Ensuring Privacy in Location-Based Services: An Approach Based on Opacity Enforcement [⋆]

Yi-Chin Wu [*] **Karthik Abinav Sankararaman** [**] **Stéphane Lafortune** [*]

[*] *Electrical Engineering and Computer Science, University of Michigan, Ann Arbor (e-mail: {ycwu;stephane}@umich.edu).*
[**] *Computer Science and Engineering, Indian Institute of Technology, Madras (e-mail: kabinav@cse.iitm.ac.in).*

Abstract: With the proliferation of mobile devices, Location-Based Services (LBS) that provide networked services based on users' locations have become increasingly popular. Such services, providing personalized and timely information, have raised privacy concerns such as unwanted revelation of users' current locations to potential stalkers. Many prior studies have proposed to address LBS privacy by sending "cloaking queries" that contain coarser location information. However, this method has been shown to be insufficient and no formal methodology exists for enforcing LBS privacy in mobile environments. In this work, we show that this problem can be formally addressed using the notion of opacity in discrete event systems. We use non-deterministic finite-state automata to capture the mobility patterns of users and label the transitions by the location information in the queries. Using opacity verification techniques, we show that the technique of sending cloaking queries to the server can still reveal the exact location of the user. To enforce location privacy, we apply the opacity enforcement technique by event insertion proposed in our prior work. Specifically, we synthesize suitable insertion functions that insert fake queries into the cloaking query sequences. The generated fake queries are always consistent with the mobility model of the user and provably ensure privacy of the user's current location. Finally, to minimize the overhead from fake queries, we design an optimal insertion function that introduces minimum average number of fake queries.

Keywords: Discrete-event systems, Opacity, Location privacy

1. INTRODUCTION

The development of network and mobile devices has stimulated the rapid growth of networked services based on users' locations. Such services, called Location-Based Services (LBS), provide personalized and timely information to users by exploiting their real-time location information. Examples of LBS applications include searching for nearby restaurants, recommending in-store coupons for nearby shops, and providing turn-by-turn navigation instructions.

While LBS provide much convenience to users, they have also raised security and privacy concerns. For example, an application called "Creepy" can serve as a cyberstalker that tracks a targeted individual's most recent location by monitoring his/her Twitter or Flickr activities. [1] Even if the attacker sees only depersonalized queries, it may still infer the senders by associating locations with individuals who exclusively correspond to the locations. Beresford and Stajano (2003) were able to correctly associate real-world identities with depersonalized pseudonyms by examining where each individual in the office building spends most of its time and which pseudonym visits a given place more often than others. The recent survey (Krumm, 2009) also discusses other existing computational threats and countermeasures.

In the attack model commonly used by the LBS privacy research community, the LBS servers are regarded as malicious observers. They use queries received from the users to infer sensitive information about the query senders. The LBS servers may have such malicious intent for commercial purposes, or because they have been compromised. Two privacy notions for LBS have been defined, depending on what type of information is of concern: location privacy and query privacy. Location privacy is the concealment of the user's real location. See, e.g., Kido et al. (2005); Duckham and Kulik (2005); Chow and Mokbel (2007); Shokri et al. (2010). Query privacy, on the other hand, is to seclude the user's private attributes such as his/her real identity. See, e.g., Gruteser and Grunwald (2003); Bettini et al. (2005); Mokbel et al. (2006); Chow and Mokbel (2007); Xu and Cai (2008); Pingley et al. (2011).

To protect both query and location privacy, the pioneering work of Gruteser and Grunwald (2003) proposed to use *location anonymizers*. As shown in Figure 1, a location anonymizer receives a query from the user and generalizes the accurate location in the original query to a *cloaking*

[⋆] This work was partially supported by the NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering (grant CCF-1138860).

