Nonlinear Analysis: Hybrid Systems 43 (2021) 101111

Contents lists available at ScienceDirect

Nonlinear Analysis: Hybrid Systems

journal homepage: www.elsevier.com/locate/nahs

Usability aware secret protection with minimum cost☆

Shoma Matsui [a], Kai Cai [b,*]

[a] Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada
[b] Department of Electrical and Information Engineering, Osaka City University, 3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan

ARTICLE INFO

Article history:
Received 11 February 2021
Received in revised form 15 June 2021
Accepted 15 October 2021
Available online xxxx

Keywords:
Usability
Cybersecurity
Secret protection
Supervisory control theory
Discrete-event systems
Cyber–physical systems

ABSTRACT

In this paper we study a cybersecurity problem of protecting system's secrets with multiple protections and a required security level, while minimizing the associated cost due to implementation/maintenance of these protections as well as the affected system usability. The target system is modeled as a discrete-event system (DES) in which there are a subset of marker states denoting the services/functions provided to regular users, a subset of secret states, and multiple subsets of protectable events with different security levels. We first introduce *usability-aware cost levels* for the protectable events, and then formulate the security problem as to ensure that every system trajectory that reaches a secret state contains a specified number of protectable events with at least a certain security level, and the highest usability-aware cost level of these events is minimum. We first provide a necessary and sufficient condition under which this security problem is solvable, and when this condition holds we propose an algorithm to solve the problem based on the supervisory control theory of DES. Moreover, we extend the problem to the case of heterogeneous secrets with different levels of importance, and develop an algorithm to solve this extended problem. Finally, we demonstrate the effectiveness of our solutions with a network security example.

1. Introduction

In real networked systems, risks and threats due to cybersecurity breach are increasingly prominent. Effectively protecting systems so that confidential information remains undisclosed to adversarial access has become an indispensable system design requirement [1,2].

Recently cyber–physical systems (CPS) has emerged to be a general modeling framework for real networked systems consisting of both physical and computational components. CPS security issues have attracted much attention in the literature [3–6]. For example, [4] discusses several attack scenarios with a typical architecture of networked control systems.

Focusing primarily on the abstracted level of dynamic systems, the research community of discrete-event systems (DES) has actively studied a number of security related problems. An earlier and widely investigated problem is *opacity* (e.g. [7–10]). This is a system property under partial observation such that an intruder cannot infer a given set of *secrets* by (passively) observing the system behavior. Depending on the definitions of secrets, opacity takes different forms. Recent work extends opacity notions to networked, nondeterministic settings as well as Petri net models (e.g. [11–13]).

☆ This work was supported in part by JSPS KAKENHI Grant no. 21H04875.
* Corresponding author.
E-mail addresses: s.matsui@queensu.ca (S. Matsui), kai.cai@eng.osaka-cu.ac.jp (K. Cai).

https://doi.org/10.1016/j.nahs.2021.101111
1751-570X/

Another well studied problem is *fault-tolerance* and *attack-resilience* (e.g. [14–18]). This is a design requirement that a supervisory controller should remain (reasonably) operational even after faults occur in the system or the system is under malicious attacks.

Intrusion detection is another problem that has recently attracted much interest (e.g. [19–23]). In this problem, the aim of the system administrator is to detect invasion of intruders by identifying abnormal behaviors in the system; if invasion is detected, an alarm can be set off before any catastrophic damage can be done by intruders.

From a distinct perspective, in our previous work a *minimum cost secret protection* problem is introduced [24–27]. This problem is concerned with the scenario that the system contains sensitive information or critical components to which attackers want to gain access, and attackers may be able to observe all events and disguise themselves as regular users without being detected. Then the system administrator is required to protect the sensitive information or critical components with proper security levels, while practically balance with the costs associated with the implementation and maintenance of the adopted protection methods.

In this paper, we make two important generalizations of the minimum cost secret protection problem. First, we take into account system's *usability*, which means regular users' convenience of using various services and functions provided by the system. These services and functions for regular users are often different from sensitive information or critical components that need to be protected. However, bad choices of protection points/locations may simultaneously affect access to services/functions by regular users. For example, when setting up a password to protect a user's credit card information, it is not reasonable that the user has to input the same password in order to access any websites or files. If system's usability is significantly reduced owing to setting up too many protections at inappropriate locations, users may stop using the system and this can be costly (to different extents depending on specific situations/applications). Accordingly, we formulate usability as another source of protection cost, in addition to the implementation/maintenance cost of protection methods (considered in previous work).

The second extension to the minimum cost secret protection problem is that on top of the usability consideration, we further differentiate sensitive information and critical components (or simply secrets) with distinct degrees of importance. This is a typical situation in practice; for instance, in e-commerce, customers' email addresses and credit card numbers are both sensitive information, but it is common that the latter are deemed more important and expected to be protected with stronger measures. Accordingly, we formulate heterogeneous secrets by a partition on the set of all secrets, and require that more important secrets be protected using more secure methods (while system usability still needs to be balanced).

The main contributions of this work are summarized as follows.

- A novel concept of system's usability is introduced and formulated. This notion was absent in our previous work [24–27], and to our best knowledge is new in the DES security literature. Roughly speaking, the formulation of usability is based on counting the number of affected services/functions provided to regular users when a protection is implemented at a certain location, and comparing this number to a prescribed threshold to determine if such a protection is too costly. Note that in [26] "minimal disruption" is considered for "protection of all secrets by protecting as few events as possible"; all events have the same cost/weight. Technically solving minimally descriptive secret protection is equivalent to solving the standard maximally permissive supervisory control; secret states are treated as unsafe states and there is *no* marker state. By contrast, this work explicitly considers marker states and interpret them as services/functions that the target system provides to regular users. Thereby the new notion of "usability" here is to describe "protection of all secrets while minimizing the impact on the paths leading to marker states"; events are weighted (i.e. their costs) according to their impacts on usability.
- A new usability-aware minimum cost secret protection problem is formulated, its solvability condition characterized, and a solution algorithm designed. In contrast to the problem without usability consideration [24–27], in our problem less secure protection methods that significantly undermine usability may be just as costly as more secure methods that make little impact on usability. This new feature due to usability makes our problem more challenging because security levels and cost levels of the same protection methods are generally different, and hence need to be treated separately (security levels and cost levels are not distinguished in [24–27] since usability is not considered).
- A new minimum cost secret protection problem featuring both usability awareness and heterogeneous secrets is formulated, its solvability condition characterized, and an solution algorithm developed. Not only are the formulated problem and developed solution algorithm new as compared to the existing literature, but also this problem covers a general and practical scenario in the context of secret protection.

We note that relevant to this work is opacity enforcement with consideration of security levels and protection costs that has been recently studied in the literature. In [28], a probabilistic system model and a corresponding opacity concept are introduced, and security level is measured by a probabilistic threshold as the upper bound for intruder's belief on secret states. In [29], a weighted finite-state system model is considered and various protection costs are modeled as a multidimensional integer-valued vector. Although [28,29] and our work are similar in terms of the target scenarios of security protection, there are several evident differences. First, the models and approaches of [28,29] are quantitative: probabilistic/weighted models are considered; security levels and protection costs are real numbers/integer vectors. By contrast, in our work the basic finite-state automaton is considered, and the concepts of security levels and protection costs are defined without needing to introduce extra quantification. Second, although system usability can be defined

in [29] as one component in the cost vector, in our work usability is defined to relate automaton's marker states and transition structure. This way of defining system usability is new in the literature. Third, our work further differentiates the importance of secret states, which is novel and not dealt with in [28,29]. Last but not least, the opacity enforcement problem studied in [28,29] is based on the assumption that intruder has only partial information of the system (most commonly partial observation [29], also partial confidence level [28]), and investigates how to use the partial information to 'confuse' intruder. By contrast, the secret protection problem considered in our work makes no such assumption on partial information; indeed, intruder can have full knowledge of the system and secret states. Accordingly our problem seeks how to suitably protect certain events to make it 'hard' for intruder to gain access to the secret states.

The rest of this paper is organized as follows. Section 2 introduces system model and definitions of cost; Section 3 formulates two usability aware minimum cost secret protection problems; Section 4 solves the first problem in which all secrets are deemed equally important, while Section 5 solves the second problem in which the secrets have different importance; finally in Section 6 we state our conclusions and future work.

2. System model

Consider that a system administrator needs to protect all secret information in the system. The administrator desires to do so in such a way that every secret is protected with at least a certain number of protections and these protections are of at least a certain security level.

Meanwhile, the administrator needs to balance secret protection with the associated cost. There are two sources of cost often considered in practice. One is the cost of purchasing, implementing, and maintaining the device or program for protection. This cost evidently varies depending on the means of protection; for example, a biometric device is much more costly than a password protection. Correspondingly, the higher the cost is, the higher the *security level* of the protection becomes.

The other source of cost is due to that secret protection can have the side effect of negatively impacting the convenience of regular users of the system. Unlike intruders, regular users when using the system do not always try to see the secret information (e.g. personal data), but more often use various services that the system provides (e.g. watching a movie, reading an e-book, launch an app). If protecting secrets simultaneously requires regular users to undergo many security checks before using any services, user experience or system's *usability* will decline, and if this causes users to stop using the system, the cost can be significant.

In this section, we will formulate the above-described system and cost considerations for secret protection. Our objective is to design for the administrator a *protection policy* that ensures the required level of secret protection while minimizes the incurred cost.

To model the system, we employ the framework of discrete-event systems (DES) [30,31], and consider the system modeled as a finite-state automaton

$$\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m). \tag{1}$$

Here Q is the set of states, Σ the set of events, $\delta : Q \times \Sigma \to Q$ the (partial) transition function,[1] $q_0 \in Q$ the initial state, and $Q_m \subseteq Q$ the set of *marker states* which models the set of services/functions provided by the system to its users. We denote by $Q_s \subseteq Q$ the set of *secret states* in $\mathbf{G}$; no particular relation is assumed between Q_s and Q_m, i.e. a secret state may or may not coincide with a marker state. This indicates that the services/functions which contain secrets, namely states in $Q_s \cap Q_m$ (if they exist), will unavoidably be affected by protection and thus deemed irrelevant to calculating usability cost. In addition, we extend the transition function δ to $\delta : Q \times \Sigma^* \to Q$ (where Σ^* is the set of all finite-length strings of events in Σ including the empty string ϵ) in the standard manner, and write $\delta(q, s)!$ to mean that string s is defined at state q. The *closed behavior* of $\mathbf{G}$, written $L(\mathbf{G})$, is the set of all strings that are defined at the initial state q_0:

$$L(\mathbf{G}) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}.$$

Also define the *marked behavior* of $\mathbf{G}$:

$$L_m(\mathbf{G}) = \{s \in L(\mathbf{G}) \mid \delta(q_0, s)! \ \& \ \delta(q_0, s) \in Q_m\}.$$

That is, every string in $L_m(\mathbf{G})$ is a member of the closed behavior $L(\mathbf{G})$, and moreover reaches a marker state in Q_m.

A state $q \in Q$ is *reachable* (from the initial state q_0) if there is a string s such that $\delta(q_0, s)!$ and $\delta(q_0, s) = q$. A state $q \in Q$ is *co-reachable* (to the set of marker states Q_m) if there is a string s such that $\delta(q, s)!$ and $\delta(q, s) \in Q_m$. $\mathbf{G}$ is said to be *trim* if every state is both reachable and co-reachable. Unless otherwise specified, we consider trim automaton $\mathbf{G}$ for the system model in the sequel.

In practice, not all events in the system can be protected by the administrator for reasons such as exceeding administrative permissions. Thus we partition the event set Σ into a disjoint union of the subset of *protectable* events Σ_p and the subset of *unprotectable* events Σ_{up}, namely $\Sigma = \Sigma_p \dot{\cup} \Sigma_{up}$. Moreover, protecting different events in Σ_p may incur different costs. As described at the beginning of this section, we consider two sources of cost.

