This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

# Privacy Preserving Group Nearest Neighbour Queries in Location-based Services using Cryptographic Techniques

Yan Huang, Roopa Vishwanathan
Department of Computer Science and Engineering
University of North Texas, Denton, TX
{huangyan, rv0029}@unt.edu

***Abstract*—Location-based services (LBS) are available on a variety of mobile platforms like cellphones, PDA's, etc. and an increasing number of users subscribe to and use these services. One of the basic privacy issues with LBS is that a user may not necessarily want to disclose their own location whenever they inquire about the location of places of interest to them e.g., nearest gas station, restaurant etc. The privacy aspect of LBS has received attention recently with a number of privacy-preserving methodologies being proposed for the client-server model where a querying client requests a location-based server to return some location that is of interest to it without revealing its own location to the server. In this paper, we consider privacy issues in the peer-to-peer model of LBS, where a group of users jointly compute a common location of interest to them such as a restaurant where they could all meet. In such scenarios, all peers in the group would like to jointly find a common location but might not want to reveal their individual locations to each other due to trust issues. We model this problem in the secure multi-party computation framework of cryptography and present a solution where all the peers can jointly compute a common location without the need for any user to reveal its individual location to anyone else. To this end, we present two privacy-preserving models and experimentally evaluate the performance of each of them.**

## I. Introduction

Some of the common forms of queries in LBS are users querying a location-based server about nearby places of interest such as restaurants, tourist attractions, stores, etc. or spatial queries about a user's nearest neighbour or $k$-nearest neighbours within a certain distance. In order to process location-related queries, a location-based server would need information about the location of the querying user, but providing information about their current locations to a potentially untrusted server might be a matter of concern for users. A user might want to avail of these services while keeping their own locations private. Generally, privacy in LBS in the client-server model has been achieved through application of well-known anonymity measures such as $k$-anonymity [1], [2], or using a centralized trusted third party anonymizer [3], [4], or by using cryptographic protocols [5].

An alternative communication model is the peer-to-peer model where a group of peers would like to co-operatively compute some location without seeking the help of a centralized location-based server. Apart from the obvious shortcomings of the centralized approach such as the server being untrusted, or the server being a communication bottleneck and a single point of failure, the power of user hand-held devices has been increasing with technologies like Bluetooth or IEEE 802.11 being deployed on a range of user hand-held devices. With the computational power of user devices increasing, users may not even need a location-based server or a base station to answer queries and could process queries among themselves. Although there can be many types of spatial queries that a group of peers can compute, in this paper we focus on *group nearest neighbour* queries in which a group of parties, $P_1, \cdots, P_n$ would want to compute their joint group nearest neighbour. In a real-world situation, this could be just a group of users wanting to compute a location that is closest to all of them, e.g., "Which is the restaurant that is closest to all of us where we can meet?". The problem we are focusing on is the situation when the parties want to jointly compute this function but do not want to reveal their individual locations to each other, and work in the absence of a trusted third party. Geometrically, the group nearest neighbour of the group of parties is the minimum of the sum of the distances of each party from each point in a given set of points in the problem space. If there are $P_1, P_2, \cdots, P_n$ parties and $L_1, L_2, \cdots, L_l$ locations, for computing the group nearest neighbour, we firstly compute the sum of the distances of each party from each location: $s_1 = \sum_{i=1}^{n} d(P_i, L_1), s_2 = \sum_{i=1}^{n} d(P_i, L_2), \cdots, s_l = \sum_{i=1}^{n} d(P_i, L_l)$. We then obtain the minimum of all these values: $min(s_1, s_2, \cdots, s_l)$. This value represents the location that is the "group nearest neighbour" of the group of users. In this paper, we propose two models for preserving privacy of all peers and experimentally evaluate the performance of them.

## II. Related Work

In our methodology, we draw upon work done in the areas of both cryptography and location-based services. In this section, we briefly go over related work in these areas.

