This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

Optimal Secret Protections in Discrete-Event Systems

Ziyue Ma, *Member, IEEE*, Kai Cai, *Senior Member, IEEE*

***Abstract*—In this paper we study a security problem of protecting secrets in discrete-event systems modeled by deterministic finite automata. In the system some states are defined as *secrets*, each of which is associated with a *security level*. The problem is to design an event-protecting policy such that any event sequence from the initial state that reaches a secret state contains a number of protected events no less than the required level of security. To solve this secret securing problem, we first develop a layered structure called the *security automaton*. Then we show that the problem is transformed to a supervisory control problem in the security automaton. We consider two criteria of optimality on protecting policies: (1) disruptiveness, i.e., protecting policies with a minimum degree of disturbance to legal users' normal operations; (2) cost, i.e., protecting policies with a minimal cost. For the optimality on disruptiveness, we prove that a minimally disruptive protecting policy is obtained by using the classical supervisory control theory in the security automaton. For the optimality on cost, we develop a method to obtain a protecting policy with minimal cost by finding a *min-cut* in the security automaton.**

***Index Terms*—Secret protection, security, discrete-event systems, cyber-physical systems, automata**

I. Introduction

Security issues of *cyber-physical systems* have drawn much attention in recent years [1], [2], [3], [4]. A system in reality usually contains some *secrets* that are not expected to be exposed or hacked by unauthorized external intruders. To protect the secrets, a system must be meticulously designed to guarantee that any user who wishes to access some crucial states to obtain these secrets must go through certain security checking steps — such as password check and SMS code verification. For example, a user of a mobile phone must pass a two-step verification to prove his/her identity, before getting access to some sensitive information such as the numbers of credit cards. In other words, an operational sequence that allows a user to reach a secret state must contain a number of *protected events* for which a user must perform an identity verification to pass the security check. In such a case, an unauthorized intruder, who wants to infiltrate the system but cannot prove his/her identity, must pay some efforts to hack through those protected event, e.g., hacking the password or forging identity tokens. If the effort of hacking these protected events is high enough, then an attack towards the secrets can be considered practically preventable.

Theoretically, a designer of a system may protect as many events as possible to prevent potential intruders. However, it is practically infeasible to protect too many events for two reasons. First, protecting an event usually incurs a cost due to the deployment of new verification protocols or purchasing expensive biometric detectors. Second, protecting too many events may degrade the experience of legal users. For example, requesting password verification for each click may be able to prevent any intruders, but will surely annoy a legal user. Hence, a system designer should carefully decide which events at which situation should be protected.

In the literature, a closely related notion in cyber-security, called *opacity*, is also extensively studied in discrete-event systems (e.g., see [4], [5], [6], [7], [8]). Opacity is a property such that an intruder cannot infer a given set of secrets by observing the behavior of a system. We point out that the problem of opacity (and its enforcement) is different from the problem of secret protection studied in this paper. In brief, opacity assumes that an intruder passively observes some events and infers some secrets based on the observation. In our problem, however, we consider the situation where an intruder can disguise as a legal user to access the system, which may not be recognized by an administrator. Moreover, opacity requires that each secret trace has the same projection as some non-secret traces. Hence, for a non-opaque system, one may modify the dynamics and/or the observation structure [9], [10], [11], [12], [13] to ensure the secret not leaked. In our case, since we require that the system be usable to legal users, the dynamics and the output of a system are not allowed to be modified. For example, we cannot disable an event in the system using supervisory control [14], [15], [16], [17], since legal users may still need the event. Instead of blocking or altering the events, we set protective measures on the operational routines to increase the difficulty of unauthorized access to the secrets, which may practically prevent potential intruders.

Another related notion is *intrusion detection* [18], [19], [20], [21]. There the aim of the administrator of a system is to detect the invasion of an intruder by detecting the abnormal behavior in the system: by doing so an alarm can be issued before some bad result (e.g., the system reaches a critical state) occurs. By contrast, in this paper we do not try to detect the existence of intruders, because an intruder may disguise as a legal user such that no abnormal behavior can be observed. Instead, we set protective barriers that prevent a potential intruder from

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61703321, Shaanxi Provincial Natural Science Foundation under Grant No. 2019JQ-022, the Fundamental Research Funds for the Central Universities under Grant JB210413, and JSPS KAKENHI Grant no. 21H04875.

Z. Ma is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (Email: maziyue@xidian.edu.cn).

K. Cai is with the Department of Electrical and Information Engineering, Osaka City University, Osaka 558-8585, Japan (Email: kai.cai@eng.osaka-cu.ac.jp).

accessing the secret states.

In this paper we study the secret protecting problem based on the above motivations. We consider systems (a.k.a., plants) modeled by *deterministic finite automata* that are fully observable (i.e., no *a priori* knowledge of which events are or are not observable by the intruder). In a plant automaton, some states are considered as *secrets*, and each of them is associated with a *security level* meaning the minimal number of security checks required before reaching the secret. In the plant, some events are *protectable*, i.e., on which we can set security checks. Each protectable event is associated with a nonnegative number indicating the cost of implementing the security check for this event. We say that a protectable event is *protected* if a check has been set on the event. Our aim is to design an event-protecting policy such that any event sequence from an initial state and reaching a secret state contains a number of protected events no less than the required security level. Moreover, we consider two criteria of optimality that evaluate protecting policies: (1) *disruptiveness*, which means the minimum degree of disturbance to the normal operations of legal users; (2) *cost*, which means to minimize the physical expenditure on implementing the protecting policy. The method developed in this work can be applied to the management and supervisory levels of automated systems to enforce safety requirements during the system design stage. The main contents of this paper are summarized as follows.

Given a plant modeled by an automaton in which some states are secrets with specified security levels, we first develop a new structure called the *security automaton* (SA) in which the information of protecting decisions are encoded. The SA of a plant has a multi-layered structure, in which each layer represents a security level of the plant. We prove that the secret protecting problem in a plant automaton can be transformed to a supervisory control problem in the SA. Precisely speaking, each supervisor for the transformed control problem in the SA corresponds to an event-protecting policy for the original security problem. This provides a necessary and sufficient condition for the existence of a protecting policy.

Second, we formally introduce the optimality criterion of disruptiveness. A protecting policy is minimally disruptive if it protects the minimal number of events for any sequences from the initial state to a secret state. We prove that the dual protecting policy of a maximally permissive supervisor in the SA is minimally disruptive. Hence, a minimally disruptive protecting policy is obtained by first computing the maximally permissive supervisor in the SA, followed by converting it to its corresponding protecting policy. Moreover, we also prove that the minimally disruptive protecting policy, if it exists, is unique.

Finally we consider the optimality criterion of cost. To obtain a protecting policy with the minimal cost, we transform the secret protecting problem to an *s-t min-cut problem* in the *s-t graph* that is augmented from the SA. Intuitively, an s-t min-cut may not necessarily be associated with an event-protecting policy whose cost is minimal, since an event in the plant may be associated with multiple arcs in the cut. However, we prove that this intuition is false: in fact, each protectable event in the plant may be associated with at most one arc in the min-cut. Therefore, an s-t min-cut of an s-t graph is always associated with an event-protecting policy in the original plant whose cost is minimal. Furthermore, we note that the complexity of all techniques developed in this paper is polynomial in the number of states and events of the plant.

The problem of secret protection with the minimal cost was also studied in [22], [23], [24]. The differences between [22], [23], [24] and this work are summarized as follows. (i) In this paper, we consider two criteria of optimality on protection policies, namely the minimal disruption and the minimal cost. In [22], [23], [24] only the minimal cost criterion is considered. (ii) For the minimal cost criterion, in the work of [22], [23], [24], the set of protectable events is partitioned into distinct *levels*, and the cost to protect an event in a higher level dominates the cost to protect all events in lower levels. The total cost is then *qualitatively* defined as the maximum of the levels of protected events, while the number of protected events is not taken into account. In this paper we define a cost on every transition, i.e., the cost to protect each transition can be an arbitrary nonnegative real value, and the total cost is *quantitatively* defined as the sum of the costs of all protected transitions. As such, these two setups are generally incomparable. (iii) In [22], [23], [24] protecting policies are assumed to be static, that is, an event is either protected or not protected in all situations, while in this paper we consider dynamic protecting policies, i.e., whether an event at a state is protected or not depends on the trajectory by which the state is reached.

The rest of this paper is organized as follows. Basic notions of automata are recalled in Section II. In Section III, the problem of secret protection is formulated, and the notion of security automaton is proposed. In Section IV, we prove a necessary and sufficient condition for the existence of a protecting policy, based on which we develop a method to compute a minimally disruptive protecting policy. In Section V, we develop an algorithm to compute a protecting policy with minimal cost using s-t min-cut in the augmented SA. Section VI draws our conclusions and states future directions of this work.

II. Preliminaries

A. Deterministic Finite Automaton

A *deterministic finite automaton* (*automaton* for short) is a four-tuple

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

where Q is a set of *states*; Σ is a set of *events*; $\delta : Q \times \Sigma \to Q$ is the partial transition function; and $q_0 \in Q$ is the initial state.

We use Σ^* to denote the *Kleene closure* of Σ, consisting of all finite sequences composed by the events in Σ (including the *empty sequence* ε). Given a sequence $s \in \Sigma^*$, $|s|$ denotes the *length* of s. The transition function δ is extended to $\delta^* : Q \times \Sigma^* \to Q$ by recursively defining $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, s\sigma) = \delta(\delta^*(q, s), \sigma)$, where $s \in \Sigma^*$ and $\sigma \in \Sigma$. The *language* of G, denoted by $L(G)$, is defined as $L(G) = \{s \in \Sigma^* \mid \delta^*(q_0, s) \in Q\}$.

We use $\Gamma_G(q) = \{\sigma \in \Sigma \mid \delta(q,\sigma) \text{ is defined}\}$ to denote the set of events that are *enabled* at state $q \in Q$, and we use $\Gamma_G(s) = \Gamma_G(\delta^*(q_0, s))$ to denote the set of events that are *enabled* after sequence s.

Given an automaton $G = (Q, \Sigma, \delta, q_0)$, the *accessible part* of G, denoted as $Ac(G)$, is the automaton $G' = (Q', \Sigma, \delta', q_0)$ obtained from G by removing all unreachable states and their corresponding transition relations. Precisely speaking, $Q' = \{q \in Q \mid (\exists s \in L(G))\ \delta^*(q_0, s) = q\}$, and δ' is the restriction of δ to $Q' \times \Sigma \to Q'$.

A sequence $\bar{s} \in \Sigma^*$ is a *prefix* of a sequence $s \in \Sigma^*$ if $s = \bar{s}s'$ where $s' \in \Sigma^*$. We use $\bar{s}_k$ (where $0 \leq k \leq |s|$) to denote the prefix of s of length k, i.e., $s = \bar{s}_k s'$ where $|\bar{s}_k| = k$ and $s' \in \Sigma^*$. The *prefix closure* of a language $L \subseteq \Sigma^*$ is the set $\overline{L} = \{s \in \Sigma^* \mid \exists s' \in \Sigma^*, ss' \in L\}$.

