Privacy-Preserving Participatory Sensing

Qinghua Li and Guohong Cao

ABSTRACT

The proliferation of mobile devices such as smartphones has enabled participatory sensing systems that collect data from users through their mobile devices and infer useful information from the data. However, users have concerns regarding possible privacy leakage from their data and lack incentives to contribute their data. To effectively motivate users to participate, both privacy and incentive issues need to be addressed. In this article, we address how to simultaneously protect privacy and provide incentives for participatory sensing. We review previous approaches, discuss their limitations, and propose new approaches for two types of participatory sensing systems.

INTRODUCTION

Smartphones have become an essential part of our daily life. Besides providing voice and data communication, smartphones are also acquiring richer functionality through various sensors. For example, the iPhone 5 includes eight different sensors: accelerometer, GPS, ambient light, dual microphones, proximity sensor, dual cameras, compass, and gyroscope. These sensors are very useful for gathering data about people and their environments. For example, GPS enables sensing of locations, microphones can record sounds of the surroundings, and an accelerometer enables sensing of users' movement and activities. Smartphone users are also considered as special types of sensors that generate human input data to be used in many surveys. Recently, these sensors have been used for *participatory sensing* in which the sensing data on multiple phones are collected by remote data collectors to support many interesting applications, including tracking the spread of disease across a city, building a noise map, monitoring traffic conditions, and so on.

In spite of many useful applications, there are two obstacles that hinder the large-scale deployment of participatory sensing applications [1, 2]. First, participatory sensing poses serious threats to user privacy. The data from mobile devices may be exploited to obtain private information about users, including their locations, health condition, lifestyle, religious activities, and so on. For example, an application that tries to build a noise map for a city may request that each user continuously upload his or her current location and the noise level at this location. However, from this data, the data collector can infer where the user has been (e.g., if he or she has gone to a hospital or church) and possibly infer the user's activities. Second, users lack incentives to join in participatory sensing. To participate, a user has to trigger various sensors to measure data (e.g., to obtain GPS locations), which may consume much of his/her smartphone's power. Also, the user needs to upload data to the server, which may consume some data of a wireless quota (e.g., when the data is photo/video). Moreover, the user may have to be in a specific location to sense the required data. To motivate participation, both obstacles should be addressed.

Based on what kind of data is needed by the collector, participatory sensing systems can be divided into two categories. In one category, raw data is needed by the collector to do data mining; hence, mobile devices directly submit their sensed data (e.g., GPS coordinates, accelerometer reading, and user input). In the other category, the collector is interested in the aggregation statistics of a group of mobile devices' data instead of each individual's raw data. In many monitoring applications, such aggregation is periodically done to continuously identify interesting phenomena and track important patterns [3, 4]. For example, the average amount of daily exercise (which can be measured by motion sensors) people get can help infer public health conditions. The average level of air pollution and pollen concentration can help people plan their outdoor activities. In such systems, it is not necessary for a mobile device to submit its raw data. Instead, it can perturb the data value in some way as long as the collector can recover the correct aggregate statistics. For convenience, we call these two categories *raw-data-based sensing* and *aggregate-data-based sensing*, respectively. Due to the difference in the nature of data collected, they can be dealt with differently.

In this article, we address how to simultaneously protect privacy and provide incentives for raw-data-based and aggregate-data-based sensing systems. We introduce the challenges in providing privacy and incentive simultaneously,

Qinghua Li is with the University of Arkansas.

Guohong Cao is with Pennsylvania State University.

0163-6804/15/$25.00 © 2015 IEEE

review previous approaches, discuss their limitations, and propose new approaches to address those challenges.

Participatory Sensing Systems and Challenges

Participatory Sensing

In a participatory sensing system, there are many mobile nodes and a remote data collector. Mobile nodes are mobile devices carried by users or mounted in vehicles, and they produce sensing data such as locations, pictures, and sound. The collector is interested in collecting different data from mobile nodes (we use node and user interchangeably). Nodes communicate with the collector through cellular networks or WiFi. To facilitate data collection, the collector publishes sensing tasks to nodes, and the latter submit data reports according to the specification of sensing tasks. To incentivize participation, the collector remits credits to nodes for their contributed data. The credits can be converted to monetary rewards in the real world, or used to purchase service from the collector. A sensing task typically specifies the type of sensor readings needed (e.g., GPS coordinates), the conditions of sensing (e.g., in Manhattan during rush hour), the number of data reports needed from each node, the number of credits paid to each node, creation time, and expiration time.