### A. Location-based services

In the peer-to-peer model of LBS, there would be a group of clients or peers who want to mutually compute some location-related function in the absence of a centralized trusted third party server. One of the common queries in this model

978-1-4244-5638-3/10/$26.00 ©2010 IEEE

are aggregate nearest neighbour or group nearest neighbour queries where the "nearest neighbours" are points of interest in the vicinity such as restaurants, hospitals, gas stations, etc. So a typical group nearest neighbour query for a group of peers would be "Which is the restaurant that is closest to all of us" or "Find a meeting spot that is within 2 miles of all of us?"

Some of the general work in this direction (non-privacy-preserving) includes [6], [7] among others. One of the widely referenced works in this area is SpaceTwist [6] which is a protocol where a querying client asks the LBS server to return a set of $k$ points closest to its own location or $k$-nearest neighbours. The authors do not apply their protocol to the group nearest neighbour problem which is quite different from $k$-nearest neighbour and has a different set of privacy requirements. There has been quite some work in the area of aggregate nearest neighbour queries [7] which show how to do query processing of aggregate nearest neighbour queries in road networks. Papadias et al. [8] propose a suite of algorithms for nearest neighbour queries, range queries, distance joins and storage scheme for objects situated on a static network.

Previous solutions for the group nearest neighbour query computation assume that each party would be willing to share its location with all other parties to compute the group nearest neighbour, or there would be a trusted third party who would be willing to compute the group nearest neighbour, to whom all the parties would reveal their individual locations. Although there has been prior work in the area of group nearest neighbour computation, none of the previous papers have suggested how to provide user privacy if the network is modeled as a peer-to-peer network, in the absence of a trusted third party, and if the peers are untrusted. We note that it is not possible to trivially extend or modify other location-based peer-to-peer protocols for the group nearest neighbour query, such as the ones that use $k$-anonymity [1], [2], since those protocols assume that peers would be willing to share their locations with each other. It can be seen that this problem fits in the secure multi-party function evaluation framework of cryptography discussed below.

### B. Cryptographic Protocols

Secure Function Evaluation (SFE) among two or more parties is a well-studied problem in cryptography in which a group of parties want to jointly compute a function $f(x_1, x_2, \cdots, x_n)$ where $x_1, x_2, \cdots, x_n$ represent their individual input bits. The goal for each party is to correctly compute the value of $f(\cdot)$ without divulging its own input bit to any other party and in the absence of a trusted third party. Pioneering work in this area was done by Yao [9], Goldreich, Micali and Wigderson (GMW) [10] and Beaver et al. [11]. There are two variants of the SFE problem: two-party/multi-party computation in the presence of semi-honest adversaries (the "semi-honest" model), and two-party/multi-party computation in the presence of malicious adversaries (the "dishonest" model). In the semi-honest model, it is assumed that all the parties involved in the computation will follow the protocol exactly, but will analyze the transcripts of their interactions with other parties to gain extra, un-intended information or in other words, will function as *passive* adversaries. In the dishonest model, it is assumed that a fraction of the parties that are "dishonest" will arbitrarily deviate from the protocol and will act as *active* adversaries. For the most part, in the semi-honest model, the various solutions/protocols developed are based on either Yao's protocol or the GMW paradigm. This paper is based on Yao's protocol in the semi-honest model.

### C. Our Contributions

Our contributions in this paper can be summarized as follows:

1) We propose a framework for preserving user privacy in the computation of group nearest neighbour queries in LBS, which is completely peer-to-peer based, does not require the use of a trusted third party and works in the presence of mutually untrusting peers.
2) We propose two models for the group nearest neighbour query computation: the centralized model and distributed model
3) We experimentally evaluate the performance in both these models and show that the costs are reasonable and applicable in practice. From our experiments, we find that the distributed model has lower computation costs for processing queries (0-2.1 sec) as compared to the centralized model (0-12 sec), while both models incur the same communication costs.

