2013 IEEE International Conference on Pervasive Computing and Communications (PerCom), San Diego (18--22 March 2013)

Providing Privacy-Aware Incentives for Mobile Sensing

Qinghua Li, Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
Email: {qxl118, gcao}@cse.psu.edu

***Abstract*—Mobile sensing exploits data contributed by mobile users (e.g., via their smart phones) to make sophisticated inferences about people and their surrounding and thus can be applied to environmental monitoring, traffic monitoring and healthcare. However, the large-scale deployment of mobile sensing applications is hindered by the lack of incentives for users to participate and the concerns on possible privacy leakage. Although incentive and privacy have been addressed separately in mobile sensing, it is still an open problem to address them simultaneously. In this paper, we propose two privacy-aware incentive schemes for mobile sensing to promote user participation. These schemes allow each mobile user to earn credits by contributing data without leaking which data it has contributed, and at the same time ensure that dishonest users cannot abuse the system to earn unlimited amount of credits. The first scheme considers scenarios where a trusted third party (TTP) is available. It relies on the TTP to protect user privacy, and thus has very low computation and storage cost at each mobile user. The second scheme removes the assumption of TTP and applies blind signature and commitment techniques to protect user privacy.**

I. Introduction

Mobile devices such as smart phones are gaining an ever-increasing popularity. These devices are equipped with various sensors such as camera, microphone, accelerometer, GPS, etc. Mobile sensing exploits the data contributed by mobile users (via the mobile devices they carry) to make sophisticated inferences about people (e.g., health, activity, social event) and their surrounding (e.g., noise, pollution, weather), and thus can help improve people's health as well as life. Applications of mobile sensing include traffic monitoring [1], environmental monitoring [2], healthcare [3], [4], etc.

Although the data contributed by mobile users is very useful, currently most mobile sensing applications rely on a small number of volunteers to contribute data, and hence the amount of collected data is limited. There are two factors that hinder the large-scale deployment of mobile sensing applications. First, there is a lack of incentives for users to participate in mobile sensing. To participate, a user has to trigger her sensors to measure data (e.g., to obtain GPS locations), which may consume much power of her smart phone. Also, the user needs to upload data to a server which may consume much of her 3G data quota (e.g., when the data is photos). Moreover, the user may have to move to a specific location to sense the required data. Considering these efforts and resources required from the user, an incentive scheme is strongly desired for mobile sensing applications to proliferate. Second, in many cases the data from individual user is privacy-sensitive. For instance, to monitor the propagation of a new flu, a server will collect information on who have been infected by this flu. However, a patient may not want to provide such information if she is not sure whether the information will be abused by the server.

Several schemes [5]–[7] have been proposed to protect user privacy in mobile sensing, but they do not provide incentives for users to participate. A recent work [8] designs incentives based on gaming and auction theories, but it does not consider privacy. Thus, it is still an open problem to provide incentives for mobile sensing without privacy leakage.

In this paper, we address the problem of providing privacy-aware incentives for mobile sensing. We adopt a credit-based approach which allows each user to earn credits by contributing its data without leaking which data it has contributed. At the same time, the approach ensures that dishonest users cannot abuse the system to earn unlimited amount of credits. Following this approach, we propose two privacy-aware incentive schemes. The first scheme is designed for scenarios where a trusted third party (TTP) is available. It relies on the TTP to protect user privacy, and thus has very low computation and storage cost at each user. The second scheme considers scenarios where no TTP is available. It applies blind signature, partially blind signature and commitment techniques to protect privacy. To the best of our knowledge, they are the first privacy-preserving incentive schemes for mobile sensing.

The remainder of this paper is organized as follows. Section II discusses related work. Section III presents system models, assumptions and cryptographic primitives. Section IV provides an overview of our approach. Section V and Section VI present our two incentive schemes, respectively. Section VII evaluates our solutions in cost. Section VIII presents discussions. Section IX concludes the paper.

II. Related Work

Many schemes [5]–[7], [9]–[14] have been proposed to protect user privacy in mobile sensing. AnonySense [5] and PEPSI [6] enable anonymous data collection from mobile users. DeCristofaro et al [15] considered scenarios where external entities query specific users' sensing data and proposed a scheme to hide which user matches a query. Gilbert et al [16] proposed to use TPM to secure user data. Privacy-aware data aggregation in mobile sensing has also been studied by several works [17]–[19]. However, these schemes cannot provide incentives for users to participate. Christin et al [7] proposed a privacy-aware reputation scheme for mobile sensing which uses reputation to filter incorrect sensor readings, but their solution does not address incentive either. Recently, Yang et al [8] proposed incentive schemes for mobile sensing based on gaming and auction theories, but their work

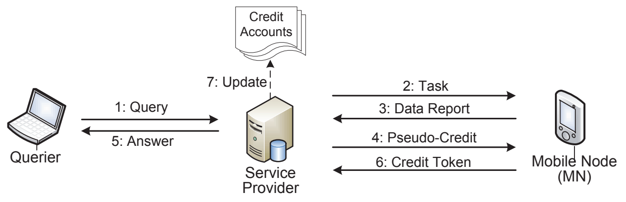

Fig. 1. System model.

does not consider the protection of user privacy. Privacy-aware incentives designed for other applications (e.g., publish-subscribe systems) [20] cannot be directly applied here since they do not consider the requirements of mobile sensing.

III. Preliminaries

A. System and Incentive Model

Figure 1 shows our system model. The system mainly consists of a dynamic set of mobile nodes (MNs), a mobile sensing Service Provider (SP), and a set of queriers. MNs are mobile devices with sensing, computation and communication capabilities, e.g., smart phones. MNs are carried by people or mounted to vehicles and other objects. They have (possibly intermittent) Internet access via 3G, WiFi or other available networks. Queriers use the mobile sensing service. They send queries to the SP to request the desired statistics and context information, e.g., "*What is the pollen level in Central Park?*" The SP collects sensor readings from MNs and answers the queries based on the collected data.

The SP pays credits to the carrier of a MN for the sensing data that it contributes[1]. The credits earned by a MN can be used to buy mobile sensing service from the SP, exchanged for discount of the MN's 3G service, or converted to other real-world rewards. Thus, MNs are incentivized to participate.

