OPAQUE: Protecting Path Privacy in Directions Search

Ken C. K. Lee[†‡] Wang-Chien Lee[†] Hong Va Leong[‡] Baihua Zheng[§]
cklee@cse.psu.edu wlee@cse.psu.edu cshleong@comp.polyu.edu bhzheng@smu.edu.sg
[†]Department of Computer Science and Engineering, Pennsylvania State University, USA.
[‡]Department of Computing, The Hong Kong Polytechnic University, Hong Kong.
[§]School of Information Systems, Singapore Management University, Singapore.

***Abstract*— Directions search returns the shortest path from a source to a destination on a road network. However, the search interests of users may be exposed to the service providers, thus raising privacy concerns. For instance, a path query that finds a path from a *resident address* to a *clinic* may lead to a deduction about "who is related to what disease". To protect user privacy from accessing directions search services, we introduce the *OPAQUE* system, which consists of two major components: (1) an *obfuscator* that formulates obfuscated path queries by mixing true and fake sources/destinations; and (2) an *obfuscated path query processor* installed in the server for obfuscated path query processing. OPAQUE reduces the likelihood of path queries being revealed and allows retrieval of requested paths. We propose two types of obfuscated path queries, namely, *independently obfuscated path query* and *shared obfuscated path query* to strike a balance between privacy protection strength and query processing overhead, and to enhance privacy protection against collusion attacks.**

I. Introduction

With proliferation of digital maps, positioning technology, and the Internet, we have witnessed a rapid growth of *location-based services (LBS)*. The analysis from Frost & Sullivan reveals that the market, covering 13 major Asia-Pacific economies, was worth $291.7 million in 2006 and is expected to grow 15.3 percent annually to reach an estimated $447 million by the end of 2009. Among many LBSs such as E-911, NextBus (http://www.nextbus.com), NearestToilet (http://www.mizpee.com) etc, directions search is one of the most popular LBSs. Well-known map service providers, such as GoogleMap, MapQuest, MS Virtual Earth, and Yahoo! Map, have already offered convenient directions search service.

With a source address s and a destination address t provided by a user, directions search service determines a path routed from s to t that satisfies some additional specified conditions (e.g., avoid highways). In this paper, we assume the server providing directions search services adopts a simple client/server architecture, as shown in Figure 1, and formulate a request for a path between a source, s, and a destination, t as a path query, denoted by $Q(s, t)$. The server maintains the map data and is responsible for path query processing. Whenever it receives a path query from a user (or client, hereafter), it invokes some well-known shortest path algorithms (e.g., Dijkstra's shortest path algorithm [1] and A* algorithm[2]) to compute the result path.

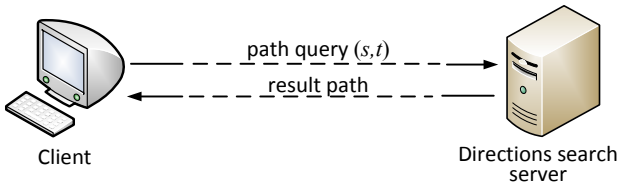


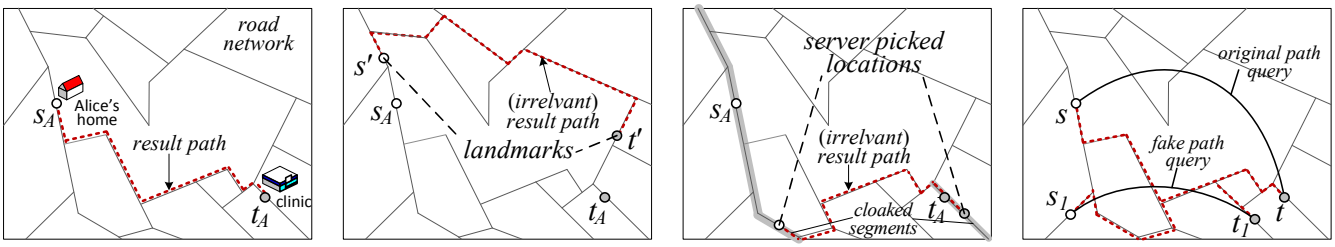
Fig. 1. Directions search based on simple client/server model

II. Motivation

Although the directions search service gains popularity among users, it raises a privacy concern due to the exposure of specified sources and destinations in path queries, especially if directions search services are run by semi-trusted parties. Consider an example, Alice wants to visit an infertility specialist located at "56, Clinton Road, Harrisburg, PA" but she does not know the way there. Hence, she requests driving directions from her home to that clinic by submitting a path query $Q(s_A, t_A)$ to a directions search server, with the source s_A set to her home address "123, Fourth Street, State College, PA" and the destination t_A set to the clinic address "56, Clinton Road, Harrisburg, PA". The server, upon receiving the query from Alice that contains accurate source and destination addresses, can return an exact path, as depicted in Figure 2(a).

