# BLOT: A Novel Phase Privacy Preserving Framework for Location-Based Services

Abdullah Albelaihy, Jonathan Cazalas, Vijey Thayananthan
King Abdulaziz University
Jeddah, Saudi Arabia

***Abstract*—The inherent challenge within the domain of location-based services is finding a delicate balance between user privacy and the efficiency of answering queries. Inevitably, security issues can and will arise as the server must be informed about the query location in order to provide accurate responses. Despite the many security advancements in wireless communication, servers may become jeopardized or become infected with malicious software. That said, it is possible to ensure queries do not generate fake responses that appear real; in fact, if a fake response is used, mechanisms can be employed for the user to identify the query's authenticity. Towards this end, the paper propose BLoom Filter Oblivious Transfer (BLOT), a novel phase privacy preserving framework for LBS that combines a Bloom filter hash function and the oblivious transfer protocol. These methods are shown to be useful in securing a user's private information. An analysis of the results revealed that BLOT performed markedly better and enhanced entropy when compared to referenced approaches.**

***Keywords*—Privacy; location-based services (LBS); oblivious transfer; BLoom Filter Oblivious Transfer (BLOT); bloom filter**

## I. INTRODUCTION

Owing to the proliferation of smartphones and similar mobile devices, location-based services (LBS) have found widespread use in recent years. In particular, the increasing use of smartphones has contributed to the growing popularity of LBS [1]. Consequently, new means of communication have been developed, and the information obtained from LBS has provided users with greater awareness of their surroundings [2]. Various iOS and Android applications enable users to download and use LBS and submit queries to LBS servers [3].

Users can receive LBS data through points of interest (POIs). For example, users can search for nearby stores or restaurants and check whether the data regarding the prices at these locations are accurate. Thus, LBS find increasing use in various applications. They provide simple solutions for location awareness and location sharing. Therefore, they are regarded as both helpful and advantageous [4].

While proven and demonstrated to be profoundly helpful, the inherent challenge of location-based services is one of juggling between two seemingly diametrically opposed requirements: user privacy and security and the efficiency of server response times when answering user queries. On one hand, the users demand accurate answers to their location-based queries, and, on the other hand, these same users expect their personal information to remain private and secure. The quagmire is that a server simply cannot answer a location-based query without having the location itself. More clearly, one cannot expect an answer to, "what are the nearest gas stations to XYZ address?", unless they are willing to, somehow, someway, reveal the said address. In short, users want to have their cake and eat it too! The challenge, therefore, lies in finding a delicate balance between providing accurate and efficient responses to queries while maintaining an acceptable level of privacy for the user. LBS queries submitted by users must not only be efficient but must also ensure privacy; unfortunately, such queries often lead to security issues. The first issue is one of implementation: is it possible to make queries to a data store (service, database, or website) in a way that reveals the minimum amount of information about the data store while providing maximum utility to the user making the queries? For many years, these conflicting requirements were thought to be difficult, if not impossible, to satisfy. However, recent studies have shown that a variety of query protocols can, in fact, be used to achieve both goals, at least in a probabilistic sense [5].

Protection can be provided by a security protocol even in the presence of a malicious server, malicious eavesdropper, or malicious man-in-the-middle attacker. The development of "1-in-q" algorithms (which are described in detail below) has facilitated the creation of security protocols that can provide probabilistic guarantees about the validity of response even in the presence of a compromised or actively malicious data store. Recently, a major cryptographic breakthrough was made, whereby the possibility of fully homomorphic encryption (FHE) was demonstrated [6]-[8]. In this scenario, the data store operates on opaque binary blobs of encrypted data and has no access to any unencrypted information. FHE makes it possible to assert that responses are unforgeable, not merely probabilistically unforgeable.

The use of advanced cryptographic algorithms, such as FHE, has led to the third issue associated with privacy-preserving queries: the question of efficiency. It is known that "1-in-q" algorithms exhibit runtime performance of the order of O(n) or even O(1); thus, the strength of the probabilistic guarantee is only limited by the user's privacy goals. The accuracy of the response can be increased by incurring a linear or even constant time computational overhead. Partially homomorphic encryption schemes, such as routing and spectrum allocation (RSA), may exhibit exponential computational growth. However, extensive experience with such algorithms has led to the development of some implementation heuristics that allow the computations to avoid the exponential neighborhood. Thus, the computations will take polynomial time, not exponential time, except with negligible probability [9], [10]. Efficient computation has not

been realized for FHE, thus far. At present, the implementation technology is immature and even a straightforward FHE computation may take several seconds [11], [12]. Obviously, current FHE performance is not compatible with end-user requirements [13], [14].

Finally, one must consider the requirements specifically imposed by location-based data. The response to an LBS query is not a single data item but an ensemble of data items [15], [16]. Thus, malicious actors, such as a man-in-the-middle eavesdropper, may find it easier to make correlations over multiple queries that lead to an unacceptable loss of privacy. Any privacy preserving solution to the query problem for location-based servers must take such behavior into account [17], [18].

Many techniques have been proposed to provide an optimal solution for privacy-preserving queries in LBS, and there are many ways to create solutions for privacy issues. The ultimate objective is to ensure the efficiency and privacy of LBS and their queries. However, this requirement raises questions regarding the security of servers in general and that of LBS in particular. One of these issues is that servers could be compromised and may become malicious. Moreover, servers may produce queries that cannot have forged responses. If for some reason, there is a forged response, the recipient of that response will be able to recognize that it is a fake response. In [19], the authors proposed mobile online social networks (mOSNs), where a location sharing service was presented to the mOSNs. They examined the current problems of location sharing and proposed BMobishare as an enhanced security mechanism that ensures location privacy. Also, they employed the Bloom filter to provide greater security for sensitive data compared to other existing methods.

To overcome the problems mentioned above, the paper proposes the BLOT approach, which combines oblivious transfer (OT) with Bloom filters (BF). Here, the sender is the LBS, while the receiver is the user. The user initiates a transaction by sending a set of q queries to the LBS. These queries cause a benign LBS to generate q responses, i.e., the messages m0 … m (q-1). The protocol proceeds as above. The benign LBS cannot guess which of the q queries were the relevant queries, except with probability 1/q; therefore, the expectation is that BLOT will enhance communication security between the client and the server against attacks.

