PEPPeR: A querier's Privacy Enhancing Protocol for PaRticipatory sensing

Tassos Dimitriou[1], Ioannis Krontiris[2], and Ahmad Sabouri[2]

[1] Athens Information Technology, 19.5 km Markopoulo Ave., 19002, Peania, Greece
tdim@ait.edu.gr
[2] Goethe University Frankfurt, Chair of Mobile Business & Multilateral Security, Grueneburgplatz 1, 60323 Frankfurt, Germany
{ioannis.krontiris,ahmad.sabouri}@m-chair.net

Abstract. In this work we study the problem of querier privacy in the Participatory Sensing domain. While prior work has attempted to protect the privacy of people contributing sensing data from their mobile phones, little or no work has focused on the problem of *querier privacy*. Motivated by a novel communication model in which clients may directly query participatory sensing networks operated by potentially untrusted adversaries, we propose PEPPeR, a protocol for privacy-preserving access control in participatory sensing applications that focuses on the privacy of the querier. Contrary to past solutions, PEPPeR enables queriers to have access to the data provided by participating users without placing any trust in third parties or reducing the scope of queries. Additionally, our approach naturally extends the traditional pull/push models of participatory sensing and integrates nicely with mobile social networks, a new breed of sensing which combines mobile sensor devices with personal sensing environments.

1 Introduction

The increasing availability of sensors on today's smartphones and other everyday devices, carried around by millions of people, has already opened up new possibilities for gathering sensed information from our environment. Currently researchers experiment with these possibilities and share the vision of a sensor data-sharing infrastructure, where people and their mobile devices provide their collected data streams in accessible ways to third parties interested in integrating and remixing the data for a specific purpose. A popular example is a noise mapping application which generates collective noise maps by aggregating measurements provided by the mobile phones of volunteers [1]. In other scenarios, people may monitor air pollution [2], road and traffic conditions [3], etc.

What is common in all the above applications, is that sensing data are proactively collected by users in a centralized server, where they are aggregated, processed and represented through various interfaces (e.g. statistical data on a map) or remain available for third parties to query and select data of interest. While this serves a specific class of applications, another approach is to enable the collection of sensing information only where and when it is needed. In that sense,

someone interested to obtain information from a specific location and within a specific context, posts a task or query to the platform and the mobile nodes satisfying the conditions react by taking the measurements and sending back a response.

In this paradigm, the question becomes how these sensing tasks can be distributed to the mobile phones. Two models have been studied in the bibliography so far. In the *push* model [4], the participating nodes (i.e. mobile phones) register with the server and the server pushes only matching tasks, based on appropriate context (e.g. location). In the *pull* model [5], sensing tasks are posted on a server, which participating nodes contact for possible download and execution of tasks.

In this work, we consider an extension to the participatory sensing model, in which a user may decide to query a participating node *directly*, without the mediation of the service provider. Contrary to the traditional model, where reporting of data is performed through a trusted report server (e.g. Anonysense), this method maybe the only feasible solution when it is impossible or even undesirable to maintain a stable connection between a mobile user and the service provider. It is also necessitated by the growing requirement for protecting users' data access privacy; a user may want to keep confidential whether (and when) she accessed the sensed data, the data types she was interested in, or from which nodes she obtained the data, as the disclosure of such information may be used against her interest. For example, information about the noise level in a particular neighborhood may leak information to the nearby home agency about Alice's desire to purchase a home in the area.

Additionally, the sensed data maybe of interest to numerous users from both the public and private sectors, ranging from individuals to business companies that may be competing of each other. To capitalize on the data, application owners might be using appropriate incentive mechanisms [6] to enable broad user participation. It is reasonable, therefore, to expect that queriers are willing to pay appropriate fees in order to obtain measurements of interest.

Our method seems also to be a perfect fit for a new breed of sensing, triggered by the ever increasing number of online Social Network users and the imminent integration of mobile sensor devices into personal sensing environments [7, 8]. Such *mobile social networks* are distributed systems that combine mobile, social, and sensing components, trying to create a contextual picture surrounding a user or group of users in order to enable new applications and services based on this context. In this new model of P2P sensing, it is thus straightforward to expect that users may ask other peers directly for sensed data. In fact this may be the only way to hide a querier's interest on certain data since, with any centralized service, a user's interests and queries may leak information about the user itself. Thus the goal of this work is to describe a generic mechanism in which queriers may ask directly for sensed data.