## III. Our Methodology

Our methodology is based on the secure function evaluation (SFE) problem and "garbled" circuits are central to solutions proposed for the SFE problem. Below we define some notations based on [12], [11] regarding garbled circuits.

**Garbled Circuit**: Let there be $n$ parties each having an $m-$bit input. We assume that the function which is represented by the boolean circuit outputs a single $m-$bit value. Let the function be represented by a garbled boolean circuit, $C$ consisting of $\Gamma$ 2-input gates. Let $W$ be the total number of wires where wires $0, \cdots, nm-1$ represent the inputs and $W-m, \cdots, W-1$ represent the outputs; in this description, we use Greek letters to refer to individual wires. Let the plaintext input bit of wire $\alpha$ be denoted by $b_\alpha \in \{0,1\}$. Each wire has a pair of *signals* associated with it: signals are random strings of the size of standard symmetric encryption keys (80, 128 or 256 bits in length) which are numbered in the same order as the wires. So, if $\alpha$ is a wire, $s_{2\alpha}$ and $s_{2\alpha+1}$ are the two signals associated with it. Each signal has a *semantics* variable associated with it which corresponds to the signal's plaintext value; the semantics variable is randomly chosen and is kept secret. For a wire $\alpha$, a semantics variable $\lambda_\alpha \in \{0,1\}$ is randomly chosen and indicates that $s_{2\alpha}$ has semantics $\lambda_\alpha$ and $s_{2\alpha+1}$ has semantics $\bar{\lambda_\alpha}$.

The garbled inputs are signals of the input wires that correspond to the actual input bits. The garbled inputs for a wire $\alpha$ would be of the form $\sigma_\alpha = s_{2\alpha+(b_\alpha \oplus \lambda_\alpha)}$ and

$\sigma'_\alpha = s_{2\alpha+1+(b_\alpha \oplus \bar{\lambda_\alpha})}$. A set of four *gate labels* are computed by the creator of the circuit. Loosely speaking, the four gate labels for each gate are generated by feeding the signals associated with each wire of the gate to a pseudorandom generator and computing the XOR of the strings generated. The gate labels can be thought of as a truth table (with four values) for gates of 2-input wires. Let $\mathcal{G}$ be a pseudorandom generator that takes as input a $k$-bit seed and outputs a $(k + 2nk)$-bit string. We define $F$, $G$, and $H$ to be the first $k$, next $nk$ and last $nk$ bits of $\mathcal{G}$'s output respectively. Let $f_j = F(s_j)$, $g_j = G(s_j)$, $h_j = H(s_j)$, for $0 \leq j \leq W - 1$ (i.e., $j$ ranges over the wire indices). Let $\oplus$ represent the logic function XOR, and let $\odot$ represent the logic function that the gate computes (AND, OR, etc.). If the left and right incoming wires and outgoing wire of a gate $i$ are $\alpha$, $\beta$, and $\gamma$ respectively, then writing $a = 2\alpha$, $b = 2\beta$, and $c = 2\gamma$, we can compute the gate labels as follows:

$$A_i = g_a \oplus g_b \oplus \begin{cases} s_c & \text{if} \lambda_\alpha \odot \lambda_\beta = \lambda_\gamma \\ s_{c+1} & \text{otherwise} \end{cases}$$

$$B_i = h_a \oplus g_{b+1} \oplus \begin{cases} s_c & \text{if} \lambda_\alpha \odot \bar{\lambda_\beta} = \lambda_\gamma \\ s_{c+1} & \text{otherwise} \end{cases}$$

$$C_i = g_{a+1} \oplus h_b \oplus \begin{cases} s_c & \text{if} \bar{\lambda_\alpha} \odot \lambda_\beta = \lambda_\gamma \\ s_{c+1} & \text{otherwise} \end{cases}$$

$$D_i = h_{a+1} \oplus h_{b+1} \oplus \begin{cases} s_c & \text{if} \bar{\lambda_\alpha} \odot \bar{\lambda_\beta} = \lambda_\gamma \\ s_{c+1} & \text{otherwise} \end{cases}$$