The contributions can be summarized as follows:

- The proposed method involves the creation of an entirely new type of cryptographic communication protocol in a problem domain and the development of a secure implementation that can be used in resource-constrained systems, such as smartphones.
- The amount of information exposed to attack is minimized. Hence, the BLOT transformation slows down information leakage from the BF from 0.5 bits per query to 0.5/N bits per query, on average, where N is arbitrarily large. Both OT (transformation 2) and BF can be used to reduce information leakage.
- More entropy means less information leakage. Thus, it is more difficult for an attacker to learn anything by studying the encrypted responses.

The remainder of this paper describes the approach to overcoming the challenges described above using OT and BF. A persuasive argument can be made that these diverse protocols can be successfully combined to yield solutions that are stronger than any individual approach.

## II. Related Work

Recently, many solutions have been proposed to protect the users of LBS. Information access control [20] is a technique that provides location privacy to LBS users. Specifically, it equips the LBS provider with a mechanism that enables the LBS users to control access to their location data. Toward this end, LBS providers enforce the access policy's usage to control access to the users' location data. The drawback of this technique is that the LBS providers could be potential adversaries who misuse the location data of the users.

One method is based on a mix zone, i.e., it depends on an intermediate server to hide the user's location [21]. The intermediate server assigns a pseudonym to the user when he/she enters a mix zone. The pseudonym is used by the user to send queries to the LBS server via the intermediate server. A new pseudonym is assigned to the user when his/her mix zone is changed. An important application area of the mix zone technique is road networks. The drawback of this technique is that the intermediate server is vulnerable to adversaries.

The k-anonymity technique [22], [23] is based on the concept that a user cannot be distinguished from other k-1 users. Toward this end, mobile users are grouped in clusters of k members. For each group, a bounding region is defined. Each user uses the bounding region of the group as his/her location and attaches this region for all the queries sent to the LBS provider. An intermediate server is required to construct the bounding region. An adversary can identify the location of the user with a maximum probability of 1/k. The drawback of this technique is that adversaries can compromise the intermediate server.

Using dummy locations is another technique for achieving location privacy [24], [25]. In this technique, the mobile user confuses the LBS server by sending his/her query many times, where one of them contains his/her real location whereas the others contain fake locations. The drawback of this technique is that adversaries can use the side information to analyze the user's sent locations and identify the dummy locations.

In [26], the authors proposed an optimal approach to tackling current location-based query issues regarding the leakage of location data from a query database, generally known as points of interest (POIs). The authors focused on the sharing of location data by the owner with every user. Toward this end, a two-stage approach was proposed; the first step involves oblivious transfer, and the second step involves private information retrieval. The proposed method was implemented on a desktop system and a mobile device with the aim of examining its efficiency.

In [27], the authors proposed the use of oblivious transfer, k -anonymous oblivious transfer, deterministic encryption, and Bloom filters. The k*l-anonymous oblivious transfer protocol was combined with symmetric key encryption (block cipher) to prevent the server from determining which client data are being used. This approach provides anonymity and is beneficial when privacy is preferred over speed.

In [28], the authors showed that oblivious transfer (OT) is essential regarding cryptography, and it is used in many protocols as a security measure for multi-party computation. There is a need for such protocols in the case of a hands-on application, and OT extensions allow the base OTs to be used to calculate a large amount of OTs at a low cost. Consequently, counterparts that use more efficient protocols are created. This model is safe when used in both the random oracle model and the standard model. If the efficacy of the OT extensions is enhanced, they become even safer.

On the other hand, the paper [29], explored bloom filters and their setbacks. Hash functions provided issues for users, even though bloom filters seemed straightforward at first. This paper proposed a method to create independent and orthogonal hash functions while limiting information leaks.

While in [30], presented a survey of the current trends of privacy methods used to protect users that had used LBSs. This paper showed the effectiveness, or lack thereof, of each technique showcased in this paper. It should be recommended that in the future, researchers should conduct a security analysis, using a simulation setup, which would evaluate their proposed algorithm using a modified Hilbert Curve. This would assure the effectiveness of the privacy in the proposed scheme.

The flexibility of sharing location in privacy protection is key to social networking services. Opaque identities developed by Smoke Screen in 2007 [31] help share presence among trusted friends and untrusted strangers. Although the previous work [32] has resolved the problems of flexibility in location sharing in a privacy protecting way, the strictness of the method prevents a direct use.

Recently a Mobishare mechanism [33], proposed by Wei et al. provides a flexible privacy-preserving location sharing to both trusted social relations and untrusted strangers in mOSNs. However, a Mobishare mechanism is actually an extension of Smoke Screen but use a dummy of technique like a real fake identity that prevents complete identities locations between LBS providers and social network providers. A later improvement by Liu et al. broadcast encryption to preserve users' location privacy allowing users to add or removal of friends [34].

Even with such improvements, multiple attacks on a location-based provider can essentially obtain all the friends' relation of user plus the locations in real time. The Paillier Cryptosystem developed by Li et al. could, however, prevent this problem [35]. Li et al. proposed a mechanism that employs a private set intersection protocol to prevent named Mobishare+, that prevent the social network server from copying individual information. While not much is improved in terms of security, this scheme should achieve a lot more complex encryption and decryption operations that sustain computation overhead appreciably. There are still other encryption-based methods particularly in a cloud environment that used for privacy protection [36]-[38].

## III. PRELIMINARIES

### A. Deficiency of BMobishare

A dummy query is used, which allows services to protect their users' real and fake entities.

A Bloom filter serves as a bridge between LBS and social network servers. Although this approach safeguards private data, it is flawed, as there is a risk of leakage between users and servers, which may increase the overhead and decrease the entropy.

