Secret Protections with Costs and Disruptiveness in Discrete-Event Systems Using Centralities

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

***Abstract*—In this paper, we study a secret protection problem in discrete-event systems. The system is modeled by an automaton in which several states are assigned with different secret levels. Our aim is to protect some of the events in the system such that any sequence yielding a secret state contains a number of protected events no less than the required security level. We first study the secret protecting problem with minimum cost, i.e., to design a protecting policy whose cost is minimal. We prove that the decision version of such a secret protection problem is NP-hard, which implies that there unlikely exists a polynomial algorithm to solve it. As a result, we developed a heuristic method to obtain a locally optimal solution to protect the secrets using the notion of *cost-weighted centrality*. Then, we consider the disruptiveness, i.e., the degree of disruptiveness of protecting policies to legal users' normal operations. We formulate the disruptiveness to users incurred by the protection on events as a penalty function which describes the impact of events and transitions on the paths leading to marker states. A heuristic method based on the notion of *cost-penalty-weighted centrality* is analogously developed to obtain a protecting policy which can well balance the cost and the disruptiveness to users.**

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

I. Introduction

Information security in cyber-physical systems has drawn much attention in the past three decades [1], [2], [3], [4], [5]. One of the main concerns in cyber-security is to ensure that the *secrets* in a system should not be accessed by unauthorized users, i.e., the *intruders*. To protect the secrets, it is necessary to set some protective measures on some actions in the system to verify the identity of users. 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. By doing so, an unauthorized intruder who cannot provide a legal identity must hack through these protected actions (i.e., *events*) before reaching the secrets. If the effort needed for hacking through these protected events is high enough, an attack towards the secrets can be considered practically prevented.

Theoretically, the administrator of a system may protect as many events as possible to ensure a security requirement. However, such a protecting policy is usually infeasible in real systems due to the following two reasons. First, protecting an event usually incurs some *cost*, e.g., to develop and deploy encrypting algorithms, or to purchase fingerprint detectors. Hence, the cost to protect all events may be too high to be affordable. Second, such protections on events usually causes some inconvenience to legal users. As a result, protecting too many events in a system (e.g., requesting a password verification for each click) may greatly degrade the experience of legal users who use the system. In practice, we are interested in designing protecting policies that satisfy the security requirement and have as little cost and disruptiveness as possible.

The problem of secret protection in discrete-event systems was first studied in [6], [7], [8], [9], and [10]. In [6], [7], the set of protectable events is partitioned into distinct *levels*, and the total cost is defined as the *maximum* of the levels of protected events. The work of [8] is a generalization of [6], [7] which takes the users' convenience into consideration: the users' convenience is treated as a one-level increase on the cost level. In the work of [10], the cost to protect each event can be an arbitrary nonnegative real value, and the total cost is defined as the sum of the costs of all protected events. Then, a polynomial algorithm is developed to find an optimal protecting policy based on a structure called *secret automaton*. Besides, [10] also considers a criteria of optimality on disruptiveness that is not to protect any event unless it has to. In [10] an assumption is introduced such that each transition is protected separately, which means that all transitions in a system are distinctly labeled.

Based on the motivation above, in this paper we further explore the secret protection problem with three important generalizations of [10] which will be explained in the sequel. In a system modeled by an automaton, some states are considered as *secrets*. Each secret state is associated with a *security level* which is the minimal number of protected events required before reaching the state (from the initial state). Moreover, in the system, a subset of events can be protected (i.e., a protectable operation such as password checks can be imposed on it by the administrator). By successfully executing a protected event, a user is granted with a certain *clearance level*. On the other hand, protecting an event incurs a nonnegative *cost*, and causes *disruptiveness* to legal users. A *secret protection problem* (SPP) is to design an event-protecting policy to enforce such a security requirement such that every sequence from the initial state and reaching a secret state contains a number of protected events such that the total granted clearance level is no less than the required security level of that secret.

The first generalization done in this work is that we consider the case in which the cost of protection is defined on the *events* (i.e., the *labels*) in the system, by which we remove a potentially restrictive assumption in [10] which requires that

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62373313, Shaanxi Provincial Natural Science Foundation under Grant No. 2022JM-323, JSPS KAKENHI Grant nos. 21H04875 and 22KK0155, the Fundamental Research Funds for the Central Universities under Grant No. ZYTS23023.

Z. Ma and J. Jiang are with School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China (email: maziyue@xidian.edu.cn, jiangjiagang@stu.xidian.edu.cn)

K. Cai is with Department of Core Informatics, Osaka Metropolitan University, Osaka 558-8585, Japan (e-mail: cai@omu.ac.jp).

all transitions are distinctly labeled. In other words, while transitions and events are isomorphic in [10], they are not in this work, and several transitions sharing the same event/label can be simultaneously protected by paying the corresponding cost only once. This setting is fairly common in practice. For example, to purchase and install a fingerprint detector is costly, but to use it after installation one just need to put his/her finger on the scanner whose cost is negligible. The second generalization is that we allow that the execution of a protected event to raise multiple clearance levels. For example, a user who passes a face recognition check is more trustable than the one who passes a password check, although they both pass only one security check. The case in [10] can be viewed as a particular case in which the raise of the clearance level is unitary when passing any protected events. In this new setting, we will show that the SPP is NP-hard (more challenging than the P-class SPP in [10] with a restrictive assumption). We tackle this new problem by proposing a novel approach borrowing a *centrality* concept [11] from network science [12].

The third generalization is that we consider the disruption on the normal usage of the system by legal users. Besides the secret states, a system typically contains some *marker states* that represent the featured functions of the system. Hence, a trajectory from the initial state to a marker state can be viewed as a normal use of the system by a legal user. This formulation of disruption is different from that in [8] where the disruption is treated as a binary variable (i.e., either "disrupted" or "undisrupted") so that the disruption, if it exists, is modeled as a "one level-up" of the cost. Moreover, note that in practice using featured functions (e.g., streaming a video clip) and checking confidential information (e.g., modifying the account's password) in a system are typically separated. Hence, we do not impose any special relation between marker states and secret states. The protection on an event that is on a trajectory of normal use incurs disruption to legal users, which can be viewed as a *penalty* of protection. Although it is expected that such disruptions should be minimized, this objective is usually incompatible with that of minimizing the cost of protection. Hence, the balance between the cost and the disruptiveness must be considered.

The main contributions of this work are summarized as follows.

- We first consider the SPP with cost optimality without specifying the marker states. It has been proved that SPP in distinctly labeled finite state automata and unitary raise of clearance level belongs to complexity class P in [10]. Here, with the removal of the distinctly labeling and unitary raise of clearance level assumptions, we prove that the decision version of such SPP is NP-complete by proving that (i) SPP is reducible from the Knapsack Problem, which implies its NP-hardness, and (ii) a candidate solution of SPP can be verified in polynomial time, which implies the NP-membership of SPP. As a result, the problem of determining an optimal event-protecting policy is NP-hard, which implies that there unlike exists an algorithm with polynomial complexity to solve it.
- Then, we develop a heuristic method that is of polynomial complexity to compute a locally optimal solution for SPP. Our method is established based on the notion of *centrality* [11], a widely used concept in the field of social networks which characterizes the importance of the arcs in a network. By considering the influence of cost, we propose a new notion called *cost-weighted centrality* which is then used to develop our heuristic algorithm.
- Finally, we consider a more general version of SPP with both secret states and marker states. We formulate a penalty function that characterizes the impact of events and transitions on the paths leading to marker states, i.e., the disruptiveness. Then, a notion called *cost-penalty-weighted centrality* is proposed with a parameter λ to balance the weight between the protection cost and the disruptiveness. For this general SPP, the centrality-based method is adapted to obtain a locally optimal policy which balances the cost and disruptiveness.

Furthermore, a series of simulation is carried out which shows that our method is practically efficient and can obtain a locally optimal solution which is close to the global optimal one in most cases.

It is worth noting that in the literature, there is a related notion in cyber-security called *opacity* [13], [14], [15], [16], [17] (also called *secrecy, anonymity*) that characterizes whether a secret concerning systems' behaviors can be hidden from an intruder who passively observes (some of) the events generated by the system. We point out that opacity is essentially different from the secret protecting problem studied in this work. Opacity is defined from the side of an intruder who has partial observability of a system, while the SPP is defined from the side of the administrator to whom the system is fully observable. In autonomous systems with eavesdropping intruders, for a non-opaque system, it is common to develop security supervisors [18], [19], [20] or to use *output modification* [21], [22] to ensure opacity or other properties of information security. Here, our method are suitable in systems that are operated by the input of the users. Since we require that the system be usable to legal users, we cannot change the dynamics and preferably not to modify the output of a system. For example, we cannot disable an event leading to a secret state since otherwise some normal functions of the system may become unavailable. Other related perspective of research include *intrusion detection* [23], [24] and *attack-resilience* [25], [26], [27]. Since we assume that an intruder may disguise as a legal user such that no abnormal behavior can be observed, we do not try to detect the existence of intruders. Instead, we set protective measures on the events in the system to increase the difficulty of unauthorized access to the secrets, which may practically prevent potential intruders.

Some preliminary results related to the approach that we develop in this paper were presented in [28]. In [28] only the minimization of cost is considered. In this paper the NP-completeness of the SPP is formally proved. Furthermore, in this work a general version of SPP with both secret states and marker states is studied, and a centrality-based method is developed accordingly.

The rest of this paper is organized in seven sections. Basic notions of nondeterministic finite automata are recalled in Section II. In Section III, multiple secret protecting problem

with minimum costs is formulated. Section IV proves the NP-hardness of the optimal secret protecting problem. In Section V, a heuristic method is developed which generates a local optimal solution, and two illustrative examples are provided. In Section VI, the penalty function to evaluate the disruptiveness is formulated, and the centrality-based method is adapted to obtain a locally optimal policy which balances the cost and disruptiveness. Section VII concludes the paper.

II. Preliminaries

A *nondeterministic finite automaton*[1] (*automaton* for short) is a five-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a finite set of *states*; Σ is a finite set of *events*; $\delta : Q \times \Sigma \to 2^Q$ is the transition function[2]; $q_0 \in Q$ is the initial state; and $Q_m \subseteq Q$ is the set of *marker states*. An automaton G is also denoted as (Q, Σ, δ, q_0) if the marker states are not specified. 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).

The transition function δ is extended to $\delta^* : 2^Q \times \Sigma^* \to 2^Q$ by recursively defining (i) $\delta^*(Q', \varepsilon) = Q'$, (ii) $\delta^*(Q', \sigma) = \bigcup_{q \in Q'} \delta(q, \sigma)$, (iii) $\delta^*(Q', s\sigma) = \delta^*(\delta^*(Q', s), \sigma)$, where $Q' \subseteq Q$, $s \in \Sigma^*$, and $\sigma \in \Sigma$. The *language* of G is defined as $L(G) = \{s \in \Sigma^* \mid \delta^*(\{q_0\}, s) \in Q\}$.

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

Given a sequence $s \in \Sigma^*$, we use $|s|$ to denote the *length* of s, i.e., the total number of events in s. 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 $\overline{L} = \{s \in \Sigma^* \mid \exists s' \in \Sigma^*, ss' \in L\}$.

