A Two-level Protocol to Answer Private Location-based Queries

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

***Abstract*—An important privacy issue in Location Based Services (*LBS*) is to hide a user's identity and location while still providing quality location based services. A user's identity can be easily hidden through anonymous web browsing services. However, a user's location can reveal a user's identity. For example, a user at home may want to ask queries such as "Find the nearest hospital around me" through a GPS enabled mobile phone but he may not be willing to dislose his own location. A common way to achieve location privacy is through cloaking, e.g. the client sends a cloaked region to the server and filters the results to find the exact answer. Recently, Private Information Retrieval has been adopted to answer private location-based queries. However, we argue that ensuring the server does not reveal more data than what is queried is important at the same time. In this paper, we propose an efficient two-level solution based on two cryptographic protocols: PIR and Oblivious Transfer. Our solution is a general-purpose one and can use either a two-level PIR [2] or it can use a combination of PIR and Oblivious Transfer [11]. Our approach provides privacy for the user/client, does not use a trusted party or anonymizer, is provably privacy-preserving, and when compared to previous approaches ensures that the server reveals as minimum data as is required, and the data that is released by the server is as fine-grained or precise as possible.**

I. Introduction

Location based services are becoming increasingly common with location-enabled client user devices like mobile phones, or PDA's making queries to location servers. Clients may not want to inadvertantly reveal their own identity and location while querying the server. A client's identity can be hidden by using a fake one, but their location is needed to answer location-based queries. In order to hide its own location, a client can query a large region and filter the query results to find its exact location within that region. As a result the server needs to respond to the clients' queries with more data than what is needed. However, the server may not want to reveal the whole database to the client. Hence, we need to acheive a trade-off between the privacy of the client and the amount of data supplied by the server.

Most solutions proposed till now take into account the privacy of the client. In addition to this, we also need to ensure that while protecting the privacy and anonymity of the client, we do not inadvertantly release more information than what is required about other data stored in the server. Typically a location server might contain thousands of point of interests (POIs) listed in a particular category (e.g. all restaurants in a given area) organized into a table, and if a user queries this table, the server does not want to return every item in its listing. In addition to being redundant, this would also enable a user to engage the server's resources for long periods of time. Hence we need to ensure that the data returned is fine-grained and precise (very close to the user's location). This would also save the user time, since (s)he does not have to sort through reams of redundant or useless data. That is, if the server maintains a number of POIs with each type organized into a table, it may allow clients to query part of the tables about a limited region around a specific location, but not access all (or large amounts of) the data stored in it. For example, the server may allow queries like: "How many people (or users) are within 1 mile of me", or "Where is the nearest gas station from me"? But the server may not want the client to ask queries along the lines of "Give me a list of restaurants in the entire state of Texas".

One of the common queries in location services is user queries about their nearest neighbour(s). A user might query the location server about finding its approximate or exact nearest neighbour. Or it might query the server about k nearest neighbours. In this paper, we present a solution for the exact nearest neighbours query. To paraphrase the problem we are trying to solve: In location-based services, we need to find an efficient way to protect the privacy and anonymity of the client, while ensuring that the server answers location-related queries in as precise a manner as possible, keeping the redundant data to a minimum, and has some control over what data is revealed. Specifically, if a user u has a location l and needs to query a LBS to obtain its nearest neighbor from a set of POIs S where $S = \{s_1, s_2, \ldots, s_m\}$, the goal is to develop an efficient protocol that does not allow the LBS to learn l while reducing the subset of S to be sent to u in order to answer the query.

II. Related work

Previous work in privacy in location services focus on cloaking techniques using anonymizers [4], [8], [10]. In this approach, service requests are first transmitted to a third-party trusted server called location anonymizing server (AS). The AS strips off a service requester's identifying information, such as user's network address and name. Various perturbation operations can be performed on the AS to obfuscate the original location information. Location based service (LBS) requests are then issued from the AS to the LBS providers. The AS can also help filter the query results and return the exact

978-1-4244-4173-0/09/$25.00 ©2009 IEEE

ISI 2009, June 8-11, 2009, Richardson, TX, USA

answers to the original requesters. One major problem with this approach is that the anonymizer becomes a single point of failure and a performance bottleneck, besides the client and server will have to trust the anonymizer.

