Application of Supervisory Control to Secret Protection in Discrete-Event Systems

Shoma Matsui*・Kai Cai**

＊ ミシガン大学 電気計算機工学科
＊＊ 大阪市立大学 電気情報工学科
＊Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, USA
＊＊Department of Electrical and Information Engineering, Osaka City University, Japan
＊E-mail: smatsui@umich.edu
＊＊E-mail: kai.cai@eng.osaka-cu.ac.jp

キーワード：離散事象システム (Discrete-Event Systems)，スーパーバイザ制御 (Supervisory Control)，セキュリティ (Security)

1. Introduction

Protecting systems and information from adversarial accesses has increasingly become an indispensable issue of modern system design. For example, [1] discusses several attack scenarios with a typical architecture of control systems. System administrators have to effectively utilize protection techniques to mitigate risks of system's abnormal behavior and/or information breach. Typically, the objective of attackers is to mislead the system to an abnormal state or to steal sensitive information. In this paper, we consider systems containing special components or realms to which attackers want to gain access, and system administrators are responsible to protect such components/realms with proper security levels. At the same time, administrators have to take into account the protection costs, because the capacity of systems may be limited and spending infinite budgets for protections is practically infeasible.

To make our methodology applicable to a wide range of systems, we abstract a dynamic system, called the *plant* in this work, into a discrete-event system (DES). In the DES framework, plants are modeled by finite-state automata where states change to other states discretely and the transitions are event-driven. In particular, DES are suitable formal models for the dynamics and architectures of computer and network systems [2]. Moreover, we employ the supervisory control theory (SCT) of DES which was proposed by Ramadge and Wonham in 1987 [3], to compute solutions of secret protection problems we introduce. The readers are referred to [4] for a comprehensive treatment of the SCT.

In terms of anonymity and secrecy, *opacity* is a well-studied concept of securing secrets using the DES framework. There are several variations of opacity, which differ specifically in what entity in the plant to be taken as a secret [5, 6, 7, 8, 9]. A common assumption among opacity notions is that the attacker has full knowledge of the structure of the plant, but partial observability, meaning that the attacker cannot observe specific events occurring in the plant. Language-based opacity (LBO) [6] is one of the opacity variations, which holds if specific secret languages generated by the plant cannot be distinguished from nominal languages. As other variations, state-based opacity properties are investigated by Saboori and Hadjicostis [7, 8]. In [7], a notion of current-state opacity (CSO) is defined, which requires that the attacker cannot determine whether or not the plant is currently in the secret state. Initial-state opacity (ISO) introduced in [8] is another definition of state-based opacity properties. If the attacker cannot surely determine whether the initial state of the plant is a secret state due to seemingly the same strings of events from the secret initial state, the ISO of the plant holds. [9] proposes a notion of initial-and-final-state opacity (IFO) extended from CSO and ISO. For IFO, secret states are both the initial state and the final state of the system. For overview of opacity, see [5].

In contrast to opacity, we do not place any assumption on the attacker's knowledge of the plant structure, and the attacker may have full observability of the plant. For protecting secrets, we consider a system equipped with several events which can be protected by administrators. In this paper, we represent events which administrators can protect as *protectable events*, and other events are called *unprotectable events*. There are various implementations of such events in real systems, for example, the access

restriction using password authentication. Moreover, we represent secrets as specific states in the plant, called *secret states*. Such states indicate the plant's components storing a particular piece of information which should be available only for permitted users, e.g. credential information or credit card numbers of users. To prevent such secret states from being reached and thereby sensitive information discovered by the attacker, system administrators must protect appropriate protectable events in the plant. In this paper, we formulate a problem of finding a *protection policy* (if it exists) which specifies certain protectable events in the plant such that (i) the attacker always has to penetrate a certain number of protected events to reach the secret states; (ii) the highest cost level to protect specified events is minimum. Moreover, we propose algorithms based on the SCT which provide a solution to this problem. The formulation of the problem and computation of a solution are explained based on our previous technical results in [10, 11].