[1] http://creepy.en.softonic.com/

region where $k-1$ other potential or active users reside. The use of location anonymizers has become the most popular technique for protecting privacy in LBS. Most anonymizers in the following work, such as (Duckham and Kulik, 2005; Mokbel et al., 2006), are designed for protecting *one-time* queries. They are, however, insufficient when users continuously make queries. The attackers, by tracking the user's continuous moving trajectory, can figure out the real user or real location in the given cloaking query if other candidates have inconsistent trajectories (Bettini et al., 2005). Some works such as (Xu and Cai, 2008; Pingley et al., 2011) have addressed privacy in such dynamic settings. Finally, we refer to Shin et al. (2012) for a comprehensive overview of the existing schemes for protecting LBS users privacy.

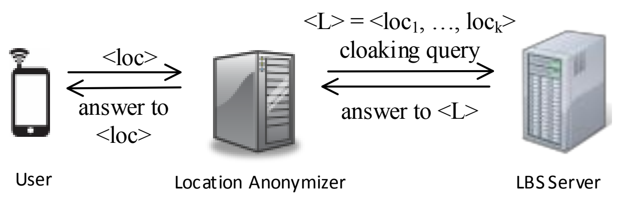

Fig. 1. The traditional anonymizer framework

In this paper, we focus on location privacy in LBS when users make continuous queries. Without loss of generality, we assume that the attacker is the LBS server. We also assume that it has statistical mobility patterns of users but is not aware of the users' real-time locations. It aims to infer a given user's real location by utilizing the location information received in the queries. Motivated by the nature of event-driven dynamics in LBS, we propose to analyze and enforce location privacy using a formal approach based on opacity techniques from Discrete-Event Systems (DES). Unlike many prior works, the approach we propose is model-based; specifically, we construct an automaton model that captures the moving patterns of users. To characterize location privacy, we adapt the notion of current-state opacity previously studied in the DES literature and introduce a related new notion called *current-state anonymity* that captures the observer's inability to know for sure the current state of the automaton. Because states of the automaton are locations in the moving patterns, location privacy is protected if and only if the constructed automaton has current-state anonymity.

A set of users' moving patterns on the Central Campus of the University of Michigan serves as a running example to demonstrate our methodology for privacy enforcement. First, a nondeterministic automaton is built from the moving patterns. We consider the anonymizer framework and label the transitions by the cloaking location information received by the LBS server. We then use the current-state estimator technique to verify current-state anonymity and show that the anonymizer is insufficient for protecting location privacy. To enforce location privacy, we propose to add to the anonymizer an *insertion function* that inserts fictitious queries to the cloaking queries from the anonymizer. As shown in Figure 2, the insertion function is placed at the interface between the anonymizer and the LBS server. It receives cloaking queries and inserts fictitious queries if the cloaking query sequence is going to reveal the user's real location. The design of a suitable insertion function follows the procedure for designing provably-correct insertion functions that we have recently developed (Wu and Lafortune, 2014). With such an insertion function, fictitious queries are inserted so that the resulting query sequence is always consistent with an existing moving pattern that does not reveal the user's real location. Finally, to minimize the overhead energy consumption or time delay caused by inserted queries, we follow the optimization procedure recently proposed in (Wu and Lafortune, 2013b) to design an *optimal* insertion function. Overall, the insertion mechanism of (Wu and Lafortune, 2014) 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.

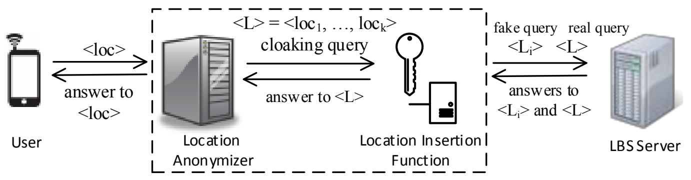

Fig. 2. The proposed location insertion mechanism

Using fake queries for location obfuscation has been proposed in the LBS privacy literature; see e.g., (Kido et al., 2005; Pingley et al., 2011). The work in (Kido et al., 2005) generates random fake queries without considering the user's mobility patterns, and this raises the question of how convincing fake queries are. On the other hand, the algorithm in (Pingley et al., 2011) relies on mobility patterns. However, it differs from this paper in that it considers query privacy instead of location privacy.

