# Dissertation Abstract

Yi-Chin Wu
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor
Email: ycwu@umich.edu

Cybersecurity is an increasingly important issue as computers and networks are integrated into every aspect of our lives. Many methods for combating cyber attacks have been developed. However, these methods often rely on experiences or cannot be easily adapted to different application domains. We need a *science of cybersecurity*, which provides theoretical foundations for designing secure systems without depending on specific application domains, and which can be used to predict attacks that exploit previously unknown vulnerabilities.

My doctoral research addresses security problems based on the control theory for Discrete Event Systems (DES). We leverage models and techniques from DES to formally analyze security, design security enforcement strategies, and perform optimal control. Central to my research is a notion called *opacity*, which is a general information-flow property that can capture other existing properties such as secrecy and anonymity. We use finite-state automata (FSA) to describe the behaviors of computer systems that need to be rendered opaque with respect to a given secret. Under the passive observation of the intruder, the secret of the system is opaque if "whenever the secret has occurred, there exists another *non-secret* behavior that is observationally equivalent."

My doctoral research involves three aspects: verification, enforcement, and optimal control for opacity.

**Formulation and Verification of Various Notions of Opacity**

To analyze security, one needs to clearly define what is the information of the system that we want to hide from the intruder, i.e., the *secret*, and the intruders' power to infer the system's secret. The secret in the study of opacity has been defined by states of the system, sequences of actions, or the combination of the two, which yields various notions of opacity. My doctoral research focuses on four notions of opacity: current-state opacity (CSO), initial-state opacity (ISO), language-based opacity (LBO), and a notion called initial-and-final-state opacity (IFO) that we introduce in [3]. The secret for CSO is the system's current state; the secret for ISO is the initial state of the system; LBO considers sequence of events; and IFO requires the initial and the current state to remain hidden simultaneously. We leverage an existing algorithm for verifying ISO to verify IFO. We also develop a more efficient algorithm that reduces the complexity for verifying ISO from $O(2^{|X|^2})$ to $O(2^{|X|})$. Furthermore, we develop polynomial transformations between all pairs of them. These transformations allow us to verify one opacity notion by transforming it to another opacity notion, and also provide a basis for developing opacity enforcement mechanisms that work for all four notions.

Most prior works on opacity notions have assumed a single intruder. However, what if there are multiple

intruders and they collude? In [3], we also consider CSO, ISO, and IFO under a coordinated attack model where a coordinator aggregates all intruders' state estimates. For each notion, a characterization of the corresponding notion of "joint opacity" and an algorithmic procedure for its verification are provided.

**Opacity Enforcement Using Event Insertion**

If a secret is not opaque, how can we enforce the secret to be opaque? Prior works have used supervisory controllers to disable the system's behaviors that are going to reveal the secret, or use dynamic observers that dynamically modify the observability of every system event. However, the former approach needs to restrict the system behaviors; the latter may generate new observable behaviors and reveal clues about the defense model to the intruder. To address such limitations, in [2], we propose a novel enforcement mechanism based on event insertion. Specifically, an insertion function is a monitoring interface placed at the output of the system, as shown in Figure 1. It monitors the system's output behaviors and inserts fictitious observable events to the output without interacting with the system.

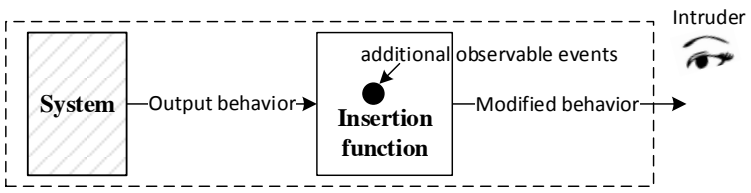


Figure 1: The insertion mechanism

The intruder is assumed to have no knowledge of the insertion function at the outset. But we want to make sure the intruder never suspects that an insertion function exists. Hence, fictitious events need to be inserted in a "convincing" manner – where the modified output is always consistent with an existing behavior that does not reveal the secret. We formulate the mathematical conditions for the interface requirement and the requirement for convincing insertions, and call insertion functions that satisfy both requirements as *i-enforcing*.