B. Supervisory Control in Discrete-Event Systems

Supervisory control theory of discrete-event systems [25], [26] was first proposed by Ramadge and Wonham [27]. For a plant automaton $G = (Q, \Sigma, \delta, q_0)$, the event set Σ is partitioned into two disjoint subsets $\Sigma = \Sigma_c \cup \Sigma_{uc}$ where Σ_c is the set of *controllable events* and Σ_{uc} is the set of *uncontrollable events*.

In [27], the control objective, called the *(language) specification*, is defined by a regular language $K \subseteq \Sigma^*$. A supervisor S that dynamically disables events of the plant such that the closed-loop language of S over G is restricted within K. Here we use S/G to denote the closed-loop system composed by the plant G under the supervision of S, and we use $L(S/G)$ to denote the language of S/G. A supervisor S runs in parallel with the plant and, when a plant generates a sequence $s \in L(G)$, make a control decision $\xi(s) \subseteq \Sigma_c$ that is to disable all controllable events not in $\xi(s)$. Note that a supervisor cannot disable any uncontrollable $\sigma \in \Sigma_{uc}$ in any case. A language K is said to be *controllable* with respect to $L(G)$ if:

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}.$$

If K is not controllable, a supervisor S enforces K can be obtained by computing the *supremal controllable sublanguage* [25] of $L(G)$ with respect to K, i.e.:

$$L(G)^{\uparrow K} = \bigcup \{H \subseteq L(G) \mid H \text{ is controllable to } L(G)\}.$$

by iterative manipulations on regular languages.

A *state specification* defines a set of forbidden states $Q_l \subseteq Q$ that requires that the plant does not reach any state in Q_l. A supervisor S that enforces Q_l can be similarly obtained by first converting the state specification Q_l into its equivalent language specification $K = \{s \in L(G) \mid \delta^*(q_0, s) \notin Q_l\}$ followed by computing the supremal controllable sublanguage of K.

III. Secret Protection and Problem Formulation

In this paper, a plant is modeled by an automaton $G = (Q, \Sigma, \delta, q_0)$; at a subset of states some secrets, such as credit numbers or crucial personal data, are stored. By reaching these states, secret information may be obtained. To protect the secrets from being accessed by unauthorized intruders, some events of the plant should be protected such that any user (legal or unauthorized) who accesses the plant from the entry of the system and wants to reach a secret state necessarily passes through a certain number of security checks. Precisely speaking, a *security requirement* is a function $\ell : Q \to \mathbb{N}$ that assigns each state a *security level*, i.e., $\ell(q)$ means that to reach state q from entry q_0, at least $\ell(q)$ protected events must be passed. All states with positive security levels are *secret states* (*secrets* for short), the set of which is denoted as $Q_S = \{q \in Q \mid \ell(q) > 0\}$. We assume that $\ell(q_0) = 0$, i.e., the initial state is not secret. The maximal security level of ℓ is denoted as l, i.e., $l = \max_{q \in Q} \ell(q)$.

We assume that in a plant $G = (Q, \Sigma, \delta, q_0)$ some events are *protectable*, which means that security checks such as password verification can be set onto the physical execution of these events. The set of events Σ is partitioned into the set of *protectable* events Σ_p and the *set of unprotectable events* Σ_{up}, i.e., $\Sigma = \Sigma_p \cup \Sigma_{up}$. Hence, to enforce a security requirement ℓ, we need to design an *event-protecting policy* ϑ (*protecting policy* for short) such that for each sequence $s \in L(G)$ generated, ϑ decides which protectable events should be protected following s. This notion is formally defined as follows.

Definition 3.1: [Protecting Policy] Given a plant $G = (Q, \Sigma, \delta, q_0)$, a *protecting policy* is a function $\vartheta : L(G) \times \Sigma_p \to \{0, 1\}$ such that after sequence $s \in L(G)$ is generated, the set of protected events following s is $\vartheta(s) \subseteq \Sigma_p$. ♢

Remark 1: In [23], protecting policies were considered to be static, that is, an event is either protected or not protected for all sequences $s \in L(G)$. Here, the protecting policies defined by Definition 3.1 is dynamic. It may happen that an event σ is protected when a user visits state q for the first time, and event σ is not protected for the second time of his/her visiting of q. Such a dynamic protecting policy may be more applicable in reality. For example, in many cases the first visit of some webpage with a high security level requires a password check. But re-visiting the same page does not require password again (within a certain period of time), since the identity of the user has been confirmed. ♢

Given a protecting policy ϑ, we define $\vartheta(s, \sigma) = 1$ (resp., $\vartheta(s, \sigma) = 0$) if $\sigma \in \vartheta(s)$ (resp., $\sigma \notin \vartheta(s)$). Given a security requirement $\ell : Q \to \mathbb{N}$, a protecting policy ϑ is *valid* if for any sequence $s \in L(G)$ that yields a state $q \in Q$, the number of events protected by ϑ in sequence s is no fewer than $\ell(q)$. This notion is formally stated by the following definition.

Definition 3.2: Given a plant $G = (Q, \Sigma, \delta, q_0)$ and a security requirement $\ell : Q \to \mathbb{N}$, a protecting policy ϑ is *valid* if for any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ such that $\delta^*(q_0, s) = q$, there holds:

$$\sum_{i=0}^{n-1} \vartheta(\bar{s}_i, \sigma_{i+1}) \geq \ell(q). \tag{1}$$

♢

Now we are ready to formalize the *secret protecting problem* (SPP) that will be studied in this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

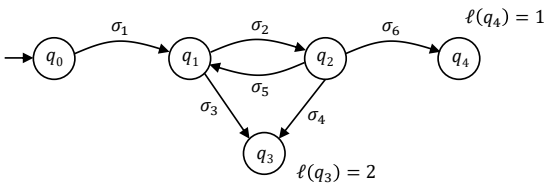


Fig. 1: A plant automaton.

Problem 1: [Secret Protecting Problem] Given a plant $G = (Q, \Sigma, \delta, Q_0)$ and a security requirement $\ell : Q \to \mathbb{N}$, determine a valid protecting policy ϑ.

Example 3.1: Consider the plant automaton shown in Figure 1 that models a computer networked system. A user who visits the system is first initialized at state q_0. The user can connect to the server via event σ_1 and then switches between states q_1 and q_2 via events σ_2 and σ_5. From state q_2 the user may reach state q_4 via event σ_6 to access the general personal information of this account, while from both states q_1 and q_2 the user may reach state q_3 via events σ_3 and σ_4, respectively, to access some crucial information of this account, e.g., credit card numbers.

Suppose that we want to enforce a security requirement ℓ such that the security level of state q_4 is 1 and that of q_3 is 2 (i.e., $\ell(q_4) = 1$ and $\ell(q_3) = 2$), while the security level of all other states is zero. In other words, to access the general personal information at least one security check is required, and to access the crucial information at least two security checks are required. Let us assume that events $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5$ are protectable, and event σ_6 is unprotectable.

To enforce such a security requirement, a valid protecting policy is the following: $\vartheta(\varepsilon) = \{\sigma_1\}$, $\vartheta(\sigma_1(\sigma_2\sigma_5)^n) = \{\sigma_3\}$ $(n \in \mathbb{N})$, $\vartheta(\sigma_1\sigma_2(\sigma_5\sigma_2)^n) = \{\sigma_4\}$ $(n \in \mathbb{N})$, and $\vartheta(s) = \emptyset$ for all other s's. $\diamondsuit$

At the remaining of this section we make the following two comments on Problem 1. First, one may have noticed that the physical basis of Problem 1 is similar to the notion of *state opacity*. However, although we know the risk that an intruder may access the secret states, to guarantee that the plant is usable to legal users, we cannot modify the dynamics of the plant nor disable any events in it (while in the opacity enforcement [14], [15], [16], [17] we can do so). For example, disabling events σ_3 and σ_4 will disallow legal users to access their data stored in q_3. Instead, we set protective measures to set barriers on the operational routines to prevent an intruder from accessing these secrets.

Second, in theory, we can protect all protectable events in the plant at any moment, by the following policy ϑ_{max}:

$$\vartheta_{max}(s) = \Sigma_p, \quad \forall s \in L(G).$$

It is not difficult to understand that if ϑ_{max} is not valid then there exists no valid protecting policy. However, such ϑ_{max} is typically infeasible in practice. On one hand, protecting too many events — for example, requesting a password verification for every click on a website — greatly degrades the experience of legal users. On the other hand, protecting an event usually incurs a cost, e.g., to deploy new verification protocols or to buy expensive biometric detectors. As a result, in the sequel of this paper two criteria of *optimality* on protecting policies are considered:

1) **minimal disruption**: protecting policies should disturb legal users' operations as little as possible;
2) **minimal cost**: protecting policies should incur implementation cost as low as possible.

Note that these two criteria of optimality are incomparable: in general there does not exist a protecting policy that satisfies both. In the following sections, we will investigate Problem 1 and develop two methods to compute a valid protecting policy from the two different criteria of optimality, respectively.

IV. Security Automata and Minimally Disruptive Protecting Policies

A. Security Automaton

In this section, we first introduce a structure called the *security automaton* in which both the plant behavior and the information of protecting requirements are encoded.

Definition 4.1: [Security Automaton] Given a plant $G = (Q, \Sigma, \delta, q_0)$ where $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ and a security requirement $\ell : Q \to \mathbb{N}$ such that the maximal security level of states in Q is l, the *security automaton* (SA) of G (with respect to ℓ) is a deterministic finite automaton $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$ where:

- $Q_H = \{q_{i,h} \mid 0 \le i \le |Q|, 0 \le h \le l\}$ is a set of *states*;
- $\Sigma_H = \Sigma_\alpha \cup \Sigma_\lambda$ is a set of *events*, where $\Sigma_\alpha = \{\alpha_1, \ldots, \alpha_n\}$ and $\Sigma_\lambda = \{\lambda_1, \ldots, \lambda_n\}$ are two duplicates of set Σ;
- $q_{0,0}$ is the initial state;
- $\delta_H : Q_H \times \Sigma_H \to Q_H$ is the partial transition function such that for all $\delta(q_i, \sigma_j) = q_{i'}$, it holds:

$$\begin{cases} \delta_H(q_{i,h}, \alpha_j) = q_{i',h}, & \forall h = 0, \ldots, l; \\ \delta_H(q_{i,h}, \lambda_j) = q_{i',h+1}, & \forall h \le l-1; \end{cases}$$

$\diamondsuit$

In plain words, the SA of a plant G with ℓ is a multi-layered automaton whose layers are numbered $0, 1, \ldots, l$, each of which is a duplicate of G. Hence, H contains no more than $|Q| \cdot (l+1)$ states and $|Q| \cdot (2l+1) \cdot |\Sigma|$ arcs. A state $q_{i,h}$ in layer h of the SA means that the plant G is at state q_i and its current security level is h. In an SA, both event α_j in Σ_α and event λ_j in Σ_λ are duplicates of the same event σ_j in the original plant G. The interpretation of an event α_j is the "unprotected event σ_j", and the interpretation of an event λ_j is the "protected event σ_j". Note that the construction of an SA does not encode the protectability of events yet, i.e., an unprotectable event σ_j also has two duplicated events α_j and λ_j. Note also that not all states in Q_H are accessible from the initial state $q_{0,0}$. In the sequel, only the accessible part of the SA H, i.e., $Ac(H)$, is considered. We use the following example to illustrate this.

