Secure, Privacy-Preserving, and Context-Restricted Information Sharing for Location-based Social Networks

Michael Dürr, Philipp Marcus, and Kevin Wiesner
Mobile and Distributed Systems Group
Department for Informatics
Ludwig-Maximilians-University Munich
80538 Munich, Germany
Email: {michael.duerr,philipp.marcus,kevin.wiesner}@ifi.lmu.de

***Abstract*—Today, *Online Social Network* (OSN) providers like Facebook or Google provide their users with web-based *Location-based Services* (LBSs) like *Facebook Places* or *Google Latitude*. To facilitate proactive LBS, most OSN providers also offer *Location-based Social Network Services* (LB-SNSs) for mobile platforms like the iPhone, Android, or Symbian. A multitude of LB-SNSs focus on proximity detection of nearby friends. However, at present almost non of these implementations guarantees a sufficient degree of information security and privacy as well as location integrity for its users. In this paper, we address these shortcomings in two ways. We define a set of requirements we deem indispensable to enable LB-SNSs that facilitate secure and privacy-preserving information sharing and guarantee for location integrity at the same time. We then present an extension to our decentralized OSN architecture Vegas which incorporates these requirements and therefore supports the secure, privacy-preserving, and location-restricted provision of LB-SNSs.**

Keywords-Online Social Networks; Proximity Detection; Location Integrity; Privacy; Security.

I. INTRODUCTION

At present we observe a remarkable increase of *Location-based Social Network Services* (LB-SNSs) for mobile platforms like Android, Blackberry, or the iPhone operating system. One reason for this development represents a steadily increasing participation in Online Social Networks (OSNs) like *Facebook Places*, *Foursquare*, *Gowalla*, or *Google Latitude*. However, it is the implementation of such services for mobile platforms that provokes the rocketing success of such services. Powerful and easy-to-use location APIs for mobile phones as well as their precise positioning technologies like GPS can be used to implement a multitude of Location-based Services (LBSs). This also facilitates proximity detection, a frequently implemented concept in the field of proactive LBS. An example are buddy trackers that generate an event in case a friend is located within a predefined distance.

To detect proximity of friends, a user has to reveal information about his current location. Unfortunately, most people are not aware of the implications and threats caused by the distribution of location information. However, recent studies [1], [2] have shown that, as people gain a deeper knowledge about how LB-SNSs work, their demand for security and privacy increases significantly. In association with LB-SNSs that focus on proximity detection of nearby persons, one has to consider the following aspects:

1) Friendship Relevance: Although some OSN providers like Facebook support fine-grained privacy settings, most of their users do not understand how to properly apply them. Once enabled friends automatically get access to location information although the relevance of this relationship can change over time. Therefore, location information can become accessible to a multitude of persons that are considered outdated friends for a long time.

2) Third Party Providers: Most OSN providers take the role of a trusted third party. However, even in case OSN providers promise secure and confidential application of personal data, they cannot guard against security holes caused by software bugs or data abuse by employees with criminal intent. Becoming the victim of a crime due to the unsolicited dissemination of location information must not be possible at all.

3) Location Integrity: At present, a user that participates in a LB-SNS focusing on proximity detection cannot verify that location information has not been forged. This security hole could be exploited for location data mining. For instance, an adversary could distribute forged location information and thereby trigger proximity alerts at another user A. Assuming a LB-SNS requires two users A and B to share their information, the proximity alert at A would trigger the distribution of his shared information to B. The question that has not been answered yet is the following: How can a user restrict his participation in an LB-SNS such that only persons whose physical proximity he can verify are able to get access to any of his shared information?

In this paper we address these problems in the scope of LB-SNSs that focus on the detection of nearby persons with similar interest and that want to share a certain type of information. An example of such a service is *location-based ride sharing* which allows its subscribers to find each other at a railway station to share a train ticket. Another example

 ISBN: 978-1-61208-140-3

is a *location based coupling* service which allows its users to receive profiles of each other when visiting the same single party.

Generalizing these examples of LB-SNSs, we aim at a solution to share any kind of information within a predefined proximity in a secure, privacy-preserving, and location-restricted way.