Meanwhile, the server, with the help of some public information such as voter registration list and yellow pages, can determine that: i) the user who submits the query is likely to be Alice (or her family members); and 2) the destination is a fertility clinic. Then, the server may deduce that Alice is going to visit an infertility specialist. One may argue that the leakage of this information, as a random case, is not an issue. However, the server can accumulate all the path queries received to learn where individuals travel, what routes they take, the travel duration and so on. Moreover, the user health status, activities, political preferences, religion or even relationships with others can be inferred. As this privacy concern arises in path searching through semi-trusted directions search servers, we refer this to as "path privacy".

It is natural to link path privacy with location privacy, as they two appear similar in the context of LBSs. However, existing solutions for protecting location privacy, including *landmark*, *cloaking* and *obfuscation* cannot be effectively applied to path privacy. The landmark approach uses other locations instead of original (true) user locations and hence it replaces a path query with another path query that has totally different source and destination [3], [4]. Although it can protect the path privacy, the cost is that the retrieved result path is completely irrelevant to the original path query. An example



(a) Path between s and t (b) Landmarks (c) Cloaking (d) Obfuscation

Fig. 2. Path search between s_A and t_A, and some existing location privacy protection techniques

is shown in Figure 2(b) where the source and destination are changed to two landmarks s' and t', respectively, and the retrieved result path cannot connect s_A to t_A.

Cloaking tries to suppress the address detail, e.g., discarding street number and street name from an exact address [5], [6], [7]. As shown in Figure 2(c), both source and destination are cloaked into locations of street level. However, existing directions search services may arbitrarily pick a point for an imprecise address to perform the path search. Consequently, it is very likely that an irrelevant path will be returned. By obfuscation, a path query is mixed with a number of fake path queries [8]. As exemplified in Figure 2(d), a path query set $\{Q(s_1,t_1), Q(s_A,t_A)\}$ that mixes a fake query $Q(s_1,t_1)$ with the real one $Q(s_A,t_A)$ is submitted to the server. The server returns the results for each path query involved into the query set and hence the result to the true path query $Q(s_A,t_A)$ is for sure to be retrieved. In addition, the privacy concern is addressed as the server cannot determine what path queries it receives are true or fake. Nevertheless, clients retrieve additional paths for the fake queries, which are redundant, resulting in overconsumption of server and network resources.

In summary, all these existing location privacy protection techniques can protect path privacy to a certain extent but they either cannot provide requested results or incur high processing overhead. In this paper, we propose the concept of *obfuscated path query* and develop a path privacy protection system called *OPAQUE*, which is named after Obfuscated PAth QUEry, to address both the privacy and the efficiency of path queries. The research objective is to protect user path privacy without sacrificing the service quality of directions searches.

III. Obfuscated Path Query

In this section, we discuss path query in the road network and then detail the concept of obfuscated path query, the theoretical foundations of the OPAQUE system.

A. Path Query

A road network is typically modeled as a weighted graph $G(N,E)$ where road segments are represented by a set of edges, E, and the endpoints of edges are represented by a set of nodes, N. Each edge $(n_i,n_j) \in E$ linking a node n_i to a node n_j (where $n_i,n_j \in N$) is associated with a non-negative distance, $d(n_i,n_j)$, that represents the traveling distance, time or toll of a corresponding road segment. A path from a *source* node s to a *destination* node t is represented by a sequence of edges, i.e., $\langle (s,n_0),(n_0,n_1),\cdots(n_y,t)\rangle$, and its *path distance* is the sum of the distances of involved edges, i.e., $d(s,n_0) + \sum_{x=0}^{y-1} d(n_x,n_{x+1}) + d(n_y,t)$. The shortest path from s to t, denoted by $P(s,t)$, is a path with its distance, represented by $||s,t||$, being the smallest among all possible paths from s to t. Given a road network $G(N,E)$, a source s and a destination t, a path query $Q(s,t)$ evaluated on G returns $P(s,t)$.

