MobiShare: Flexible Privacy-Preserving Location Sharing in Mobile Online Social Networks

Wei Wei, Fengyuan Xu, Qun Li
Computer Science, The College of William and Mary
{wwei, fxu, liqun}@cs.wm.edu

***Abstract*—Location sharing is a fundamental component of mobile online social networks (mOSNs), which also raises significant privacy concerns. The mOSNs collect a large amount of location information over time, and the users' location privacy is compromised if their location information is abused by adversaries controlling the mOSNs. In this paper, we present MobiShare, a system that provides flexible privacy-preserving location sharing in mOSNs. MobiShare is flexible to support a variety of location-based applications, in that it enables location sharing between both trusted social relations and untrusted strangers, and it supports range query and user-defined access control. In MobiShare, neither the social network server nor the location server has a complete knowledge of the users' identities and locations. The users' location privacy is protected even if either of the entities colludes with malicious users.**

I. Introduction

In the past few years, online social networks (OSNs) have gained great popularity and are among the most frequently visited sites on the Web. Besides, the popularity of mobile devices such as cell phones and tablets keeps being exploding, and these mobile devices are becoming smarter. Most cell phones sold today are capable of accessing the Internet over WiFi or cellular networks, and determining their locations through GPS or cellular geolocation. As a result, it is not surprising to see the rapid fusion of OSNs with mobile computing, that is, a new paradigm called mobile online social networks (mOSNs).

The mOSNs can be classified into two types. The first type consists of the existing OSNs, like Facebook and Twitter, turning mobile. That is, they tail the contents and access mechanisms for mobile users, and allow accesses from mobile devices. The second type mOSNs are the newly emerging OSNs that are dedicated to mobile users, such as Foursquare and Gowalla. These new mOSNs are designed to explicitly take advantage of the location information provided by the mobile devices. Compared with traditional OSNs, both of these two kinds of mOSNs take a step further in that they provide the location-based services. Instead of explicitly inputting their locations, recent smartphone platforms that support various localization technologies make it much easier for the users to access and share their locations with each other.

While the location-based features make mOSNs more popular, they also raise significant privacy concerns. Users' locations may reveal sensitive private information, such as interests, habits, and health conditions, especially when they are in the hands of the adversaries. The threat is even more serious when it comes to mOSNs, because users' physical locations are now being correlated with their profiles. Considering that all the current mOSNs are under centralized control, users' location privacy will be compromised if the location data collected by the mOSNs are abused, inadvertently leaked, or under the control of hackers. Without a guarantee of privacy, users may be hesitant to share locations through mOSNs [3].

Given the popularity of mOSNs and the sensitivity of the location data users place at them, it is critical to limit the privacy risks posed by today's mOSNs while retaining their functions. As indicated in previous research [7], location and presence are two sources of privacy leakage introduced by mOSNs. SmokeScreen [4] solves the problem of how to flexibly share presence with both friends and strangers while preserving user privacy. However, until now no scheme has been proposed to address the same problem for location sharing in mOSNs. Previous work [8], [12] discussed sharing locations between established relations in a privacy-preserving way. However, restricting location sharing to established social relations makes a large class of mobile social applications, such as Serendipity [5], impossible. Besides, E-SmallTalker [11] and E-Shadow [9] preserve location privacy by limiting social information sharing in physical proximity via local wireless communications. In a mOSN, users may want to see the locations of both friends and strangers within some ranges, while at the same time they should be able to control how their own location information is accessed by others. To protect users' location privacy, the system should work in a way that an adversary controlling the mOSN cannot obtain users' location information. Unfortunately, no scheme proposed so far meets these requirements.

In this paper, we present a privacy management system called MobiShare, which provides flexible privacy-preserving location sharing in mOSNs. Our system is *flexible* in that it supports the features of location sharing in real-world mOSNs, including sharing locations between both trusted social relations and untrusted strangers, querying locations within a certain range, and user-defined access control. MobiShare leverages the existing OSNs and requires no change to their architectures, but these OSNs are not trusted to access users' location information. In MobiShare, the social network server stores users' identity-related information, while an untrusted third-party location server stores users' anonymized location updates mixed with dummy location updates. The adversary cannot link a precise location to an identified user, as long as

he cannot control both entities. A user's location information is not leaked to the malicious users who are unauthorized to access his locations either, even if these malicious users collude with the social network server or the location server.