Example 4.1: Again consider the plant automaton in Figure 1 with $\ell(q_4) = 1, \ell(q_3) = 2$ and $\ell(q) = 0$ for $q \neq q_3, q_4$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

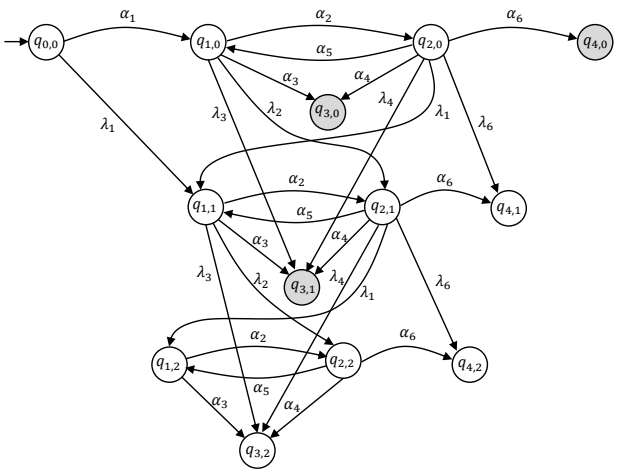


Fig. 2: Security automaton of the plant in Figure 1 with $\ell(q_0) = \ell(q_1) = \ell(q_2) = 0$, $\ell(q_3) = 2, \ell(q_4) = 1$.

The corresponding security automaton H is shown in Figure 2 (inaccessible states $q_{0,1}$ and $q_{0,2}$ are removed). In the SA H, transition $\delta_H(q_{0,0}, \alpha_1) = q_{1,0}$ means that if the plant is at state q_0, the current security level is 0, and event σ_1 is not protected, then by executing σ_1 the plant moves to state q_1 while the current security level remains 0 (i.e., by executing α_1 the SA reaches $q_{1,0}$). Transition $\delta_H(q_{0,0}, \lambda_1) = q_{1,1}$ means that if σ_1 is protected, by executing σ_1 the plant moves to state q_1 while the current security level increases from 0 to 1 (i.e., by executing λ_1 the SA reaches $q_{1,1}$). $\diamondsuit$

From Example 4.1 one can see that an SA contains the information of both the current state of the corresponding plant and the current security level. Moreover, an SA can be used to simulate the evolution of a plant with a protecting policy ϑ. In plain words, suppose that an SA is at state $q_{i,h}$, which means that the plant is at state q_i while the current security level is h, and suppose that the plant executes the next event σ_j with $\delta(q_i, \sigma_j) = q_{i'}$:

- if σ_j is protected, then the SA executes event λ_i to reach $q_{i',h+1}$, i.e., the plant reaches $q_{i'}$ while the security level is increased by 1;
- if σ_j is not protected, then the SA executes event α_i to reach $q_{i',h}$, i.e., the plant reaches $q_{i'}$ while the security level does not increase.

The following definition associates each sequence of events in G with a *protected sequence* in H with respect to a protecting policy ϑ.

Definition 4.2: Given a plant $G = (Q, \Sigma, \delta, q_0)$ with security requirement ℓ and a sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$, the *protected sequence* of s with respect to a protecting policy ϑ is $s' = \sigma'_1 \cdots \sigma'_n \in (\Sigma_\alpha \cup \Sigma_\lambda)^*$ such that:

$$\sigma'_i = \begin{cases} \alpha_i & \text{if } \sigma_i \notin \vartheta(\bar{s}_{i-1}) \\ \lambda_i & \text{if } \sigma_i \in \vartheta(\bar{s}_{i-1}). \end{cases}$$

This is denoted as $\theta(s) := s'$.[1] If $\theta(s) = s'$, s is said to be the *original sequence* of s', which is denoted as $\theta^{-1}(s') := s$. $\diamondsuit$

The following proposition shows that SA can be used to simulate G under any protecting policies.

Proposition 4.1: Given $G = (Q, \Sigma, \delta, q_0)$ and its SA $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$, for any protecting policy ϑ and sequence $s \in L(G)$, it holds that $\theta(s) \in L(H)$ and $\delta^*_H(q_{0,0}, \theta(s)) = q_{i,j}$ where i is the subscript of $q_i = \delta^*(q_0, s)$ and $j = \sum_{r=0}^{|s|} \vartheta(\bar{s}_r, \sigma_{r+1})$.

Proof: We prove this proposition by induction. First, for $|s| = 0$ (i.e., $s = \varepsilon$), the statement holds.

Now, suppose that for all $s = \sigma_1 \cdots \sigma_k$ (i.e., $|s| = k$), the statement holds. Consider an arbitrary sequence $s\sigma_{k+1}$ with $\sigma_{k+1} \in \Sigma$ and $\delta^*(q_0, s) = q_i$, $\delta(q_i, \sigma_{k+1}) = q_{i'}$. By the assumption above, it holds that $\delta^*_H(q_{0,0}, \theta(s)) = q_{i,j} \in Q_H$. By the definition of the SA, at state $q_{i,j}$, event α_{k+1} is defined such that $\delta_H(q_{i,j}, \alpha_{k+1}) = q_{i',j}$, and event λ_{k+1} is defined if $j < \max_{q \in Q} \ell(q)$ such that $\delta_H(q_{i,j}, \lambda_{k+1}) = q_{i',j+1}$. Hence, if $\sigma_{k+1} \in \vartheta(s)$ (which implies that $j < \max_{q \in Q} \ell(q)$), then at state $q_{i,j}$ the SA H can execute λ_{k+1} to reach state $q_{i',j+1}$, otherwise it executes α_{k+1} to reach state $q_{i',j}$. This indicates $\delta^*_H(q_{0,0}, \theta(s\sigma_{k+1})) = q_{i,j} \in Q_H$, $j = \sum_{r=0}^{k+1} \vartheta(\bar{s}_r, \sigma_{r+1})$, which concludes the proof. $\square$

In the following, we recall the supervisory control problem in discrete-event systems with state specifications, and then we show that Problem 1 can be reduced to a supervisory control problem in the corresponding SA.

Problem 2: [Supervisory Control with State Specification] Given a plant $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_c \cup \Sigma_{uc}$ where Σ_c denotes the set of controllable events and Σ_{uc} denotes the set of uncontrollable events, and given a state specification $Q_l \subseteq Q$ defining a set of forbidden states, determine a supervisor $\xi : L(G) \to 2^{\Sigma_c}$ such that G does not reach any state in Q_l when controlled by ξ.

It has been known that the supervisor ξ (if exists) can be easily obtained from G by removing all states in $Q \setminus Q_l$ and all states co-reachable to $Q \setminus Q_l$ via uncontrollable sequences in Σ^*_{uc}, resulting in a new automaton G_ξ that is a subautomaton of G [25]. Moreover, such a supervisor is *maximally permissive*. The control action for a sequence s is given by

$$\xi(s) = \Gamma_{G_\xi}(q) \cap \Sigma_c, \text{ where } q = \delta'(q'_0, s),$$

i.e., all controllable events that are not defined after executing s are disabled. Now, given a secret protection problem (i.e., SPP in Problem 1), we define its corresponding *supervisory control problem in the corresponding SA* (SCP-SA) as follows.

Problem 3: [SCP-SA] Given a plant $G = (Q, \Sigma, \delta, q_0)$ with security requirement ℓ, let $H = (Q_H, \Sigma_\alpha \cup \Sigma_\lambda, \delta_H, q_{0,0})$ be the corresponding SA. Design a supervisor in automaton H with respect to

$$\begin{cases} \Sigma_c = \{\alpha_i \in \Sigma_\alpha \mid \sigma_i \in \Sigma_p\} \\ \Sigma_{uc} = \Sigma_\lambda \cup \{\alpha_i \in \Sigma_\alpha \mid \sigma_i \in \Sigma_{up}\} \end{cases} \tag{2}$$

[1] Remind that $\vartheta : L(G) \times \Sigma_p \to \{0, 1\}$, i.e., $\vartheta(s)$ is the protecting decision after sequence s. On the other hand, θ is a *projection* from Σ^* to $(\Sigma_\alpha \cup \Sigma_\lambda)^*$ depending on ϑ, and $\theta(s)$ denotes the protected sequence of s according to ϑ.

and a state specification $Q_\ell = \{q_{i,h} \mid h < \ell(q_i)\}$.

In short words, the corresponding SCP-SA of an SPP is a supervisory control problem in the corresponding SA such that: (i) the set of controllable events Σ_c consists of all events α_i in Σ_α whose associated events σ_i are protectable; (ii) the set of uncontrollable events Σ_{uc} consists of all events λ_i in Σ_λ and all α_i in Σ_α whose associated events σ_i are unprotectable; (iii) the set of forbidden states consists of all states $q_{i,h}$ whose security level h is lower than the required level $\ell(q_i)$. Now we introduce a notion of duality of the solution of an SPP and the solution of its corresponding SCP-SA. Such a duality property establishes the link between the solutions of the two problems.

Definition 4.3: Consider an SPP for plant G and the corresponding SCP-SA for the SA H. A control policy ξ of the SCP-SA and a protecting policy ϑ of the SPP are said to be *dual* if for each sequence $s \in L(G)$ and each $\sigma_i \in \Sigma_p$, the following condition holds:

$$\sigma_i \in \vartheta(s) \quad \Leftrightarrow \quad \alpha_i \in \Gamma_H(\theta(s)) \setminus \xi(\theta(s)) \tag{3}$$

$\diamondsuit$

The duality of protecting policy ϑ of an SPP and control policy ξ of the corresponding SCP-SA means that, if we run ϑ and ξ in parallel, the ϑ protects event σ_i after a sequence $s \in L(G)$ if and only if ξ disables event α_i after a sequence $\theta(s) \in L(H)$. Now we prove that the solutions of an SPP and the solutions of the corresponding SCP-SA are indeed dual.

Theorem 4.1: If ϑ is a solution of an SPP, then its dual supervisor ξ is a solution of the corresponding SCP-SA, and vice versa.