### B. Oblivious Transfer Scenario in LBS

Create a data transfer protocol between an LBS client and an LBS server with the following properties:

*1) The protocol has 1-in-q capability.*

*2) The LBS server cannot distinguish information-bearing queries.*

*3) Even if an eavesdropper obtains all queries and responses, no information is leaked.*

*4) A malicious LBS server can be detected with probability q.*

*5) The probability q can be increased exponentially, close to 1, by increasing the queries.*

Oblivious transfer (OT) is a 1-in-q type of algorithm in which the sender transmits a series of messages to the receiver without knowing (except with negligible probability) which, if any, of these messages, contain information used by the receiver, will describe OT for the case of two messages, but it is obviously extendable to the case of q messages. Note that OT is typically built on top of a public key cryptosystem, such as RSA [39], [40], although there is no compelling reason why a system based on symmetric cryptography could not also be used [41].

The steps of the protocol are as follows. The sender has two messages, m0 and m1, and wishes to provide both messages to the receiver without being able to determine which message is actually used by the receiver. The sender generates an RSA key pair. Let M denote the modulus of this key pair; pe, the public exponent; and pd, the private exponent. The sender generates two random nonces, x0 and x1, and sends them to the receiver along with the RSA public key. The receiver generates a random bit b and uses it to set x = xb. The receiver then generates a large random large number L and computes the value V = (x + Lpe) mod M [42]. Then, the value V is transmitted to the sender.

The sender cannot know (except with random probability) the value of b. Therefore, the sender computes two values, v0 = (V – x0)pd mod M and v1 = (V – x1)pd mod N. The sender can deduce that the value of L is either v0 or v1 [43]. The RSA cryptosystem is based on the presumed intractability of the discrete logarithm problem. Thus, the sender cannot know (except with random probability) which of these two values is

correct. The sender computes m0' = m0 + v0 and m1 = m1 + v1, and sends both to the receiver. The receiver knows the value of b and is thus able to determine which of m0' and m1' is valid. Note that no eavesdropper can obtain any information from the value pair (m0', m1') except with negligible probability. In the 1-in-q generalization, the receiver has a secret index, i, and can receive a set of q messages from the sender such that the sender has no way of determining the value of i.

In the OT implementation of an LBS, the sender is the LBS, and the receiver is the user. The user initiates a transaction by sending a set of q queries to the LBS. These queries cause a benign LBS to generate q responses, i.e., the messages m0 … m (q-1). The protocol proceeds as above. The benign LBS cannot guess which of the q queries are the relevant queries, except with probability 1/q. If N exchanges are performed, rather than one, the benign LBS cannot guess which is the relevant query, except with probability 1/qN [44]. Thus, by adjusting q and N, the user can achieve the desired degree of certainty. Note that a malicious LBS server will fail to produce a meaningful response, except with random probability; hence, in the worst case, a malicious LBS server can use the OT protocol to launch a denial-of-service attack. Such an attack can be readily detected by the user with probability (1 - 1/qN), with as much certainty as the user desires.

### C. Bloom Filter Scenario in LBS

A protocol between a client, an LBS server, and a second server must have the following properties:

*1) The client and second server can send a query if there is something in the range of the LBS server.*

*2) The original server can only determine one bit of information from a list of queries, and the second server cannot determine any of these data.*

*3) No set of objects in the range of the server needs to be itemized.*

*4) Any malicious LBS server can be detected with probability q.*

*5) The probability q can be exponentially increased close to 1 by increasing the queries.*

The Bloom filter (BF) allows testing of a fixed membership and never reveals more than one bit of information regarding this set. If A is a reputed element of set S, then the BF creates a probabilistic algorithm to determine whether A is an element of S without going through the elements of S [45].

The LBS includes a Bloom filter that uses location information encoded as bit vectors. The user creates q pairs (query, assertion). The queries will be sent to the LBS, which will create a bit vector of q values. If the i-th position of this vector is 0, then the pair (Qi, Ai) is absolutely inconsistent; if the i-th position of this vector is 1, then the pair (Qi, Ai) is probabilistically consistent [46]. If the server is benign, the evaluations will be accurate. However, if the server is malignant, the evaluations will be inaccurate. Because users can create pairs (Q, A) with a known state, the server will perform tests against known values. A false positive means that the server is malicious. False positives can be reduced with N iterations of q queries. Thus, a malicious server can be found with probability (1 – pN), where p is the legitimate false positive. Malicious servers cannot learn anything from these exchanges, except with a negligible probability due to the inverse Hk.

### D. BLOT Overview in LBS

The ultimate objective of both protocols is to reduce information leakage. Both BF and OT can be used. We achieve this by applying a two-stage approach shown in Fig. 1. The first stage is based on Bloom filter and the second stage is based on an oblivious transfer.

### E. Security for LBS: The BLOT protocol

This paper describes a new approach for LBS applications that provides a user with three security guarantees: (a) it will take at least Q1 queries for an eavesdropper to learn a single bit of information from the protocol exchange; (b) it will take at least Q2 queries for a malicious LBS provider to learn a single bit of information from the protocol exchange; and (c) the information leaked by the database handling the LBS information is at most Q3 bits for each query. In these statements Q1, Q2, and Q3 are adjustable parameters based on the specific implementation of the protocol. If the total number of queries that the user must make is A, and if the protocol can be adjusted such that A < Q1, A < Q2, and (A*Q3) < 1, then the security guarantees become absolute: no eavesdropper or malicious LBS provider can learn any information from the protocol exchange, except with negligible probability.

First, describe the component technologies. BLOT approach is in the form of a framework. The LBS provider will always use one particular storage mechanism: Bloom filters. The protocol will be a pluggable protocol that can take any form as long as it satisfies the security guarantees stated above. In this paper, will consider the OT transfer protocol. Then, will show how it can be integrated to create the BLOT protocol. The construction of the Bloom filter hash functions enables us to demonstrate security guarantee (c). Next, the precise steps in the BLOT exchanges are described in detail, with full mathematical justification, allowing us to demonstrate the first two security guarantees.

In the remainder of this paper, the entity requesting LBS information will be known as the user or the receiver, whereas the entity giving out LBS information will be known as the provider or the sender. The terms client and server will not be used because the BLOT protocol does not require client/server architecture to be deployed. Toward the end of this paper, will explain how it is possible to deploy the BLOT protocol entirely on a mobile device, entirely within the LBS provider, or in some combination of both.