The evaluator of the circuit learns all the above gate labels and the garbled inputs from all players. If the evaluator holds two signal inputs: $s_{a+p}$ for the left incoming wire and $s_{b+q}$ for the right wire, where $p, q \in \{0, 1\}$, the evaluator can easily compute:

$$g_{a+p} \oplus g_{b+q} \oplus A_i \quad \text{if } p = 0 \text{ and } q = 0$$

$$h_{a+p} \oplus g_{b+q} \oplus B_i \quad \text{if } p = 0 \text{ and } q = 1$$

$$g_{a+p} \oplus h_{b+q} \oplus C_i \quad \text{if } p = 1 \text{ and } q = 0$$

$$h_{a+p} \oplus h_{b+q} \oplus D_i \quad \text{if } p = 1 \text{ and } q = 1$$

Our goal is to study the application of garbled circuits for answering group nearest neighbour queries in LBS in the semi-honest user model in two settings: centralized and distributed. Let us consider a group of parties $P_1, \cdots, P_n$ who want to jointly compute their group nearest neighbour; we describe two ways to realize this:

1) Centralized setting: In the centralized setting, a single user, Alice creates the boolean garbled circuit that represents the group nearest neighbour function. For evaluating the circuit we need another user who can be called an evaluator, Bob. All the other users then just get their encrypted input bits from Alice via Oblivious Transfer (OT) and send their encrypted input bits to the circuit evaluator Bob, who after getting all the parties' encrypted input bits (including Alice's) evaluates the circuit by himself. At the end of the circuit evaluation process, Bob will make known publicly the output of the last gate of the circuit, which would be the group nearest neighbour of the group of parties. Note that since all the users get their input bits from Alice via OT and then send their encrypted input bits to Bob, neither Alice nor Bob would get to know any user's input bits other than their own. Figure 1 represents the centralized model with $n$ users.
2) Distributed setting: In the distributed setting, the entire region is divided into groups of users and locations, with a pair of users being responsible for creating and evaluating a boolean circuit for a group of locations. If there are $n$ users and $l$ locations, the users can be divided into groups of pairs and we can have upto $n/2$ such groups. The locations can be divided into groups and we can have upto $n/2$ such location groups. In a fully distributed scenario, where all the users actively participate in circuit creation and evaluation, we would have $n/2$ groups of users and $n/2$ groups of locations with $2l/n$ locations in each group. The actual number of groups would be determined by the number of users and the number of locations in each individual case. If the number of locations is greater than or equal to the number of users, we would have $n/2$ location groups and user groups with each user actively participating in circuit creation and evaluation.
On the other hand, if the number of locations is lesser than the number of users, then there would be a small subset of users who would do the circuit creation and evaluation and most of the other users would just interact with the subset members to get their own input bits. In both cases each group of locations would be managed by 2 users - Alice who acts as a creator and Bob who acts as an evaluator. Alice creates an encrypted circuit for computing the group nearest neighbour of all users among the locations in her group and Bob acts as the evaluator of the circuit. All users have to perform OT with Alice to get their encrypted inputs bits and then transmit their encrypted input bits to Bob. In addition Alice herself transmits her encrypted input bits to Bob. This procedure has to be repeated for all groups of locations which can be done in parallel. By doing this, we divide the time taken for creating and evaluating the circuit among the users besides introducing more parallelism than what was originally there in the protocol. Figure 2 represents the distributed model with $n/2$ groups of users.

Below we give a high-level overview of an algorithm for our secure multi-party group nearest neighbour function evaluation protocol (GNN) in a fully distributed setting. The centralized version is a generalized, simpler version of this and isn't given here. The notation used is the same as given in the paragraph titled "Garbled circuit" above.