[1] It is sometimes convenient to view δ as a set of triples: $\delta = \{(q, \sigma, q') \mid (q, \sigma) \mapsto q'\}$.

For the first source of purchasing/implementing/maintaining the protection device/program, we partition the set of protectable events Σ_p further into n disjoint subsets Σ_i where $i \in \{0, 1, \ldots, n-1\}$, namely

$$\Sigma_p = \dot{\bigcup}_{i=0}^{n-1} \Sigma_i. \tag{2}$$

The index i of Σ_i indicates the cost level when the system administrator protects one or more events in Σ_i; the larger the index i, the higher the cost level of protecting events in Σ_i. For simplicity we assume that the index is the deciding factor for the first source of cost; that is, the cost of protecting one event in Σ_i is sufficiently higher than the cost of protecting all events in Σ_{i-1}. While this assumption might be restrictive, it is also reasonable in many situations: for example, the cost of purchasing/installing/maintaining a biometric sensor is more costly than setting multiple password protections. Since this source of cost is directly related to the strength of protection, we will also refer to these cost levels as *security levels*.

For the second source of cost regarding regular users' convenience, we investigate the impact of protecting an event $\sigma \in \Sigma_p$ at a state q on the *usability* of services/functions provided by the system (which are modeled by the marker states in Q_m). In particular, we define for each pair (q, σ), with $\delta(q, \sigma)!$, the following set of non-secret marker states that can be reached from the state $\delta(q, \sigma)$:

$$U(q, \sigma) := \{q' \in Q_m \setminus Q_s \mid (\exists s \in \Sigma^*)\delta(\delta(q, \sigma), s)! \ \&\ \delta(q, \sigma s) = q'\}. \tag{3}$$

This $U(q, \sigma)$ is the set of (non-secret) marker states that would be affected if σ is protected at q; namely, regular users potentially would also have to go through the protected σ in order to use any of the services in this set.[2] The reason why we focus on marker states that are *not* secrets is because it is unavoidable to cause inconvenience of the users if the services/functions to be used coincide with the secrets to be protected. Note also that for different states q, q' where σ is defined, $U(q, \sigma)$ and $U(q', \sigma)$ generally have different values. Since q' can be a downstream state from q, for U to be well-defined, it is important to consider only those pairs (q, σ) where $(q, \sigma)!$, namely σ is (the first event) defined at state q.

With the set defined in (3), it is intuitive that the cost of protecting σ at q is large (resp. small) if the size of this set, i.e. $|U(q, \sigma)|$, is large (resp. small). In case the cost is overly large, this event σ (at q) belonging to (say) Σ_i (i.e. the ith cost level of the first source) may be just as costly as those events in one-level higher Σ_{i+1}. For example, if setting up a password at a particular point to protect a secret simultaneously requires all regular users to enter a password for most services the system provides, this could largely reduce the users' satisfaction; hence this password protection may be as costly as using a biometric sensor (when the latter is used to protect a secret but affecting no regular users' experience).

In practice it is case dependent as for how large this cost (measured by $|U(q, \sigma)|$) should we treat σ (defined at q) as having one-level higher cost: different systems (or business) have different criteria. Thus we consider using a positive integer $T(\geq 1)$ as a threshold number: if the cost of the second source exceeds this threshold, i.e. $|U(q, \sigma)| \geq T$, the event σ at q belong to Σ_i (say) will be treated as having the same cost level as those in Σ_{i+1}. The more important the system deems user experience, the smaller threshold T should be set.[3] Finally as noted above, the same event σ at different q generally has different $|U(q, \sigma)|$; hence this second source of cost is state-dependent (in contrast with the state-independent first source of cost).

With the above preparation, we now synergize the aforementioned two sources of cost as follows. Consider the partition of Σ_p in (2) and let $T \geq 1$ be the threshold. First define

$$C_0 := \{(\sigma, |U(q, \sigma)|) \mid q \in Q \ \&\ \sigma \in \Sigma_0 \ \&\ \delta(q, \sigma)! \ \&\ |U(q, \sigma)| < T\}. \tag{4}$$

Thus C_0 is the set of pairs in which the event belongs to Σ_0 (the lowest level of the first cost) and the $|U(q, \sigma)|$ (the second cost) is below the threshold T. In other words, these events at their respective states are the least costly ones when the first and second costs combined.

Next for each $i \in \{1, \ldots, n-1\}$, define

$$\begin{aligned} C_i :=& \{(\sigma, |U(q, \sigma)|) \mid q \in Q \ \&\ \sigma \in \Sigma_i \ \&\ \delta(q, \sigma)! \ \&\ |U(q, \sigma)| < T\} \\ & \cup \{(\sigma, |U(q, \sigma)|) \mid q \in Q \ \&\ \sigma \in \Sigma_{i-1} \ \&\ \delta(q, \sigma)! \ \&\ |U(q, \sigma)| \geq T\}. \end{aligned} \tag{5}$$

As defined, C_i is the union of two sets of pairs. The first set is analogous to C_0 (here for events in Σ_i). The second set is the collection of those pairs in which the event belongs to Σ_{i-1} (one lower level of the first cost) and the $|U(q, \sigma)|$ (the

[2] While this formulation of $U(q, \sigma)$ provides a reasonable way of capturing usability cost, this need not be the unique way. For example, an alternative is to formulate $U(q, \sigma)$ such that it does not include those (non-secret) marker states that can still be reached from q by another path not starting from σ. For clarity we focus on the formulation of $U(q, \sigma)$ in (3), while pursuing the study of other possible formulations in future work.

[3] More generally, one can consider setting up a sequence of threshold numbers $T_1 < T_2 < T_3 \cdots$, and if $|U(q, \sigma)| \in [T_1, T_2)$, treat $\sigma \in \Sigma_i$ as having the same cost as those in Σ_{i+1}; if $|U(q, \sigma)| \in [T_2, T_3)$, treat $\sigma \in \Sigma_i$ as in Σ_{i+2}, and so on. For clarity, however, of the henceforth presentation and the fact that this more general case is in principle derivable from the simpler case presented here, we will focus on the case of a single threshold number.

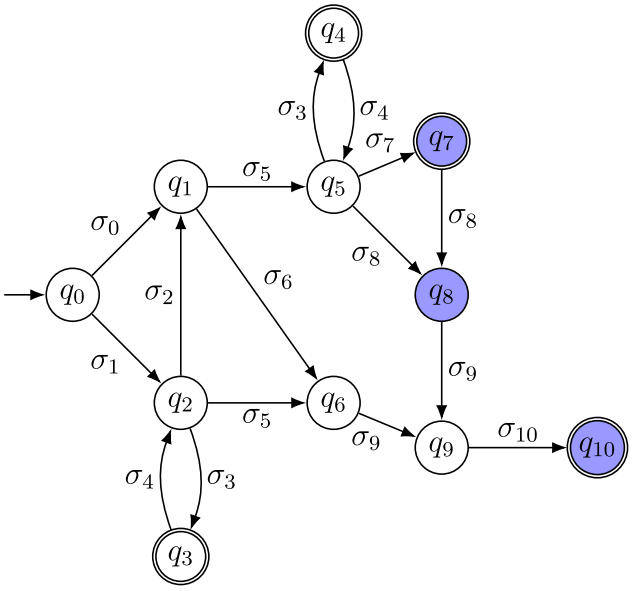

Fig. 1. System **G**: initial state q_0 (circule with an incoming arrow), marker state set $Q_m = \{q_3, q_4, q_7, q_{10}\}$ (double circles), secret state set $Q_s = \{q_7, q_8, q_{10}\}$ (shaded circles).

second cost) is larger than or equal to the threshold T. Thus the events corresponding to the second set have different levels when only the first cost is considered and when the two costs are combined.

Finally define

$$C_n := \{(\sigma, |U(q,\sigma)|) \mid q \in Q \ \& \ \sigma \in \Sigma_{n-1} \ \& \ \delta(q,\sigma)! \ \& \ |U(q,\sigma)| \geq T\}. \tag{6}$$

Thus C_n is the set of pairs in which the event belongs to Σ_{n-1} (the highest level of the first cost) and the $|U(q,\sigma)|$ (the second cost) exceeds the threshold T. That is, these events at their respective states are the most costly ones when the first and second costs combined.

It is convenient to define the set of events corresponding to C_i ($i \in [0, n]$), by projecting the elements (i.e. pairs) to their first components. Hence for $i \in [0, n]$ we write

$$\Sigma(C_i) := \{\sigma \mid (\exists q \in Q)(\sigma, |U(q,\sigma)|) \in C_i\}. \tag{7}$$

From (4)–(6), it is evident that

$$\Sigma(C_0) \subseteq \Sigma_0, \quad \Sigma(C_n) \subseteq \Sigma_{n-1}, \quad (\forall i \in [1, n-1])\Sigma(C_i) \subseteq \Sigma_{i-1} \cup \Sigma_i. \tag{8}$$

To illustrate the system modeling and cost definitions presented so far, we provide the following example, which will also be used as the running example in subsequent sections.

Example 2.1. The finite-state automaton **G** in Fig. 1 represents a simplified system model of using a software application in which there are three restricted realms. There are also four services that this system provides. Consider that this application works according to the users' permission levels. Several authentication points can be (though need not be) set up so that the users have to pass them in order to obtain the permission to reach the restricted realms. States q_7, q_8 and q_{10} represent the restricted realms modeled as secret states, i.e. $Q_s = \{q_7, q_8, q_{10}\}$. On the other hand, states q_3, q_4, q_7, and q_{10} represent the services provided by the system and hence the set of marker states is $Q_m = \{q_3, q_4, q_7, q_{10}\}$. Thus q_7 and q_{10} are simultaneously marker and secret states.

The initial state q_0 indicates that a user is about to log into the system. Accordingly, events σ_0 and σ_1 represent logging into the system as a regular user or a system administrator respectively; then q_1 and q_2 mean that the user has logged in corresponding to σ_0 and σ_1 respectively. Typically, an administrator has higher-level permission in the system compared to a regular user. Also, σ_2 indicates switching permission from the administrator to a regular user, σ_5 denotes launching the application, and σ_6 means that a regular user launches the application with the administrative permission, e.g. *sudo* in Unix-like operating systems. Events σ_3 and σ_4 are respectively the starting and finishing actions of using a system service. Moreover, σ_7 and σ_8 indicate the authentication points to obtain access to the secret states q_7 and q_8. On the other hand, the administrative realm denoted by the secret state q_{10} requires users to pass two-factor authentication represented by σ_9 (first factor) and σ_{10} (second factor). In order to keep secret states secure, the system administrator needs to configure several authentication points for restrict access.

According to the above description, the set of protectable events is

$$\Sigma_p = \{\sigma_0, \sigma_1, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}$$

which can be partitioned into four different cost/security levels (low to high):

$$\Sigma_0 = \{\sigma_0, \sigma_1, \sigma_5\}, \quad \Sigma_1 = \{\sigma_6, \sigma_7, \sigma_8\}, \quad \Sigma_2 = \{\sigma_9\}, \quad \Sigma_3 = \{\sigma_{10}\}. \tag{9}$$

That is, $\Sigma_p = \Sigma_0 \dot\cup \Sigma_1 \dot\cup \Sigma_2 \dot\cup \Sigma_3$ and $n = 4$. This is the first source of cost we consider, which corresponds to the level of security of these events. The remaining events are deemed unprotectable, i.e. $\Sigma_{up} = \{\sigma_2, \sigma_3, \sigma_4\}$.

For the second source of cost due to usability (user experience), in this example we set the threshold $T = 2$, namely if protecting an event at a state affects two or more (non-secret) services provided by the system, this cost is deemed so large that the event at the state needs to be moved one level up in terms of the total cost. In fact in **G**, there are exactly two marker states that are not secret states: q_3, q_4; hence if both these states are affected when protecting an event at a state, the threshold is reached.