The basic workflow is as follows. A querier sends a query to the SP. To answer the query, the SP transforms the query into one or more tasks and adds them into a *task queue*. A task specifies what sensor readings to report, and when and where to sense. The task also specifies an expiration time, after which it should be deleted. When a MN has network access, it (using a random pseudonym generated by itself) polls the SP to retrieve tasks. After retrieving a task, the MN decides whether to accept the task based on certain criteria (e.g., if it has the required sensing capability). If the MN accepts the task, it will collect its sensor data at the time and location specified by the task, and generate one report. Then it submits the report to the SP using a new pseudonym in a new communication session. In the same communication session, the SP pays a certain number of credits to the reporting MN. Since it does not know the real identity of the MN, it issues pseudo-credits to the MN which will be used to generate real credit tokens. After the SP collects enough reports for a task, it deletes the task from the task queue. It aggregates the reported data to obtain the answer for the appropriate query, and sends the answer back to the querier. When the MN receives the pseudo-credits, it transforms them into credit tokens. After a random time (to avoid timing attacks), it deposits each credit token to the SP with its real identity. The SP updates the MN's credit account accordingly. Cashing a pseudo-credit for a credit token relies on a secret which is only known by the MN, such that the SP cannot link the credit token to the pseudo-credit and hence does not know the report from which the credit was earned.

When a MN retrieves tasks, the SP sends a random subset of tasks (e.g., 100 tasks) in the queue to the MN[2]. Since there may be many tasks, delivering a subset of them can reduce the communication cost. In this approach, a MN may repeatedly retrieve the same tasks. Here performance is sacrificed for privacy. If the MN reveals the tasks that it retrieved before, it is easier for the SP to link the tasks accepted by the MN. Although the MN may retrieve a task multiple times, it accepts the task at most once. To mitigate timing attacks, the MN waits a random time between successive task retrievals.

Among the tasks that a MN retrieves in the same communication session with the SP, at most one task will be accepted by the MN. Here we sacrifice performance for privacy. If the MN accepts multiple tasks retrieved in the same session and the SP does not send these tasks to other MNs, the SP knows that the collected reports must be submitted by the same MN. Such knowledge can help the SP to infer the real identity of the MN.

The amount of credits paid for different reports may be different. It depends on the type of sensor reading to report and the amount of effort needed to submit the report. For example, suppose it requires the carrier to take a high-definition photograph to generate a report. Since the generation process needs user intervention and the submission of the report causes much network traffic, more credits should be paid for the report. In contrast, if a report just needs an accelerator reading which can be obtained without human intervention and does not induce much communication cost, less credits can be paid for it. Let c denote the number of credits paid for a report. The value of c is set by the SP. A MN may not accept a task if the amount of credits paid for the task is less than its expectation. Thus, the SP should set an appropriate c for each task (e.g., higher values for more challenging tasks). We assume that c has integer values ranging from 1 to a certain maximum c_{max}. We expect that c_{max} is not large in practice, e.g., $c_{max} = 5$.

The SP may charge queriers fees for using its service. The fee charged for a query is more than the amount of credits that the SP pays for the reports collected to answer the query, such that the SP can make a profit. To control the cost of answering queries, the SP needs to control the number of reports that each MN can submit for each task. In this paper, we assume that each task allows one report from each MN.

In practice, many queries can be answered by a single report containing the required sensor reading, e.g., *What is the temperature in Central Park?* Other queries that need a series of reports can be answered by creating multiple tasks each of which only needs one report. For a query that requires periodical sensor readings (e.g., *What is the hourly pollen level in Central Park?*), the SP can create one single-report task in every period. For a query that requires event-driven reports

[1]The carrier of a MN is the person that carries the MN, or the owner of the vehicle where the MN is mounted. We use MN and carrier interchangeably.

[2]In future work, we will consider other methods to determine which subset of tasks the SP should send. For example, it may send recently created tasks. Also, a MN may specify certain attributes that retrieved tasks should satisfy, on condition that the revealed attributes do not cause much privacy leakage.

(e.g., *What is your location when you drive over a pothole?*), the SP can ensure that there is always a single-report task for this query in the task queue, such that MNs can always retrieve it. We will explore more efficient techniques to support such queries in future work.

B. Adversary and Trust Model

1) Threats to Incentive: MNs may want to earn as many credits as possible. To achieve this goal, a MN may submit a lot of reports for each task (using a different pseudonym to submit each report), and try to obtain more than c credit tokens for the task. A number of MNs may collude. A malicious MN may compromise some other MNs, and steal their credentials to earn more credits. For example, it steals their credit tokens and deposits these tokens to its own credit account.

With respect to incentive, we assume that the SP behaves honestly. It will not repudiate valid credit tokens deposited by MNs, since this will discourage the MNs from contributing sensing data in the future. As discussed in Section III-A, the SP may make profits from providing mobile sensing service, and thus it is not of interest for the SP to discourage MNs from participation. Also, the SP will not manipulate any MN's credit account (e.g., reducing the credits without its approval).

Malicious MNs may submit false sensor readings to prevent the SP from obtaining correct answers. Such data pollution attacks are outside the scope of this paper. However, the effect of false readings can be mitigated by using an anonymous reputation scheme (e.g., IncogniSense [7]) to filter the reports submitted by the MNs with low reputations.

2) Threats to Privacy: The SP is curious about which tasks a MN has accepted, and what reports the MN has submitted.

The SP may craft a task which targets a narrow set of MNs and thus makes it easier to identify the MNs accepting this task. For example, the crafted task only allows the faculty members of a university to report their data. For such a task, even a single report may leak the reporter's affiliation. This problem is not unique to our scenario, and it can be addressed as follows [5]: A registration authority is used to verify that no task targets a narrow set of MNs, and only verified tasks with the authority's signature can be published by the SP. In this paper, we do not consider tasks that target a narrow set of MNs, but the aforementioned solution to this problem can be easily applied to our approach.

3) Trust Model: We assume that the SP and each MN have a pair of public and private keys, which can be used to authenticate each other. These keys are issued by a (possibly offline) certificate authority. An adversary may compromise a MN and know its keys, but the adversary cannot bind the compromised MN to a new pair of authentication keys. Similar to [5], we assume that the communications between MNs and the SP are anonymized (e.g., with IP and MAC address recycling techniques and Mix Networks).

C. Our Goals

With respect to incentive, we ensure that no MN can earn more credits than allowed by the SP. More formally, suppose the SP is willing to pay c credits for one report of a task. Then each MN can earn at most c credits by submitting reports for this task. We have two goals in preserving privacy. First, given a report, the SP cannot tell which MN has submitted this report. Second, given multiple reports submitted by the same MN, the SP cannot tell if these reports are submitted by the same MN.

D. Cryptographic Primitives

1) Blind Signature: A blind signature scheme [21] enables a user to obtain a signature from a signer on a message, such that the signer learns nothing about the message being signed. More formally, to obtain a signature on message m, the user first blinds the content of m with a random blinding factor, and then sends the blinded message m' to the signer. The signer signs on m' using a standard signing algorithm (e.g., RSA), and passes the signature σ' back to the user. The user removes the blinding factor from σ', and obtains a signature σ on m which can be verified using the signer's public key. This process satisfies two properties. The first property is blindness, which ensures that the signer cannot link $\langle m, \sigma \rangle$ to m' or σ'. The second property is unforgeability, which ensures that from σ' the user cannot obtain a valid signature for any other message $m'' \neq m$.