B. Basic Idea of Obfuscated Path Query

Usually, each node in a network represents one location in the geographical space. People and/or business associated with nodes can be found via public information (e.g., voter registration list, yellow pages etc). Issuing $Q(s,t)$ indicates an intent of a user to travel from a node s to another node t, implying certain relationships between people/business associated with s and t. Further, a user is very likely to take the returned path $P(s,t)$, and the information leakage may threaten personal safety. To address this privacy concern, we introduce the concept of *obfuscated path query*, as a key technique for path privacy protection. It is formally defined in Definition 1. Each obfuscated path query $\mathcal{Q}(S,T)$ mixes s and t of a path query $Q(s,t)$ with some fake locations. In other words, an obfuscated path query represents a set of path queries with their sources and destinations included in S and T, respectively, i.e., $\mathcal{Q}(S,T) = \bigcup_{s\in S \wedge t\in T}\{Q(s,t)\}$. Take Alice's $Q(s_A,t_A)$ as an example. Obfuscated path query $\mathcal{Q}(S_A,T_A)$ can be formulated with $S_A = \{s_A,s_1\}$ and $T_A = \{t_A,t_1,t_2\}$ as depicted in Figure 3(a).

***Definition** 1:* **Obfuscated Path Query**. Given a path query $Q(s,t)$, an obfuscated path query $\mathcal{Q}(S,T)$ is formulated such that $s \in S$ and $t \in T$. □

As the obfuscated path query is proposed to protect path privacy, we introduce a new metric, called *breach probability*, to quantify the power of the obfuscation in terms of protecting path privacy, as defined in Definition 2. For the case of $\mathcal{Q}(S_A,T_A)$, the probability that $Q(s_A,t_A)$ can be revealed is $\frac{1}{2\cdot 3} = \frac{1}{6}$. Intuitively, as the sizes of S and/or T (i.e., $|S|$ and $|T|$, respectively) increase, the breach probability that $Q(s,t)$ can be revealed is decreased. Then, one naive strategy to increase the privacy protection is by including more locations in S and/or T.

***Definition** 2:* **Breach Probability**. Assume that a path query $Q(s,t)$ is obfuscated into an obfuscated path query $\mathcal{Q}(S,T)$ such that $s \in S$ and $t \in T$. The breach probability is the probability that $Q(s,t)$ can be revealed from $\mathcal{Q}(S,T)$, i.e., $\frac{1}{|S|\times|T|}$. □

However, an obfuscated path query $\mathcal{Q}(S,T)$ is a combination of $|S| \times |T|$ path queries. From the query processing point of view, we should reduce the sizes of S and/or T in order to reduce server workload and improve the search performance. Thus, how to strike a balance between the privacy and performance is an important issue. In order to tackle this issue, we analyze the nature of path query processing in road network and propose several query processing optimization techniques that can effectively lower the processing overhead while retaining large $|S|$ and $|T|$.

Conventionally, processing a path query $Q(s,t)$ on a road network G is based on some well-known search algorithms, for example, Dijkstra's algorithm. The basic idea of Dijkstra's algorithm is to form a spanning tree rooted at s and gradually expand the search space until t is reached. The computational cost and I/O cost, two of the major search processing costs, are bounded by the size/area of a subgraph covered by the spanning tree, assuming that nodes and their edges are clustered and stored on disk [9]. With s as the center and the distance from s to t as the radius of a search area, the cost is estimated as $O(||s,t||^2)$.

Further, Dijkstra's algorithm is extensible to search paths from a single source to multiple destinations by forming a spanning tree until all the destinations are reached. Suppose that T is a set of destinations, the search processing cost will be $O(\max_{t\in T}||s,t||^2)$. In other words, if the difference between $||s,t||$ and $\max_{t'\in T}||s,t'||$ is not significant, searching paths from a single source to multiple destinations incurs a search cost similar to or slightly more expensive than that of searching a path from one source to one destination. Then, the overall processing cost of an obfuscated path query is $O(\sum_{s\in S}\max_{t\in T}||s,t||^2)$, as derived in Lemma 1. Motivated by this observation, we balance the power of path privacy protection and the processing cost by setting appropriate $|S|$ and $|T|$.

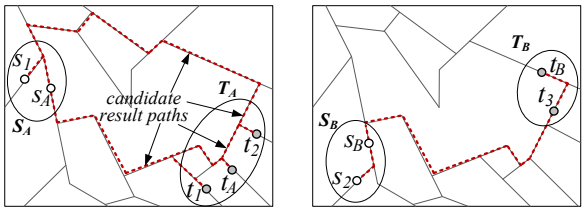
Lemma 1: The processing cost of an obfuscated path query $\mathcal{Q}(S,T)$ is $O(\sum_{s\in S}\max_{t\in T}||s,t||^2)$. □

Proof. The total processing cost is obtained by summing up all costs of shortest path searches initiated at $s \in S$ if there is no sharing of partial spanning time. As the cost of individual search started from s to all destinations in T, i.e., $O(\max_{t\in T}||s,t||^2)$, the total cost of processing an obfuscated path query is $O(\sum_{s\in S}\max_{t\in T}||s,t||^2)$. ■

C. Variants of Obfuscated Path Query

As privacy protection is subject to personal needs, in OPAQUE, each user can specify her preferred obfuscation power by informing the obfuscator her desired sizes of S and T of the obfuscated path query, denoted by f_S and f_T, respectively. The larger f_S and f_T are set, the larger S and T will be formed and the stronger a protection will be resulted. In this paper, we propose two variants of obfuscated path queries, namely, *independent obfuscated path query* and *shared obfuscated path query*.