Proof: ($\Rightarrow$) Let ϑ be an arbitrary valid protecting policy that enforces ℓ. We prove that its dual supervisor ξ is a solution to the SCP-SA. The proof is by contradiction. Suppose that ξ permits a sequence $s' = \sigma'_1 \cdots \sigma'_n$ in H to reach a forbidden state $q_{i,h}$ with $h < \ell(q_i)$. Since the index of the layer increases by one at each time an event in Σ_α is disabled, the total number of disablement of events in Σ_α is h. Then, according to Definition 4.3, the total number of protected events σ_i in $s = \theta^{-1}(s')$ is also h, which indicates that $\sum_{i=1}^{n} \vartheta(\bar{s}_i, \sigma_{i+1}) = h < \ell(q)$ holds. This indicates that ϑ is not valid, which is a contradiction.

($\Leftarrow$) Let ξ be a supervisor of the SCP-SA problem, whose supervisor automaton G_ξ is a subautomaton of H. We prove that its dual protecting policy ϑ is a valid protecting policy that enforces ℓ. The proof is again by contradiction. Suppose that there exists a sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ such that Eq. (1) does not hold with respect to ϑ. It implies that in the SA by executing $s' = \theta(s)$ a state $q_{i,h}$ with $h < \ell(q_i)$ is reached. Since each execution of unprotected event σ_i corresponds to a control action that permits α_i, it indicates that ξ permits s' in H, i.e., the forbidden state $q_{i,h}$ is reachable under the control of ξ, which is a contradiction. $\square$

Theorem 4.1 shows that the solution of an SPP and the solution of the corresponding SCP-SA are dual. Then we immediately have the following corollary.

Corollary 4.1: An SPP has a solution if and only if the corresponding SCP-SA has a solution.

The reduction from an SPP to its corresponding SCP-SA is clearly polynomial (since for a given plant the corresponding SA can be constructed in polynomial time according to Definition 4.1), and the existence of solutions of SCP-SA can be verified using existing supervisory control methods. Therefore, existing methods [25] can be used to check the existence of protecting policies of SPP, by checking the emptiness of the supremal controllable sublanguage of the corresponding SCP-SA.

Theorem 4.2: An SPP admits a solution if and only if the supremal controllable sublanguage of the SA H with respect to $K = \{s \in L(H) \mid \delta^*_H(q_{0,0}, s) = q_{i,h}, h < \ell(q_i)\}$ is non-empty, where Σ_c and Σ_{uc} are defined by Eq. (2).

Proof: Directly from Theorem 4.1. $\square$

Before proceeding, we remark that although we reduce the SPP to the control problem SCP-SA, it does not mean that we will use the supervisor of SCP-SA to disable events in the plant to protect the secrets. Once a supervisor ξ (or its automaton form G_ξ) of the SCP-SA is obtained, ξ is converted to its dual protecting policy according to Definition 4.3 which does not disable any events in the plant to guarantee the usability for normal users.

Finally, we note that given a control policy ξ, its dual protecting policy ϑ will not necessarily be explicitly computed. Instead, for any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$, $\vartheta(s)$ can be computed stepwise by computing $\vartheta(\bar{s}_i), i = 0, 1, \ldots, |s|$, starting from the empty sequence ε. The reason is: (i) $\theta(\varepsilon) = \varepsilon$ according to Definition 4.2; (ii) by Eq. (3), $\vartheta(\bar{s}_k) = \Gamma_H(\theta(\bar{s}_k)) \setminus \xi(\theta(\bar{s}_k))$ holds. In other words, the protected events after $\bar{s}_k$ are those α-events disabled at state $q_{i,j} = \delta^*_H(q_{0,0}, \theta(\bar{s}_k))$ in the SA, which means that the protecting decision after $\bar{s}_k$, i.e., $\vartheta(\bar{s}_k)$, is computable. Therefore, for the next event σ_{k+1} after $\bar{s}_k$ in s, $\theta(\bar{s}_k\sigma_{k+1})$ can be computed according to Definition 4.2 (since $\theta(\bar{s}_k)$ and $\vartheta(\bar{s}_k)$ are both known). Thus, for any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$, all protecting decisions made by ϑ (i.e. $\vartheta(\varepsilon), \vartheta(\sigma_1), \vartheta(\sigma_1\sigma_2), \ldots, \vartheta(\sigma_1 \cdots \sigma_n)$) can be computed stepwise during the execution of s. An illustrative example can be found in the next subsection (Example 4.2).

B. Minimally Disruptive Protecting Policies

In this subsection, we solve Problem 1 for the convenience of legal users. In plain words, a protecting policy is *minimally disruptive* if it does not protect any events unless it has to. For example, when a user is using a cell phone, in general no security check is triggered when he/she uses the calculator and the clock apps. Security check is required only if he/she wants to inspect the credit cards and other private information of the account.

Definition 4.4: [Minimal Disruptiveness] A protecting policy ϑ is *minimally disruptive* if for any other protecting policy $\vartheta' \neq \vartheta$ and any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ (i.e., $|s| = n$), the following condition holds:

$$\sum_{i=0}^{n-1} \vartheta(\bar{s}_i, \sigma_{i+1}) \leq \sum_{i=0}^{n-1} \vartheta'(\bar{s}_i, \sigma_{i+1}).$$

$\diamondsuit$

In plain words, a protecting policy ϑ is minimally disruptive if it protects the minimal number of events for any sequence s. It is not difficult to understand that a minimally disruptive protecting policy exists if and only if there exists at least one valid protecting policy for Problem 1. We also remark that, interestingly, the minimally disruptive protecting policy is unique, which will be proved in the rest of this subsection.

By Theorem 4.1, for each supervisor in an SCP-SA, its dual protecting policy is a valid protecting policy of the original SPP. In this subsection we prove the first main result of this paper, that is: for the maximally permissive supervisor of an SCP-SA, its dual protecting policy is minimally disruptive.

Theorem 4.3: Given a plant $G = (Q, \Sigma, \delta, q_0)$ with security requirement ℓ, let $H = (Q_H, \Sigma_\alpha \cup \Sigma_\lambda, \delta_H, q_{0,0})$ be the corresponding SA and ξ_{max} be the maximally permissive supervisor of the SCP-SA. The protecting policy ϑ_{min} that is dual to ξ_{max} is minimally disruptive.

Proof: By contradiction, suppose that ϑ_{min} is not minimally disruptive. By Definition 4.4 there exists another valid protecting policy ϑ and a sequence $s = \sigma_1 \cdots \sigma_n$ such that $\vartheta(\bar{s}_k) = \vartheta_{min}(\bar{s}_k)$ holds for all $k \in \{1, \ldots, n-2\}$, and $\vartheta(\bar{s}_{n-1}) = \vartheta_{min}(\bar{s}_{n-1}) \setminus \{\sigma_n\}$ holds. In other words, the protecting decisions of ϑ_{min} and ϑ are the same for all events in s except the last one: ϑ_{min} protects the last event σ_n while ϑ does not protect σ_n. This implies that, in the SA, sequence $\theta(\bar{s}_{n-1})\alpha_n$ is permitted by the dual supervisor ξ of ϑ, which means that sequence $\theta(\bar{s}_{n-1})\alpha_n$ is legal in the SCP-SA. However, such a sequence is forbidden by ξ_{max}. This contradicts the fact that ξ_{max} is the maximally permissive supervisor of the SCP-SA. □

Corollary 4.2: The minimally disruptive protecting policy ϑ_{min}, if it exists, is unique.

Proof: Since ξ_{max} is dual to the minimal disruptive protecting policy ϑ_{min}, by the uniqueness of the maximally permissive supervisor ξ_{max} (if it exists), ϑ_{min} is unique. □

Algorithm 1 Determining a minimally disruptive protecting policy

Input: A plant $G = (Q, \Sigma, \delta, q_0)$ where $\Sigma = \Sigma_p \cup \Sigma_{up}$, a security requirement $\ell : Q \to \mathbb{N}$

Output: A minimally disruptive protecting policy $\vartheta : L(G) \to 2_p^\Sigma$

1: Compute the SA $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$ of G with respect to ℓ, where $\Sigma_H = \Sigma_\alpha \cup \Sigma_\lambda$.
2: Solve the SCP-SA in H for a maximally permissive supervisor ξ_{max}.
3: **if** ξ_{max} is empty, **then**
4: output "no solution" and exit.
5: **else**
6: compute the dual protecting policy ϑ_{min} of ξ_{max}.
7: **end if**
8: Output $\vartheta = \vartheta_{min}$ and exit.

By Theorems 4.2, and 4.3, we summarize the procedure of solving an SPP for a minimally disruptive protecting policy in Algorithm 1. In short words, given an SPP, we first construct its corresponding SA and solve the SCP-SA to obtain the maximally permissive supervisor ξ_{max}. If ξ_{max} is not empty, it is then converted to its dual protecting policy ϑ_{min} that is minimally disruptive by Theorem 4.3. If ξ_{max} is empty, then by Theorem 4.2 there does not exist a protecting policy for the security requirement. We use the following example to illustrate Algorithm 1.

Example 4.2: Let us consider the plant in Figure 1 with $\ell(q_3) = 2, \ell(q_4) = 1$ and its corresponding SA in Figure 2. Then the corresponding SCP-SA is to enforce a state specification

$$Q_\ell = \{q_{i,h} \mid h < \ell(q_i)\} = \{q_{3,0}, q_{3,1}, q_{4,0}\}$$

in the SA with $\Sigma_c = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, , \alpha_5\}$ and $\Sigma_{uc} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \alpha_6, \lambda_6\}$.

The maximally permissive supervisor ξ_{max} of the SCP-SA is depicted in Figure 3. By computing its dual protecting policy according to Definition 4.3, the minimally disruptive protecting policy ϑ_{min} is

$$\vartheta_{min}(s) = \{\sigma_i \in \Sigma_p \mid \alpha_i \notin \xi_{max}(\theta(s))\}.$$

For example, for a plant sequence $\sigma_1\sigma_2\sigma_5\sigma_2\sigma_4$ that reaches the secret state q_3 whose required security level is 2, the control decision made by ξ_{max} and the protecting decision made by ϑ_{min} for each step is summarized in Table I. In particular, the protecting decision of ϑ_{min} for each step is:

$$\begin{cases} \vartheta(\varepsilon) = \{\sigma_1\} \\ \vartheta(\sigma_1) = \{\sigma_3\} \\ \vartheta(\sigma_1\sigma_2) = \{\sigma_4\} \\ \vartheta(\sigma_1\sigma_2\sigma_5) = \{\sigma_3\} \\ \vartheta(\sigma_1\sigma_2\sigma_5\sigma_2) = \{\sigma_4\} \\ \vartheta(\sigma_1\sigma_2\sigma_5\sigma_2\sigma_4) = \emptyset \end{cases}$$

One can see that, after a user has reached state q_1 (by passing the first protected event σ_1 at q_0), no more security check is triggered when the user switches between states q_1 and q_2. The second security check is triggered only when the user is attempting to reach state q_3. ♢

V. Protecting Policies with Minimal Costs

In this section we consider the cost of implementing protecting policies. Our aim is to find a protecting policy with the minimal cost of implementation.