In this paper, we use the blind RSA signature scheme [22] due to its simplicity. However, our approach can be easily adapted to other schemes as well. The blind RSA signature scheme works as follows. Let Q denote the public modulus of RSA, e denote the signer's public key and d denote the signer's private key. To obtain a blind signature on message m, the user selects a random value z which is relatively prime to Q, and computes $m' = mz^e \mod Q$. The signer computes the signature $\sigma' = (m')^d \mod Q$. From σ', the user obtains the signature for m by computing $\sigma = (\sigma' \cdot z^{-1}) \mod Q$.

2) Partially Blind Signature: A partially blind signature scheme (e.g., [23]) is quite similar to a blind signature scheme in that it also allows a user to obtain a signature from a signer on a message, without revealing the content of the message to the signer. The only difference is that it allows the signer to explicitly include some common information (e.g., date of issue), which is under agreement with the user, in the signature. If the common information is attached to many signatures, the signer cannot link a signature to the secret message. Our approach does not assume any specific partially blind signature scheme. We simply use $PBS_K(p, m)$ to denote a partially blind signature, where K is the signing key, m is the secret message and p is the common information attached to the signature. Note that the signer cannot link the signature to the communication session in which the signature is generated.

IV. An Overview of our Approach

A. Basic Approach

To achieve the incentive goal that each MN can earn at most c credits from each task, our approach satisfies three conditions: (i) each MN can accept a task at most once, (ii) the MN can submit at most one report for each accepted task, and (iii) the MN can earn c credits from a report. To satisfy the first condition, the basic idea is to issue one *request token* for each task to each MN. The MN consumes the token when it accepts the task. Since it does not have more tokens for the task, it cannot accept the task again. Similarly, to satisfy the second condition, each MN will be given one *report token* for each task. It consumes the token when it submits a report for the task and thus cannot submit more reports. To satisfy the last condition, when the SP receives a report, it issues pseudo-credits to the reporting MN which can be transformed to c *credit tokens*. The MN will deposit these tokens to its credit account.

To achieve the privacy goals, all tokens are constructed in a privacy-preserving way, such that a request (report) token cannot be linked to a MN and a credit token cannot be linked to the task and report from which the token is earned.

Thus, our approach precomputes privacy-preserving tokens for MNs which are used to process future tasks. To ensure that MNs will use the tokens appropriately (i.e., they will not abuse the tokens), commitments to the tokens are also precomputed such that each request (report) token is committed to a specific task and each credit token is committed to a specific MN.

B. Scheme Overview

Following the aforementioned approach, we propose two schemes. The first scheme assumes a trusted third party (TTP), and uses the TTP to generate tokens for each MN and their commitments. This scheme relies on the TTP to protect each MN's privacy, and thus has very low computation and storage cost at each MN. The second scheme does not assume any TTP. Each MN generates its tokens and commitments in cooperation with the SP using blind signature and partially blind signature techniques. The use of blind and partially blind signatures protects the MN's privacy against attacks by any third party. Certainly, such unconditional privacy is not free: each MN has higher computation and storage overhead.

Both schemes work in five phases as follows.

Setup In this phase, the tokens and their commitments that each MN and the SP will use to process the next M (which is a system parameter) tasks are precomputed, and distributed to each MN and the SP. The distribution process ensures that each MN cannot get the report token for a task unless it is approved by the SP to accept the task, and it cannot get the credit tokens for a task unless it submits a report for the task.

Task assignment Suppose a MN has retrieved a task i from the SP via an anonymous communication session. If the MN decides to accept this task, it sends a request to the SP. The request includes the MN's request token. The SP verifies that the token has been committed for task i in the setup phase. If the SP allows the MN to accept this task, it returns an approval message to the MN. From the approval message, the MN can compute a report token for task i. However, the MN cannot derive a valid report token without the approval message.

Report submission After the MN generates a report for task i, it submits the report via another anonymous communication session. The MN's report token for task i is also submitted. The SP verifies that the report token has been committed for task i, and then sends pseudo-credits to the MN. From the pseudo-credits, the MN computes c credit tokens, where c is the number of credits paid for each report of task i. It cannot obtain any credit token without the pseudo-credits.

Credit deposit After the MN gets a credit token, it deposits the token to the SP after a random period of time to mitigate timing attacks. The SP verifies that the token has been committed for the MN, and then increases the MN's credit account by one.

TABLE I
NOTATIONS

M	The num. of tasks for which credentials are precomputed
N, V	The num. of real MNs and virtual MNs in the system
$c_i \in [1, c_{max}]$	The number of credits paid for each report of task i
τ, δ, ϵ	Request token, report token, credit token
r, r_1, r_2, r_3	The secrets assigned to a MN
$K_0, ..., K_3$	The private keys of the SP to generate signatures
e, d	The SP's public and private key for blind RSA signature
sk	The secret key assigned to the SP
NID, PID	The real identity and pseudonym of a MN
H	A cryptographic hash function

Token and commitment renewal When the previous M tasks have been processed, the tokens and their commitments for the next M tasks should be precomputed and distributed similar to the setup phase.

Note that in the setup, credit deposit and token renewal phases, each MN communicates with the SP using its real identity. However, in the task assignment and report submission phases, each MN uses a random pseudonym generated by itself to communicate with the SP. The pseudonym cannot be linked to the real identity of the MN.

The notations used in this paper are summarized in Table I.

V. A TTP-based Scheme

This scheme assumes the existence of a TTP which always has Internet access.

A. The Basic Scheme

1) Setup: In this phase, the TTP precomputes and distributes the tokens and commitments that will be used to process the next M tasks. Without loss of generality, suppose the IDs of these tasks are $1, 2, ..., M$.

The TTP assigns and delivers a secret r to each MN and a secret key sk to the SP. The secrets for different MNs are different. The TTP also generates a nonce ρ to identify this set of secrets, and sends it to each MN and the SP. If a new set of secrets are assigned to the SP and MNs later, a new nonce will be generated. The TTP computes other credentials using the set of secrets and the nonce.

We first describe how to generate the tokens and commitments for a single MN. Let r denote the secret of this MN. From r and the nonce ρ, the TTP derives three other secrets $r_1 = H(r|\rho|1)$, $r_2 = H(r|\rho|2)$ and $r_3 = H(r|\rho|3)$. Then it generates the tokens and commitments in three steps.