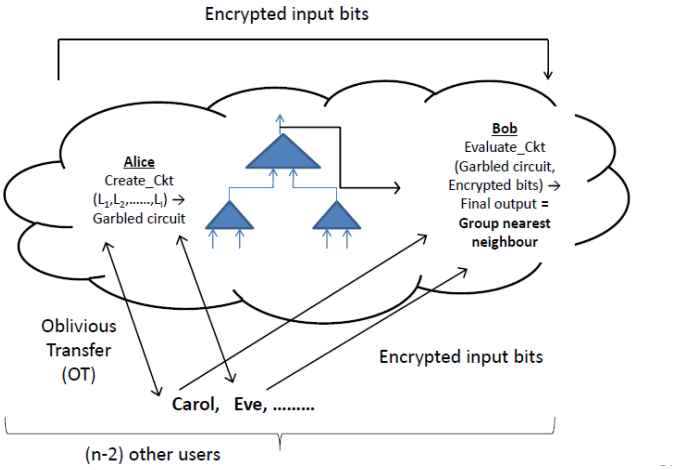


Fig. 1. Illustration of centralized model with $n/2$ users

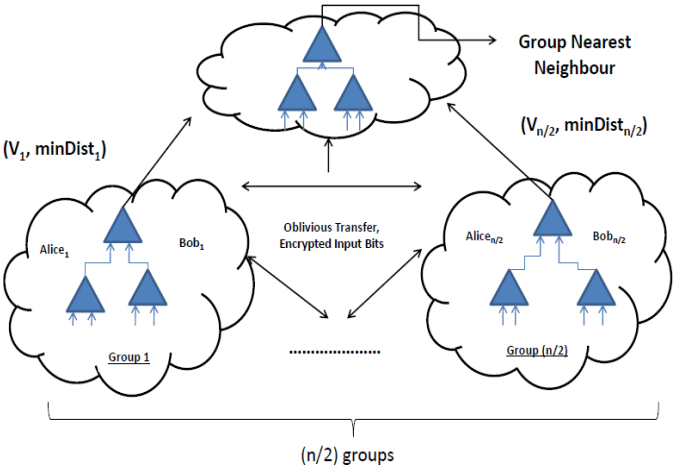


Fig. 2. Illustration of distributed model with $n/2$ users

**Algorithm** GNN_DISTRIBUTED

1) Initialization: In this step, the $n$ users and $l$ locations are divided into groups and a pair of users are assigned to be responsible for circuit creation (Alice) and evaluation (Bob) for a single group of locations. If we have a set of users/parties $P_1, \cdots, P_n$, arranged in the ascending order of their network ID's, then we randomly choose one user, $P_i$ and create pairs or groups of users from that user on in a round-robin fashion. For example, after randomly picking user $P_i$, the groups of users would be created thus: $(P_i, P_{i+1}), (P_{i+2}, P_{i+3}), \cdots, (P_{i-2}, P_{i-1})$. At the end of this step, each group of locations will have 2 users: Alice and Bob who are responsible for creating and evaluating the circuit for computing the group nearest neighbour of all users among this group of locations. Regarding formation of groups, in the case where $l \geq n$, the users are divided into groups of up to $n/2$ users or $(n-1)/2$ users if $n$ is odd. The locations too are divided into $n/2$ or $(n-1)/2$ groups depending on whether $n$ is even or odd with $2l/n$ locations in each group. If $l < n$, the locations are divided into groups with $l \bmod n$ locations each. In this case, we would start forming user pairs/groups starting from a random user $P_i$ and assign a location group to each successive pair until we have exhausted all locations.
2) Circuit creation: Each group's circuit creator - Alice creates a boolean circuit $C$, with $W$ wires. Next, Alice creates the signals and semantics for each wire, and finally the garbled inputs and the gate labels for each gate of the circuit. This process is the same as outlined in the garbled circuit description in the above paragraph titled "Garbled Circuit". Note that the circuits of all groups will be created and evaluated in parallel as the circuit creation and evaluation process of each group is independent of the others.
3) Circuit evaluation: After each group's Alice creates the circuit for her group, Bob - the evaluator of each group evaluates the final output bit of his circuit. Once each sub-circuit is evaluated, the evaluators of all groups will together construct a circuit that computes the location that has the minimum of sum of distances from all their individual output locations, which will be the group nearest neighbour of the entire group.

