Resilient Collaborative Privacy for Location-Based Services

Hongyu Jin(✉) and Panos Papadimitratos

Networked Systems Security Group, KTH Royal Institute of Technology, Stockholm, Sweden
{hongyuj,papadim}@kth.se
http://www.ee.kth.se/nss

Abstract. Location-based Services (LBSs) provide valuable services, with convenient features for users. However, the information disclosed through each request harms user privacy. This is a concern particularly with *honest-but-curious* LBS servers, which could, by collecting requests, track users and infer additional sensitive user data. This is the motivation of both *centralized* and *decentralized* location privacy protection schemes for LBSs: anonymizing and obfuscating LBS queries to not disclose exact information, while still getting useful responses. Decentralized schemes overcome the disadvantages of centralized schemes, eliminating anonymizers and enhancing users' control over sensitive information. However, an insecure decentralized system could pose even more serious security threats than privacy leakage. We address exactly this problem, by proposing security enhancements for mobile data sharing systems. We protect user privacy while preserving accountability of user activities, leveraging pseudonymous authentication with mainstream cryptography. Our design leverages architectures proposed for large scale mobile systems, while it incurs minimal changes to LBS servers as it can be deployed in parallel to the LBS servers. This further motivates the adoption of our design, in order to cater to the needs of privacy-sensitive users. We provide an analysis of security and privacy concerns and countermeasures, as well as a performance evaluation of basic protocol operations showing the practicality of our design.

Keywords: Location-based service · Security and privacy · Pseudonymous authentication

1 Introduction

The evolution and popularization of mobile Internet brings forth opportunities for service providers to cater to people's needs. Location-based Services (LBSs) in particular respond to user queries based on their locations, available at their location-aware mobile devices. However, the improved relevance and precision of responses comes at a cost: users' privacy can be harmed [8,26]; user location information can be used to reconstruct user trajectories and profile their

S. Buchegger and M. Dam (Eds.): NordSec 2015, LNCS 9417, pp. 47–63, 2015.
DOI: 10.1007/978-3-319-26502-5_4

activities or even infer their interests. In fact, the LBS itself, i.e., its server(s), is uniquely positioned to undermine users' privacy, collecting rich information over time, for all locations a user (client or mobile device/application) submits queries from. Moreover, it can have a financial motivation to do so, seeking to push advertisements to users. As a result, increased concerns have been voiced and numerous efforts to safeguard user privacy led to a number of proposals, both *centralized* and *decentralized*.

Centralized schemes [13,23,25] introduce a new entity, an *anonymizer*: it anonymizes a received client query (removing its identity attributes), obfuscates and/or blends the queries of multiple clients, and then sends them to the LBS server(s). The location is obfuscated to a corresponding region with the client and at least $k-1$ other clients included, to achieve k-anonymity: the user is indistinguishable among these k users. Clearly, these schemes are effective but this centralized approach seeks to solve the problem at hand based on the assumption that originally raised concerns for the LBS servers; they presume the anonymizer is trustworthy. But still, the anonymizer has all the rich information collected from client queries. If an LBS server can be curious and track or profile users, the question rises naturally: *Why couldn't an anonymizer also breach the user privacy the same way*?

This challenge motivated a number of works that proposed decentralized privacy schemes. Similar, in spirit, to decentralized approaches overcoming disadvantages of centralized approaches for privacy-related problems in many areas [12,17,24], LBSs user privacy protection can be achieved in a collaborative manner: without relying on an anonymizer. In particular, users can hide from the LBS server by obtaining LBS-provided information from their neighbors [29].

Nonetheless, opening up the system functionality is a double-edged sword: it reduces the user exposure to the curious provider (LBS or anonymizer) but it also exposes her to possibly faulty or misbehaving peers. In fact, risks in and abuses of, for example, peer-to-peer (P2P) systems [18,20,30] show that insecure decentralized schemes face serious problems. For example, users are threatened by exposure of their sensitive information to other peers or injected bogus data from malicious nodes. In [29], responses from the LBS server are signed, thus they are self-verifiable while passed to other peers. However, this does not comply with many existing LBS servers, which authenticate themselves and secure only pairwise communication over, e.g., a TLS channel, instead of signing the responses. In either case, peers could be uncooperative or offending.

This challenge exactly motivates our work, in the context of enhancing LBS user privacy. The decentralized or *collaborative* approach has clear advantages, enhancing the users' control over sensitive information: their exposure can be significantly reduced while they still obtain their sought quality of service (trading off mild delay for much better privacy). But this would be of no use if user peers could disrupt or even debilitate the collaborative querying part, by passing on bogus or irrelevant information, or excessively querying their peers. Even if LBS responses were signed, still, misbehaving peers can aggressively consume resources of benign peers and obstruct the peer query-response operation. Or,

worse even, abuse peer queries to also harm users' privacy based on the peer-to-peer data exchange.[1]

This is what we address in this paper: we propose a security architecture for decentralized/collaborative privacy protection for LBSs. We are cognizant that already deployed LBS servers would be unwilling to change their operations, thus we propose new components that are orthogonal to the LBS servers and new functionality for the privacy-sensitive users. In fact, we conjecture that our scheme could even motivate LBS servers to adopt and offer the collaborative privacy-enhancing scheme to their interested users. While, in turn, the users would be further motivated to embrace it knowing that it protects them from unwanted risks (manipulation, overloading) and, at the same time, safeguards their privacy. Moreover, we impose constraints on pseudonym usage to further protect users from being inundated with bogus data.

