^\pi \in \Gamma^\pi$, the corresponding safety-enhancement decision $\pi(s,\sigma)$ is defined by: for each $\sigma \in \Delta_{\mathcal{L}(G)}(s)$, if $(\sigma,0) \notin \gamma^\pi$, then $\pi(s,\sigma) = 1$. Therefore, we also denote $\Gamma^\pi(s) \in 2^{\Sigma\times\{0,1\}}$ as the corresponding decision of the safety-enhancement policy π upon the occurrence of s with a slight abuse of the notation. Now, we define the unfolded SLA as follows.

Definition 4 (Unfolded SLA)**.** An unfolded safety level automaton w.r.t. SLA L is a tuple

$$U = (Q_Y, Q_Z, Q_W, \delta_{YZ}, \delta_{ZW}, \delta_{WY}, \Gamma, \Gamma^\pi, \Sigma_L, y_0),$$

where

- $Q_Y \subseteq Q \times \{0,1\}$ is the set of Y-states;
- $Q_Z \subseteq Q \times \{0,1\} \times \Gamma \times \Gamma^\pi$ is the set of Z-states, where $\Gamma(z)$ and $\Gamma^\pi(z)$ denote its control decision and the safety-enhancement components, respectively;
- $Q_W = Q \times \Sigma \times \{0,1\}$ is the set of W-states;
- $\delta_{YZ} : Q_Y \times \Gamma \times \Gamma^\pi \to Q_Z$ is a deterministic transition function from Y-states to Z-states which satisfies the following constraint: for any $y \in Q_Y$, $\gamma \in \Gamma$, $\gamma^\pi \in \Gamma^\pi$, and $z \in Q_Z$ such that $z = \delta_{YZ}(y,\gamma,\gamma^\pi)$, we have:
$$z = (y, \gamma, \gamma^\pi);$$
- $\delta_{ZW} : Q_Z \times \Sigma \times \{0,1\} \to Q_W$ is the deterministic transition function from Z-states to W-states which satisfies the following constraint: for any $z = ((x,n),\imath',\gamma,\gamma^\pi) \in Q_Z$, $(\sigma,\imath) \in \Sigma \times \{0,1\}$, and $w \in Q_W$ such that $w = \delta_{ZW}(z,(\sigma,\imath))$:
  - if $\imath' = 0$ and $x \in X_V$, then we have
  $$[\sigma \in \gamma \cup \Sigma_a] \wedge [(\sigma,\imath) \in \gamma^\pi] \wedge [w = ((x,n),\sigma,\imath)];$$
  - otherwise, we have
  $$[\sigma \in \gamma] \wedge [(\sigma,\imath) \in \gamma^\pi] \wedge [w = ((x,n),\sigma,\imath)];$$
- $\delta_{WY} : Q_W \times \Sigma_L \to Q_Y$ is the deterministic transition function from W-states to Y-states which satisfies the following constraint: for any $w = ((x,n),\sigma,\imath) \in Q_W$, $(\sigma,\imath,\imath') \in \Sigma_L$, and $y \in Q_y$ such that $y = \delta_{WY}(w,(\sigma,\imath,\imath'))$, we have
$$y = (f((x,n),(\sigma,\imath,\imath')),\imath');$$
- Γ is the set of admissible decisions of the supervisor;
- Γ^π is the set of admissible decisions of the policy;
- $y_0 = (q_0, 0)$ is the initial Y-states.

In the unfolded SLA L, a Y-state y is constructed by expanding a state $q \in Q$ of L with a binary variable $\imath' \in \{0,1\}$ representing whether or not q is currently defended. Starting from Y-states, the supervisor S and the safety-enhancement policy π make decisions $(\gamma,\gamma^\pi) \in \Gamma \times \Gamma^\pi$ and U moves to Z-states by remembering this decision. Then at this Z-state, according to the defense status of the current state, all possible events $(\sigma,\imath) \in \Sigma \times \{0,1\}$ allowed by both γ and γ^π are took into consideration and the U moves to W-states by remembering each $(\sigma,\imath)$. Finally at W-states, two events $(\sigma,\imath,\imath') \in \Sigma_L$ with $\imath' \in \{0,1\}$ are considered and U moves back to Y-states by following the dynamic of the SLA and remembering the state-defense decision $\imath'$. Intuitively, the unfolded SLA can be treated as a game arena of the supervisor, safety-enhancement policy (together operate on Y-states), the attacker (operates on Z-states), and state-defense policy (operates on W-states).

