Synthesis of Event Insertion Functions for Enforcement of Opacity Security Properties

Stéphane Lafortune
joint work with Yi-Chin Wu

EECS Department, University of Michigan, USA

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

Presentation at Georgia Tech
October 1, 2014
Research supported by NSF and TerraSwarm

- NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering (CCF-1138860)
  - https://excape.cis.upenn.edu/
  - Synthesis...

- 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
  - http://www.terraswarm.org/
  - Security and privacy...
What is Opacity?

- “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 cryptographic protocols.
- Example used in this talk: Location-Based Services.
Outline

1 Opacity: Definition and Enforcement
   - Opacity Problem Setting
   - 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
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 G and Intruder](image)

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 $P : E^* \to E_o^*$
  - $P(e) = e$ if $e \in E_o$; $P(e) = \varepsilon$ if $e \in E_{uo} \cup \{\varepsilon\}$.

\begin{align*}
E_o &= \{a\} \\
E_{uo} &= \{b\}
\end{align*}
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 Laforetune, 2013]

Other related opacity notions: $K$-step opacity, strong/weak opacity, concurrent opacity
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, 2013]

Other related opacity notions: $K$-step opacity, strong/weak opacity, concurrent opacity
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](image)

**The system is CSO**

$E_0 = \{b\}$

**The system is not CSO**

$E_0 = \{a, b\}$
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 Laafortune, 2013]

There is no polynomial time test for opacity [Cassez et al., 2012]
Maximally permissive opacity-enforcing supervisory controllers
[Dubreil et al., 2010, Ben-Kalefa and Lin, 2011,
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 \emph{and} maximally permissive
Dynamic observers [Cassez et al., 2012]
- Dynamically modify the observability of every event

Runtime enforcement mechanism using delays [Falcone and Marchand, 2013]
- Delay outputting event occurrences until the secret expires
Our approach: Leave system intact but add an “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 Setting
   - 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 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 mobility patterns of users
- Receives location information in LBS queries

Can the intruder know that the user is at a secret location?
The Anonymizer Technique

First proposed in [Gruteser and Grunwald, 2003]
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., 2014])
Workflow

Mobility patterns

→ Extract automaton model

G

→ Current-location opaque?

Y

→ Done!

N

→ Existence of opacity-enforcing insertion functions?

Y

→ Insertion function synthesis

N

→ Optimal insertion function synthesis

Designer
LBS Privacy Problems as Current-Location Opacity

- 8 locations as states:
  - 0 Social Work, 1 Ross School,
  - 2 Law School, 3 Law Library,
  - 4 Natural Science, 5 Chemistry,
  - 6 Cancer Center, 7 Medical Center

- Mobility patterns as transitions
- Region information of queries as event labels
- Users can start from any location \( X_0 = X \)
Nondeterministic Finite-State Automaton Model

Can the user keep location 6 private?
Is it Current-Location Opaque?

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

Synthesize a location insertion function to enforce location opacity
Synthesize A Location Insertion Function

- Existence problem: determine if one exists
  - All-Insertion Structure (AIS): enumerate all valid insertion functions in a game structure
  - A two-player structure between the system player and the insertion player
  - Game against nature
- Synthesis problem: synthesize an insertion function from the AIS
The All-Insertion Structure (AIS) Enumerates All Valid Insertion Functions

1. Compute safe insertions:
   An insertion (RHS) is safe if it can match a 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 $G$:
2. Enumerate all safe insertions in a finite game structure
2. Enumerate all safe insertions in a finite game structure
The All-Insertion Structure (AIS) Enumerates All Valid Insertion Functions

2. Enumerate all safe insertions in a finite game structure
2. Enumerate all safe insertions in a finite game structure
2. Enumerate all safe insertions in a finite game structure

"information states" of the system

"information states" of the insertion function
3. Prune away **inadmissible** insertions (iteration necessary)
3. Prune away inadmissible insertions (iteration necessary)
3. Prune away inadmissible insertions (iteration necessary)

Algorithm 4 Construct the AIS

Input: \( V_u = (Y \cup Z, E_o \cup M_{v,mo}, f_y, f_z, y_0) \)

Output: \( \text{AIS} = (Y \cup Z, E_o \cup M_{v,mo}, f_{\text{AIS},y}, f_{\text{AIS},z}, y_0) \)

1. Mark all the \( Y \) states in \( V_u \)
2. Let \( E_o \) be uncontrollable and \( M_{v,mo} \) be controllable
3. Trim \( V_u \) and let \( V_u^{trim} \) be the specification automaton
4. Obtain \( [\mathcal{L}_m(V_u^{trim})]^\dagger C \) w.r.t \( \mathcal{L}(V_u) \) by following the standard \( \dagger C \) algorithm in [3]
5. Return the resulting automaton representation of \( [\mathcal{L}_m(V_u^{trim})]^\dagger C \), which is a sub-automaton of \( V_u^{trim} \)

Supervisory Control Problem without Blocking
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 $\iff$ the AIS is not the empty automaton
Synthesize One Insertion Function