III. Secret Protection with Minimal Costs

A. Secret Protecting Problem Formulation

In this section, we recall some preliminary notions in [10]. Given a system $G = (Q, \Sigma, \delta, q_0)$ that models a physical system, some states in it may contain sensitive data (such as credit card numbers) that should be kept secret. To protect the secrets from being accessed by unauthorized intruders, some events in the system should be protected (e.g., by adding security checks on them) such that any user who accesses the plant from the entry of the system must passes through a certain number of security checks before reaching a secret state. Formally, 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. Note that in this section and Sections IV, V, we do not consider marker states Q_m which represents featured functions of the system: we will consider Q_m in Section VI where the measure of disruptiveness is involved.

We assume that in the system $G = (Q, \Sigma, \delta, q_0)$ some events are *protectable*. Precisely speaking, 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., $\Sigma = \Sigma_p \cup \Sigma_{up}$.

Definition 1: [10] A protecting policy is a function:

$$\vartheta : L(G) \to 2^{\Sigma_p} \tag{1}$$

such that, after a sequence $s \in L(G)$ is generated, the set of protected events following s is $\vartheta(s) \subseteq \Sigma_p$.[3] ♢

The execution of a protected event σ (meaning that the user passes the corresponding security check on σ) grants the user a clearance level $\gamma(\sigma)$. Formally, γ is a *clearance function* that assigns each event in Σ_p a nonnegative integer, i.e., $\gamma : \Sigma_p \to \mathbb{N}$. Given a protecting policy ϑ, we define $\theta(s, \sigma) = 1$ (resp., $\theta(s, \sigma) = 0$) if $\sigma \in \vartheta(s)$ (resp., $\sigma \notin \vartheta(s)$). To ensure that the secret states are not visited by unauthorized intruders, our objective is to design a policy that ensures that any sequence yielding a secret state must contain a sufficient number of protected events such that the total granted clearance level by executing the sequence is no fewer than the corresponding security level $\ell(q)$.

Definition 2: [10] Given a system $G = (Q, \Sigma, \delta, q_0)$ 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:

$$\sum_{j=0}^{n-1} \theta(\bar{s}_j, \sigma_{j+1}) \cdot \gamma(\sigma) \geq \ell(q). \tag{2}$$

♢

On the other hand, in practice, to protect events usually incurs cost due to the deployment of new encrypting algorithms or purchasing of expensive biometric detectors. Hence, we consider a cost function $\mathbf{c} : \Sigma \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ that assigns each event a nonnegative real number or ∞, i.e.:

$$\begin{cases} \mathbf{c}(\sigma) \in \mathbb{R}_{\geq 0} \setminus \{\infty\}, & \sigma \in \Sigma_p \\ \mathbf{c}(\sigma) = \infty, & \sigma \in \Sigma_{up} \end{cases} \tag{3}$$

In plain words, the cost of protecting a protectable event in Σ_p is a finite number while that of protecting an unprotectable event in Σ_{up} is ∞. The cost of a protecting policy is thus defined as the following.

[1]Preliminary notions on finite automaton used in this paper are derived from [29].

[2]The nondeterministic finite automaton considered in this paper does not include silent transitions, i.e., each transition in it is assigned with a label in Σ. On the other hand, it allows that from one state two or more outbound arcs are assigned with the same label.

[3]Policy ϑ may protect an event $\sigma \in \Sigma_p$ after s where $s\sigma$ is not in $L(G)$. In such a case, whether protecting σ or not after s does not affect the result of the information security.

Definition 3: [10] Given a system $G = (Q, \Sigma, \delta, q_0)$, a security requirement $\ell : Q \to \mathbb{N}$, and a cost function defined by (3). The cost of the protecting policy ϑ is defined as:

$$C(\vartheta) = \sum_{\sigma \in P(\vartheta)} \mathbf{c}(\sigma) \tag{4}$$

where $P(\vartheta) = \{\sigma \in \Sigma_p \mid (\exists s \in L(G))\ \sigma \in \vartheta(s)\}$ is the set of protected events of ϑ. $\diamondsuit$

Definition 4: [10] A protecting policy ϑ is *optimal* if there does not exist any other protecting policy ϑ' such that $C(\vartheta') < C(\vartheta)$. $\diamondsuit$

The cost $\mathbf{c}(\sigma)$ of event σ is counted (as part of the cost of ϑ) if σ is protected after at least one sequence $s \in L(G)$. Note that such a cost incurred by the protection of event σ is irrelevant to the number (when it is not zero) of sequences after 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. The problem studied in this paper is then formulated as the following.

Problem 1: [Secret Protecting Problem] Given a system $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_p \cup \Sigma_{up}$, a security requirement ℓ, a *clearance function* γ, and a cost function $\mathbf{c}$, determine an optimal protecting policy ϑ. $\diamondsuit$

Remark 1: Different from the work in [10] in which all transitions are assumed to be distinctly labeled (Assumption 1 in [10]), in this paper we remove this possibly restrictive assumption and allow that one event (or label) $\sigma \in \Sigma$ be assigned to several transitions in the system.

We emphasize that, when modeling a physical system as an automaton, the way to assign the events/labels to transitions depends on the physical means of protection. Precisely speaking, a group of transitions will be assigned with the same event/label if and only if these transitions can be protected simultaneously. For example, all transitions that can be equipped with the same password check program are assigned with the same event/label. If the corresponding event is protected (i.e., the program is developed and deployed), all such transitions are protected simultaneously. On the other hand, two transitions that cannot be protected simultaneously by a single means will be assigned with different labels. For example, two transitions that represent two secrets at distant locations whose protections require the deploy of biometric detectors will be assigned with different labels, since the administrator cannot buy one detector and use it at two distant locations. $\diamondsuit$

B. Protecting Policy Conversion

In the sequel, we perform a protecting policy conversion such that the secret protection problem can be solved by exploring the so-called *maximal protecting policies*.

Definition 5: A protecting policy $\vartheta : L(G) \to 2^{\Sigma_p}$ is called *maximal* if the following condition holds:

$$(\exists s \in L(G))\ \sigma \in \vartheta(s) \Leftrightarrow (\forall s \in L(G))\ \sigma \in \vartheta(s). \tag{5}$$

A maximal protecting policy is denoted as ϑ_m. $\diamondsuit$

A maximal protecting policy is that if it protects an event σ after some sequence $s \in L(G)$, then it protects σ after all sequences. In other words, each event is either protected for all $s \in L(G)$ or not protected for any $s \in L(G)$. Then, for any protecting policy ϑ defined by (1), we can always define a protecting policy ϑ_m as the following.

$$(\exists s \in L(G))\ \sigma \in \vartheta(s) \Leftrightarrow (\forall s \in L(G))\ \sigma \in \vartheta_m(s). \tag{6}$$

The following proposition shows that ϑ_m is valid if ϑ is valid, and ϑ_m has the same cost as ϑ.

Proposition 1: Given a system $G = (Q, \Sigma, \delta, q_0)$, a security requirement $\ell : Q \to \mathbb{N}$, a *clearance function* γ, and a cost function defined by (3). If a protecting policy ϑ is valid, then (i) the corresponding maximal protecting policy ϑ_m is valid; (ii) $C(\vartheta) = C(\vartheta_m)$.

Proof: First, for any sequence $s \in L(G)$, $\vartheta(s) \subseteq \vartheta_m(s)$. This indicates that for any sequence $s = \sigma_1 \cdots \sigma_n \in L(G)$ such that $q \in \delta^*(\{q_0\}, s)$, $\sum_{j=0}^{n-1} \theta_m(\bar{s}_j, \sigma_{j+1}) \cdot \gamma(\sigma_{j+1}) \geq \sum_{j=0}^{n-1} \theta(\bar{s}_j, \sigma_{j+1}) \cdot \gamma(\sigma_{j+1}) \geq \ell(q)$ holds. Hence, ϑ_m is valid. On the other hand, $C(\vartheta) = C(\vartheta_m)$ holds since $P(\vartheta) = P(\vartheta_m)$ according to (6). $\square$

Proposition 1 shows that for any optimal protecting policy, its corresponding maximal protecting policy ϑ_m is also optimal. Notice that in a maximal protecting policy an event σ is either protected or not protected in all cases. Thus, a maximal protecting policy ϑ_m can be represented by a set $\mathcal{P} \subseteq \Sigma_p$ that is the set of all events it protects:

$$\mathcal{P} = \{\sigma \in \Sigma_p \mid \sigma \in P(\vartheta_m)\}. \tag{7}$$

Hence, to solve the secret protection problem, we can design a maximal protecting policy ϑ_m instead of seeking (possibly non-)maximal ones since the former can be simply described by a subset of Σ_p. In the sequel, we simply use $\mathcal{P}$ to denote a maximal protecting policy.

We use $\varphi(s, \sigma)$, where s is a string in Σ^* and σ is an event in Σ, to denote the number of occurrences of event σ in string s. Given a maximal protecting policy $\mathcal{P}$ and a sequence $s \in L(G)$, the clearance level after executing s is $\sum_{\sigma \in \mathcal{P}} \varphi(s, \sigma) \cdot \gamma(\sigma)$. The validity and the optimality of a protecting policy (Definitions 2 and 4) can thus be rewritten as follows.

Definition 6: A protecting policy $\mathcal{P}$ is valid if for any sequence $s \in L(G)$ such that $q \in \delta^*(\{q_0\}, s)$, $\sum_{\sigma \in \mathcal{P}} \varphi(s, \sigma) \cdot \gamma(\sigma) \geq \ell(q)$ holds. $\diamondsuit$

Definition 7: A protecting policy $\mathcal{P}$ is *optimal* if there does not exist any other protecting policy $\mathcal{P}'$ such that $\sum_{\sigma \in \mathcal{P}'} \mathbf{c}(\sigma) < \sum_{\sigma \in \mathcal{P}} \mathbf{c}(\sigma)$. $\diamondsuit$

Next, we use the following example (augmented from [6]) to illustrate the secret protecting problem that will be studied in this paper.

Example 1: Consider the automaton G in Figure 1 which represents a computer network. A user who visits the system is first initialized at the initial state q_0. State q_1 means that the user has accessed a wireless router. The user can connect to or disconnect from the router via σ_1 or σ_2, respectively. States q_2 and q_4 represent two different LANs. From the state q_1, the user can switch between states q_1 and q_2, or between q_1 and q_4, via two cyclic transitions. From both states q_2 and

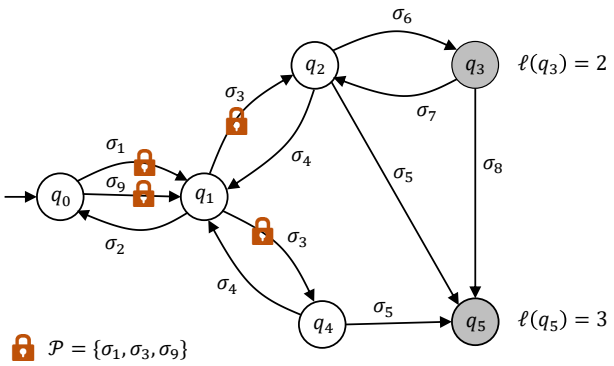


Fig. 1: The system G

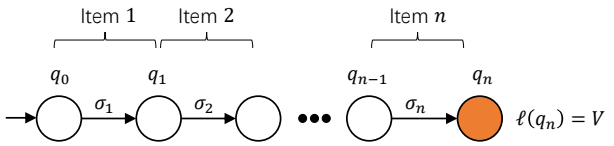


Fig. 2: The automaton G constructed in the proof of Proposition 2.

q_4, the user can directly reach the state q_5 to access secret files, such as credit card numbers. From state q_2 the user can switch between states q_2 and q_3, where q_3 is a bastion server which allows users to access to q_5 via a different channel. We consider a security requirement ℓ as $\ell(q_3) = 2$, $\ell(q_5) = 3$ and $\ell(q_i) = 0$ where $i \in \{0, 1, 2, 4\}$. That is, the security level of state q_3 (resp., state q_5) is two (resp., three).