II. System Architecture and Threat Model

A. System Model

To protect users' location privacy, MobiShare stores users' identity-related information and anonymized location updates at two separate entities, the social network server and the location server. Figure 1 shows the system architecture of MobiShare. The social network server can be a server of any existing OSN that wants to provide the location-sharing service. It manages users' identity-related information, e.g., their profiles and friend lists. The location server is an untrusted third-party server that stores the anonymized location updates of the users. For example, A company may implement the location server so as to profit from the OSNs or the users. Besides, some privacy advocacy organization, like the Electronic Frontier Foundation (EFF), can provide the location server to help protect user privacy. Given that all the current smartphones are able to access the Internet with wireless techniques like 3G, it is reasonable to assume that users can communicate with the servers through cellular networks.

We assume that each user has a unique identifier at the social network server. This identifier is used as his identity in MobiShare. Each user generates by himself a public-private key pair and a symmetric session key, and shares the session key with all his social network friends. Each cellular tower has a unique identifier and generates by itself a symmetric secret key, and shares them with the location server. The location server also shares a symmetric secret key with the cellular towers. The servers and the cellular towers are connected by high-speed secure links, and the social network server cannot identify the communicating cellular towers by observing the IP addresses in the connections. For example, this can be achieved by using proxies provided by the cellular carriers.

B. Trust and Threat Model

The social network server and the location server are not trusted to access users' location information. We assume that either the social network server or the location server can be compromised and controlled by an adversary seeking to link users' identities to their locations, but the adversary cannot control both entities. This model is rational in that many security breach cases usually involve the hack of databases or logs in a single system, or dishonest insiders within a system trying to fetch sensitive information [1]. It is unlikely that the two servers operated by independent organizations can be controlled by the same adversary. Besides, some users may also be malicious, who seek to obtain the location information they are unauthorized to access. The social network server or the location server may collude with these malicious users. For example, an employee of the social network company may register for the location-sharing service, and collude with the server to extract other users' location information.

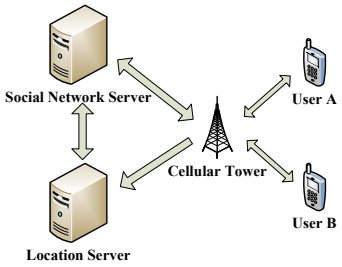

Fig. 1. System architecture

This work does not investigate how to improve location privacy within the cellular networks. The wireless Enhanced 9-1-1 rules [2] of the Federal Communications Commission (FCC) require that the cellular carriers can locate the subscribed cell phones with an accuracy of 50 to 300 meters. Also, for each subscribed cell phone the cellular carrier generally knows the owner's name and address. Therefore, we make no attempt to conceal the devices' locations from the cellular networks, i.e., *the cellular towers are trusted*.

III. System Design

We separate the problem of privacy-preserving location sharing into two cases, sharing locations between friends and between strangers, and solve them separately. A summary of the notations used in this section is given in Table I.

A. Service Registration

Before using the location-sharing service, each user needs to register for the service at the social network server. During registration, user A shares his public key $PubKey_A$ with the social network server, and defines his access control settings, which consist of two threshold distances, df_A and ds_A. df_A is the threshold distance within which A is willing to share his location with his social network friends. If the distances between A and some of his friends are larger than df_A, they cannot access A's current location. Similarly, ds_A is the threshold distance within which A agrees to share his location with arbitrary users. After registration, A keeps a record of his social network identifier ID_A, while the social network server stores an entry as $\langle ID_A, PubKey_A, df_A, ds_A \rangle$ in its *subscriber* table, where the user identity is the primary key.