Step 1. The TTP computes M request tokens for the MN. Each token will be used for one task. The token for task i ($i \in [1, M]$) is $\tau_i = H(0|H^i(r_1))$. Here, the one-wayness of hash chain is exploited to calculate τ_i (see explanations in Section V-B2). The commitment to τ_i is $\langle H(\tau_i), i \rangle$.

Step 2. The TTP computes M report tokens for the MN, with each token used for one task. The token for task i is $\delta_i = \text{HMAC}_{r_2}(i|\text{HMAC}_{sk}(\rho|\tau_i))$. Its commitment is $\langle H(\delta_i), i \rangle$.

Step 3. The TTP computes $M \cdot c_{max}$ credit tokens. Since at this time the TTP does not know the number of credits that the SP will pay for each task, it generates the maximum possible number of credit tokens for each task. The tokens for task i are computed as $\epsilon_{ij} = \text{HMAC}_{r_3}(j|i|(s' \oplus H^j(s'')))$ for $j = 0, ..., c_{max} - 1$, where $s' = \text{HMAC}_{sk}(0|\rho|\delta_i)$ and $s'' =$

HMAC$_{sk}(1|\rho|\delta_i)$. The commitment of ϵ_{ij} is $\langle H(\epsilon_{ij}), NID\rangle$, where NID is the MN's real identity.

Following these steps, the TTP can also generate the tokens and commitments for other MNs. The TTP randomly shuffles each category of commitments and sends them to the SP.

At the end of this phase, each MN gets one secret and one nonce. The SP gets one secret key, one nonce, $N \cdot M$ commitments for request (report) tokens and $N \cdot M \cdot c_{max}$ commitments for credit tokens. The TTP stores the secret key of the SP, the secret of each MN and the nonce.

2) Task Assignment: Suppose a MN has retrieved a task i. If it decides to accept this task, it sends a request to the SP using a pseudonym PID_1. The request contains its request token for this task, which is $\tau_i = H(0|H^i(r_1))$ where $r_1 = H(r|\rho|1)$.

$$\text{MN} \rightarrow \text{SP:}\ \ PID_1, i, \tau_i \tag{1}$$

The SP verifies that $\langle H(\tau_i), i\rangle$ is a valid commitment and deletes this commitment to avoid token reuse. Then it sends an approval message to the MN:

$$\text{SP} \rightarrow \text{MN:}\ \ \text{HMAC}_{sk}(\rho|\tau_i) \tag{2}$$

From this message, the MN computes its report token for task i, i.e., $\delta_i = \text{HMAC}_{r_2}(i|\text{HMAC}_{sk}(\rho|\tau_i))$ where $r_2 = H(r|\rho|2)$.

3) Report Submission: When the MN, using a pseudonym PID_2, submits a report for task i, it also submits its report token δ_i for this task:

$$\text{MN} \rightarrow \text{SP:}\ \ PID_2, i, \delta_i, report \tag{3}$$

The SP verifies that $\langle H(\delta_i), i\rangle$ is a valid commitment and deletes this commitment to avoid token reuse. Then it computes $s' = \text{HMAC}_{sk}(0|\rho|\delta_i)$, $s'' = \text{HMAC}_{sk}(1|\rho|\delta_i)$ and $s''' = H^{c_{max}-c_i}(s'')$ and sends the following back to the MN.

$$\text{SP} \rightarrow \text{MN:}\ \ s', s''' \tag{4}$$

Using s' and s''', the MN computes c_i credit tokens $\epsilon_j = \text{HMAC}_{r_3}(j|i|(s' \oplus H^j(s'''))) = \text{HMAC}_{r_3}(j|i|(s' \oplus H^{c_{max}-c_i+j}(s'')))$ for $j = 0, ..., c_i - 1$. Due to the one-way property of H, the MN cannot obtain other credit tokens.

4) Credit Deposit: After the MN gets a credit token ϵ, it waits a length of time randomly selected from $(0, T]$ to mitigate timing attacks (see Section V-C) and then deposits the token using its real identity NID:

$$\text{MN} \rightarrow \text{SP:}\ \ NID, \epsilon \tag{5}$$

The SP verifies that $\langle H(\epsilon), NID\rangle$ is a valid commitment and deletes this commitment to avoid token reuse. Then it increases the MN's credit account by one.

5) Commitment Renewal: When the first M tasks have been processed, the SP should communicate with the TTP again to obtain another set of commitments for the next M tasks. The commitments for the previous M tasks will be deleted later as discussed in Section V-D. The SP's secret key, each MN's secret and the nonce are not changed.

B. Dealing with Dynamic Joins and Leaves

1) Join: In the setup phase, the TTP assumes the existence of V (a system parameter) virtual MNs besides the N real MNs. It generates the tokens and commitments for both real and virtual MNs. Also, it sends the commitments to the request and report tokens of the virtual MNs, mixed with the commitments for the real MNs, to the SP.

When a new MN joins, the TTP maps it to an unused virtual MN and sends the virtual MN's secret r to it. Also, the TTP generates the credit tokens for the new MN (i.e., the mapped virtual MN) and sends their commitments to the SP. Afterward, it tags the mapped virtual MN as used.

If there is no available unused virtual MN when the new MN joins, the TTP reruns the setup phase again in which a new set of secrets are issued to the SP and all the current MNs as well as a new set of virtual MNs. Some MNs may not have network access during the setup phase and hence cannot receive the new nonce and their new secrets. To address this problem, whenever a MN retrieves tasks from the SP, it checks if it has the same nonce ρ with the SP. Note that the SP always has the latest version of nonce. If the MN's nonce is out of date, it means that the MN has missed the previous setup phase and its secret is also out of date. In this case, the MN connects to the TTP to update its secret and nonce.

In practice, the value of parameter V can be adjusted according to churn rate. If the churn rate is high (i.e., new MNs join frequently), a larger V can be used to reduce the number of reruns of the expensive setup phase, at the cost of higher storage at the SP. If the churn rate is low, a smaller V can be used to reduce the storage overhead at the SP.

2) Leave: When a MN leaves, its request tokens for future tasks should be invalidated at the SP. Note that if the request token for a future task is invalidated, the report token and credit tokens for the same task are also invalidated automatically, since the the leaving MN will not be able to compute them. Let r denote the leaving MN's secret, ρ denote the current nonce and $r_1 = H(r|\rho|1)$. The TTP releases $\lambda = H^i(r_1)$ to the SP, where i is the next task to be published. From λ, the SP can compute the request tokens of the leaving MN for future tasks. For example, the token for a future task $i + j$ is $H(0|H^j(\lambda))$. The SP will invalidate these tokens. However, due to the one-way property of H, the SP cannot derive the tokens that the leaving MN used in previous tasks. No changes are made to other MNs.