The contribution of this paper is two-fold. First, we introduce a formal methodology to LBS privacy studies. The method leverages theoretical results from the study of opacity in DES to analyze and provably enforce location privacy. Second, this paper provides a concrete application of opacity techniques to a real-world problem.

The remaining sections of this paper are organized as follows. Section 2 introduces the common LBS architecture and the privacy concerns. Section 3 reviews the basics of the opacity problem in DES. In Section 4, we construct an automaton model from a set of mobility patterns on the Central Campus of the University of Michigan; this example is then used as a running example in the remaining two sections. We show in Section 5 how to verify location privacy using techniques for opacity verification. Then, in Section 6, we present the construction of an optimal insertion function for enforcement of location privacy. Finally, Section 7 concludes the paper.

2. LOCATION-BASED SERVICES

A common LBS architecture, as illustrated in Figure 3, consists of four major components: mobile devices, positioning systems, communication networks, and the LBS server. The user makes queries from his/her mobile devices. The location information in the queries is obtained via positioning systems such as the Global Positioning System (GPS). To protect privacy, the user identification in the queries is replaced with an untraceable pseudonym.

The queries and their responses are transmitted between the user and the LBS server via communication networks.

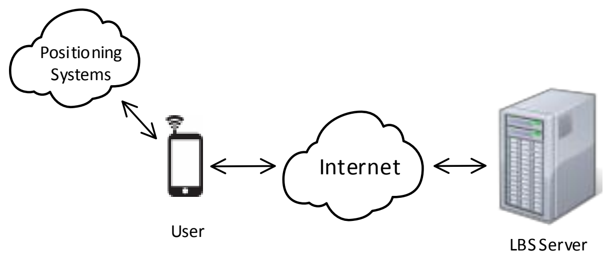

Fig. 3. The common LBS architecture

2.1 Privacy Concerns and the Attack Model

We assume the attack model that is commonly used in the LBS community. The LBS server is a malicious observer and other components are benign. Specifically, we consider one attack scenario discussed in (Shokri et al., 2010) where the server, i.e., the *attacker* in this paper, knows the statistical information of users' moving patterns but is not aware of users' real-time location. The attacker relies on the location information it receives to perform attacks.

Two types of privacy notions have been defined: location privacy and query privacy. Location privacy is to prevent inferring the real locations where queries are made from the queries themselves. Query privacy is to seclude the private attributes, such as the user's real identity, embedded in the queries. In this work, we assume that the attacker aims to associate a user query with its real location and focus on location privacy only.

2.2 The Anonymizer Framework

The location *anonymizer* is the most popular technique used in the literature that aims to solve both location and query privacy. A location anonymizer, as shown in Figure 1, generalizes the accurate location in a given user's original query to a *cloaking* region where $k-1$ other potential or active users reside. Implemented on a trusted third party, the anonymizer receives queries from users and forwards the cloaking queries to the LBS server.

This typical cloaking technique protects privacy for one-time queries only, but may fail in a dynamic environment where users continuously make queries (see, e.g., (Bettini et al., 2005)). In this paper, we will show this result by using opacity techniques from DES. We will also show how to enforce location privacy using the opacity enforcement technique developed in (Wu and Lafortune, 2014). Technical details will be presented in Sections 4, 5, and 6.

3. OPACITY IN DISCRETE EVENT SYSTEMS

Opacity is an information flow property that was first introduced in the computer science literature in (Mazaré, 2003). It has become an active research topic in DES, as this class of dynamic systems provides suitable formal models and analytical techniques for investigating such properties. The existing research on opacity include the verification and the enforcement of various opacity notions such as current-state opacity, initial-state opacity, and language-based opacity; see e.g., (Saboori and Hadjicostis, 2008; Lin, 2011; Cassez et al., 2012; Wu and Lafortune, 2013a).

