Synthesis of Event Insertion Functions for Enforcement of Opacity Security Properties

Stéphane Lafortune

EECS Department, University of Michigan, USA

joint work with Yi-Chin Wu

http://web.eecs.umich.edu/~stephane/
http://www-personal.umich.edu/~ycwu/

ExCAPE Webinar, Feb. 3, 2014
“Opacity” is a security notion that captures whether a secret of a system can be inferred by an intruder based on its observation of the system behavior and its knowledge of the system structure. Originated in [Mazaré, 2003] to analyze cryptography protocols. Example used in this talk: Location-Based Services.
Outline

1 Opacity: Definition and Enforcement
   - Opacity Problem Settings
   - Preliminaries and Notations
   - Opacity Notions
   - When the System is Not Opaque
   - Problem Definitions

2 Solution Methodology: Application to Location-Based Services
   - Location-Based Services
   - Verification of Current-Location Opacity
   - Existence of Opacity-Enforcing Insertion Functions
   - Synthesis of An Optimal Insertion Function

3 Discussion and Conclusion
Opacity Problem Settings

- The system is a **partially observable** finite-state automaton $G$
- The system has a **secret**
- The **intruder** is an observer that knows the system structure

![Diagram of system and intruder](https://via.placeholder.com/150)

The secret is opaque if for every secret behavior, there is a "non-secret" behavior that is observationally-equivalent.
The system is a partially observable finite-state automaton $G$.

The system has a secret.

The intruder is an observer that knows the system structure.

The secret is opaque if for every secret behavior, there is a “non-secret” behavior that is observationally-equivalent.
Preliminaries and Notations

Given automaton $G = (X, E, f, X_0)$

Language

- $\mathcal{L}(G) := \{ s \in E^* : (\exists i \in X_0)[f(i, s) \text{ is defined}]\}$
- $\mathcal{L}(G, i) := \{ s \in E^* : [f(i, s) \text{ is defined}]\}$

Projection map $P$

- System is partially observable $E = E_o \cup E_{uo}$
- Projection map $P : E^* \rightarrow E_o^*$
  - $P(e) = e$ if $e \in E_o$; $P(e) = \varepsilon$ if $e \in E_{uo} \cup \{\varepsilon\}$. 

\[ \begin{array}{c}
\text{a} \quad \text{b} \\
\text{a} \quad \text{a} \\
\text{P} \text{P} \\
\varepsilon \quad \text{E}_o = \{\text{a}\} \\
\text{E}_{uo} = \{\text{b}\} \\
\text{a} \quad \text{a} \\
\end{array} \]
What is the Form of the Secret?

- Set of current states (Current-State Opacity)
- Set of initial states (Initial-State Opacity)
- Sublanguage (Language-Based Opacity)
- Set of initial-final state pairs (Initial-and-Final-State Opacity)

There exist polynomial transformations between these notions [Wu and Lafortune, 2013a]
What is the Form of the Secret?

- **Set of current states (Current-State Opacity)**
- Set of initial states (Initial-State Opacity)
- Sublanguage (Language-Based Opacity)
- Set of initial-final state pairs (Initial-and-Final-State Opacity)

There exist polynomial transformations between these notions
[Wu and Lafortune, 2013a]
Current-State Opacity (CSO)

Definition

Given $G = (X, E, f, X_0)$, projection map $P$, and set of secret states $X_S \subseteq X$, $G$ is current state opaque if $\forall i \in X_0, \forall t \in L(G, i)$ s.t. $f(i, t) \subseteq X_S$, $\exists j \in X_0, \exists t' \in L(G, j)$ s.t. (i) $f(j, t') \cap (X \setminus X_S) \neq \emptyset$ and (ii) $P(t) = P(t')$.

