Automatica 93 (2018) 369–378

Contents lists available at ScienceDirect

Automatica

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

Enforcement of opacity by public and private insertion functions☆

Yiding Ji [a,*], Yi-Chin Wu [a,b,1], Stéphane Lafortune [a]

[a] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA
[b] Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Received 13 October 2016
Received in revised form 26 October 2017
Accepted 3 February 2018

Keywords:
Discrete event systems
Privacy
Opacity
Opacity enforcement
Insertion function

ABSTRACT

We study the enforcement of opacity, an information-flow security property, using insertion functions that insert fictitious events at the output of the system. The intruder is characterized as a passive external observer whose malicious goal is to infer system secrets from observed traces of system events. We consider the problems of enforcing opacity under the assumption that the intruder either knows or does not know the structure of the insertion function; we term this requirement as public–private enforceability. The case of private enforceability alone, where the intruder does not know the form of the insertion function, is solved in our prior work. In this paper, we address the stronger requirement of public–private enforceability, that requires opacity be preserved even if the intruder knows or discovers the structure of the insertion function. We formulate the concept of public–private enforceability by defining the notion of public safety. This leads to the notion of public–private enforcing (PP-enforcing) insertion functions. We then identify a necessary and sufficient condition for an insertion function to be PP-enforcing. We further show that if opacity is privately enforceable by the insertion mechanism, then it is also public–private enforceable. Using these results, we present a new algorithm to synthesize PP-enforcing insertion functions by a greedy-maximal strategy. This algorithm is the first of its kind to guarantee opacity when insertion functions are made public or discovered by the intruder.

1. Introduction

Opacity is an information-flow security property that characterizes whether or not "secrets" of a given dynamic system can be inferred by an outside observer termed the *intruder*, because of its potentially malicious intentions. Due to its general formulation that is applicable to many security and privacy issues that arise in networked systems, opacity has received a lot of attention in the literature on security and privacy since it was first introduced in Mazaré (2004). The intruder is an outside observer that knows the system structure and tries to infer the occurrence of the secret by passively observing the output of the system. The system is said to be opaque if for every behavior induced by the secret (termed *secret behavior*), there is another observationally-equivalent behavior that is not induced by the secret (termed *non-secret behavior*). When opacity holds, the intruder is never sure if the system's output corresponds to a secret or a non-secret behavior.

Various representations of the system secret have been considered in the study of opacity. These representations have led to the formalization of several notions of opacity for event-driven models of dynamic systems. In the context of automata models, the notions of initial-state opacity, current-state opacity, language-based opacity, K-step opacity and infinite step opacity, have been proposed; see, e.g., Cassez, Dubreil, and Marchand (2012), Lin (2011), Saboori and Hadjicostis (2012b, 2013) and Yin and Lafortune (2017). Opacity for infinite state systems is considered in Chédor, Morvan, Pinchinat, and Marchand (2015) while opacity under so-called Orwellian observers is investigated in Mullins and Yeddes (2014). Opacity for Petri net models has been considered in Bryans, Koutny, and Ryan (2005) and Tong, Li, Seatzu, and Giua (2017b), among others. In addition, several stochastic notions of opacity have been defined and investigated; see, e.g., Bérard, Chatterjee, and Sznajder (2015), Bérard, Mullins, and Sassolas (2015), Chen, Ibrahim, and Kumar (2017), Keroglou and Hadjicostis (2013) and Saboori and Hadjicostis (2014). In Yin, Li, Wang, and Li (2017), an algorithm was proposed for verification of infinite-step opacity in stochastic discrete event system. The recent

☆ This work was partially supported by NSF grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and CNS-1421122, and by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The material in this paper was partially presented at the 54th IEEE Conference on Decision and Control, December 15–18, 2015, Osaka, Japan. This paper was recommended for publication in revised form by Associate Editor Christoforos Hadjicostis under the direction of Editor Christos G. Cassandras.

* Corresponding author.
E-mail addresses: jiyiding@umich.edu (Y. Ji), ycwu@umich.edu (Y.-C. Wu), stephane@umich.edu (S. Lafortune).
[1] Current address of the second author: Pure Storage, Mountain View, CA 94041, USA.

https://doi.org/10.1016/j.automatica.2018.03.041
0005-1098/

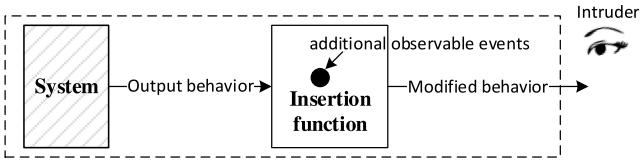

Fig. 1. The insertion mechanism.

survey paper (Jacob, Lesage, & Faure, 2016) may be consulted for a detailed review of the literature on this topic.

When a given notion of opacity is violated, researchers have proposed various methods for its enforcement. One popular approach is to design a minimally restrictive supervisor, which disables behaviors that violate opacity (Darondeau, Marchand, & Ricker, 2015; Dubreil, Darondeau, & Marchand, 2010; Saboori & Hadjicostis, 2012a; Takai & Oka, 2008). The work in Tong, Li, Seatzu, and Giua (2017a) adopts a similar approach but focuses on enforcing opacity with incomparable observations. The approach in Yin and Lafortune (2016) is to embed in a finite structure all feasible supervisors that enforce opacity and use this structure to synthesize one supervisor with desired properties. The work in Zhang, Shu, and Lin (2015) also lies in this category but discusses the problem from the perspective of maximum information release. Several works, such as Cassez et al. (2012), Yin and Lafortune (2015) and Yin and Li (2018), apply a sensor activation framework to enforce opacity by building dynamic observers or most-permissive observers. In Ylies and Hervé (2015), the authors consider a delay mechanism, which enforces K-step opacity or infinite-step opacity by delaying outputting system events until the secret expires.