A. The Cost of Protecting Policies

In general, the costs incurred by the protection of the transitions are separate due to the physical implementation of the protection (e.g., by deploying a device or an encrypting procedure). Hence, we consider a *cost function* that assigns each transition a nonnegative real number or ∞:

$$\mathbf{c} : \Delta \to \mathbb{R}_{\geq 0} \cup \{\infty\} \tag{4}$$

where $\Delta = \{(q, e, q') \mid \delta(q, e) = q'\}$ denotes all transitions in a plant $G = (Q, \Sigma, \delta, Q_0)$. Assigning a transition with ∞ cost implies that it is practically unprotectable.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

s	$\delta^*(q_0, s)$	$\Gamma_G(s)$	$\theta(s)$	$\Gamma_H(\theta(s))$	$\xi_{max}(\theta(s))$	$\vartheta_{min}(s)$	Current Security Level
ε	q_0	σ_1	ε	λ_1	$\emptyset$	σ_1	0
σ_1	q_1	σ_2, σ_3	λ_1	$\alpha_2, \lambda_2, \alpha_3, \lambda_3$	α_2	σ_3	1
$\sigma_1\sigma_2$	q_2	$\sigma_4, \sigma_5, \sigma_6$	$\lambda_1\alpha_2$	$\alpha_4, \lambda_4, \alpha_5, \lambda_5, \alpha_6, \lambda_6$	α_5	σ_4	1
$\sigma_1\sigma_2\sigma_1$	q_1	σ_2, σ_3	$\lambda_1\alpha_2\alpha_1$	$\alpha_2, \lambda_2, \alpha_3, \lambda_3$	α_2	σ_3	1
$\sigma_1\sigma_2\sigma_1\sigma_2$	q_2	$\sigma_4, \sigma_5, \sigma_6$	$\lambda_1\alpha_2\alpha_1\alpha_2$	$\alpha_4, \lambda_4, \alpha_5, \lambda_5, \alpha_6, \lambda_6$	α_5	σ_4	1
$\sigma_1\sigma_2\sigma_1\sigma_2\sigma_4$	q_3	$\emptyset$	$\lambda_1\alpha_2\alpha_1\alpha_2\lambda_4$	σ_1	$\emptyset$	$\emptyset$	2

TABLE I: Illustration for sequence $\sigma_1\sigma_2\sigma_1\sigma_2\sigma_4$ in Example 4.2.

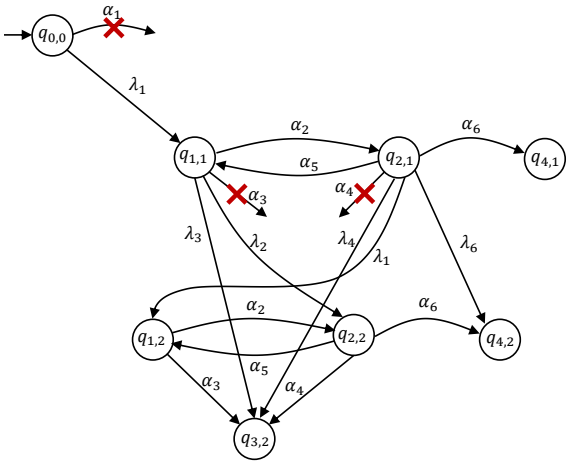


Fig. 3: The maximally permissive supervisor of the SA in Figure 2.

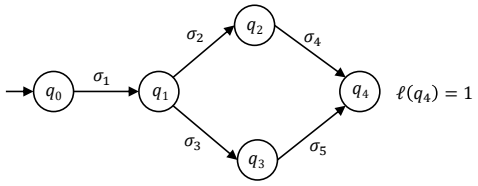


Fig. 4: Maximal permissive and non-maximally permissive protecting policies.

Since the cost to protect each transitions are separated, for mathematical convenience, in the sequel we assume that the protectable events in the plant are distinctly assigned to transitions. By doing so, the costs of different transitions can be differentiated by their labels. Note that the notion of "labels" considered here is different from that in nondeterministic automata (where a label describes how a transition *looks like* from the viewpoint of an observer). Thus, such an assumption is purely technical.

Assumption 1: All protectable events in the plant are distinctly assigned to transitions, i.e., for all $\sigma \in \Sigma_p$:

$$\delta(q, \sigma) \text{ is defined } \Rightarrow (\forall q' \neq q)\, \delta(q', \sigma) \text{ is not defined.}$$

$\diamondsuit$

By doing so, the cost function Eq. (4) can be equivalently defined as:

$$\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\} \tag{5}$$

that assigns each event a nonnegative cost. Then, the cost of a protecting policy is defined as the following.

Definition 5.1: Consider a plant $G = (Q, \Sigma, \delta, Q_0)$ that satisfies Assumption 1 and a cost function $\mathbf{c}$ defined by Eq. (5). For a protecting policy ϑ, the *set of protected events* of ϑ is

$$P(\vartheta) = \{\sigma \in \Sigma_p \mid (\exists s \in L(G))\sigma \in \vartheta(s)\}. \tag{6}$$

The *cost* of the protecting policy ϑ is:

$$C(\vartheta) = \sum_{\sigma \in P(\vartheta)} \mathbf{c}(\sigma). \tag{7}$$

$\diamondsuit$

In plain words, the cost $\mathbf{c}(\sigma)$ of event σ is counted (as part of the cost of ϑ) if σ is protected for at least one sequence $s \in L(G)$. Note that such a cost $\mathbf{c}(\sigma)$ is irrelevant to the number (when it is not zero) of sequences s in Eq. (6) in which σ is protected. This setting may be useful in practice: for example, to protect a transition one may purchase a biometric detector once, while the cost of each subsequent usage of the detector is comparatively negligible.

Based on Definition 5.1, the criterion of optimality of cost we consider in this section is defined as the following.

Definition 5.2: Given an SPP and a cost function $\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, a protecting policy ϑ is of the *minimal cost* if for any other valid protecting policy $\vartheta' \neq \vartheta$, $C(\vartheta) \leq C(\vartheta')$. $\diamondsuit$

Remark 2: We note that the definitions of the cost function and the minimal cost in [22], [23] are different from those in this section. In [22], [23] the set of protectable events are partitioned into distinct *levels*, i.e., Σ_p is partitioned into n levels $\Sigma_p = \Sigma_{p,1} \cup \cdots \cup \Sigma_{p,n}$. The cost to protect an event in level i dominates the cost to protect all events in lower levels $j < i$. Moreover, the cost of a protecting policy is defined as the maximal level of protected events, while the number of protected events/transitions is not taken into account. In other words, the objective on cost in [22], [23] is to minimize the maximal index i such that $P(\vartheta) \cap \Sigma_{p,i} \neq \emptyset$. Since such notions are incomparable to the notions of the cost function and the cost of a protecting policy in Definition 5.1 here, the notion of the minimal cost in this paper is also different from that in [22], [23]. $\diamondsuit$

The following example shows that a minimally disruptive protecting policy we have obtained so far does not guarantee the minimal cost.

Example 5.1: Consider the automaton in Figure 4 where all events are protectable and have a uniform cost for protection, i.e., $\mathbf{c}(\sigma) = 1$ for all $\sigma \in \Sigma$. Suppose that the security level of state q_4 is $\ell(q_4) = 1$ while other states are not secret. By computing the maximally permissive supervisor for the SCP-SA and the corresponding dual protecting policy, we obtain ϑ_{min} that protects events σ_4 and σ_5 when the plant state is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

q_2 and q_3, respectively. To implement ϑ_{min}, we need to pay 2 unit price (e.g., to buy devices) for protecting events σ_4 and σ_5. However, one can readily verify that it is sufficient to protect a single event σ_1 at state q_0 (although such a policy is not minimally disruptive) that requires only 1 unit price. $\diamondsuit$

Example 5.1 shows that in general there does not exist a protecting policy both minimally disruptive and of the minimal cost. This fact also indicates that the method we developed in Section IV, which was based on the maximally permissive supervisor in the SA, cannot be used to find a protecting policy with the minimal cost. Hence in this section we develop a different method to solve the problem. The basic idea is to transform a security protecting problem into an *s-t min-cut problem* in the corresponding SA that can be solved with polynomial methods. The solution of the s-t min-cut problem is then converted to a protecting policy of the original SPP with the minimal cost.

B. s-t Minimal-Cut Problem

Definition 5.3: [s-t Min-Cut][28] Consider a weighted digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$ whose set of vertices and set of directed edges are $\mathcal{V}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, respectively. Function $\mathbf{w} : \mathcal{E} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ assigns each edge $d \in \mathcal{E}$ a nonnegative weight $w(d)$. Let v_s, v_t be two vertices in $\mathcal{V}$ which are called the *source vertex* and the *terminal vertex*, respectively. A set of edges $\mathcal{C} \subseteq \mathcal{E}$ is an *s-t cut* if the subgraph of $\mathcal{G}$ induced by $(\mathcal{V}, \mathcal{E} \setminus \mathcal{C})$ does not contain any directed path from vertex v_s to vertex v_t. $\mathcal{C}$ is an *s-t minimal-cut* (*s-t min-cut* for short) if for any other s-t cut $\mathcal{C}'$, $\sum_{d \in \mathcal{C}} w(d) \leq \sum_{d \in \mathcal{C}'} w(d)$ holds, i.e., the total weight of edges in $\mathcal{C}$ is minimal. $\diamondsuit$

Note that the s-t min-cut may not be unique, i.e., there may exist multiple cuts all of which are of the same minimal cost. Determining an s-t min-cut in a diagraph, which is called the *s-t min-cut problem*, has been well studied in the literature. To keep this paper focused, we do not introduce the details but briefly explain how it is solved. An s-t min-cut problem can be reduced to its dual *max-flow problem* using *Max-Flow Min-Cut Theorem*, and the latter can be solved by various polynomial-time algorithms, e.g., *Edmonds-Karp* [28] with complexity $O(|\mathcal{V}| \cdot |\mathcal{E}|^2)$, *push-relabel algorithm* [29] with complexity $O(|\mathcal{V}|^2 \cdot |\mathcal{E}|)$. Once a max-flow is obtained, a min-cut can be obtained by inspecting the corresponding residue graph. Now, given an SPP, we establish its corresponding s-t min-cut problem as follows.

