Operative and Secured Privacy enhanced Location Based Services System

M. Shakul Hameed, Mrs. V. InduRani
Asst. Prof MCA
Asst. Prof IT

Abstract- **Location based services (LBS) are becoming increasingly important to the success and attractiveness of next generation wireless systems. However, a natural tension arises between the need for user privacy and the flexible use of location information. In this paper we present a framework to support privacy enhanced location based services using a certificate less-effective key management (CL-EKM) protocol for secure communication. The CL-EKM supports efficient key updates when a node leaves or joins a cluster and ensures forward and backward key secrecy. Recently, wireless sensor networks (WSNs) have been deployed for a wide variety of applications, including military sensing and tracking, patient status monitoring, traffic flow monitoring, where sensory devices often move between different locations. Securing data and communications requires suitable encryption key protocols. In this paper, we propose a certificateless-effective key management (CL-EKM) protocol for secure communication in dynamic WSNs characterized by node mobility. The CL-EKM supports efficient key updates when a node leaves or joins a cluster and ensures forward and backward key secrecy. The protocol also supports efficient key revocation for compromised nodes and minimizes the impact of a node compromise on the security of other communication links. A security analysis of our scheme shows that our protocol is effective in defending against various attacks.**

***Index Terms*— Wireless sensor networks, certificate less public key cryptography, key management scheme**

I INTRODUCTION

Users of mobile devices tend to frequently have a need to find Points of Interest (POIs), such as restaurants, hotels, or gas stations, in close proximity to their current locations. Collections of these POIs are typically stored in databases administered by Location Based Service (LBS) providers such as Google, Yahoo!, and Microsoft, and are accessed by the company's own mobile client applications or are licensed to third party independent software vendors. A user first establishes his or her current position on a Smartphone such as a RIM BlackBerry, Apple iPhone, or Google Android device through a positioning technology such as GPS (Global Positioning System) or cell tower triangulation, and uses it as the origin for the search. The problem is that if the user's actual location is provided as the origin to the LBS, which performs the lookup of the POIs, then the LBS will learn that location. In addition, a history of locations visited may be recorded and could potentially be used to target the user with unexpected content such as local advertisements, or worse, used to track him or her. The user's identity may be divulged through the inclusion of the originating dynamic IP address, e-mail address, or phone number in requests to the LBS server so that the results of an LBS query can be routed back to the correct user via a TCP data connection, e-mail reply, or SMS reply, respectively. If a location can always be correlated to each request, then the user's current pattern of activity and even personal safety is being entrusted to a third party, potentially of unknown origin and intent. Although search engines routinely cache portions of previous queries in order to deliver more relevant results in the future, we are concerned when the user's exact location history is tracked, and not just the key words used in the search.

However, significant challenges still remain in the design of privacy enhanced LBS, and new challenges arise particularly due to data outsourcing. In recent years, there is a growing trend of outsourcing data including LBS data because of its financial and operational benefits. Lying at the intersection of mobile computing and cloud

computing, designing privacy-preserving outsourced spatial range query faces the obstacles below:

• *Obstruction on querying encrypted LBS data*. The LBS provider is not willing to disclose its valuable LBS data to the cloud. As illustrated in Fig. 2, the LBS provider encrypts and outsources private LBS data to the cloud, and LBS users query the encrypted data in the cloud. As a result, querying encrypted LBS data without privacy breach is a big challenge, and we need to protect not only the user locations from the LBS provider and cloud, but also LBS data from the cloud.

•*Obstruction on the resource consumption in mobile devices*. Many LBS users are mobile users, and their terminals are smart phones with very limited resources. However, the cryptographic or privacy-enhancing techniques used to realize privacy-preserving query usually result in high computational cost and/or storage cost at user side.

•*Obstruction on the efficiency of POI searching*. Spatial range query is an online service, and LBS users are sensitive to query latency. To provide good user experiences, the POI search performing at the cloud side must be done in a short time (e.g. a few seconds at most). Again, the techniques used to realize privacy-preserving query usually increase the search latency.