The remaining of this paper is organized as follows. Section 2 formulates the problem of secret protection with multiple protections and minimum costs. In Section 3, we present the methodology to compute a solution of the formulated problem. Section 4 explains our methodology using an illustrating example. Finally, we conclude this paper in Section 5.

2. Problem Formulation

In this section, we first formulate *Secret Securing with Multiple Protections and Minimum Cost Problem*, then provide a necessary and sufficient condition under which there exists a solution of the problem.

We consider abstracting a dynamic system into a DES, which is modeled as a finite-state automaton (FSA). In the FSA model, the *states* are partitioned into two groups, *nominal states* and *secret states*. Each secret state represents "accessing a realm/component containing secret information". Thus the intruder's objective is to reach one of the secret states in the plant. We denote the set of states in the plant by Q, and the set of secret states by Q_s. Note that Q_s is a subset of Q, i.e. $Q_s \subseteq Q$. A state transition in the FSA is represented by a directed edge between two states. Each transition between two states is called an *event*. For example, we express a transition from state q_i to state q_j labeled by e as "an event e happened at state q_i and then the plant moved to state q_j".

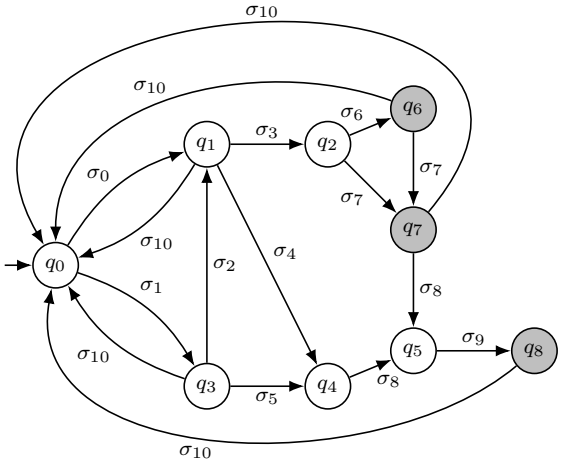

Fig. 1: Example plant; shaded states q_6, q_7, and q_8 contain secret information. The initial state q_0 indicates that a user is yet to log into the system. Events σ_0 and σ_1 represent logging into the system as a standard user (User) and as a system administrator (Admin), respectively. σ_2 indicates switching permission from Admin to User. The application is launched by σ_3 as User and by σ_5 as Admin. σ_4 means User launches the application as Admin, e.g. *sudo* in Unix-like operating systems. σ_6, σ_7, and σ_9 indicate the authentication points to obtain access to q_6, q_7, and q_8 respectively. σ_8 is an optional authentication point at which administrators can apply a weaker protection than σ_9. σ_{10} is an event indicating a "log-off" function, which leads User and Admin to the initial state q_0.

Let us explain the plant modeled as DES with an illustrating example shown in Fig. 1. Each state has a unique label and is drawn as a circle. Shaded states q_6, q_7, and q_8 indicate secret states, and other states are nominal. Thus $Q = \{q_i \mid i \in [0, 8]\}$ and $Q_s = \{q_6, q_7, q_8\}$. q_0 is the initial state of this plant, meaning that the plant dynamics starts from state q_0 and no events have happened yet. The states are interconnected with events, which are state transitions. Note that some transitions at different states have the same labels, e.g. σ_7, σ_8, and σ_{10}. We denote the set of all events by Σ. Thus in the plant of Fig. 1, $\Sigma = \{\sigma_i \mid i \in [0, 10]\}$.

The plant in Fig. 1 models a simplified system of a software application which has three restricted realms. Consider that this application works according to the user's permission levels. The users have to pass several authentication points to obtain the permission to reach the restricted realms. Secret states

q_6, q_7, and q_8 are the restricted realms which require the same level of permission, meaning that all three secret states have the same level of importance.