3.1 Automata Models

In this paper, we use techniques from opacity problems in DES modeled as finite-state automata. The automaton, denoted by G, is nondeterministic in general. $G = (X, E, f, X_0)$ has a set of states X, a set of events E, a nondeterministic (potentially partial) state transition function (extended to strings) $f : X \times E^* \to 2^X$, and a set of initial-states X_0. Define $\mathcal{L}(G, X_i) := \{t \in E^* : (\exists x \in X_i)[f(x,t) \text{ is defined}]\}$. The language generated by G is the system behavior that is defined by $\mathcal{L}(G, X_0)$. For simplicity, when $X_i = \{x\}$, we write $\mathcal{L}(G, x)$. In general, the system is partially observable. Hence, the event set is partitioned into an observable set E_o and an unobservable set E_{uo}. Given an event $e \in E$, its observation is the output of the natural projection $P : E \to E_o$ such that $P(e) = e$ if $e \in E_o$ and $P(e) = \varepsilon$ if $e \in E_{uo}$ where ε is the empty string. With this definition at hand, projection P is extended from $E \to E_o$ to $P : E^* \to E_o^*$ in a recursive manner: $P(te) = P(t)P(e)$ where $t \in E^*$ and $e \in E$.

3.2 Opacity Problems

The settings of an opacity problem are: (1) G has a *secret*; (2) G is partially observable and/or nondeterministic; and (3) the attacker is an observer of G that has full knowledge of G but only observes G through projection P; namely, the attacker's observations are strings in $P[\mathcal{L}(G, X_0)]$. With the knowledge of G and its observations, the attacker infers the real behavior from the system by constructing estimates. The secret is said to be opaque if no attacker's estimate reveals the occurrence of the secret. That is, opacity holds if *for any secret behavior, there exists at least one other non-secret behavior that is observationally equivalent to the attacker.* Therefore, the attacker is not sure whether the secret or the non-secret has occurred.

Different definitions of the secret yield different notions of opacity. In current-state opacity (CSO), the secret is defined in terms of the current state of the system. Below we recall the formal definition of CSO:

Definition 1. (Current-State Opacity). Given system $G = (X, E, f, X_0)$, projection P, and the set of secret states $X_S \subseteq X$, current-state opacity holds if $\forall i \in X_0$ and $\forall t \in \mathcal{L}(G, i)$ such that $f(i,t) \subseteq X_S$, $\exists j \in X_0$, $\exists t' \in \mathcal{L}(G, j)$ such that: (i) $f(j, t') \cap (X \setminus X_S) \neq \emptyset$ and (ii) $P(t) = P(t')$.

To fit the application of location privacy in LBS, we adapt the CSO definition and propose a new notion called *Current-State Anonymity* (CSA).

Definition 2. (Current-State Anonymity). Given system $G = (X, E, f, X_0)$ and projection P, the system is current-state anonymous if $\forall i \in X_0$ and $\forall t \in \mathcal{L}(G, i)$ such that $f(i,t) = \{x\} \subseteq X$, $\exists j \in X_0$, $\exists t' \in \mathcal{L}(G, j)$, $\exists x' \neq x$ such that: (i) $x' \in f(j, t')$ and (ii) $P(t) = P(t')$.

Current-state anonymity can be thought as CSO with multiple pairs of secret and non-secret states. Specifically, a given system is CSA if it is CSO with respect to $X_S = \{i\}, X_{NS} = X \setminus \{i\}, \forall i \in X$. The similarity between the two notions allows us to use the current-state estimator of G, which is also the observer automaton of G as defined in (Cassandras and Lafortune, 2008), to verify CSA. Hereafter, we denote the current-state estimator of G by $\mathcal{E}_G$. The following proposition then follows immediately.

Proposition 3. A given automaton G is current-state anonymous if and only if no state in $\mathcal{E}_G$ is a singleton.