The contribution of our paper is twofold. We present *a)* a requirements analysis for secure, privacy-preserving, and location-restricted LB-SNS, and *b)* an extension to our decentralized OSN architecture Vegas [3] which facilitates the implementation of LB-SNS that comply with our requirements for location integrity.

The paper is organized as follows. We present our requirements analysis in Section II and a detailed use case in Section III. Section IV gives a short overview of Vegas and details our extension for LB-SNS provision. Section V discusses our approach. Related work is presented in Section VI and Section VII concludes the paper.

II. Requirements

In our previous work [3], we identified four major requirements a secure and privacy-preserving OSN has to fulfill. These requirements encompass a user's *informational self-determination*, *strong trust relationships* between friends, anywhere and anytime *profile availability*, and transparent *mobility support*. However, to provide for secure, privacy-preserving, and location-restricted LB-SNSs, we must consider further requirements specific to location.

A. Context-Dependent LB-SNS Provision

LB-SNSs like Facebook Places allow their users to receive push notifications on their mobile device as soon as a friend initiates a *check-in event*. A check-in event is generated each time a friend registers his current location within his Facebook profile. Unfortunately, virtually no LB-SNS (including Facebook Places) offers a mechanism that allows its users to validate location information of friends. This facilitates an easy violation of location integrity as an adversary could easily trigger unsolicited LB-SNS alerts by pretending to be located at an arbitrarily chosen place. We therefore identify the requirement of *Context-Dependent LB-SNS Provision*: It must be possible to restrict LB-SNS provision by location-restricted context information. Communication between two participants must not be activated until both are located within their predefined proximity and mutually proved their physical presence. This proof could be based on frequently alternating context information exclusively available to participants at the given location. As a user can restrict LB-SNS provision to a specific context, an adversary physically absent lacks this information. Hence, he is no longer able to pretend proximity.

B. User-Selective LB-SNS Provision

A repeatedly criticized aspect of LB-SNSs are inadequate capabilities to restrict access to shared information. Facebook Places represents an extreme example, as it allows even a user to publish private location information of his friends. Hence, a user looses control over his informational self-determination [3] as well. Even in case this feature was disabled, a user cannot selectively restrict access to his location information to certain friends and has no influence on how these friends distribute his location information. To address such problems we define the requirement of *User-Selective LB-SNS Provision*. In order to increase privacy, user must be able to define a certain persons or a subset of persons in advance to restrict the provision of LB-SNS services.

C. Decentralized LB-SNS Provision

At present, virtually every LB-SNS is operated in a centralized way. Although most providers guarantee compliance with more or less restrictive privacy policies, we observe recurring privacy leaks due to unconscious software development or employees' criminal intent. To eliminate this problem, personal information must be under control of its originator. This also implies that each user should be able to control when to trigger the provision of a LB-SNS. We formulate the requirement for *Decentralized LB-SNS Provision* as unsolicited distribution of personal data by a third party must not be possible. It is obvious that this requirement also helps to achieve compliance with requirement II-A as each user is responsible for the distribution of his shared information.

III. Use Case

To better understand our design principles, we describe a sophisticated LB-SNS usage scenario that we reference throughout this paper.

A university organizes a conference that focuses on security and privacy aspects in mobile networks. In conjunction with the obligatory registration process, the conference committee decides to offer a voluntary subscription to their conference LB-SNS. They want to provide interested conference participants with a *location-based business card browser* for mobile devices. The idea of the browser service is to allow conference participants to browse each other's business card information. The browser service should be location-restricted i.e it only displays present participants and dynamically updates visible profile information of recently arriving participants.

To meet the aforementioned requirements we define further constraints:

- The browser service must not be usable anywhere else except at the conference location (req. II-A).

 ISBN: 978-1-61208-140-3

- Business cards are broadcasted automatically as a participant arrives at the conference (req. II-A and req. II-C).
- The visibility of business cards must be restricted to subscribers only (req. II-B).
- Apart from the subscription process, the browser service should work in a decentralized way (req. II-C).