Given an unfolded SLA U, we define the set of decisions at Y-state $y \in Q_Y$ as $\Gamma_U(y) := \{(\gamma,\gamma^\pi) \in \Gamma \times \Gamma^\pi : \delta_{YZ}(y,(\gamma,\gamma^\pi))!\}$. We also define the set of decisions at W-state $w = (q,\sigma,\imath) \in Q_W$ as $D_U(w) := \{\imath' \in \{0,1\} : \delta_{WY}(w,(\sigma,\imath,\imath'))!\}$. For the purpose of control, we require the supervisor, the safety-enhancement, and the state-defense policies to always react to any events that occur. Formally, given an unfolded SLA U, we say

- a Y-state $y \in Q_Y$ is *complete* if $\Gamma_U(y) \neq \emptyset$;
- a Z-state $z = (((x,n),\imath'),\gamma,\gamma^\pi) \in Q_Z$ is *complete* if
  - $\delta(x,\sigma)! \Rightarrow \delta_{ZW}(z,(\sigma,\imath))!$ for any $\sigma \in \gamma \cup \Sigma_a$, $(\sigma,\imath) \in \gamma^\pi$, when $\imath' = 0$ and $x \in X_V$;
  - $\delta(x,\sigma)! \Rightarrow \delta_{ZW}(z,(\sigma,\imath))!$ for any $\sigma \in \gamma$, $(\sigma,\imath) \in \gamma^\pi$, otherwise.

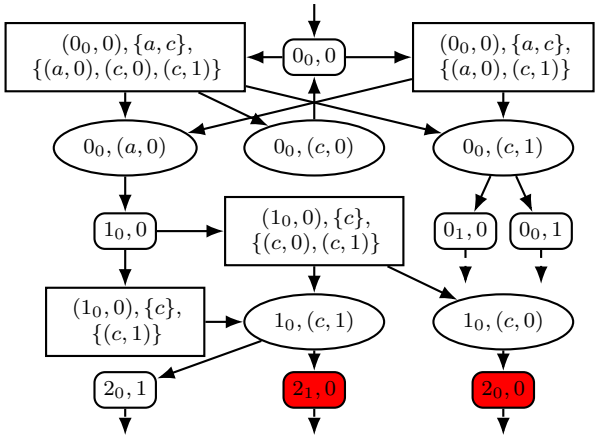


Figure 3: Partial of the unfolded SLA U w.r.t. L in Figure 2(b).

- a W-state $w=((x,n),\sigma,\imath)\in Q_W$ is *complete* if $D_U(w)\neq\emptyset$.

We say U is complete if all states in it are complete.

Note that the unfolded SLA contains multiple policy patterns in general. Given a policy pattern (S,π,D), for any $s=\sigma_1\sigma_2\cdots\in\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)$, the corresponding state in U is the state reached recursively by applying the corresponding decisions along string s, which we denote as $Y(s)$, $Z(s)$ and $W(s)$, respectively.

Definition 5 (Included Policy Pattern)**.** We say a policy pattern (S,π,D) is *included* in unfolded SLA U if for any possible attacker A, for any $s\in\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)$, we have (i) $(S(s),\Gamma^{\pi}(s))\in\Gamma_U(Y(s))$; and (ii) $D(s)\in D_U(W(s))$.

Let "$\sqsubseteq$" denote the standard sub-graph inclusion, the following theorem shows that Problem 1 can be correctly solved by using unfolded SLA and sub-automaton H.

Theorem 1. For any policy pattern (S,π,D), we have $\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)\subseteq K$ for any possible A, if and only if, (S,π,D) is included in a complete unfolded SLA $U^*\sqsubseteq U$ such that:

(1) for any $y=((x,n),\imath')\in Q^*_Y$, we have $x\in X_H$;

(2) for any complete unfolded SLA $U'\sqsubseteq U$ satisfies condition (1), we have $U'\sqsubseteq U^*$.

Proof. The "if" part directly follows property (1) of U^*. Now we prove the "only if" part by contradiction. Suppose that there exists a combination (S,π,D) with $\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)\subseteq K$ for any possible A, but (S,π,D) is not included in U^*. However, we know that (S,π,D) is included in another complete unfolded SLA U which can be constructed by applying $S(s)$, $\Gamma^{\pi}(s)$ and $D(s)$ repeatedly under the strings $s\in\mathcal{L}(S^{\pi,D}_{A,\oplus}/G)$ for any possible A. Since (S,π,D) is not in U^*, we have that $U\cup UA^*$ is strictly larger than U^* and satisfies condition (1). It contradicts the fact that U^* is the largest structure that satisfies (1). □

To construct U^*, one can (i) construct the largest unfolded SLA U that enumerates all the feasible transitions δ_{YZ}, δ_{ZW}, and δ_{WY}; and (ii) delete all Y-states $y=((x,n),\imath')\in Q_Y$ for which $x\notin X_H$; (iii) Finally, delete all the incomplete states recursively until the structure converges. A similar procedure is used in [19], and readers are referred to this work for further details.

Given U^*, in execution, the combination (S,π,D) can work as follows: at each instant, S and π will remember the current Y-state y and randomly pick a decision pair (γ,γ^{π}) from $\Gamma_{UA^*}(y)$. Then we update the current state to Z-state according to (γ,γ^{π}) and wait for the next event to occur. After that, we update the current state to W-state w, and D will randomly pick a decision from $D_{UA^*}(w)$ and so forth. Since the policy pattern operates over the state space of the unfolded SLA, which is itself finite. We can always synthesize a finite-state supervisor by always picking the same decision on the same Y-state and W-state.