Notice that the process to connect to LAN q_2 and q_4 from q_1 is similar. Hence, we can develop a password verification module and deploy it on transitions $q_1 \to q_2$ and $q_1 \to q_4$. The development of the module may be costly, while the cost of deploying it is comparatively negligible (since we just make another copy of the module). Hence, in the system the two transitions $q_1 \to q_2$ and $q_1 \to q_4$ are assigned with the same event σ_3. That is, when imposing a protection on event σ_3, both transitions are protected without doubling the cost. Similarly, σ_4 and σ_5 are also assigned to two transitions, respectively, since σ_4 means to disconnect from the LANs while σ_5 is to access the secret file q_5 from LANs.

For simplicity, assume that all events are protectable while the cost to protect each event is unitary. Namely, $\Sigma = \Sigma_p$ and $\mathbf{c}(\sigma_i) = 1$ where $i \in \{1, 2, \ldots, 9\}$. On the other hand, the channels between server q_1 and q_2, q_4 are robust and difficult to be penetrated. Hence, a successfully execution of a protected σ_3 grants 2 clearance levels to the user while a successfully execution of other protected events grants 1 clearance level, i.e., $\gamma(\sigma_3) = 2$ and $\gamma(\sigma_i) = 1$ for $i \neq 3$. In such a case, one can readily verify that an optimal protecting policy is $\mathcal{P} = \{\sigma_1, \sigma_3, \sigma_9\}$ (marked with locks in Figure 4) whose total cost is 3 which ensures that any user who reaches secret states q_3 and q_5 must get 2 and 3 clearance levels, respectively. $\diamondsuit$

IV. Complexity Analysis of Determining Protecting Policies with Minimal Costs

A. NP-hardness

In [10], by assuming that each transition is assigned with a unique event and each event only raises the clearance level by one, the problem of optimal secret protection can be solved in polynomial time. In this paper, we remove the two assumptions: by doing so the problem studied in this work is more general than that in [10]. In this section, we show that such a generalization makes the secret protection problem NP-hard, which implies that there unlikely exists a polynomial algorithm to obtain an optimal solution.

In the following, we prove that the decision version of the secret protection problem is *NP-Complete*, which implies that the global optimization version of the problem is *NP-Hard*. To see this, we reduce the *0-1 Knapsack problem* (decision version) that is known to be NP-Complete to the optimal secret protection problem (decision version).

Problem 2: [0-1 Knapsack Problem] Given a set of n items each of which with a weight $w_i \in \mathbb{N}$ (a set of natural numbers) and a value $v_i \in \mathbb{N}$, and a knapsack with a maximal weight limit $W \in \mathbb{N}$, determine if a total value equal to or greater than a given threshold $V \in \mathbb{N}$ can be achieved without exceeding the weight limit W.

Problem 3: [Secret Protecting Problem: Decision Version] Given a system $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma_p = \Sigma$, a secret state q_s with $\ell(q_s) = V$, a cost function $\mathbf{c} : \Sigma \to \mathbb{N}$, and a budget limit $W \in \mathbb{N}$, determine if there exists a maximal protecting policy $\mathcal{P}$ such that $C(\mathcal{P}) \leq W$.

Proposition 2: Problem 2 is polynomially reducible to Problem 3.

Proof: Given an instantiation of Knapsack Problem 1 $(n, \mathbf{w}, \mathbf{v}, V, W)$ where $\mathbf{w} = [w_1 \cdots w_n]$ and $\mathbf{v} = [v_1 \cdots v_n]$, we construct an automaton $G = (Q, \Sigma, \delta, q_0)$ by the following procedure:

1) let $Q = \{q_0\}$ where q_0 is the initial state;
2) for each i in $1, 2, \ldots, n$, add a state q_i to Q;
3) for each state pair (q_{i-1}, q_i) where i in $1, \ldots, n$, define $\delta(q_{i-1}, \sigma_i) = q_i$, i.e. add a transition labeled by σ_i from q_{i-1} to q_i, respectively.

Such a construction is illustrated in Figure 2. Eventually, automaton G has $n + 1$ states and n transitions, which is polynomial with respect to the size of the Knapsack problem. We then define:

- security requirement $\ell(q_n) = V$ and $\ell(q_{i,j}) = 0$ for all $q_{i,j} \neq q_n$;
- $\Sigma_p = \Sigma$, i.e. all events are protectable;
- cost function $\mathbf{c}(\sigma_i) = v_i$ and clearance function $\gamma(\sigma_i) = w_i$ for all $\sigma_i \in \Sigma$.

Then, the answer of the instantiation $(n, \mathbf{w}, \mathbf{v}, V, W)$ of the Knapsack Problem 2 is positive if and only if in G there exists a protecting policy $\mathcal{P} \subseteq \Sigma_p$ such that the following two statements hold:

1) for any sequence $s \in \Sigma^*$ such that $q_n \in \delta^*(\{q_0\}, s)$, $\sum_{\sigma \in \mathcal{P}} \varphi(s, \sigma) \cdot \gamma(\sigma) \geq V$ holds (i.e. s must go through the events in $\mathcal{P}$ such that the total increase of the

clearance level is no less than V, which means that $\mathcal{P}$ is valid);

2) $\sum_{\sigma\in\mathcal{P}} \mathbf{c}(\sigma) \leq W$ (i.e. the cost of $\mathcal{P}$ is less than or equal to W).

Therefore, Problem 2 is polynomially reducible to Problem 3. □

On the other hand, we prove that Problem 2 is in NP by showing that a protecting policy candidate can be validated in polynomial time. Given a system $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_p \cup \Sigma_{up}$, a *clearance function* γ, a security requirement ℓ, and a protecting policy $\mathcal{P} \subseteq \Sigma_p$, let $\mathcal{G}_\mathcal{P} = (Q, T, \mathbf{w})$ be the weighted underlying digraph of G where Q is the set of vertices, T is the set of directed arcs, and $\mathbf{w}$ is the weight function that assigns each transition $t = (q, \sigma, q')$ in T a weight:

$$\mathbf{w}(t) = \begin{cases} \gamma(\sigma), & \sigma \in \mathcal{P} \\ 0, & \sigma \notin \mathcal{P} \end{cases}$$

Clearly, $\mathcal{G}_\mathcal{P}$ can be constructed in polynomial complexity. The following proposition shows that whether $\mathcal{P}$ is valid can be determined by solving the *shortest path problem* (which belongs to class P [30]) in $\mathcal{G}_\mathcal{P}$. Since one shortest path can be solved in $O(|Q|^2)$ [30], the complexity of determining the validity of a protecting policy $\mathcal{P}$ is $O(|Q|^3)$.

Proposition 3: A protecting policy $\mathcal{P} \subseteq \Sigma_p$ is valid if and only if for all $q_s \in Q_s$, the weight of a shortest path from q_0 to q_s in $\mathcal{G}_\mathcal{P}$ is equal to or greater than $\ell(q)$.

Proof: (Only if) By contrapositive. Suppose that there exists a shortest path from q_0 to q_s in $\mathcal{G}_\mathcal{P}$ whose weight is less than $\ell(q)$. Then the sum of the raised clearance level on the trajectory from q_0 to q_s in G corresponding to such a path is less than $\ell(q)$, which indicates that $\mathcal{P}$ is not valid.

(If) Suppose that $\mathcal{P}$ is not valid, i.e., there exists a sequence $s \in \Sigma^*$ such that $\delta^*(\{q_0\}, s) = q$, $\sum_{\sigma\in\mathcal{P}} \varphi(s,\sigma)\cdot\gamma(\sigma) \geq \ell(q)$. From s we can extract a sequence s' such that $\delta^*(\{q_0\}, s') = q$ and the corresponding trajectory does not visit any state more than once. Since s' contains no more events than s (in the sense of set containment), $\sum_{\sigma\in\mathcal{P}} \varphi(s',\sigma) \cdot \gamma(\sigma) \leq \sum_{\sigma\in\mathcal{P}} \varphi(s,\sigma) \cdot \gamma(\sigma) < \ell(q)$ holds, which indicates that the weight of the corresponding shortest path in $\mathcal{G}_\mathcal{P}$ is less than $\ell(q)$. □

Theorem 1: Problem 3 is NP-Complete.

Proof: Straightforward by Propositions 2 and 3. □

Since Problem 3 is NP-Complete, its optimization version (Problem 1, i.e., to determine an optimal protecting policy) is NP-hard, which implies that a corresponding algorithm with polynomial complexity unlikely exists.

One may notice that the NP-hardness of Problem 1 arises from the value of W in Problem 3, and W comes from the bound of protecting cost and may not be very large in practice. Hence, we conjecture that there may exist some pseudo-polynomial-time algorithm or exact algorithm whose complexity is polynomial in the number of states in the system. However, this problem is quite challenging, requires further investigation which is beyond the scope of this work, and hence remains open at this moment. In particular, we point out that the method developed in our previous work [10] (whose complexity is $O(n^2)$ in the number of states in G) cannot be used to achieve an optimal solution for Problem 1, since the min-cut approach requires all arcs be dependent. Overall, the complexity class of weak NP-hard is a theoretical lower bound of computational complexity of the SPP Problem 1. As a result, in the next section we develop a heuristic algorithm to design a locally optimal protecting policy.

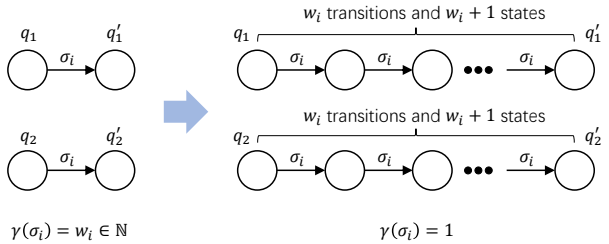


Fig. 3: Construction in the proof of Theorem 2.

Definition 8: A protecting policy $\mathcal{P}$ is *locally optimal* if there does not exist any other protecting policy $\mathcal{P}'$ such that $\mathcal{P}' \subsetneq \mathcal{P}$. ♢

B. Modeling Power of SPPs with Uniform/Non-uniform Clearance Levels

So far, we focus on Problem 1 in which the clearance function γ is arbitrary, possibly non-uniform. This naturally leads to the question: what if in Problem 1 the clearance function γ is uniform (as was considered in [10])? In this subsection, we show that the introducing the clearance function γ does not increase the modeling power of the problem, or conversely, restricting $\gamma(\sigma) = 1$ for all $\sigma \in \Sigma$ does not decrease the modeling power. However, the computationally complexity of the problem is different in SPPs with uniform/non-uniform clearance levels. To simplify the presentation, in this subsection let us denote an SPP (i.e., Problem 1) with a non-uniform clearance level γ as SPP-N while we denote an SPP with a uniform clearance level ($\gamma(\sigma) = 1$ for all $\sigma \in \Sigma$) as SPP-U.

Theorem 2: SPP-N and SPP-U has the same modeling power.

Proof: Since SPP-U is a subclass of SPP-N, we only need to prove that any SPP-N can be converted to an SPP-U with the same solution space.