Inspecting the set $U(q, \sigma)$ as defined in (3), we find $U(q_0, \sigma_1) = \{q_3, q_4\}$ because $\delta(q_0, \sigma_1\sigma_3) = q_3$ and $\delta(q_0, \sigma_1\sigma_2\sigma_5\sigma_3) = q_4$. As a result, $|U(q_0, \sigma_1)| = 2 = T$ and $\sigma_1 \in \Sigma_0$ at q_0 must be moved one level up in the total cost. Continuing this inspection, in fact $U(q_0, \sigma_1)$ is the only case where the threshold $T = 2$ is reached. Also note that event σ_5 has different $|U(\cdot, \sigma_5)|$ at different states where it is defined: $|U(q_1, \sigma_5)| = 1$ whereas $|U(q_2, \sigma_5)| = 0$. This shows that the second cost is state-dependent.

Finally we present the cost level sets with the two sources of cost combined:

$$\begin{aligned} C_0 &= \{(\sigma_0, 1), (\sigma_5, 1), (\sigma_5, 0)\} \\ C_1 &= \{(\sigma_1, 2), (\sigma_6, 0), (\sigma_7, 0), (\sigma_8, 0)\} \\ C_2 &= \{(\sigma_9, 0)\} \\ C_3 &= \{(\sigma_{10}, 0)\} \\ C_4 &= \emptyset. \end{aligned} \tag{10}$$

3. Problem formulation

Given the system model **G** in (1), the n security levels $\Sigma_0, \ldots, \Sigma_{n-1}$ in (2), and the $n+1$ cost levels $C_0, \ldots, C_n$ in (4)–(6), we formulate in this section two secret protection problems.

To proceed, we need several definitions. Let $u \geq 1$ be the least number of events that are required to be protected before any secret state may be reached from any system trajectory from the initial state. Also let $v \geq 0$ be the least security level that is needed for protecting the secrets. Write

$$\Sigma_p^{\geq v} := \dot\bigcup_{i=v}^{n-1} \Sigma_i \tag{11}$$

for the collection of protectable events where security levels are at least v. The following definition formalizes the notion that the secret states are protected with at least u number of protections with at least v security level of protectable events.

Definition 3.1 ($u-v$*−secure Reachability*)**.** Consider a system **G** in (1) with a set of secret states Q_s, the security level sets Σ_i ($i \in [0, n-1]$) in (2), and let $u \geq 1$, $v \geq 0$, and $\tilde\Sigma$ be a nonempty subset of $\Sigma_p^{\geq v}$ in (11). We say that Q_s is *reachable with at least u protectable events of security level at least v w.r.t.* $\tilde\Sigma$ (or simply Q_s is $u-v$−securely reachable) if the following condition holds:

$$(\forall s \in \Sigma^*)(\delta(q_0, s)! \,\&\, \delta(q_0, s) \in Q_s) \Rightarrow s \in \underbrace{\Sigma^* \tilde\Sigma \Sigma^* \cdots \Sigma^* \tilde\Sigma \Sigma^*}_{\tilde\Sigma \text{ appears } u \text{ times}}. \tag{12}$$

Condition (12) means that every string from the initial state that can reach a secret state must contain at least u protectable events of security level at least v.

Next we define a *protection policy* that identifies which protectable events to protect at which states. Such a policy is what we aim to design for the system administrator.

Definition 3.2 (*Protection Policy*)**.** For the system $\mathbf{G} = (Q, \Sigma = \Sigma_p \cup \Sigma_{up}, \delta, q_0, Q_m)$ in (1) and a nonempty subset of protectable events $\tilde\Sigma \subseteq \Sigma_p$, a *protection policy* $\mathcal{P}$ is a mapping that assigns to each state a subset of protectable events:

$$\mathcal{P} : Q \to \mathrm{Pwr}(\tilde\Sigma) \tag{13}$$

where $\mathrm{Pwr}(\tilde\Sigma)$ denotes the power set of $\tilde\Sigma$.

Note that what a protection policy specifies can also be interpreted as the protection of a transition labeled by a protectable event at a given state. For example, $\mathcal{P}(q) = \{\sigma_i, \sigma_j\}$ represents that protectable events σ_i and σ_j occurring at state q are protected.

Now we are ready to formulate two secret protection problems studied in this paper. The first problem is to find a protection policy (if it exists) that protects all the secret states with at least a prescribed number of protections of at least a prescribed security level, and moreover the protection cost should be minimum.

Problem 3.3 (*Usability Aware Secret Securing with Multiple Protections and Minimum Cost Problem, USCP*)**.** Consider a system **G** in (1) with a set of secret states Q_s, the cost level sets C_i ($i \in [0, n]$) in (4)–(6), and let $u \geq 1$, $v \geq 0$. Find a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, such that Q_s is $u - v-$securely reachable w.r.t. $\tilde{\Sigma}$ and the highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ is minimum (i.e. $\max\{i \in [0, n] \mid C_i \cap \tilde{\Sigma} \neq \emptyset\}$ is minimum).

Problem 3.3 states that we need to find which transitions to be protected so that all paths to secret states from the initial state have at least u protected transitions whose cost levels are equal to or higher than v, while the highest cost level of the protectable event the policy specifies is minimum.

More generally, and this is typical in practice, secrets may have different importance. For example in online shopping systems, customers' credit card information is (likely) more important than their email address information (though the latter certainly also needs to be protected). Thus the set of secret states Q_s may be partitioned into $k \geq 1$ disjoint (nonempty) subsets $Q_{s1}, \ldots, Q_{sk}$; the level of importance rises as the index increases.

Naturally the administrator wants to protect secrets of higher importance with events of higher security levels. Hence we associate each Q_{sj} ($j \in [1, k]$) with a number v_j that indicates the least security level required for protecting the secrets in Q_{sj}. These v_j satisfy $0 \leq v_1 \leq \cdots \leq v_k (\leq n - 1)$ according to the rising importance. With this additional consideration, we formulate our second problem.

Problem 3.4 (*Usability Aware Heterogeneous Secret Securing with Multiple Protections and Minimum Cost Problem, UHSCP*)**.** Consider a system **G** in (1), a set of secret states Q_s partitioned into disjoint (nonempty) subsets $Q_{s1}, \ldots, Q_{sk}$ with rising importance, the cost level sets C_i ($i \in [0, n]$) in (4)–(6), and let $u \geq 1$, $0 \leq v_1 \leq \cdots \leq v_k \leq n - 1$. Find a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, such that for every $j \in [1, k]$ the jth important secret state subset Q_{sj} is $u - v_j-$securely reachable w.r.t. $\tilde{\Sigma}$ and the highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ is minimum (i.e. $\max\{i \in [0, n] \mid C_i \cap \tilde{\Sigma} \neq \emptyset\}$ is minimum).

Let us revisit Example 2.1 to explain the above formulated two problems.

Example 3.5. Consider the system model **G** in Fig. 1, with the secret state set $Q_s = \{q_7, q_8, q_{10}\}$, the security level sets Σ_i ($i \in [0, n - 1]$) in (9), and the cost level sets C_i ($i \in [0, n]$) in (10).

For Problem 3.3, let $u = 2$ and $v = 0$; namely it is required that at least 2 events be protected for every system trajectory (from the initial state) that may reach a secret state in Q_s, and the least security level is 0. Then our goal is to find a protection policy (if it exists) $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, such that Q_s is $2 - 0-$securely reachable w.r.t. $\tilde{\Sigma}$, and moreover the highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ is minimum (i.e. least cost).

Next for Problem 3.4, we consider that Q_s is partitioned into two disjoint subsets $Q_{s1} = \{q_7, q_8\}$ and $Q_{s2} = \{q_{10}\}$. This means that q_{10}, the administrative realm, is a more important secret than q_7 and q_8 (regular users' secrets). Accordingly, let $v_1 = 0$ and $v_2 = 1$, namely the least security level for Q_{s1} is 0 while the least security level for Q_{s2} is 1; the latter means that when protecting the secret state $q_{10} \in Q_{s2}$, events $\sigma_0, \sigma_1, \sigma_5 \in \Sigma_0$ cannot be used due to their insufficient security level. As for the required number of protections, we again let $u = 2$. Then the objective here is to find a protection policy (if it exists) $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, such that Q_{s1} is $2-0-$securely reachable w.r.t. $\tilde{\Sigma}$, Q_{s2} is $2-1-$securely reachable w.r.t. $\tilde{\Sigma}$, and moreover the highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ is minimum (i.e. least cost).

4. Usability aware secret securing with minimum cost

In this section, we address Problem 3.3 (USCP). We start by characterizing the solvability of Problem 3.3, then present an algorithm to compute a solution, and finally illustrate the results using the running example (Example 2.1).

4.1. Solvability of USCP

It is evident that if there are too few protectable events or the requirement for protection numbers and security levels is too high, then there might not exist a solution to Problem 3.3. The following theorem provides a necessary and sufficient condition under which there exists a solution of Problem 3.3.

Theorem 4.1. *Consider a system* **G** *in* (1) *with a set of secret states Q_s, the cost level sets C_i ($i \in [0, n]$) in* (4)–(6)*, the required least number of protections $u \geq 1$, and the required lowest security level $v \geq 0$. Problem 3.3 is solvable (i.e. there exists a*

protection policy $\mathcal{P}: Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, *such that* Q_s *is* $u-v-$*securely reachable w.r.t.* $\tilde{\Sigma}$ *and the highest index* i *of* C_i *that has nonempty intersection with* $\tilde{\Sigma}$ *is minimum) if and only if either*

$$Q_s \text{ is } u-0\text{—securely reachable w.r.t. } \tilde{\Sigma} = \Sigma(C_0); \tag{14}$$

or there exists $i \in [v, n]$ *such that*

$$\begin{aligned} &Q_s \text{ is } u-v\text{—securely reachable w.r.t. } \tilde{\Sigma} = \bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1} \\ &\qquad\qquad \& \\ &Q_s \text{ is not } u-v\text{—securely reachable w.r.t. } \tilde{\Sigma} = \bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1}. \end{aligned} \tag{15}$$

Condition (14) means that in the special case where the required lowest security level $v = 0$, every system trajectory reaching the secret states in Q_s contains at least u protectable events in $\Sigma(C_0) \subseteq \Sigma_0$. This is the easiest case, and the index 0 is minimum.

More generally, condition (15) means that there exists an index $i \in [v, n]$ for which every system trajectory reaching the secret states in Q_s contains at least u protectable events in $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1} \subseteq \Sigma_p^{\geq v}$, but there exists at least one trajectory reaching Q_s that contains fewer than u protectable events in $\bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1} \subseteq \Sigma_p^{\geq v}$. That these two conditions in (15) simultaneously hold indicates that the index i of the cost level sets C_i is minimum. Note that in (15) the set minus "$\setminus \Sigma_{v-1}$" is needed because $\Sigma(C_v) \subseteq \Sigma_{v-1} \cup \Sigma_v$ (as in (8)), and the protectable events in Σ_{v-1} do not satisfy the required security level v.

Proof ($\Rightarrow$)**.** If condition (14) holds, i.e. Q_s is $u-0-$securely reachable w.r.t. $\Sigma(C_0) \subseteq \Sigma_0$, then the index 0 is evidently the smallest. In this case, there exists a protection policy $\mathcal{P}: Q \to \mathrm{Pwr}(\Sigma(C_0))$ as a solution for Problem 3.3 using protectable events only in $\Sigma(C_0) \subseteq \Sigma_0$ which satisfies the required security level 0. Therefore, if (14) holds, then Problem 3.3 is solvable (for the special case $v = 0$).