Synthesis problem: select insertions in a breadth-first search manner

- “System” states: select all actions
- “Insertion” states: select one action
Synthesize One Insertion Function

**Synthesis** problem: select insertions in a breadth-first search manner

- “System” states: select all actions
- “Insertion” states: select one string

![Diagram of state transitions](image-url)

- States: (m₀,m₀), (m₀,m₀), a, (m₀,m₀), b, (m₁,m₁), b, (m₁,m₁), a, (m₂,m₂), (m₂,m₂), a, (m₂,m₂), (m₂,m₂), (m₂,m₂), a, (m₂,m₂), (m₂,m₂), (m₂,m₂), a
- Edges: a, b, ε, a_1, a

- Pruning: ε edges indicate removable transitions.
Synthesize One Insertion Function

Merge states and build an insertion automaton:

![Diagram of an insertion automaton with states (m₀,m₀), (m₁,m₁), (m₂,m₂), and (m₂,m₃). The transitions are labeled with 'a/a', 'b/a/b', 'b/b', and 'a/a'. The system output is indicated as 'insertion + system output'.]
Find An Optimal Insertion Function

- Assign the same unit cost to every inserted event
- Is There an Insertion Function with a Finite Total Cost?
  - Yes: minimize the **worst-case total-cost**
  - No: minimize the **worst-case average cost** (per system output)
- See [Wu and Lafortune, 2014] for more details
Find An Optimal Insertion Function

- Assign the same unit cost to every inserted event
- Is There an Insertion Function with a Finite Total Cost?
  - Yes: minimize the worst-case total-cost
  - No: minimize the **worst-case average cost** (per system output)
- See [Wu and Lafortune, 2014] for more details
Cost Structure on the AIS

Let’s use a more complicated AIS to illustrate the optimization procedure:

- Define cost structure and perform optimization on the AIS

Replace the transition labels by the set of inserted strings

Rename the states by numbers

\[(a_ib_i)^* \cdot (b_ia_i)^* \cdot (c_ia_i)^* \cdot (a_ic_i)^* \cdot (c_ia_i)^* \cdot (a_ic_i)^* \cdot (c_ia_i)^* \cdot (a_ic_i)^* \cdot (c_ia_i)^* \cdot (a_ic_i)^* \cdot (c_ia_i)^* \]
Cost Structure on the AIS

- \(c(c_i) = 2, c(b_i) = 1\)
- For each edge
  - pick the string with the minimum cost
Cost Structure on the AIS

- $c(c_i) = 2$, $c(b_i) = 1$
- For each edge
  - pick the string with the **minimum cost**
  - assign the corresponding insertion cost
Find An Optimal Mean-Cost Insertion Function

Algorithm (adapted from [Zwick and Paterson, 1996])

- ∀ state, calculate the long-run average (determined by finite average)

[assume integer cost values]
Find An Optimal Mean-Cost Insertion Function

Algorithm (adapted from [Zwick and Paterson, 1996])
- \(\forall\) state, calculate the long-run average (determined by finite average)
- \(\forall\) insertion state, find the optimal action using “binary search”
Find An Optimal Mean-Cost Insertion Function

Algorithm (adapted from [Zwick and Paterson, 1996])

- ∀ state, calculate the long-run average (determined by finite average)
- ∀ insertion state, find the optimal action using “binary search”
- Combine states and build the insertion automaton:
  optimal mean cost = 1
Complexity Result

All-Insertion Structure (AIS)
- Enumerates all valid insertion functions
- Complexity $O(\|X_{obs}\|^2|E_o|)$ polynomial in the state space of the estimator

Synthesize an insertion function
- $n = |X_{AIS}|, m = |f_{AIS}|$
- Synthesize one insertion function: $O(m)$
- Synthesize an optimal total-cost insertion function: $O(n^2mW_{max})$
- Synthesize an optimal mean-cost insertion function:
  $O(n^4m \log \frac{m}{n} W_{max})$
Let’s Go Back to the LBS Privacy Problem
Existence of Opacity-Enforcing Insertion Functions

The AIS has 84 states ⇒ A valid insertion function exists
(drawn using DESUMA [DESUMA Team, 2014])
Optimal Location Insertion Function for the LBS Example

- 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)
Outline

1 Opacity: Definition and Enforcement
   - Opacity Problem Setting
   - 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.
Future Work

What if the intruder knows the implementation of the insertion function?

- $f_1^1(a) = d_ia, f_1^1(b) = a_ib$, when the intruder observes $ab$, it knows...
- synthesize an insertion function that can be publicly known: $f_1^2(a) = d_ia, f_1^2(b) = d_ia_ib$

- Consider stochastic and distributed opacity notions
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
- Many new interesting open research problems
Bibliography I

Supervisory control for opacity of discrete event systems.

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.

Runtime enforcement of K-step opacity.
In Proc. of the 52nd IEEE Conference on Decision and Control.

Anonymous usage of location-based services through spatial and temporal cloaking.
Opacity of discrete event systems and its applications.  

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

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.

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.

The complexity of mean payoff games on graphs.