Next, consider that we are responsible to mitigate the risk that intruders reach secret states in the plant, and for this task, we need to install protections at appropriate points on paths to secrets. We express installing protections at certain points as protecting events at certain states. Meanwhile, the costs to protect secrets must be minimum. Secret states in the plant are said to be *protected* with m protections if there exist at least m protected events in every path reaching secrets from the plant's initial state q_0.

In addition to the partition of secret states and nominal states, we consider two classes of events in the plant: *protectable events* and *unprotectable events*. This classification reflects that system administrators are not necessarily able to install protections everywhere in the plant. The subsets of protectable events and unprotectable events are denoted by Σ_p and Σ_{up} respectively. Note that every event belongs to either Σ_p or Σ_{up}, namely $\Sigma = \Sigma_p \dot{\cup} \Sigma_{up}$. For the plant in Fig. 1, $\Sigma_p = \{\sigma_0, \sigma_1, \sigma_4, \sigma_6, \sigma_7, \sigma_8, \sigma_9\}$ and $\Sigma_{up} = \{\sigma_2, \sigma_3, \sigma_5, \sigma_{10}\}$. In general, Admin does not need to pass authentication to become a standard user because of its higher permission than User, thus σ_2 is unprotectable. Moreover, σ_3 and σ_5 are unprotectable because the application of our example plant does not have an authentication function. The log-off function can generally be invoked by the system users without any permissions, and can also represent "disconnect" or "terminate", thus σ_{10} is unprotectable.

In order for administrators to effectively protect secret states in the plant, it must be determined where to set up protections. In other words, we need to determine which events should be protected. We call "which events to be protected at which states" a *protection policy*. That is, a protection policy specifies which event(s) to be protected each state in the plant. For example in Fig. 1, a protection policy may specify protecting event σ_0 at state q_0. If we are able to obtain a policy that effectively protects all secret states of the plant and the administrators follow it, then secret information can be protected. At least administrators can impose that intruders must always go through and unlock a predetermined number of protections in order to reach any secret states from the initial state. Whether or not this is hard for intruders depends on their capabilities, which we make no assumptions, nor do we have control over.

In real systems, however, administrators must also take into account the *costs* of protections, because there generally are no infinite budgets for protecting secrets, as well as because the plant typically have a limited capacity for security equipments. For example, purchasing biometrics devices to lock a door is often more costly than using a padlock, even though biometrics is a more secure way of locking than a padlock. As another example, complicated security applications which consume a large amount of memory cannot be installed in small embedded computers, in comparison with personal computers for general purposes. To represent that different protections have different costs, we partition the set of protectable events Σ_p according to the cost levels. For example, protectable events in Fig. 1 are divided into three subsets: $\Sigma_0 = \{\sigma_0, \sigma_1, \sigma_4\}$, $\Sigma_1 = \{\sigma_6, \sigma_7, \sigma_8\}$, $\Sigma_2 = \{\sigma_9\}$. Events in Σ_0 represent logging-in functions; thus these can be protected by passwords of users. Meanwhile, specific security configurations of the system or application are necessary to protect events in Σ_1; thus protecting events in Σ_1 is more costly than setting up passwords as in Σ_0. Also consider that σ_9 represents an internal access to the secret state q_8; thus it is necessary to implement a highly secured authentication technique in the application so as to protect σ_9, which consequently is the most costly. As the subscript, or *index*, of the subsets increases, the cost level becomes higher. For simplicity, we consider that the cost level of each subset of protectable events is not comparable, e.g. the cost to protect only σ_7 in Σ_1 is sufficiently larger than that to protect all events in Σ_0. This also means that the total cost of protecting secrets according to the derived protection policy is determined by the largest index of the subset of protectable events specified by the policy.