Another solution is to organize the network as a peer-to-peer network, instead of having a centralized location server [3], [6]. In this approach, a mobile client first finds a peer group that meets the privacy requirement (i.e. find $\mathcal{K}-1$ neighbors) through peer searching. Then the client calculates a region that includes the $\mathcal{K}-1$ peers (called K-anonymity [7]). It randomly chooses one of its neighbors as the agent to request *LBS* using the cloaking region. The query results are eventually forwarded to the original client through the agent. Since the disclosed location of a requester is expanded to an area that includes at least $\mathcal{K}-1$ other mobile users, any of the $\mathcal{K}$ people within the disclosed area could have been the user. For private locations that are likely to release a requester's identification, the cloaking area is generally large if we use K-anonymity due to the small number of users in private areas. This approach requires the construction of a large peer network of mutually trusting and honest uers and cannot guarantee service availability when clients are sparsely distributed.

The paper by Shahabi et al. [5] presents a solution based on Private Information Retrieval [2]. This approach has several advantages: it does not reveal any spatial information, it is resistant against correlation attacks, and it does not require any trusted third party among others. The paper first presents a PIR-based framework for location based services and then presents solutions to the problems of finding approximate nearest neighbours and exact nearest neighbours. The approach is expensive in terms of communication and computation; some of the improvements and optimizations they have proposed include compressing the data sent by the server, having rectangular grids/matrices in addition to square ones, and using offline data mining to avoid redundant computation. However the server still reveals large amounts of extra information which results in exposing large portions of the server's data assets to the client. Another paper that deals with nearest neighbour queries is the one by Khoshgozaran and Shahabi [9] which uses tamper-proof hardware on both the server and client side. The basic idea is to utilize one-way transformations to map the space of all static and dynamic objects to another space and resolve the query blindly in the transformed space.This approach requires that the transformation used preserves relative proxmity of POIs.

A. Our Contributions and Outline

We present an efficient solution for the exact nearest neighbour query where along with ensuring the client's privacy, the data returned by the server is as precise as possible. Our solution is a general-purpose one and can use either a two-level PIR [2] or it can use a combination of PIR and Oblivious Transfer [11]. It uses the layout of the server database similar to that in [5]. In our appoach, the server super-imposes a grid on the POIs. A user u in a grid cell wants to know its nearest neighbor but does not want to release its own location or the grid cell that (s)he is in. Using the single PIR approach proposed in [5] on the grid will result in the server returning an entire column of the grid, thus releasing more information than is necessary to the client. We note that if the data (locations) are not arranged in a grid, but as a one-dimensional list, we can still use PIR to return just one cell, but this approach would be very expensive. Using 1-of-n Oblivious Transfer (OT) [11] by itself will require the server to transmit the whole grid of encrypted POIs to the client, the cost of which is prohibitive. Thus, by using OT over the entire grid, we would have incurred very high communication costs, using single-level PIR over the columns of the grid, the volume of data given/revealed by the server would be too high, and using single-level PIR over cells of the grid would return precise data, but would be too expensive. One way to solve this problem would be to use a two-level protocol wherein the server recursively performs a PIR on the grid, and the other way is to define a protocol that combines the good features of PIR and OT. In Section III, we describe and analyze the PIR and OT protocol, and in Section IV, we give a high-level description of the two-level PIR protocol and describe how the client can get information only associated with its cell in the grid, and not any other data using 1-of-n Oblivious Transfer (OT) [11] along with PIR. In Section V, we formally define the security properties of our protocol and give a proof sketch. We algebraically analyze the costs of our approach compared with regular PIR and OT approaches and show that for an m-bit string and a grid of n cells, our PIR + OT approach is substantially ($O(m \cdot n + \sqrt{n} \cdot \log \sqrt{n})$ less expensive than OT ($O(n + \log n)$), and is comparable to the cost of PIR ($O(m \cdot 2\sqrt{n})$), while the two-level PIR approach has a cost of $O(m \cdot \sqrt{n})$. The alternative two level PIR approach is described in Section VI.

III. Preliminaries and Background

In this section, we give a brief introduction to PIR and OT; the two cryptographic protocols which are used as building blocks in our protocol.