Given a set of path queries, $Q(s_1,t_1), \cdots Q(s_k,t_k)$ along with their protection settings, $(f_{S1},f_{T1}), \cdots (f_{Sk},f_{Sk})$, we can obfuscate each path query into an *independent obfuscated path query* $\mathcal{Q}(S_i,T_i)$ with $s_i \in S_i$, $t_i \in T_i$, $|S_i| = f_{Si}$ and $|T_i| = f_{Ti}$. Suppose there are two path queries, i.e., $Q(s_A,t_A)$ submitted by Alice and $Q(s_B,t_B)$ issued by Bob. We formulate two independent obfuscated path queries, namely $\mathcal{Q}(S_A,T_A)$ with $S_A=\{s_A,s_1\}$ and $T_A=\{t_A,t_1,t_2\}$ and $\mathcal{Q}(S_B,T_B)$ with $S_B=\{s_B,s_2\}$ and $T_B=\{t_B,t_3\}$, shown in Figure 3 and Figure 3(b), respectively.



(a) Independent obf. path query, $\mathcal{Q}(\{s_A,s_1\},\{t_A,t_1,t_2\})$ (b) Independent obf. path query, $\mathcal{Q}(\{s_B,s_2\},\{t_B,t_3\})$

Fig. 3. Independent obfuscated path query

Alternatively, we can obfuscate all k path queries into a shared obfuscated path query $\mathcal{Q}(S,T)$ such that $\{s_1,\cdots s_k\} \subseteq S$ and $\{t_1,\cdots t_k\} \subseteq T$ where $|S| \geq \max_{1\leq i\leq k} f_{Si}$ and $|T| \geq \max_{1\leq i\leq k} f_{Ti}$. Reconsider Alice's and Bob's path queries. Rather than generating two independent obfuscated path queries, Alice's $Q(s_A,t_A)$ and Bob's $Q(s_B,t_B)$ can be obfuscated into a shared obfuscated path query $\mathcal{Q}(S,T)$ where $S=\{s_A,s_B,s_1\}$ and $T=\{t_A,t_B,t_1,t_2\}$, as shown in Figure 4.

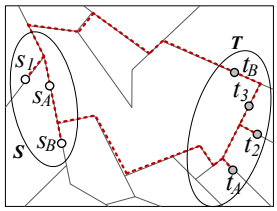


Fig. 4. Shared obfuscated path query

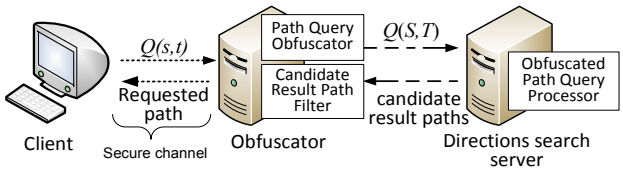


Fig. 5. OPAQUE system model

IV. Proposed OPAQUE System

OPAQUE architecture is based on the client-obfuscator-server model, as depicted in Figure 5, which offers several advantages. First, clients only need to trust a single trusted entity, i.e., the obfuscator. Although the obfuscator would become the attack point of the system or performance bottleneck, a trusted third party is very often assumed in the field of security [7], [10]. Second, centralized obfuscation based on the obfuscator is expected to provide a high obfuscation efficiency. Third, complicated obfuscation logic, such as determining fake

sources and destinations that needs knowledge of underlying networks, is hidden from the clients. Last but not the least, it offers a stronger privacy protection as clients do not directly communicate with the server and their path queries are obfuscated and anonymized to the server.

In detail, the system contains three main components, namely, *path query obfuscator*, *obfuscated path query processor* and *candidate result path filter*. Their interactions are illustrated in Figure 6. Path query obfuscator and candidate result path filter are installed in the obfuscator to obfuscate path queries and to perform candidate result path filtering. Obfuscated path query processor is equipped in directions search server for obfuscated path query evaluation.