For convenience, we define Σ_p^k as the disjoint union of subsets of protectable events until index k (from index 0), namely $\Sigma_p^k := \Sigma_0 \dot{\cup} \Sigma_1 \dot{\cup} \ldots \dot{\cup} \Sigma_k$. We also define that the set of secret states Q_s is *m-securely reachable* with respect to Σ_p^k if every trajectory from the initial state to secret states contains *at least* m protectable events which belong to Σ_p^k. Based on the above discussion of secret states, event partition, and protection policy, our question here is "how can we find a protection policy (if one exists) to protect secrets with m protections and minimum protection

costs?". We formulate our problem by rephrasing this question.

Problem 1 (Secret Securing with Multiple Protections and Minimum Costs Problem, or m-SSMCP)**.** Given the plant with the partitions of states and events and a positive integer $m \geq 1$, find a protection policy (if it exists) such that the set of secret states Q_s is m-securely reachable with respect to Σ_p^k and k is the least index.

The least index k means that the largest index of the subsets of protectable events specified by the protection policy is minimum. For example in the plant of Fig. 1, $k = 1$ if every path from the initial state to secret states contains at least m protectable events in either Σ_0 or Σ_1.

3. Solution Existence and Computation

3.1 Solvability of m-SSMCP

The existence of the solution of m-SSMCP depends on the given plant and the given partition of events. In this section, we introduce a necessary and sufficient condition under which there exists a solution of m-SSMCP. Recall that m-SSMCP is a problem of finding a protection policy such that Q_s is m-securely reachable with the minimum protection costs. As can be seen from the definition of m-secure reachability, if there exists a trajectory containing fewer than m protectable events, then Q_s cannot be m-securely reachable. Thus a protection policy of m-SSMCP does not exist. Moreover, it can be observed that if Q_s is m-securely reachable with respect to Σ_p^k $(k > 0)$, then Q_s should not be m-securely reachable with respect to Σ_p^{k-1}, because k is required to be the least index. The following theorem provides a necessary and sufficient condition under which there exists a solution for m-SSMCP, namely m-SSMCP is solvable.

Theorem 1. m-SSMCP is solvable if and only if either of the following conditions holds:

1. Q_s is m-securely reachable w.r.t. Σ_0
2. Q_s is m-securely reachable w.r.t. Σ_p^k but not w.r.t. Σ_p^{k-1}

Condition 2 means that if $1 \leq k \leq n-1$, then secret states in Q_s can be protected with m protections by protecting the events in Σ_p^k, and some states in Q_s can be protected only with $m-1$ or fewer protections if we only protect the events in Σ_p^{k-1}. Condition 1 is similar to Condition 2, but if $k = 0$, then there does not exist Σ_p^{k-1}.

3.2 Policy Computation

When the necessary and sufficient condition in Theorem 1 holds, we compute a protection policy of m-SSMCP by converting it to a control problem and resorting to the SCT. First, we convert m-SSMCP to a corresponding control problem. Next, we derive a solution of the control porblem, namely a control policy, by employing the SCT, then have that solution revert to a solution of m-SSMCP, namely a protection policy.

In the SCT framework, we compute a supervisor which disables particular events so that the behavior of the target plant does not violate the given specification. In the control problem, generally there are two classes of events, controllable events and uncontrollable events. Events are called controllable if they can be disabled by the supervisor. Disabled events are never invoked in the plant. We change m-SSMCP to a supervisory control problem by interpreting the set of protectable events Σ_p as controllable events denoted by Σ_c, and the set of unprotectable events Σ_p as uncontrollable events denoted by Σ_{uc}. Σ_c inherits the same partition of Σ_p, namely $\Sigma_c^k = \Sigma_0 \dot{\cup} \Sigma_1 \dot{\cup} \ldots \dot{\cup} \Sigma_k$. Note that in real systems, secret states should still be reachable. It is not suitable to disable events to protect secrets because it can inhibit the nominal behavior of regular system users.