Contribution: In this work, we present PEPPeR (Privacy Enhancing Protocol for PaRticipatory sensing), that aims to protect the privacy of queriers, by letting them obtain *tokens* from the service provider (or application owner) in order to have access to the data provided by participating users (custodians of mobile

phones). The difference from prior work is in the decentralized character of our approach. The querier may decide to spend the token with *any* producer (mobile phone user) directly, who first has to validate the token and then provide the querier with the proper amount of requested data. Using appropriate cryptographic mechanisms, we show how the validity of the token can be verified by any mobile node, without, however, leaking the identity of the querier to the node or to the application owner. At the heart of PEPPeR lies a mechanism to detect double-spent tokens, which involves the use of a witnesses service, a simple scheme that can attest to the validity of the token or provide proof for token reuse.

2 Related Work

Privacy preserving access control has been studied extensively in traditional sensor networks ([9, 10, 11, 12]). While the work in [9] is a centralized approach in which users acquire data through one or more base stations, the rest take advantage of the inherent redundancy in sensor networks to either store and discover double-spent tokens ([10]) or forward queries in a privacy-preserving manner ([11, 12]). However, all proposed solutions are based on the multi-hop routing communication model of sensor networks in which nodes operate both as sensing devices and routers of information, a model that cannot be adopted in participatory sensing.

In the participatory sensing domain, various *centralized* solutions for distributing tasks or queries to sensor nodes have been proposed. In PRISM [4], participating nodes (i.e. mobile phones) register with the server and the server tracks the nodes and pushes only matching tasks to them, based on their context (e.g. location). For example, Alice may be assigned the task "measure temperature in area X", when she is entering this area. However, this solution does not consider privacy for any of the involved entities, queriers or mobile nodes.

A solution that offers a privacy-friendly way of task distribution is AnonySense [5]. Sensing tasks are posted on a server and the participating nodes download the tasks and match them to their context to decide which one to execute. This approach has the advantage that the nodes do not reveal anything about their context to the service provider, in order to receive the sensing task. Still, AnonySense does not consider privacy for the entities posting the tasks.

Recently, PEPSI [13] was suggested as a system designed with the privacy of the queriers in mind, queriers being entities external to the platform, who are interested in some specific sensing information. PEPSI is based on a *centralized* solution and to protect the privacy of the queriers, it introduces a Registration Authority, a trusted third party which collects queries from the queriers and provides back the corresponding cryptographic material. In that sense, the queries reach the platform in an encrypted form. However, the problem is shifted to the Registration Authority, where essentially all queries are known in advance, along with the identities of the queriers, leading to the assumption that this entity must be trusted.

PEPSI is designed to work in a different setting than the one suggested in this paper. Due to the cryptographic mechanism employed by the protocol, queries have to be composed out of specific, *predefined* keywords. At the same time, mobile nodes *proactively* sense and report data, including the same keywords. At the end, following a centralized communication paradigm, queries and sensing data are both collected by the service provider, where they are matched against each other.

3 Network model and security goals

The main goal of PEPPeR is to protect the privacy of the parties posting sensing queries or tasks to mobile nodes. To do this efficiently, we *decouple* the process of discovering the nodes that are able to answer a query from the access control mechanisms utilized in the system to contact these nodes. This is due to the fact that each of these processes concerns a different entity in the system. From a platform provider's point of view, no matter how a querier finds her desired mobile nodes, it is critical to stop her from making benefit of the platform's services without getting permission from the provider. How this permission can be obtained depends on the business model of the platform provider. For example, it could be that the querier has to pay for each "sensing quantum". On the other hand, from the querier's perspective, no matter what access control mechanism is employed in the system, she should be able to communicate with any mobile node in a privacy-respecting way, without disclosing her interest to the provider.

Figure 1 depicts this setting and shows the focus of PEPPeR. The role of the service provider is limited only to providing the means for queriers to contact the mobile nodes. That is, mobile nodes first register with the platform and through a privacy-respecting mechanism (agnostic to this paper) the service provider is able to offer the querier the contact details (e.g. in anonymous or pseudonymous way) of the mobile nodes currently in the geographic area of interest. For example, this could be achieved through a distributed directory service for looking up mobile nodes.

In addition, the querier anonymously purchases a cryptographic token from the platform, which enables her to directly contact the mobile nodes and forward the query to them. Due to the cryptographic properties of this token, there is no need to restrict the scope of the queries or introduce a trusted third party, as in past works.

The privacy of the mobile nodes still needs to be protected and our work at no means hinders protocols built for this purpose. However this work is *not* about how to a) support the discovery of sensor information, b) allow for sensor subscription, and c) facilitate sensor task placement. These topics have been covered, to some extend, by prior work [4, 5, 7] and are not considered here. Here we address the topic of querier privacy that can be of value to the various denominations of participatory sensing networks.