![Diagram of CSO and non-CSO systems](image)
Verification of Opacity

- Current-State Opacity: [Cassez et al., 2012]
- Initial-State Opacity: [Saboori and Hadjicostis, 2008]
- Language-Based Opacity: [Cassez et al., 2012, Lin, 2011]
- Initial-and-Final-State Opacity: [Wu and Lafortune, 2013a]

There is no polynomial time test for opacity [Cassez et al., 2012]
Maximally permissive opacity-enforcing supervisory controllers
[Dubreil et al., 2010, Saboori and Hadjicostis, 2012]

- Disable transitions by feedback control when the secret is going to be revealed
- Limitation: need to interfere the system’s behavior

\[ E = E_c \cup E_{uc} \text{ and } E = E_o \cup E_{uo} \]

- Given: \( G, E_c, E_o, X_S \)
- Synthesize: supervisor \( S \) such that \( S/G \) is: opaque and maximally permissive
Opacity Enforcement Using Insertion Functions

Our approach: Leave system intact but add “interface” at its output

Enforce opacity using insertion functions [Wu and Lafortune, 2012]:
- Change the system’s output behavior by inserting observable events, without modifying the system’s behavior
  - Inserted events and genuine observable events are indistinguishable
Specifications for Insertion Functions

[Assumption] The intruder has no knowledge of the insertion function

- **Admissible**: allows all system’s output behavior
- **Safe**: modified behavior must always be observationally equivalent to an existing non-secret behavior
The Insertion Opacity Enforcement Mechanism

- Insertion automaton is a **finite encoding** of an insertion function
Problem 1: Synthesis of Insertion Functions

Opacity specification

Synthesis Algorithm

Provably correct opacity-enforcing insertion function
Problem 2: Optimal Synthesis of Insertion Functions

Opacity specification

Optimal Synthesis Algorithm

Optimal opacity-enforcing insertion function

Cost criterion
Outline

1 Opacity: Definition and Enforcement
   - Opacity Problem Settings
   - Preliminaries and Notations
   - Opacity Notions
   - When the System is Not Opaque
   - Problem Definitions

2 Solution Methodology: Application to Location-Based Services
   - Location-Based Services
   - Verification of Current-Location Opacity
   - Existence of Opacity-Enforcing Insertion Functions
   - Synthesis of An Optimal Insertion Function

3 Discussion and Conclusion
Location-Based Services (LBS)

Attack model for the intruder:
- Is located at the LBS server
- Has statistical mobility patterns of users
- Receives location information in LBS queries
Location-Based Services (LBS)

Attack model for the intruder:
- Is located at the LBS server
- Has statistical mobility patterns of users
- Receives location information in LBS queries

How can we prevent the intruder from knowing that the user is at a secret location?
The Anonymizer Technique

First proposed in [Gruteser and Grunwald, 2003]

\[ <L> = <loc_1, \ldots, loc_k> \]

- cloaking query
- answer to \( <L> \)

User \xleftarrow{\text{answer to}} <loc> \rightarrow \text{Location Anonymizer}

User \xleftrightarrow{\text{answer to}} <loc>

LBS Server
The Anonymizer Technique

First proposed in [Gruteser and Grunwald, 2003]

Insufficient when the intruder tracks the user’s continuous queries [Bettini et al., 2005]
The Anonymizer Technique

First proposed in [Gruteser and Grunwald, 2003]

Insufficient when the intruder tracks the user’s continuous queries [Bettini et al., 2005]

Our approach: Formulate “Current-Location Opacity” problem and use insertion functions to enforce opacity (Joint work with K. A. Sankararaman [Wu et al., 2013])
Workflow

Mobility patterns → Extract automaton model → G → Current-location opaque?

Y: Done!

N: Existence of opacity-enforcing insertion functions?

Y: Insertion function synthesis → Optimal insertion function synthesis

N: Designer

Solution Methodology: Application to Location-Based Services
Location-Based Services

Stéphane Lafortune
Synthesis of Event Insertion Functions for Enforcement of Opacity Security Properties 16 / 37
LBS Privacy Problems as Current-Location Opacity

- 8 locations as states
- Build transitions from the statistical mobility patterns
- Users can start from every building \((X_0 = X)\)
- The intruder observes the (source) region labels in the queries
- The secret location is the Cancer Center (state 6)
Nondeterministic Finite-State Automaton Model

Solution Methodology: Application to Location-Based Services

Stéphane Lafortune

Synthesis of Event Insertion Functions for Enforcement of Opacity Security Properties
Is it Current-Location Opaque?

No. The intruder is sure that the user is in state 6 (Cancer Center) when it observes query sequence \( cdd \).