If (15) holds, then Q_s is $u-v-$securely reachable w.r.t. $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$, and moreover the index i of C_i is minimum. The latter is because Q_s is not $u-v-$securely reachable w.r.t. $\bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1}$ and $\bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1} \subseteq \bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$. In this case, there exists a protection policy $\mathcal{P}: Q \to \mathrm{Pwr}(\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1})$ as a solution for Problem 3.3 using protectable events in $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1} \subseteq \Sigma_p^{\geq v}$ which satisfies the required security level v. Therefore, if (15) holds, then Problem 3.3 is solvable.

($\Leftarrow$) If Problem 3.3 is solvable with the minimum index of C_i being $i = 0$, then Q_s is $u-0-$securely reachable w.r.t. $\Sigma(C_0)$. This is exactly condition (14).

If Problem 3.3 is solvable with the minimum index of C_i satisfying $v \leq i \leq n$, then Q_s is $u-v-$securely reachable w.r.t. $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$. Since the index i is minimum, it indicates that Q_s is not $u-v-$securely reachable w.r.t. $\bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1}$. Therefore (15) holds. □

4.2. Policy computation for USCP

When Problem 3.3 is solvable under the condition presented in Theorem 4.1, we design an algorithm to compute a solution, namely a protection policy.

To compute such a protection policy, our approach is to convert Problem 3.3 (a security problem) to a corresponding control problem and adapt methods from the supervisory control theory.

By this conversion, the sets of protectable events Σ_p and unprotectable events Σ_{up} are interpreted as the sets of *controllable events* Σ_c and *uncontrollable events* Σ_{uc}, respectively. Accordingly, a system $\mathbf{G}$ in (1) is changed to

$$\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m) \tag{16}$$

where $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$ and $\Sigma_c = \dot{\bigcup}_{i=0}^{n-1} \Sigma_i$. Recall from (2) that Σ_i ($i = 0, \ldots, n-1$) denote the partition of protectable events in Σ_p as the index i represents the security level (and the first source of cost); accordingly, here Σ_i denote the partition of controllable events in Σ_c. Similar to (11), for a given $v \geq 0$ write

$$\Sigma_c^{\geq v} := \dot{\bigcup}_{i=v}^{n-1} \Sigma_i. \tag{17}$$

In addition, protection policy $\mathcal{P}: Q \to \mathrm{Pwr}(\Sigma_p)$ is changed to *control policy* $\mathcal{D}: Q \to \mathrm{Pwr}(\Sigma_c)$, which is a control decision (of a supervisor) specifying which controllable events to disable at any given state. More specifically, let $\mathbf{S} = (X, \Sigma, \xi, x_0, X_m)$ be a supervisor for system $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ and assume without loss of generality that $\mathbf{S}$ is a subautomaton of $\mathbf{G}$. The control policy $\mathcal{D}: Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_c$, is given by

$$\mathcal{D}(q) := \begin{cases} \{\sigma \in \tilde{\Sigma} \mid \neg \xi(q, \sigma)! \,\&\, \delta(q, \sigma)!\}, & \text{if } q \in X \\ \emptyset, & \text{if } q \in Q \setminus X \end{cases} \tag{18}$$

Based on the above conversion, Definition 3.1 and Problem 3.3 are changed to the following definition and problem.

Definition 4.2 ($u-v$*−controllable Reachability*)**.** Consider a system $\mathbf{G}$ in (16) with a set of secret states Q_s, the (security) level sets Σ_i ($i \in [0, n-1]$) in (2), and let $u \geq 1$, $v \geq 0$, and $\tilde{\Sigma}$ be a nonempty subset of $\Sigma_c^{\geq v}$ in (17). We say that Q_s is *reachable with at least u controllable events of (security) level at least v w.r.t.* $\tilde{\Sigma}$ (or simply Q_s is $u-v$−controllably reachable) if the following condition holds:

$$(\forall s \in \Sigma^*)(\delta(q_0, s)! \,\&\, \delta(q_0, s) \in Q_s) \Rightarrow s \in \underbrace{\Sigma^* \tilde{\Sigma} \Sigma^* \cdots \Sigma^* \tilde{\Sigma} \Sigma^*}_{\tilde{\Sigma} \text{ appears } u \text{ times}}. \tag{19}$$

Problem 4.3 (*Usability Aware Reachability Control with Multiple Controllable Events and Minimum Cost Problem, UCCP*)**.** Consider a system $\mathbf{G}$ in (16) with a set of secret states Q_s, the cost level sets C_i ($i \in [0, n]$) in (4)–(6), and let $u \geq 1$, $v \geq 0$. Find a control policy $\mathcal{D} : Q \to \text{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_c$, such that Q_s is $u-v$−controllably reachable w.r.t. $\tilde{\Sigma}$ and the highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ is minimum (i.e. $\max\{i \in [0, n] \mid C_i \cap \tilde{\Sigma} \neq \emptyset\}$ is minimum).

The solvability condition of Problem 4.3, stated in the corollary below, follows directly from Theorem 4.1 and the above presented conversion.

Corollary 4.4. *Consider a system* $\mathbf{G}$ *in* (16) *with a set of secret states* Q_s*, the cost level sets* C_i ($i \in [0, n]$) *in* (4)–(6)*, the required least number of protections* $u \geq 1$*, and the required lowest (security) level* $v \geq 0$*. Problem 4.3 is solvable (i.e. there exists a control policy* $\mathcal{D} : Q \to \text{Pwr}(\tilde{\Sigma})$*,* $\tilde{\Sigma} \subseteq \Sigma_c$*, such that* Q_s *is* $u-v$*−controllably reachable w.r.t.* $\tilde{\Sigma}$ *and the highest index* i *of* C_i *that has nonempty intersection with* $\tilde{\Sigma}$ *is minimum) if and only if either*

$$Q_s \text{ is } u-0\text{−controllably reachable w.r.t. } \tilde{\Sigma} = \Sigma(C_0); \tag{20}$$

or there exists $i \in [v, n]$ *such that*

$$\begin{aligned} &Q_s \text{ is } u-v\text{−controllably reachable w.r.t. } \tilde{\Sigma} = \bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1} \\ &\qquad\qquad \& \\ &Q_s \text{ is not } u-v\text{−controllably reachable w.r.t. } \tilde{\Sigma} = \bigcup_{l=v}^{i-1} \Sigma(C_l) \setminus \Sigma_{v-1}. \end{aligned} \tag{21}$$

When Problem 4.3 is solvable (equivalently Problem 3.3 is solvable), we present an algorithm to compute a control policy as a solution for Problem 4.3. Such a control policy specifies at least u controllable events of (security) level at least v to disable in every string from the initial state q_0 to the secret state set Q_s. This control policy will finally be converted back to a protection policy as a solution for Problem 3.3 (our original security problem).

The algorithm that we design to solve Problem 4.3 is presented on the next page (Algorithm 1 UCCu). In the following we explain the main ingredients and steps of this algorithm.

First, the inputs of Algorithm 1 are the system $\mathbf{G}$ in (16), a set of secret states Q_s, the least number of protections $u \geq 1$, and the least (security) level $v \geq 0$. Then Algorithm 1 will output u supervisors $\mathbf{S}_0, \ldots, \mathbf{S}_{u-1}$ for $\mathbf{G}$ (if they exist) as well as the minimum cost index $i_{\min}$. Each supervisor is computed by the UCC function (lines 14–24), and provides a different control policy such that every string reaching secret states has at least one controllable event of (security) level at v. So in total, $\mathbf{S}_0, \ldots, \mathbf{S}_{u-1}$ specify u controllable events to disable in every string reaching Q_s.

To compute the first supervisor $\mathbf{S}_0$, at line 1 of Algorithm 1 we need to design the control specification $\mathbf{G}_K$. This is done by removing from $\mathbf{G}$ all the secret states in Q_s and the transition to and from the removed states. Hence

$$\mathbf{G}_K = (Q \setminus Q_s, \Sigma, \delta_K, q_0, Q \setminus Q_s) \tag{22}$$

where $\delta_K = \delta \setminus \{(q, \sigma, q') \mid q \text{ or } q' \in Q_s, \sigma \in \Sigma, \delta(q, \sigma)!, \delta(q, \sigma) = q'\}$.[4] We remark that for $\mathbf{G}_K$ we let all of its states be marked; this is because we do not want to introduce extra control actions owing to ensuring nonblocking behavior.

Example 4.5. Displayed in Fig. 2 is the specification automaton $\mathbf{G}_K$ derived from the system $\mathbf{G}$ in Example 2.1 and the secret state set $Q_s = \{q_7, q_8, q_{10}\}$. To design $\mathbf{G}_K$, secret states in $Q_s = \{q_7, q_8, q_{10}\}$ and transitions (q_5, σ_7, q_7), (q_5, σ_8, q_8), (q_7, σ_8, q_8), (q_8, σ_9, q_9) and $(q_9, \sigma_{10}, q_{10})$ are removed from $\mathbf{G}$ in Fig. 1 and all the states of $\mathbf{G}_K$ are marked.

[4] Note that in real systems, secret states should still be reachable. Even though the computed supervisors specify which controllable events to disable in the control context, we consider the *protection* of these specified events so that secret states are still reachable but protected. Our view is that in real systems, it is not desirable to disable controllable events and make secret states unreachable, because it would prevent regular users from ever accessing these secret states as well.

Algorithm 1: UCCu.

Input: System $\mathbf{G}$, secret state set Q_s, protection number u, security level v
Output: Supervisors $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{u-1}$, minimum cost index $i_{\min}$
1: $\mathbf{G}_0 = (Q, \Sigma^0, \delta^0, q_0, Q_m) = \mathbf{G}$, $\mathbf{G}_{K,0} = \mathbf{G}_K$ as in (22)
2: **for** $j = 0, 1, \ldots, u-1$ **do**
3: $(\mathbf{S}_j, i_j) = \text{UCC}(\mathbf{G}_j, \mathbf{G}_{K,j}, v)$
4: **if** $\mathbf{S}_j$ is nonempty **then**
5: Derive $\mathcal{D}_j$ from $\mathbf{S}_j$ as in (18)
6: Form $\mathbf{G}_{j+1} = (Q, \Sigma^{j+1}, \delta^{j+1}, q_0, Q_m)$ from $\mathbf{G}_j$ and $\mathcal{D}_j$ as in (25)
7: $\delta_K^{j+1} = \delta^{j+1} \setminus \{(q, \sigma, q') \mid q \text{ or } q' \in Q_s, \sigma \in \Sigma^{j+1}, \delta^{j+1}(q, \sigma) = q'\}$
8: $\mathbf{G}_{K,j+1} = (Q \setminus Q_s, \Sigma^{j+1}, \delta_K^{j+1}, q_0, Q \setminus Q_s)$
9: **else**
10: **return** Empty supervisors, index -1
11: **end if**
12: **end for**
13: **return** $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{u-1}$, $i_{\min} = i_{u-1}$

14: **function** UCC($\mathbf{G}, \mathbf{G}_K, v$)
15: $K = L(\mathbf{G}_K)$
16: **for** $i = v, v+1, \ldots, n$ **do**
17: $\Gamma = \bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$
18: Compute a supervisor $\mathbf{S}$ s.t. $L(\mathbf{S}) = \sup \mathcal{C}(K)$ w.r.t. $\mathbf{G}$ and Γ
19: **if** $\mathbf{S}$ is nonempty **then**
20: **return** $(\mathbf{S}, i)$
21: **end if**
22: **end for**
23: **return** (empty supervisor, index -1)
24: **end function**

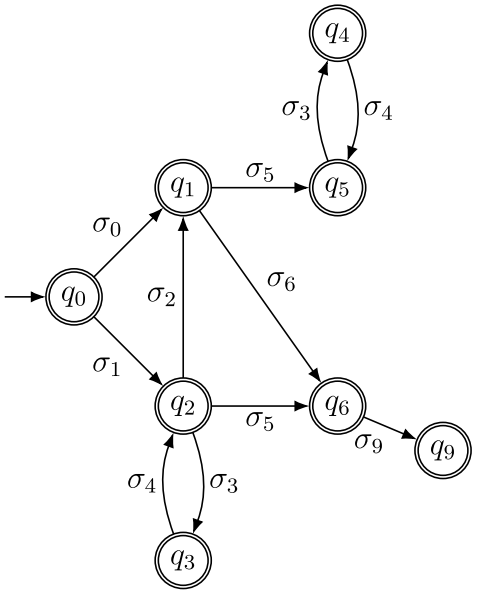

Fig. 2. Specification automaton $\mathbf{G}_K$.