1) Private Information Retrieval (PIR): Private Information Retrieval first introduced by Goldreich et al. [2] is a technique or protocol to let a user retrieve information from a database without the database server knowing the element or record that has been retrieved. If the database server has an n-bit string $X_1, ..., X_n$, and the client wants to know the value of X_I, the client sends a request to the server in the form of an obfuscated vector, $E(I) = q$, where E denotes the algorithm used for generating the obfuscated vector. The server responds with a value $v(X, q)$. Using this, the client can compute the value of X_I. PIR theory is based on the computational intractibility of deciding whether a given number is a quadratic residue of a composite modulus or not. In the following paragraph, we give a high-level description of PIR.

Let p and p' be large primes, and $N = p \cdot p'$ be a composite modulus. Let Z_N^* denote the set of numbers that are co-prime or relatively prime with N. The set of *quadratic residues* is defined by: $QR = \{y \in Z_N^* | \exists x \in Z_N^* : y = x^2 \bmod N\}$, and

the set of *non-quadratic residues* is the complement of QR, say QR'. If we choose a set $Z_N^{+1} = \{y \in Z_N^* | (y/N) = 1\}$, where (y/N) denotes the Jacobi symbol, exactly half of the numbers in Z_N^{+1} are in the set QR and the other half are in QR'. It is computationally intractable to determine whether a given number is in QR or in QR'. For a detailed theoretical exposition, readers are referred to [2]. The query $E(I) = q = [q_1, ..., q_n]$ $(q_i \in Z_N^{+1}, i = 1, 2, \cdots, n)$ has $q_I \in QR'$ and $\forall i \neq I, q_i \in QR$. The server computes $v(X, q) = \Pi_{j=1}^{n} w_j$, where $w_j = q_j^2$ if $X_j = 0$, or q_j otherwise. Upon receiving $v(X, q)$, the client knows the value of $X_I = \neg((v(X, q)^{\frac{p-1}{2}} = 1 \, mod \, p) \wedge (v(X, q)^{\frac{p'-1}{2}} = 1 \, mod \, p'))$.

2) Oblivious Transfer (OT): Oblivious Transfer is a protocol that was introduced by Brassard, Crépeau and Roberts [1] and is used in various two-party secure function evaluation protocols. Consider a setting where Bob has a string of bits $X_1, ..., X_n$ and Alice wants to know one of them. Alice does not want Bob to know which bit she has chosen, and Bob does not want Alice to know any bit other than the one she has chosen. The solution for this is known as the 1-of-n OT protocol. There are two variants of this protocol: the 1-of-2 OT protocol and the k-of-n OT protocol. In 1-of-2 OT, Bob has 2 elements, X_0 and X_1. Alice chooses to know one of them. Alice receives exactly one element without learning anything about the other element, while Bob remains *oblivious* as to which element was sent. In the k-of-n OT protocol, Bob has a string of bits $X_1, ...X_n$, Alice wants to know a substring $X_i, ..., X_m$, where $1 \leq i \leq m \leq n$. The general approach is that Alice and Bob agree on a set of symmetric keys, and Bob masks each input bit with a product of a *different subset* of keys (No two bits are masked with the same keys). Bob then sends all the masked bits to Alice. Alice picks the key subset for the bit she wants to know, and uses this to decrypt her bit. The details of the 1-of-n protocol are given in the next section.

IV. A TWO-LEVEL PIR + OT PROTOCOL

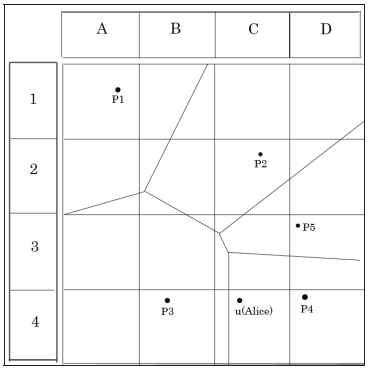
For organizing the location database, we follow the same basic methodology given in [5]. As part of pre-processing, the whole space is divided into Voronoi tesselations using the set of POIs, $S = \{s_1, s_2, \ldots, s_m\}$ and a grid or matrix of size $M = \lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ is superimposed on top of it as shown in Figure 1. For each grid cell gc (which can also be referred to as $M_{a,b}$ where a is the row number and b is the column number), the server computes the POIs located in all the Voronoi cells that intersect gc and prepares a neighbor list that contains the possible nearest neighbors for any point in the cell. Figure 1 shows the neighbour list for each grid cell. The server sets the maximum number of POIs in the lists to a constant fixed number, P_{max} (e.g. in Figure 1, $P_{max} = 3$). In case some of the cells do not have enough POIs in the list, the server pads up the list with duplicate entries till it reaches the value of P_{max}. This can be seen in Figure 1 in the neighbor lists for A_1, B_2, etc. In the figure, we have used duplicate entries for the padding, but one can use a pre-agreed padding value as well.