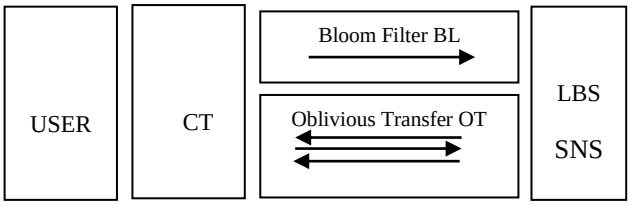


Fig. 1. BLOT overview in LBS.

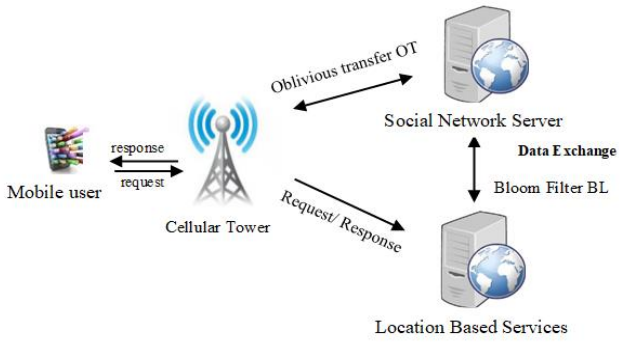


Fig. 2. BLOT architecture in LBS.

### F. BLOT System Architecture

As shown in Fig. 2, the scenario of LBS consists of four entities in the mobile online network.

- The mobile device user can access the Internet with wireless technology (such as 3G/4G) and then share his or location and inquire about the location of friends who are nearby.
- SNS manages the user's identity-related information (user profiles, friend lists, etc.).
- LBS stores the user's anonymized location information and provides LBS, as per the user's request, with the real-time locations of people nearby.
- CT aids the user's communication with SNS and LBS.

BLOT assumes that SNS, LBS, and CT are connected with links that are high-speed and secure.

### G. OT Standard

The purpose of OT is to securely transfer information from a sender to the receiver such that both sides learn the minimum amount of information about the requested transfer. Specifically, suppose that the sender has N pieces of information, (I0 … IN-1), and the receiver has an index n in the range (0,N). The receiver wishes to receive IN without the sender knowing the value of n. The sender also wishes to ensure that the receiver will only be able to decrypt one of the received N encrypted messages. Both sides do not want an eavesdropper to learn any information. OT is a type of minimal knowledge protocol [47].

OT is typically built on top of some form of public key cryptography. The RSA protocol is typically used. The RSA's security is based on the inversion of the discrete logarithm in polynomial time with negligible probability [48]. The algorithm below [49] shows the standard form for OT (note that this exchange shows the protocol for a single sender and a single receiver, as is typical for LBS transactions; however, OT can be extended to multiparty communications).

*1) The Sender generates RSA key pair, with the public modulus MO and the public exponent e.*

*2) The Sender generates N random numbers rI.*

*3) The Sender sends MO, e, and all the ri to the Receiver.*

*4) The Receiver selects the random number rn where n is the index of the desired message.*

*5) The Receiver generates a random number z and computes: v = (rn + ze) mod MO.*

*6) The Receiver sends v, known as the RSA blind value of rn, to the Sender.*

*7) The Sender computes all N values vi = ( v - ri ) d mod MO, where d is the private exponent.*

*8) The Sender computes all I'i = Ii + vi and sends all I'i to the Receiver.*

*9) The Receiver computes the correctly decrypted value In = I'n - z.*

The Sender has a 1-in-N chance of guessing the value n. Thus, from a security standpoint, both parties would like to make N as large as possible. However, from a performance standpoint, the transfer time increases linearly with the value of N. LBS often involve a large number of data transfers. Hence, the performance issue can be partially offset by modification of the messaging protocol; nevertheless, linear performance degradation will occur as the security of the system enhances.

Although we can assume that the receiver (the user) is honest, the same may not be true for the sender (the LBS provider). A dishonest sender could deliberately send weak RSA values, as well as weak random numbers, to improve the odds of guessing the value n. The selected system architecture is summarized in Table I.

The selected notations are summarized in Table II.

In the next section, will describe the BLOT protocol, which significantly improves the security of the system by combining BF with OT.

TABLE I. SYSTEM ARCHITECTURE

| Symbol | Description |
|---|---|
| User (receiver) | Mobile user |
| LBS | Location-based server (backing storage db) |
| SNS (sender) | Social network server |
| CT | Cellular tower |
| ID | An authorized user's unique ID |
| qf | Distance threshold in friends' locations query |
| df | User's friend-case distance |
| qd1 | Friends' locations query 1 |
| qd2 | Friends' locations query 2 |

TABLE II. NOTATIONS

| Symbol | Description |
|---|---|
| $ID_A$ | User A's unique social network identifier |
| $PubMO_A$ | Public modulus |
| Pub $e_A$ | Public exponent |
| $ID_{CT}$ | Cellular tower's identifier |
| PubKey | Public key encryption-decryption |
| Skey (AES) | Symmetric key encryption-decryption |
| ds | User's strange-case distance |
| $BF_f$ | Bloom filter with inserted friend-list |

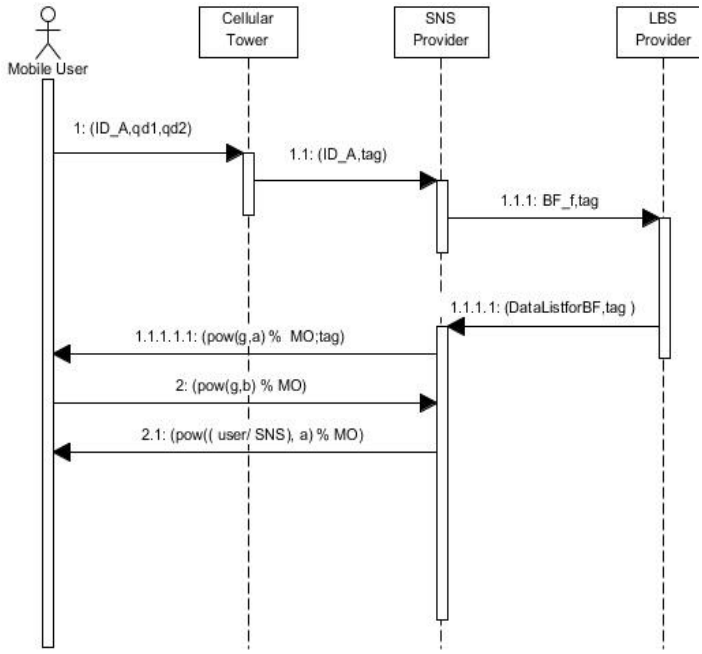


Fig. 3. BLOT Querying friends' locations.