4. AUTOMATA MODELS FOR LBS

We consider the framework where the anonymizer is used. The attacker has statistical information about users' mobility patterns, but can only perform attacks using the cloaking information sent to the server.

To map LBS privacy to opacity problems in DES, one key element is to build a finite-state automaton model that is consistent with the knowledge of the attacker (i.e., the server). We discretize the physical map and capture users' mobility patterns in a finite-state automaton. Specifically, we show our modeling methodology using a set of walking paths on the Central Campus of the University of Michigan. The constructed automaton will be our running example that illustrates the use of opacity techniques for location privacy. Shown in Figure 4 is the map of the Central Campus. We select eight locations as states, marked in red in the figure. Mobility patterns, as shown in blue lines, define transitions between states. Regions A, B, C, D are cloaking regions precomputed by the anonymizer.[2] We omit the unobservable details and label transitions by the cloaking information received by the server. Specifically, transitions are labeled by the cloaking regions of their source nodes, meaning that the user makes queries when s/he is about to move to the next location. The users can start their walking paths from any location and thus $X_0 = X$. The constructed model is the nondeterministic finite-state automaton shown in Figure 5.

5. VERIFICATION OF LOCATION PRIVACY

To verify if the given moving patterns have location privacy, we need to know the server's location estimates and check if any estimate contains only one single location. Our methodology for constructing G from the moving patterns allows us to formulate location privacy as current-state anonymity. Location privacy holds if and only if the constructed G is current-state anonymous. As stated in Proposition 3, current-state anonymity can be verified using the current-state estimator $\mathcal{E}_G$. The moving patterns have location privacy if and only if no estimate state is a singleton.

Let us go back to the Central Campus example in Figure 4. We construct the current-state estimator $\mathcal{E}_G$ in Figure 6 to verify current-state anonymity. Before verifying, we first look at all estimate states in $\mathcal{E}_G$ that are reached by single events from the initial state of $\mathcal{E}_G$, such as state $\{4, 6, 7\}$ reached by event d. It can be found that no such state reveals the true location of the user. Because each cloaking region covers two distinct point locations and

[2] Shokri et al. (2010) has shown that computing cloaking regions in real-time does not improve privacy.

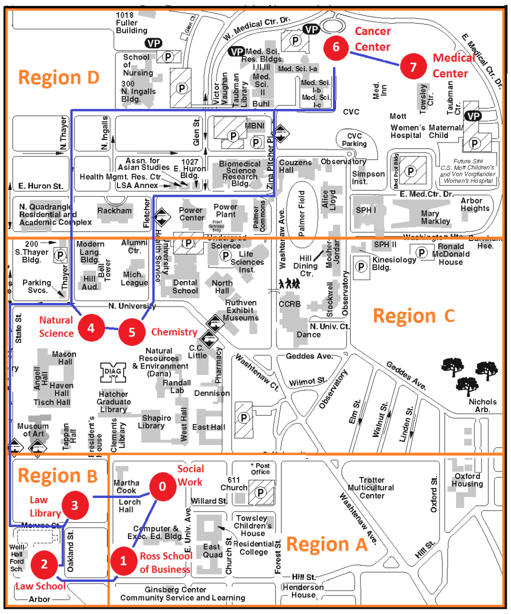

Fig. 4. The University of Michigan Central Campus map

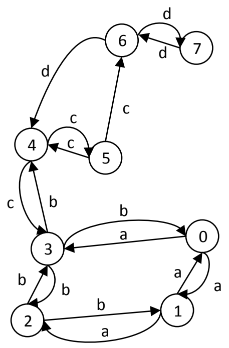

Fig. 5. The constructed automaton model G

no such two locations lead to only one single location, location privacy for one-time queries can be guaranteed. However, to perform full verification, we need to examine the complete structure of $\mathcal{E}_G$. This corresponds to the server's knowledge in a dynamic environment where users continuously make queries. The estimate state $\{6\}$ shows that the user will reveal his/her true location at state 6 (i.e., Cancer Center) after querying sequences such as *cdd*. Hence, current-sate anonymity does not hold. This revelation is because the server knows that consecutive queries d's can only be made between states 6 and 7 and that the user came from region C from the first received query. This also shows that the anonymizer's cloaking technique does not necessarily provide location privacy.