•*Obstruction on security*. LBS data are about POIs in real world. It is reasonable to assume that the attacker may have some knowledge about original LBS data.

To address security, encryption key management protocols for vibrant WSNs have been proposed in the past based on symmetric key encryption. Such type of encryption is well-suited for sensor nodes because of their limited energy and processing capability. However, it suffers from high communication overhead and requires large memory space to store shared pair wise keys. It is also not scalable and not resilient against compromises, and unable to support node mobility. Therefore symmetric key encryption is not suitable for vibrant WSNs More recently, asymmetric key based approaches have been proposed for vibrant WSNs, These approaches take advantage of public key cryptography such as elliptic curve cryptography (ECC) or identity-based public key cryptography (ID-PKC) in order to simplify key establishment and data authentication between nodes. Public key cryptography (PKC) is relatively more expensive than symmetric key encryption with respect to computational costs. However, recent improvements in the implementation of ECC have demonstrated the feasibility of applying public key cryptography to WSNs. PKC is more resilient to node compromise attacks and is more scalable and flexible. We analysed the critical security flaws of that the static private key is exposed to the other when both nodes establish the session key. Moreover, these ECC-based schemes with certificates when directly applied to vibrant WSNs, suffer from the certificate management overhead of all the sensor nodes and so are not a practical application for large scale WSNs. The pairing operation based ID-PKC [5], [9], schemes are inefficient due to the computational overhead for pairing operations.

We present a certificate less effective key management (CL-EKM) scheme for vibrant WSNs. In certificate less public key cryptography (CL-PKC) [10], the user's full private key is a combination of a partial private key generated by a key generation center (KGC) and the user's own secret value. The special organization of the full private/public key pair removes the need for certificates and also resolves the key escrow problem by removing the responsibility for the user's full private key.

In order to dynamically provide both node authentication and establish a pairwise key between nodes, we build CL-EKM by utilizing a pairing-free certificate less hybrid signcryption scheme. Due to the properties of CL-HSC, the pairwise key of CL-EKM can be efficiently shared between two nodes without requiring taxing pairing operations and the exchange of certificates. To support node mobility, our CL-EKM also supports lightweight processes for cluster key updates executed when a node moves, and key revocation is executed when a node is detected as malicious or leaves the cluster permanently. CL-EKM is scalable in case of additions of new nodes after network deployment. CL-EKM is secure against node compromise, cloning and impersonation, and ensures forward and backward secrecy. The security analysis of our scheme shows its effectiveness

II. MODELS AND DESIGN GOAL

In this section, we formalize the system model and attack models considered in this paper, and identify the design goal. *A. System Model*
Privacy-preserving POI query has been studied in two settings of LBS: public LBS and outsourced LBS[4]. In this paper, we focus on the latter setting. In the former setting, there
is an LBS provider holding a spatial database of POI records in plaintext, and LBS users query POIs at the provider's site. In outsourced LBS, as shown in Fig. 2, the system consists of three kinds of entities: LBS provider, LBS users and cloud.

The LBS provider has abundant of LBS data, which are POI records. The LBS provider allows authorized users (i.e. LBS users) to utilize its data through location-based queries. Because of the financial and operational benefits of data outsourcing, the LBS provider offers the query services via the cloud. However, the LBS provider is not willing to disclose the valuable LBS data to the cloud. Therefore, the LBS provider encrypts the LBS data, and outsources the encrypted data to the cloud.

• The cloud has rich storage and computing resources. It stores the encrypted LBS data from the LBS provider, and provides query services for LBS users. So the cloud has to search the encrypted POI records in local storage to find the ones matching the queries from LBS users.

• LBS users have the information of their own locations, and query the encrypted records of nearby POIs in the cloud. Cryptographic or privacy-enhancing techniques are usually utilized to hide the location information in the queries sent to the cloud. To decrypt the encrypted records received from the cloud, LBS users need to obtain the decryption key from the LBS provider in advance.