IV. System Architecture

In the following, we present our extension in order to implement secure, privacy-preserving, and location-restricted LB-SNSs. We decided to build our solution on top of Vegas, a decentralized secure and privacy-aware OSN that has been developed on the design principles of our previous work [3]. We first review some parts of Vegas that are relevant to understand the functionality of our approach, before we delve into the details of our design.

A. Vegas Design

Vegas represents an OSN architecture that focuses on its users' security and privacy-demands. Its design was motivated by a set of requirements that we identified as inevitable in a secure and privacy preserving OSN. These requirements encompass a user's *informational self-determination*, *strong trust relationships* between friends, anywhere and anytime *profile availability*, and transparent *mobility support*.

The Vegas core concept does not support communication between participants that are not directly connected by an edge of the underlying social graph. This restriction is motivated by a problem we termed *social network pollution*. To give but a few examples of social network pollution, present OSNs offer the possibility for search operations on their social graphs, provide unsolicited friendship recommendations, and offer support for non-authorized linkage of a friend's friends. This causes a multitude of unwanted friendship establishments, i.e., links in the social graph which not necessarily represent a real friendship.

Figure 1 illustrates the fundamental communication principles and components of Vegas. In Vegas each user interacts with the OSN through one or more mobile or stationary clients. In order to support delay-tolerant network communication, we apply an asynchronous message exchange scheme based on the concept presented in [4]. We rely on well known services like email, SMS, or instant messaging which can be exploited to implement the *exchanger* instance. An exchanger represents the abstract concept of a message queue which is used to transmit messages or any other kind of content. Any two Vegas friends A and B are aware of one or more such exchanger addresses of each other. In addition, since a user cannot be expected to be permanently online, we introduced the *datastore* component. A datastore represents the abstract concept of a user-writable storage space with world-readable access. A datastore can be simply implemented through some simple-to-administer and cost-free web space. Each user provides one or more datastores to place his user profile encrypted and signed for each of his friends. As we will see in the next section, any kind of information flow in Vegas is subject to cryptographic operations. Every message as well as each profile are separately encrypted for each friend.

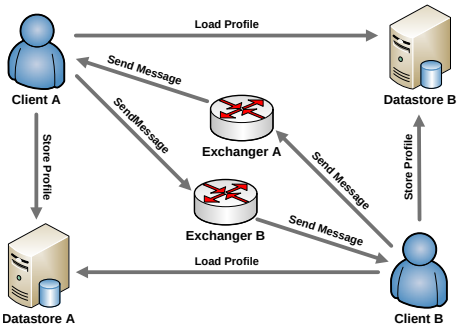

Figure 1. In Vegas any piece of information exchanged between users is encrypted and signed. Each user maintains one or more client instances and performs encrypted messaging over one or more exchanger instances. A user publishes individual profiles for each friend at one or more datastores.

B. Vegas Operation

In a nutshell, Vegas messaging and profile distribution works as follows: Any two Vegas friends A and B generate a unique public key pair which must not be applied for messaging and profile generation except in the context of A and B. In the following, we term such a key pair a *link-specific* key pair. As user A holds a unique key pair $K^{-}_{A \to X_i} / K^{+}_{A \to X_i} (i \in 1, \ldots, n)$ for each of his n friends $X_1, ..., X_n$, a key pair represents nothing else than a directed edge in the overall social graph. The notion of a key $K^{[-|+]}_{A \to X_i}$ means that this key is a private/public key generated by A for exclusive communication with X_i. A utilizes X_i's public key $K^{+}_{X_i \to A}$ to encrypt messages as well as profile information intended for X_i. In order to allow X_i to map a received message to its originator A, a fingerprint of A's public key $K^{+}_{A \to X_i}$ is included into each message sent to X_i. In case A wants to send a message to X_i, A applies X_i's public key $K^{+}_{X_i \to A}$ to encrypt the message content. After signing the message with $K^{-}_{A \to X_i}$, A sends this message to one of X_i's exchangers. Now X_i can fetch this messages and identify sender A through his attached public key fingerprint. Since X_i is the only user that knows about the mapping of the included fingerprint, X_i represents the only user that is able to map this fingerprint to the identity of A. In case A considers the mapping of a public key to X_i's identity compromised, A can trigger a key refresh operation in order to replace all former key pairs shared with X_i.