6. ENFORCEMENT OF LOCATION PRIVACY

6.1 I-Enforcing Insertion Functions

To resolve the location revelation identified in Section 5, we propose to add to the anonymizer an insertion function that inserts fake queries to the cloaking query sequence from the anonymizer. An insertion function, as introduced in (Wu and Lafortune, 2014) for opacity enforcement, is placed between the system and the outside attacker; it modifies the system behavior seen by the attacker. In the

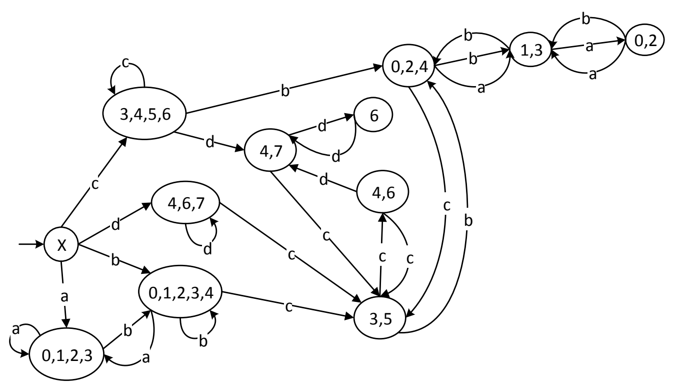

Fig. 6. The current-state estimator $\mathcal{E}_G$ of G

framework of LBS, we place the insertion function at the output of the anonymizer, as shown in Figure 2. It inserts fake queries and drops their replies without affecting the quality of the server's replies to real queries.

We have characterized in (Wu and Lafortune, 2014) a property called *i-enforceability* that an insertion function needs to satisfy in order to enforce opacity. We have also constructed the so-called *All Insertion Structure*, or AIS, that enumerates in a finite-state transition structure all i-enforcing insertion functions. This AIS can then be employed to synthesize one insertion function. Both the i-enforceability property and the construction of the AIS are based on the *safe language*, denoted by L_{safe}, which contains all observable strings that are "safe" for the system to output (i.e., that do not lead to a violation of opacity). Hence, by suitably defining L_{safe}, we can use the techniques from (Wu and Lafortune, 2014) to develop insertion functions for current-state anonymity, the property of interest in this paper.

The safe language in the LBS location privacy problem consists of all observable strings that do not reveal the current location of the user. That is, it contains all strings in $\mathcal{L}(\mathcal{E}_G)$ that do not visit singleton states (i.e., state $\{6\}$ for $\mathcal{E}_G$ in Figure 6). An insertion function f_I is i-enforcing if (1) every output behavior from f_I is in L_{safe}; and (2) f_I does not block or change any query from the anonymizer. We construct the AIS of G by following the procedure in (Wu and Lafortune, 2014), adapted for our L_{safe}. The AIS enumerates all i-enforcing insertion functions using a game structure that describes the interaction of the system (i.e., the user's continuous queries from the anonymizer) and the insertion function. The entire AIS for the Central Campus example has 84 states; part of its structure is shown in Figure 7. As can be seen in the figure, the AIS is a bipartite graph with "square" states and "round" states. The shapes of these states tell us whether the anonymizer or the insertion function is acting in the game. At square states, the anonymizer enumerates all possible user queries according to the moving patterns. For example, initially at state 0, the anonymizer can output queries a, b, c, d. At round states, the insertion function enumerates all i-enforcing insertion choices, determined from the construction procedure of the AIS, that respond to the queries from the anonymizer. For example, after the anonymizer outputs d, at state 4, the insertion function can insert $\varepsilon, b_i, c_i, b_i c_i c_i$. By listing all actions of the two players in this manner, the AIS enumerates all i-enforcing insertion functions. While the user continuously makes queries, the stages of this game are classified into a finite number of states in the AIS, thereby resulting in a finite structure. To synthesize one i-enforcing insertion function, one needs to select all edges from the square states (in order to consider all user's queries) and one insertion edge for each round state that is reached. In the next section, we will discuss how to select insertion edges and synthesize one *optimal* insertion function from the AIS.