Given an instantiation of Problem 1 $(G, \ell, \gamma, \mathbf{c})$, we convert G into a new automaton by replacing each σ_i-transition in G with $\gamma(\sigma_i) = w_i > 1$ by a sequence of w_i σ-transitions and $w_i - 1$ new states, yielding automaton G'. Then we define $\gamma'(\sigma_i) = 1$ for all events in G', and $(G', \ell, \gamma', \mathbf{c})$ is an instantiation with uniform clearance function γ'. Such a construction is illustrated in Figure 3. Clearly, $(G, \ell, \gamma, \mathbf{c})$ and $(G', \ell, \gamma', \mathbf{c})$ have the same solution space, which concludes the proof. □

Consequently, introducing the non-uniform clearance level γ into the problem does not increase its modeling power. However, SPP-N and SPP-U may have different computational complexity. As we have proved in the previous subsection, SPP with non-uniform clearance level γ is weakly NP-complete. On the other hand, although SPP-U and SPP-N have the same modeling power, SPP-U may not be weakly NP-complete since the conversion from an SPP-N to its equivalent SPP-U using

the manipulation above is not binarily polynomial. In fact, the conversion from SPP-N to SPP-U using the manipulation in Figure 3 will generate w_i states for each σ_i-transition with $\gamma(\sigma_i) = w_i > 1$: the size of the resulting SPP-U is of $\sum w_i$ which is exponential to $\sum \log(w_i)$ in the Knapsack problem. So far, the complexity class of SPP-U remains open and will be explored in our future work.

V. Computing Local Optimal Solutions Based on Centrality

In this section, we develop a heuristic method to solve the secret protecting problem with a locally optimal policy. The intuition of our method is to protect the events that are assigned to some transitions that block a large number of routes to secrets. Intuitively, for two transitions t_1 and t_2, if all *simple paths*[4] from q to q' pass t_1 more often than t_2, then t_1 is likely to be more "important" than t_2 since protecting t_1 raises the protection level of more paths than protecting t_2 (i.e., protecting t_1 blocks out more paths from the side of intruders). Analogously, an event that is assigned to important transitions is likely to be more important than an event assigned to transitions that are less important.

For example, in Figure 1, transition $q_0 \xrightarrow{\sigma_1} q_1$ is more "important" than transition $q_3 \xrightarrow{\sigma_8} q_5$. The reason is that when σ_1 is protected all trajectory with length 1 has to pass the security check. In comparison, transition $q_3 \xrightarrow{\sigma_8} q_5$ is not so "important" since protecting it does not set a security check on the trajectory: $q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_3} q_3 \xrightarrow{\sigma_5} q_5$. Hence, among all events we determine which events are possibly more "important" — for the aim of secret protection — than others. To this end, we introduce a notion called *centrality* to measure the importance of the transitions in a system.

A. Notion of Centrality

The notion of *centrality* plays an important role in the context of *social networks* [11] [31] [32]. In brief, centrality describes which vertices/edges are most central in a (directed or undirected) network. In the literature, various different notions of centrality (such as *betweenness centrality*, *degree centrality*, *closeness centrality*) have been proposed. In this paper, the notion of centrality we consider is the so-called *betweenness centrality* [11] which well describes the importance of transitions from the aspect of the secret protection problem studied in this work.

Definition 9: [11] Given a digraph (V, E) where V, E are sets of nodes and arcs, respectively, the *(betweenness) centrality* of an arc e in E is defined as:

$$\mathcal{C}_B(e) = \sum_{\forall v, v' \in V} \frac{N(v, v'|e)}{N(v, v')} \tag{8}$$

where $N(v, v')$ is the number of shortest paths from v to v' and $N(v, v'|e)$ is the number of shortest paths from v to v' that passing through the arc e. ♢

[4]In an automaton $G = (Q, \Sigma, \delta, q_0)$, a path from state $q_{i_1} \in Q$ to state $q_{i_n} \in Q$ is a sequence of alternated states and transitions: $q_{i_1} \xrightarrow{\sigma_{i_1}} q_{i_2} \xrightarrow{\sigma_{i_2}} \cdots \xrightarrow{\sigma_{i_{n-1}}} q_{i_n}$. Such a path is said to be *simple* if all states $q_{i_1}, \ldots, q_{i_n}$ appearing in it are distinct.

In this section, to simplify the presentation, the sum operator $\sum$ is such that it ignores the summands whose denominator is zero. That is, in Eq. (8), any item with $N(v, v') = 0$ (i.e., v' is not reachable from v) are ignored.

B. Transition Centrality and Event Centrality

Given two states $q, q' \in Q$, we denote by $\Pi(q, q')$ (resp., $\Pi_{min}(q, q')$) the set of all simple paths (resp., all shortest paths) starting from q and ending at q'. Moreover, we denote by $\Pi(q, q'|t)$ (resp., $\Pi_{min}(q, q'|t)$) the set of all simple paths (resp., all shortest paths) starting from q, ending at q', and passing transition t. Analogous to Definition 9, the *transition centrality* can be defined as the following.

Definition 10: Given an automaton $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_p \cup \Sigma_{up}$, the centrality of a transition $t = (q, \sigma, q')$ in G is defined by:

$$\mathcal{C}_B(t) = \begin{cases} \sum_{q, q' \in Q} \frac{|\Pi_{min}(q, q'|t)|}{|\Pi_{min}(q, q')|}, & \sigma \in \Sigma_p \\ 0, & \sigma \in \Sigma_{up}. \end{cases} \tag{9}$$

♢

In plain words, the centrality of a protectable transition is the sum of the fraction of all shortest paths from q to q' that pass through transition t. On the other hand, the centrality for any unprotectable event is zero. If for each pair of states (q, q') in G there exists at most one transition from q to q', G can be viewed as a digraph such that computing $\mathcal{C}_B(t)$ can be done using the method of Brandes [33] in $O(|Q| \cdot |E| + |Q|^2 \log |Q|)$ time. However, in general some states in G may have more than one outgoing transitions such that Brandes's algorithm cannot be directly applied. Hence, we propose an implementation of Brandes's algorithm to compute the centrality of a transition. The procedure is explained in the Appendix. After we have the transition centralities, the event centrality for event $\sigma \in \Sigma$ is defined as the maximal centrality of transitions labeled by it.

Definition 11: Given an automaton $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_p \cup \Sigma_{up}$, the centrality $\mathcal{C}_B(\sigma)$ of protecting the event are defined as:

$$\mathcal{C}_B(\sigma) = \max_{\forall t = (q, \sigma, q') \in \delta} \{\mathcal{C}_B(t)\} \tag{10}$$

♢

The event centrality characterizes the importance of an event σ in the automaton from the graphical point of view. Hence, if the cost of event protection and the raise of the clearance level are both unitary, we can choose those events with high centrality so that the security is ensured by protecting a few of those events. When the cost for protection is not unitary, besides the centrality, the cost of protection must also be taken into account. Intuitively, an event that is very expensive to be protected (i.e., $\mathbf{c}(\sigma)$ is high) will have a low score so that it is unlikely to be chosen in our algorithm; on the other hand, an event whose protection raises the clearance level a lot (i.e., $\gamma(\sigma)$ is high) should have a high score so that it is likely to be chosen. Hence, we introduce the *cost-weighted centrality* of an event as the following.

Definition 12: Given an automaton $G = (Q, \Sigma, \delta, q_0)$, the *cost-weighted centrality* of an event σ is defined as:

$$\mathcal{E}_c(\sigma) = \frac{\gamma(\sigma)}{\mathbf{c}(\sigma)} \cdot \mathcal{C}_B(\sigma) \tag{11}$$

$\diamondsuit$

C. Heuristic Algorithm

In this section, we provide a heuristic method to compute a locally optimal protecting policy. The method is summarized in Algorithm 1. In the beginning, the centrality of transitions and events are computed according to Eqs. (9) and (10). Set $\mathcal{P}$ is initialized as the empty set, i.e., no event is protected. In the first phase (steps 4–9), in each iteration, the validity of $\mathcal{P}$ is determined by solving a series of shortest path problems in a weighted digraph of G (see Proposition 3) whose complexity is $O(|Q|^3)$. If $\mathcal{P}$ is not valid, an unsafe shortest path π will be returned on which the number of protected events is fewer than $\ell(q)$. Then, among all protectable but not protected events in $(\pi \setminus \mathcal{P}) \cap \Sigma_p$, an event with the maximal cost-weighted centrality value ($\mathcal{E}_c(\sigma)$) is added to $\mathcal{P}$. If no such σ exists, the algorithm quits since there is no solution. Otherwise, a candidate solution $\mathcal{P}$ is found which, however, may not be locally optimal. Then, in the second phase (steps 10–12), a local search is performed to remove the redundant events from $\mathcal{P}$ and eventually outputs a locally optimal solution (in step 13).

In Algorithm 1, Steps 1–2 compute $\Pi_{min}(q, q')$ for $|Q|^2$ times. Computing one $\Pi_{min}(q, q')$ has complexity $O(|Q|^2 \cdot |E| + |Q|^2 \log |Q|)$. Since in a nondeterministic automaton the number of transitions is $|E| \leq |Q|^2 \cdot |\Sigma|$, $O(|Q|^2 \cdot |E| + |Q|^2 \log |Q|)$ becomes $O(|Q|^4 \cdot |\Sigma| + |Q|^2 \log |Q|)$. Hence, Steps 1–2 has complexity $O(|Q|^6 \cdot |\Sigma| + |Q|^4 \log |Q|) = O(|Q|^6 \cdot |\Sigma|)$. Steps 3–12 solve at most $2 \cdot |\Sigma_p|$ shortest path problems, whose complexity is $O(|Q|^2 \cdot |\Sigma|)$ which is dominated by that of Steps 1–2. Therefore, the gross complexity of Algorithm 1 is $O(|Q|^6 \cdot |\Sigma|)$ that is polynomial in the size of the system.

Theorem 3: The protecting policy $\mathcal{P}$ returned by Algorithm 1 is a locally optimal protecting policy.