B. Authentication

After user A's handset connects to cellular tower C, an encrypted data transmission link is established based on mobile telecommunication techniques such as 3G/4G. To let the cellular tower authenticate his identity, A sends an authentication request as $(ID_A, ts, Sig_A(ID_A, ts))$ to the cellular tower, where ID_A is A's social network identifier, and ts is a timestamp used to prevent replay attack. The message is signed by A's private key. The cellular tower forwards this message to the social network server. Upon receiving the message, the social network server searches its subscriber table for A's registration information, including A's public key $PubKey_A$ and the threshold distances df_A and ds_A.

The social network server first uses $PubKey_A$ to verify A's signature. If the verification succeeds, it sends a reply as (ID_A, df_A, ds_A) to the cellular tower. The cellular tower forwards this message to A. A checks if ID_A, df_A, and ds_A are correct. If so, A sends an OK message to the cellular tower. On the reception of the OK message, the cellular tower stores an entry as $\langle ID_A, df_A, ds_A \rangle$ in its *user info* table, where the user identity is the primary key. After this an authenticated and secure communication link is established between A and the cellular tower, and A's identity is attached to this link.

C. Location Updates

When the users upload their location updates, the task of the cellular towers is to perform anonymization such that the user identities cannot be inferred from the anonymized location updates. This is achieved by leveraging both pseudonyms and dummy location updates. We assume that each cellular tower periodically generates fake IDs, and saves them in a fake ID pool. The number of needed fake IDs depends on the number of users connecting to this cellular tower and their location update frequency. Each location update from a user consumes k fake IDs. The fake IDs can be efficiently generated using a cryptographic hash function, such as SHA-1, and a random salt value as follows: $fake\ ID_i = SHA(fake\ ID_{i-1} \oplus salt)$.

Assuming user A periodically gets his current location through techniques such as GPS or cellular geolocation, to update his location, A sends a message to the cellular tower as $(ID_A, (x, y), Sess_A(x, y))$, where (x, y) is A's current location, and $Sess_A(x, y)$ is the location encrypted with A's session key. This session key is shared with all his social network friends. Upon receiving the location update from A, the cellular tower performs coarse location verification by checking if (x, y) is within its working range. If so, the cellular tower keeps a record of A's current location in its user info table. Then the cellular tower picks k fake IDs from its fake ID pool. One of the k fake IDs is used to replace A's identity in the real location update. Let this fake ID be FID_A. The other $k-1$ fake IDs are used by the cellular tower to construct dummy location updates. The cellular tower stores FID_A at A's entry in the user info table, and sends the mapping between A's identity and the k fake IDs to the social network server, which stores an entry as $\langle ID_A, FID_A, FID_1, ..., FID_{k-1} \rangle$ in its *fake ID* table, where the user identity is the primary key. With this table the social network server knows, given any user identity, the fake ID used in the latest real location update and the fake IDs used in the latest dummy location updates.

To anonymize the location update from A, the cellular tower will send k location updates to the location server. Only one location update contains A's real location, while the other $k - 1$ are dummies. The real location update is of the form as $(FID_A, (x, y), Sess_A(x, y), df_A, ds_A)$. It contains A's fake ID, plaintext and encrypted locations, and the threshold distances. To construct the dummy location updates, the cellular tower follows the method proposed by Kido et al. [6] and generates $k - 1$ dummy locations within its coverage. The i^{th} dummy location update is of the form as $(FID_i, (x_i, y_i), str_i, df_i, ds_i)$. FID_i is one of the k fake IDs from the fake ID pool; (x_i, y_i) is one of the dummy locations generated by the cellular tower; str_i is a random string imitating the encrypted location; df_i and ds_i are the threshold distances of a random user whose information is stored in the cellular tower's user info table. Note that for the dummy location updates, the cellular tower does not need to really encrypt the locations. Instead, it only needs to generate arbitrary strings with the length of an encrypted location. Like generating fake IDs, these strings can be created efficiently using a hash function and a salt value.