According to the conversion from the security problem to the control problem, the definition of m-securely reachability are changed to *m-controllable reachability*, that is, Q_s is m-controllably reachable with respect to Σ_c^k if every path from the initial state to secret states contains at least m controllable events which belong to Σ_c^k. Also, a protection policy corresponds to *a control policy* indicating the decision of which events to be disabled by the supervisor. Therefore, the following formulation of the control problem is derived from Problem 1.

Problem 2 (Reachability Control with Multiple Controllable Events and Minimum Costs Problem, or m-RCMCP)**.** Given the plant with the partitions of states and events and a positive integer $m \geq 1$, find a control policy (if it exists) such that the set of secret states Q_s is m-controllably reachable with

respect to Σ_c^k and k is the least index.

In the same way, the necessary and sufficient condition under which m-RCMCP is solvable is derived from the conditions in Theorem 1.

Corollary 1. m-RCMCP is solvable if and only if either of the following conditions holds:

1. Q_s is m-controllably reachable w.r.t. Σ_0
2. Q_s is m-controllably reachable w.r.t. Σ_c^k but not w.r.t. Σ_c^{k-1}

To utilize the SCT to compute a solution of m-RCMCP in Problem 2, we design the control specification automaton. This is done by removing from the plant automaton all secret states and the transitions to and from removed secret states. This implies that we want to make secret states unreachable from the initial state, as a solution to the supervisory control problem. Fig. 2 depicts

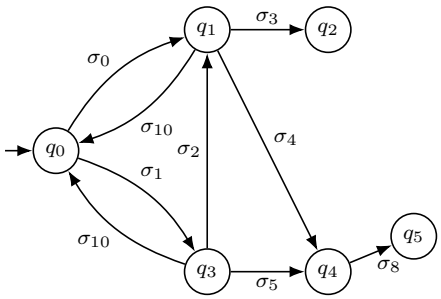

Fig. 2: Specification Automaton for Fig. 1

the specification automaton for the plant in Fig. 1, which is derived by removing the secret states q_6, q_7, and q_8. The transitions connected to these states, (q_2, σ_6, q_6), (q_6, σ_7, q_7), (q_2, σ_7, q_7), (q_7, σ_8, q_5), (q_5, σ_9, q_8), (q_6, σ_{10}, q_0), (q_7, σ_{10}, q_0) and (q_8, σ_{10}, q_0) are also removed.

Given the plant and the specification designed above, we compute by the SCT a supervisor to determine which controllable events should be disabled so that the secret states are unreachable. We point out that the supervisor computed by the SCT allows the *largest* behavior of the plant as long as the specification is not violated. For example, in Fig. 1, if reaching q_1 is not followed by eventually leaving the states in the specification in Fig. 2, namely eventually reaching secret states, by uncontrollable events, then σ_0 is not disabled by the supervisor. The supervisor does not exist, if the plant always leaves the realm of the specification by uncontrollable events, resulting in that a control policy for m-RCMCP does not exist. This means that we cannot prevent the plant from reaching prohibited states in this case the secret states. By the SCT, one supervisor only specifies a single event of each trajectory reaching secret states from the initial state. Therefore, we need to compute m supervisors which specify controllable events differently, to determine m distinct events on each path from the initial state to secret states.

For example, consider $m = 2$. To compute the second supervisor (SUP2) that specifies different controllable events from the first supervisor (SUP1), we relabel controllable events already specified by the control policy derived from SUP1 as uncontrollable events. For instance, if event σ_0 ($\in \Sigma_c$) at state q_0 is specified by the control policy of SUP1 to be disabled, then we add uncontrollable event σ_0' to Σ_{uc} and relabel σ_0 *at state* q_0 as σ_0'. There is no need to remove σ_0 from Σ_c.