In contrast to the above approaches for enforcing opacity, we proposed in our prior work (Wu & Lafortune, 2014) an insertion mechanism that enforces opacity by inserting fictitious events at the system's output. Such events are assumed to be indistinguishable from genuine ones from the viewpoint of the intruder. As in Wu and Lafortune (2014), our approach in this paper considers event-driven dynamic systems modeled as automata. Specifically, the system is a partially-observed and/or nondeterministic finite-state automaton, and the secret is modeled as a set of secret states in the automaton's state space. The insertion mechanism, which is depicted in Fig. 1, acts as an interface at the output of the system; hence, it does not interfere with the system, in contrast to the supervisory control based approaches. Insertion functions can be generalized to edit functions that allow event erasure and replacement, as considered in Ji and Lafortune (2017) and Wu, Raman, Rawlings, Lafortune, and Seshia (2017). However, we focus on insertion functions in this paper. The method of insertion functions has also been extended in Ji, Yin, and Lafortune (2018) to study opacity enforcement under quantitative constraints.

In Wu and Lafortune (2014), it is assumed that the insertion function used by the system is always kept private from the intruder. With this assumption, we have shown how to synthesize insertion functions that only output strings consistent with the non-secret behavior of the system and thus prevent the intruder from being certain that a secret behavior has occurred. In this paper, we relax that assumption. While the implementation of the insertion function may be kept private at first, a sophisticated intruder may learn the full set of modified behaviors output by the insertion function, compare it with the system model, and potentially reverse engineer the insertion function. Also, if the intruder knows the system's optimality criteria, it may follow the optimal synthesis algorithm in Wu and Lafortune (2016) and discover the correct insertion function. It may also be the case that the system designers decide to make the insertion function public, as is done in public-key cryptography, for example. *Hence, there is a need to design insertion functions that enforce opacity even when their implementation becomes known to the intruder.* Under the same insertion mechanism as in Fig. 1, to enforce opacity regardless of whether or not the intruder knows the implementation of the insertion function, we formally characterize a property called *public-and-private enforceability*, or *PP-enforceability* for short. A PP-enforcing insertion function is guaranteed to enforce opacity when the insertion function is kept private *and* when it becomes known to the intruder. In the former case, the insertion function outputs only behaviors consistent with non-secret behaviors of the system. In the latter case, the insertion function is designed such that for every secret behavior of the system, there is a non-secret behavior of the system that has the same modified output from the insertion function.

The main contributions of this paper are as follows. First, we formally characterize the properties of public enforceability and of public–private enforceability, in the context of opacity enforcement by insertion functions. We present conditions for PP-enforceability and use them to derive an effective test under which opacity is public–private enforceable. It turns out that if there exists an insertion function that is privately enforcing, then there also exists a (potentially different) insertion function that is PP-enforcing. This result is established by defining a so-called greedy criterion for selecting insertion functions in the All Insertion Structure (AIS) introduced in Wu and Lafortune (2014). These new results lead to an algorithmic procedure, called Algorithm INPRIVALIC-G, that is guaranteed to synthesize a PP-enforcing insertion function if one exists.

The remaining sections of this paper are organized as follows. Section 2 introduces the system model and the notion of opacity. Section 3 formally introduces insertion functions and the notion of *public-and-private enforceability*, along with conditions under which private enforceability and public–private enforceability hold for a given insertion function. Section 4 starts by reviewing the construction procedure of the All Insertion Structure (AIS) from Wu and Lafortune (2016) and then identifies relevant concepts and properties. In Section 5, we first present a sufficient condition for insertion functions to be PP-enforcing, then define the greedy criterion and show that a greedy insertion function is PP-enforcing. Then, in Section 6, the INPRIVALIC-G Algorithm is presented, which synthesizes PP-enforcing insertion functions by using a greedy-maximal insertion criterion within the AIS. Finally, Section 7 concludes the paper.

Preliminary versions of some of the results in sections 3.3 and 5.1 appear in Wu and Lafortune (2015). The results in Sections 4.2, 5.2 and 6 are new and do not appear in Wu and Lafortune (2015). In particular, Algorithm INPRIVALIC-G of Section 6 is guaranteed to output a PP-enforcing insertion function (if one exists) and is a generalized and improved version of Algorithm INPRIVALIC in Wu and Lafortune (2015), which outputs such a function only under certain conditions.

2. Opacity notions for automata models

We consider opacity problems in event-driven dynamic systems. We assume the system's state space is finite. Thus, the dynamic system of interest is modeled as an automaton $G = (X, E, f, X_0)$, where X is the finite set of states, E is the finite set of events, f is the partial state transition function $f : X \times E \to 2^X$, and $X_0 \subseteq X$ is the set of initial states. We allow G to be nondeterministic, which explains why the codomain of f is the power set of X. The transition function is extended to domain $X \times E^*$ in the standard manner (Cassandras & Lafortune, 2008); we still denote the extended function by f. Also, we use the notation $s < u$ to denote that string s is a prefix of string u. In opacity problems,