## IV. BLOT MECHANISM

### A. BLOT Querying Friends' Locations

OT builds a security mechanism between users and the SNS, which ensures privacy while avoiding leaks. BF creates a security mechanism between LBS and SNS, and these protect each other's privacy. Fig. 3 shows methods for discovering friends' locations:

Mobile User: First, the user must submit a friend's location query (IDA, qd1, qd2) to CT.

CT: The appropriate entry < IDA, dfA > for IDA will be read in its table. The sequence number and identifier are appended, and the query is forwarded as (IDA, tag) to SNS, where tag = (IDCT, qd1, qd2, seq) and seq is a sequence number.

### B. Bloom Filter Stage BL

*1) First, BF initializes itself and updates itself using the file (DataListforBF.txt).*

*2) This file contains the list of words that will be set in BF for fast searches in the data structure.*

*a) In the filter, Add Element(line) method in the initializing function will operate adding the BF element.*

*b) It uses two hash functions, "FNV" and "MURMUR," to generate the two indices for an element and sets the bit to 1 at the selected indices [50].*

*3) Now, with another file, "DataToQuery.txt.". This file has the list of words that the receiver (USER) is going to query from the SNS.*

*4) Then, the SNS sends the query to the LBS and asks about the two words (QueryData1"qd1" and QueryData2"qd2).*

*5) The LBS takes these 2 words and searches in the BF. It again generates the indices using the same FNV and MURMUR hash functions and checks whether the bit is set to 1.*

*6) If it does not find the data in the BF, then it sends a message to the receiver (USER): "Element (QueryData) is not here."*

Otherwise, if it finds the data in the filter, it confirms the same by searching for the data in the LBS having backing storage. The backing storage, in this case, is the same file DataListforBF.txt. It has been used here as a database.

*7) If both sets of data are found in the backing storage in the LBS, they are forwarded to the SNS, and then the process of OT is initiated for transferring the desired encrypted data to the mobile user.*

### C. OT Stage

*1) For OT, consider the following parameters:*

```
int g = 7; // public exponent
int MO = 101;//public modulus
int N = 2; // number of encrypted packets to send
int C = 0; // indicates that message to be selected by sender SNS (it could be 0 or 1)
```

*2) The Sender (SNS) generates a random number 'a' between 1 and 5.*

*3) Based on the generated random number, SNS calculates the following and sends it to the user:*

```
Double Sender = (long long int) pow(g,a) % MO;
```

*4) After receiving the SNS input, the Receiver (user) performs the following mathematical operations and sends it to the SNS:*

```
int b = rand() % 5 + 5;//generate random number between 5 and 10
if (C == 0) Receiver(user) = (long int) pow(g,b) % MO;
else if(C == 1) Receiver(user) = Sender SNS *(( long int) pow(g,b) % MO);
```

*5) The Sender (SNS) does not know which message the Receiver (user) is going to decrypt. Hence, it will encrypt both pieces of data and send it to the user.*

*a) SNS generates two different keys using the "sha256" key generator.*

*b) Using these keys, it encrypts both data1 and data2 and sends it to the user. For encryption, it uses "AES256-ECB"*

*6) After receiving the data, the Receiver (user) generates a key based on the value of "SNS" received from the user. The key is generated using the same "sha256" key generator.*

*7) The Receiver (USER) then uses this key to decrypt data. The user will only be able to decrypt one of the two datasets correctly.*

Fig. 3 shows how friends' locations are queried for BLOT sequences in LBS.

## V. BLOT Performance Measures

We have already discussed the three performance measures (Q1, Q2, and Q3) that will use as measures for the security of the system. Also introduce a functional performance measure, namely the performance entropy (PE). Both of the protocols have packets that flow between the user and the LBS server as well as in the reverse direction.

Entropy is related to the length (L) of the messages [51], we can divide the contents of each packet into message-carrying information and non-message-carrying information. If the total length of all packets in a single exchange is L, then we can write L = C + N, where C is the sum of the lengths of the message-carrying subset, and N is the sum of the lengths of the non-message-carrying subset. Then, define the performance entropy of the protocol exchange as

$$PE = N/L \quad (1)$$

The larger the value of PE, the higher is the entropy. Thus, for a practical system, want this measure to be as small as possible. For example, if all data was transmitted in the clear, then N = 0 and PE = 0; there is no entropy. Of course, this situation is completely insecure [51], so it is important to minimize this performance measure for the two encrypted protocols described in this paper.

## VI. Performance Entropy Calculation of BMobishare

To calculate the performance entropy of BMobishare, considered the same scenario as that considered in BLOT, i.e., querying a friend's location can be divided into two steps: service registration and authentication, and querying a friend's location. The selected sizes of information (bits) are summarized in Table III.

### A. Service Registration and Authentication

This scenario involves some message exchanges between the user, SNS, and CT, as shown in the message sequence chart (MSC). Based on MSC and Table III, the number of bits exchanged is calculated first, and then PE is calculated. Fig. 4 shows the MSC.

Service registration and authentication involve 6192 bits of information exchanged, which is considered to be non-message-carrying information.

### B. Query a Friend's Location

Querying a friend's location involves 7936 bits of information exchanged between User, CT, SNS, and LBS, as shown in Fig. 5, shows how friends' locations are queried.

The sizes of message-carrying information (C) and non-message-carrying information (N) are listed in Table IV.