Definition 5.4: Given an SPP and a cost function $\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, let $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$ be the corresponding SA. The *s-t graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$ of the SA is defined as follows:

1) let $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ be the underlying unweighted digraph of the SA, i.e.

$$\begin{cases} \mathcal{V}' = Q_H, \\ \mathcal{E}' = \{(q_{i,j}, q_{i',j'}) \mid (\sigma \in \Sigma_H)\, \delta_H(q_{i,j}, \sigma) = q_{i',j'}\}; \end{cases}$$

2) let $\mathcal{V} = \mathcal{V}' \cup \{q_{\infty,\infty}\}$;
3) let $\mathcal{E} = \mathcal{E}' \cup \{(q, q_{\infty,\infty}) \mid q \in Q_\ell\}$, where $Q_\ell = \{q_{i,h} \mid h < \ell(q_i)\}$ is the set of forbidden states in the SA;

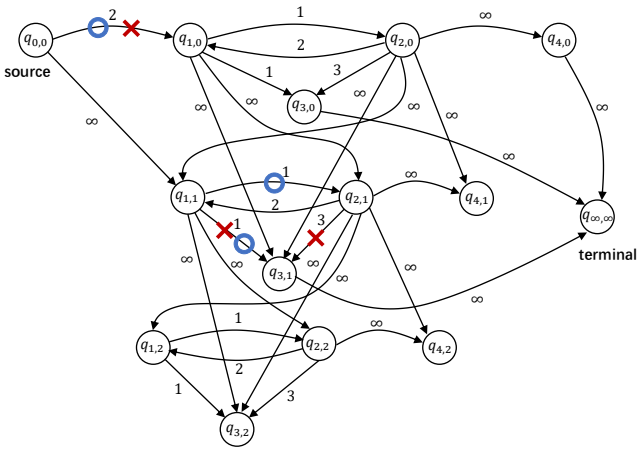


Fig. 5: The s-t graph $\mathcal{G}$ of the SA in Figure 2 with costs $2, 1, 1, 3, 1, \infty$, respectively for events $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$. The "blue circles" denote the arcs in the s-t min-cut $\mathcal{C}_{min}$ in Example 5.2, which are different from the "red crosses" denoting the disablement of events by supervisor ξ_{max} in Example 4.2.

4) the weight $\mathbf{w}(q_{i,j}, q_{i',j'})$ that is assigned to each arc $(q_{i,j}, q_{i',j'}) \in \mathcal{E}$ is defined as:

$$\mathbf{w}(q_{i,j}, q_{i',j'}) = \begin{cases} \mathbf{c}(\sigma_i), & \text{if } \delta_H(q_{i,j}, \alpha_i) = q_{i',j'} \\ \infty, & \text{if } \delta_H(q_{i,j}, \lambda_i) = q_{i',j'} \\ & \quad \vee q_{i',j'} = q_{\infty,\infty} \end{cases}$$

Problem 4: [MCP-SA] Given an SPP and a cost function $\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$, let H be the corresponding SA. The *s-t min-cut problem in the security automaton* (MCP-SA) is to determine an s-t min-cut in the corresponding s-t graph $\mathcal{G}$ with $q_{0,0}$ the source vertex and $q_{\infty,\infty}$ the terminal vertex.

We briefly explain how an s-t graph is obtained according to Definition 5.4. Given an SPP and a cost function $\mathbf{c}$, the corresponding SA is first computed and then converted into the underlying digraph. Then, a terminal vertex $q_{\infty,\infty}$ is added to the digraph, and for each vertex $q_{i,j}$ that corresponds to a forbidden state in Q_ℓ, an arc is added from $q_{i,j}$ to $q_{\infty,\infty}$. Finally, the weights on arcs are defined as: (1) $\mathbf{c}(\sigma_i)$, if the arc is labelled $\alpha_i \in \Sigma_\alpha$ in the SA, (2) ∞, if the arc is labelled $\lambda_i \in \Sigma_\lambda$ in the SA, or if the arc is from a forbidden state in Q_ℓ to $q_{\infty,\infty}$.

Once an s-t graph $\mathcal{G}$ is obtained, existing methods can be applied to find an s-t min-cut in $\mathcal{G}$ where $q_{0,0}$ is considered as the source vertex and $q_{\infty,\infty}$ is considered as the terminal vertex. We use the follow example to illustrate this.

Example 5.2: Again consider the SPP in Example 3.1 and the corresponding SA in Figure 2. Now suppose that the cost to protect each event is: $\mathbf{c}(\sigma_1) = 2, \mathbf{c}(\sigma_2) = 1, \mathbf{c}(\sigma_3) = 1, \mathbf{c}(\sigma_4) = 3, \mathbf{c}(\sigma_5) = 1$, and $\mathbf{c}(\sigma_6) = \infty$ (i.e., σ_6 is unprotectable). According to Definition 5.4, the corresponding s-t graph $\mathcal{G}$ is depicted in Figure 5.

Let $q_{0,0}$ and $q_{\infty,\infty}$ be the source and the terminal vertices, respectively. The s-t min-cut of the MCP-SA contains three

arcs :

$$\{(q_{0,0}, q_{1,0}), (q_{1,1}, q_{2,1}), (q_{1,1}, q_{3,1})\}$$

marked with the "blue circles" in Figure 5. The total weight of this min-cut is $2+1+1=4$. $\diamondsuit$

Since an SA contains at most $|Q| \cdot (l+1)$ states and $|Q| \times (2l+1) \cdot |\Sigma|$ arcs (where l is the maximal security level among all states), the corresponding s-t graph contains at most $|Q| \cdot (l+1)+1$ states and $|Q| \cdot (2l+1) \cdot |\Sigma| + |Q| \cdot l$ edges (the term $|Q| \cdot l$ comes from the arcs added from forbidden states to $q_{\infty,\infty}$). Hence, in the s-t graph, $O(\mathcal{V}) = O(|Q| \cdot l)$ and $O(\mathcal{E}) = O(|Q| \cdot l \cdot |\Sigma|)$. The total complexity to obtain an s-t min-cut (using *push-relabel* [29]) is $O(|\mathcal{V}|^2 \cdot |\mathcal{E}|) = O([|Q| \cdot l]^2 \cdot [|Q| \cdot l \cdot |\Sigma|]) = O(|Q|^3 \cdot l^3 \cdot |\Sigma|)$.

Remark 3: The maximally permissive supervisor obtained in Section IV is also associated with a cut in the SA: an arc in the SA belongs to the cut if the corresponding arc is disabled in the SA. However, the cost of such a cut is not always minimal. For example, supervisor ϑ_{max} in Example 4.2 corresponds to a cut that contains three arcs:

$$\{(q_{0,0}, q_{1,0}), (q_{1,1}, q_{3,1}), (q_{2,1}, q_{3,1})\}.$$

The total weight of this cut is $2+1+3=6$, and thus is not minimal. $\diamondsuit$

C. Converting a Min-Cut to a Protecting Policy with the Minimal Cost

Let us first introduce the duality of a protecting policy and an s-t cut in the s-t graph. Such a notion is analogous to the duality of protecting policies and supervisors in Definition 4.3.

Definition 5.5: Consider an SPP for plant G and the corresponding MCP-SA for the SA $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$. An s-t cut $\mathcal{C}$ of the SCP-SA and a protecting policy ϑ of the SPP are said to be *dual* if for each sequence $s \in L(G)$ and each $\sigma_i \in \Sigma_p$, the following condition holds:

$$\sigma_i \in \vartheta(s) \;\Leftrightarrow\; (\delta_H^*(q_{0,0}, \theta(s)), \delta_H^*(q_{0,0}, \theta(s)\sigma_i)) \in \mathcal{C} \qquad (8)$$

$\diamondsuit$

In plain words, the duality of protecting policy ϑ and cut $\mathcal{C}$ means that ϑ protects event σ_i after a sequence $s \in L(G)$ if and only if the corresponding arc $(\delta_H^*(q_{0,0}, \theta(s)), \delta_H^*(q_{0,0}, \theta(s)\sigma_i))$ in the s-t graph belongs to the cut $\mathcal{C}$. Similar to the result in Section IV, a protecting policy is valid for an SPP if and only if its dual s-t cut is valid for the corresponding MCP-SA. Note that, since an s-t cut can be viewed as a control policy, its dual protecting policy can be computed analogously to the procedure described at the end of Section IV-A.

Theorem 5.1: If ϑ is a solution of an SPP, then its dual s-t cut $\mathcal{C}$ is a solution of the corresponding MCP-SA, and vice versa.

Proof: This theorem can be proved analogously to Theorem 4.1, since an s-t graph with an s-t cut can be transformed to its dual protecting policy by Definition 5.5 and then to the dual supervisor in the SA. $\square$

Now, one may have realized that, although Theorem 5.1 guarantees that each s-t cut in the s-t graph is always associated with a valid protecting policy, that a cut is minimal in the s-t graph does not immediately imply that its dual protecting policy is of the minimal cost by Definition 5.2. Since each plant event σ may be associated with more than one arcs in the s-t graph, one may intuitively think that the cost of a protecting policy given by Definition 5.1 could be less than the cost of its dual s-t min-cut in the s-t graph, i.e., in equality "$C(\vartheta) \leq \sum_{d \in \mathcal{C}} \mathbf{w}(d)$" the "$<$" may hold. For instance, suppose that in the s-t graph in Figure 5 we obtain a cut $\mathcal{C}$ that contains both arcs $(q_{1,0}, q_{3,0})$ and $(q_{1,1}, q_{3,1})$ — two arcs associated with the same event σ_2 of the original plant. If such a cut is coincidentally a min-cut, then $C(\vartheta) < \sum_{d \in \mathcal{C}} \mathbf{w}(d)$ holds: since in the min-cut problem the cost $\mathbf{c}(\sigma_2)$ is counted twice to cut the two arcs, while by Definition 5.1 in $C(\vartheta)$ the cost $\mathbf{c}(\sigma_2)$ is counted only once. If this is the case, even if $\mathcal{C}$ is an s-t min-cut, we may not conclude that the cost of the dual protecting policy of $\mathcal{C}$ is minimal.

However, in the next subsection, we prove a key result that shows that *the above intuition is false*. Precisely speaking, given an SPP, if $\mathcal{C}_{min}$ is a min-cut of the corresponding MCP-SA, then $C(\vartheta) = \sum_{d \in \mathcal{C}} \mathbf{w}(d)$ always holds, and hence its dual protecting policy, denoted as ϑ_{minC}, is necessarily a protecting policy of the original SPP whose cost is minimal.

Remark 4: It is worth noting that a similar idea of using s-t min-cut in discrete-event systems was dated back to 1995 by Kumar and Garg [30]. The work of [30] studies the optimal supervisory control problem in automata, that is, to achieve the control goal by disabling some events with costs, and a supervisor was designed by determining a min-cut in a graph induced from a plant automaton. In our approach, we use s-t min-cut to determine an optimal protecting policy, which is developed in the following subsections. $\square$

D. Optimality of Protecting Policies by MCP-SA

Let $\mathcal{G}_{SA} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$ be the s-t graph and $\mathcal{C} \subseteq \mathcal{E}$ be an s-t min-cut of $\mathcal{G}_{SA}$. A state $q_{i,h}$ is said to be on the *s-side* if it is reachable from $q_{0,0}$ in the subgraph $(\mathcal{V}, \mathcal{E} \setminus \mathcal{C}, \mathbf{w})$, and $q_{i,h}$ is said to be on the *t-side* if it is coreachable to $q_{\infty,\infty}$ in the subgraph $(\mathcal{V}, \mathcal{E} \setminus \mathcal{C}, \mathbf{w})$. To prove that $C(\vartheta) = \sum_{d \in \mathcal{C}} \mathbf{w}(d)$ always holds, we first present the following lemma.