With $\mathbf{G}_K$ constructed, line 2 of Algorithm 1 starts from $j = 0$ and line 3 calls the RCMC function (with arguments $\mathbf{G}_0 = \mathbf{G}$, $\mathbf{G}_{K,0} = \mathbf{G}_K$, v) to compute the first supervisor $\mathbf{S}_0$ and the minimum cost index i_0. To this end, several standard concepts of supervisory control theory (SCT) [30,32,33] are employed and briefly reviewed below.

Consider a system $\mathbf{G} = (Q, \Sigma = \Sigma_c \cup \Sigma_{uc}, \delta, q_0, Q_m)$ in (16), and let $K = L(\mathbf{G}_K) \subseteq L(\mathbf{G})$ be a specification language derived from the specification automaton $\mathbf{G}_K$ in (22). For a subset of the controllable events $\Gamma (\subseteq \Sigma_c)$, K is said to be

controllable with respect to $\mathbf{G}$ and Γ if $\overline{K}(\Sigma \setminus \Gamma) \cap L(\mathbf{G}) \subseteq \overline{K}$ where $\overline{K}$ is the *prefix closure* of K. We denote by the family $\mathcal{C}(K) := \{K' \subseteq K \mid \overline{K}(\Sigma \setminus \Gamma) \cap L(\mathbf{G}) \subseteq \overline{K}\}$ the set of all controllable sublanguages of K with respect to $\mathbf{G}$ and Γ, and by $\sup \mathcal{C}(K) := \bigcup\{K' \mid K' \in \mathcal{C}(K)\}$ the supremal controllable sublanguage of K with respect to $\mathbf{G}$ and Γ (which is known to always exist).

Lemma 4.6 (*cf. [30]*). *Consider a plant* $\mathbf{G} = (Q, \Sigma = \Sigma_c \cup \Sigma_{uc}, \delta, q_0, Q_m)$ *in* (16) *and a specification language* $K \subseteq L(\mathbf{G})$. *It holds that*

$$\sup \mathcal{C}(K) = \emptyset \text{ (w.r.t. } \mathbf{G} \text{ and } \Sigma_{uc}) \Leftrightarrow (\exists s \in \Sigma_{uc}^*)s \in L(\mathbf{G}) \setminus K. \tag{23}$$

From Lemma 4.6 and the construction of $\mathbf{G}_K$ in (22), letting $K = L(\mathbf{G}_K)$ and $i \in [v, n]$, we know that the first supervisor $\mathbf{S}_0 = \sup \mathcal{C}(K)$ (with respect to $\mathbf{G}$ in (16) and $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$) is nonempty if and only if every string reaching the secret states in Q_s from the initial state q_0 has at least one controllable event belonging to $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$. In other words, $\sup \mathcal{C}(K) \neq \emptyset$ (with respect to $\mathbf{G}$ and $\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$) if and only if

$$\left(\forall s \in \left(\Sigma \setminus \left(\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}\right)\right)^*\right) \delta(q_0, s) \notin Q_s. \tag{24}$$

The computation of $\mathbf{S}_0$ is carried out in lines 15–22 of Algorithm 1. If a nonempty $\mathbf{S}_0$ is obtained (line 19; condition (24) holds), then it is returned together with the current index i of the cost level sets (line 20). Since the index is incrementally increased (line 16), we know that the index i in line 20 is minimum (for this is the first time that $\mathbf{S}_0$ is nonempty).

Once a nonempty supervisor $\mathbf{S}_j$ ($j \geq 0$) is obtained (line 4), Algorithm 1 proceeds to compute the next supervisor $\mathbf{S}_{j+1}$ (until we acquire u nonempty supervisors). To ensure that each supervisor provides a different control policy (disabling different transitions) so as to meet the requirement of u protections, we need to change the status of those transitions already disabled by $\mathbf{S}_j$ from controllable to uncontrollable, so that the next supervisor $\mathbf{S}_{j+1}$ is forced to disable other controllable transitions.

This status change is done by event relabeling. Specifically, let $\mathbf{G}_j = (Q, \Sigma^j = \Sigma_{uc,j} \dot{\cup} \Sigma_{c,j}, \delta^j, q_0, Q_m)$ be the jth system model and $\mathcal{D}_j$ be the control policy in (18) corresponding to supervisor $\mathbf{S}_j$. Then the set of controllable transitions specified (or disabled) by $\mathcal{D}_j$ is

$$\delta_{\mathcal{D}_j} := \{(q, \sigma, q') \mid q \in Q \,\&\, \sigma \in \mathcal{D}_j(q) \,\&\, q' = \delta^j(q, \sigma)\}.$$

We relabel the above transitions and obtain

$$\delta'_{\mathcal{D}_j} := \{(q, \sigma', q') \mid (q, \sigma, q') \in \delta_{\mathcal{D}_j} \,\&\, \sigma' \notin \Sigma^j\}.$$

Moreover, we designate these relabeled transition as uncontrollable, so the new uncontrollable event set is:

$$\Sigma_{uc,j+1} = \Sigma_{uc,j} \dot{\cup} \{\sigma' \mid (q, \sigma', q') \in \delta'_{\mathcal{D}_j}\}.$$

On the other hand, the new controllable event set is:

$$\Sigma_{c,j+1} = \Sigma_{c,j} \setminus \{\sigma \mid (\forall q \in Q)\delta^j(q, \sigma)! \,\&\, \delta^j(q, \sigma) = q' \Rightarrow (q, \sigma, q') \in \delta_{\mathcal{D}_j}\}.$$

In words, those controllable events whose corresponding transitions are all specified by $\mathcal{D}_j$ and therefore relabeled no longer exist and are consequently removed from the controllable event set. Therefore we obtain the new system model

$$\mathbf{G}_{j+1} = (Q, \Sigma^{j+1}, \delta^{j+1}, q_0, Q_m) \tag{25}$$

where

$$\Sigma^{j+1} = \Sigma_{uc,j+1} \dot{\cup} \Sigma_{c,j+1} \tag{26}$$

$$\delta^{j+1} = (\delta^j \setminus \delta_{\mathcal{D}_j}) \dot{\cup} \delta'_{\mathcal{D}_j}. \tag{27}$$

The above is carried out in lines 5–6 of Algorithm 1. Moreover, lines 7–8 update the specification model $\mathbf{G}_{K,j+1}$ similar to (22).

With the updated system $\mathbf{G}_{j+1}$ and specification $\mathbf{G}_{K,j+1}$, Algorithm 1 again calls the UCC function (line 3) to compute the next supervisor $\mathbf{S}_{j+1}$ and the corresponding minimum cost index i_{j+1}. This process continues until $j = u - 1$, unless an empty supervisor is returned by the UCC function. In the latter case, Algorithm 1 returns empty supervisors and index -1.

If Algorithm 1 succeeds to compute u nonempty supervisors $\mathbf{S}_0, \ldots, \mathbf{S}_{u-1}$, then these supervisors will be returned, together with the minimum cost index $i_{\min} = \max(i_0, \ldots, i_{u-1})$ (line 13). It is evident from the above construction that the inequality chain $v \leq i_0 \leq \cdots \leq i_{u-1} \leq n$ holds; hence $i_{\min} = i_{u-1}$.

Let $\mathcal{D}_j$ be the control policy of $\mathbf{S}_j$ ($j = 0, \ldots, u-1$). Then define the overall control policy $\mathcal{D} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_c$, by taking the union of the controllable events specified by individual $\mathcal{D}_j$ at every state, namely

$$\mathcal{D}(q) = \bigcup_{j=0}^{u-1} \mathcal{D}_j(q), \quad q \in Q. \tag{28}$$

Since each control policy $\mathcal{D}_j$ ($j \in [0, u-1]$) specifies controllable events such that every string reaching secret states has at least one disabled event, $\mathcal{D}$ in (28) specifies at least u controllable events to disable in every string reaching secret states from the initial state. Moreover, it follows from line 16 of Algorithm 1 that the (security) level of all these u events are at least v.

The time complexity of Algorithm 1 is $O(u(n-v)|Q|^2)$, where u is from line 2, $n-v$ from line 16, and $|Q|^2$ from line 18 (from the standard supervisory control theory [30]). The correctness of Algorithm 1 is asserted in the following proposition.

Proposition 4.7. *Algorithm 1 (with inputs* $\mathbf{G}$, Q_s, u *and* v*) returns* u *nonempty supervisors and minimum cost index* $i_{\min}(\in [v, n])$ *if and only if Problem 4.3 is solvable.*

Proof. By the aforementioned constructions in Algorithm 1, in particular line 16 (incrementally increasing the index of cost level sets) and line 17 ($\bigcup_{l=v}^{i} \Sigma(C_l) \setminus \Sigma_{v-1}$ monotonically becoming larger as index i increases), Algorithm 1 returns u nonempty supervisors and minimum cost index $i_{\min} \in [v, n]$ if and only if either of the two conditions (20), (21) holds. By Corollary 4.4, the latter is a necessary and sufficient condition for the solvability of Problem 4.3. Therefore our conclusion ensues. □

From the derived control policy $\mathcal{D}$ in (28), a solution for Problem 3.3, namely a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, is obtained by inverse conversion of controllable events back to protectable events. In terms of $\mathcal{P}$, we interpret disabled events by $\mathcal{D}$ as *protected events*.

Finally, we state the main result in this section, which provides a solution to our original security protection problem USCP (Problem 3.3).

Theorem 4.8. *Consider a system* $\mathbf{G}$ *in (1) with a set of secret states* Q_s*, the cost level sets* C_i *($i \in [0, n]$) in (4)–(6), the required least number of protections* $u \geq 1$*, and the required lowest security level* $v \geq 0$*. If Problem 3.3 is solvable, then the protection policy* $\mathcal{P}$ *derived from* $\mathcal{D}$ *in (28) is a solution.*

Proof. Suppose that Problem 3.3 is solvable. Then Problem 4.3 is also solvable by conversion of protectable events to controllable events. Then by Proposition 4.7, Algorithm 1 returns u nonempty supervisors and the minimum cost index $i_{\min} \in [v, n]$. Based on these u supervisors, control policies $\mathcal{D}_0, \ldots, \mathcal{D}_{u-1}$ may be derived as in (18). Hence, a combined control policy $\mathcal{D}$ in (28) is obtained. Due to the event relabeling in (25), each control policy uniquely specifies transitions in $\mathbf{G}$ to disable. Also it follows from the specifications $\mathbf{G}_{K,0}, \ldots, \mathbf{G}_{K,u-1}$ in Algorithm 1 that Q_s is $1-v$—controllably reachable under each of $\mathcal{D}_0, \ldots, \mathcal{D}_{u-1}$. Therefore, under control policy $\mathcal{D}$, Q_s is $u-v$—controllably reachable. Hence, the control policy $\mathcal{D}$ is a solution for Problem 4.3. Consequently, from the inverse conversion of controllable events back to protectable events, the protection policy $\mathcal{P}$ derived from $\mathcal{D}$ is a solution for Problem 3.3. □

4.3. Running example

Let us again use Example 2.1 to demonstrate our developed solution via Algorithm 1 for Problem 3.3.

Consider the system $\mathbf{G}$ in Fig. 1, with the secret state set $Q_s = \{q_7, q_8, q_{10}\}$, the security level sets Σ_i ($i \in [0, 3]$) in (9), and the cost level sets C_i ($i \in [0, 4]$) in (10). Let $u = 2$ and $v = 0$; namely it is required that at least 2 events be protected for every system trajectory (from the initial state) that may reach a secret state in Q_s, and the least security level is 0. We demonstrate how to use Algorithm 1 to compute a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, and the minimum highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ as a solution for Problem 3.3.