We apply the same Vegas operations for the placement and update of profile information which we use to send

 ISBN: 978-1-61208-140-3

messages. In case user A wants to update his profile, A re-encrypts the corresponding information for each of his friends $X_1, ..., X_n$ and places this information at the corresponding datastores. As A utilizes the same key material to encrypt messages as well as profile information for X_i, X_i can simply perform an interval-based read operation at A's data store in order to receive pending updates. As an alternative, A can trigger such an update by sending each of his friends a corresponding notification.

It should be stressed that a user A can cancel a friendship with user X_i by simply deleting the link-specific key pair. Hence, key revocation does not involve complex maintenance and distribution of key revocation lists.

C. Directory Buddies

Our architecture already includes the basic functionality necessary to support reactive LBSs comparable to those provided by Gowalla, Foursquare, Google Latitude, or Facebook Places. Such LBSs can be easily implemented by extending profile descriptions with location information and introducing location-dependent profile updates. To enable sophisticated LB-SNS applications like a location-based ride sharing, location-based coupling, or location-based business card browsing, we have to extend Vegas as it disallows the distribution of information to other users except Vegas friends.

To guarantee compliance with the requirements formulated in section II, we decided to extend Vegas by the concept of *directory buddies*. A directory buddy represents a special kind of Vegas friend that supports the establishment of obfuscated connections between all participants of a LB-SNS. Although not a requirement, it is likely that a directory buddy is not operated by a single person but an institution or a company that can provide additional contents. In essence, it is the task of a directory buddy to provide for anonymous connections between foreign persons. Figure 2 details how a directory buddy integrates into Vegas. A Vegas user X_i that wants to subscribe for a LB-SNS L first has to establish a friendship with a directory buddy C associated with L. X_i and C require a (semi-) trusted out-of-band (OOB) channel to exchange their public keys ($K^+_{X_i \to C}$, $K^+_{C \to X_i}$), their exchanger addresses (Ex_{X_i}, Ex_C), and their datastore addresses (DS_{X_i}, DS_C) (1). In case C does not require detailed profile information of X_i to support the provision of L, exchanger and datastore addresses of X_i need not necessarily to be exchanged. A secure exchange of this information is out of scope of this paper. An example for the application of an email-based OOB channel for this purpose can be found in [3]. Considering our use case from section III, the establishment of a friendship with C corresponds to the subscription process with L.

A LB-SNS provider determines a certain point in time at which C initiates its service provision for currently registered users (2). Assuming n users $X_1, \ldots, X_n$ registered

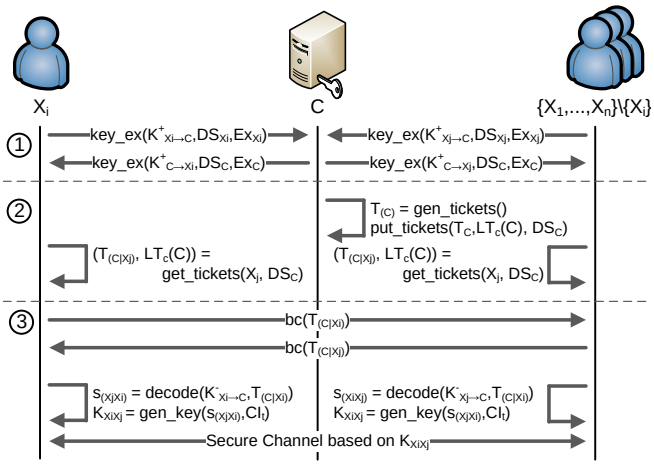

Figure 2. Message exchange between a directory buddy and its subscribers. After users have registered (1) the directory buddy generates all necessary tickets and places them for pickup at his datastore (2). As a users location task activates the LB-SNS, he broadcasts his tickets (3). Each user can decrypt tickets encrypted with his public key and create a secure channel.