We will synthesize an insertion function to enforce opacity.
Existence of Opacity-Enforcing Insertion Functions

- The design of an insertion function can be thought of as a game between the system and the insertion function.
- The insertion function tries to react to the system with insertion actions.

Idea: Enumerate all valid insertion functions in a game structure “The All-Insertion Structure (AIS)”
The All-Insertion Structure (AIS) Enumerates All Valid Insertion Functions

Identify all safe insertions:
Use insertions (RHS) to synchronize the safe behavior (LHS)
The All-Insertion Structure (AIS)  
Enumerates All Valid Insertion Functions

Let’s use a smaller $V$ to illustrate

$(m_i, m_j) = (\text{state estimate with insertion, genuine state estimate})$

This is the $V$ for system $G$:

\[
\begin{align*}
(m_0, m_0) &\quad m_0: \{0, 2\} \\
(m_1, m_1) &\quad m_1: \{1\} \\
(m_1, m_0) &\quad m_2: \{3\} \\
(m_2, m_2) &\quad m_3: \{4\}
\end{align*}
\]
The All-Insertion Structure (AIS) enumerates all valid insertion functions.

Does it include all of $P[\mathcal{L}(G)]$, i.e., are these insertions admissible?

$$\mathcal{L}(\text{system output}) = P[\mathcal{L}(G)]$$

**Theorem**

$$P_{und}[\mathcal{L}(V)] \neq P[\mathcal{L}(G)] \Rightarrow \text{No valid insertion function}$$
Enumerate all safe insertions in a finite game structure

\[ A = \{(m_0, m_0), (m_1, m_0), (m_2, m_0)\} \]
The All-Insertion Structure (AIS) Enumerates All Valid Insertion Functions

Enumerate all safe insertions in a finite game structure

\[ A = \{(m_0, m_0), (m_1, m_0), (m_2, m_0)\} \]
Prune away inadmissible insertions (iteration necessary)
Prune away inadmissible insertions (iteration necessary)
The All-Insertion Structure (AIS) Enumerates All Valid Insertion Functions

At convergence: All valid insertion functions are enumerated in the AIS

Theorem

Exists a valid insertion function ⇐⇒ the AIS is not the empty automaton
Synthesize One Insertion Function

- Information state “Y”: select all actions
- Information state “Z”: select one string
Synthesize One Insertion Function

- Information state “Y”: select all actions
- Information state “Z”: select one string

![Diagram of state transitions]

- (m₀,m₀), A
- (m₀,m₀), A, a
- (m₁,m₁), B
- (m₁,m₁), B, b
- (m₂,m₂), D
- (m₂,m₂), D, a
- (m₀,m₀), A, b
- (m₀,mₐ), A, a
- (m₂,m₃), C
- (m₂,m₃), C, a
- (m₂,m₃), C
- (m₂,m₃)
- (m₀,m₀)
- (m₁,m₁)
- (m₂,m₂)
- (m₃,m₄)

Cost criterion

Opacity specification

opacity

uo

b

a

Current-location

opaque? Optimal insertion function synthesis

Y

Extract information state “Z”

(m₁,m₁) (m₁,m₀)

(m₂,m₂) (m₂,m₁) (m₂,m₃) (m₂,m₀)

b b bb a a

a a a

(m₀,m₀)

a
Synthesize One Insertion Function

- Information state “Y”: select all actions
- Information state “Z”: select one string

Insertion automaton:

System output: insertion + system output
Existence of Opacity-Enforcing Insertion Functions

Let’s go back to the campus example: The AIS has 84 states, drawn using DESUMA [DESUMA Team, 2014]
In this example, every valid insertion function needs to continuously insert events

Assign costs to inserted events

Minimize the worst-case average cost (per system output)

Greedy algorithm does not always work

Apply algorithms for mean payoff games on weighted graphs
  [Zwick and Paterson, 1996]

See [Wu and Lafortune, 2013b] for more details
Find An Optimal Insertion Function

For each state, calculate the optimal worst-case average cost

**Algorithm 5**: Find the optimal mean cost

- **input**: $\text{AIS}^a = (Y \cup Z, E_o \cup 2E^r, f, y_0)$ and the weight function $w_o$
- **output**: $z^*$