In the rest of the paper, we first outline requirements and discuss related work (Sec. 2). Then, we present our proposed scheme (Sec. 3), we analyze the achieved security and privacy protection (Sec. 4), benchmark our protocol on mainstream mobile devices (clients) and provide a performance evaluation (Sec. 5), and conclude with our next steps (Sec. 6).

2 Problem Statement and Related Work

2.1 System and Adversary Model

System: Consider a general model, as illustrated in Fig. 1: Users' mobile clients, termed *nodes* in the rest of the paper, can be smartphones, tablet PCs, On-board Units (OBUs) in vehicles, etc. They are connected to the Internet through different channels (Wi-Fi and/or cellular network) and they are interested in different types of location-dependent information; e.g., specific Point of Interest (POI) information, traffic status, environmental conditions, etc. Nodes are able

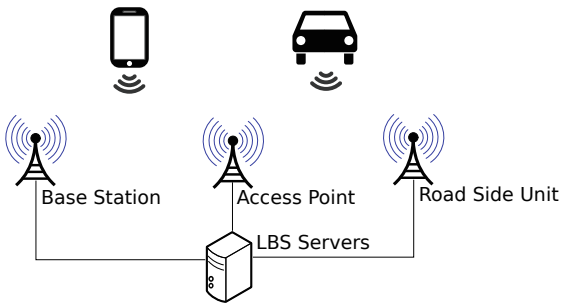

Fig. 1. System Model

[1] Although, of course, an adversary would need a massive number of peers to collect, each one locally, the same information an LBS would and is able to collect simply through its regular operations.

to request these information from LBS servers through the Internet. Nodes can in addition communicate with other nodes through ad-hoc connectivities, including Wi-Fi ad-hoc network, Wi-Fi Direct [5], LTE Direct [2] and Bluetooth. This allows them to exchange information with other nodes; thus sharing information with each other or aggregating data obtained from multiple peers. We assume this is an alternative way of obtaining location-dependent information while hiding from the LBS servers. This can be achieved by running an application on mobile devices; which obtains information from LBS servers through provided APIs [1,4], and shares the information with other nodes.

Adversaries: We assume that LBS servers are *honest-but-curious*: they follow the protocols, responding faithfully to their users' (nodes') queries. But they can trace the nodes (linking their queries), profile the nodes (recording their queries), and even deanonymize the nodes (inferring home and work sites). Such inferred sensitive data, based on the collected queries, could be commercially exploited. We maintain the same assumption for any third party, including the ones we introduce in our architecture (see Sec. 3).

Nodes can be honest, honest-but-curious or malicious. In the latter case, they can deviate from the collaborative protocol functionalities and policies, attacking the systems, notably their peer nodes: forging or tampering with responses, masquerading other nodes, excessively posting queries to their peers, seeking to exhaust their resources. The result could be degradation of the service users receive (i.e., being misled) or even a Denial of Service (DoS) on the collaborative exchange. These would not only affect quality of service but also force honest nodes to expose themselves to the LBS servers.

2.2 Security and Privacy Protection Requirements

We seek to thwart the aforementioned node misbehavior, while maintaining the benefit of "hiding" from the LBS servers and obtaining useful information.

Authentication and Integrity: Node messages, queries and responses, should allow their receiver to authenticate their sender and verify they were not modified or replayed from a previous exchange. We do not require strict identification of the sender (querier or responder) but at least validation that the sender is a legitimate participant of the P2P operation.

Non-repudiation and Accountability: The sender of any message (any action, in general) cannot deny having sent the message (taken the action). Any node can be tied to its actions, if need arises, and held accountable. Accordingly, it should be possible to have such nodes evicted from the system (the P2P operation).

Anonymity/Pseudonymity and Unlinkability: Nodes should not be identifiable, based on their P2P interactions, and have their messages linked to their identities. Anonymity should be conditional, allowing the system to identify a misbehaving node (and evict it). Ideally, we want to make it impossible for any

observer to link any two messages (e.g., queries) to the same node. But, for practical operation and efficiency/lower cost reasons, we require that node actions (messages) can be linked at most over a protocol selectable period τ. Accordingly, any node can maintain one temporary identifier, a *pseudonym*, for that same period.

Confidentiality (optionally): The LBS-originating content should be accessible only by legitimately participating nodes, possibly registered with the LBS and the system/application that enables collaborative privacy protection.