Requirements for Privacy

The collector is untrusted. It is eager to infer the private information of nodes from the system. To protect privacy, the following properties are required:

- Given a readable data report, the collector does not know which node has submitted it. This is required since a data report may contain private information such as the node's location.
- Given a sensing task, the collector does not know if a given node has accepted the task. Sometimes, accepting a task itself causes privacy leakage. For instance, a task asks for the current noise level at Central Park. If the collector knows that Alice has accepted this task, even if it does not know which report is submitted by Alice, it can infer that most likely Alice is around Central Park now.
- The collector cannot link multiple readable reports submitted by the same node. If the collector can do so, and each of the linked reports includes when and where this report is generated, the collector can construct the node's movement path, which in turn helps the collector infer who the node is.
- The collector cannot link multiple sensing tasks accepted by the same node.

Requirements for Incentive

Nodes are greedy and want to earn as many credits as they can. For this purpose, they may abuse the system in many ways:

- A node may want to earn credits from a task without submitting data for it.
- A node may submit more reports for a task than allowed by the collector to earn more credits.
- Suppose there is a set of credentials for each particular task. Then a node may try to use the credentials for one task to earn credits from another task, because the latter task is paid at a higher rate.
- A node may spend a single credit token twice, causing the double-spending problem.
- A malicious node may compromise other nodes, steal their credentials, and use them to earn credits.

Such misbehavior must be mitigated.

Resource Constraints of Mobile Devices

Besides the requirements for privacy and incentive, another challenge is that mobile devices usually have limited power supply, computational resources, and bandwidth. Hence, a solution must be efficient in energy, computation, and communication. Another constraint is that peer-to-peer communications among mobile devices are expensive. Therefore, a solution should avoid or minimize the use of communications between nodes.

Approaches for Raw-Data-Based Sensing

Previous Approaches and Their Limitations

Privacy in participatory sensing has been addressed by a lot of work. The simplest solution is to suppress or anonymize a node's identity during tasking and reporting. However, the protection provided by this solution is weak. The problem is that if multiple reports can be linked as being from the same node (e.g., through analyzing the submission times of those reports or through the same pseudonym included in those reports), the node's identity might be inferred [5].

Even if the node's identifier is suppressed and timing analysis is mitigated, if it uses the same IP or medium access control (MAC) address when submitting multiple reports, those reports can also be linked easily. To address these problems, Mix networks (e.g., Mixmaster and Tor) can be used when a node submits data reports. Mix networks enable hard-to-trace communications by sending a message through a chain of Mix nodes (proxy servers on the Internet). Each Mix node receives messages from multiple previous hops, randomly shuffles them, and forwards them to the next hop. In this way, the collector does not know the real source of a request or report. Dynamically changing the IP and MAC addresses also helps protect privacy.

Combining these techniques, AnonySense [6], a general-purpose privacy-preserving framework of participatory sensing, has been proposed. It delivers sensing tasks to anonymous nodes, and collects unlinkable data reports from anonymous nodes. DeCristofaro *et al.* [7] aim at cryptographic treatment of privacy and propose a different framework. In this framework, external entities can query specific users' data, and the

To facilitate data collection, the collector publishes sensing tasks to nodes, and the latter submit data reports according to the specification of sensing tasks. To incentivize participation, the collector remits credits to nodes for their contributed data.

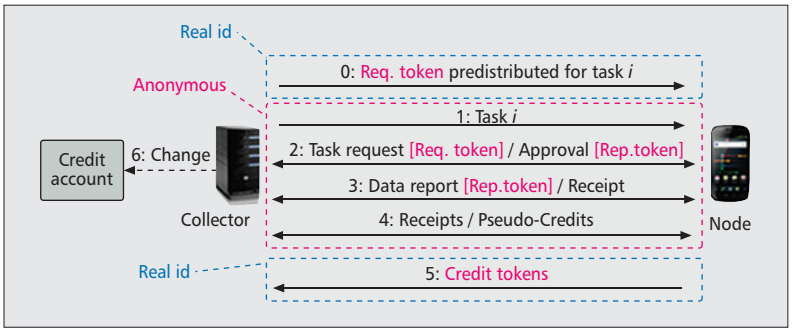

Figure 1. Providing privacy-preserving incentives [2].

framework can hide which mobile node matches a query.