C. Addressing Timing Attacks

If a MN deposits a credit token earned from a report immediately after it submits the report, since it uses its real identity to deposit the token, the SP may be able to link the report to it via timing analysis. Thus, the MN should wait some time before it deposits the credit token. Specially, after a MN gets a credit token, it waits a length of time randomly selected from $(0, T]$ and then deposits the token.

The parameter T is large enough (e.g., one month) such that, in each time interval T, many tasks can be created and most MNs have chances to connect to the SP. The SP will store the commitments to the credit tokens for a time period of at least $2T$ (see Section V-D), such that most MNs can deposit their credit tokens before the commitments are deleted. If a MN (e.g., with very infrequent network access) wants to deposit some credit tokens after their commitments are deleted, the SP can check the validity of these tokens with the TTP, and update the MN's credit account accordingly.

D. Commitment Removal

The SP removes the commitments to the previous M tasks as follows. Note that part of commitments are removed to avoid token reuse immediately after the corresponding tokens are verified. Since not all MNs accept all tasks, some commitments may remain after the previous M tasks have been processed. Let t_{exp} denote the maximum time at which each of the previous M tasks will expire. Note that all reports for the M tasks are submitted before t_{exp} and all credit tokens paid for these reports are sent to MNs before t_{exp}. Thus, the SP can remove the remaining commitments to request and report tokens after time t_{exp}. To allow MNs to deposit their earned credit tokens, the SP stores the remaining commitments to credit tokens for another time period of $2T$ (as discussed in Section V-C), and removes them after time $t_{exp} + 2T$.

E. Security Analysis

1) Attacks on Incentive: Without loss of generality, let us consider a task i which is paid at a rate of c credits per report.

Our scheme ensures that each MN can earn at most c credits from the task by satisfying three conditions. First, the MN can only obtain one request token for the task and hence can only be approved by the SP once to report for the task. Second, from the approval message sent by the SP, the MN can only get one report token and thus can submit at most one report for the task. Third, after submitting a report, the MN can only obtain c credit tokens.

Also, if a MN is not approved by the SP to accept the task, it cannot obtain the report token and thus cannot submit a report for the task. Without submitting a report, it cannot obtain the credit tokens and thus cannot earn credits from the task.

A dishonest MN may forge tokens, but the forged tokens cannot pass the commitment check. The MN may use the request token of task j (i.e., τ_j) to request for task i, but this will fail since the commitment to τ_j (i.e., $\langle H(\tau_j), j\rangle$) has bound τ_j to j. Similarly, the MN cannot use the report token of one task to another task.

A dishonest MN may have compromised a number of other MNs and obtained their secrets. It may use the request token of a compromised MN to request for task i, use the report token of the compromised MN to submit a report and obtain some credit tokens. However, the obtained credit tokens have been bound to the compromised MN by their commitments, and thus cannot be deposited to the dishonest MN's credit account. If the dishonest MN is not approved to report for task i but those compromised MNs are, it may submit its own tokens from one compromised MN and earn c credits. However, it still cannot increase its credits using the tokens of the compromised MNs. Thus, the dishonest MN cannot earn more than c credits from task i no matter how many MNs it compromises.

2) Attacks on Privacy: Since each MN uses pseudonyms to retrieve tasks, request tasks and submits reports, the SP cannot know the MN that has submitted a specific report from whom it is communicating with. Also, the SP cannot know the information from the request and report tokens, since the tokens have been randomized (i.e., anonymized) by each MN's secrets that the SP does not know. For similar reasons, the SP cannot link multiple reports submitted by the same MN.

The SP can link a credit token to a MN, since the MN uses its real identity to deposit the credit token. However, the SP cannot link the credit token to the report and task (or the report token and request token) from which the credit is earned, since the connection between them has been anonymized using the MN's secret r_3. Thus, linking a credit token to a MN does not help the SP to break the MN's privacy.

F. Cost Analysis

Each MN only stores one secret and one nonce. The TTP stores the secret key of the SP, the secret of each MN and the nonce. It can be seen that the storage at each MN and the TTP is low. The SP mainly stores $M(N+V)(2c_{max}+1)$ commitments for the next M tasks, where N (V) is the number of real (virtual) MNs. It also stores $(N+V)c_{max}$ credit token commitments for each task created in the past time window $2T$. Let us consider a simple case. Suppose $N = 10000$, $V = 1000$, $M = 1000$, $c_{max} = 5$, $T = 30$ days and 1000 tasks are generated per day. Also, suppose SHA-256 is used as the hash function H, and each task ID or node ID has 8 bytes. Then the storage at the SP is about 137GB. We expect that such storage cost is not an issue for modern servers.

For each task (throughout the setup, task assignment, report submission and credit deposit phases), each MN computes at most $c_{max}+2$ hashes and $c_{max}+1$ HMACs, the SP computes at most $2N(c_{max}+1)$ hashes and $3N$ HMACs, and the TTP computes $(N+V)(3c_{max}+4)$ hashes and $(N+V)(c_{max}+4)$ HMACs. Since hash and HMAC are extremely efficient, the computation cost is low.

Since each message that a MN sends and receives mainly contains one or two hash values, the communication cost is about two hundred bytes per task for each MN, which is low.

VI. A TTP-Free Scheme

To provide privacy-aware incentives for the scenarios where no TTP is available, we design a TTP-free scheme. It uses blind signature and partially blind signature to generate tokens and commitments for MNs in a privacy-preserving way.

A. The Scheme

The SP has three private keys K_1, K_2 and K_3 which are used to generate partially blind signatures, and another private key K_0 which is used to generate traditional digital signatures. The SP also has a private key d and public key e which are used to generate blind RSA signatures. These keys are assigned by a (possibly offline) certificate authority. Besides, the SP has a secret key sk generated by itself, and each MN has three secrets r_1, r_2 and r_3 generated by itself.

1) Setup: Each MN communicates with the SP with its real identity to obtain the tokens and commitments for the first M tasks. Suppose these tasks' identifier are $1, 2, ..., M$.

For each task i ($i = 1, 2, ..., M$), the MN computes c_{max} random values $m_{ij} = H(i|j|H^i(r_1))$ where $j = 1, 2, ..., c_{max}$. Each value and the SP's RSA signature over the value constitute one credit token, which is $\epsilon_{ij} = \langle m_{ij}, SIG_d(m_{ij})\rangle$. However, the MN cannot obtain the RSA signature until it submits a report for this task.