## IV. EXPERIMENTAL EVALUATION

We developed a prototype of our framework in Java and implemented the centralized and distributed models of garbled circuit creation and evaluation. We varied the number of locations between 200-2000 and considered the number of users to be 10. In the experiments, we considered a fully distributed model i.e., where all users in the system participate in circuit creation and evaluation. One of the main goals of our experiments was to show how scalable our models were and whether or not they are applicable in real-world situations. Hence we had to consider as large a number of locations as realistically possible. The garbled circuit we built for the group nearest neighbour function consists of two kinds of gates: a set of adder gates and a comparator gate. We assumed the input bit length of each party to be 64 bits ($m = 64$). For computing the sum of the distances of each party $P_1, P_2, \cdots, P_n$ from each location $L_1, L_2, \cdots, L_l$ — $s_1 = \sum_{i=1}^{n} d(P_i, L_1), s_2 = \sum_{i=1}^{n} d(P_i, L_2), \cdots, s_l = \sum_{i=1}^{n} d(P_i, L_l)$, we created an adder gate for each location; in total we would have $lm-$bit adder gates, one for each location. Each adder was implemented as a ripple-carry adder [13], e.g., a 64-bit adder was constructed by connecting 64 1-bit adders in series. After the sums of distances have been computed, we need to compare them to find the minimum sum of distance, which we modeled using a single $m-$ bit comparator gate. The total number of gates used in our approach is $(l+1)m-$bit gates.

We measured the performance of the centralized model and distributed models with the number of locations ranging from 200-2000. Fig 3 shows the time taken in seconds in the centralized and distributed models for a single user to create

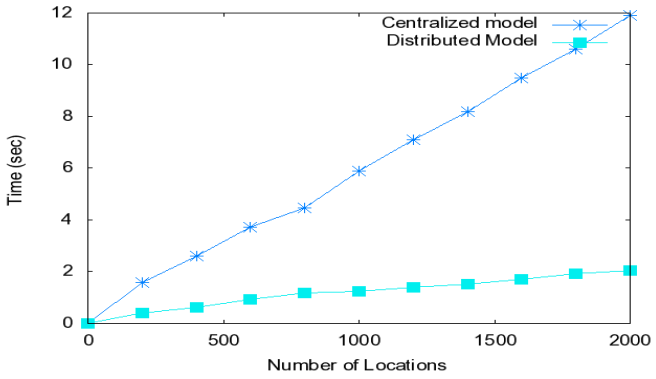


Fig. 3. Time taken for creating the circuit in the centralized and distributed models

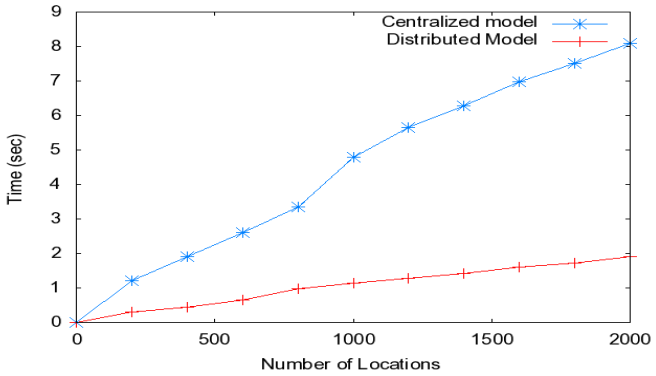


Fig. 4. Time taken for evaluating the circuit in the centralized and distributed models

the circuit. Fig 4 shows the time taken in seconds for a single user to evaluate the circuit in the centralized and distributed models. The time taken for creating a circuit is more than that for evaluating a circuit since creating a circuit involves more expensive steps like generating random keys, encrypting the plaintext values with them, and generating the truth tables. In the centralized model, a single user creates and a single user evaluates the circuit; in the distributed model, the locations and users are divided into groups, with each user group being responsible for creating and evaluating the circuit for their assigned group of locations, hence the workload on each user in distributed setting is far lower than that in the centralized setting.