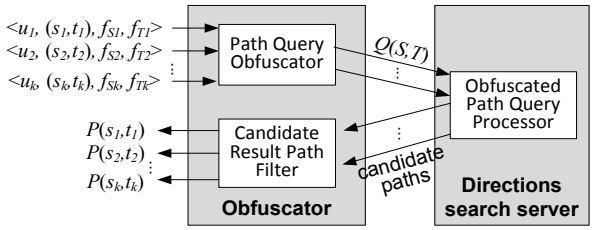


Fig. 6. OPAQUE components

All clients, u_i, send their query $Q(s_i, t_i)$ along with protection settings f_{Si} and f_{Ti} as a request $\langle u_i, (s_i, t_i), (f_{Si}, f_{Ti})\rangle$ to the obfuscator. The communication channel between each client and the obfuscator is secured so that every request is not exposed to parties other than the obfuscator and the corresponding client. In the obfuscator, the received queries are obfuscated by the path query obfuscator into obfuscated path queries. The detailed query obfuscation involves two steps, i.e., *path query clustering* and *query obfuscating*. The former step partitions the received queries into disjoint query sets, while the latter obfuscates queries in each cluster into obfuscated path queries, which could be (i) independent obfuscated path query, or (ii) shared obfuscated path query, as defined earlier. OPAQUE has designed and implemented efficient path query obfuscation algorithm to minimize the processing overhead and meanwhile maintain the privacy protection.

Thereafter, the obfuscated path queries are sent to the directions search server for processing. In the mean time, the requests are kept for later result path filtering. Since finding fake sources and destinations for path query obfuscation requires the knowledge of the underlying road network, we assume a simple road map (e.g., obtained from Tiger/Line [11]) maintained with the path query obfuscator. Different from sophisticated one maintained in the directions search server, this road map does not have real-time traffic information, and driving directions needed for path search.

Upon receiving obfuscated path queries, the directions search server evaluates the queries. In order to efficiently evaluate obfuscated path queries, each of which represents a set of path queries, the obfuscated path query processor is devised. A set of efficient multiple source multiple destination path search algorithms have been designed and implemented by OPAQUE. After evaluation, result paths are returned to the obfuscator. At last, those paths are screened by the candidate result path filter and the requested paths are returned to the corresponding clients. Thereafter, the satisfied requests are immediately discarded in the obfuscator, for sake of security.

V. Conclusion

Directions search, one of the most popular location-based services, supports search for a path from a source to a destination in a road network. Users specify their source s and destination t, and the server will return the shortest path from s and t. As users express their search interests in form of a path query to a semi-trusted server, a privacy concern is raised. To address this problem, we propose the concept of obfuscated path queries that mixes some extra fake sources and destinations to path queries and develop OPAQUE system hires a trusted obfuscator to sit between the directions search server and the users. By doing the query obfuscation at the obfuscator, the original true path queries are hidden from the server and hence the privacy issue is addressed. On the other hand, with the help of efficient obfuscated path query formation algorithms and multiple source multiple destination path search algorithms, the efficiency issue is also addressed.

Acknowledgment

In this research, Wang-Chien Lee and Ken C. K. Lee are supported in part by the National Science Foundation under Grant no. IIS- 0328881 and IIS-0534343. Besides, Wang-Chien Lee is supported in part by the National Science Foundation under Grant no. CNS- 0626709, and Ken C. K. Lee and Hong Va Leong are supported in part by the Research Grant Council, Hong Kong SAR under grant HKBU 1/05C.

References

[1] E. W. Dijkstra, "A Note on two Problems in Connexion with Graphs," in *Numerische Mathematik*, 1959, pp. 269–271.

[2] R. Dechter and J. Pearl, "Generalized Best-First Search Strategies and the Optimality of A*," *Journal of ACM*, vol. 32, no. 3, pp. 505–536, 1985.

[3] J. I. Hong and J. A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, 2002, pp. 177–189.

[4] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, Cancun, Mexico, Apr 7-12*, 2008, pp. 366–375.

[5] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003), San Francisco, CA, USA*, 2003.

[6] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 12, pp. 1719–1733, 2007.

[7] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, Sep 12-15*, 2006, pp. 763–774.

[8] M. Duckham and L. Kulik, "A Formal Model of Obfuscation and Negotiation for Location Privacy," in *Proceedings of Third International Conference on Pervasive Computing (PERVASIVE), Munich, Germany, May 8-13*, 2005, pp. 152–170.

[9] S. Shekhar and D.-R. Liu, "CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 1, pp. 102–119, 1997.

[10] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2008.

[11] U.S. Census Bureau, "Tiger/Line."