2.3 Related Work

We discuss briefly decentralized approaches to enhance privacy for LBSs. As with centralized approaches, many decentralized schemes seek to provide k-anonymity: P2P spatial cloaking [11] and MobiHide [14] achieve k-anonymity by finding $k-1$ neighboring peers within a cloaked region. We do not dwell on how effectively this can be done (e.g., unlinkability under the assumption that nodes are not likely to move in the same direction). However, we note that the trust model among nodes (users) was not considered [11]; while MobiHide [14] relies on a central server who maintains a list of active nodes and supports them on joining the clusters, which is a privacy threat for users. Along the same lines, AMOEBA [28] protects user privacy by forming groups and delegating LBS queries to group leaders, proposed for vehicular communication systems; the predictable mobility helps in that case. The formation of groups, of course, imposes additional complexity and overhead. If the conditions allow such group operation and it is effective, it could be beneficial. But such group formation and provision of k-anonymity is orthogonal to our work here, and it could possibly co-exist with (and even be facilitated by) our scheme, by explicitly addressing trust assumptions and providing a security architecture.

Passing/sharing self-verifiable information among users helps to provide authentication and integrity [29]. Nodes cache information received from the LBS server and pass to its neighbors when requested, thus decreasing exposure to the LBS server. This is the approach we extend in this paper. It assumes that responses signed by the LBS server are self-verifiable (manipulation by a node will be detected). However, even with such signatures, assuming of course a change on the side of LBS (possibly considered unrealistic by some providers), a misbehaving node passing on tampered responses would remain "invisible" and continue attacking the system. This is exactly where this work comes in, protecting the system against node misbehavior. Moreover, it is interesting that this does not comply with many existing LBS servers: user-/node-server communication is authenticated (and kept confidential) through end-to-end security (a secure channel, e.g., TLS, with the LBS server), without signed responses.

The EU PRIME project [6] proposes the use of anonymous credentials in the context of LBS. This is related to our work, but a location intermediary is assumed between the LBS and the mobile operator. The use of non-traditional

public key cryptographic protocols has also been considered in [9,10,22,27], with special care for sybil-free operations, in spite of the relatively higher overhead for those cryptographic primitives.

3 Our Scheme

3.1 Overview

We assume basic collaborative functionality for nodes, sharing location-dependent information [29]: before querying the LBS, a node queries its neighbors/peers, who respond if they can; if no appropriate response is obtained in this P2P manner, then the querier cannot but query the LBS. To secure such a system, as per the requirements in Sec. 2, we propose a security architecture and augment the basic P2P functionality, and the node-to-LBS communication. We assume nodes keep track of all communication in the vicinity, and react appropriately to P2P queries; listening if a query was already served by other nodes. This is straightforward to support in commodity wireless networks (e.g., Wi-Fi). Table 1 summarizes the used notation.

Table 1. Notation

$LTCA$	*Long-Term Certification Authority*
Lk/LK	*Long-term Private/Public Key*
LTC	*Long-Term Certificate*
PCA	*Pseudonymous Certification Authority*
Sk/SK	*Short-term Private/Public Key*
PC	*Short-term (Pseudonymous) Certificate*
$\{msg\}_{\sigma}$	*Signed msg*
$type_{poi}$	*Type of POI*
id	*Node id or Query id*
t/t_{now}	*Timestamp/A fresh timestamp indicating current time*
$T_{timeout}$	*Timeout for peer response reception*
SN	*Serial Number*
N	*Number of needed responses to a peer query*

Fig. 2 illustrates the proposed system architecture. We mandate that nodes are registered with an *identity and credential management facility* that equips them with *short-lived anonymized credentials*. To do so, we require the nodes be registered with an Long-Term Certification Authority (LTCA) that maintains their long-term identities and issues Long-Term Certificates (LTCs) for them. With the LTC, a node obtains a ticket from the LTCA and present the ticket to the Pseudonymous Certification Authority (PCA) to obtain *Pseudonymous Certificates (PCs)/pseudonyms*. The ticket is authenticated by the LTCA but

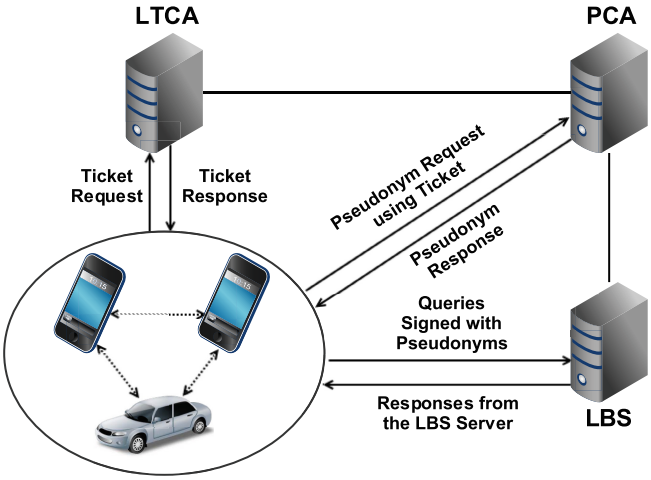

Fig. 2. System Architecture

anonymized: it does not reveal real identity of the node to the PCA. Therefore, neither the LTCA nor the PCA can link the real identity of the node to the issued pseudonyms (thus, the messages signed under the pseudonyms). The details of the operation are presented next (Sec. 3.2, 3.3).