When a user u with location l requests its nearest neighbour (the nearest neighbor of l), it simply requests the neighbor list associated with the grid cell that contains l and filters the results to find the nearest neighor of l. Let $M_{a,b}$ be the matrix cell corresponding to the user's location l. Using PIR, in [5], the user selects a query message $y = [y_1, ..., y_{\lceil \sqrt{n} \rceil}]$ $(y_i \in Z_n^{+1}, i = 1, 2, \ldots, n)$ where $y_b \in QR'$ and $\forall_j \neq b, y_j \in QR$ (Here b is the column the user is located in). The server computes for each row, r of the matrix, M, the value $z_r = \Pi_{j=1}^{\lceil \sqrt{n} \rceil}(w_{r,j})$ and returns $z = [z_1, ..., z_{\lceil \sqrt{n} \rceil}]$. Here $w_{r,j} = y_j^2$ if $M_{r,j} = 0$, or y_j otherwise. The client can then use a similar method as introduced in Section III to calculate the values of $M_{*,b}$ where $*$ represents any row in M.

Since the data (locations) are organized in a grid, this leads to the server returning extra information which the user has not queried about, wherein most of the data returned is redundant data. Specifically, if the user requests one grid or matrix cell $M_{a,b}$, the server, along with $M_{a,b}$ returns the contents of the entire column b. This also leads to the additional problem that the size of the data returned increases with the size of the grid. Hence the user has to sift through a lot of redundant data, or data that might not be particularly useful. In the optimization techniques proposed in [5], data compression and using a rectangular matrix instead of a square one are cited as ways to reduce the amount of data returned by the server. By compressing the query results returned by the server, we would just be removing duplicates and reducing the size of the data returned, without reducing the actual content returned. Organizing the data as a rectangular matrix would return as many extra locations as there are rows in the grid. Both of these approaches do not ensure that the server *always* returns just one grid cell and its corresponding neighbours list. This problem may be avoided by organizing the location database as a one-dimensional list or array, but this approach would be too expensive. In our two-level approach, the user first retrieves its encrypted column, b using PIR as in [5] and then uses 1-of-n Oblivious Transfer to retrieve its cell in the column i.e. $M_{a,b}$. Firstly the server generates the matrix M and sends the granularity of the grid to the client. The client and the server then agree on a set of keys. The client chooses which cell (s)he is located in and sends a PIR request to the server e.g. $PIR(M_{a,b})$. The server then encrypts each row of the grid with a *different subset* of keys. The client then engages in a 1-of-n OT protocol with the server for obtaining the particular row (s)he is in. Below we give our two-level protocol: EXACT_NN and two sub-protocols which are used in it: OTSERVER and OTCLIENT.

Algorithm EXACT_NN

1) Server - Bob uses the POIs S to create Voronoi tesselations of the space.
2) Bob divides the space into a grid of $M = \sqrt{n} \times \sqrt{n}$; So the grid is represented by $(M_{1,1}, \cdots, M_{\sqrt{n},\sqrt{n}})$; The neighbor list for each grid cell is prepared. Here, the contents of each grid cell is an m-bit string.
3) User - Alice initiates a query in order to find the



A1	P1,P1,P1
A2	P1,P3,P3
A3, A4, ...	.....
B2	P1,P2,P3
C2	P2,P5,P5
C4	P3,P4,P4

Fig. 1. Sample dataset for algorithm illustration

neighbor list of the cell she is in (for each bit in the neighbour list). Alice also randomly generates a composite modulus $N = p \cdot p'$ and sends N to Bob (p, p' are large primes). Note that only Alice knows the factorization of N.