Proof: According to Proposition 3, a protecting policy $\mathcal{P}$ is valid if and only if its corresponding shortest path problem returns no solution. Hence, $\mathcal{P}$ returned by Algorithm 1 is valid. The local optimality is guaranteed by the local optimization procedure in Steps 10-12 in Algorithm 1. $\square$

Example 2: [Ex. 1 cont.] Again consider the system in Figure 1 with $\ell(q_3) = 2$, $\ell(q_5) = 3$ and $\ell(q_i) = 0$ for $i \in \{0, 1, 2, 4\}$. Assume that the costs of all events in Σ are unitary, and $\gamma(\sigma_3) = 2$ and $\gamma(\sigma_i) = 1$ for $i \neq 3$. The centrality of transitions and events are depicted in Tables I and II, respectively. For example, consider transition $\hat{t} : q_2 \to q_5$ labeled by σ_5. Since q_5 is reachable from q_2, we have $\mathcal{C}_B(\hat{t}) = \sum_{q,q' \in Q} \frac{|\Pi_{min}(q,q'|\hat{t})|}{|\Pi_{min}(q,q')|}$. We observe that for the following $(q, q') \in Q \times Q$, $|\Pi_{min}(q, q'|\hat{t})|/|\Pi_{min}(q, q')| > 0$:

$$|\Pi_{min}(q_0, q_5|\hat{t})| = 2, \qquad |\Pi_{min}(q_0, q_5)| = 4$$
$$|\Pi_{min}(q_1, q_5|\hat{t})| = 1, \qquad |\Pi_{min}(q_0, q_5)| = 2$$
$$|\Pi_{min}(q_2, q_5|\hat{t})| = 1, \qquad |\Pi_{min}(q_0, q_5)| = 1$$

Algorithm 1 Compute a Locally Optimal Protecting Policy

Input: A system $G = (Q, \Sigma, \delta, q_0)$ with $\Sigma = \Sigma_p \cup \Sigma_{up}$, a *clearance function* γ, a cost function $\mathbf{c} \in \mathbb{R}_{\geq 0}$, a security requirement ℓ, an integer $k \in \mathbb{N} \setminus \{0\}$

Output: A locally optimal protecting policy or NONE

1: compute centrality of transitions in G by Eq. (9);
2: compute centrality and the cost-weighted centrality of events in Σ by Eqs. (10) and (11);
3: let $\mathcal{P} = \varnothing$, determine if $\mathcal{P}$ is valid;
4: **while** $\mathcal{P}$ is not valid (i.e., an unsafe path π is detected), **do**
5: if $(\pi \setminus \mathcal{P}) \cap \Sigma_p = \varnothing$, output NONE and exit.
6: select an event $\sigma \in (\pi \setminus \mathcal{P}) \cap \Sigma_p$ whose $\mathcal{E}(\sigma)$ is maximal;
7: let $\mathcal{P} = \mathcal{P} \cup \{\sigma\}$;
8: determine if $\mathcal{P}$ is valid;
9: **end while**
10: **for all** $\sigma \in \mathcal{P}$, **do**
11: if $\mathcal{P} \setminus \{\sigma\}$ is valid, let $\mathcal{P} = \mathcal{P} \setminus \{\sigma\}$;
12: **end for**
13: output $\mathcal{P}$.

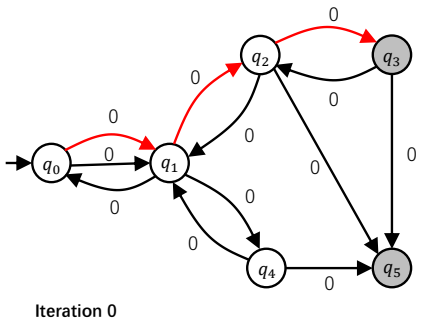
Hence, $\mathcal{C}_B(\hat{t}) = 2/4 + 1/2 + 1/1 = 2$. Analogously we have $\mathcal{C}_B(\tilde{t}) = 2$ for $\tilde{t} : q_3 \to q_5$ labeled by σ_5. Therefore, $\mathcal{C}_B(\sigma_5) = \max_{\forall t=(q,\sigma_5,q') \in \delta} \{\mathcal{C}_B(t)\} = \max\{2, 2\} = 2$.

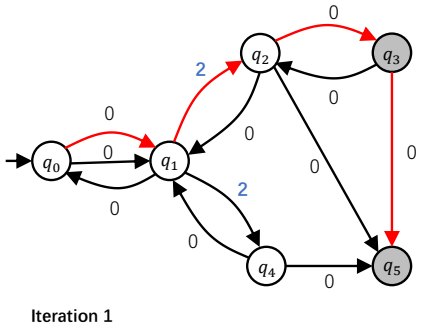
Once the cost-weighted event centrality is obtained, the heuristic algorithm runs as follows:

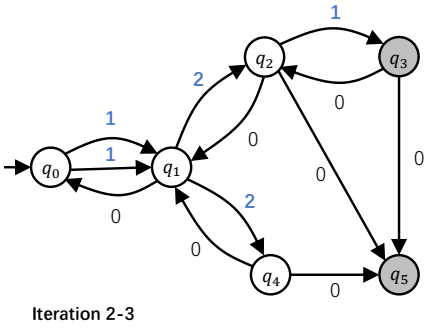
1) Initially, $\mathcal{P} = \varnothing$;
2) Solve two shortest path problems (from q_0 to q_3 and q_5, respectively) in the weighted digraph in Figure 4(a) (corresponding to $\mathcal{P} = \varnothing$). Since there exists a shortest path (red) $\pi = q_0 \to q_1 \to q_2 \to q_3$ whose weight is 0 that is lower than $\ell(q_3) = 2$, protecting policy $\mathcal{P} = \varnothing$ is not valid.
3) Add an event on the shortest path with the highest centrality that is σ_3 (with $\mathcal{C}_B(\sigma_3) = 14.0$).
4) Solve two shortest path problems (from q_0 to q_3 and q_5, respectively) in Figure 4(b) (corresponding to $\mathcal{P} = \{\sigma_3\}$). Since there exists a shortest path (red) $\pi = q_0 \to q_1 \to q_2 \to q_3 \to q_5$ whose weight is 2 that is lower than $\ell(q_5) = 3$, protecting policy $\mathcal{P} = \{\sigma_3\}$ is not valid.
5) Add an event on the shortest path with the highest centrality that is σ_6 (with $\mathcal{C}_B(\sigma_6) = 4.0$).
6) By repeatedly doing the above procedure for another two iterations we obtain $\mathcal{P} = \{\sigma_1, \sigma_3, \sigma_6, \sigma_9\}$. There does not exist a shortest path whose weight is fewer than $\ell(q)$, which indicates that $\mathcal{P}$ is valid.
7) Examine if $\mathcal{P}' = \mathcal{P} \setminus \{\sigma_i\}$, $i = 1, 3, 6, 9$ is valid. In this case, $\mathcal{P}' = \mathcal{P} \setminus \{\sigma_6\}$ is valid. Hence, by removing σ_6 we obtain a locally optimal protecting policy $\{\sigma_1, \sigma_3, \sigma_9\}$.

Hence, Algorithm 1 outputs the locally optimal protecting policy $\mathcal{P} = \{\sigma_1, \sigma_3, \sigma_9\}$ whose cost is 3. $\diamondsuit$

Example 3: Consider the system G in Figure 5 which represents the architecture of a website. This system consists of 19 states, 24 transitions, and 16 events. In the system there are three databases whose states are secret ones (marked in







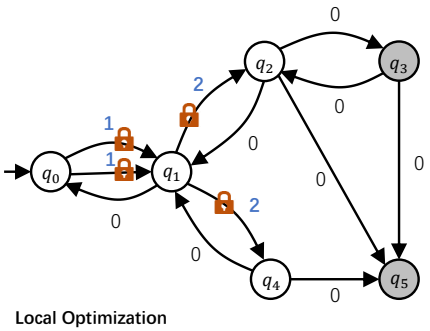


Fig. 4: The weighted underlying digraph in each iteration in Example 2: (a) iteration 0, (b) iteration 1, (c) iteration 2-3, (d) after the local optimization. The unsafe trajectories are marked in red.

Transition	label	T-centrality	Transition	label	T-centrality
$0 \to 1$	σ_1, σ_9	2.5	$2 \to 5$	σ_5	2.0
$1 \to 0$	σ_2	4.0	$3 \to 2$	σ_7	4.0
$1 \to 2$	σ_3	7.0	$3 \to 5$	σ_8	1.0
$1 \to 4$	σ_3	5.0	$4 \to 1$	σ_4	4.0
$2 \to 1$	σ_4	6.0	$4 \to 5$	σ_5	2.0
$2 \to 3$	σ_6	4.0			

TABLE I: The centrality of transitions in Example 2.

σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	σ_9
2.5	4.0	14.0	6.0	2.0	4.0	4.0	1.0	2.5

TABLE II: The centrality of events in Example 2.

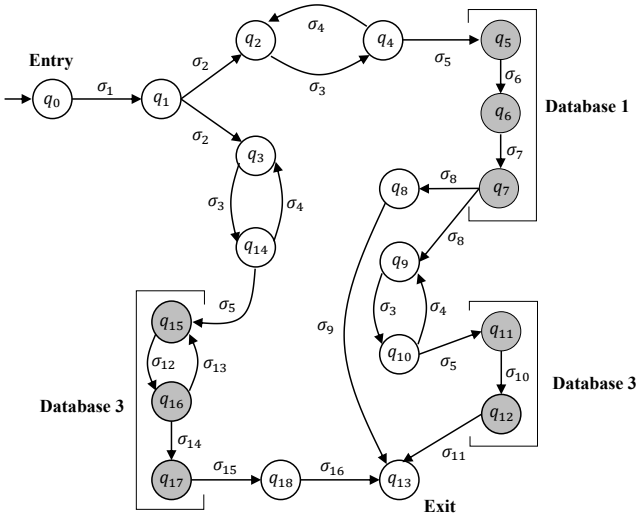


Fig. 5: The system in Example 3.

	Run1	Run2	Run3	Run4	Run5
$\mathbf{c}(\sigma_1)$	1	8	2	7	7
$\mathbf{c}(\sigma_2)$	∞	∞	∞	∞	∞
$\mathbf{c}(\sigma_3)$	1	8	6	6	6
$\mathbf{c}(\sigma_4)$	1	5	5	5	5
$\mathbf{c}(\sigma_5)$	1	7	4	4	4
$\mathbf{c}(\sigma_6)$	1	5	2	5	1
$\mathbf{c}(\sigma_7)$	1	5	1	5	2
$\mathbf{c}(\sigma_8)$	∞	∞	∞	∞	∞
$\mathbf{c}(\sigma_9)$	1	1	1	2	3
$\mathbf{c}(\sigma_{10})$	1	5	5	5	5
$\mathbf{c}(\sigma_{11})$	1	5	8	5	5
$\mathbf{c}(\sigma_{12})$	1	4	4	4	4
$\mathbf{c}(\sigma_{13})$	1	1	2	2	2
$\mathbf{c}(\sigma_{14})$	1	6	3	6	6
$\mathbf{c}(\sigma_{15})$	1	4	1	4	6
$\mathbf{c}(\sigma_{16})$	1	4	1	4	4

TABLE III: Cost of benchmark in Example 3.

	A1 Solution	Cost	GO Solution	Cost
Run1	σ_3, σ_5	2	σ_1, σ_5	2
Run2	σ_3, σ_5	15	σ_3, σ_5	15
Run3	σ_1, σ_5	6	σ_1, σ_5	6
Run4	$\sigma_3, \sigma_5, \sigma_7$	15	$\sigma_3, \sigma_5, \sigma_7$	15
Run5	$\sigma_3, \sigma_5, \sigma_7$	12	$\sigma_3, \sigma_5, \sigma_6$	11

TABLE IV: Results of the benchmark in Example 3. In the headline, "A1 Solution" denotes the locally maximal protecting policy obtained by Algorithm 1, and "GO Solution" denotes a globally optimal solution obtained by a bruteforce search.

shadow), i.e., visiting these databases requires a certain confidential level. In this system, events σ_2, σ_8 are unprotectable, while all other events are protectable.

We execute Algorithm 1 with $k = 2$ (2-sampled centrality) for five runs with different parameters. The cost functions in five runs are generated at random and are listed in Table III. The security requirement for each server $i = 1, 2, 3$ is $1, 2, 3$ in Runs 1-3 and $1, 2, 5$ in Runs 4-5. Precisely speaking:

1) in Runs 1-3, $\ell(q_5) = 1, \ell(q_6) = 1, \ell(q_7) = 1$, $\ell(q_{15}) = 2, \ell(q_{16}) = 2, \ell(q_{17}) = 2$, $\ell(q_{11}) = 3, \ell(q_{12}) = 3$;
2) in Runs 4-5, $\ell(q_5) = 1, \ell(q_6) = 1, \ell(q_7) = 1$, $\ell(q_{15}) = 2, \ell(q_{16}) = 2, \ell(q_{17}) = 2$, $\ell(q_{11}) = 5, \ell(q_{12}) = 5$.