Given the above setting, the following security and privacy goals are required to be fulfilled by the access control mechanism:

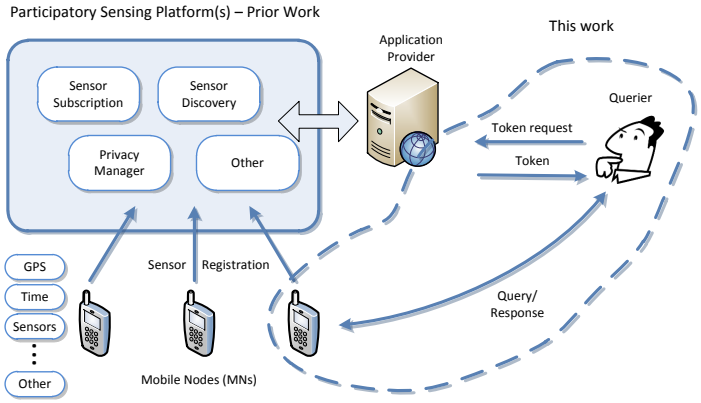

Fig. 1. Query privacy in the context of participatory sensing.

- *Untraceability*: In order to preserve the querier's privacy, the access control mechanism should not leak any information about the identity of the authorized querier, when she contacts the service provider.
- *Unlinkability*: A single querier can use the platform as many times as she is allowed to, in order to access and query mobile nodes. It should not be possible to link multiple accesses back to the same quierier, as long as she is eligible and has the access credentials to use the service.
- *Misuse Resistance*: The access control mechanism should prohibit unauthorized users from using the system. In addition, it should also prevent queriers from misusing the service, according to the conditions agreed upon. For example, if the querier acquired credentials for accessing and querying the nodes only one-time, then she should not be able to reuse them for a second time.
- *Misuse Provability*: The access control mechanism should support providing evidence, in case of a misuse. For example, if the querier is permitted to use the platform only once and makes a second attempt, the access control mechanism should be able to provide convincing proof of the fact that this is the second time access. This will enable mobile nodes to deny service, however the privacy of the querier should still be respected.

4 Querier Privacy Protocol

In this section, we present the protocol behind PEPPeR that satisfies the above requirements and offers a privacy-respecting access control for querying mobile nodes of interest. Figure 2 shows a high-level overview of the steps, which we will describe in detail in the following sections.

A querier $\mathcal{Q}$ wishes to access the participatory sensing network for data. She first contacts the application owner $\mathcal{S}$ and retrieves a token $\mathcal{T}$ which can spent

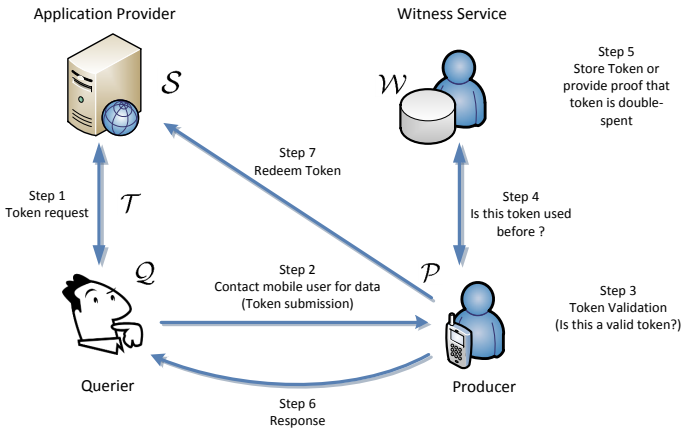

Fig. 2. The protocol steps followed by PEPPeR.

with any producer $\mathcal{P}$ (mobile phone user) of sensed data (Section 4.1). The token reveals no information about either $\mathcal{Q}$ or the desire of $\mathcal{Q}$ to spend it with any specific $\mathcal{P}$. Once $\mathcal{P}$ retrieves the token, it first has to verify its validity and then test whether the token is an attempt of doublespending (Section 4.2). For that reason it contacts an appropriate witness service $\mathcal{W}$ which can either ascertain the coin's freshness or provide proof that the coin has been doublespent (Section 4.2). This service can have the form of a bulletin board (a simple repository of used tokens) in which nodes may consult for reused tokens. The evidence provided by $\mathcal{W}$ again reveals nothing about the identity of the querier $\mathcal{Q}$, only the fact that the coin has been used before. Finally, the producer $\mathcal{P}$ can redeem the token for credit or additional services from $\mathcal{S}$. These steps are shown in Figure 2 and are described in more detail below.