TABLE III. Size of Information (bits)

| S. No. | Information | Size of information (bits) |
|---|---|---|
| 1 | PubKey | 2048 |
| 2 | Skey (AES) | 128 |
| 3 | $df_A$ | 64 |
| 4 | $ds_A$ | 64 |
| 5 | $ID_A$ | 64 |
| 6 | Time stamp (ts) | 64 |

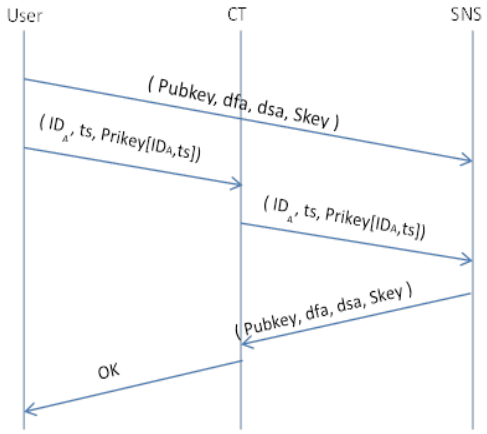


Fig. 4. Message Sequence Chart (MSC).

$$L = C + N = 14128\ bits \quad (2)$$

$$P_1 = \frac{C}{L} = \frac{1600}{14128} = 0.11325 \quad (3)$$

$$P_2 = \frac{N}{L} = \frac{12528}{14128} = 0.886 \quad (4)$$

By using (13), (shown in the Appendix at the end of the paper), we can calculate entropy as [51]:

$$H_{BMobishare} = (0.11325) log_2 \left(\frac{1}{0.11325}\right) + (0.886) log_2 \left(\frac{1}{0.886}\right) = 0.5105 \quad (5)$$

While,

$H_{BLOT}(Avg)$ has been calculated from simulation founds to be = 0.9756 (6)

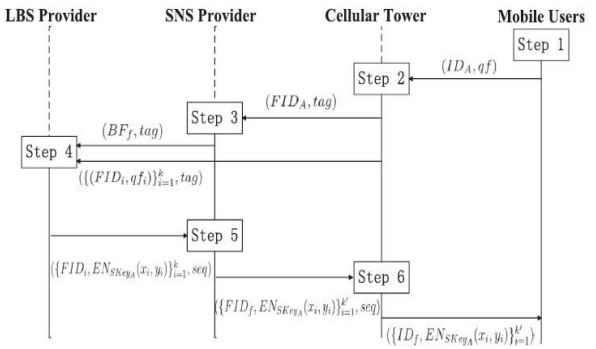


Fig. 5. Querying friends' locations.

TABLE IV. The Total Size Of Non-/Message-Carrying Information

| S. No. | Function | C | N |
|---|---|---|---|
| 1 | Service registration and authentication | 0 | 6192 |
| 2 | Query friends' locations | 1600 | 6336 |
| | Total | 1600 | 12528 |

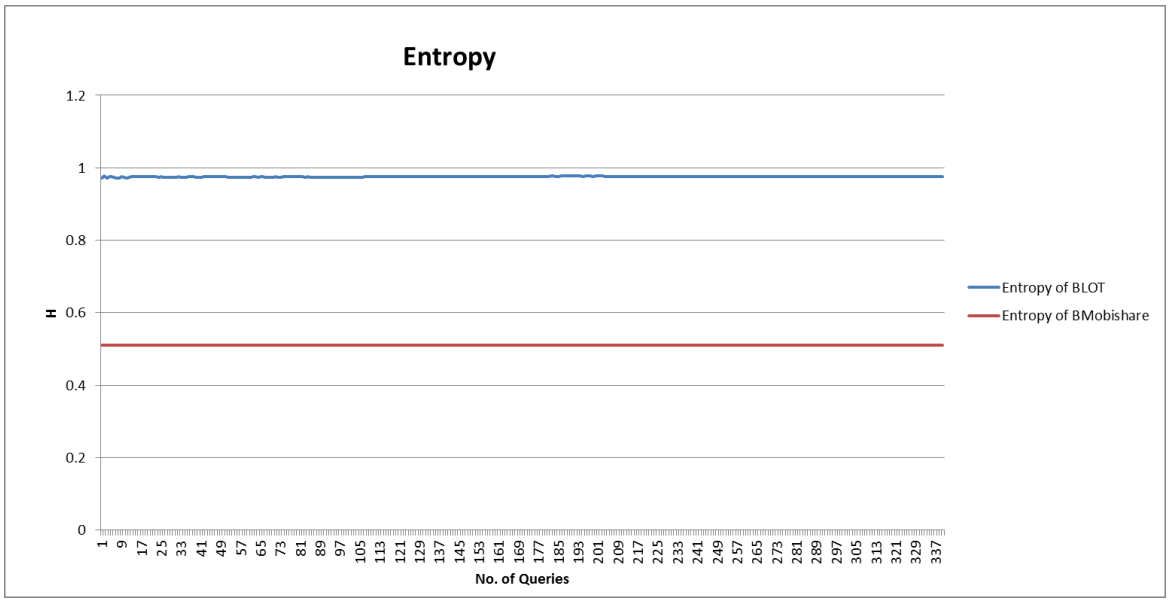


Fig. 6. Performance entropy comparison.

## VII. Analysis of Results and Discussion

The total number of information exchanges in BLOT is around 960 bits as per simulation, which is much smaller than that of BMobishare. Fig. 6 shows the performance entropy comparison. The average performance entropy of BLOT is 0.9756, which is greater than that of BMobishare 0.5105, because BMobishare uses a large number of anonymous messages for increasing security, whereas BLOT ensures security by using an encryption algorithm (AES256).

Based on the result, we can realize a significant entropy value, which means less information leakage. Further, can confirm the effectiveness of the BLOT mechanism as well as low overhead and high performance owing to the avoidance of massive numbers of fake IDs, in contrast to BMobishare.

## VIII. Performance Results

**Note**: In the given paper on BMobishare, the author did not mention what he means by "connection performance" or "computing performance". The conclusion below is based on the assumption that "connection performance" and "computing performance" have the following definitions:

### A. Connection Performance

The time a sender takes to establish a connection to the receiver to start further secure communication using any protocol. Connection performance of BLOT would be better than that of BMobishare because BMobishare establishes the connection before querying the friend's location using the service registration and authentication mechanism. In BLOT there is no need for service registration and authentication. Hence connection time is 0s. In BMobishare, average connection time is 1.5s.