1. Compute $V_h(y_0)$ for $h = 4n^3W_{\text{max}}$ using Equation (2)
2. Compute the $h$-step mean cost $V_h(y_0)/h$
3. Find the only rational number $r$ with a denominator at most $n$ that lies in the interval $[V_h(y_0)/h - \alpha, V_h(y_0)/h + \alpha]$ with $\alpha = \frac{1}{2n(n-1)}$
4. Return $2r$
For each insertion state: find the **optimal string** using “binary search”

**Algorithm 5**: Find the optimal mean cost

- **input**: AIS$^n = (Y \cup Z, E_o \cup 2E^+, f, y_0)$ and the weight function $w_a$
- **output**: $z^*$

1. Compute $V_h(y_0)$ for $h = 4n^3W_{max}$ using Equation (2)
2. Compute the $h$-step mean cost $V_h(y_0)/h$
3. Find the only rational number $r$ with a denominator at most $n$ that lies in the interval $[V_h(y_0)/h - \alpha, V_h(y_0)/h + \alpha]$ with $\alpha = \frac{1}{2n(n-1)}$
4. Return $2r$
Optimal Insertion Automaton

- Information state “Y”: select all actions
- Information state “Z”: select the optimal string

Combine states and build the insertion automaton...
Optimal worst-case average cost = 2

Modifies $cdd$ to $cd(c_ic_i)d$
Interpretation of Optimal Insertion Automaton

- Modifies $cdd$ to $cd(c_i c_i) d$
- Direct the intruder’s inference to $\{4, 7\}$ (Natural science or Medical school) while the true state is 6 (Cancer Center)

![Automaton Diagram]
Outline

1 Opacity: Definition and Enforcement
   - Opacity Problem Settings
   - Preliminaries and Notations
   - Opacity Notions
   - When the System is Not Opaque
   - Problem Definitions

2 Solution Methodology: Application to Location-Based Services
   - Location-Based Services
   - Verification of Current-Location Opacity
   - Existence of Opacity-Enforcing Insertion Functions
   - Synthesis of An Optimal Insertion Function

3 Discussion and Conclusion
Discussion

- Insertion functions can be stand-alone add-on interfaces without the anonymizer.
- This method can be applied to “location anonymity” where there is no secret state and the user wants to hide its current location.

Furthermore...

- What if the intruder knows the implementation of the insertion function? Synthesize an insertion function that enforces opacity regardless of whether the intruder knows the implementation.
- From passive to active intruders...
Conclusion

- Reviewed opacity security properties
- Proposed a new opacity enforcement mechanism based on event insertion
- Developed a formal procedure that synthesizes an (optimal) opacity-enforcing insertion function
  - Information states
  - All-Insertion Structure (AIS)
- Illustrated the results on the location-based services example
- *Synthesis, Synthesis, Synthesis!*
Bibliography I

Protecting privacy against location-based personal identification.
In Secure Data Management, pages 185–199. Springer.

Synthesis of opaque systems with static and dynamic masks.

DESUMA Team (2014).
DESUMA software tool.

Supervisory control for opacity.

Anonymous usage of location-based services through spatial and temporal cloaking.

Opacity of discrete event systems and its applications.
Automatica, 47(3):496–503.

Using unification for opacity properties.
Proc. of the 4th IFIP WG1, 7:165–176.
Bibliography II

Verification of initial-state opacity in security applications of DES.
*Proc. of the 9th International Workshop on Discrete Event Systems*, pages 328–333.

Opacity-enforcing supervisory strategies via state estimator constructions.

Enforcement of opacity properties using insertion functions.
In *Proc. of the 51th IEEE Conference on Decision and Control*, pages 6722–6728 (Journal version to appear in *Automatica*).

Comparative analysis of related notions of opacity in centralized and coordinated architectures.

Synthesis of optimal insertion functions for opacity enforcement.
Technical report, University of Michigan.

Ensuring privacy in location-based services: An approach based on opacity enforcement.
*Submitted to the 14th International Workshop of Discrete Event Systems*.

The complexity of mean payoff games on graphs.