for L, C calls the *gen_tickets()* function to generate a set of tickets $T_{(C)} = \{t_{(X_i X_j)} | 1 \leq i, j \leq n \wedge i \neq j\}$ for each ordered combination of users X_i and X_j. A ticket $t_{(X_i X_j)} = K^+_{X_j \to C}(s_{(X_i X_j)} || cs(s_{(X_i X_j)}))$ includes some unique piece of information $s_{(X_i X_j)}$ (where $s_{(X_i X_j)} = s_{(X_j X_i)}$) which is generated by C to provide two registered users X_i and X_j with a common secret. $cs(s_{(X_i X_j)})$ represents the checksum of $s_{(X_i X_j)}$. C generates a *location task* $LT_c(C)$ for each user X_i. A location task corresponds to the description of some context c a user X_i has to observe before X_i initiates his active participation in the corresponding LB-SNS. Referring to our use case, $LT_c(C)$ could be defined as the geographic dimensions of the conference building. In this example each participant would receive the same location task. Now C calls the *put_tickets()* function to place disjoint subsets $T_{(C|X_i)} = \{t_{(X_i X_j)} | 1 \leq j \leq n \wedge i \neq j\} \subset T_{(C)}$ and the location task $LT_c(C)$ individually encrypted based on $K^+_{X_i \to C}$ for each user X_i at datastore DS_C. Dependent of a preconfigured interval, X_i calls the *get_tickets()* function to receive his private subset of tickets $T_{(C|X_i)}$ and the location task $LT_c(C)$ from DS_C.

When X_i observes the occurrence of $LT_c(C)$, X_i starts to broadcast $T_{(C|X_i)}$ (3). For instance, in case of our usage scenario from section III, the context c corresponds to the arrival at the conference building. Any other registered user X_j ($i \neq j$) who already observed the occurrence of $LT_c(C)$ also broadcasts his ticket subset $T_{(C|X_j)}$. When X_i receives a broadcast from X_j, X_i calls the *decode()* function to decode the secret $s_{(X_i X_j)}$ shared between X_i and X_j. In case there exists a ticket $t_{(X_i X_j)} \in T_{(C|X_j)}$, X_i can extract the secret $s_{(X_i X_j)}$ by applying his private key $K^-_{X_i \to C}$ to $t_{(X_i X_j)}$. As X_i has no previous knowledge about the public key that was used to encrypt a certain ticket, X_i has to

 ISBN: 978-1-61208-140-3

calculate the checksum of an encrypted secret in order to determine that this secret corresponds to $s_{(X_iX_j)}$. Section V discusses this process in detail. It should be stressed that, although X_i and X_j now share a common secret, C is always able to decrypt information which was previously encrypted based on $s_{(X_iX_j)}$.

Although X_i and X_j cannot trigger their participation in a LB-SNS before they were able to exchange their tickets, i.e., before they are located within the same broadcast domain, the shared secret $s_{(X_iX_j)}$ does not suffice to give evidence for the occurrence of $LT_c(C)$. For instance, a fraudulent user M could pretend to be located within the broadcast domain physically restricted by $LT_c(C)$. M could connect to the corresponding subnetwork by establishing a VPN tunnel link or driving a wormhole attack. To generate a secret key from $s_{(X_iX_j)}$ that cannot be determined before X_i and X_j share the same broadcast domain and are physically located at an area described by $LT_c(C)$, we require an additional context information CI_t which cannot be determined except at the location described by $LT_c(C)$. Referring to our use case, such additional context information could be an identifiable but randomly chosen and frequently alternating essid of the wireless conference network. A shared secret key $K_{(X_iX_j)}$ could then be generated by the *gen_key()* function based on the shared secret $s_{(X_iX_j)}$ and a context information CI_t. As decryption could fail due to the application of an outdated CI_t for the generation of $K_{(X_iX_j)}$, X_i and X_j temporarily store recently expired keys. This facilitates the decryption of messages that were encrypted with a predecessor of $K_{(X_iX_j)}$ although $K_{(X_iX_j)}$ represents the present key.

It is worth mentioning that the generation and distribution of tickets in step (2) of Figure 2 is not necessarily bound to a fixed point in time. This process can be executed incrementally. In case a new participant subscribes with the LB-SNS, C updates all ticket subsets $T_{(C|X_i)}$ for each participant X_i on demand and places them at DS_C. Eventually, X_i receives this update in dependence of his pre-configured update interval.