The major challenge here is how to automatically synthesize one i-enforcing insertion function; that is, how to design an insertion strategy such that all modified outputs are consistent with the original system while no system behavior is blocked. In many cases, an insertion is valid for the current system output but may become invalid later after the system outputs more events in the future. To solve this synthesis problem, we first solve the problem of whether or not there exists an i-enforcing insertion function by constructing the "All Insertion Structure" (AIS). The AIS, as it is called, is a finite structure that enumerates all i-enforcing insertion functions. It enumerates them in a game structure, as one can interpret the interactions between the system and the insertion function as a game where the insertion function tries to react to the output from the system. The idea for constructing the AIS is to first enumerate all valid insertions for the current system output and then prune away insertions that turn invalid after we consider the system's future behaviors. One

can view the AIS as containing all the "winning strategies" for the insertion function; that is, all deterministic i-enforcing insertion functions.

Because the AIS contains only the "winning strategies", it can be an empty automaton if no insertion function satisfies the i-enforcing property. We prove that there exists an i-enforcing insertion function if and only if the AIS is not the empty automaton. Finally, because the AIS contains all the "winning strategies", one can synthesize an i-enforcing insertion function by letting the system play all its actions (to consider all system behaviors) and selecting one insertion action in the game structure.

**Optimal Enforcement of Opacity Properties**

An insertion function inserts fictitious events to the output of the cyber system in order to enforce opacity. The inserted events cause overhead that consumes, for instance, power and/or bandwidth. In [4], we are interested in synthesizing an "optimal" insertion function that minimizes the overhead cost introduced by inserted events. We use the AIS as the structure to consider this optimal synthesis problem, as it enumerates all valid insertion functions. We assign to each inserted event a non-negative integer cost value. It turns out that an insertion function may need to insert an infinite number of events and thus incur an infinite total cost. With this consideration, we solve in [4] two optimization problems, one with respect to the *maximum total cost* and the other with respect to the *maximum mean cost*. The former captures the total insertion cost and the latter considers the average insertion cost (per system output), both in the worst-case scenario.

We first compute on the AIS the optimal *maximum total cost* for insertion functions. If this value is finite, we synthesize an optimal *total-cost* insertion function. Otherwise, we construct an optimal *mean-cost* insertion function. As the AIS is a game structure, the synthesis of an optimal insertion function is formulated as finding an optimal insertion strategy on the AIS. Depending on whether the total cost or the mean cost is considered, the synthesis procedure is *multiple minmax games* or *multiple mean payoff games* between two players. Specifically, because the system must execute *all* of its actions instead of selecting one, there is one minmax or mean payoff game per system action. Finally, in either case, an algorithmic procedure that encodes the optimal strategy in a finite-state I/O automaton is provided.

**Ensuring Privacy in Location-Based Services Based on Opacity Enforcement**

We consider the application of Location-Based Services (LBS) and formulate the issue of hiding the user's accurate location from the LBS server as an opacity problem in DES. Nondeterministic automata are used to model the user's moving patterns. In particular, the states are point locations on the physical map, and transitions captures the movement between point locations. Transitions are labeled by the location information in the LBS queries sent to the server because the LBS server is assumed to be the only malicious intruder.

The most popular existing technique for protecting LBS privacy is the use of *location anonymizers*. However, many works have argued that this technique is insufficient for enforcing LBS privacy when users continuously make queries. We are able to illustrate this argument by leveraging the opacity verification technique

on the automaton model that represents moving patterns. To enforce location privacy, we propose to add to the anonymizer an *insertion function* that inserts fictitious queries to the user's original queries. The design of such an insertion function follows the synthesis algorithms for an i-enforcing insertion function we developed in [2]. Furthermore, an *optimal* insertion function that minimizes the overhead cost from insertions can be designed by following the optimization procedure in [4]. Overall, the insertion mechanism fits well in the framework of LBS applications as insertion functions insert fictitious queries and drop their replies without affecting the quality of the LBS servers' replies to real queries.

# References


[1] Y.-C. Wu, Sankararaman K. A., and S. Lafortune. Ensuring privacy in location-based services: An approach based on opacity enforcement. *Submitted to the 14th International Workshop of Discrete Event Systems*, 2013.

[2] Y.-C. Wu and S. Lafortune. Enforcement of opacity properties using insertion functions. In *Proc. of the 51th IEEE Conference on Decision and Control*, pages 6722–6728, December 2012.

[3] Y.-C. Wu. and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, September 2013.

[4] Y.-C. Wu and S. Lafortune. Synthesis of optimal insertion functions for opacity enforcement. *Submitted to Transactions on Automatic Control*, 2013.