To commit to credit token ϵ_{ij}, the MN sends $H(m_{ij})$ and its real identity NID to the SP. The SP signs on $\langle H(m_{ij}), NID\rangle$

with key K_0 and returns the signature $SIG_{K_0}(H(m_{ij})|NID)$. Then the MN obtains the commitment to ϵ_{ij}, which is $\langle H(m_{ij}), NID, SIG_{K_0}(H(m_{ij})|NID)\rangle$. To reduce the computation cost, the SP can build a Merkle hash tree [24] over all $\langle H(\epsilon_{ij}), NID\rangle$ of the same MN, and generates just one digital signature for all of them. The SP ensures that no two $H(m_{ij})$ (of the same MN or different MNs) are identical. Since each m_{ij} is a result of the hash function H, the probability of generating two identical m_{ij} (and $H(m_{ij})$) is negligible.

For each m_{ij}, the MN generates a random blinding factor $z_{ij} = H(i|j|H^i(r_2)|x)$, where x is the smallest positive integer such that z_{ij} is relatively prime to the pubic modulus Q of the RSA signature. From each pair of m_{ij} and z_{ij}, the MN computes $\mu_{ij} = m_{ij} \cdot z_{ij}^e \mod Q$. Then it computes c_{max} report token components for task i which are $b_{ij} = H(\mu_{i1}|\mu_{i2}|...|\mu_{ij}|i|j)$ where $j = 1, 2, ..., c_{max}$. Each b_{ij} and the SP's partially blind signature $PBS_{K_2}(i, b_{ij})$ over it constitute one possible report token for task i, which is $\delta_{ij} = \langle b_{ij}, PBS_{K_2}(i, b_{ij})\rangle$. However, the MN cannot obtain the signature until it is approved by the SP to accept this task. Only one report token for task i will be obtained by each MN.

To commit to report token δ_{ij}, the MN obtains a partially blind signature $PBS_{K_3}(i, b_{ij})$ from the SP. However, the SP does not send this signature to the MN in plaintext. It encrypts the signature with key $k_{ij} = H(sk|i|j)$, and sends the ciphertext $E_{k_{ij}}(PBS_{K_3}(i, b_{ij}))$ to the MN. Given i and j, the SP uses the same encryption key k_{ij} for all MNs.

For each task i, the MN generates a request token $\tau_i = H(0|H^i(r_3))$. Similar to the TTP-based scheme, hash chain is used to calculate τ_i. To commit to token τ_i, it obtains a partially blind signature $PBS_{K_1}(i, \tau_i)$ from the SP.

In the end, each MN obtains M commitments to request tokens, $M \cdot c_{max}$ encrypted commitments to report tokens, and $M \cdot c_{max}$ commitments to credit tokens. The tokens τ_i, δ_{ij}, and ϵ_{ij} can be stored for later use or generated on the fly.

Since b_{ij} is committed, $\mu_{i1}, \mu_{i2}, ..., \mu_{ij}$ are also committed due to the one-way property of H. Considering that m_{ij} is committed, the random blinding factor z_{ij} is fixed. Thus, each MN cannot change its credentials after the setup phase.

2) Task Assignment: When the SP publishes task i, it also publishes c_i, which is the number of credits paid for each report of task i. In the following, we omit the subscript and refer to the number as c for simplicity. The SP also publishes a key $k' = k_{ic} = H(sk|i|c)$. Note that in the setup phase, each MN generated c_{max} report token components for task i and obtained an encrypted commitment for each of them. From c, each MN knows that the report token component it should use for task i is $b_{ic} = H(\mu_{i1}|\mu_{i2}|...|\mu_{ic}|i|c)$. Using key k', it can decrypt the commitment to b_{ic}, which is $PBS_{K_3}(i, b_{ic})$.

Suppose a MN has retrieved a task i. If it decides to accept task i, it sends a request to the SP using a pseudonym PID_1. The request includes the MN's request token τ_i and its commitment $PBS_{K_1}(i, \tau_i)$.

$$\text{MN} \rightarrow \text{SP:}\quad PID_1, i, \tau_i, PBS_{K_1}(i, \tau_i) \qquad (6)$$

The SP verifies the signature $PBS_{K_1}(i, \tau_i)$, and knows that token τ_i has been committed for task i. Then the MN obtains a partially blind signature $PBS_{K_2}(i, b_{ic})$ on b_{ic} from the SP via a standard partially blind signature scheme. Conceptually, the SP delivers the signature in an approval message:

$$\text{SP} \rightarrow \text{MN:}\quad PBS_{K_2}(i, b_{ic}) \qquad (7)$$

Now the MN obtains a report token for task i, which is $\delta_{ic} = \langle b_{ic}, PBS_{K_2}(i, b_{ic})\rangle$.

3) Report Submission: When the MN, using a pseudonym PID_2, submits a report for task i, it also submits its report token for this task and the commitment:

$$\begin{aligned}\text{MN} \rightarrow \text{SP:}&PID_2, i, b_{ic}, PBS_{K_2}(i, b_{ic}), PBS_{K_3}(i, b_{ic}),\\ &\mu_{i1}, \mu_{i2}, ..., \mu_{ic}, report\end{aligned} \qquad (8)$$

Signature $PBS_{K_2}(i, b_{ic})$ ensures that the MN has been approved by the SP to report for task i, and signature $PBS_{K_3}(i, b_{ic})$ ensures that b_{ic} has been committed for task i. If the two signatures are valid, the SP can issue c pseudo-credits to the MN. Specifically, the SP first verifies that $b_{ic} \equiv H(\mu_{i1}|\mu_{i2}|...|\mu_{ic}|i|c)$. Due to the one-way property of H, the SP knows that each μ_{ij} ($j = 1, 2, ..., c$) has also been committed for task i. Then it signs each μ_{ij} with the private key d and sends the signatures to the MN.

$$\text{SP} \rightarrow \text{MN:}\quad SIG_d(\mu_{i1}), SIG_d(\mu_{i2}), ..., SIG_d(\mu_{ic}) \qquad (9)$$

From each signature $SIG_d(\mu_{ij})$, the MN removes the blinding factor $z_{ij}^e \mod Q$ and gets a blind RSA signature for m_{ij} which is $SIG_d(m_{ij})$. Then it obtains c credit tokens $\epsilon_{ij} = \langle m_{ij}, SIG_d(m_{ij})\rangle$ ($j = 1, 2, ..., c$).

4) Credit Deposit: After the MN gets a credit token $\epsilon = \langle m, SIG_d(m)\rangle$, it waits a length of time randomly selected from $(0, T]$ to mitigate timing attacks (see Section V-C) and then deposits the token using its real identity NID:

$$\text{MN} \rightarrow \text{SP:}\ NID, m, SIG_d(m), SIG_{K_0}(H(m)|NID) \quad (10)$$

The second signature means that the credit token has been committed to this MN. The SP verifies the two signature and then increases the MN's credit account by one.