V. Discussion

Our concept of directory buddies provides the foundation for secure, privacy-preserving, and context-restricted LB-SNSs. In this section we discuss compliance with our requirements from section II. Furthermore we give some notes on performance and security.

A. Compliance with Our Requirements

In this work we extended the existing OSN architecture Vegas which already meets the requirements to provide for a secure and privacy-preserving OSN [3]. In the following, we only discuss requirements II-A, II-B, and II-C.

As the provision of a LB-SNS is based on a location task $LT_{(C)}$, no information is shared before a user observes the occurrence of $LT_{(C)}$. Furthermore, LB-SNS provision is restrict to participants that have knowledge of some predefined additional context information. To share information, two users have to provide each other with an evidence that they fulfill $LT_{(C)}$, i.e., they are located within physical proximity. Hence, our approach meets the requirement for Context-Dependent LB-SNS Provision. It should be mentioned, that our requirement for Context-Dependent LB-SNS Provision impedes security attacks as an adversary has to be physically present or at least needs an in situ accomplice.

Each user that wants to participate in a LB-SNS has to subscribe with a directory buddy in advance. This component only serves for the establishment of anonymous links between users which they can use to provide each other with encrypted information, i.e., no identities are revealed. As each link is secured with a separate secret, a user can provide another user with additional information only shared between both of them. Therefore, we achieve User-Selective LB-SNS Provision.

As each subscriber has control over the point in time when to trigger his LB-SNS participation, he has also full control over the provision of shared information. As all information is broadcasted by each user himself, we achieve a completely decentralized, fully meshed communication scheme. Therefore, our solution adheres to the requirement for Decentralized LB-SNS Provision.

B. Comments on Performance

As mentioned before, our design applies one ticket per anonymous link. In case a user X_i receives a ticket broadcast $T_{(C|X_j)}$ from X_j, X_i has to apply his private key $K^{-}_{X_i \to C}$ to each included ticket in order to determine the shared secret $s_{(X_iX_j)}$. Hence, the process to decrypt each ticket $s_{(X_iX_j)}$ for all subscribers $X_j \in \{X_1, \ldots, X_n\} \setminus \{X_i\}$ has a complexity of $O(n^2)$. However, we can decrease the complexity to $O(n)$ if we add an identifier to each ticket. For instance C could determine a set of unique identifiers for each ticket $t_{(X_iX_j)}$, add one identifier to each ticket, and announce relevant identifiers within each subscriber's profile information at DS_C.

Except the necessity of a single broadcast domain, LB-SNSs are completely independent of the underlying network. Nevertheless it should be mentioned that, in case of an ad hoc network, a LB-SNS could suffer from packet loss due to frequent ticket broadcasts. Dependent of the MAC layer, subscribers should carefully choose a suitable broadcast interval.

C. Comments on Security

Since tickets are distributed via broadcast, it takes an adversary no effort to replay once received ticket broadcasts. However, this does not represent a security problem as a ticket $t_{(X_iX_j)}$ cannot be replayed before X_j sent the ticket for the first time. Since X_i maintains a list of all decrypted

 ISBN: 978-1-61208-140-3

tickets, replayed tickets are simply ignored as soon as X_i received $t_{(X_iX_j)}$ for the first time. As long as a directory buddy can be trusted, our solution facilitates secure, privacy-preserving, and location-restricted LB-SNSs.

VI. Related Work

A plethora of research has been conducted in the field of secure and privacy-aware LBS. In the following, we review related work from the area of privacy-aware proximity detection and location-based access control as we consider it the most relevant fields in our context.