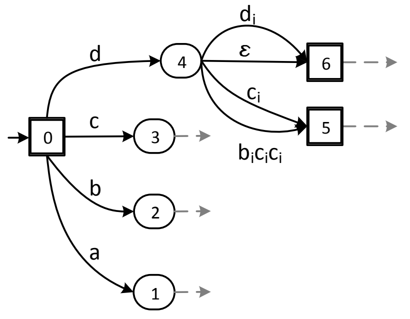

Fig. 7. Partial representation of the AIS (7 out of 84 states)

6.2 Optimal Insertion Functions

One can synthesize an i-enforcing insertion function by "randomly" selecting insertion edges at the round states in the AIS. But as inserting fictitious queries introduces extra delay and consumes energy, a more interesting problem is how to minimize the overhead cost introduced by insertion, i.e., how to synthesize an *optimal* insertion function.

In the LBS privacy problem, we assign the same unit cost to each inserted query, for the sake of simplicity. We then employ the optimization algorithms developed in (Wu and Lafortune, 2013b) to synthesize an optimal insertion function from the AIS. These algorithms are not reviewed here because of space constraints. We first observe that no i-enforcing insertion function has a *finite* worst-case total cost for the case of G of Figure 5. That is, to enforce location privacy in this example, an insertion function needs to continuously add fictitious queries as the user makes queries (note that there are many cyclic paths that return to singleton state $\{6\}$ in Figure 6.) Hence, we consider the quantitative objective for optimization purposes to be the long-run average insertion cost (i.e., per user query), in the worst case. (See (Wu and Lafortune, 2013b) for further technical details about this cost structure.) The minimum value that we find in our example is 2, meaning that at most two fake queries per user query need to be inserted. Using this value, we then synthesize from the AIS an insertion function f_I that achieves this optimal value. The result for our example is shown in Figure 8, where f_I is encoded as an I/O automaton where the input labels are queries from the anonymizer and the output labels are the modified queries with insertion. Events with subscript i denote fake queries from the insertion function.

Let us now look at query sequence cdd, which is found in Section 5 to reveal the true location of the user. We can see from Figure 8 that the optimal insertion function will modify cdd to $cdc_i c_i d$. The server does not distinguish fictitious and genuine queries; i.e., c and c_i are indistinguishable by the LBS. Thus, when the server receives cdc_ic_id, it interprets the sequence as $cdccd$ and infers that the user is at $\{4, 7\}$, while the true location is 6. Hence, location privacy is enforced.

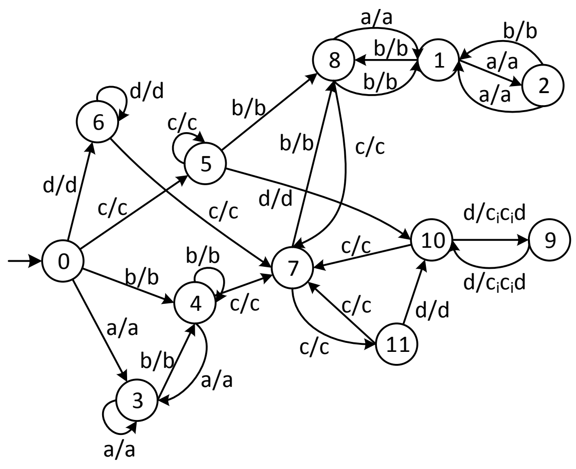

Fig. 8. The I/O automaton representation of the optimal insertion function

Remark 4. While we have synthesized an insertion function in the anonymizer framework, the insertion mechanism does not require an anonymizer. To develop the insertion mechanism without an anonymizer, one needs to modify the automaton model by labeling transitions using the original queries instead of the cloaking queries. An i-enforcing insertion function can then be constructed by following the same procedure. Note that each query is made when the user is about to move and is labeled by the region of the source location.