5) Token and Commitment Renewal: When the first M tasks have finished or expired, each MN should communicate with the SP to obtain a new set of commitments (as done in the setup phase) for the next M tasks whose identifiers are $M+1, M+2, ..., 2M$. The keys used by each MN and the SP do not change. Similarly, renewal is needed for every task group $[k \cdot M + 1, (k+1)M]$ ($k \geq 0$).

The SP may launch isolation attacks, in which it only issues the commitments for the next M tasks to one MN. When the MN submits reports for these tasks, the SP can link the reports to it. To address this attack, each MN generates a signature over the task ID range (i.e., the smallest and largest ID) of the next M tasks. Before submitting a report for a task, each MN makes sure that the SP has collected signatures from a large-enough number of MNs (e.g., half of the MNs).

B. Dealing with Dynamic Joins and Leaves

Let i denote identifier of the most recently created task. Suppose $k \cdot M \leq i < (k+1)M$. When a new MN joins, it obtains the commitments for the tasks in range $[i+1, (k+1)M]$ from the SP, as done in the setup phase. When a node leaves, its credentials for the tasks in range $[i+1, (k+1)M]$ should be revoked. Let r_1, r_2 and r_3 denote its secrets. The MN releases $H^{i+1}(0|r_1)$, $H^{i+1}(0|r_2)$ and $H^{i+1}(0|r_3)$ to the SP. From them, the SP can compute the leaving MN's tokens and commitments for those tasks, and invalidate them. Due to the one-way property of hash function H, the SP cannot derive the credentials that the leaving MN used for previous tasks.

C. Credential Removal

A MN removes the credentials for a task if it decides not to accept the task. For the tasks that it accepted, the tokens and commitments can be removed after they are submitted or deposited to the SP.

D. Security Analysis

1) Attacks on Incentive: Similar to the analysis in Section V-E1, the TTP-free scheme also ensures that each MN can earn at most c credits from a task.

A dishonest MN may forge tokens but these tokens will fail the commitment check. Since the commitment to each request (report) token binds the token to a task identifier, the MN cannot use the request (report) token of one task to process another task.

A dishonest MN may compromise a number of other MNs but it still cannot earn more than c credits from a task. Since the commitment of each credit token binds the token to a specific MN, the dishonest MN cannot deposit the credit tokens of the compromised MNs to its own account. It may use the report tokens of a compromised MN to submit reports, but the obtained credit tokens have been committed to the compromised MN. This is because, given b_{ij}, the μ_{i1}, ..., μ_{ij} used to generate b_{ij} are determined due to the one-wayness of hash function H. Due to the unforgeability of blind signature, the m_{i1}, ..., m_{ij} are also determined. In the next token renewal phase, the dishonest MN may want to bind the credit tokens of the compromised MNs (in addition to its own credit tokens) to its own identity, but the binding of more than one set of credit tokens to the same MN will be detected by the SP.

2) Attacks on Privacy: Since each request and report token (as well as its commitment) is constructed using a partially blind signature, the SP cannot know the MN that has submitted a specific report, and cannot link multiple reports submitted by the same MN. Although the SP can link a credit token to a MN, since the connection between the credit token and its corresponding report token is anonymized using a blind RSA signature (i.e., the connection between m_{ij} and μ_{ij} is blinded with the random factor z_{ij}), the SP cannot link the credit token to the report from which the credit is earned. Thus, linking a credit token to a MN does not lead to any privacy leakage.

E. Cost Analysis

Each MN mainly stores $M(2c_{max}+1)$ commitments for the next M tasks. It also stores the credit tokens earned during the past time interval T. Let us consider a simple case. Suppose the parameters are the same as in Section V-F, and a MN accepts 100 tasks per day. Also, suppose RSA (1024-bit) is used as the digital signature scheme and partially blind RSA [25] is used as the partially blind signature scheme. Then the storage at each MN is about 2.7MB. The cost is not an issue for modern smart phones with many gigabytes of storage.

TABLE II
THE RUNNING TIME OF OUR SCHEMES WHEN $M = 1000$ AND $c_{max} = 5$

		Setup	Task Assignment	Report Submission	Credit Deposit
TTP-based	TTP	45ms*	-	-	-
	SP	-	4μs	8μs	4μs
	MN	-	0.56ms	2.4ms	-
TTP-free	SP	32s*	4.1ms	8.2ms	0.2ms
	MN	37s	6.3ms	0.5ms	-

* The time is needed to generate credentials for each MN.

For the whole life cycle of each task, each MN mainly performs one modular exponentiation and participates in the generation of $c_{max} + 2$ partially blind signatures. In the setup phase, the SP generates $N \cdot M \cdot c_{max}$ digital signatures and $NM(c_{max} + 1)$ partially blind signatures. Later on, when the SP receives a request, report or credit token, it generates at most c_{max} signatures and verifies at most two signatures.

Each message mainly contains at most c_{max} signatures or hashes. Since expectedly c_{max} is not large in practice, the communication cost is low. Under the aforementioned parameter settings, it is about 4.5KB per task. Considering that setup and credit deposit can run when the MN has WiFi access, the remaining cost is about 1.4KB per task.

VII. EVALUATIONS

To study the feasibility of our solution, we have built a prototype and implemented our schemes in Java. The prototype has three components: a MN, a SP and a TTP. The MN is implemented on Android Nexus S Phone, which is equipped with 1GHz CPU, 512MB RAM, running Android 4.0.4 OS. The SP and TTP are implemented on a Windows Laptop with 2.6GHz CPU and 4GB RAM. For simplicity, our prototype uses RSA as the digital signature scheme and partially blind RSA [25] as the partially blind signature scheme. SHA-256 is used as the hash function.

Table II shows the running time of our schemes when $M = 1000$ and $c_{max} = 5$. In the TTP-based scheme, every phase can be finished within tens of milliseconds, which means the computation cost is very low. In the TTP-free scheme, the task assignment, report submission and credit deposit phases only take several milliseconds. The running time of the setup phase is long (around half a minute) for both the MN and the SP, which means high computation cost. However, such cost is amortized among 1000 tasks. Also, the running time can be significantly reduced if other more efficient signature schemes are used instead of RSA. Using other native libs (e.g., OpenSSL) in combination with Java can also reduce the running time. Moreover, the SP can be implemented on more powerful high-end servers to further shorten the running time. Lastly, since the setup phase for certain tasks is run before these tasks are created, the running time is not a big issue even for tasks with real-time requirements.