B. Attack Models
Similar as most previous works on outsourced data query, the cloud is assumed *honest but curious* and considered as the potential attacker in this work. That is, the cloud would honestly store and search data as requested, however the cloud would also have financial incentives to learn those stored LBS
data and user location data in query. Because both LBS data and user location data are valuable, they should be protected and hidden from the cloud. In general, in the outsourced LBS setting, the cloud can observe both queries from LBS users and encrypted LBS data from the LBS provider, which could
be an advantage to learn user locations. Therefore, assuming different abilities of the attacker, there are mainly four attack models in oursourced LBS setting.

• *Ciphertext-only attack*. In this model, the attacker is able to observe the ciphertexts of POIs' locations and queries, but does not know the plaintexts. Obviously, every cloud has this ability. This is a weak attack model.

• *Known-sample attack*. In this model, the attacker knows the plaintexts of some POIs' locations and/or queries. The attacker also knows that their corresponding ciphertexts must exist in all the ciphertexts observed by the attacker. However, the attacker does not know which ciphertext is corresponding to a known plaintext. Utilizing such information, the attacker may be able to reveal the plaintext corresponded to any given ciphertext. Such information is not hard to obtain if the attacker has the background knowledge that the LBS database must contain the POIs of certain type in a certain area.

• *Known-plaintext attack*. In this model, the attacker knows the plaintexts of some POIs' locations and/or queries as well as their corresponding ciphertexts. Utilizing this information, the attacker may be able to reveal the plaintexts corresponded to other ciphertexts.

• *Access-pattern attack*. In this model, the attacker has some background knowledge about the pattern of POI accessing. For example, the attacker knows that a known POI would be the most popular POI. If an encrypted POI appears most frequently in query results, it must be the encrypted version of the known POI. Then the attacker knows that corresponding query points must be close to the known POI.

In addition to the above attacks, other attacks such as insider attacks may be possible. In this paper, we consider ciphertext only and known-sample attacks, which do not require attackers with very strong

abilities. We will leave the attacks requiring very strong abilities for future study.

C. Design Goal

Under the outsourced LBS system model, our design goal is to develop an efficient, accurate and secure solution for privacy-preserving spatial range query [3]. Specifically, the following three objectives should be achieved:

• *Efficiency*. As discussed in Section I, spatial range query has stringent performance requirements. A good solution should not consume many resources of mobile LBS users, and the POI search latency should be acceptable for online query.

• *Accuracy*. It's desirable that a query result contains exact the records matching the query. False negatives would hurt user experience, while false positives would increase communication cost. Additional computational cost is also required at the user side to filter out false positives.

• *Security*. The proposed solution should be resilient to cipher text-only attacks and known-sample attacks. An accurate and efficient solution for spatial range query [1] already exists, which is resilient to ciphertext-only attacks but not to known-sample attacks and more powerful attacks. The proposed solution should be more secure than the solution in [1]. Though subject to more powerful attacks such as known plaintext attacks, the solution proposed in this paper still can be used in many situations where the attackers do not have the required abilities or knowledge. Our solution also has advantages over the solutions resilient to such attacks. As we will see in the related works in Section VIII, such solutions are either very computationally costly or not applicable to outsourced LBS.