Many proposals have been published for privacy-aware proximity detection for LBS. (e.g., [5]–[7]). However, it has been shown [8] that enabling privacy-awareness for LBSs is a challenging task. Recently, Šikšnys et al. presented their *vicinitylocator* [9], a proposal for private and flexible proximity detection in mobile social networks. Their architecture supports proximity detection by checking for inclusion of one user's location inside another user's vicinity. A server instance attempts to map encrypted proximity regions (granules) presented by one user with the granules presented by another one. Although this scheme facilitates a better degree of privacy, proximity detection is still performed server-sided, i.e. a user cannot validate location integrity due to the lack of a common context. Zhong et al. developed different protocols [10] to support privacy-preserving proximity-detection of nearby friends. However, even in case they apply a trusted third-party to avoid situations where friends learn locations of users that must no longer be considered nearby, their approach cannot guarantee location integrity. A paper recently published by Puttaswamy et al. [11] probably shares most similarities with our approach. It describes a decentralized approach to enable privacy-preserving location-based mobile social applications. Mobile devices place their encrypted location data at third-party servers. Participants of the social network may download and decrypt that information in case they share a secret key with the data originator. Although their approach facilitates decentralized proximity detection it does not adhere to our requirement for location integrity. Users can preserve privacy but they cannot prove location integrity.

Some work published by Sastry et al. [12] focuses on the *secure in-region verification problem*. Their approach is related to our location integrity problem, as a prover has to prove its claimed location to a verifier. To address this problem the authors measure emitted ultrasound signals to verify a location. As recent mobile devices are not equipped with ultrasound measurement facilities, their approach will not be applicable in the near future.

VII. Conclusion

We presented our approach for secure, privacy-preserving, and location-restricted location-based services for social networks (LB-SNS) that focus on information sharing with nearby persons. We first elaborated a set of requirements we deem indispensable to provide for the provision of LB-SNSs. We presented the concept of directory buddies, an extension to our decentralized, secure, and privacy-preserving OSN architecture Vegas and illustrated its application in context of LB-SNSs by exemplifying a location-based business card browser. We showed that this design fully complies with our requirements.

Our future work will focus on the evaluation of our design in a realistic setup. Furthermore, we will investigate novel possibilities to infer context information to provide for location integrity from environmental impacts.

References

[1] J. Tsai, P. Kelley, L. Cranor, and N. Sadeh, "Location-sharing technologies: Privacy risks and controls," *ISJLP*, vol. 6, pp. 119–317, 2010.

[2] K. Tang, J. Lin, J. Hong, D. Siewiorek, and N. Sadeh, "Rethinking location sharing: exploring the implications of social-driven vs. purpose-driven location sharing," in *Proceedings of UbiComp'10*. ACM, 2010, pp. 85–94.

[3] M. Dürr, M. Werner, and M. Maier, "Re-Socializing Online Social Networks," in *Proc. of CPSCom'10*. IEEE, Dec 2010.

[4] M. Werner, "A Privacy-Enabled Architecture for Location-Based Services," in *Proc. of MobiSec '10*, 2010.

[5] P. Ruppel, G. Treu, A. Küpper, and C. Linnhoff-Popien, "Anonymous User Tracking for Location-Based Community Services," in *Location- and Context-Awareness*, ser. LNCS. Springer Berlin/Heidelberg, 2006, vol. 3987, pp. 116–133.

[6] A. Küpper and G. Treu, "Efficient proximity and separation detection among mobile targets for supporting location-based community services," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 10, pp. 1–12, July 2006.

[7] S. Mascetti, C. Bettini, and D. Freni, "Longitude: Centralized Privacy-Preserving Computation of Users Proximity," in *Secure Data Management*, ser. LNCS. Springer Berlin/Heidelberg, 2009, vol. 5776, pp. 142–157.

[8] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proc. of SSTD '07*. Springer Berlin/Heidelberg, 2007, pp. 239–257.

[9] L. Šikšnys, J. Thomsen, S. Šaltenis, and M. Yiu, "Private and Flexible Proximity Detection in Mobile Social Networks," in *Proc. of MDM 2010*. IEEE, 2010, pp. 75–84.

[10] G. Zhong, I. Goldberg, and U. Hengartner, "Louis, Lester and Pierre: three protocols for location privacy," in *Proc. of PET '07*. Springer Berlin/Heidelberg, 2007, pp. 62–76.

[11] K. P. N. Puttaswamy and B. Y. Zhao, "Preserving privacy in location-based mobile social applications," in *Proc. Hotmobile '10*. New York, NY, USA: ACM, 2010, pp. 1–6.

[12] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Proc. of WiSe '03*. New York, NY, USA: ACM, 2003, pp. 1–10.

 ISBN: 978-1-61208-140-3