We also measure the power consumption of the smartphone. In the experiment, the smartphone iteratively runs the whole life cycle of one task (setup, task retrieval & assignment, report submission, and credit deposit) for 2500 (500) tasks in the TTP-based (TTP-free) scheme. The smartphone communicates with the laptop using TCP connections via WiFi, and it initiates a new TCP connection for each phase. In this process, we use Agilent E3631A Power Supply to power the

TABLE III
THE POWER CONSUMPTION OF OUR SCHEMES ON A SMARTPHONE

Scheme	TTP-based	TTP-free
Power (W)	0.363	0.349
Energy cost per task (J)	0.05	0.22
Num. of Tasks per Battery (3.7V, 1500mAh)	399600	90818

smartphone at a constant voltage and program it to capture the current of the smartphone. From the voltage and current, the power consumption can be easily calculated. Table III shows the results. The TTP-based scheme only consumes 0.05 joules to process each task. The TTP-free scheme consumes a little more energy, which is 0.22 joules, due to the use of computation expensive cryptography. However, the power consumption of both schemes is low. For example, a fully-charged standard battery for a Nexus S phone (3.7V, 1500 mAh) can support 400 (90) thousand tasks when the TTP-based (TTP-free) scheme is used.

VIII. DISCUSSIONS

The SP may infer if a MN has accepted a task from the number of credits that the MN has earned, and then cause privacy leakage. For instance, suppose the SP has published 100 tasks, each of which is paid at a rate of one credit per report. If a participant Bob has earned 100 credits, the SP can infer that Bob has submitted a report for every task. If one of the tasks is "Report the temperature at 10:00 AM in Central Park," the SP knows that Bob is in Central Park at 10:00 AM. To launch this attack, the SP may create multiple tasks that require the MN to appear at close-by times (e.g., 10:01 AM) and locations. In the above example, suppose 51 tasks require a temperature reading near Central Park around 10:00AM. If Bob has earned 50 credits, at least one credit is earned from those 51 tasks. Thus the SP knows that Bob is near Central Park around 10:00 AM.

To address this attack, each MN should carefully select the tasks that it will accept and limit the number of accepted tasks. One possible approach is as follows. Among the tasks that it is able to report for, the MN identifies the "similar" tasks which may reveal the same privacy information about it (e.g., its location around a certain time). For each group of similar tasks, it accepts one of them with a certain probability (e.g., 0.5). This ensures that the number of its accepted tasks does not exceed the number of similar-task groups. From the number of credits earned by a MN, the SP does not know which tasks the MN has reported for, and thus cannot infer any private information about the MN. Since each MN intentionally omits some tasks, this approach sacrifices some chances of earning credits for better privacy. Due to the space limitation, we plan to explore this topic in a separate work.

IX. CONCLUSIONS

To facilitate large-scale deployment of mobile sensing applications, we proposed two credit-based privacy-aware incentive schemes for mobile sensing to promote user participation, corresponding to scenarios with and without a TTP respectively. Based on hash and HMAC functions, the TTP-based scheme has very low computation and storage cost at each MN. Based on blind and partially blind signatures, the TTP-free scheme has higher overhead at each MN but it ensures that no third party can break the MN's privacy. Both schemes can efficiently support dynamic joins and leaves.

REFERENCES

[1] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, "Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *Proc. ACM SenSys*, 2009, pp. 85–98.

[2] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "Peir, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proc. MobiSys*, 2009, pp. 55–68.

[3] J. Hicks, N. Ramanathan, D. Kim, M. Monibi, J. Selsky, M. Hansen, and D. Estrin, "Andwellness: an open mobile system for activity and experience sampling," in *Proc. Wireless Health*, 2010, pp. 34–43.

[4] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell, "Bewell: A smartphone application to monitor, model and promote wellbeing," in *5th Intl. ICST Conf. on Pervasive Computing Technologies for Healthcare*, 2011.

[5] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing," in *Proceedings of the 6th international conference on Mobile systems, applications, and services (MobiSys)*. ACM, 2008, pp. 211–224.

[6] E. D. Cristofaro and C. Soriente, "Short paper: Pepsi—privacy-enhanced participatory sensing infrastructure," in *Proceedings of the fourth ACM conference on Wireless network security (WiSec)*, 2011, pp. 23–28.

[7] D. Christin, C. Rosskopf, M. Hollick, L. A. Martucci, and S. S. Kanhere, "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications," in *Proc. IEEE PerCom*.

[8] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. ACM MobiCom*, 2012.

[9] S. Pidcock, R. Smits, U. Hengartner, and I. Goldberg, "Notisense: An urban sensing notification system to improve bystander privacy," in *PhoneSense*, 2011.

[10] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," in *Proc. IEEE INFOCOM*, 2008.

[11] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *IEEE INFOCOM*, 2011.

[12] K. L. Huang, S. S. Kanhere, and W. Hu, "Towards privacy-sensitive participatory sensing," in *The 5th International Workshop on Sensor Networks and Systems for Pervasive Computing*, 2009.

[13] M. Shao, S. Zhu, W. Zhang, and G. Cao, "pdcs: Security and privacy support for data-centric sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 8, pp. 1023–1038, 2009.

[14] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," in *Proc. ACM WiSec*, 2008.

[15] E. De Cristofaro and R. Di Pietro, "Preserving query privacy in urban sensing systems," in *Proc. of the 13th intl. conf. on Distributed Computing and Networking (ICDCN)*. Springer-Verlag, 2012, pp. 218–233.

[16] P. Gilbert, L. P. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proc. ACM HotMobile*, 2010, pp. 31–36.

[17] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, "Poolview: stream privacy for grassroots participatory sensing," in *Proc ACM SenSys*, 2008, pp. 281–294.

[18] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "Prisense: privacy-preserving data aggregation in people-centric urban sensing systems," in *Proc. IEEE INFOCOM*, 2010, pp. 758–766.

[19] Q. Li and G. Cao, "Efficient and privacy-preserving data aggregation in mobile sensing," in *Proc. IEEE ICNP*, 2012.

[20] M. Ion, G. Russello, and B. Crispo, "Supporting publication and subscription confidentiality in pub/sub networks," in *SecureComm*, 2010.

[21] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of CRYPTO '82*. Plenum, 1982.

[22] ——, "Blind signature system," in *Advances in Cryptology: Proceedings of CRYPTO '83*. Plenum, 1983.

[23] M. Abe and T. Okamoto, "Provably secure partially blind signatures," in *Proc CRYPTO*, 2000, pp. 271–286.

[24] R. Merkle, "Protocols for public key cryptosystems," *IEEE S&P*, 1980.

[25] M. Abe and E. Fujisaki, "How to date blind signatures," in *Proc. ASIACRYPT*, 1996, pp. 244–251.