III. NETWORK AND ADVERSARY MODELS

A. Network Model

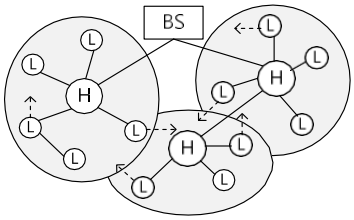

Fig. 1. Heterogeneous vibrant wireless sensor network

We consider a heterogeneous vibrant wireless sensor network (See Fig. 1). The network consists of a number of stationary or mobile sensor nodes and a Base Station (BS) that manages the network and collects data from the sensors [1]. Sensor nodes can be of two types: (i) nodes with high processing capabilities, referred to as H-sensors, and (ii) nodes with low processing capabilities, referred to as L-sensors. We assume to have N nodes in the network with a number N1 of H-sensors and a number N2 of L-sensors, where N = N1 + N2, and N1<<N2. Nodes may join and leave the network, and thus the network size may dynamically change. The H-sensors act as cluster heads while L-sensors act as cluster members. They are connected to the BS directly or by a multi-hop path through other H-sensors. H-sensors and L-sensors can be stationary or mobile. After the network deployment, each H-sensor forms a cluster by discovering the neighbouring L-sensors through beacon message exchanges. The L-sensors can join a cluster, move to other clusters and also re-join the previous clusters. To maintain the updated list of neighbours and connectivity, the nodes in a cluster periodically exchange very lightweight beacon messages. The H-sensors report any changes in their clusters to the BS, for example, when an L-sensor leaves or joins the cluster. The BS creates a list of legitimate nodes, M, and updates the status of the nodes when an anomaly node or node failure is detected. The BS assigns each node a unique identifier. An L-sensor nLi is uniquely identified by node ID Li whereas a H-sensor nHj is assigned a node ID Hj. A Key Generation Center (KGC), hosted at the BS, generates public system parameters used for key management by the BS and issues certificate less public/private key pairs for each node in the network. In our key management system,

a unique individual key, shared only between the node and the BS is assigned to each node. The certificate less public/private key of a node is used to establish pairwise keys between any two nodes. A cluster key is shared among the nodes in a cluster.

B. Adversary Model and Security Requirements

We assume that the adversary can mount a physical attack on a sensor node after the node is deployed and retrieve secret information and data stored in the node. The adversary can also populate the network with the clones of the captured node. Even without capturing a node, an adversary can conduct an impersonation attack by injecting an illegitimate node, which attempts to impersonate a legitimate node. Adversaries can conduct passive attacks, such as, eavesdropping, replay attack, etc to compromise data confidentiality and integrity the adversary can perform a known-key attack to learn pairwise master keys if it somehow learns the short-term keys, e.g., pairwise encryption keys [11], [12]. In order to provide a secure key management scheme for WSNs supporting mobile nodes, the following security properties are critical:

Compromise-Resilience:

A compromised node must not affect the security of the keys of other legitimate nodes. In other words, the compromised node must not be able to reveal pairwise keys of non-compromised nodes. The Compromise - resilience definition does not mean that a node is resilient against capture attacks or that a captured node is prevented from sending false data to other nodes, BS, or cluster heads.

Resistance against Cloning and Impersonation:

The scheme must support node authentication to protect against node replication and impersonation attacks.

Forward and Backward Secrecy:

The scheme must assure forward secrecy to prevent a node from using an old key to continue decrypting new messages. It must also assure backward secrecy to prevent a node with the new key from going backwards in time to decrypt previously exchanged messages encrypted with prior keys. Forward and backward secrecy are used to protect against node capture attacks.

Resilience against Known-Key Attack:

The scheme must be secure against the known-key attack

IV.OVERVIEW OF THE CERTIFICATELESS EFFECTIVE KEY MANAGEMENT SCHEME

In this paper, a Certificate less Key Management scheme (CL-EKM) that supports the establishment of four types of keys, namely: a certificate less public/private key pair, an individual key, a pairwise key, and a cluster key. This scheme also utilizes the main algorithms of the CL-HSC scheme in deriving certificate less public/private keys and pairwise keys. We briefly describe the purpose of these keys and how they are setup.

A. Types of Keys

Certificate less Public/Private Key:

Before a node is deployed, the KGC at the BS generates a unique certificate less private/public key pair and installs the keys in the node. This key pair is used to generate a mutually authenticated pairwise key.

Individual Node Key:

Each node shares a unique individual key with BS. For example, a *L*-sensor can use the individual key to encrypt an alert message sent to the BS, or if it fails to communicate with the *H*-sensor. An *H*-sensor can use its individual key to encrypt the message corresponding to changes in the cluster. The BS can also use this key to encrypt any sensitive data, such as compromised node information or commands. Before a node is deployed, the BS assigns the node the individual key.