We require that the pseudonyms be used to authenticate (with the corresponding cryptographic private key) P2P queries and responses. They can be optionally used to authenticate queries to the LBS (if its functionality allows that). The pseudonyms attest to the *legitimate participation* of the node in node-to-LBS or P2P communication. Furthermore, to prevent abuse of the node anonymity, our scheme provides *conditional anonymity* and allows *revocation of anonymity and eviction*. The node interactions with the facility entities are explained below. Moreover, we enforce ticket and pseudonym (lifetime) policies on the Certification Authorities (CAs) and nodes, so that user privacy is protected to the full extent and the nodes have *regulated access* to P2P part of the LBS. Finally, our extension of the P2P functionality allows for increasing resilience and user-control: the querying node can seek multiple responses and can regulate the maximum rate at which it responds to queries.

3.2 Protocols

Registration: All nodes register with an LTCA, which essentially acts as an identity provider. (1) A node generates a pair of long-term public/private keys, LK and Lk, and (2) submits a Certificate Signing Requests (CSR) (a self-signed LK and other relevant information) to the LTCA; (3) the node is issued with an LTC. The whole exchange is secured with a TLS channel or done offline.

$$C : Lk, LK \tag{1}$$

$$C \rightarrow LTCA : \{id_C, LK, others\}_{\sigma_{Lk}} \quad (2)$$

$$LTCA \rightarrow C : LTC_C = \{SN_{LTC}, id_C, LK, others\}_{\sigma_{LTCA}} \quad (3)$$

Ticket and Pseudonym Acquisition: (4) A node requests a ticket with a desired pseudonym validity starting time, t_{start}. The length of ticket validity period is defined by system policy, thus no need to be specified by the node. (5) The LTCA checks if a ticket was issued with an overlapping lifetime; if not, (6) it issues a ticket with validity period $[t^{'}_{start}, t^{'}_{end}]$. The ticket validity period is computed by the LTCA based on t_{start} and the policy defined in [19] to prevent ticket and pseudonym linkability.

With the ticket in hand, the node can obtain a set of pseudonyms (7-9) from any associated PCAs, which acts as a service provider itself. There can be multiple that recognize/accept the LTCA tickets; for the sake of presentation, without loss of generality, we refer to a single PCA. The anonymized ticket does not reveal anything about the identity of the node (and the user) to the PCA. This separation of duties concept is based on the work done in the context of vehicular communication systems [16,19]. Both ticket and pseudonym acquisitions are protected with TLS channels. The ticket request is protected by mutual authentication; while the pseudonym request is protected by unidirectional (PCA-only) authentication, since the node is authenticated with the presented ticket.

$$C \rightarrow LTCA : ticket\_req\{t_{start}\}_{\sigma_C} \quad (4)$$

$$LTCA : check(id_C, t_{start}) \quad (5)$$

$$LTCA \rightarrow C : ticket = \{SN_{ticket}, t^{'}_{start}, t^{'}_{end}\}_{\sigma_{LTCA}} \quad (6)$$

$$C : Sk, SK \quad (7)$$

$$C \rightarrow PCA : pseudonym\_req\{ticket, \{SK\}_{\sigma_{Sk}}\} \quad (8)$$

$$PCA \rightarrow C : PC = \{SN_{pc}, SK, t^{'}_{start}, t^{'}_{end}\}_{\sigma_{PCA}} \quad (9)$$

P2P Query: Algorithm 1 illustrates the querying thread of a node. As stated above, nodes cache locally responses received from the LBS server (and other peers). When POI information is needed, the local cache is checked first. If there is no match, it generates a signed query. To ensure unlinkability after a change of pseudonym, the node can randomly reset its IP and MAC address. The node waits for and possibly receives responses from its neighbors. It can specify in the query that N responses are required in total from its peers and assign a query id (id_q). It then verifies the responses and combine them to form a final response. Each receiver could overhear the responses to the same query while the query is queued, and serve the query only while less than N responses are overheard from the network (see Sec. 3.3 for detail). Moreover, a node could adapt to current CPU usage and battery amount remaining, the rate at which to serve peer queries for further reducing the overhead. We do not formulate what is a satisfactory response to a query, it can be specified in the preferences of the application or determined through UI (e.g., a button indicating the user wants to query the LBS server directly) after being presented the peer responses.

Algorithm 1. Querying thread (of a node)

Possesses a valid PC
$query = \{loc, type_{poi}\}$
$resp_{local} = search(query)$
if $resp_{local}$ is satisfactory **then**
    $resp_{final} = resp_{local}$
else
    $QUERY = \{id_q, t_{now}, query\}_{\sigma_{PC}}$
    $broadcast(\{QUERY, PC\})$
    Let $resp_{final} = \phi$, $n = 0$, $t = t_{now} + T_{timeout}$
    while $\{RESP_i, PC_i\} = receiveRespBefore(t)$ and $n < N$ **do**
        $RESP_i = \{id_q, t_{now}, resp\}_{\sigma_{PC_i}}$
        if $verify(PC_i, LTC_{PCA})$ and $verify(RESP_i, PC_i)$ **then**
            $resp_{final} = combine(resp_{final}, resp_i)$
            $n = n + 1$
        end if
    end while
    if $resp_{final}$ is not satisfactory **then**
        $resp_{final} = queryLBS(QUERY)$
    end if
    $cache(resp_{final})$
end if
return $resp_{final}$

LBS Query: Nodes query the LBS server only when they have to, e.g., when the information obtained from their neighbors is not satisfactory. The information obtained from the LBS server is the essential resource for supporting P2P function of our scheme. Nodes send the signed queries to the LBS server. Then, the responses from the LBS server are cached by the nodes.