First, convert protectable events to controllable events such that

$$\Sigma_c = \{\sigma_0, \sigma_1, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}.$$

Accordingly the uncontrollable event set $\Sigma_{uc} = \{\sigma_2, \sigma_3, \sigma_4\}$. Then input Algorithm 1 with the converted system model $\mathbf{G}$, Q_s, $u = 2$ and $v = 0$.

In the first iteration ($j = 0$), system $\mathbf{G}_0 = \mathbf{G}$ in Fig. 1 and specification $\mathbf{G}_{K,0} = \mathbf{G}_K$ in Fig. 2. Then the RCMC function is called to compute the first supervisor $\mathbf{S}_0$. It is verified that when $i = 0$ (line 16), the supervisor $\mathbf{S}$ is empty (line 18), whereas when $i = 1$, the supervisor $\mathbf{S}$ is nonempty. Thus this nonempty supervisor is returned as $\mathbf{S}_0$ and the index 1 is returned as i_0 (line 20). The control policy $\mathcal{D}_0$ corresponding to $\mathbf{S}_0$ is:

$$\mathcal{D}_0(q_1) = \{\sigma_6\}, \quad \mathcal{D}_0(q_2) = \{\sigma_5\}, \quad \mathcal{D}_0(q_5) = \{\sigma_7, \sigma_8\},$$
$$(\forall q \in Q \setminus \{q_1, q_2, q_5\})\mathcal{D}_0(q) = \emptyset.$$

Fig. 3 depicts the control policy $\mathcal{D}_0$ over the plant $\mathbf{G}$ in Fig. 1, indicating the disabled transitions by "✖".

We remark that since the lowest security level set is $\Sigma_0 = \{\sigma_0, \sigma_1, \sigma_5\}$, it would have been sufficient to disable σ_0, σ_1 at q_0 to satisfy the required $v = 0$. However, disabling σ_1 would simultaneously affect regular users' accessing the (non-secret) marker states q_3, q_4, and this is deemed too costly in this example setting (threshold number is $T = 2$ for the

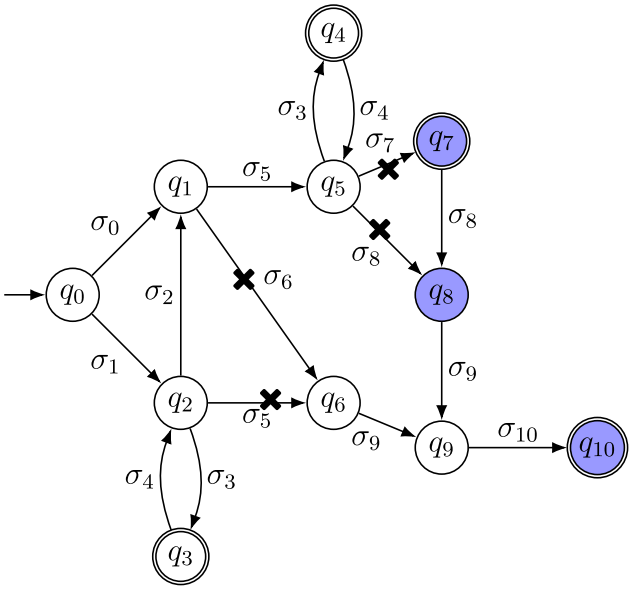

Fig. 3. Control policy $\mathcal{D}_0$ of $\mathbf{S}_0$.

number of affected non-secret marker states). This observation makes it evident that taking into account the cost of usability generally requires the administrator to adopt a different protection policy.

After obtaining $\mathcal{D}_0$, Algorithm 1 proceeds to relabel the disabled transitions by $\mathcal{D}_0$ as follows:

$$\delta_{\mathcal{D}_0} = \{(q_1, \sigma_6, q_6), (q_2, \sigma_5, q_6), (q_5, \sigma_7, q_7), (q_5, \sigma_8, q_8)\}$$
$$\delta'_{\mathcal{D}_0} = \{(q_1, \sigma'_6, q_6), (q_2, \sigma'_5, q_6), (q_5, \sigma'_7, q_7), (q_5, \sigma'_8, q_8)\}.$$

The relabeled events are designated to be uncontrollable events; thus the new uncontrollable event set is

$$\Sigma_{uc,1} = \Sigma_{uc} \dot{\cup} \{\sigma'_5, \sigma'_6, \sigma'_7, \sigma'_8\}.$$

On the other hand, the new controllable event set is

$$\Sigma_{c,1} = \Sigma_c \setminus \{\sigma_6, \sigma_7\}.$$

Note that events σ_5, σ_8 remain in $\Sigma_{c,1}$ since they have other instances (of transitions) that are not disabled by $\mathcal{D}_0$. From the above, the new system becomes $\mathbf{G}_1 = (Q, \Sigma^1, \delta^1, q_0, Q_m)$ where

$$\Sigma^1 = \Sigma_{uc,1} \dot{\cup} \Sigma_{c,1}, \quad \delta^1 = (\delta \setminus \delta_{\mathcal{D}_0}) \dot{\cup} \delta'_{\mathcal{D}_0}$$

and the new specification automaton becomes

$$\mathbf{G}_{K,1} = (Q \setminus Q_s, \Sigma^1, \delta^1_K, q_0, Q \setminus Q_s)$$

where

$$\delta^1_K = \delta^1 \setminus \{(q, \sigma, q') \mid q \text{ or } q' \in Q_s, \sigma \in \Sigma^1, \delta^1(q, \sigma) = q'\}.$$

The new system $\mathbf{G}_1$ and specification $\mathbf{G}_{K,1}$ are displayed in Fig. 4 and Fig. 5, respectively.

With $\mathbf{G}_1$ and $\mathbf{G}_{K,1}$, Algorithm 1 in the second iteration ($j = 1$) again calls the RCMC function to compute the second supervisor $\mathbf{S}_1$. Like in the first iteration, when $i = 0$ (line 16) the supervisor $\mathbf{S}$ is empty (line 18), whereas when $i = 1$ the supervisor $\mathbf{S}$ is nonempty. Thus this nonempty supervisor is returned as $\mathbf{S}_1$ and the index 1 is returned as i_1 (line 20). The control policy $\mathcal{D}_1$ corresponding to $\mathbf{S}_1$ is:

$$\mathcal{D}_0(q_0) = \{\sigma_0, \sigma_1\}, \quad (\forall q \in Q \setminus \{q_0\}) \mathcal{D}_0(q) = \emptyset.$$

By now Algorithm 1 has succeeded in computing two nonempty supervisors. Since $u = 2$, Algorithm 1 terminates and returns $\mathbf{S}_0$, $\mathbf{S}_1$, and the minimum cost index $i_{\min} = i_1 = 1$. Now we combine the two corresponding control policies into

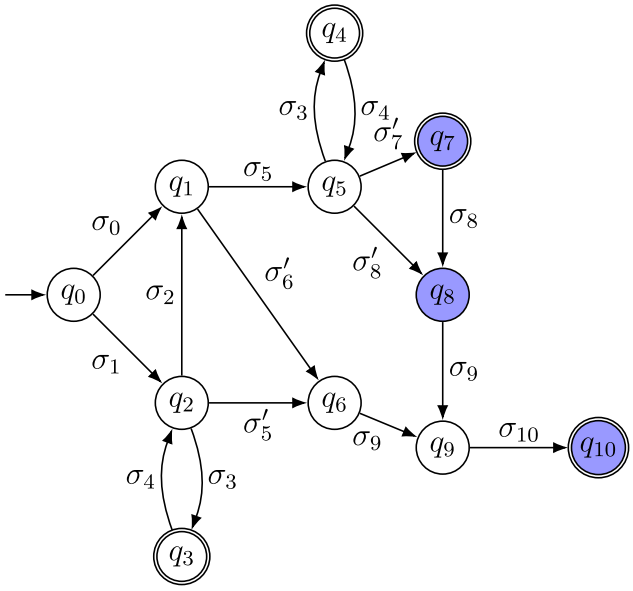

Fig. 4. Relabeled system $\mathbf{G}_1$.

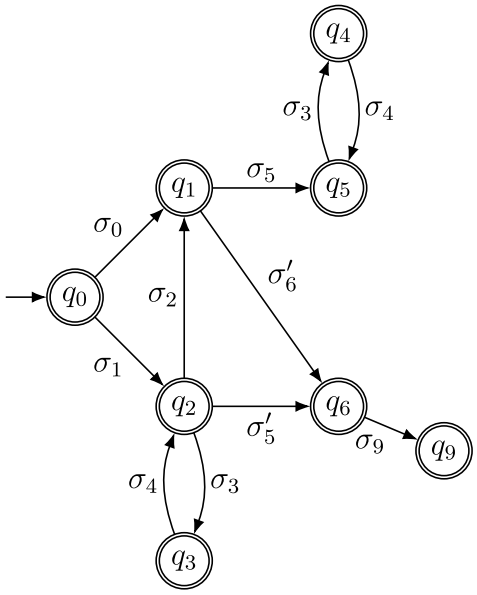

Fig. 5. Updated specification $\mathbf{G}_{K,1}$.

$\mathcal{D}$ as follows:

$$\mathcal{D}(q) = \begin{cases} \{\sigma_0, \sigma_1\}, & \text{if } q = q_0 \\ \{\sigma_6\}, & \text{if } q = q_1 \\ \{\sigma_5\}, & \text{if } q = q_2 \\ \{\sigma_7, \sigma_8\}, & \text{if } q = q_5 \\ \emptyset, & \text{if } q \in Q \setminus \{q_0, q_1, q_2, q_5\} \end{cases}$$

This $\mathcal{D}$ is a solution of Problem 4.3.

Finally, by inverse conversion of controllable events back to protectable events we obtain a corresponding protection policy $\mathcal{P}$ as a solution of the original Problem 3.3. Fig. 6 illustrates this protection policy $\mathcal{P}$, where "🔒" means the transitions that need to be "protected".

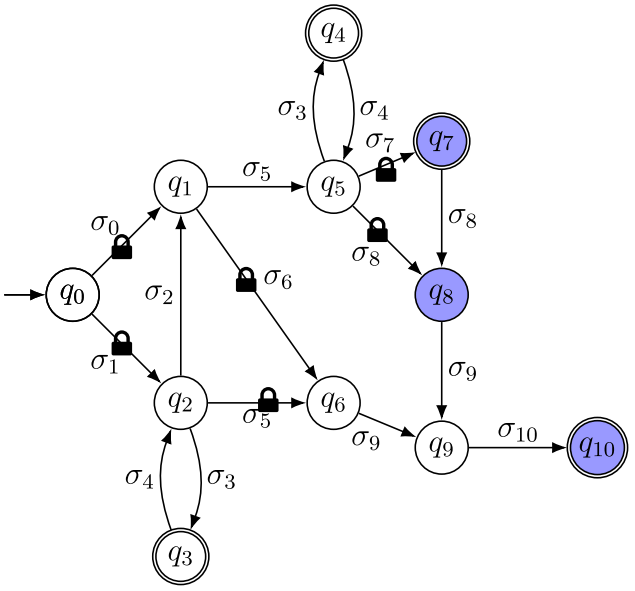

Fig. 6. Protection policy $\mathcal{P}$ for **G**.

Observe that based on this protection policy $\mathcal{P}$, every string from q_0 that can reach the secret states in Q_s has at least two protected events in $\Sigma(C_0) \cup \Sigma(C_1) \subseteq \Sigma_0 \cup \Sigma_1$. Thus the least number of protections $u = 2$ and the lowest security level $v = 0$ are satisfied; moreover, the minimum cost index is $i_{\min} = 1$.