The cellular tower sends the k location updates to the location server in a random order with random time intervals following the exponential distribution. The location server stores them in its *location update* database. The database consists of a number of tables. Each table represents a geographic region. The updates of locations within a region will be stored in the corresponding table, where the fake ID is the primary key. Organizing the location update database in this way improves search efficiency and reduces computation overhead. For instance, given one location, to find all the fake IDs within a range, instead of checking all the stored location updates, the location server only needs to search the tables of the regions that overlap the queried circular area. Note that the entries in the location update database expire after a certain period of time (15 minutes in our implementation).

TABLE I
SUMMARY OF NOTATIONS

ID_A	User A's social network identifier, used as his identity
FID_A	User A's fake ID
$PubKey_A$	User A's public key
$PrivKey_A$	User A's private key
$SessKey_A$	User A's session key, shared with all his friends
df_A	User A's friend-case threshold distance
ds_A	User A's stranger-case threshold distance
CID_C	Cellular tower C's identifier
$SecKey_C$	Cellular tower C's secret key, shared with the location server
$SecKey_{Loc}$	Location server's secret key, shared with the cellular towers

D. Querying Friends' Locations

Figure 2 shows the messages involved in querying friends' locations. To query the locations of his friends within a certain range, say 1 mile, user A sends *query*$(ID_A$, *'f'*, *'1mi'*$)$ to the cellular tower. The cellular tower appends its identifier and a sequence number, which are encrypted by the location server's secret key, to this message, and forwards *query*$(ID_A$, *'f'*, *'1mi'*, $SecKey_{Loc}(CID_C, seq))$ to the social network server. The identifier will be used by the location server to find the secret key shared by this cellular tower to encrypt the reply.

Upon receiving the query, the social network server looks up the currently used fake IDs of A and all A's friends in its fake ID table. Let $FIDlist$ be a list consisting of the fake IDs of all A's friends in random order, including the fake IDs used in each friend's latest real location update and the fake IDs used in each friend's latest dummy location updates. Assume A has f friends, then the size of $FIDlist$ is kf. The social

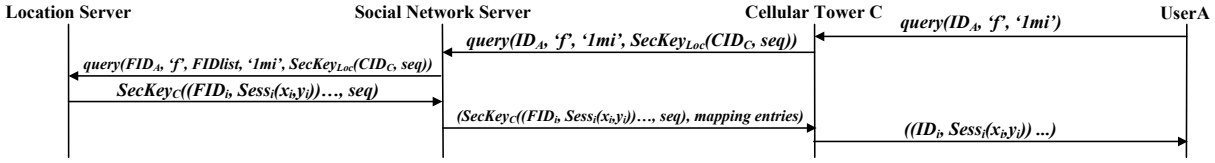

Fig. 2. Querying friends' locations

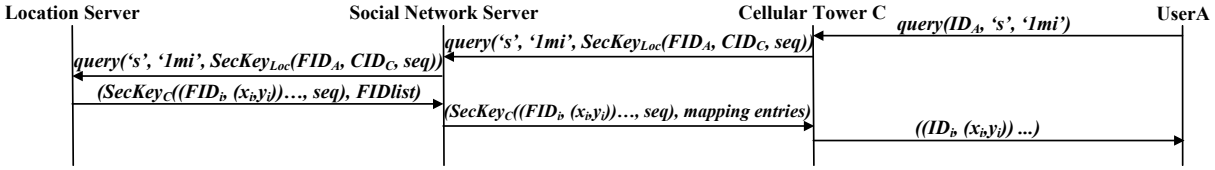

Fig. 3. Querying strangers' locations

network server replaces A's identity with FID_A, which is the fake ID used in A's latest real location update, and sends *query*$(FID_A$*, 'f', FIDlist, '1mi',* $SecKey_{Loc}(CID_C, seq))$ to the location server. On the reception of the query, the location server checks which fake IDs in $FIDlist$ are within 1 mile away from FID_A. For each of these nearby fake IDs, the location server enforces access control based on that fake ID's friend-case threshold distance stored in the location update database. For example, if one fake ID is 0.7 miles away from FID_A, but its friend-case threshold distance is 0.5 miles, the location server will not send the location of this fake ID to A, even though it is within the queried range.