Based on the above discussions, we propose Algorithm 1 to compute a solution of m-RCMCP. Al-

Algorithm 1 RCMCm

Input: Automata of the plant and the specification; integer m
Output: m supervisors

for $i = 1, 2, \ldots, m$ **do**
  Compute the supervisor (SUPi) by RCMC1 with the plant and specification
  if SUPi exists **then**
    Derive a control policy (POLi) from SUPi
    Relabel controllable events specified by POLi as new uncontrollable events
    Derive the specification from the plant whose events are relabeled
    Set the relabeled plant and specification as the new automata of plant and specification
  else
    return Nothing
  end if
end for
return SUP1, SUP2, ..., SUPm

function RCMC1(plant, specification)
  for $k = 0, 1, \ldots, n-1$ **do** ▷ assuming n classes in Σ_c
    $\Sigma_c^k = \dot{\bigcup_{i=0}^{k}} \Sigma_i$
    Compute by the SCT a supervisor (SUP) which can disable events in Σ_c^k so that the plant does not violate the specification
    if SUP exists **then**
      return SUP
    end if
  end for
  return Nothing
end function

gorithm 1 returns supervisors if they exist with respect to the given plant and the specification. If we obtain m supervisors from Algorithm 1, then there exist corresponding control policies that solves m-RCMCP. From the construction of the specification, these m control policies specify different m controllable events on every path from the initial state to secret states. Thus we merge these m control policies derived from the m supervisors by Algorithm 1 into one control policy. Observe that the largest index of the subsets to which events specified the merged control policy belong is minimum, because the index in RCMC1 starts from 0 and is incremented by 1 at each iteration. Therefore, the control policy derived from policies of SUP1, ..., SUPm is a solution of m-RCMCP. It is also true that if Algorithm 1 returns nothing, then there does not exist a solution of m-RCMCP, and consequently there is no solution of m-SSMCP.

Finally as introduced in the beginning of this section, by interpreting disabled events as protected events, a solution of m-SSMCP is derived from the control policy which has been computed by Algorithm 1 and the merging of control policies.

4. Illustrating Example

In this section, we take the example in Fig. 1 again to demonstrate our solution for m-SSMCP where $m = 2$, namely 2-SSMCP.

First, 2-SSMCP is changed to 2-RCMCP. Recall that the set of controllable events of this plant are partitioned into three subsets: $\Sigma_0 = \{\sigma_0, \sigma_1, \sigma_4\}$, $\Sigma_1 = \{\sigma_6, \sigma_7, \sigma_8\}$, and $\Sigma_2 = \{\sigma_9\}$. From the event partition of the plant in Fig. 1 and the specification in Fig. 2, SUP1 in line 2 of Algorithm 1 ($i = 1$) exists and results in the following control policy:

- At state q_0, disable σ_0 and σ_1.

Based on this policy, we relabel σ_0 and σ_1 of controllable events at state q_0 to σ_0' and σ_1' of uncontrollable events. Thus from the plant and the specification after relabeling, SUP2 in line 2 of Algorithm 1 ($i = 2$) exists and results in the following control policy:

- At state q_2, disable σ_6 and σ_7.
- At state q_4, disable σ_8.

Finally by merging two policies and change disabling to protecting, we obtain the following protection policy as a solution of 2-SSMCP:

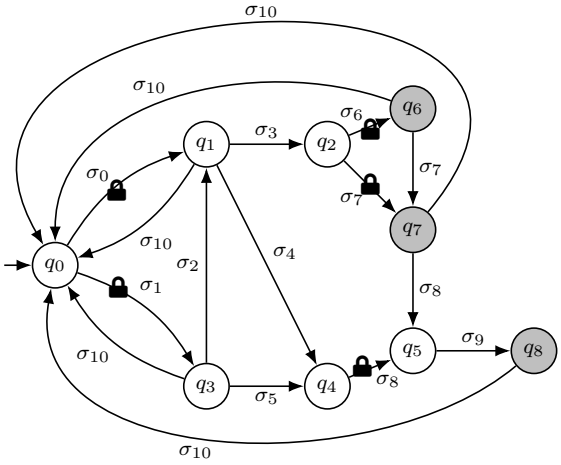

Fig. 3: Solution of 2-SSMCP w.r.t. the plant in Fig. 1

- At state q_0, protect σ_0 and σ_1.
- At state q_2, protect σ_6 and σ_7.
- At state q_4, protect σ_8.