For this example, the protections of each protected event specified by the policy $\mathcal{P}$ may be implemented as follows:

- σ_0, σ_1: setting up a password on each account of the regular user and the administrator.
- σ_5: setting up a password for launching the application.
- σ_6: setting up one-time password authentification.
- σ_7, σ_8: setting up fingerprint authentication.

5. Usability aware heterogeneous secret securing with minimum cost

In this section, we move on to address Problem 3.4 (UHSCP), in which the set of secret states Q_s is partitioned into $k(\geq 1)$ groups $Q_{s1}, \ldots, Q_{sk}$ with heterogeneous importance; as the index $j \in [1, k]$ increases, the importance of Q_{sj} rises. Similar to the preceding section, we begin with a characterization of the solvability of Problem 3.4, then present a solution algorithm, and finally use our running example to illustrate the results.

5.1. Solvability of UHSCP

The following theorem provides a necessary and sufficient condition under which there exists a solution to Problem 3.4.

Theorem 5.1. *Consider a system* **G** *in* (1)*, a set of secret states* $Q_s = \dot{\bigcup}_{j=1}^{k} Q_{sj}$*, the cost level sets* C_i $(i \in [0, n])$ *in* (4)–(6)*, the required least number of protections* $u \geq 1$*, and the required lowest security levels* $v_j \geq 0$ *for* Q_{sj} *such that* $v_1 \leq \cdots \leq v_k$*. Problem 3.4 is solvable (i.e. there exists a protection policy* $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$*,* $\tilde{\Sigma} \subseteq \Sigma_p$*, such that for every* $j \in [1, k]$ *the jth important secret state subset* Q_{sj} *is* $u - v_j$*−securely reachable w.r.t.* $\tilde{\Sigma}$ *and the highest index i of* C_i *that has nonempty intersection with* $\tilde{\Sigma}$ *is minimum) if and only if there exists* $i \in [v_1, n]$ *such that*

$$(\forall j \in [1, k])Q_{sj} \text{ is } u - v_j\text{−securely reachable w.r.t. } \tilde{\Sigma}_j = \bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1}$$

$$\& \tag{29}$$

$$(\exists j \in [1, k])Q_{sj} \text{ is not } u - v_j\text{−securely reachable w.r.t. } \tilde{\Sigma}_j = \bigcup_{l=v_j}^{i-1} \Sigma(C_l) \setminus \Sigma_{v_j-1}.$$

Condition (29) means that there exists an index $i \in [v_1, n]$ such that for every $j \in [1, k]$, the secret states in Q_{sj} can be protected with at least u protections using protectable events in $\bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1} \subseteq \Sigma_p^{\geq v_j}$, but there is $j \in [1, k]$ such that if only protectable events in $\bigcup_{l=v_j}^{i-1} \Sigma(C_l) \setminus \Sigma_{v_j-1} \subseteq \Sigma_p^{\geq v_j}$ are used, secrets cannot be protected with u protections. That these two conditions in (29) simultaneously hold indicates that the cost level index i is minimum.

Proof ($\Rightarrow$)**.** If condition (29) holds, then for every $j \in [1, k]$, the secret subset Q_{sj} is $u - v_j$−securely reachable w.r.t. $\bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1}$, and moreover the index i of C_i is minimum. The latter is because at least one secret subset $Q_{sj'}$ ($j' \in [1, k]$) is not $u - v_{j'}$−securely reachable w.r.t. $\bigcup_{l=v_{j'}}^{i-1} \Sigma(C_l) \setminus \Sigma_{v_{j'}-1}$ and $\bigcup_{l=v_{j'}}^{i-1} \Sigma(C_l) \setminus \Sigma_{v_{j'}-1} \subseteq \bigcup_{l=v_{j'}}^{i} \Sigma(C_l) \setminus \Sigma_{v_{j'}-1}$. In this case, for every Q_{sj} there exists a protection policy $\mathcal{P}_j : Q \to \mathrm{Pwr}(\bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1})$ such that protectable events in $\bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1}$ may be used to satisfy the required least number of protections u and the lowest security level v_j. These protection policies $\mathcal{P}_j$ ($j \in [1, k]$) together comprise a solution for Problem 3.3. Therefore, if (29) holds, then Problem 3.4 is solvable.

($\Leftarrow$) If Problem 3.4 is solvable with the minimum index of C_i being $i \in [v_1, n]$, then for every $j \in [1, k]$, Q_{sj} is $u - v_j$−securely reachable w.r.t. $\bigcup_{l=v_j}^{i} \Sigma(C_l) \setminus \Sigma_{v_j-1}$. Since the index i is minimum, it indicates that there exists at least one $j' \in [1, k]$ such that $Q_{sj'}$ is not $u - v_{j'}$−securely reachable w.r.t. $\bigcup_{l=v_{j'}}^{i-1} \Sigma(C_l) \setminus \Sigma_{v_{j'}-1}$. Therefore (29) holds. □

5.2. Policy computation for UHSCP

When Problem 3.4 is solvable under the condition presented in Theorem 5.1, we design an algorithm to compute a solution protection policy.

To compute such a protection policy, like in Section 4.2 we again convert the security problem to a corresponding control problem (call it UHCCP) by changing protectable events to controllable events. Then we employ Algorithm 1 to compute a control policy for each secret subset Q_{sj} ($j \in [1, k]$) to satisfy the required least number of protections u and the lowest security level v_j. This is done by inputting Algorithm 1 with $\mathbf{G}$ in (16), Q_{sj}, u and v_j.

If a solution exists, Algorithm 1 outputs u supervisors $\mathbf{S}_{0,j}, \ldots, \mathbf{S}_{u-1,j}$ and the minimum cost index $i_{\min,j}$. For these supervisors, one obtains the corresponding control policies $\mathcal{D}_{0,j}, \ldots, \mathcal{D}_{u-1,j}$, which may be combined into a single control policy

$$\mathcal{D}_j(q) = \bigcup_{l=1}^{u-1} \mathcal{D}_{l,j}(q), \quad q \in Q. \tag{30}$$

If the above holds for all $j \in [1, k]$, further combining all resulting $\mathcal{D}_j$ ($j \in [1, k]$) yields an overall control policy $\mathcal{D}$ as follows:

$$\mathcal{D}(q) = \bigcup_{j=1}^{k} \mathcal{D}_j(q), \quad q \in Q. \tag{31}$$

One the other hand, the overall minimum cost index $i_{\min}$ satisfies:

$$i_{\min} = \max(i_{\min,1}, \ldots, i_{\min,k}).$$

Algorithm 2: UHCC*u*.

Input: System $\mathbf{G}$ in (16), secret state set $Q_s = \dot{\bigcup}_{j=1}^{k} Q_{sj}$, protection number u, security levels $0 \leq v_1 \leq \cdots \leq v_k \leq n$.
Output: Control policy $\mathcal{D}$, minimum cost index $i_{\min}$

1: **for** $j = 1, \ldots, k$ **do**
2: $\quad \mathbf{S}_{0,j}, \ldots, \mathbf{S}_{u-1,j}, i_{\min,j} = \mathrm{UCC}u(\mathbf{G}, Q_{sj}, u, v_j)$
3: $\quad$ **if** all $\mathbf{S}_{0,j}, \ldots, \mathbf{S}_{u-1,j}$ are nonempty (or equivalently $i_{\min,j} \neq -1$) **then**
4: $\quad\quad$ Derive $\mathcal{D}_j$ from $\mathbf{S}_{0,j}, \ldots, \mathbf{S}_{u-1,j}$ as in (30)
5: $\quad$ **end if**
6: **end for**
7: **if** all $i_{\min,1}, \ldots, i_{\min,k}$ are not equal to -1 **then**
8: $\quad$ Derive $\mathcal{D}$ from $\mathcal{D}_1, \ldots, \mathcal{D}_k$ as in (31)
9: $\quad$ **return** $\mathcal{D}$ and $i_{\min} = \max(i_{\min,1}, \ldots, i_{\min,k})$
10: **end if**
11: **return** Empty control policy $\mathcal{D}$ and index -1

The above procedure is summarized in Algorithm 2 UHCC*u*. The time complexity of Algorithm 2 is k (from line 1 and k is the number of heterogeneous secret subsets) times that of Algorithm 1, namely $O(ku(n - v_1)|Q|^2)$. In fact, the k calls to

Algorithm 1 in line 2 can be done independently; hence the k executions of lines 2–5 may be implemented on multi-core processors in a distributed (thus more efficient) manner.

If Algorithm 2 successfully outputs a (nonempty) control policy $\mathcal{D}$, then we convert it to a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, by changing all controllable events back to protectable events. In terms of $\mathcal{P}$, we interpret disabled events by $\mathcal{D}$ as *protected events*.

Our main result in this section below asserts that the converted protection policy $\mathcal{P}$ is a solution for our original security problem UHSCP (Problem 3.4).

Theorem 5.2. *Consider a system* $\mathbf{G}$ *in* (1)*, a set of secret states* $Q_s = \dot{\bigcup}_{j=1}^{k} Q_{sj}$*, the cost level sets* C_i $(i \in [0, n])$ *in* (4)–(6)*, the required least number of protections* $u \geq 1$*, and the required lowest security levels* $v_j \geq 0$ *for* Q_{sj} *such that* $v_1 \leq \cdots \leq v_k$*. If Problem 3.4 is solvable, then the protection policy* $\mathcal{P}$ *derived from* $\mathcal{D}$ *in* (31) *(computed by Algorithm 2) is a solution.*

Proof. Suppose that Problem 3.4 is solvable. Then it follows from Theorem 5.1 that (29) holds, i.e. there is $i \in [v_1, n]$ such that the two conditions in (29) are satisfied.

Convert all protectable events to controllable events. The first condition in (29) ensures that Algorithm 2 passes the test in line 3 for all $j \in [1, k]$. Hence, k control policies $\mathcal{D}_j$ $(j \in [1, k])$ are obtained, each $\mathcal{D}_j$ ensuring that the secret subset Q_{sj} is protected by u protections, and the lowest security level of these protections is v_j. Again by the first condition in (29), Algorithm 2 passes the test in line 7 and a combined control policy $\mathcal{D}$ is obtained from $\mathcal{D}_j$ $(j \in [1, k])$. Converting all controllable events back to protectable events, we derive the corresponding protection policy $\mathcal{P}$ which ensures $u - v_j$—secure reachability of Q_{sj} for all $j \in [1, k]$.

Finally, since each index $i_{\min,j}$ $(j \in [1, k])$ is minimum for the respective call to UCC$u(\mathbf{G}, Q_{sj}, u, v_j)$ and $i_{\min} = \max_{j\in[1,k]} i_{\min,j}$, it follows from the second condition in (29) that $i_{\min}$ is the minimum cost index for the derived protection policy $\mathcal{P}$ as a solution for Problem 3.4. □

5.3. Running example

For illustration let us revisit Example 2.1. Consider the system $\mathbf{G}$ in Fig. 1, with the secret state set Q_s partitioned into two subsets: $Q_{s1} = \{q_7, q_8\}$ (regular users' secrets) and $Q_{s2} = \{q_{10}\}$ (administrator's secret). Accordingly, we require the lowest security levels to be $v_1 = 0$ and $v_2 = 1$, respectively. For the required number of protections, we let $u = 2$ (the same as Section 4.3).

In addition, the security level sets are Σ_i $(i \in [0, 3])$ as in (9), and the cost level sets are C_i $(i \in [0, 4])$ as in (10). We demonstrate how to use Algorithm 2 to compute a protection policy $\mathcal{P} : Q \to \mathrm{Pwr}(\tilde{\Sigma})$, $\tilde{\Sigma} \subseteq \Sigma_p$, and the minimum highest index i of C_i that has nonempty intersection with $\tilde{\Sigma}$ as a solution for Problem 3.4.