Results are summarized in Table IV. We can see that four cases the solution is optimal, while in Run 5 the solution is very close to the optimal one. ♢

D. Large Benchmark in Randomized Automata

A benchmark for several of randomly generated systems with $50-100$ states whose results are summarized in Table V.[5] The cost to protect each event is randomly set from 1 to 10, while each event is assigned to label on average 1.5–2 transitions. The columns $|\mathcal{P}|$ and $C(\mathcal{P})$ are the number of protected events and total cost of the solutions (protecting policy) obtained by Algorithm 1. The columns $|\mathcal{P}_{go}|$ and $C(\mathcal{P}_{go})$ are those of the global optimal solutions obtained by a brute-force search. In 37% of the cases, the solution by our algorithm is globally optimal, i.e., its cost is minimal. In 50% and 70% of the cases our solution can get a solution with $<10\%$ and $<20\%$ cost deviated from the global optimal one, respectively. It is also notable that in those entries where the solutions are suboptimal, the number of protected events by Algorithm 1 are usually close to the globally optimal one: the number of events in $\mathcal{P}$ is 1–2 more than that in $\mathcal{P}_{go}$ (note $\mathcal{P}$ is locally optimal and is not a superset of $\mathcal{P}_{go}$). As far as we know, there exists no efficient way to find a global optimal solution for an SPP except the brute-force search which is very time consuming (taking about eight hours for each entry in Run 11 to obtain $\mathcal{P}_{go}$) comparing with Algorithm 1 (less than five minutes for Run 11 in Table V), which demonstrates the efficiency of the centrality-based heuristic algorithm proposed in this section.

[5]Some cases have fewer experiments because a cutoff time (8h) was set to exclude exceedingly time-consuming brute-force searches.

VI. Secret Protecting Problem with Disruptiveness

In previous sections we consider SPP with the cost criterion. In this section, besides the cost, we consider the *disruption* to legal users of the system incurred by the protections. In general, a system contains some featured functions designed for legal users for daily use (e.g., streaming a video clip, sending an email). We model the featured functions of a system as *marker states* Q_m in an automaton. That is, each sequence of activities ending at a marker state represents a completed use of the system, and thus language $L_m(G)$ characterizes the normal behavior of legal users of the system. Note that in practice using featured functions (e.g., streaming a video clip) and checking confidential information (e.g., account password) in a system are typically separated. Hence, we do not impose any special relation between marker states Q_m and the secret level function ℓ.

As we have mentioned before, to protect secrets in the system from unauthorized visits, some events in the system must be protected. However, the protection (such as password check) on an event that is on a trajectory of normal use incurs disruption to users, which can be viewed as a *penalty* of protection. In theory, to minimize such disruptions the protections should be placed on the events that are not on the trajectory between the initial and the marker states whenever possible. However, this goal is usually incompatible with the goal of minimizing the cost of protection discussed in previous sections. Hence, the balance between the cost and the disruptiveness must be considered. Note that since the SPP with cost criterion solely is NP-hard, the general form of SPP studied in this section with both criteria of cost and disruptiveness is also NP-hard.

A. Penalty Function and CP-weighted Centrality

In this subsection first we formulate the notion of disruptiveness incurred by event/transition protections as a *penalty function*. Intuitively, to reduce the disruption to users, a transition should not be protected unless we have to. That is, the penalty of protecting a transition close to reach marker states should be no higher (and possibly lower) than the penalty of protecting a transition far to reach marker states. For example, in the system in Figure 6 to protect σ_1 is more disruptive than to protect σ_3, since a user may repeatedly visit q_0 and q_1 before reaching the marker state q_2. On the other hand, for two transitions t_1 and t_2, if t_1 is coreachable to more marker states than t_2, to protect t_1 should cause a higher penalty than to protect t_2 since it disrupts more featured functions. Hence, we expect a penalty function that assigns a high penalty to the transitions that are *far* from the marker states (in the sense of co-reachability) and those transitions that are co-reachable to many marker states. By such a motivation, we define the *event penalty* as the following.

Definition 13: Given a system $G = (Q, \Sigma, \delta, q_0, Q_m)$, let t be a transition from state $q' \in Q$ to state $q'' \in Q$. We define $N(t, Q_m) = |\{q \in Q_m \mid (\exists s \in \Sigma^*)\ \delta(q'', s) = q\}|$ as the number of marker states Q_m reachable from transition t. We define $d(t, q) = \pi_{\min}(q'', q) + 1$ where $\pi_{\min}(q'', q)$ is the

Run	Size	No. Trans.	No. Events	No. Secrets	Security Level	$\|\mathcal{P}\|$	$C(\mathcal{P})$	$\|\mathcal{P}_{go}\|$	$C(\mathcal{P}_{go})$	Deviation
1	50	90	44	3	1,1,2	3	14	3	13	0.08 (0)
			34	3	2,3,3	7	24	6	22	0.09 (1)
			33	2	4,4	11	56	9	42	0.33 (2)
			35	3	2,4,5	9	43	8	38	0.13 (1)
			35	2	4,5	11	65	10	60	0.08 (1)
			39	3	1,2,2	3	19	3	19	0 (0)
			37	3	1,2,2	3	10	3	10	0 (0)
			43	3	1,2,2	4	11	3	11	0 (1)
2	55	107	49	3	1,1,1	1	2	1	2	0 (0)
			45	3	1,2,2	3	13	2	10	0.30 (1)
			39	3	3,4,4	8	22	5	17	0.29 (3)
			39	3	2,3,3	4	23	3	19	0.21 (1)
			46	3	1,2,2	2	4	2	4	0 (0)
			45	3	1,1,2	2	10	2	8	0.25 (0)
			43	3	1,1,2	2	4	2	4	0 (0)
3	60	102	44	3	1,1,1	2	3	2	3	0 (0)
			43	3	1,1,2	4	10	3	9	0.11 (1)
			37	4	2,2,2,3	8	42	7	38	0.11 (1)
			43	3	1,2,2	6	15	6	13	0.15 (0)
			37	2	3,4	11	65	9	53	0.23 (2)
			44	2	1,2	3	21	3	19	0.11 (0)
			43	3	1,1,2	4	16	4	16	0 (0)
4	65	122	54	3	1,1,2	8	16	6	11	0.45 (2)
			50	3	1,1,2	4	18	2	17	0.06 (2)
			45	2	4,5	10	46	10	45	0.02 (0)
			52	3	1,1,1	1	2	1	2	0 (0)
			51	3	1,2,2	5	11	4	11	0 (1)
			54	3	1,2,2	6	11	4	11	0 (2)
5	70	111	48	3	1,2,2	7	28	6	23	0.22 (1)
			43	2	3,4	7	33	7	33	0 (0)
			46	3	1,1,2	4	22	3	18	0.22 (1)
			50	3	2,2,2	6	24	5	19	0.26 (1)
			46	3	1,2,2	6	32	6	28	0.14 (0)
			49	2	1,2	3	10	3	10	0 (0)
6	75	133	49	3	1,3,3	8	34	9	29	0.17 (-1)
			51	3	1,1,2	3	11	3	11	0 (0)
			51	3	2,2,3	12	39	10	27	0.44 (2)
			51	2	2,2	10	52	8	39	0.33 (2)
			55	3	1,1,2	7	30	4	27	0.11 (3)
7	80	148	53	4	1,1,2,3	6	28	4	27	0.04 (2)
			58	2	2,2	3	9	3	9	0 (0)
8	85	155	58	3	1,2,2	3	11	3	11	0 (0)
			61	3	2,2,3	11	47	6	31	0.52 (5)
			60	2	1,4	9	31	8	27	0.15 (1)
			60	3	1,2,2	3	6	3	6	0 (0)
			55	3	2,3,3	10	27	5	20	0.35 (5)
9	90	169	63	2	1,2	6	21	6	18	0.17 (0)
			65	2	1,2	4	15	4	11	0.36 (0)
10	95	156	58	3	1,2,2	3	4	3	4	0 (0)
			56	2	3,3	6	18	6	18	0 (0)
			57	3	2,2,2	6	14	4	12	0.17 (2)
			59	3	3,3,4	9	33	8	29	0.14 (1)
			57	4	2,2,3,3	8	42	7	33	0.27 (1)
			61	3	3,3,3	6	29	6	22	0.32 (0)
11	100	189	70	3	1,1,1	4	13	3	12	0.08 (1)
			73	2	1,1	3	4	3	4	0 (0)
			72	2	1,4	6	32	6	32	0 (0)

TABLE V: Benchmark on randomized systems. The "Deviation" is denoted as "a (b)" in which $a = C\mathcal{P}/C\mathcal{P}_{go} - 1$ is the cost ratio and $b = |\mathcal{P}| - |\mathcal{P}_{go}|$ is the difference of number of protected events between the solution by Algorithm 1 and the global optimal solution by brute-force.

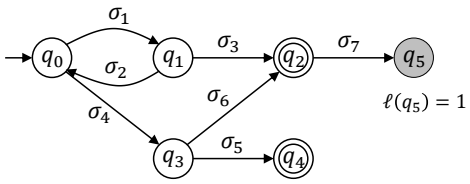


Fig. 6: A system with marker states q_2, q_4 and secret state q_5.

length of the shortest path from state q'' to state q.[6] We also note that the penalty for disruptiveness is not related to the security specification ℓ. ♢

Definition 14: [Transition Penalty] Given a system $G = (Q, \Sigma, \delta, q_0, Q_m)$, the penalty of a transition t is defined as:

$$\begin{aligned} P(t) &= N(t, Q_m) \cdot H(t, Q_m) \\ &= N(t, Q_m) \cdot \frac{N(t, Q_m)}{\sum_{q \in Q_m} \frac{1}{d(t,q)}}. \end{aligned} \tag{12}$$

where $H(t, Q_m) = \frac{N(t,Q_m)}{\sum_{q \in Q_m} \frac{1}{d(t,q)}}$ is the *harmonic mean distance* from transition t to the states in Q_m.

Eq. (12) is motivated by the work of [34] in which harmonic mean distance is introduced to measure the connectivity of a node in a graph. Since $P(t) = N(t, Q_m) \cdot H(t, Q_m)$, one can verify that $P(t)$ increases when: (i) $N(t, Q_m)$ increases, meaning that t is coreachable to more marker states, and (ii) $H(t)$ increases, meaning that t is far from the marker states. Consequently, we define the *event penalty* as the sum of transition penalties that the event is assigned to, since the protection of an event incurs disruptions on all those transitions.