Therefore, we can conclude that the connection performance of BLOT is better while computing performances are comparable.

### B. Computing Performance

The time a computer takes to execute a code/algorithm/protocol. Computing performance can be calculated based on the number of packets communicated in each protocol because a higher number of packets involved means more computation required to process them.

*a) In BLOT, a total of 7 messages need to be communicated to query a friend's location.*

*b) In BMobishare, 7 messages are communicated to query a friend's location.*

### C. Computing Performance Measurement

In BLOT average computation time measured from the simulation is 10 ms. while the BMobishare shows that the average computing time is 1.75s.

But the author of BMobishare measured the time with a processor of 1.2 GHz and 1GB RAM.

While in BLOT have measured time over 2.5 GHz and 4GB RAM. As the protocol code is not very large, RAM cannot be a factor in execution time. 1GB RAM is also enough to accommodate the code data as well. Here we can see that the BLOT processor is twice the speed of the BMobishare processor. Therefore to make the comparison fair, we can assume that if run BLOT on a 1.2 GHz machine, it will double the execution time. (That is, it would multiply the time by a factor of 2. 10ms x2 = 20ms can give us the time if the code runs on a 1.2 GHz machine). The following table shows the performances result in Table V.

TABLE V. Performance Results In Summary

| S. No. | Protocol | Computation time (ms) | Connection time (s) |
|---|---|---|---|
| 1 | BLOT | 20 | 0 |
| 2 | BMobishare | 1.75 | 1.5 |

## IX. CONCLUSION

In this paper, is described the BLOT mechanism, which combines two protocols that attempt to address the security deficiencies commonly found in current LBS application environments. In particular, BLOT addresses the security of backing storage information, information that is visible to an eavesdropper, or information that is vulnerable to a malicious presence on the LBS provider machine. Also, it enables a user to perform consistency checks on the LBS provider.

Each component individually provides high security, while the deployment of both components significantly enhances the security using either transfer approach. Analyzed the performance entropy of BLOT and found it greater than the solution by BMobishare; additionally, the performance was more efficient.

## X. APPENDIX

### A. Entropy Derivation

By the definition of Shannon's entropy [51]:

$$H(x) = -\sum_{i=1}^{n} P_i(x)\ log_2(P_i(x)) \quad (7)$$

$$H(x) = \sum_{i=1}^{n} P_i(x)\ [log_2(1) - log_2(P_i(x)] \quad (8)$$

$$H(x) = \sum_{i=1}^{n} P_i(x)\ log_2(\frac{1}{P_i(x)}) \quad (9)$$

In BLOT, case n=2

$$i = 1,\ for\ message\ acrrying\ information \quad (10)$$

$$i = 2,\ for\ non-\ message\ acrrying\ information \quad (11)$$

Therefore the above equation becomes:

$$H(x) = \sum_{i=1}^{2} P_i(x)\ log_2(\frac{1}{P_i(x)}) \quad (12)$$

$$H(x) = P_1(x)\ log_2\left(\frac{1}{P_1(x)}\right) + P_2(x)\ log_2\left(\frac{1}{P_2(x)}\right) \quad (13)$$

$$P_1 = probability\ \ message\ acrrying\ information = \frac{C}{L} \quad (14)$$

$$P_2 = probability\ nonmessage\ acrrying\ information = \frac{N}{L} \quad (15)$$

Equation (13) has been used to calculate the entropy.

## REFERENCES

[1] B. Niu, Z. Zhang, X. Li, and H. Li, "Privacy-area aware dummy generation algorithms for location-based services," in Proc. Of IEEE ICC 2014.

[2] J. Krumm, "A survey of computational location privacy," Pers. Ubiquit. Comput., vol. 13, no. 6, pp. 391–399, Aug. 2009.

[3] Through spatial and temporal cloaking," in Proc. Of ACM MobiSys 2003.

[4] A. Albelaihy, J. Cazalas, "A Fine-Grained Spatial Cloaking With Query Probability Levels for Privacy in LBS," International Journal of Computer Networks and Applications (IJCNA), vol. 2, no. 5, pp. 212-221, 2015.

[5] C. Cachin, C. Crepeau, J. Marcil and G. Savvides, "Information-Theoretic Interactive Hashing and Oblivious Transfer to a Storage-Bounded Receiver," IEEE Trans. Inform. Theory, vol. 61, no. 10, pp. 5623-5635, 2015.

[6] N. Aggarwal, C. Gupta, and I. Sharma, "Fully Homomorphic symmetric scheme without bootstrapping," in Cloud Computing and Internet of Things (CCIOT), 2014 International Conference on, 2014, pp. 14-17.

[7] J. Alwen, M. Barbosa, P. Farshim, R. Gennaro, S. D. Gordon, S. Tessaro, et al., "On the relationship between functional encryption, obfuscation, and fully homomorphic encryption," in Cryptography and Coding, ed: Springer, 2013, pp. 65-84.

[8] V. Garg and M. Jhamb, "A Review of Wireless Sensor Network on Localization Techniques," International Journal of Engineering Trends and Technology (IJETT)-Volume4Isssue4-April, 2013.

[9] C. Gentry, J. Groth, Y. Ishai, C. Peikert, A. Sahai, and A. Smith, "Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs," Journal of Cryptology, pp. 1-24, 2014.

[10] N. Smart, "Investigations of Fully Homomorphic Encryption (IFHE)," DTIC Document2015.

[11] Z. Zhang, T. Plantard, and W. Susilo, "Reaction attack on outsourced computing with fully homomorphic encryption schemes," in Information Security and Cryptology-ICISC 2011, ed: Springer, 2012, pp. 419-436.

[12] A. Leiva, N. Pavez, A. Beghelli, and R. Olivares, "A joint RSA algorithm for dynamic flexible optical networking," in Communications (LATINCOM), 2014 IEEE Latin-America Conference on, 2014, pp. 1-6.

[13] G. D. Sutter, J.-P. Deschamps, and J. L. Imaña, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," Industrial Electronics, IEEE Transactions on, vol. 58, pp. 3101-3109, 2011.