First, convert protectable events to controllable events and input Algorithm 2 with the converted $\mathbf{G}$, $Q_s = Q_{s1}\dot{\cup}Q_{s2}$, $u = 2$, $v_1 = 0$ and $v_2 = 1$.

For $j = 1$, call UCC$u(\mathbf{G}, Q_{s1}, u, v_1)$ to compute u (nonempty) supervisors $\mathbf{S}_{0,1}, \ldots, \mathbf{S}_{u-1,1}$ and the minimum cost index $i_{\min,1} = 1$. From these supervisors, we obtain the corresponding control policy $\mathcal{D}_1$ as in (30):

$$\mathcal{D}_1(q) = \begin{cases} \{\sigma_5\}, & \text{if } q = q_1 \\ \{\sigma_7, \sigma_8\}, & \text{if } q = q_5 \\ \emptyset, & \text{if } q \in Q \setminus \{q_1, q_5\} \end{cases}$$

Similarly for $j = 2$, call UCC$u(\mathbf{G}, Q_{s2}, u, v_2)$ to compute u (nonempty) supervisors $\mathbf{S}_{0,2}, \ldots, \mathbf{S}_{u-1,2}$ and the minimum cost index $i_{\min,2} = 3$. From these supervisors, we obtain the corresponding control policy $\mathcal{D}_2$ as in (30):

$$\mathcal{D}_2(q) = \begin{cases} \{\sigma_9\}, & \text{if } q = q_6 \\ \{\sigma_9\}, & \text{if } q = q_8 \\ \{\sigma_{10}\}, & \text{if } q = q_9 \\ \emptyset, & \text{if } q \in Q \setminus \{q_6, q_8, q_9\} \end{cases}$$

It is interesting to observe that due to the required lowest security level $v_2 = 1$, events in $\Sigma_0 = \{\sigma_0, \sigma_1, \sigma_5\}$ cannot be used (even though the event σ_1 at state q_0 belongs to $\Sigma(C_1)$). Consequently in this example, the events in the highest two security levels Σ_2, Σ_3 have to be used in order to meet this requirement.

Finally combining the above $\mathcal{D}_1$ and $\mathcal{D}_2$ yields an overall control policy $\mathcal{D}$ as in (31), which is shown in Fig. 7. Observe that every string from the initial state q_0 that can reach the secret states in $Q_{s1} = \{q_7, q_8\}$ has at least two disabled events in $\Sigma(C_0) \cup \Sigma(C_1) \subseteq \Sigma_0 \cup \Sigma_1$. Thus the least number of protections $u = 2$ and the lowest security level $v_1 = 0$ are satisfied. Moreover, every string from q_0 that can reach the secret state in $Q_{s2} = \{q_{10}\}$ has at least two disabled events in $(\Sigma(C_1) \cup \Sigma(C_2) \cup \Sigma(C_3)) \setminus \Sigma_0 \subseteq \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$. Thus the least number of protections $u = 2$ and the lowest security level $v_1 = 1$ are also satisfied.

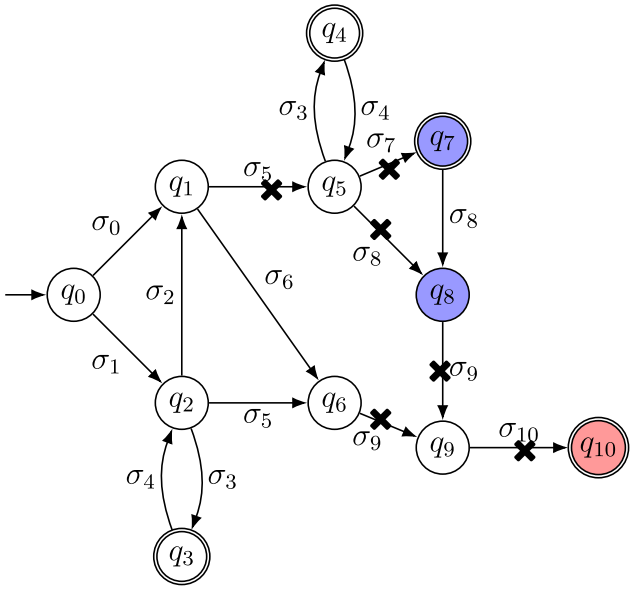

Fig. 7. Overall control policy $\mathcal{D}$ for **G** (with protectable events converted to controllable events).

Now changing all disabled transitions in Fig. 7 denoted by "✖" to "🔒", we obtain a protection policy $\mathcal{P}$ for the system **G** as follows:

$$\mathcal{P}(q) = \begin{cases} \{\sigma_5\}, & \text{if } q = q_1 \\ \{\sigma_7, \sigma_8\}, & \text{if } q = q_5 \\ \{\sigma_9\}, & \text{if } q = q_6 \\ \{\sigma_9\}, & \text{if } q = q_8 \\ \{\sigma_{10}\}, & \text{if } q = q_9 \\ \emptyset, & \text{if } q \in Q \setminus \{q_1, q_5, q_6, q_8, q_9\} \end{cases}.$$

Finally, the minimum cost index is $i_{\min} = \max(i_{\min,1}, i_{\min,2}) = 3$.

For this example, the protections of each protected event specified by the policy $\mathcal{P}$ may be implemented as follows:

- $\sigma_5, \sigma_7, \sigma_8$: already described at the end of Section 4.3.
- σ_9: setting up the first of two-factor authentification with a security question.
- σ_{10}: setting up the second of two-factor authentification with a physical security key.

6. Conclusions

We have studied a cybersecurity problem of protecting system's secrets with multiple protections and a required security level, while minimizing the associated cost due to implementation/maintenance of these protections as well as the affected system usability. Two usability-aware minimum cost secret protection problems have been formulated; the first one considers secrets of equal-importance, whereas the second considers heterogeneous secrets. In both cases, a necessary and sufficient condition that characterizes problem solvability has been derived and when the condition holds, a solution algorithm has been developed. Finally, we have demonstrated the effectiveness of our solutions with a running example.

In future work, we aim to extend the usability-aware secret protection problem to the setting of decentralized systems (which are typical in CPS), and develop efficient distributed protection policies (by leveraging existing decentralized/hierarchical approaches in the supervisory control theory). Other directions of extension from a broader perspective include generalizing the system model from deterministic purely-logical finite-state automaton with full observation to nondeterministic/probabilistic, timed, nonterminating, or partially-observed settings, and formulate/solve the usability-aware secret protection problem in those settings with different features.

CRediT authorship contribution statement

Shoma Matsui: Method developments, Proofs, Examples, Initial writing. **Kai Cai:** Idea developments, Technical checking, Writing quality improvements.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.P. Barrett, Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1, National Institute of Standards and Technology, 2018, http://dx.doi.org/10.6028/NIST.CSWP.04162018.
[2] C. Brooks, C. Grow, P. Craig, D. Short, Cybersecurity Essentials, John Wiley & Sons, 2018.
[3] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, ACM Comput. Surv. 42 (1) (2009) 1–31.
[4] A. Teixeira, D. Perez, H. Sandberg, K.H. Johansson, Attack models and scenarios for networked control systems, in: Proc. 1st International Conference on High Confidence Networked Systems, 2012, pp. 55–64.
[5] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, A survey of intrusion detection techniques in Cloud, J. Netw. Comput. Appl. 36 (1) (2013) 42–57.
[6] F. Pasqualetti, F. Dorfler, F. Bullo, Control-theoretic methods for cyberphysical security: geometric principles for optimal cross-layer resilient control systems, IEEE Control Syst. Mag. 35 (1) (2015) 110–127.
[7] F. Lin, Opacity of discrete event systems and its applications, Automatica 47 (3) (2011) 496–503.
[8] A. Saboori, C.N. Hadjicostis, Verification of K-step opacity and analysis of its complexity, IEEE Trans. Autom. Sci. Eng. 8 (3) (2011) 549–559.
[9] S. Lafortune, F. Lin, C. Hadjicostis, On the history of diagnosability and opacity in discrete event systems, Annu. Rev. Control 45 (2018) 257–266, http://dx.doi.org/10.1016/j.arcontrol.2018.04.002.
[10] Y. Tong, Z.W. Li, C. Seatzu, A. Giua, Verification of state-based opacity using Petri nets, IEEE Trans. Automat. Control 62 (6) (2017) 2823–2837.
[11] X. Yin, S. Li, Opacity of networked supervisory control systems over insecure multiple channel networks, in: Proc. 58th IEEE Conference on Decision and Control, 2019, pp. 7641–7646.
[12] Y. Xie, X. Yin, S. Li, Opacity enforcing supervisory control using non-deterministic supervisors, in: Proc. IFAC World Congress, 2020.
[13] H. Lan, Y. Tong, C. Seatzu, Verification of infinite-step opacity using labeled petri nets, in: Proc. IFAC World Congress, 2020.
[14] T. Moor, A discussion of fault-tolerant supervisory control in terms of formal languages, Annu. Rev. Control 41 (2016) 159–169.
[15] R. Fritz, P. Zhang, Overview of fault-tolerant control methods for discrete event systems, IFAC-PapersOnLine 51 (24) (2018) 88–95.
[16] L. Lin, Y. Zhu, R. Su, Towards bounded synthesis of resilient supervisors, in: Proc. 58th IEEE Conference on Decision and Control, 2019, pp. 7659–7664.
[17] N. Paape, J. van de Mortel-Fronczak, L. Swartjes, M. Reniers, Efficient failure-recovering supervisors, in: Proc. IFAC World Congress, 2020.
[18] J. Yao, X. Yin, S. Li, On attack mitigation in supervisory control systems: a tolerance control approach, in: Proc. 59th IEEE Conference on Decision and Control, 2020, pp. 4504–4510.
[19] L.K. Carvalho, Y.-C. Wu, R. Kwong, S. Lafortune, Detection and mitigation of classes of attacks in supervisory control systems, Automatica 97 (2018) 121–133.
[20] R. Fritz, 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.
[21] 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), 2019, pp. 1813–1818.
[22] C. Gao, C. Seatzu, Z. Li, A. Giua, Multiple attacks detection on discrete event systems, in: Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 2352–2357.
[23] R. Meira-Goes, C. Keroglou, S. Lafortune, Towards probabilistic intrusion deetection in supervisory control of discrete event systems, in: Proc. IFAC World Congress, 2020.
[24] S. Matsui, K. Cai, Secret securing with minimum cost, in: Proceedings of the 61st Japan Joint Automatic Control Conference, 2018, pp. 1017–1024.
[25] S. Matsui, K. Cai, Secret securing with multiple protections and minimum costs, in: Proc. the 58th IEEE Conference on Decision and Control, 2019, pp. 7635–7640.
[26] Z. Ma, K. Cai, Optimal secret protections in discrete-event systems, IEEE Trans. Automat. Control (2021) http://dx.doi.org/10.1109/TAC.2021.3091438.
[27] S. Matsui, K. Cai, Application of supervisory control to secret protection in discrete-event systems, J. Soc. Instrument Control Eng. 60 (1) (2021) 14–20.
[28] B. Wu, H. Lin, Privacy verification and enforcement via belief abstraction, IEEE Control Syst. Lett. 2 (4) (2018) 815–820.
[29] Y. Ji, X. Yin, S. Lafortune, Enforcing opacity by insertion functions under multiple energy constraints, Automatica 108 (2019) 108476.
[30] W.M. Wonham, K. Cai, Supervisory Control of Discrete-Event Systems, Springer, 2019.
[31] C.G. Cassandras, S. Lafortune, Introduction To Discrete Event Systems, Springer, 2008.
[32] K. Cai, W.M. Wonham, Supervisory control of discrete-event systems, in: Encyclopedia of Systems and Control, second ed., Springer, 2020, http://dx.doi.org/10.1007/978-1-4471-5102-9_54-2.
[33] W.M. Wonham, K. Cai, K. Rudie, Supervisory control of discrete-event systems: a brief history, Annu. Rev. Control 45 (2018) 250–256.