Lemma 1: Let G be the plant and $\mathcal{G}_{SA}$ be the digraph of the corresponding SA, and let $\mathcal{C}$ be a cut of $\mathcal{G}_{SA}$ with source vertex $q_{0,0}$ and terminal vertex $q_{\infty,\infty}$. For each transition $q_i \xrightarrow{\sigma} q_{i'}$ in the plant G, the following two statements hold in $\mathcal{G}_{SA}$:

1) if $q_{i',h+1}$ is on the t-side, then $q_{i,h}$ is on the t-side;
2) if $q_{i,h}$ is on the s-side, then $q_{i,h+1}$ is on the s-side;

Proof: By the construction of the SA and the s-t graph, $q_i \xrightarrow{\sigma} q_{i'}$ in G implies that in the SA there exist an arc $q_{i,h} \xrightarrow{\alpha} q_{i',h}$ for each $h = 1, \ldots, l$, and an arc $q_{i,h} \xrightarrow{\lambda} q_{i',h+1}$ for each $h = 1, \ldots, l-1$. Since the arc from $q_{i,h}$ to $q_{i',h+1}$ is labeled λ, which implies that the corresponding arc in $\mathcal{G}_{SA}$ is weighted ∞ and does not belong to the cut, then statement 1) is true.

Now we prove statement 2) by contradiction. Suppose that in $\mathcal{G}_{SA}$, state $q_{i,h}$ is on the s-side while state $q_{i,h+1}$ is on the t-side. Since $q_{i,h+1}$ is on the t-side, in $\mathcal{G}_{SA}$ there exists a path $\pi = q_{i,h+1} q_{i_1,h_1} \cdots q_{i_n,h_n}$ from $q_{i,h+1}$ to a forbidden

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2021.3091438, IEEE Transactions on Automatic Control

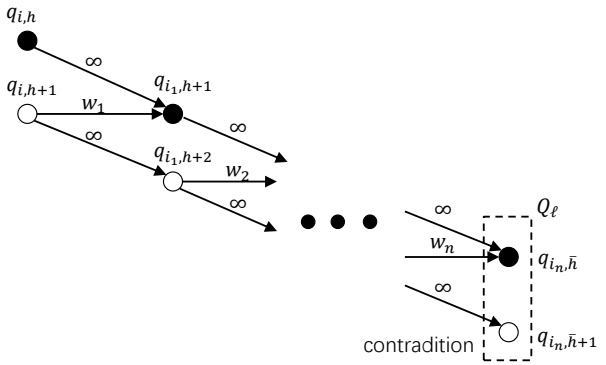


Fig. 6: The illustration of the contradiction in the proof of Lemma 1 statement 2. Solid dots and hollow circles represent s-side states and t-side states, respectively.

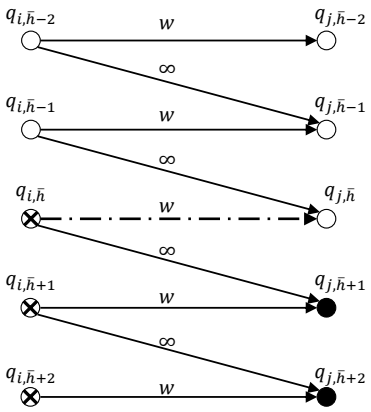


Fig. 7: The illustration of the proof of Proposition 5.1. Solid dots and hollow circles represent s-side states and t-side states, respectively, while the side of state $\bigotimes$ is uncertain.

state q_{i_n,h_n} in Q_ℓ such that all states q_{i_j,h_j} on the path are on the t-side. Since $q_{i,h}$ is on the s-side, by statement 1), $q_{i_1,h+1}$ is also on the s-side, which implies that the next state of the path π must be $q_{i_1,h_1} = q_{i_1,h+2}$ that is on the t-side. Hence, state $q_{i_1,h+1}$ is on the s-side and $q_{i_1,h+2}$ is on the t-side (see Figure 6). This reasoning can be repeated applied, which eventually leads to a conclusion that state $q_{i_n,\bar{h}} \notin Q_\ell$ and state $q_{i_n,\bar{h}+1} \in Q_\ell$. This contradicts the fact that ℓ requires that $q_{i,\bar{h}} \notin Q_\ell$ implies $q_{i,\bar{h}+1} \notin Q_\ell$. □

Now we present a main proposition of this subsection which states that if $\mathcal{C}$ is an s-t min-cut in $\mathcal{G}_{SA}$, then for each transition in the plant G, among all arcs in $\mathcal{G}_{SA}$ that are associated with the transition, at most one such arc belongs to $\mathcal{C}$.

Proposition 5.1: Given an SPP with cost function $\mathbf{c}$, let $\mathcal{G}_{SA}$ be the digraph of its corresponding SA, and $\mathcal{C}_{min}$ be an s-t min-cut of $\mathcal{G}_{SA}$ with source vertex $q_{0,0}$ and terminal vertex $q_{\infty,\infty}$. Then, for each transition $q_i \xrightarrow{\sigma} q_j$ in the plant, there exists at most one integer $h \in \{0, \ldots, l\}$ such that $(q_{i,h}, q_{j,h}) \in \mathcal{C}_{min}$.

Proof: By the construction of the SA and the s-t graph, the part of the s-t graph associated with $q_i \xrightarrow{\sigma} q_j$ is depicted in Figure 7. By Lemma 1 statement 2, there necessarily exists an nonnegative integer $\bar{h} \leq l$ such that state $q_{j,h}$ is on the t-side if and only if $h \leq \bar{h}$. Now consider state $q_{i,h}$. By Lemma 1 statement 1), for every $h \leq \bar{h} - 1$, state $q_{i,h}$ is on the t-side, since state $q_{j,\bar{h}}$ is on the t-side. Then we have:

- for all $h < \bar{h}$, arc $(q_{i,h}, q_{j,h})$ does not belong to $\mathcal{C}_{min}$ since $q_{i,h}$ and $q_{j,h}$ are both on the t-side;
- for all $h > \bar{h}$, arc $(q_{i,h}, q_{j,h})$ does not belong to $\mathcal{C}_{min}$ since $q_{j,h}$ is on the s-side.

Therefore, among all arcs that are associated with transition $q_i \xrightarrow{\sigma} q_j$, at most one arc (i.e., $(q_{i,\bar{h}}, q_{j,\bar{h}})$) may belong to $\mathcal{C}_{min}$. □

By Proposition 5.1, each plant event σ is only associated with at most one arc in an s-t min-cut $\mathcal{C}_{min}$. Recall that in Definition 5.1 we have assumed that each plant event is protected separately. We immediately have the following corollary.

Corollary 5.1: If $\mathcal{C}_{min}$ is an s-t min-cut of an MCP-SA, then $C(\vartheta_{minC}) = \sum_{d \in \mathcal{C}} \mathbf{w}(d)$ holds, where ϑ_{minC} is the dual protecting policy of $\mathcal{C}_{min}$.

Proof: This corollary directly follows from Proposition 5.1. Since each plant event σ is associated with at most one arc in $\mathcal{C}_{min}$, $C(\vartheta_{minC}) = \sum_{\sigma \in P(\vartheta_{minC})} \mathbf{c}(\sigma) = \sum_{d \in \mathcal{C}} \mathbf{w}(d)$ holds. □

Now we can finally state the main result of this section.

Theorem 5.2: Given an SPP that satisfies Assumption 1, let $\mathcal{C}_{min}$ be an s-t min-cut of the corresponding MCP-SA. The dual protecting policy ϑ_{minC} is a protecting policy of the SPP whose cost is minimal.

Proof: This proof is by contradiction. Suppose that there exists a different ϑ such that $C(\vartheta) < C(\vartheta_{minC})$. Now consider the dual cut $\mathcal{C}$ of ϑ in the s-t graph $\mathcal{G}_{SA}$. By Definition 5.5, for each protected event in $P(\vartheta)$ there exists at least one corresponding arc in $\mathcal{G}_{SA}$ which belongs to $\mathcal{C}$. Following the argument in the proof of Proposition 5.1, by removing the redundant arcs (that are associated with the same plant transition) from $\mathcal{C}$ we obtain a new cut $\mathcal{C}'$ whose corresponding protecting policy ϑ' satisfies $C(\vartheta') \leq C(\vartheta)$. Since $C(\vartheta') \leq C(\vartheta) < C(\vartheta_{minC})$, and each protected event of ϑ' and of ϑ_{minC} is only associated with one arc in $\mathcal{G}_{SA}$, we can conclude that

$$\sum_{d \in \mathcal{C}'} \mathbf{w}(d) = C(\vartheta') \leq C(\vartheta) < C(\vartheta_{minC}) = \sum_{d \in \mathcal{C}_{min}} \mathbf{w}(d),$$

i.e., $\sum_{d \in \mathcal{C}'} \mathbf{w}(d) < \sum_{d \in \mathcal{C}_{min}} \mathbf{w}(d)$. This contradicts the fact that $\mathcal{C}_{min}$ is an s-t min-cut in $\mathcal{G}_{SA}$. □

To summarize the results of this section, we present Algorithm 2 that computes a protecting policy with the minimal cost. Algorithm 2 first converts the given SPP into the corresponding MCP-SA, and attempts to determine an s-t min-cut in the s-t graph with source state $q_{0,0}$ and terminal state $q_{\infty,\infty}$. If MCP-SA does not have an s-t min-cut, then by Theorem 5.1 there does not exist a protecting policy to enforce the security requirement ℓ. On the other hand, if an s-t min-cut $\mathcal{C}_{min}$ is found, it is then converted to its dual protecting policy ϑ_{minC} whose cost is minimal by Theorem 5.2. We use the following example to illustrate Algorithm 2.

Algorithm 2 Determining a protecting policy with the minimal cost

Input: A plant $G = (Q, \Sigma, \delta, q_0)$ where $\Sigma = \Sigma_p \cup \Sigma_{up}$, a security requirement $\ell : Q \to \mathbb{N}$, and a cost function $\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$

Output: A protecting policy $\vartheta : L(G) \times \Sigma_p \to \{0, 1\}$ with the minimal cost

1: Compute the SA $H = (Q_H, \Sigma_H, \delta_H, q_{0,0})$ of G with respect to ℓ, where $\Sigma_H = \Sigma_\alpha \cup \Sigma_\lambda$;
2: Solve the MCP-SA in H for an s-t min-cut $\mathcal{C}_{min}$ with source state $q_{0,0}$ and terminal state $q_{\infty,\infty}$.
3: **if** $\mathcal{C}_{min}$ does not exist, **then**
4: output "no solution" and exit.
5: **else**
6: compute the dual protecting policy ϑ_{minC} of $\mathcal{C}_{min}$.
7: **end if**
8: Output $\vartheta = \vartheta_{minC}$ and exit.