After finishing the distance computation and the access control enforcement, the location server sends a reply to the social network server as $SecKey_C((FID_i, Sess_i(x_i, y_i))..., seq)$. To prevent the social network server from prying the contents, this message is encrypted with the cellular tower's shared secret key. The reply may contain multiple location entries. Each entry is of the form as $(FID_i, Sess_i(x_i, y_i))$, where FID_i is a fake ID in $FIDlist$ which is within the queried range and has passed the access control enforcement. $Sess_i(x_i, y_i)$ is FID_i's encrypted location stored in the location update database. As mentioned in the previous subsection, if FID_i is a fake ID used in the latest real location update of a user, then $Sess_i(x_i, y_i)$ is FID_i's location encrypted with this user's session key shared with all his friends, including A. Otherwise, if FID_i is a fake ID used in one of the dummy location updates, then $Sess_i(x_i, y_i)$ is an arbitrary string with the length of an encrypted location.

Upon receiving the reply, the social network server appends a mapping entry for each of A's friends to the message, and forwards it to the cellular tower. Each mapping entry is of the form as (FID_j, ID_j), where FID_j is the fake ID used in friend j's latest real location update, and ID_j is friend j's identity. Note that the social network server does not provide the mapping entries for the fake IDs used in the dummy location updates. Assume A has f friends, then f mapping entries are appended to the reply. The reply received by the cellular tower contains the encrypted location entries and the mapping entries. The cellular tower first uses its secret key shared with the location server to decrypt the location entries. Then for each location entry that has a matching mapping entry with the same fake ID, it replaces the fake ID in the location entry with the user identity. The location entries that do not have a matching mapping entry, which are from the dummy location updates, are all discarded by the cellular tower. Until now each remaining location entry has the form as $(ID_i, Sess_i(x_i, y_i))$, which includes both the user identity and the encrypted location. The cellular tower sends these entries to A. Since we assume that all of A's friends have shared their session keys with A, A can decrypt and get the plaintext locations of the nearby friends.

E. Querying Strangers' Locations

Figure 3 shows the messages involved in querying strangers' locations. To query the locations of arbitrary users within a certain range, say 1 mile, A sends *query*$(ID_A$*, 's', '1mi'*$)$ to the cellular tower. The cellular tower keeps a record of the queried range at A's entry in the user info table. Then it removes ID_A, and appends FID_A, which is the fake ID used in A's latest real location update, the cellular tower identifier, and a sequence number to the message, all of which are encrypted by the location server's secret key. The cellular tower sends *query*$('s'$*, '1mi',* $SecKey_{Loc}(FID_A, CID_C, seq))$ to the social network server, which directly forwards the query to the location server.

On the reception of the query, the location server looks up the fake IDs that are within 1 mile away from FID_A. For each of these fake IDs, the location server enforces access control based on its stranger-case threshold distance. Assuming there are n nearby fake IDs that pass the access control enforcement, the location server randomly picks another recently received $(k-1)n$ fake IDs from the location update database. These fake IDs are mixed with the n nearby fake IDs in the reply to achieve k-anonymity. The reply sent from the location server to the social network server is of the form as $(SecKey_C((FID_i, (x_i, y_i))..., seq), FIDlist)$. This message includes n location entries, each of which contains a nearby fake ID and its plaintext location. All these location entries are encrypted with the cellular tower's secret key. $FIDlist$

consists of the n nearby fake IDs mixed with the $(k-1)n$ randomly selected fake IDs. On the reception of the reply, the social network server cannot pry the contents of the encrypted location entries, nor can it learn which fake IDs in $FIDlist$ are currently close to A, since it cannot distinguish the n nearby fake IDs from the $(k-1)n$ padded fake IDs.

As mentioned in Section III-C, to anonymize each location update from a user, the cellular tower generates $k-1$ dummy location updates and sends all the k updates to the location server. Therefore, approximately $(k-1)/k$ of the kn fake IDs in $FIDlist$ come from dummy location updates. The social network server can simply filter out all these fake IDs based on its fake ID table. For each of the remaining fake IDs, the social network server appends to the reply a mapping entry as (FID_j, ID_j, ds_j), where ID_j is user j's identity, and ds_j is user j's stranger-case threshold distance. Then it sends $(SecKey_C((FID_i, (x_i, y_i))..., seq), mapping\ entries)$ to the cellular tower.