Pairwise Key:

Each node shares a different pairwise key with each of its neighboring nodes for secure communications and authentication of these nodes. For example, in order to join a cluster, a *L*-sensor should share a pairwise key with the *H*-sensor. Then, the *H*-sensor can securely encrypt and distribute its cluster key to the *L*-sensor by using the pairwise key. In an aggregation supportive WSN, the *L*-sensor can use its pairwise key to securely transmit the sensed data to the *H*-sensor. Each node can dynamically establish the pairwise key between itself and another node using their respective certificate-less public/private key pairs.

Cluster Key:

All nodes in a cluster share a key, named as cluster key. The cluster key is mainly used for securing broadcast messages in a cluster, e.g., sensitive commands or the change of member status

in a cluster. Only the cluster had can update the cluster key when a *L*-sensor leaves or joins the cluster.

V. THE DETAILS OF CL-EKM

The CL-EKM is comprised of 7 phases: *system setup, pairwise key generation, cluster formation, key update, node movement, key revocation,* and *addition of a new node*.

A. System Setup

Before the network deployment, the BS generates system parameters and registers the node by including it in a member list M.

1) Generation of System Parameters:

The KGC at the BS runs the following steps by taking a security parameter $k \in Z$

- Choose a k-bit prime q
- Determine the tuple {Fq , E/Fq , Gq , P}.
- Choose the master private key x ∈ Z and compute the system public key Ppub = x P.
- Choose cryptographic hash functions

2) Node Registration:

The BS assigns a unique identifier, denoted by Li, to each L-sensor nLi and a unique identifier, denoted by Hj, to each H-sensor nHj, where $1 \leq i \leq N1$, $1 \leq j \leq N2$, $N = N1 + N2$. Here we describe the certificate less public/private key and individual node key operations for Li, the same mechanisms apply for H-sensors. During initialization, each node nLi chooses a secret value xLi ∈R Z and computes PLi = xLi P. Then, the BS requests the KGC for partial private/public keys of nLi with the input parameters Li and PLi.

After the key generation for all the nodes, the BS generates a member list M consisting of identifiers and public keys of all these nodes. It also initializes a revocation list R that enlists the revoked nodes. The public/private key Ω, and the individual key are installed in the memory of each node.

B. Pairwise Key Generation

After the network deployment, a node may broadcast an advertisement message to its neighbourhood to trigger the pairwise key setup with its neighbors. The advertisement message contains its identifier and public key. At first, two nodes set up a long-term pairwise master key between them, which is then used to derive the pairwise encryption key. The pairwise encryption key is short-term and can be used as a session key to encrypt sensed data.

1) Pairwise Master Key Establishment:

We describe the protocol for establishing a pairwise master key between any two nodes *nA* and *nB* with unique IDs *A* and *B*, respectively. We utilize the CL-HSC scheme as a building block. When *nA* receives an advertisement message from *nB*, it executes the following *encapsulation* process to generate a long-term pairwise master key *KAB* and the encapsulated key information, $\phi A = (UA, WA)$. Then, *nA* sends *A*, *pkA*, τA and ϕA to *nB*. *nB* then performs *decapsulation* to obtain *KAB*.

2) Pairwise Encryption Key Establishment:

Once *nA* and *nB* set the pairwise master key *KAB*, they generate an HMAC of *KAB* and a nonce $r \in_R Z$ The HMAC is then validated by both *nA* and *nB*. If the validation is successful, the HMAC value is established as the short-term *pairwise encryption key kAB*.

C. Cluster Formation

Once the nodes are deployed, each *H*-sensor discovers neighboring *L*-sensors through *beacon* message exchanges and then proceeds to authenticate them. If the authentication is successful, the *H*-sensor forms a cluster with the authenticated *L*-sensors and they share a common cluster key. The *H*-sensor also establishes a pairwise key with each member of the cluster. To simplify the discussion, we focus on the operations within one cluster and consider the *j* th cluster. We also assume that the cluster head *H*-sensor is *nHj* with *nLi* $(1 \leq i \leq n)$ as cluster members. *nHj* establishes a cluster key *GKj* for secure communication in the cluster.