4.1 Purchasing a token

To make a request for sensor data, a querier $\mathcal{Q}$ must first obtain a valid token from the application provider. In order to make the token untraceable and protect the privacy of $\mathcal{Q}$, the token will not be associated with a particular querier, however it should contain such information as date, expiration date and amount of data to be retrieved as well as the signature of the application provider $\mathcal{S}$ on it. This information is the *common part* of the token which is necessary in order for the mobile user (supplier of sensed data) to be able to provide a commensurate amount of information and perform an initial test on the validity of the token (i.e. check expiration date and signature of $\mathcal{S}$).

As the token can be purchased by $\mathcal{Q}$ through an anonymous channel using (say) a gift card or a trusted third party, the common part of the coin, denoted by $\langle CommonInfo \rangle$, leaks no information about the identity of $\mathcal{Q}$. We need, however, to provide a mechanism for detecting double spending, and this mechanism

has to be associated with *identifying* information, denoted by $\langle UniqueInfo \rangle$, supplied by the querier $\mathcal{Q}$. This uniquely identifying piece of information is in no way related to the identity of $\mathcal{Q}$; its purpose is to deter double spending but it could potentially be used by $\mathcal{S}$ to trace the token and this is why it has to be *blinded* before it is signed by $\mathcal{S}$.

Blind signatures provide perfect confidentiality to a message and signature pair, however the signer must be assured that the message contains valid information in order to prevent abuse [14]. This valid information corresponds to the $\langle CommonInfo \rangle$ part. An elegant solution to cope with the necessity of checking the correctness of messages contained in blind signatures is to use the so-called *partially* blind signatures introduced in [15]. In our scenario, the provider $\mathcal{S}$ will sign messages made of two parts; the $\langle CommonInfo \rangle$ part which is visible by the signer and the $\langle UniqueInfo \rangle$ part containing identifying information to detect double spending which is invisible and blindly signed. An instantiation of this process is described below.

Let k be the security parameter. Let p' and q' be two large primes of size $k/2$ such that $p = 2p' + 1$ and $q = 2q' + 1$ are also prime. Let $N = pq$ be an RSA modulus, (e, N) be the public key of the application provider $\mathcal{S}$ and (d, N) its corresponding private key in the RSA key generation process. Following the general method outlined in [15], one way to include common information to any message m is to *embed* it in the signer's key and generate new, per message signing keys (in our case per token keys). So, let $h()$ be a secure cryptographic hash function such as SHA1. The new public (and private) exponent e_T (resp. d_T) for token $\mathcal{T}$, generated from e, is obtained as follows:

Algorithm 1
Creating partial signature keys from (e, N)

$$h_1 \leftarrow h(e, \langle CommonInfo \rangle)$$
$$e_T \leftarrow h_1 \parallel h(h_1) \parallel 00000001$$
$$d_T \leftarrow 1/e_T \mod \phi(N)$$

The existence of the inverse, signing keys d_T is due to the following facts. Since $N = (2p' + 1)(2q' + 1)$, we have $\phi(N) = 4p'q'$. Both p' and q' are large primes (ca. 511 bits) while the concatenation $h_1 \parallel h(h_1)$ is 320 bits long. In addition, e_T is enforced to be an odd integer, hence it is not divisible by any of $\{2, 4, p', q'\}$. Thus e_T is relatively prime to $\phi(N)$ and has an inverse d_T, which can be computed by $\mathcal{S}$. Additionally, a secure RSA setting requires the decryption key to be larger than $\sqrt{N}$ [16]. This is guaranteed by making e_T around 320 bits long which ensures that d_T will be longer than 512 bits in case of 1024-bit modulus. (*The proof of resistance to chosen message attacks of signatures generated this way is shown in the Appendix.*)

A querier $\mathcal{Q}$ can ask the service provider to provide a token by using the procedure shown in Protocol 1. Recall that $\langle UniqueInfo \rangle$ is the part that has to be blinded and contains identifying information to prevent double spending

(the exact format of this part will become clear in the next section). Let also r be a random number in Z_N and $m = h(\langle UniqueInfo \rangle)$.

Protocol 1
Obtaining a blind signature from $\mathcal{S}$

1. $\mathcal{Q}$ sends $m^* = mr^{e_T} \mod N$ to $\mathcal{S}$ by evaluating the public key e_T from public information available (e.g. e and $\langle CommonInfo \rangle$).
2. The application owner $\mathcal{S}$ returns the signature $\sigma^* = (m^*)^{d_T} \mod N$ to $\mathcal{Q}$.
3. $\mathcal{Q}$ computes $\sigma = r^{-1}\sigma^* \mod N$, which is the application owner's signature on $h(\langle UniqueInfo \rangle)$.