After the cellular tower receives the reply, it first uses its secret key to decrypt the location entries. To defend against the attack that the location server colludes with a malicious user, the cellular tower randomly selects one location entry that has a matching mapping entry in the reply, and checks if the distance between the location in this entry and A's current location stored in the user info table is smaller than both the queried range and the stranger-case threshold distance in the mapping entry. If the check fails, the reply is discarded and the location server is suspected of behaving maliciously. Otherwise, for each location entry that has a matching mapping entry, the cellular tower replaces the fake ID with the user identity and gets the location entry as $(ID_i, (x_i, y_i))$. The cellular tower sends all these entries to A. Until now A learns both the identities and the locations of the nearby users who are willing to share their location information.

IV. Evaluation

We have implemented an experimental system based on the design presented in Section III. The client is implemented in JAVA on a MOTOROLA DROID 2 Global smartphone. A laptop is set up to emulate the cellular tower. The smartphone communicates with the laptop through Verizon's 3G data service. The social network server and the location server are deployed on two third-party cloud hosting services, provided by JoyentCloud and Linode, respectively. In our experiments we use a data set consisting of 48,014 users and the social network topology among them as a social network sample, which is collected in a separate research project [10]. We set k, the anonymity level, to be 5, and we use 128-bit AES for symmetric key encryption and decryption.

The size of our client executable is 252KB. When running, it has a memory footprint of 12MB. Figure 4 shows the client interface. The client is set to update its location every 30 seconds, and query the locations of friends or nearby strangers every 1 minute. Our evaluation shows that each hour the client only consumes 1.5% of the battery power, with average CPU utilization of 0.3%. This indicates that both the power consumption and the computation overhead incurred by the client is small.

Fig. 4. Client interface

To investigate the overhead incurred by our scheme on the cellular towers, we create a large number of dummy users on another laptop, and connect those users to the emulated cellular tower. When there are 1000 connecting users, the cellular tower service only uses 4.1% of the CPU power and 91MB memory, which shows that the cellular tower service consumes a very limited amount of system resources.

V. Conclusion

In this paper, we present MobiShare, a privacy management system that provides flexible privacy-preserving location sharing in mOSNs. By separating user identities and anonymized location updates onto two entities, users' location privacy is protected if either entity is compromised by the adversary.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-1117412 and CAREER Award CNS-0747108.

References

[1] Chronology of data breaches security breaches 2005-present. http://www.privacyrights.org/data-breach.
[2] Enhanced 9-1-1 wireless services. http://www.fcc.gov/pshs/services/911-services/enhanced911/.
[3] L. Barkhuus and A. K. Dey. Location-based services for mobile telephony: a study of users' privacy concerns. In *INTERACT*, 2003.
[4] L. P. Cox, A. Dalton, and V. Marupadi. Smokescreen: Flexible privacy controls for presence-sharing. In *ACM MobiSys*, 2007.
[5] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.
[6] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *ICPS*, 2005.
[7] B. Krishnamurthy and C. E. Wills. Privacy leakage in mobile online social networks. In *USENIX WOSN*, 2010.
[8] K. P. N. Puttaswamy and B. Y. Zhao. Preserving privacy in location-based mobile social applications. In *HotMobile*, 2010.
[9] J. Teng, B. Zhang, X. Li, X. Bai, and D. Xuan. E-shadow: Lubricating social interaction using mobile phones. In *IEEE ICDCS*, 2011.
[10] W. Wei, F. Xu, C. C. Tan, and Q. Li. Sybildefender: Defend against sybil attacks in large social networks. In *IEEE INFOCOM*, 2012.
[11] Z. Yang, B. Zhang, J. Dai, A. Champion, D. Xuan, and D. Li. E-smalltalker: A distributed mobile system for social networking in physical proximity. In *IEEE ICDCS*, 2010.
[12] G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies*, 2007.