Example 5.3: Again consider the plant in Figure 1 with $\ell(q_3) = 2, \ell(q_4) = 1$ and costs $2, 1, 1, 3, 1, \infty$ respectively for events $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6$. The corresponding SA and s-t graph are depicted in Figures 2 and 5, respectively. As discussed in Example 5.2, there exists a unique s-t min-cut $\mathcal{C} = \{(q_{0,0}, q_{1,0}), (q_{1,1}, q_{2,1}), (q_{1,1}, q_{3,1})\}$ in this s-t graph. Hence, by Definition 5.5, the protecting policy ϑ_{minC} is

$$\vartheta_{minC}(s) = \{\sigma \in \Sigma_p \mid (\delta_H^*(q_{0,0}, \theta(s)), \delta_H^*(q_{0,0}, \theta(s)\sigma_i)) \in \mathcal{C}\}$$

whose set of protected events is $P(\vartheta_{minC}) = \{\sigma_1, \sigma_2, \sigma_3\}$, and whose total cost is $C(\vartheta_{minC}) = 2 + 1 + 1 = 4$ that is equal to the cost of its dual min-cut $\mathcal{C}_{min}$. On the other hand, the minimally disruptive protecting policy ϑ_{min} obtained in Example 4.2 protects events in $P(\vartheta_{min}) = \{\sigma_1, \sigma_3, \sigma_4\}$ (protecting decisions are marked by "crosses" in Figure 5), and whose total cost is $C(\vartheta_{minC}) = 2 + 1 + 3 = 6$.

For a plant sequence $\sigma_1\sigma_2\sigma_1\sigma_2\sigma_4$ that reaches the secret state q_4 whose required security level is 2, the protecting decision for each step is given as follows:

$$\begin{cases} \vartheta(\varepsilon) = \{\sigma_1\} \\ \vartheta(\sigma_1) = \{\sigma_2, \sigma_3\} \\ \vartheta(\sigma_1\sigma_2) = \vartheta(\sigma_1\sigma_2\sigma_1) = \vartheta(\sigma_1\sigma_2\sigma_1\sigma_2) \\ \qquad\qquad = \vartheta(\sigma_1\sigma_2\sigma_1\sigma_2\sigma_4) = \emptyset \end{cases}$$

One can see that, a security check is triggered when the user switches from state q_1 to state q_2 for the first time, since the fixed cost to protect event σ_2 is less than the cost to protect event σ_4 (i.e., $\mathbf{c}(\sigma_2) < \mathbf{c}(\sigma_4)$). We also note that no more security check is triggered when the user switches from state q_1 to state q_2 for the second time and subsequently. $\diamondsuit$

At the end of this paper, we note that the complexity of all the methods proposed in this paper are polynomial. Given a plant G with n states, m events, and a security requirement ℓ such that the maximal security level required is l, the corresponding SA contains at most $n \cdot (l+1)$ states and $n \times (2l+1) \cdot m$ arcs, i.e., its structural complexity is $O(n \cdot m \cdot l)$. The conversion from an SA to its corresponding SCP-SA and MCP-SA are both linear to r where r is the number of states in the SA, and the complexity to solve the SCP-SA and the MCP-SA is of $O(r^2)$ and of $(r^2 \cdot (n \cdot m \cdot l))$, respectively. Since $O(r) = O(m \cdot n)$ and in general $n, m \gg l$, the total complexity to compute a protecting policy that is minimally disruptive is $O(m^2 \cdot n^2)$ and that to compute a protecting policy with the minimally cost is $O(n^3 \cdot m^3 \cdot l)$.

VI. Conclusions

In this paper, we have introduced a secret protection problem in discrete-event systems modeled by automata. Two notions of optimality on protecting policies, i.e., disruptiveness and cost, have been considered. To solve the problem, we have proposed a new structure called the security automaton, based on which we have developed two polynomial methods to obtain an event-protecting policy to enforce the security requirement. For the criterion of disruptiveness, a minimally disruptive protecting policy has been obtained by applying the classical supervisory control theory in the security automaton. For the criterion of cost, a new method has been developed based on the computation of an s-t min-cut in the security automaton.

In future work, we would like to extend our developed method to address cases with other criteria of optimization. On one hand, we intend to consider situations in which some specified tasks should be disturbed as little as possible when protections are applied (e.g., protections should minimally interfere reaching some non-secret states infinitely often). Another situation we are interested in investigating is that the protection costs are defined on *packages of events*, e.g., the protection of any number (> 0) of events in a package incurs a constant cost. On the other hand, if the cost of executing a protected transition is not negligible, an intruder may repeatedly executing such transitions to exhaust the budget of the system. How to minimize the cost under such type of attacks will also be investigated in our future research.

References

[1] D. Thorsley and D. Teneketzis, "Intrusion detection in controlled discrete event systems," in *Proceedings of the 45th IEEE Conference on Decision and Control*, Dec 2006, pp. 6047–6054.

[2] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[3] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys*, vol. 42, no. 1, pp. 1–31, 2009.

[4] S. Lafortune, F. Lin, and C. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annual Reviews in Control*, vol. 45, pp. 257–266, 2018.

[5] J. W. Bryans, M. Koutny, L. Mazaré, and P. Y. Ryan, "Opacity generalised to transition systems," *International Journal of Information Security*, vol. 7, no. 6, pp. 421–435, 2008.

[6] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

[7] L. Li and C. N. Hadjicostis, "Least-cost transition firing sequence estimation in labeled Petri nets with unobservable transitions," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 394–403, 2011.

[8] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua, "Verification of state-based opacity using Petri nets," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2823–2837, 2017.

[9] Y. Ji, X. Yin, and S. Lafortune, "Opacity enforcement using nondeterministic publicly known edit functions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4369–4376, 2019.

[10] Y. Ji, X. Yin, and S. Lafortune, "Optimal supervisory control with mean payoff objectives and under partial observation," *Automatica*, vol. 123, p. 109359, 2021.

[11] Y.-C. Wu and S. Lafortune, "Synthesis of insertion functions for enforcement of opacity security properties," *Automatica*, vol. 50, no. 5, pp. 1336–1348, 2014.

[12] Y. Ji, Y.-C. Wu, and S. Lafortune, "Enforcement of opacity by public and private insertion functions," *Automatica*, vol. 93, no. 7, pp. 369–378, 2018.

[13] X. Yin and S. Li, "Synthesis of dynamic masks for infinite-step opacity," *IEEE Transactions on Automatic Control*, vol. 65, no. 4, pp. 1429–1441, 2020.

[14] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Current-state opacity enforcement in discrete event systems under incomparable observations," *Discrete Event Dynamic Systems*, vol. 28, no. 2, pp. 161–182, 2018.

[15] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, 2016.

[16] 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, 2012.

[17] J. Dubreil, P. Darondeau, , and H. Marchand, "Supervisory control for opacity," *IEEE Transactions on Automatic Control*, vol. 55, no. 5, pp. 1089–1100, 2010.

[18] 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.

[19] R. Fritz and P. Zhang, "Modeling and detection of cyber attacks on discrete event systems," in *Proceedings of the 14th IFAC Workshop on Discrete Event Systems*, Sorrento, Italy, 2018, pp. 285–290.

[20] M. Agarwal, "Rogue twin attack detection: A discrete event system paradigm approach," in *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct 2019, pp. 1813–1818.

[21] C. Gao, C. Seatzu, Z. Li, and A. Giua, "Multiple attacks detection on discrete event systems," in *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Oct 2019, pp. 2352–2357.

[22] S. Matsui and K. Cai, "Secret securing with minimum cost," in *Proceedings of the 61st Japan Joint Automatic Control Conference*, 2018, pp. 1017–1024.

[23] ——, "Secret securing with multiple protections and minimum costs," in *Proceedings of the 58th IEEE Conference on Decision and Control*, 2019, pp. 7635–7640.

[24] ——, "Application of supervisory control to secret protection in discrete-event systems," *Journal of the Society of Instrument and Control Engineers*, vol. 60, no. 1, pp. 14–20, 2021.

[25] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Springer, 2019.

[26] K. Cai and W. M. Wonham, *Supervisory control of discrete-event systems*. Encyclopedia of Systems and Control, 2nd ed., Springer, 2020.

[27] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

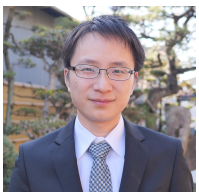
[28] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972.

[29] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 1986, pp. 136–146.

[30] R. Kumar and V. K. Garg, "Optimal supervisory control of discrete event dynamical systems," *SIAM Journal on Control and Optimization*, vol. 33, no. 2, pp. 419–439, 1995.

Ziyue Ma (S'-15, M'-17) received the B.Sc. degree and the M.Sc. degree in Chemistry from Peking University, Beijing, China, in 2007 and 2011, respectively. In 2017 he got the Ph.D degree in cotutorship between the School of Electro-Mechanical Engineering of Xidian University, China (in Mechatronic Engineering), and the Department of Electrical and Electronic Engineering of University of Cagliari, Italy (in Electronics and Computer Engineering). He joined Xidian University in 2011, where he is currently an Associate Professor in the School of Electro-Mechanical Engineering. His current research interests include control theory in discrete event systems, automata and Petri net theories, fault diagnosis/prognosis, resource optimization, and information security.

Dr. Ma is a member of Technical Committee of IEEE Control System Society (IEEE-CSS) on Discrete Event Systems. He is serving/has served as the Associate Editor of the IEEE Conference on Automation Science and Engineering (CASE'17–'21), European Control Conference (ECC'19–'21), and IEEE International Conference on Systems, Man, and Cybernetics (SMC'19–'21). He is/was the Track Committee Member of the International Conference on Emerging Technologies and Factory Automation (ETFA'17–'21). In 2016 he received the Outstanding Reviewer Award from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.



Kai Cai (S'08-M'12-SM'17) received the B.Eng. degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 2006; the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto, Toronto, ON, Canada, in 2008; and the Ph.D. degree in Systems Science from the Tokyo Institute of Technology, Tokyo, Japan, in 2011. He is currently a Professor at Osaka City University. Previously, he was an Associate Professor at Osaka City University (2014–2020), an Assistant Professor at the University of Tokyo (2013–2014), and a Postdoctoral Fellow at the University of Toronto (2011–2013).

Dr. Cai's research interests include distributed control of discrete-event systems and cooperative control of networked multi-agent systems. He is the co-author (with W.M. Wonham) of Supervisory Control of Discrete-Event Systems (Springer 2019) and Supervisor Localization (Springer 2016). He is serving as the Chair for the IEEE CSS Technical Committee on Discrete Event Systems and an Associate Editor for the IEEE Transactions on Automatic Control. He was the recipient of the Pioneer Award of SICE in 2021, the Best Paper Award of SICE in 2013, the Best Student Paper Award of the IEEE Multi-Conference on Systems and Control, and the Young Author's Award of SICE in 2010.