Due to the blinding factor r, the network owner cannot derive m and σ from m^*. In other words, given $\langle m, \sigma \rangle$, the network owner cannot link it to $\mathcal{Q}$. Additionally, since the $\langle CommonInfo \rangle$ part is clear to the querier and is negotiated at the beginning of (or before) the protocol, $\mathcal{S}$ cannot include any information in it and hence trace the querier. An implicit assumption here is that the parameters in the common part do not contain any one-time elements that may help distinguish the transaction and narrow down the search such as for example a strange combination of date and amount of data, and so on. If this is a concern, users may depend on a trusted third party to purchase tokens or use alternative schemes such as anonymous gift cards. Once the signature σ is retrieved, the final token becomes $\mathcal{T} = \langle CommonInfo, UniqueInfo, m, \sigma \rangle$.

4.2 Spending and Redeeming a Token

Let $\mathcal{P}$ be the producer of data (mobile phone user) that $\mathcal{Q}$ has decided to spend the token to. In order to provide the required amount of sensed data, $\mathcal{Q}$ has to be able to tell if the token $\mathcal{T}$ has been used before. For that reason, $\mathcal{P}$ will contact the *witness* service $\mathcal{W}$ in order to attest on the validity of $\mathcal{T}$ or provide proof that the token has been used before. Crucial to the above is the structure of the $\langle UniqueInfo \rangle$ existing in $\mathcal{T}$. This element will allow $\mathcal{P}$ (as well as $\mathcal{W}$) to provide the necessary evidence for the token's validity.

Let P and Q be primes such that $Q|P-1$ and g a generator of order Q in the group Z_P^*. Typically, P and Q will have length 1024 bits and 160 bits, respectively. The querier $\mathcal{Q}$ will select two secret values $s, r \in Z_P$ and compute $v = g^{-s} \mod P$ and $x = g^r \mod P$. The $\langle UniqueInfo \rangle$ element will consist of the two values v and x, which will be signed by $\mathcal{S}$ as explained in the previous section.

When the querier wishes to spend the token $\mathcal{T}$, it will first have to demonstrate the *validity* of the coin by proving knowledge of the two secret values s, r using appropriate zero-knowledge proofs. This protocol is based on the identification scheme of Schnorr [17] (see also Okamoto [18] for a generalization and [19] for an instantiation on the e-cash setting) and can be made non-interactive

by making the challenge of the verifier equal to the hash value of the token and committed protocol parameters. The interaction between $\mathcal{Q}$ and $\mathcal{P}$ is shown below.

Protocol 2
Checking the validity of a token $\mathcal{T}$

1. $\mathcal{Q}$ sends $\mathcal{P}$ $\langle \mathcal{T}, y, date/time \rangle$, where $y = r + es \mod Q$ and $e = h(\langle \mathcal{T}, date/time \rangle)$.
2. $\mathcal{P}$ verifies the signature of $\mathcal{S}$ on the token and checks that $x = g^y v^e \mod P$.

If the tests in Step 2 of Protocol 2 succeed, $\mathcal{P}$ considers the token *valid*. However, $\mathcal{P}$ still has to determine whether the token is *fresh* or an attempt of doublespending. For that reason it has to contact the witness service $\mathcal{W}$ that can attest on the freshness of the token. The interaction between $\mathcal{P}$ and a $\mathcal{W}$ is shown below:

Protocol 3
Checking for doublespending

1. $\mathcal{P}$ sends $\mathcal{W}$ the transcript of the interaction with $\mathcal{Q}$, i.e. $\langle \mathcal{T}, y, date/time \rangle$.
2. $\mathcal{W}$ verifies the signature of $\mathcal{S}$ on $\mathcal{T}$ and checks that $x = g^y v^e \mod P$. Then, based on the expiration date of the token, searches its records for a token containing the same $\langle UniqueInfo \rangle$ element. If no match is found, the token is considered fresh and $\mathcal{W}$ records the values $\langle \mathcal{T}, y, date/time \rangle$.
   If a match is found then $\mathcal{W}$ returns evidence that the token has been used before. This evidence has the form of the secret values s, r selected by $\mathcal{Q}$ in forming the values v and x contained in the $\langle UniqueInfo \rangle$ element.

To see why $\mathcal{W}$ can provide evidence[1] that a token has been used before, notice that in this case it will have two transcripts $\langle \mathcal{T}, y, date/time \rangle$ and $\langle \mathcal{T}, y', date'/time' \rangle$ such that $x = g^y v^e \mod P$ and $x = g^{y'} v^{e'} \mod P$. $\mathcal{W}$ can then compute $s = (y - y')/(e - e') \mod Q$ by solving

$$y = r + es \mod Q,$$
$$y' = r + e's \mod Q.$$

In a similar manner $\mathcal{W}$ can obtain r. Notice that these values are not connected with the ID of $\mathcal{Q}$, hence they are not used in identifying $\mathcal{Q}$. They are only used to provide evidence that a token has been double-spent. Hence the privacy of $\mathcal{Q}$ is maintained even in this case.