P2P Query Processing: As shown in Algorithm 2, when a node receives a peer query, it first verifies the attached pseudonym and checks if the attached

Algorithm 2. Serving thread (of a node)

Possesses a valid PC
$\{QUERY_i, PC_i\} = receiveQuery()$
$QUERY_i = \{id_q, t_{now}, query\}_{\sigma_{PC_i}}$
if $verify(PC_i, LTC_{PCA})$ and $verify(QUERY_i, PC_i)$ **then**
    $query = \{loc, type_{poi}\}$
    $resp = search(query)$
    if $resp \neq \phi$ **then**
        $RESP = \{id_q, t_{now}, resp\}_{\sigma_{PC}}$
        $send(i, RESP)$
    end if
end if

pseudonym has been over-used (i.e., received queries signed under the same pseudonym exceeds the query rate allowed for one pseudonym). If not, it verifies the query and searches its cache. If successful in finding matching information, it signs and sends the response to the sender. This, of course, depends on whether or not N responses to the same query have been overheard from the network.

Reporting Misbehavior: We address post-misbehavior processing while the misbehavior detection is out of scope of this paper. However, we note that some types of misbehavior are straightforward to detect and confirm. For example, the honest LBS responses can help finding out which of the contradictory responses to a query is/are bogus information. When a misbehavior is detected by a node, (10) it sends to the Resolution Authority (RA), the messages related to the misbehavior with pseudonyms attached. In case (11) the messages are proved to be related to a misbehavior case, (12) it sends the pseudonym (or multiple pseudonyms) to the PCA, and (13) the PCA derives the SN_{ticket} of the ticket that had been used to issue the pseudonym. (14-15) With the help of the LTCA, the misbehaving node is exposed (and possibly evicted from the system).

$$C \rightarrow RA : \{\{msg\}_{\sigma_{PC_i}}, PC_i\}_{\sigma_{PC}} \tag{10}$$

$$RA : judge(msg) \tag{11}$$

$$RA \rightarrow PCA : PC_i \tag{12}$$

$$PCA \rightarrow RA : SN_{ticket} \tag{13}$$

$$RA \rightarrow LTCA : SN_{ticket} \tag{14}$$

$$LTCA \rightarrow RA : id_i \tag{15}$$

3.3 Optimizations for Query Processing

Processing a query requires searching the node cache and two signature verifications: one for the attached certificate (pseudonym) and one for the sender's signature. Similarly, the response validation requires two signature validations, and checking if the nonce and location matches that of the query. To reduce communication and processing overhead, we propose the following optimizations:

- **Optimization 1:** A node could sign multiple queries/responses under the same pseudonym, thus it may omit attaching it to some of those - thus reducing communication overhead. Accordingly, the receiving nodes need to validate the pseudonym of the said node only once throughout the pseudonym lifetime. Then, they cache validated pseudonyms and omit their verification for successive queries/responses signed under cached pseudonyms [10]. This way, the processing overhead is reduced, as only one signature needs to be validated for peer queries and responses. This is important, as the PCA key would in principle have high security level, thus relatively higher verification delay.
- **Optimization 2:** Each node could opportunistically cache responses to popular queries (both overheard and locally generated), assuming such information is likely to become useful later. The popularity can be determined by

occurrence frequency of type of POI in queries or responses, while it is protocol selectable. In case an incentive scheme is used, caching popular responses would provide increased rewards.
- **Optimization 3:** Requesting multiple, N, responses allows cross-checking (Sec. 4), but could waste resources if responses are sent after the querier obtained all needed responses. Responders can back-off randomly and overhear communications, counting responses to the specific query, based on its id_q; then, at the end of the back-off they respond only if less than N responses were overheard.

4 Security and Privacy Analysis

In this section, we explain how the security and privacy requirements are addressed and how malicious behavior is thwarted.

Authentication, Integrity, and Confidentiality: The communication of the nodes with the CAs and the LBS server is carried over TLS channels, thus providing end-to-end security. Signing P2P messages under pseudonyms provides authentication and integrity. While confidentiality of P2P communication is optional, any two nodes can establish a shared session key and mutually authenticate each other leveraging their pseudonymous certificates, encrypting the response(s) with the session key. Thus, only users registered with the system have access to LBS-provided information.

Non-repudiation and Accountability: The use of public key cryptography and the digital signatures ensure non-repudiation. Any suspected misbehavior (messages deemed to be inconsistent, bogus, etc., by a node) can be linked to the signer's pseudonym. This can be reported to the security infrastructure and the LTCA and PCA can jointly identify the node and if necessary evict it - revoking valid pseudonyms and/or preventing it from obtaining new pseudonyms.