Definition 15: [Event Penalty] Given a system $G = (Q, \Sigma, \delta, q_0, Q_m)$, the penalty of a transition t is:

$$P(\sigma) = \begin{cases} \Sigma_{\ell(t)=\sigma} P(t), & \sigma \in \Sigma_p \\ \infty, & \sigma \in \Sigma_{up} \end{cases} \tag{13}$$

♢

Example 4: Consider the system in Figure 6. Since all transitions are labeled distinctly, the penalty of each event σ_i is equal to the transition penalty $P(t)$ where t is the transition that σ_i labels. According to Eqs. (12) and (13), we have $P(\sigma_1) = 5.33, P(\sigma_2) = 6, P(\sigma_4) = 4$, $P(\sigma_3) = P(\sigma_5) = P(\sigma_6) = 1$, and $P(\sigma_7) = 0$. This coincide with our intuition that to protect σ_1, σ_4 is more disruptive than to protect $\sigma_3, \sigma_5, \sigma_6$ since σ_1, σ_4 are coreachable to two marker states while $\sigma_3, \sigma_5, \sigma_6$ are coreachable to only one marker state. Moreover, to protect σ_1 is more disruptive than to protect σ_4: although they are both coreachable to two marker states, the distance between σ_1 to the marker states is larger than the distance between σ_4 to the marker states. Besides, the penalty to protect event σ_7 is zero, since the transition labeled by σ_7 is not coreachable to any marker state. ♢

Now let us consider the SPP with both criteria of cost and disruption. In [8] the disruption is treated as a binary variable (i.e., either "disrupted" or "undisrupted") so that the disruption, if it exists, is modeled as a "one level-up" of the cost. However, in practice, it is usually difficult to quantify the "disruption" in monetary terms. Therefore, in this section we introduce a parameter $\lambda \in [0, 1]$ to balance the cost and the disruption of event protection.

Definition 16: Given an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, the *cost-penalty-weighted centrality* (CP-weighted centrality) of an event σ is defined as:

$$\mathcal{E}_{cp}(\sigma) = \lambda \cdot \frac{\gamma(\sigma)}{\mathbf{c}(\sigma)} \cdot \mathcal{C}_B(\sigma) - (1 - \lambda) \cdot P(\sigma). \tag{14}$$

♢

Compared with Eq. (11) which contains a single term associated with the cost-weighted centrality, Eq. (14) contains an additional term $-P(\sigma)$ which characterize the affect of penalty. When $\lambda = 1$, $\mathcal{E}_{cp}(\sigma) \sim \mathcal{E}_c(\sigma)$ which is equivalent to the event centrality developed in Section V, meaning that the disruptiveness to users is not considered. On the other hand $\lambda = 0$ implies $\mathcal{E}_{cp}(\sigma) = -P(\sigma)$, meaning that only the disruptiveness is interested regardless the cost. Note that if σ is unprotectable, $\mathcal{E}_{cp}(\sigma)$ is $-\infty$ (such that σ will never be chosen to be protected) due to the definition of $P(\sigma)$.

Remark 2: In this work we use linear combination function since it is commonly used to effectively describe the tradeoff of two aspects. Other meaningful ways of combination will be considered in our future work. ♢

Remark 3: In this section, we consider static protecting policies as a set of events. Another interesting type of protecting policies is the *dynamic protecting policies* in our previous work [10]. We point out that for dynamic protecting policies, disruptiveness may not be defined as penalties on transitions and events (like Definitions 14 and 15). Hence, the SPP with both cost and disruptiveness and dynamic protecting policies requires further investigation. ♢

B. Heuristic Algorithm

To find a locally optimal solution for an SPP with both cost and disruptiveness, Algorithm 1 in Section V.C can be analogously used. In brief, a Step 0 needs to be added to compute the penalty function $P(\sigma)$, and the CP-weighted centrality $\mathcal{E}_{cp}(\sigma)$ needs to be computed for each event σ and then be used in Step 6 instead of $\mathcal{E}_c(\sigma)$. The gross complexity of such heuristic algorithm is also $O(|Q|^4 \cdot k + |Q|^2 \cdot |\Sigma|)$ as discussed in the previous section. We use the following example to illustrate the method.

Example 5 (Ex. 3 continued): Now, let us consider the system in Fig. 5 with both cost and disruptiveness. In this system, state q_{13} is the only marker state that represents the featured service provided by the system. Since a user may repeatedly visit states q_2, q_4, q_9, q_{10} and so on before visiting marker state q_{13}, intuitively, to protect event σ_3 is more disruptive than to protect σ_{16}. In other words, to reduce the disruptiveness to the user, we prefer to protect events close to the marker state. So, the penalty of each transition and each event are computed according to Eqs. (12) and (13). Then the modified Algorithm 1 is applied whose results are summarized in Table VI. From Run 3 and Run 5 one can see that with the increase of the parameter λ from 0 to 1, our algorithm

[6] In plain words, $d(t, q)$ is the number of states that appear on the shortest path from q'' to q. If $q = q''$, then $d(q, t) = 1$.

	λ	A2 Solution	Total Cost	Total Disruptiveness
Run 1	0.0	σ_1, σ_5	2	21
	0.2	σ_1, σ_5	2	21
	0.4	σ_1, σ_5	2	21
	0.6	σ_3, σ_5	2	29
	0.8	σ_3, σ_5	2	29
	1.0	σ_3, σ_5	2	29
Run 2	0.0	σ_1, σ_5	15	21
	0.2	σ_1, σ_5	15	21
	0.4	σ_1, σ_5	15	21
	0.6	σ_1, σ_5	15	21
	0.8	σ_3, σ_5	15	29
	1.0	σ_3, σ_5	15	29
Run 3	0.0	σ_1, σ_5	6	21
	0.2	σ_1, σ_5	6	21
	0.4	σ_1, σ_5	6	21
	0.6	σ_1, σ_5	6	21
	0.8	σ_1, σ_5	6	21
	1.0	σ_1, σ_5	6	21
Run 4	0.0	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	21	28
	0.2	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	21	28
	0.4	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	21	28
	0.6	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	21	28
	0.8	$\sigma_3, \sigma_5, \sigma_7$	15	32
	1.0	$\sigma_3, \sigma_5, \sigma_7$	15	32
Run 5	0.0	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	14	28
	0.2	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	14	28
	0.4	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	14	28
	0.6	$\sigma_1, \sigma_5, \sigma_6, \sigma_7$	14	28
	0.8	$\sigma_3, \sigma_5, \sigma_6$	11	33
	1.0	$\sigma_3, \sigma_5, \sigma_6$	11	33

TABLE VI: Results of the benchmark in Example 5 with $k = 2$.

tends to output a protecting policy with less cost but more disruptiveness. On the other hand, when λ is relatively small, the algorithm will output a protecting policy with more cost but less disruption. $\diamondsuit$

C. A Large Benchmark

Note that the system in Example 5 is relatively small so that sometimes (e.g., Run 3) the output is not sensitive with the parameter λ. Hence, we carry out a benchmark for several of randomly generated systems with $100 - 300$ states whose results are summarized in Table VII. For better readability, the results are also illustrated in Figure 7. The cost to protect each event is randomly set from 1 to 30, while each event is assigned to about 1.5-2 transitions in average. The total cost of a protecting policy is defined by Definition 3, while the "disruptiveness" of a protecting policy $\mathcal{P}$ is defined as $\sum_{\sigma \in \mathcal{P}} P(\sigma)$, i.e., the sum or all penalties of the events protected. One can see that with the increase of the parameter λ, our algorithm tends to output a protecting policy with less cost. For example, in Run 2 with an automaton with 100 states, 224 transitions, 136 events, 2 secret states (whose security levels are 1, 2, respectively), and 10 marker states, while we increase λ from 0 to 1, the cost of the output protecting policy decreases from 35 to 18, while the measure of disruptiveness P increases from 252 to 474. As a result, by adjusting the parameter λ, the administrator of a system can use the proposed approach to compute a protecting policy which well balance the cost and the disruptiveness.

Second, we compare our performance with a greedy non-centrality-based algorithm with randomized starting point. The result of each run is shown on the last row of the run (marked as "Greedy") in Table VII. One can see that the non-centrality-based greedy search always falls into some locally optimum whose total cost and total penalty are higher than those obtained by centrality-based heuristics. Empirically, the cost of the solution obtained by the non-centrality-based greedy search is 1$\sim$1.5 times higher than that of the solution obtained by centrality-based heuristics, while the disruptiveness of the former is 2$\sim$5 times higher than the latter.

Remark 4: The locally optimal solution obtained by Algorithm 1 may not be always optimal, which is due to the nature of the heuristics. However, due to the NP-completeness of the problem, for large-scale systems (such as those presented in this subsection) it is difficult or even impossible to get the global optimum. Hence, we do not evaluate the closeness of our solutions to global optimum. On the other hand, according to the centrality defined in Eq. (11), if some events has high $\mathcal{E}_c(\sigma)$ with both dominatingly high centrality and dominatingly high cost of protection, the proposed heuristic may prefer these transitions so that the solution may deviate from the global optimum. In these systems, our method could also be combined with other heuristic methods such as simulated annealing, genetic algorithm to improve its performance in these systems. $\diamondsuit$

VII. Conclusion

In this paper, we have studied a secret protection problem with minimum costs. We have proved that computing a global optimal solution is NP-hard when events are assigned to multiple transitions. Therefore, we have developed a heuristic method using the notion of centrality to find a local optimal protection policy. Furthermore, the disruptiveness caused by the protected events to users have been taken into consideration. Then, we have considered the disruptiveness, i.e., the degree of disruptiveness of protecting policies to legal users' normal operations. A heuristic method is developed to obtain a protecting policy which can well balance the cost and the disruptiveness to users.

References

[1] 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.

[2] F. Pasqualetti, F. Dorfler, and F. Bullo, "Control-theoretic methods for cyberphysical security: Geometric principles for optimal cross-layer resilient control systems," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 110–127, 2015.

[3] R. S. L. Lin, Y. Zhu, "Synthesis of covert actuator attackers for free," *Discrete Event Dynamic Systems:Theory and Applications*, 2020.

[4] 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.

[5] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems — A Model-Based Approach with Finite Automata*. Springer-Verlag US, 2020.

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

[7] ——, "Secret securing with minimum cost," in *Proceedings of the 61st Japan Joint Automatic Control Conference*, 2018, pp. 1017–1024.

[8] ——, "Usability aware secret protection with minimum cost," *Nonlinear Analysis: Hybrid Systems*, vol. 43, p. 101111, 2021.

Run	Size	No. Trans.	No. Events	No. Secrets	Security Level	No. Marker States	λ	$\lvert\mathcal{P}\rvert$	Cost	Disruptiveness
1	100	188	109	2	$1,2$	13	$\lambda = 0.0$	6	40	437
							$\lambda = 0.2$	6	40	437
							$\lambda = 0.4$	5	25	519
							$\lambda = 0.6$	5	21	598
							$\lambda = 0.8$	5	19	784
							$\lambda = 1.0$	5	19	784
							Greedy	8	61	1296
2	100	224	136	2	$1,2$	10	$\lambda = 0.0$	5	35	252
							$\lambda = 0.2$	6	26	315
							$\lambda = 0.4$	6	26	315
							$\lambda = 0.6$	4	18	474
							$\lambda = 0.8$	4	18	474
							$\lambda = 1.0$	4	18	474
							Greedy	12	52	1478
3	200	436	264	2	$2,2$	25	$\lambda = 0.0$	32	195	5325
							$\lambda = 0.2$	32	184	5890
							$\lambda = 0.4$	35	174	7386
							$\lambda = 0.6$	32	136	8122
							$\lambda = 0.8$	31	133	9213
							$\lambda = 1.0$	25	94	9580
							Greedy	30	186	9828
4	200	426	228	2	$1,2$	25	$\lambda = 0.0$	14	103	2197
							$\lambda = 0.2$	21	100	3881
							$\lambda = 0.4$	21	97	4374
							$\lambda = 0.6$	21	90	4442
							$\lambda = 0.8$	17	68	4540
							$\lambda = 1.0$	17	65	4947
							Greedy	21	111	6124
5	300	924	509	2	$1,2$	16	$\lambda = 0.0$	81	436	10027
							$\lambda = 0.2$	81	389	10530
							$\lambda = 0.4$	80	360	10906
							$\lambda = 0.6$	79	334	11502
							$\lambda = 0.8$	63	252	10685
							$\lambda = 1.0$	64	232	12730
							Greedy	84	420	15142

TABLE VII: Benchmark on randomized systems. The "Disruptiveness" of a protecting policy $\mathcal{P}$ is defined as $\sum_{\sigma\in\mathcal{P}} P(\sigma)$.