Example 3. We still consider the system G as shown in Figure 2(a). We show a partial of the unfolded SLA in Figure 3, where state (x,n) in L are represented by x_n. We note that some decisions in Z-states may contain redundancy, e.g., decision $\{c\}$ at state $0_0,0$, since event a can be enabled when state 0 is not defended. We omit these redundant decisions in each Z-state. In system G, state 3 is not included by X_H. Thus in the unfolded SLA, we need to first remove all Y-states $((3,n),\imath')\in Q_Y$. This will finally cause states $(2_1,0)$ and $(2_0,0)$ incomplete since state 2 in G must be defended to disable event a. We also remove them from the unfolded SLA. Then W-state $1_0,(c,0)$ becomes a new incomplete state and also be deleted from the structure. Such a procedure converges to a single feasible decision at state $1_0,0$, i.e., $\{c\}$ and $\{c,1\}$. hence, event c must be protected when the state is 1 and the current safety level is 0. Then followed by the occurrence of c, state 2 must be defended.

Note that the SLA contains at most $|X| \cdot (\ell_m + 1)$ states, and the state space of the unfolded SLA U is polynomial in $|X|$ and ℓ_m, but exponential in $|\Sigma|$. Pruning unsafe Y-states is linear in the size of U, while removing incomplete states is quadratic. Thus, the overall complexity is polynomial in both the state space size and the safety level bound, but exponential in the size of the event set.

5 Conclusion

In this paper, we introduce the attack-resilient supervisory control problem under dynamic-event-protection mechanisms. To solve this problem, we use a multilayered duplication structure of the original system and show that the problem can be formulated as a supervisory control problem on this structure with an integrated decision architecture. In the future, we plan to investigate the trade-off between system permissiveness and protection frequency through a Pareto-optimal synthesis framework. Also, we plan to extend our results to the partial observation setting.

References

[1] Lilian Kawakami Carvalho, Yi-Chin Wu, Raymond Kwong, and Stéphane Lafortune. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, 97:121–133, 2018.

[2] Christos G Cassandras and Stéphane Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[3] Chao Gao, Carla Seatzu, Zhiwu Li, and Alessandro Giua. Multiple attacks detection on discrete event systems. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 2352–2357, 2019.

[4] Ahmed Khoumsi. Sensor and actuator attacks of cyber-physical systems: A study based on supervisory control of discrete event systems. In *International Conference on Systems and Control*, pages 176–182, 2019.

[5] Liyong Lin, Sander Thuijsman, Yuting Zhu, Simon Ware, Rong Su, and Michel Reniers. Synthesis of supremal successful normal actuator attackers on normal supervisors. In *American Control Conference*, pages 5614–5619, 2019.

[6] Liyong Lin, Yuting Zhu, and Rong Su. Towards bounded synthesis of resilient supervisors. In *58th IEEE Conference on Decision and Control*, pages 7659–7664, 2019.

[7] Ziyue Ma and Kai Cai. Optimal secret protections in discrete-event systems. *IEEE Trans. Automatic Control*, 67(6):2816–2828, 2021.

[8] Ziyue Ma and Kai Cai. Secret protection in discrete-event systems with generalized confidentiality requirements. *IEEE Trans. Automatic Control*, 2024.

[9] Shoma Matsui and Kai Cai. Usability aware secret protection with minimum cost. *Nonlinear Analysis: Hybrid Systems*, 43:101111, 2021.

[10] Rômulo Meira-Góes, Eunsuk Kang, Raymond H Kwong, and Stéphane Lafortune. Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems. *Automatica*, 121:109172, 2020.

[11] Rong Su. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, 94:35–44, 2018.

[12] Rong Su. On decidability of existence of nonblocking supervisors resilient to smart sensor attacks. *Automatica*, 154:111076, 2023.

[13] Ruochen Tai, Liyong Lin, and Rong Su. On decidability of existence of resilient supervisors against covert sensor-actuator attacks. In *63rd IEEE Conference on Decision and Control*, pages 4462–4467, 2024.

[14] David Thorsley and Demosthenis Teneketzis. Intrusion detection in controlled discrete event systems. In *45th IEEE Conference on Decision and Control*, pages 6047–6054, 2006.

[15] Yin Tong, Kai Cai, and Carla Seatzu. Actuator attack mitigation using the detection-protection mechanism. *IFAC-PapersOnLine*, 58(1):180–185, 2024.

[16] Yu Wang, Alper Kamil Bozkurt, Nathan Smith, and Miroslav Pajic. Attack-resilient supervisory control of discrete-event systems: a finite-state transducer approach. *IEEE Open Journal of Control Systems*, 2023.

[17] Gang Xie, Yin Tong, Xiaomin Wang, and Carla Seatzu. Resilient supervisor synthesis for labeled petri nets against sensor attacks. *IEEE Trans. Automatic Control*, 2025.

[18] Jingshi Yao, Shaoyuan Li, Xunyuan Yin, and Xiang Yin. Attack-resilient supervisory control under energy-bounded attacks. *IFAC-PapersOnLine*, 56(2):9624–9629, 2023.

[19] Xiang Yin and Stéphane Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 61(8):2140–2154, 2016.