[1] Alternatively, $\mathcal{W}$ returns the previous transcript $\langle \mathcal{T}, y', date'/time' \rangle$ and $\mathcal{P}$ can check itself if the coin is not used before.

Once $\mathcal{P}$ is convinced that the token is both valid and fresh, it can provide the data requested by $\mathcal{Q}$. Then it can go on redeeming the token in exchange for other services or credit provided by the application provider. A slight complication might arise, however, if $\mathcal{P}$ redeems the token but *denies* to offer $\mathcal{Q}$ the data she "paid" for. A solution to this problem is for $\mathcal{Q}$ to ask for a signed commitment from $\mathcal{P}$ that it will provide the data, assuming the token is both valid and fresh. This exchange can have the form shown below and should be executed before Step 1 of Protocol 2.

$$\mathcal{Q} \rightarrow \mathcal{P} : h(\mathcal{T}), N_Q$$
$$\mathcal{P} \rightarrow \mathcal{Q} : Sig_{\mathcal{P}}(h(\mathcal{T}), N_Q, \text{"Commit to Serve"})$$

The signature on the second message is a commitment that $\mathcal{P}$ agrees to serve the token that hashes to $h(\mathcal{T})$. Once $\mathcal{Q}$ verifies the signature, it proceeds with the remaining steps of Protocol 2.

5 Security Analysis

In this section, we demonstrate the security properties of the protocol which match the goals set forth in Section 3. Some are new (Appendix) and some are inherited by the use of appropriate cryptographic protocols used throughout.

Token Unforgeability: The process described in Section 4.1 (Algorithm 1) of embedding common information in the signer's key has been proven secure in the Appendix. This is another contribution of this paper since in the work of [15], no concrete method has been presented. Thus, a token obtained during token purchase cannot be altered by a querier without affecting the validity of the signature. In a similar manner, no querier can create a valid token without knowledge of the secret signing keys. Security is based on the hash-then-sign paradigm of RSA and the strength of hash functions acting as random oracles.

Token Untraceability: Due to blindness, the only information learnt by $\mathcal{S}$ is the $\langle CommonInfo \rangle$ element. Going back to Protocol 1, $\mathcal{S}$ cannot derive $h(\langle UniqueInfo \rangle)$ and σ from m^*, without knowing the blinding factor r. In other words, given the pair $(h(\langle UniqueInfo \rangle), \sigma)$, the application owner cannot link it to a querier $\mathcal{Q}$. Thus when a token is redeemed by a producer $\mathcal{P}$, the application owner will not be able to tell which querier requested the token.

An implicit assumption here is that the $\langle CommonInfo \rangle$ element used to create the signing key in Algorithm 1 does not contain any information that can be used to identify this particular transaction. If this is the case, $\mathcal{Q}$ may simply deny to take part in the token generation process.

However, even if it's difficult to associate tokens with the identities of their holders, the application owner may still narrow down the owner of a particular token to the queriers who purchased tokens. This might be a concern if the number of token buyers is limited. In such a case, users may rely on trusted third parties to purchase tokens from $\mathcal{S}$.

Double Spending Prevention: Resilience to double spending is due to the construction of the $\langle UniqueInfo \rangle$ element. This element consists of the two values v and x which are equal to $v = g^{-s} \mod P$ and $x = g^{r} \mod P$, respectively. When a querier $\mathcal{Q}$ sends the value y to $\mathcal{P}$ (recall Protocol 2) essentially proves knowledge of the values r and s that make up v and x.

One strategy, therefore, that $\mathcal{Q}$ can use to double spend a token $\mathcal{T}$ is to come up with a different representation of the values v and x. But that's equivalent to computing discrete logarithms [19], which is considered an intractable problem. In a similar manner, no other querier will be able to double spent this token as it would have to prove the token's validity (Protocol 2) first. But that would also require knowledge of the values r and s, otherwise the NIZK protocol would fail. Thus, we conclude that a token cannot be used more than once.

Finally, it is clear from Protocol 3 that if $\mathcal{Q}$ tries to double spend a token $\mathcal{T}$, then a producer can recover the secret values making up v and x, thus providing evidence that a coin has been double-spent.

Querier's Privacy: PEPPeR preserves the querier's privacy and fulfils all the requirements of Section 3. There are several potential points where the querier's privacy can get compromised. We briefly discuss below, how our solution copes with these cases.