7. CONCLUSION

We have considered location privacy in Location-Based Services in the context of opacity problems in DES and presented a formal approach to solve location privacy problems. To characterize location privacy, we have adapted the notion of current-state opacity and defined *current-state anonymity.* We have developed a modeling methodology that capture users' mobility patterns using an automaton model. By using the current-state estimator to verify current-state anonymity, we have illustrated that the traditional anonymizer framework is in general insufficient to protect location privacy in a dynamic environment. To enforce location privacy, we have proposed to use the mechanism of *insertion functions* developed in our prior work, that inserts fictitious queries in the cloaking query sequences. Furthermore, we have shown how to synthesize an *optimal* insertion function to minimize the overhead costs caused by insertion. In the future, it would be interesting to look into other privacy concerns, such as query privacy.

REFERENCES

Beresford, A.R. and Stajano, F. (2003). Location privacy in pervasive computing. *Pervasive Computing*, 2(1), 46–55.

Bettini, C., Wang, X.S., and Jajodia, S. (2005). Protecting privacy against location-based personal identification. In *Secure Data Management*, 185–199. Springer.

Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems, 2nd Edition.* Springer.

Cassez, F., Dubreil, J., and Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 1–28.

Chow, C.Y. and Mokbel, M.F. (2007). Enabling private continuous queries for revealed user locations. In *Advances in Spatial and Temporal Databases*, 258–275. Springer.

Duckham, M. and Kulik, L. (2005). A formal model of obfuscation and negotiation for location privacy. In *Pervasive Computing*, 152–170. Springer.

Gruteser, M. and Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of the 1st International Conference on Mobile Systems, Applications and Services*, 31–42.

Kido, H., Yanagisawa, Y., and Satoh, T. (2005). An anonymous communication technique using dummies for location-based services. In *Proc. of International Conference on Pervasive Services*, 88–97.

Krumm, J. (2009). A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6), 391–399.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3), 496–503.

Mazaré, L. (2003). Using unification for opacity properties. *Proc. of the 4th IFIP WG1*, 7, 165–176.

Mokbel, M.F., Chow, C.Y., and Aref, W.G. (2006). The new Casper: Query processing for location services without compromising privacy. In *Proc. of the 32nd International Conference on Very Large Data Bases*, 763–774.

Pingley, A., Zhang, N., Fu, X., Choi, H.A., Subramaniam, S., and Zhao, W. (2011). Protection of query privacy for continuous location based services. In *Proc. of the IEEE INFOCOM*, 1710–1718.

Saboori, A. and Hadjicostis, C.N. (2008). Verification of initial-state opacity in security applications of DES. *Proc. of the 9th International Workshop on Discrete Event Systems*, 328–333.

Shin, K.G., Ju, X., Chen, Z., and Hu, X. (2012). Privacy protection for users of location-based services. *Wireless Communications, IEEE*, 19(1), 30–39.

Shokri, R., Troncoso, C., Diaz, C., Freudiger, J., and Hubaux, J.P. (2010). Unraveling an old cloak: k-anonymity for location privacy. In *Proc. of the 9th ACM Workshop on Privacy in the Electronic Society*, 115–118.

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

Wu, Y.C. and Lafortune, S. (2013b). Synthesis of optimal insertion functions for opacity enforcement. Technical report, University of Michigan.

Wu, Y.C. and Lafortune, S. (2014). Synthesis of insertion functions for enforcement of opacity security properties. *Automatica.*

Xu, T. and Cai, Y. (2008). Exploring historical location data for anonymity preservation in location-based services. In *Proc. of the IEEE INFOCOM*, 547–555.