4) Bob sends the granularity of the grid.
5) Let Alice be located in $M_{a,b}$. Alice selects the column in which her cell is located i.e. column b and issues a PIR request $PIR(b)$ to Bob. Let $PIR(b) = [y_1, ..., y_{\lceil\sqrt{n}\rceil}]$.
6) Bob prepares the PIR response, $z = [z_1, \cdots, z_{\lceil\sqrt{n}\rceil}]$ for each bit of the m-bit string. So there will be m responses for each z_i (denoted by $z_i[1], z_i[2], \cdots, z_i[m]$). Bob then runs the algorithm OTSERVER(Z) where z is the PIR response for each bit of the m-bit string.
7) Alice engages in a 1-of-n OT protocol with Bob for obtaining the a^{th} cell in z. For doing this Alice runs the OTCLIENT algorithm.
8) From the previous step, Alice gets the encrypted values $Y = [Y_1, \cdots, Y_{\lceil\sqrt{n}\rceil}]$, of which Alice can only decrypt z_a. After decrypting z_a Alice can check if each bit $i (i = 1, 2, \ldots, m)$ of $z_a[i] \in QR$ or $z_a[i] \in QR'$ using the formula $(z_a[i]^{\frac{p-1}{2}} = 1 \bmod p) \wedge (z_a[i]^{\frac{p'-1}{2}} = 1 \bmod p')$. For each bit, if $z_a[i] \in QR$, then Alice concludes that bit is 0, else if $z_a \in QR'$, the bit is 1. Alice repeats this process for each bit of the m-bit string. Once she does this, she can get the contents of her cell $M_{a,b}$. From the contents of $M_{a,b}$, she can get the list of neighbours (POIs) associated with $M_{a,b}$. Once Alice knows the set of POI's, she can calculate their individual distances and find her nearest neighbour.
9) Return the nearest neighbour.

Algorithm OTSERVER(X)

1) Let $X = [X_1, X_2, \cdots, X_n]$.
2) The client would like to know the contents of one element X_I. Each element X_i of X can be represented by a bit string: $X_i \in \{0,1\}^l$, where where $l = \log_2 n$.
3) The server chooses a symmetric encryption algorithm (e.g. AES, 3DES, etc.) E_K, which uses key K and generates l random pairs of keys. The pairs of keys are represented by $(K_1^0, K_1^1), (K_2^0, K_2^1), \cdot, (K_l^0, K_l^1)$. So, K_j^b is a key for the encryption algorithm E_K $\forall 1 \leq j \leq l$ and $\forall b \in \{0,1\}$.
4) For each $1 \leq i \leq n$, let $\langle p_1, p_2, ..., p_l \rangle$ represent the bits of i, the server creates an encryption: $Y_i = X_i \oplus \bigoplus_{j=1}^{l} (E_{K_j^{p_j}}(i))$. Thus, there are a total of n encryptions.
5) For all $1 \leq j \leq l$, the server engages in a 1-of-2 OT with the client on the strings $\langle K_j^0, K_j^1 \rangle$.
6) The server sends the strings $Y = Y_1, \cdots Y_n$ to the client.

Algorithm OTCLIENT

1) The client or user picks an index I where $\langle p_1, p_2, ..., p_l \rangle$ represents the bits of I to get from the server.
2) The client engages in a 1-of-2 OT with the server to learn the key $K_j^{p_j}$ for all j $(1 \leq j \leq l)$.
3) Once the client gets the value of $Y = Y_1, \cdots Y_n$ from the server, it can reconstruct X_I thus: $X_I = Y_I \oplus \bigoplus_{j=1}^{l} (E_{K_j^{p_j}}(i))$.

A. An example

We give a sample execution of our protocol outlining the PIR and OT steps w.r.t. Figure 1. Let Bob, the LBS, set up a grid as shown in Figure 1. The user or client, Alice, is located in cell $C4$ and wants to know the list of POIs associated with her cell, i.e. $\{P3, P4, P4\}$. Here $n = 16$ where n is the total number of cells in the grid, and $\sqrt{n} = 4$. We use $PIR(l)$ to denote that Alice initiates the PIR protocol to retrieve privately a location l and the associated POIs of l from Bob. Using our protocol:

1) Alice knows her location in the grid i.e. cell $C4$ which corresponds to $M_{4,3}$, and sends a PIR request to Bob to obtain the contents of her cell: $PIR(C4) = y = [y_1, y_2, y_3, y_4]$. Here $y_1, y_2, y_4 \in QR$ and $y_3 \in QR'$.
2) Bob prepares a PIR reponse, $z = [z_1, z_2, z_3, z_4]$. Here $z_r = \Pi_{j=1}^{\lceil 4 \rceil}(w_{r,j})$, and $w_{r,j} = y_j^2$ if $M_{r,j} = 0$, or y_j otherwise. So, if $z_4 \in QR$, $M_{4,3} = 0$, else $M_{4,3} = 1$.
3) Alice then initiates a 1-of-4 OT with Bob to obtain z_4 out of z. For doing this, Alice and Bob agree on a symmetric encryption algorithm E which uses key k, and a set of keys, $l = \log_2 \sqrt{n} = 2$. Bob then encrypts each element of z with a different subset of keys. So for each element i in $z = [z_1, z_2, z_3, z_4]$, Bob creates $Y_i = z_i \oplus \bigoplus_{j=1}^{l} (E_{K_j^{p^j}}(i))$ where $\langle p_1, p_2 \rangle$ represents the bits of z_i.
4) Bob sends the strings $Y_1, \cdots, Y_4$ to Alice.
5) Alice reconstructs or decrypts the value of z_4 thus: $z_4 = Y_4 \oplus \bigoplus_{j=1}^{l} (E_{K_j^{p_j}}(4))$.
6) After getting z_4, Alice checks if $z_4 \in QR$ or $z_4 \in QR'$. Alice can get her location, $M_{4,3}$ thus: If $z_4 \in QR$, $M_{4,3} = 0$, else $M_{4,3} = 1$. Here getting $M_{4,3}$ means obtaining the contents of $M_{4,3}$ i.e. the (list of) nearest neighbours of $M_{4,3}$.
7) Once Alice gets her list of neighbours $\{P3, P4, P4\}$, she can calculate which one of them is her nearest neigbour based on their distance.

V. PRIVACY/SECURITY PROPERTIES AND PROOF SKETCH

In this section, we define the security properties that would be required for the client and server and then show that our protocol satisfies them. Since our protocol only deals with computational security (it is secure under standard intractability assumptions and not unconditionally secure in an information theoretic sense), we consider both, the server and client to be probabilistic polynomial time Turing machines (*PPT*).

A. Definition of privacy and security properties

1) Intuitive Description: While expressed formally, the properties in Definition 5.1 have simple intuitive descriptions. The *Correctness* property states that if the client and server are honest and the client requests a grid/matrix cell i, the server will return i and no other cell. The *Client's Privacy* property states that the view of the transaction that the server sees in case the client requests cell i and cell i' are computationally indistinguishable from each other. The *Server's Privacy* property states that if the client requests a cell i, the client does not learn the contents of any other cell $i' \neq i$ in the server's grid.

2) Formal definition:

*Definition 5.1 (*EXACT_NN *Protocol Security Properties):* Let $S = \{S_1, \cdots, S_{n_1}\}$ and $C = \{C_1, \cdots, C_{n_2}\}$ denote the sets of *PPT* servers and clients respectively. Let $P = \{P_1, \cdots, P_n\}$ denote the locations of $c \in C$ and $M_1, \cdots, M_n$ be the total number of cells in the grid or matrix, where $n \in \mathbf{Z}$. Let $i = M_{a,b}$ denote a location the client has requested. Let $V_s(\cdot)$ be a function that denotes the server's view of what the client requests, i.e. the index of the client's location, and $V_c(\cdot)$ a function that denotes the client's view of the server's grid, i.e. the contents of the grid cells. Let $R_c(\cdot)$ be a request function (for a single request of the client c for a cell i), and $Rec_c(\cdot)$ be a receive function (for the contents of the cell received by the client c). Let ϵ be a negligible constant, where $\epsilon \rightarrow 0$. The following properties must hold:

1) Correctness: For all *PPT* $c \in C$, $s \in S$ and $1 \leq i \leq n$

$$Pr[R_c(i), Rec_c(i'); i \neq i'] = \epsilon$$

2) Client's Privacy: For $1 \leq i \leq n$, $1 \leq i' \leq n$

$$Pr[R_c(i), V_s = i] = \epsilon;$$

3) Server's Privacy: For all $1 \leq j \leq n$,

$$Pr[R_c(i), V_c = j; j \neq i] = \epsilon$$

B. Theorem

Theorem 5.1: The Exact_NN protocol outlined above preserves the privacy of both, the LBS server and a querying client.

Proof: (Proof sketch - the complete proof is in the full version of the paper.)