1) Node Discovery and Authentication:

For node discovery, *nHj* broadcasts an advertisement message containing *Hj* and *pkHj*. Once *nLi* within *Hj* 's radio range receives the advertisement, it checks *Hj* and *pkHj* , and initiates the *Pairwise Key Generation* procedure. Note that *nLi* may receive multiple advertisement messages if it is within the range of more than one *H*-sensor. However, *nLi* must choose one *H*-sensor, may be by prioritizing over the proximity and signal strength. Additionally, *nLi* can record other *H*-sensor advertisements as backup cluster heads in the event that the primary cluster head is disabled. If *nLi*

selects multiple cluster heads and sends a response to all of them, it is considered as a compromised node. *nLi* and *nHj* perform the *Pairwise Key Generation* procedure to obtain a pairwise master key, *KLi Hj* and a pairwise encryption key, *kLi Hj* .

2) Cluster Key Generation:

nHj chooses $x_j \in_R Z$ to generate a cluster key *GKj* as follows :$GKj = HMAC(x_j , Hj)$ of the *j* th cluster, M*j* .

3) Membership Validation:

After discovering all the neighboring nodes *nLi* ($1 \leq i \leq n$) in the *j* th cluster, *nHj* computes *C*4 = *EK*0*H j (Hj,* M*j)* and transmits *C*4 and *Hj* to the BS. After receiving messages from *nHj* , the BS checks the validity of the nodes listed in M*j* . If all nodes are legitimate, the BS sends an acknowledgement to *nHj*. Otherwise, the BS rejects M*j* and investigates the identities of invalid nodes (false or duplicate ID). Then, the BS adds the identities of invalid nodes to the revocation list and reports it to *nHj* . Upon receiving the acknowledge message, *nHj* computes *C*5 = *EGKj (Hj* ,M*j) and* broadcasts *C*5 to all the nodes in *j* th cluster.

D. Key Update

In order to protect against cryptanalysis and mitigate damage from compromised keys, frequent encryption key updates are commonly required. In this section we provide the pairwise key update and cluster key update operations.

1) Pairwise Key Update:

To update a pairwise encryption key, two nodes which shared the pairwise key perform a *Pairwise Encryption Key Establishment* process. On the other hand, the pairwise master key does not require periodical updates, because it is not directly used to encrypt each session message. As long as the nodes are not compromised, the pairwise master keys cannot be exposed. However, if a pairwise master key is modified or needs to be updated according to the policy of the BS, the *Pairwise Master Key Establishment* process must be executed.

2) Cluster Key Update: Only cluster head *H*-sensors can update their cluster key. If a *L*-sensor attempts to change the cluster key, the node is considered a malicious node.

E. Node Movement

When a node moves between clusters, the *H*-sensors must properly manage the cluster keys to ensure the forward/backward secrecy. Thus, the *H*-sensor updates the cluster key and notifies the BS of the changed node status. Through this report, the BS can immediately update the node status in the M. We denote a moving node as *nLm.*

1) Node Leave:

A node may leave a cluster due to node failure, location change or intermittent communication failure. There are both proactive and reactive ways for the cluster head to detect when a node leaves the cluster

2) Node Join:

Once the moving node *nLm* leaves a cluster, it may join other clusters or return to the previous cluster after some period. For the sake of simplicity, we assume that *nLm* wants to join the *l*th cluster or return to the *j* th cluster.

(i) *Join a New Cluster: nLm* sends a join request which contains *Ln*+1 and *pkLn*+1 to join a *l*th cluster. After *nHl* receives the join request, *nLm* and *nHl* perform *Pairwise Key Generation* procedure to generate *KLm Hl* and *kLm Hl* respectively. Next, *nHl* transmits *EK*0*H (NodeJoin, Lm)* to the BS. The BS decrypts the message and validates whether *nLm* is a legitimate node or not and sends an acknowledgement to *nHl* if successful. The BS also updates the node member list, M.