Unlinkability: Keeping ticket request records at the LTCA prevents nodes from excessively requesting pseudonyms with overlapping lifetimes - we enforce non-overlapping pseudonym lifetimes and similarly to [19] we enforce that all tickets and pseudonyms are issued at discrete times for all requests to the PCA. This precludes linkability of pseudonyms of the same node based on time of issuance and lifetime - the likelihood is inversely proportional to the number of all active pseudonyms in the system. Actions of a node are linkable only as long as the same private key (under the same pseudonym) is used, that is, only over the period τ. Setting this is a trade-off between unlinkability and efficiency. Changing node identifiers across the protocol stack (IP, MAC) precludes linkability across pseudonym changes.

Node Authentication and Exposure to the LBS Server: For subscriber-based LBSs, node authentication is necessary. Optionally, nodes can be authenticated to the LBS server with long-term credentials or pseudonyms based on the requirements of a specific LBS server. Use of long-term credentials would make

the nodes identifiable and queries linkable. While pseudonymous authentication ensures nodes authentication without revealing their identities and breaching unlinkability, if done under different pseudonyms.

Non-verifiable Responses: If the LBS server signs responses, their integrity and timeliness can be readily verified. Otherwise, any malicious node could forge bogus responses, or any node create an arbitrary, valid yet unverifiable response based on its cache. We don't address this aspect here; we only suggest that queriers request redundant responses in order to cross-check them and infer valid information, e.g., extracting and using only information included in the majority of the responses from distinct peer nodes. Such a scheme warrants a separate investigation and it is part of future work. We note, however, that the authentication of the nodes and the constraints imposed by our scheme prevent an adversary from posing as multiple nodes and inundating the receiver with bogus responses.

Essentially, honest LBS responses can serve as ground truth for detecting injected bogus data. Nodes can examine suspicious responses (e.g., contradictory responses from different peers) by querying the LBS server, and consequently, downgrading and reporting deviant responders. Actually, a node dissatisfied by other responders trades off its exposure to the LBS server for precise and genuine information, while at same time, it contributes to the common objective: decrease and balance exposure of nodes to the LBS server.

Thwarting Clogging Attacks: An internal attacker could post a large amount of queries to fetch cached information from its neighbors and drain their resources. Limiting the number of received peer queries signed under a same pseudonym and enforcing non-overlapped pseudonym lifetimes address this problem. It ensures each node has only one valid pseudonym at any point; thus when the quota (with respect to one receiver) of currently valid pseudonym has been consumed completely, it will not have any more valid pseudonyms to generate queries. However, flooding with bogus pseudonyms or messages that attached with bogus signatures could consume a lot of client resources for verification, while they are not avoidable. This is the same even if LBS-obtained information are signed. Attackers can still pass forged data to their neighbors to consume resources of benign nodes. As a remedy for our system, keys with relatively low security levels could be used. Considering the ephemeral nature of information transmitted in the system and short lifetimes of credentials; even if the keys are cracked, the attacker will no longer be interested in expired credentials by that time. The decision on key choice will be made based on cryptographic benchmarks (see Sec. 5).

Exposure to the Security Infrastructure and Collusion with the LBS: Though authorities we introduce are designed in a manner that protects the nodes from being traced, they could be honest-but-curious. However, any of the honest-but-curious LTCAs or PCAs cannot trace a user's actions (based on an eavesdropped transcript) - we refer to the analysis in [19]. Moreover, if the the LBS server authenticated nodes with pseudonyms, its collusion with the LTCA

would not reveal any information; collusion with the PCA would only reveal the batch of pseudonyms obtained with the one presented by the LBS server but not with past ones issued to the same node under a different ticket. Only the unlikely collusion of all three, the LBS server, the LTCA and the PCA, would expose the user.

5 Performance Evaluation

In this section, we demonstrate the practicality and applicability of our scheme. We show performance evaluation results for basic operations in our system with off-the-shelf components and popular platforms, i.e., RSA and ECDSA for public key cryptography and Android smartphone as user device. We find that a smartphone can easily handle high query rates from its neighbors, especially by using RSA keys with relatively low security levels.

Table 2. Processing delay of cryptographic operations

Key Type	***Security Level*** (bits)	***Generation*** (ms)	***Sign*** (ms)	***Verify*** (ms)	***Signature Size*** (bytes)
RSA-1024	80	400.86	4.63	0.78	128
RSA-2048	112	2104.59	21.18	1.21	256
ECDSA-192	96	214.65	210.01	286.44	56
ECDSA-224	112	251.66	251.91	345.95	63

The most frequent and time consuming operations are signature generation and verification.[2] Encryption of P2P communication would incur also key establishment cost, thus some additional public key encryption, whose processing delay is on the same order of magnitude of signature verification delay. However, we do not explicitly consider it here, as it is optional. Table 2 shows processing delays for cryptographic operations on a *Sony Xperia Ultra Z* smartphone, which has a *Quad-core 2.2 GHz Krait 400* CPU. We choose RSA and ECDSA algorithms, commonly used for public key cryptography. Note that ECDSA is standard in other applications, notably Vehicular Ad-hoc Networks (VANETs) [7], due to its low key generation and signing delays and short signature sizes. However, the Spongy Castle library [3], the only library available for Android supporting ECDSA, is inefficient. We found that RSA key generation takes more time than ECDSA, but sign/verify operations take much less time than ECDSA. Actually, all the execution delays of ECDSA are abnormally high (compared to those in other libraries for other platforms). By checking the system logs of