- Correctness : The correctness of the protocol depends on the client knowing its location, $M_{a,b}$ in the server's grid before querying the server and the server being able to return the exact (encrypted) location. The client would certainly know its location in the grid. Now let us consider the first step of retrieving a column using PIR. We know that the client sends a query $y = [y_1, ..., y_{\lceil\sqrt{n}\rceil}]$ which has $y_b \in QR'$ and $\forall j \neq b, y_j \in QR$. The server then computes for every row r, $z_r = \Pi_{j=1}^{n} w_{(r,j)}$, where $w_{(r,j)} = y_j^2$ if $M_{r,j} = 0$, or y_j otherwise. Based on [2] and PIR theory, the array or list returned by the server, $z = [z_1, ..., z_{\lceil\sqrt{n}\rceil}]$, has to contain z_a. (Either $z_a \in QR$ or $z_a \in QR'$).
  In the second step, where the client engages in a 1-of-n OT transfer with the server on one column, the server encrypts the array/list, z and sends the encrypted column to the client. Since one of the encrypted elements would be z_a, the client would inevitably get the exact cell it requested (in the grid) as long as it knows its own location.
- Client's privacy: Since our protocol is based on PIR [2] and OT [11], and the security proofs in those papers hold good (are privacy preserving), we just need to look at the transition between the two steps. After the server prepares $z = [z_1, \cdots, z_{\lceil\sqrt{n}\rceil}]$ in the last stage of the PIR protocol, it encrypts them with the shared keys established between the client and the server for the 1-of-n OT protocol. z cannot leak any informtion (about z_a, which is the client's row) to the server, since the server does not know the factorization of the composite modulus N. After this, the server just executes the OT algorithm with z as the input array. Hence the server would never get to know $M_{a,b}$.
- Server's privacy: After the server returns the encrypted list $Y_1, \cdots, Y_n$ to the client, the client can recover the value of z_a which is the client's row and test if $z_a \in QR$ or $z_a \in QR'$. Since the OT protocol is privacy-preserving, the client cannot recover any other element in z.

The above arguments show that the desired properties of a privacy preserving LBS protocol are met. ∎

C. Complexity discussion

The communication complexity of our two-level protocol as compared to regular PIR and OT is shown in Table I.

The complexity or cost of our protocol is the sum of the costs of the PIR protocol and the Oblivious Tranfer protocol. The computation cost would be significant or relevant only on the server's side, as the client just has to perform one decryption in the 1-of-n OTCLIENT protocol which is $O(1)$. The total cost would be the sum of the computation cost on the server's side and the communication costs (of both server and client). The protocol we described retrieves only one bit of information. We can extend this to retrieve an object which is represented by a m-bit binary string. In the analysis,we will assume that we are retrieving an m-bit object.

- The communication cost of the user requesting an encrypted column using PIR = $O(\sqrt{n})$

TABLE I
COMPARISON OF OUR PROTOCOL WITH REGULAR OT AND PIR

Action	PIR	OT	Our two-level protocol
Req. by user	$O(\sqrt{n})$	O($\log n$)	$O(\sqrt{n})$
Res. by server	$O(m \cdot \sqrt{n})$	$O(n)$	$O(m \cdot n + \sqrt{n} \log \sqrt{n})$
Total time	$O(m \cdot 2\sqrt{n})$	$O(n + \log n)$	$O(m \cdot n + \sqrt{n} \log \sqrt{n})$

- The computation cost of the server preparing an encrypted column in response to the PIR request = $O(m \cdot n)$
- The communication cost of the user requesting a 1-of-n OT is $O(\log \sqrt{n})$
- The computation cost of the server for OT = $O(\sqrt{n} \cdot \log \sqrt{n})$.
- The communication cost of the server responding to the 1-of-n OT protocol = $O(\sqrt{n})$
- The computation cost of the client decrypting the server's response to the 1-of-n OT protocol is $O(1)$.

Hence the total computation cost for retrieving an m-bit string in our protocol is $O(m \cdot n + \sqrt{n} \log \sqrt{n})$, and the total communication cost is $O(m \cdot \sqrt{n} + \log \sqrt{n})$.

If we had used Oblivious Transfer over the entire grid, the complexity would have been high enough to be un-acceptable. This is due to the fact that the server would have needed to send the the encryptions of *all* cells in the grid to the user. If we compare our approach to the PIR-based approach in [5], the communication cost in [5] is lower: $O(m \cdot \sqrt{n})$. But the drawback of [5] is that it does not return a single cell, it returns the *entire column* in which the cell is located. Hence the communication complexity would be $(m+1) \cdot k \cdot n$, where k is the number of bits in the composite modulus N (refer Section III). A comparison of the communication costs of our protocol with PIR and OT is given in Table I.