Token Purchase Whenever a financial transaction is involved in the scenario, there is a risk of linkability to the real identity of the buyer (in our case the querier) if the underlying payment methods are not privacy preserving. PEPPeR is independent from the credit transfer mechanism, so if it is done in a privacy friendly way, the whole operation will not leak any identifying information about the querier.

Mobile Node Lookup The access control mechanism is designed to abstract away the mobile node lookup process. As a result, the level of privacy achieved by the query delivery method in the platform will stay unaffected by PEPPeR. For example, a distributed directory service by the mobile nodes can elevate the privacy level significantly, compared to the existing query-task matching schemes by a centralized server that can be found frequently in the literature (e.g. [13]). Integrating PEPPeR to any such a privacy-friendly method will not have any negative impact and will not introduce any new risk to existing systems.

Token Spending When a querier wants to retrieve data from a Mobile Node and spend her access token, she cannot be identified or linked to the purchase phase. The cryptography used underneath enables us to break the connection between these two steps and prevent any correlation between purchase and spending phase. Therefore, even if the identity of the querier was somehow revealed while purchasing the token, she can ensure that the usage of the token will stay unlinkable to her. The other important aspect of PEPPeR is that misuse and double-spending cases can be detected and proved without a need for disclosing querier's identity information.

6 Conclusions and Future Work

In this paper, we have considered the problem of query privacy in people-centric sensing networks, in which queriers do not need to trust and rely on a service provider (or application owner) $\mathcal{S}$, in order to get access to the data produced by mobile users. We have described PEPPeR, a generic protocol that protects querier's privacy, by letting a querier $\mathcal{Q}$ obtain tokens from $\mathcal{S}$, which reveal nothing about either the identity of $\mathcal{Q}$ or its desire to spend the token with a *specific* supplier of sensed data. Using appropriate cryptographic mechanisms to ensure token validity, double-spending prevention, etc., PEPPeR does not restrict the scope of queries or introduce trusted third parties as in past solutions; the role of the service provider is limited only to providing the means for queriers to contact the mobile nodes, a service which can be agnostic to our protocol.

As part of a future work, we are planning to integrate our protocol with traditional sensing platforms, even social ones, in order to further demonstrate the applicability and viability of our approach. An interesting point of research would also be to eliminate entirely the witness service provided centrally for double-spent tokens and replace it with one by which mobile users *themselves* would be able to attest the validity of tokens. While such methods have been considered in traditional P2P systems (see for example the work in [20] in the e-cash setting and the references therein), these methods generally assume the presence of witnesses *at all times*. This would be difficult to assume in the case of participatory/social sensing, in which nodes are free to come and go as they like. Additionally, these nodes may become single points of failure or introduce collusion in the token spending process. Hence more robust mechanisms are needed to ensure the validity of such a (distributed) witness service in terms of persistence, consistency as well as scalability.

7 Acknowledgements

The first author would like to thank Angelos Kiayias for useful discussions. This work has been funded by the European Community's FP7 project SafeCity (Grant Agreement no: 285556).

References

1. Maisonneuve, N., Stevens, M., Ochab, B.: Participatory noise pollution monitoring using mobile phones. Information Policy **15** (2010) 51–71
2. Honicky, R., Brewer, E.A., Paulos, E., White, R.: N-smarts: networked suite of mobile atmospheric real-time sensors. In: Proceedings of the 2nd ACM SIGCOMM workshop on Networked systems for developing regions (NSDR '08). (2008) 25–30
3. Mohan, P., Padmanabhan, V.N., Ramjee, R.: Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08). (2008) 323–336