[2] Compared to other domains [15,19], which need frequent pseudonym changes to ensure unlinkability of messages transmitted in high rate, a relatively longer pseudonym lifetime is acceptable in our scheme, as one can expect relatively lower message rates. Thus, the performance is less affected by key generation operations.

the smartphone, we found that for each cryptographic operation of ECDSA the application needs to free the heap 2 or 3 times. As a result, it increases significantly cryptographic latencies, due to the limited heap size of each application in Android and the high spatial overhead of ECDSA operations. Due to abnormalities of ECDSA operations in Android, RSA is preferred for our scheme.

Table 3. Processing overhead for different operations

Operation	*Processing Overhead*
Message verification with cached pseudonym	Message Verification
Message verification with non-cached pseudonym	Pseudonym Verification, Message Verification
Query generation	Message Signing
Response generation	Database Query, Message Signing

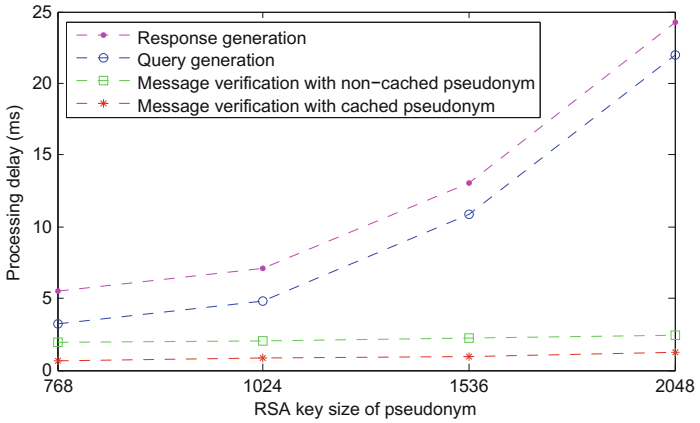

Fig. 3. Processing delay under pseudonyms with different RSA key sizes, assuming an RSA-2048 certificate of the PCA

Table 3 shows processing overhead for different operations and Fig. 3 shows processing delays on the smartphone. Consider, for example, mobile phone user density in Spanish cities [21]: Barcelona, the most densely populated in terms of mobile phone users in Spain, has around 3000 mobile phone users per km^2. Assuming Wi-Fi radio range of $100m$, there are around 100 peers within range. Assume all peers (e.g., in a landmark site or in the city center) need to query with a query rate per user equal to 1 $query/min$: this implies that a node would receive approximately 1.7 $queries/sec$. From Fig. 3, we can see that verifying

a query, even with non-cached pseudonyms, would incur processing overhead for less than 3 *msec*, thus more than 300 *queries/sec* could be verified. Peer response generation delay is highly dependent on local cache implementation and size of the cache. It includes a searching process and a signing operation. In our experiment, we use SQLite database. We assume 50 pieces of POIs are stored in the cache and 5 pieces of them matches each query. From Fig. 3, we see it takes approximately 7 *msec* to generate a peer response with RSA-1024 key. This implies that a node could, if able and needing to, respond to 1.7 *queries/sec* (the query rate we calculated earlier).

Based on availability of relevant information in the cache, only a part of them could be served; moreover, the actual latency could be significantly lower thanks to optimizations we proposed in Sec. 3. More basically, a node can "decide," based on, e.g., current CPU usage and battery amount remaining, whether or not to serve peer queries. A protocol-selectable parameter, the maximum rate at which to respond, can further reduce the overhead.

Peer query and response sizes are application and implementation dependent. In our experiments, we assume 25 byte queries. By encoding public keys and signatures into Base64 format and encapsulating into JSON format; the total size of a peer query, with an RSA-1024 pseudonym (public key and signature from the PCA) attached, is 980 bytes. The size of a peer response depends on how many information pieces it contains; while the signature and attached pseudonym sizes are same as those for a query. With most smartphones supporting IEEE 802.11 b/g/n with throughput between 11 and 300 Mbps, the above mentioned query rate would not incur heavy communication overhead. Clearly, receiving queries would affect posting own queries, as responses are received over the same channel. But, as mentioned above, beyond optimizations, a node can always stop serving queries beyond a threshold, to ensure it can obtain own needed information.

6 Conclusion

We presented a decentralized secure and privacy protection scheme for LBSs. We leverage the concept of information sharing in P2P systems for POI information sharing, and further secure it in a privacy-preserving manner with pseudonym-based authentication. Through security and privacy analysis and performance evaluation, we show a system with high resiliency to different attacks and high practicality for the deployment. Our scheme can be extended in terms of optimizations we proposed. In our evaluation, we assume mobile nodes are evenly distributed. However, efficiency of our scheme in flash crowds needs to be evaluated with large scale simulation to show how peer queries for varying types of POI information can be handled by load balancing among the nodes in the crowds. Moreover, with simulation, we can quantify user privacy and determine optimal parameters for the optimizations we proposed. An incentive scheme and cross-checking mechanism can be integrated to promote user participation and improve attack resiliency.