The main advantage of our two-level scheme over regular PIR and OT is that the amount of data revealed by the server is very low. Hence if we have a server that allows the user to query some part of the database, but does not want to reveal the entire database, it can do so more effectively with our protocol than with just regular OT and PIR. It is possible to acheive this with PIR over a single-dimensional list or array of data (not a grid), but that would be too expensive. Additionally, this also reduces the burden on the client/user to sort through redundancies in the data the server sends to find the data element or item that they (client) are looking for.

VI. ALTERNATIVE APPROACH: A TWO-LEVEL PIR PROTOCOL

In the previous section, we described a protocol that combines PIR and OT; an alternative approach would be to recursively apply PIR on the grid. In the original PIR paper by Shahabi et al [5], performing PIR once on the grid helped the client to retrieve the column containing its location. If we apply PIR once more on that column, we can retrieve one cell from the column. In this approach, we use the same grid M, of size $\lceil \sqrt{n} \rceil \times \lceil \sqrt{n} \rceil$ as in Section IV, and the pre-processing steps are the same (server dividing the grid into Voronoi tessalations and computing the POIs and neighbour list). For retrieving cell $M_{a,b}$, as a first step, using PIR, the client chooses a composite modulus $N = p \cdot p'$ and sends a query message, $y = [y_1, ..., y_{\lceil \sqrt{n} \rceil}]$ ($y_x \in Z_n^{+1}, x = 1, 2, .., \sqrt{n}$) where $y_b \in QR'$ and $\forall_j \neq b, y_j \in QR$. The server computes for each row, r of the matrix, M, the value $z_r = \Pi_{j=1}^{\lceil \sqrt{n} \rceil}(w_{r,j})$ and returns $z = [z_1, ..., z_{\lceil \sqrt{n} \rceil}]$ (please note this process needs to be done for every bit of the contents of the cell). In the next step, the client sends a request $y' = [y'_1, \cdots, y'_{\lceil \sqrt{n} \rceil}]$ with only $y'_a \in QR'$. For each bit i of the elements in the vector z denoted by $z_j[i]$, the server computes the value $z' = \Pi_{j=1}^{\lceil \sqrt{n} \rceil}(w_{j,i})$ where $w_{j,i} = y'^2_j$ if $z_j[i] = 0$, y'_j otherwise. The server returns z' to the client. The client first recovers z_a from z' (bit by bit) and then $M_{a,b}$ from z_a using a similar approach as described before. The total cost (sum of computation and communication costs) is $O(m\sqrt{n})$.

VII. DISCUSSION AND FUTURE WORK

In this paper, we have proposed a way to acheive user privacy in location-based services, while ensuring the server disseminates as precise data as possible, and does not reveal the entire database to the user. We have also defined the privacy properties desired of such protocols in general and provided a proof sketch for our protocol. As a next step, we plan to compare the performance of our protocol with other schemes e.g. Private Information Retrieval (PIR) [5] and k-anonymity with respect to computataion and communication costs.

There is increasing awareness about Location based services *LBS*, and consequently researchers are examining different ways to provide privacy to the user and the location server. One of the interesting directions of research would be to explore the use of trusted computing technologies in *LBS*.

REFERENCES

[1] G. Brassard, C. Crépeau, and J.M. Roberts. All or nothing disclosures of secrets. In *CRYPTO '86*, pages 234–238, 1986.

[2] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *Proc. of the IEEE FOCS*, pages 41–50, 1995.

[3] 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*, pages 247–256, 2006.

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

[5] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.L. Tan. Private queries in location based services: Anonymizers are not necessary. In *Proc. of the ACM SIGMOD*, pages 121–132, 2008.

[6] 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)*, pages 371–380, 2007.

[7] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03*, pages 31–42, 2003.

[8] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. In *Proc. of the IEEE TKDE*, pages 1719–1733, 2007.

[9] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbour queries using spatial transformation to preserve location privacy. In *Proc. of SSTD*, pages 239–257, 2007.

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

[11] Benny Pinkas and Moni Naor. Computationally secure oblivious transfer. *Journal of Cryptology*, pages 1–35, 2005.