4. Das, T., Mohan, P., Padmanabhan, V.N., Ramjee, R., Sharma, A.: PRISM: platform for remote sensing using smartphones. In: Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys '10), San Francisco, California, USA (2010) 63–76
5. Shin, M., Cornelius, C., Peebles, D., Kapadia, A., Kotz, D., Triandopoulos, N.: AnonySense: A system for anonymous opportunistic sensing. Journal of Pervasive and Mobile Computing (2010)
6. Krontiris, I., Albers, A.: Monetary incentives in participatory sensing using multi-attributive auctions. International Journal of Parallel, Emergent and Distributed Systems (2012)
7. Beach, A., Gartrell, M., Xing, X., Han, R., Lv, Q., Mishra, S., Seada, K.: Fusing mobile, sensor, and social data to fully enable context-aware computing. In: Proceedings of the 11th Workshop on Mobile Computing Systems & Applications (HotMobile '10), Annapolis, Maryland (2010) 60–65
8. Krontiris, I., Freiling, F.C.: Integrating people-centric sensing with social networks: A privacy research agenda. In: Proceeding of the IEEE International Workshop on Security and Social Networking (SESOC). (2010)
9. Carbunar, B., Yu, Y., Shi, W., Pearce, M., Vasudevan, V.: Query privacy in wireless sensor networks. ACM Transactions on Sensor Networks **6**(2) (2010)
10. Zhang, R., Zhang, Y., Ren, K.: DP2AC: Distributed Privacy-Preserving Access Control in Sensor Networks. In: Proceeding of the 28th Conference on Computer Communications (INFOCOM '09). (2009)
11. De Cristofaro, E., Ding, X., Tsudik, G.: Privacy-preserving querying in sensor networks. In: Proceeding of the International Conference on Computer Communications and Networks (ICCCN '09). (2009)
12. Dimitriou, T., Sabouri, A.: Privacy preservation schemes for querying wireless sensor networks. In: Proceedings of the 7th IEEE PerCom International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS 2011). (2011) 178–183
13. De Cristofaro, E., Soriente, C.: Short paper: PEPSI – privacy-enhanced participatory sensing infrastructure. In: Proceedings of the fourth ACM conference on Wireless network security (WiSec '11). (2011) 23–28
14. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO. (1982) 199–203
15. Abe, M., Fujisaki, E.: How to date blind signatures. In: ASIACRYPT'96. (1996) 244–251
16. Boneh, D.: Twenty Years of Attacks on the RSA Cryptosystem. Notices of the American Mathematical Society (AMS) **46**(2) (1999) 203–213
17. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3) (1991) 161–174
18. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: CRYPTO. (1993) 31–53
19. Brands, S.: Untraceable off-line cash in wallets with observers (extended abstract). In: CRYPTO. (1993) 302–318
20. Osipkov, I., Vasserman, E.Y., Hopper, N., Kim, Y.: Combating double-spending using cooperative P2P systems. In: Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07). (2007)
21. Bellare, M., Rogaway, P.: The exact security of digital signatures-how to sign with RSA and rabin. In: EUROCRYPT. (1996) 399–416
22. Coron, J.S.: On the exact security of full domain hash. In: CRYPTO. (2000) 229–235

Appendix

In this section we prove the security of signatures against chosen message attacks when multiple signing keys are generated from the same public/private key pair (Algorithm 1). The reader is referred to [21, 22] for the various definitions.

More precisely, we define an extension to the Full Domain Hash Signature scheme in which the key generation algorithm on input 1^k generates various public/private key pairs (N, e_i, d_i), as in Algorithm 1, and the signing/verification algorithms have oracle access to a hash function $H_{FDH} : \{0,1\}^* \rightarrow Z_N^*$. We will now relate the security of this scheme to the security of solving the RSA problem (i.e. recovering a plaintext from an encrypted message).

Proof (high level) Let $\mathcal{F}$ be a forger that breaks the extended FDH. Using $\mathcal{F}$, we will build an inverter $\mathcal{I}$ that can be used to break RSA. The goal of $\mathcal{I}$ is to find $x = f^{-1}(y)$ for some key e_i and a random $y \in Z_N^*$, where f is the RSA exponentiation function.

The inverter starts by running $\mathcal{F}$. When $\mathcal{F}$ makes hash and signing queries, $\mathcal{I}$ answer those itself. In particular, when $\mathcal{F}$ makes a hash oracle query for M, the inverter increments a counter i, sets $M_i = M$ and picks a random $r_i \in Z_N^*$. It then returns $h_i = r_i^{e_1 e_2 \cdots e_k} \mod N$ with probability p and $h_i = y r_i^{e_1 e_2 \cdots e_k} \mod N$ with probability $1-p$. It also maintains all $r_i^{e_1 e_2 \cdots e_k / e_j}$ values, $1 \leq j \leq k$, for simulating the signing oracle.

Eventually, $\mathcal{F}$ halts and outputs a forgery (M, s) for some public key e_i (wlog. assume this is e_1). We also assume that $\mathcal{F}$ has requested the hash of M before. If not, $\mathcal{I}$ goes ahead and makes the hash query itself, so that in any case $M = M_i$ for some i.

Then, if $h_i = y r_i^{e_1 e_2 \cdots e_k} \mod N$ we have $s = h^{d_1} = y^{d_1} r_i^{e_2 \cdots e_k} \mod N$ and $\mathcal{I}$ outputs $y^{d_1} (= s / r_i^{e_2 \cdots e_k} \mod N)$ as the inverse for y. Thus the intractability of the RSA problem is reduced to the intractability of the extended FDH signing algorithm. Also by tuning the probability p, we can make the probability of forging a signature almost equally low as inverting RSA (details omitted due to space restrictions, refer to [21, 22] for similar arguments). This proves the correctness of the signature generation process.