References

1. Google maps api. https://developers.google.com/maps/
2. Lte direct. https://www.qualcomm.com/invention/technologies/lte/direct
3. The Spongy Castle Cryptography APIs. https://rtyley.github.io/spongycastle/
4. Uber api. https://developer.uber.com/
5. Wi-fi direct. https://rtyley.github.io/spongycastle/
6. PRIME Framework Version. 3 (2008). https://www.prime-project.eu/prime_products/reports/fmwk/
7. IEEE Standard for Wireless Access in Vehicular Environments Security Services for Applications and Management Messages. IEEE Std 1609.2-2013 (2013)
8. Barkhuus, L., Dey, A.K.: Location-based services for mobile telephony: a study of users' privacy concerns. In: INTERACT, Cape Town, South Africa, September 2003
9. Calandriello, G., Papadimitratos, P., Hubaux, J.-P., Lioy, A.: Efficient and robust pseudonymous authentication in vanet. In: ACM VANET, Montreal, Canada, September 2007
10. Calandriello, G., Papadimitratos, P., Hubaux, J.-P., Lioy, A.: On the performance of secure vehicular communication systems. In: IEEE TDSC (2011)
11. Chow, C.-Y., Mokbel, M.F., Liu, X.: A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: ACM GIS, New York, NY, November 2006
12. Cutillo, L.A., Molva, R., Strufe, T.: Privacy preserving social networking through decentralization. In: IEEE/IFIP WONS, Snowbird, Utah, February 2009
13. Gedik, B., Liu, L.: Protecting location privacy with personalized k-anonymity: Architecture and algorithms. IEEE Transactions on Mobile Computing, January 2008
14. Ghinita, G., Kalnis, P., Skiadopoulos, S.: Mobihide: a mobilea peer-to-peer system for anonymous location-based queries. In: SSTD, Boston, MA, July 2007
15. Gisdakis, S., Giannetsos, T., Papadimitratos, P.: Sppear: security & privacy-preserving architecture for participatory-sensing applications. In: ACM WiSec, Oxford, UK, July 2014
16. Gisdakis, S., Laganà, M., Giannetsos, T., Papadimitratos, P.: Serosa: Service oriented security architecture for vehicular communications. In: IEEE VNC, Boston, MA, December 2013
17. Han, L., Nath, B., Iftode, L., Muthukrishnan, S.: Social butterfly: Social caches for distributed social networks. In: PASSAT, Boston, MA, October 2011
18. Johnson, M., McGuire, D., Willey, N.: The evolution of the peer-to-peer file sharing industry and the security risks for users. In: HICSS, Waikoloa, Big Island, Hawaii, January 2008
19. Khodaei, M., Jin, H., Papadimitratos, P.: Towards deploying a scalable & robust vehicular identity and credential management infrastructure. In: IEEE VNC, Paderborn, Germany, December 2014
20. Kwok, S.H., Lang, K.R., Tam, K.Y.: Peer-to-peer technology business and service models: risks and opportunities. Electronic Markets (2002)
21. Louail, T., Lenormand, M., Cantu Ros, O.G., Picornell, M., Herranz, R., Frias-Martinez, E., Ramasco, J.J., Barthelemy, M.: From mobile phone data to the spatial structure of cities. Scientific Reports, June 2014
22. Martucci, L.A., Kohlweiss, M., Andersson, C., Panchenko, A.: Self-certified sybil-free pseudonyms. In: ACM WiSec, Alexandria, VA, April 2008

23. Mascetti, S., Bettini, C., Freni, D., Wang, X.S.: Spatial generalisation algorithms for lbs privacy preservation. Journal of Location Based Services (2007)
24. Mezzour, G., Perrig, A., Gligor, V., Papadimitratos, P.: Privacy-preserving relationship path discovery in social networks. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 189–208. Springer, Heidelberg (2009)
25. Mokbel, M.F., Chow, C.-Y., Aref, W.G.: The new casper: query processing for location services without compromising privacy. In: Proceedings of the 32nd International Conference on Very large Data Bases, Seoul, Korea, September 2006
26. Myles, G., Friday, A., Davies, N.: Preserving privacy in environments with location-based applications. IEEE Pervasive Computing (2003)
27. Papadimitratos, P., Calandriello, G., Lioy, A., Hubaux, J.-P.: Impact of vehicular communication security on transportation safety. In: IEEE INFOCOM MOVE, Phoenix, AZ, April 2008
28. Sampigethaya, K., Li, M., Huang, L., Poovendran, R.: Amoeba: Robust location privacy scheme for vanet. IEEE JSAC (2007)
29. Shokri, R., Theodorakopoulos, G., Papadimitratos, P., Kazemi, E., Hubaux, J.-P.: Hiding in the mobile crowd: Location privacy through collaboration. IEEE TDSC (2014)
30. Zhou, L., Zhang, L., McSherry, F., Immorlica, N., Costa, M., Chien, S.: A first look at peer-to-peer worms: threats and defenses. In: Proceedings of the 4th International Conference on Peer-to-Peer Systems, Konstanz, Germany, August 2005