[14] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in Advances in Cryptology–EUROCRYPT 2012, ed: Springer, 2012, pp. 483-501.

[15] Z. Brakerski and V. Vaikuntanathan, "Lattice-based FHE as secure as PKE," in Proceedings of the 5th conference on Innovations in theoretical computer science, 2014, pp. 1-12.

[16] S. Devadas, M. van Dijk, C. W. Fletcher, and L. Ren, "Onion ORAM: A Constant Bandwidth and Constant Client Storage ORAM (without FHE or SWHE)," 2015.

[17] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?," in Proceedings of the 3rd ACM workshop on Cloud computing security workshop, 2011, pp. 113-124.

[18] S. Mudda and S. Giordano, "Mobile P2P queries over temporal data," in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, 2014, pp. 278-283.

[19] N. Shen, J. Yang, K. Yuan, C. Fu and C. Jia, "An efficient and privacy preserving location sharing mechanism", Computer Standards & Interfaces, vol. 44, pp. 102-109, 2015.

[20] M. Youssef, V. Atluri, and N. R. Adam, Preserving mobile customer privacy: An access control system for moving objects and custom proles. s.l. : In Proc. MDM 2005.

[21] Mobimix, B. Palanisamy and L. Liu, Protecting location privacy with mix-zones over road networks. s.l. : In Proc. ICDE 2011.

[22] B. Bamba, L. Liu, P. Pesti, and T. Wang, Supporting anonymous location queries in mobile environments with PrivacyGrid. s.l. : In Proc. WWW 2008.

[23] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," ACM Trans. Database Syst., vol. 34, no. 4, 2009.

[24] P. Shankar, V. Ganapathy and L. Iftode, Privately querying location-based services with SybilQuery. s.l. : In Proc. Ubicomp 2009.

[25] O. Han, H. Zhao, Z. Ma, K. Zhang, and H. Pan, Protecting Location Privacy Based on Historical Users over Road Networks .. s.l. : In Proc. WASA 2014.

[26] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," Knowledge and Data Engineering, IEEE Transactions on, vol. 26, pp. 1200-1210, 2014.

[27] V. Gupta, T. S. Vineeth, and V. Aggarwal, "Make Your Query Anonymous With Oblivious Transfer," in Proceedings of the Sixth International Conference on Computer and Communication Technology 2015, 2015, pp. 345-349.

[28] G. Asharov, Y. Lindell, T. Schneider, M. Zohner: More Efficient Oblivious Transfer Extensions with Security for Malicious Adversaries. EUROCRYPT (1) 2015: 673-701.

[29] A. Albelaihy and J. Cazalas, "Privacy preserving queries for LBS: Hash function secured (HFS)," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, 2017, pp. 7-12.

[30] A. Albelaihy and J. Cazalas, "A survey of the current trends of privacy techniques employed in protecting the Location privacy of users in LBSs," 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, 2017, pp. 19-24.

[31] L.P. Cox, A. Dalton, V. Marupadi, Smokescreen: flexible privacy controls for presence-sharing, Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, ACM 2007, pp. 233–245.

[32] K.P. Puttaswamy, B.Y. Zhao, Preserving privacy in location-basedmobile social applications, Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, ACM 2010, pp. 1–6.

[33] W. Wei, F. Xu, Q. Li, Mobishare: flexible privacy-preserving location sharing in mobile online social networks, INFOCOM, 2012 Proceedings IEEE, IEEE 2012, pp. 2616–2620.

[34] Z. Liu, J. Li, X. Chen, J. Li, C. Jia, New privacy-preserving location sharing system for mobile online social networks, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on, IEEE 2013, pp. 214–218.

[35] J. Li, J. Li, X. Chen, Z. Liu, C. Jia, Mobishare+: security improved system for location sharing in mobile online social networks, J. Internet Serv. Inf. Secur. (JISIS) 4 (2014) 25–36.

[36] Z. Liu, J. Li, X. Chen, J. Yang, C. Jia, Tmds: thin-model data sharing scheme supporting keyword search in cloud storage, Information Security and Privacy, Springer 2014, pp. 115–130.

[37] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, D.S. Wong, L-encdb: a lightweight framework for privacy-preserving data queries in cloud computing, Knowl.-Based Syst. 79 (2015) 18–26.

[38] Z. Liu, X. Chen, J. Yang, C. Jia, I. You, New order preserving encryption model for outsourced databases in cloud environments, J. Netw. Comput. Appl. (2014),

[39] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: a new vision for public-key cryptography," Communications of the ACM, vol. 55, pp. 56-64, 2012.

[40] A. J. Menezes, Elliptic curve public key cryptosystems vol. 234: Springer Science & Business Media, 2012.

[41] K. B. Swamy and K. K. Raju, "Multi-keyword Ranked Search over Encrypted Cloud Data Using RSA Algorithm," IJSEAT, vol. 3, pp. 307-311, 2015.

[42] S. Bai, R. Brent, and E. Thomé, "Root optimization of polynomials in the number field sieve," Mathematics of Computation, 2015.

[43] V. Grolmusz, "Separating the communication complexities of MOD m and MOD p circuits," in Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on, 1992, pp. 278-287.

[44] O. Chowdhury, D. Garg, L. Jia, and A. Datta, "Equivalence-based Security for Querying Encrypted Databases: Theory and Application to Privacy Policy Audits," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 1130-1143.

[45] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet mathematics, vol. 1, pp. 485-509, 2004.

[46] W. Zhang, S. Liu, and Y. Xiaoyuan, "RLWE-Based Homomorphic Encryption and Private Information Retrieval," in Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on, 2013, pp. 535-540.

[47] R. Ahlswede, "Founding Cryptography on Oblivious Transfer," in Hiding Data-Selected Topics, ed: Springer, 2016, pp. 337-344.

[48] D. R. Stinson, Cryptography: theory and practice: CRC press, 2005.

[49] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[50] "Oblivious transfer", Asecuritysite.com, 2017. [Online]. Available: https://asecuritysite.com/encryption/ot. [Accessed: 26- Jul- 2017].

[51] M. Eigen, From strange simplicity to complex familiarity. Oxford: Oxford Univ. Press, 2013.