The communication costs in both the centralized and distributed models are dominated by the Oblivious Transfer (OT) operation between the circuit creator and all other users and the number of OT's would be the same in both the centralized and distributed models. This is because in the centralized model, all users would perform an OT with a single circuit creator over all locations. In the distributed model, each user will perform OT's with many other circuit creators, but the number of OT's performed with each circuit creator would be the same as the number of locations that creator is responsible for. For example, if there are 200 locations and 10 users, in the centralized model, each user would perform 200 OT's with a single circuit creator. In the distributed model, each user will perform 40 OT's with 5 circuit creators; in both cases, the total number of OT's remain the same. Hence the communication costs of the centralized and distributed models are the same.

## V. Conclusion

In this paper, we have explored the application of secure multi-party computation techniques to address privacy issues in location-based queries. We have considered the group nearest neighbour query and shown how one can guarantee privacy of all the users involved without requiring a trusted third party and when none of the users trust each other. Compared to previous work in this area which has mainly considered the client-server model and proposed ways to achieve client privacy, we consider the peer-to-peer model and show how to preserve privacy of all peers involved. Our experimental results show that our approach is practical and has reasonable costs. As future work, it might be interesting to explore privacy issues in other kinds of location-based queries, such as $kNN$ queries, extended nearest neighbour queries, and investigate if one can apply cryptographic protocols to address them. Also, another direction of research is to explore a more stringent adversary model than the semi-honest model considered in this paper such as the dishonest model of peers.

## References

[1] C. Y. Chow, M. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based services," in *Proc. of the ACM GIS*, 2006, pp. 247–256.

[2] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous location-based queries in distributed mobile systems," in *Proc. of the 1st Int. Conference on World Wide Web (WWW)*, 2007, pp. 371–380.

[3] B. Gedik and L. Liu, "Privacy in mobile systems: A personalized anonymization model," in *Proc. of ICDCS*, 2005, pp. 620–629.

[4] M. Mokbel, C. Chow, and W. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proc. of VLDB*, 2006, pp. 219–229.

[5] A. Khoshgozaran and C. Shahabi, "Private information retrieval techniques for enabling location privacy in location-based services," in *Proc. of Privacy in Location-based Applications*, 2009, pp. 59–83.

[6] M. Yiu, C. Jensen, X. Huang, and H. Lu, "Spacetwist: managing the trade-offs among location privacy, query performance and query accuracy in road networks," in *Proc. of ICDE '08*, 2008, pp. 366–375.

[7] M. Yiu, N. Mamoulis, and D. Papadias, "Aggregate nearest neighbour queries in road networks," in *Proc. of IEEE TKDE*, 2005, pp. 820–833.

[8] D. Papadias, Y. Tao, J. Zhang, and N. Mamoulis, "Query processing in spatial network databases," in *Proc. of VLDB*, 2003, pp. 802–813.

[9] A. Yao, "How to generate and exchange secrets," in *Proc. of* $27^{th}$ *IEEE Symposium on Foundations of Computer Science FOCS '86*, 1986, pp. 162–167.

[10] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. of 19th ACM Symposium on Theory of Computing*, 1987, pp. 218–229.

[11] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," in *Proc. of ACM Symposium on Theory of Computing STOC '88*, 1988, pp. 1–10.

[12] S. Tate and K. Xu, "On garbled circuits and constant round secure function evaluation," CoPS Lab, University of North Texas, Tech. Rep. 2003-02, 2003.

[13] V. Nelson, H. Nagle, B. Carroll, and D. Irwin, *Digital logic circuit analysis and design*. New Jersey: Prentice Hall, 1995.

978-1-4244-5638-3/10/$26.00 ©2010 IEEE