(ii) *Return to the Previous Cluster: nLm* sends a join request which contains *Ln*+1 and *pkLn*+1 to join a *j* th cluster. Once *nHj* receives the join request, it checks a timer for *nLm* which is initially set to the *Thold* . *Thold* indicates the waiting time before discarding the pairwise master key when a L-sensor leaves. If *nLm* returns to the *j* th cluster before the timer expires, *nLm* and *nHj* perform only the *Pairwise Encryption Key Establishment* procedure

F. Key Revocation

We assume that the BS can detect compromised *L-sensors* and *H-sensors*. The BS may have an intrusion detection system or mechanism to detect malicious nodes or adversaries. Although we do not cover how the BS can discover a compromised node or cluster head in this paper, the BS can utilize the updated node status information of each cluster to investigate an abnormal node. In our protocol, a

cluster head reports the change of its node status to the BS, such as whenever a node joins or leaves a cluster. Thus, the BS can promptly manage the node status in the member list, M. For instance, the BS can consider a node as compromised if the node disappears for a certain period of time.

G. Addition of a New Node

Before adding a new node into existing networks, the BS must ensure that the node is not compromised. The new node *nLn*+1 establish a full private/public key through the *node registration* phase. Then, the public system parameters, a full private/public key and individual key *K*0*L n*+1 are stored into *nLn*+1.

VI. CONCLUSIONS AND FUTURE WORKS

The concept of answering location-related information for encrypted positions is promising to improve security needs. Indeed, such a mechanism can strongly attract the attention of researchers as it supports the preservation of the users' privacy.

In this paper we developed a novel fully secure location-based mechanism based on a CL-EKM encryption scheme. We described the circuits that allow a LBS server to process encrypted inputs to retrieve targeted records that match the user's request. We also discussed the limitations and drawbacks of our proposed system and suggested some solutions to make it more practical. The performance of our system was tested through extensive experiments to extract useful results related to the noise generated and the processing time consumed.

As future work, we are planning to improve the performance of the encryption scheme to be able to support a large number of services. This step is mandatory to make it possible for a commercial deployment of our LBS system.

REFERENCES

[1] Seung-Hyun Seo, Jongho Won, Salmin Sultana, and Elisa Bertino, *Fellow, IEEE* "Effective Key Management in Dynamic Wireless Sensor Networks", IEEE Transactions on Information Forensics And Security, Vol. 10, No. 2, Feb 2015 pp.371 - 383

[2] Lichun Li, Rongxing Lu, *Senior Member, IEEE* and Cheng Huang," EPLQ: Efficient Privacy-Preserving Location-based Query over Outsourced Encrypted Data, IEEE Internet of Things Journal

[3] A. Gutscher, "Coordinate transformation - a solution for the privacy problem of location based services?" in *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings,25-29 April 2006, Rhodes Island, Greece*, 2006. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2006.1639681

[4] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *SIGMOD*. ACM, 2009, pp.139–152.

[5] M. Rahman and K. El-Khatib, "Private key agreement and secure communication for heterogeneous sensor networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 8, pp. 858–870, 2010.

[6] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: anonymizers are not necessary," in *SIGMOD*. ACM, 2008, pp. 121–132.

[7] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical k nearest neighbor queries with location privacy," in *ICDE*. IEEE, 2014, pp.640–651.

[8] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 965–981,1998.

[9] K. Chatterjee, A. De, and D. Gupta, "An improved ID-based key management scheme in wireless sensor network," in *Proc. 3rd Int. Conf. ICSI*, vol. 7332. 2012, pp. 351–359.

[10] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. 9th Int. Conf. ASIACRYPT*, vol. 2894. 2013, pp. 452–473.

[11] X. He, M. Niedermeier, and H. de Meer, "Dynamic key management in wireless sensor networks: A survey," *J. Netw. Comput. Appl.*, vol. 36, pno. 2, pp. 611–622, 2013.

[12] S. Agrawal, R. Roman, M. L. Das, A. Mathuria, and J. Lopez, "A novel key update protocol in mobile sensor networks," in *Proc. 8th Int. Conf. ICISS*, vol. 7671. 2012, pp. 194–207.