Fig. 3 indicates the plant with the solution policy of m-SSMCP. "🔒" means that the event is protected.

5. Conclusions

We have introduced the problem of protecting secret states in the plant with at least m (≥ 1) protections and minimizing protection costs. This problem has been formulated as that of finding a protection policy such that every string leading to secret states from the initial state has at least m protectable events, and the protection costs are minimum simultaneously. We have presented an algorithm to provide a solution of the problem. This algorithm has been demonstrated with an illustrating example.

In real systems, secrets can belong to different classes of importance. For example, credit card numbers are generally more important than addresses of users in an e-commerce website, whereas both kinds of information should not be revealed to non-permitted users. In future work, we aim to address such a case by partitioning Q_s into subsets according to the importance levels of secrets.

（年月日受付）

参 考 文 献

1) A. Teixeira, D. Pérez, H. Sandberg and K. H. Johansson : Attack models and scenarios for networked control systems, Proceedings of the 1st international conference on high confidence networked systems, 55/64 (2012)

2) C. G. Cassandras and S. Lafortune : Introduction to Discrete Event Systems, Springer-Verlag (2008)
3) P. J. Ramadge and W. M. Wonham : Supervisory control of a class of discrete event processes, SIAM Journal on Control and Optimization, 25-1, 206/230 (1987)
4) W. M. Wonham and K. Cai : Supervisory Control of Discrete-Event Systems, Springer International Publishing (2019)
5) Y. Guo, X. Jiang, C. Guo, S. Wang and O. Karoui : Overview of Opacity in Discrete Event Systems, IEEE Access, 8, 48731/48741 (2020)
6) M. Ben-Kalefa and F. Lin : Supervisory control for opacity of discrete event systems, the 49th Annual Allerton Conference on Communication, Control, and Computing, 1113/1119 (2011)
7) A. Saboori and C. N. Hadjicostis : Notions of security and opacity in discrete event systems, Proceedings of 46th IEEE Conference on Decision and Control, 5056/5061 (2007)
8) A. Saboori and C. N. Hadjicostis : Verification of initial-state opacity in security applications of des, Proceedings of 9th International Workshop on Discrete Event Systems, 328/333 (2008)
9) Y. C. Wu and S. Lafortune : Comparative analysis of related notions of opacity in centralized and coordinated architectures, Discrete Event Dynamic Systems: Theory and Applications, 23-3, 307/339 (2013)
10) S. Matsui and K. Cai : Secret Securing with Minimum Cost, Proceeding of the 61st Japan Joint Automatic Control Conference, 1017/1024 (2018)
11) S. Matsui and K. Cai : Secret Securing with Multiple Protections and Minimum Costs, Proceedings of the 58th IEEE Conference on Decision and Control, 7635/7640 (2019)

［著　者　紹　介］

Shoma Matsui

Shoma Matsui received the B. Eng. degree in Electrical and Information Engineering from Osaka City University in 2019. He is currently a master student in Electrical and Computer Engineering at the University of Michigan, Ann Arbor. His research interests include secrecy, privacy, and safety of discrete-event systems, and applications to computer and network systems.

Kai Cai

Kai Cai received the B. Eng. degree in Electrical Engineering from Zhejiang University in 2006, the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto in 2008, and the Ph.D. degree in Systems Science from Tokyo Institute of Technology in 2011. He is currently an Associate Professor in Osaka City University. His research interests include distributed control of discrete-event systems and cooperative control of networked multi-agent systems.