[9] Z. Ma and K. Cai, "Optimal secret protection in discrete event systems with dynamic clearance levels," in *Proceedings of the 22nd IFAC World Congress (IFAC WC'23)*, 2023, p. to appear.

[10] ——, "Optimal secret protections in discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 2816–2828, 2022.

[11] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[12] A.-L. Barabási and M. Pósfai, *Network science*. Cambridge: Cambridge University Press, 2016. [Online]. Available: http://barabasi.com/networksciencebook/

[13] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annual Reviews in Control*, vol. 41, pp. 135–146, 2016.

[14] 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.

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

[16] Z. Ma, X. Yin, and Z. Li, "Verification and enforcement of strong infinite- and k-step opacity using state recognizers," *Automatica*, vol. 133, p. Article 109838, 2021.

[17] A. Saboori and C. N. Hadjicostis, "Verification of k-step opacity and analysis of its complexity," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 549–559, 2011.

[18] Y. Tong, Z. W. 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.

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

[20] 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.

[21] Y. Ji, X. Yin, and S. Lafortune, "Enforcing opacity by insertion functions under multiple energy constraints," *Automatica*, vol. 108, p. 108476, 2019.

[22] 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.

[23] 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.

[24] 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.

[25] R. Meira-Goes, E. Kang, R. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber-physical systems," *Automatica*, vol. 121, p. Article 109172, 2020.

[26] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," *Dynamic Games and Applications*, vol. 9, pp. 965–983, 2019.

[27] L. Lin and R. Su, "Synthesis of covert actuator and sensor attackers," *Automatica*, vol. 130, p. Article 109714, 2021.

[28] J. Jiang, Z. Ma, and K. Cai, "Secret protections in discrete-event systems with minimum costs," in *2022 American Control Conference*, 2022, pp. 3740–3745.

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

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

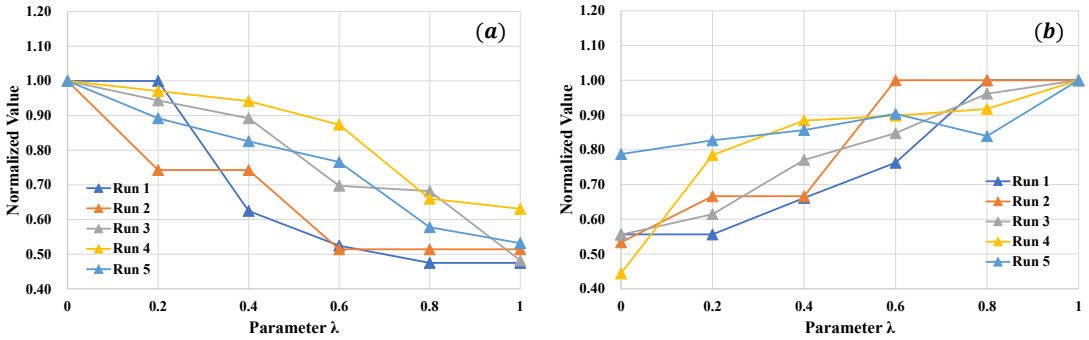


Fig. 7: The graphical illustration of the results (normalized) in Table VI: (a) the trend of cost and (b) the trend of disruptiveness of the solutions obtained with different parameter λ.

[31] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39–54, 2005.

[32] T. Alahakoon, R. Tripathi, N. Kourtellis, R. Simha, and A. Iamnitchi, "K-path centrality: A new centrality measure in social networks," in *Proceedings of the 4th Workshop on Social Network Systems*, 2011, pp. 1–6.

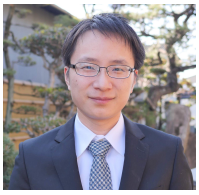
[33] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," *Social Networks*, vol. 30, no. 2, pp. 136–145, 2008.

[34] M. Marchiori and V. Latora, "Harmony in the small-world," *Physica A: Statistical Mechanics and its Applications*, vol. 285, no. 3, pp. 539–546, 2000.

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–'23), European Control Conference (ECC'19–'23), and IEEE International Conference on Systems, Man, and Cybernetics (SMC'19–'23). He is/was the Track Committee Member of the International Conference on Emerging Technologies and Factory Automation (ETFA'17–'22). In 2016 and 2022 he received the Outstanding Reviewer Award from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL and IEEE CONTROL SYSTEM LETTERS, respectively.

Jiagang Jiang Jiagang Jiang received the B.Sc. Degree in automation from Xidian University, Xi'an, China, in 2020. He is currently pursuing the Master Degree in Control Theory and Control Engineering in the School of Electro-Mechanical Engineering, Xidian University, Xi'an, China. His current research interests include information security in discrete event systems.



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 Full Professor at Osaka Metropolitan 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 cooperative control of multi-agent systems, discrete-event systems, and cyber-physical 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 an Associate Editor for IEEE Transactions on Automatic Control and a Senior Editor for Nonlinear Analysis: Hybrid Systems. He was the Chair for IEEE CSS Technical Committee on Discrete Event Systems (2019–2022), and a member of IEEE CSS Conference Editorial Board (2017–2022). He received the Pioneer Award of SICE in 2021, the Best Paper Award of SICE in 2013, the Best Student Paper Award of IEEE Multi-Conference on Systems & Control in 2010, and the Young Author's Award of SICE in 2010.

APPENDIX

Here we show how to compute the proportion of shortest paths that passes a transition. The method consists of four main steps: (i) unfold the automaton from each node to get n unfolding graphs; (ii) in each unfolding graph compute the number of shortest paths from the root node to other nodes; (iii) in each unfolding graph, compute the reverse flow of each arc; (iv) sum the flow of all corresponding arcs in all unfolding graphs to obtain the transition centrality. An illustration is shown in Figure 8.

The correctness of the method is based on the fact that it is the standard Brandes's algorithm (Algorithm 1 in [33]) with an additional treatment on multiple transitions. For better understanding, instead of presenting the technical algorithmic procedure (which can be found in [33]), we use the following example to illustrate the implementation, since most of the steps are self-explanatory. Consider the automaton in Figure 8(a) (labels are omitted).

- **Step 1: Compute unfolding graphs.** We take state A for example. We explore from state A and discard all non-shortest branches. Observe that in this automaton, state A is 0 transition away from itself, state B is 1 transition away, states C/D are 2 transitions away, and state E is

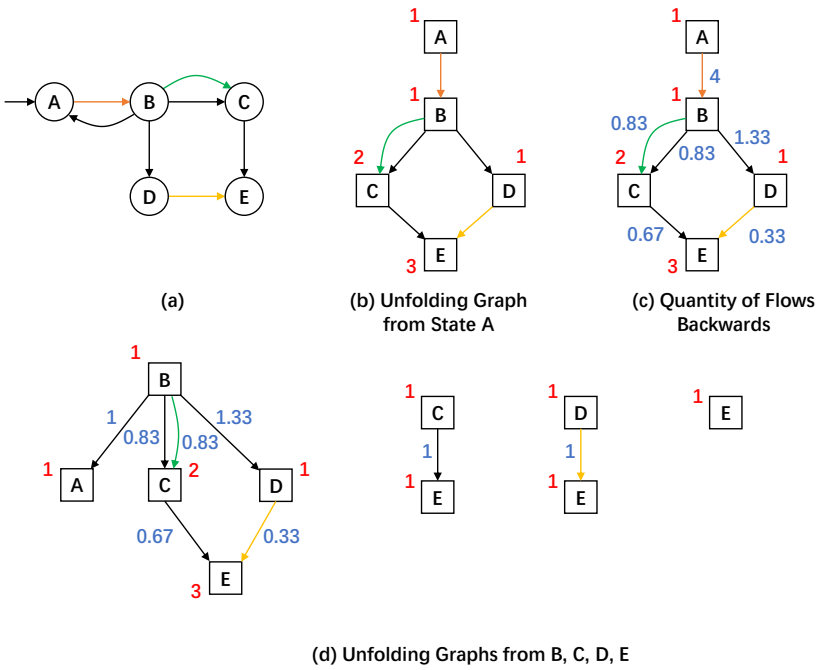


Fig. 8: The implementation to compute transition centrality.

3 transitions away. The unfolding graph from state A is shown in Figure 8(b).

- **Step 2: Compute the number of shortest paths from the root node to other nodes in unfolding graphs.** In Figure 8(b), we first set the shortest path count $\phi(A)$ for state A as 1 and iteratively compute ϕ for other states (red numbers in the figure). For multiple transitions, the shortest path count will be multiplied by the degree (multiplication) of the transitions, e.g., $\phi(C) = 2\phi(B) = 2$. For a joint node, its shortest path count is the sum of all shortest path counts of its successors. For example, $\phi(E) = \phi(C) + \phi(D) = 2 + 1 = 3$, meaning that there are 3 shortest paths from A to E.
- **Step 3: Compute the backward flow.** In Figure 8(b), we first determine the terminal nodes, which is only one state E. We assign the initial flow on E as 1, denoted as $\rho(E) = 1$. Then we compute the flows in the graph backwardly from the terminal node (blue numbers in the figure) following two rules. First, for a joint node, the flow on each incoming transition is determined according to their proportions of the shortest path count, e.g., arcs $C \to E$ and $D \to E$ contribute 2 and 1 shortest paths, respectively. So, the flow distribution on $C \to E$ and $D \to E$ is 2:1 that is 0.67:0.33. Second, whenever passing a node (backwardly), the flow is summed and is increased by 1 (one). For example, the flow on transition $C \to E$ is 0.67. When passing state C backwardly, the flow is increased to $1.67 = 0.67 + 1$ which is then equally distributed to the two arcs from B to C: each has 0.83 flow.
- **Step 4: Sum the flows to obtain the transition centrality.** We perform the procedure (Steps 1–3) for other nodes B, C, D, E whose results are shown in Figure 8(d).

Now, to compute transition centrality of a transition in the automaton, we sum the flows of the corresponding arcs in all unfolding graphs. For example, for the transition t colored in green, we have:

$$\mathcal{C}_B(t) = 0.83 + 0.83 + 0 + 0 + 0 = 1.67$$

One can verify that: from A to C there are 2 shortest paths among which 1 passes t (0.5), from A to E there are 3 shortest paths among which 1 passes t (0.33), from B to C there are 2 shortest paths among which 1 passes t (0.5), from B to E there are 3 shortest paths among which 1 passes t (0.33), and t does not appear in any other shortest paths. Hence, $\mathcal{C}_B(t) = 0.5 + 0.33 + 0.5 + 0.33 = 1.67$ according to Eq. (9). Similarly, for the transitions t' and t'' colored in orange and yellow, respectively, we have $\mathcal{C}_B(t') = 4 + 0 + 0 + 0 + 0 = 4$ and $\mathcal{C}_B(t'') = 0.33 + 0.33 + 0 + 1 + 0 = 1.66$.