Note that even if tasking and reporting are anonymous, the reported data itself may still cause privacy leakage. For instance, if a report includes a user's home as the location, the collector can link the report to the user. This problem has been addressed by many other techniques that are orthogonal to the approaches discussed in this article.

Several incentive schemes have been designed for participatory sensing to promote participation by paying credits to users. In [8], gaming and reverse auction theories are used to optimize the payment for a task. For example, each user bids for a sensing task with an expected payment, and the collector preferably assigns the task to those users with low bids to minimize the total payment. An optimal solution is found in which each user bids according to a certain strategy.

However, previous approaches address privacy and incentive *separately*, and thus cannot effectively promote user participation. It is challenging to address these two issues simultaneously. One option is to use blind signature to implement privacy-preserving credits, since it has been widely used for anonymous electronic payment. However, blind signature cannot *directly* solve the problem, since a malicious user that has compromised other users can steal and spend their credits without being detected. Other anonymous credential systems cannot be directly used for similar reasons. Privacy-preserving mechanism design and auctions can protect participants' types and valuations of a good, but they cannot be applied in mobile sensing to protect users' interest in sensing tasks.

It is natural to consider the combination of a privacy protection scheme and a credit-based incentive scheme to address both issues, but as pointed out in [2], the problem is not as simple as it appears. Specifically, when anonymity is provided as the protection for privacy, a greedy user can use different anonymous identifiers to submit unlimited data reports for the same sensing task (which may not be desirable) and earn unlimited credits. Also, a malicious node can compromise other nodes and use their credentials to earn credits without being detected. To address these problems, new designs are required to simultaneously address privacy and incentive.

Providing Privacy-Preserving Incentives

We proposed a solution for providing privacy-aware incentives [1, 2]. Besides satisfying the privacy requirements, the solution ensures that, for a task paid at the rate of c credits per node ($1 \le c \le C$, where C is a system parameter), one node cannot earn more than c credits from this task.

The approach relies on a set of tokens (including task *request token*, *report token*, receipt, and credit token) to achieve the goals of incentive and privacy. Each node predetermines the tokens it will use to process each future task, and commits to the collector that it will really use them. Later, when a node processes a sensing task using the predetermined tokens, the collector will verify that those tokens have been appropriately committed. The design of protocol and commitment guarantees that no node can misuse tokens to earn more credits than it should. To protect privacy, tokens and their commitments are constructed and used in a privacy-preserving way.

Basic Protocol: A request token for each future task is predistributed to each node before the task is created. At random intervals, each node, say Alice, anonymously retrieves tasks from the collector. If Alice wants to submit data for a task, she anonymously sends a request to the collector. This request contains her request token for this task. If the collector approves this request, it sends back an approval message and issues a number of report tokens to Alice. At this time, the task is assigned to Alice. Alice collects data in the way specified by the task. Then she anonymously submits each data report attached to a report token, and receives a receipt for each report. An independent communication session is used to submit each report. After Alice submits all reports for a task, she anonymously submits the receipts of this task to the collector to redeem credits. The collector issues Alice some pseudo-credits, which are transformed into credit tokens by Alice. For each credit token, Alice waits a random time and then deposits the token with the collector. The collector maintains a credit account for each node in the system, and updates Alice's credit account accordingly. The collector ensures that each credit token can only be spent once, which solves the double-spending problem. Alice uses her real ID to obtain the request token. In this way, the collector can make sure that only one request token for each task is issued to Alice, such that Alice cannot request a task multiple times. When Alice deposits credit tokens, she also uses her real ID.

Credit Token: Since the user uses her real identity to deposit a credit token, and the collector knows the data report from which the corresponding pseudo-credit is earned, it is necessary to break the link between the credit token and the pseudo-credit such that the collector cannot know the data report from which the credit is earned. Blind signature is used to address this challenge. A blind signature scheme [9] enables a user to obtain a signature from a signer on a message without exposing the message content to the signer. In this approach, a credit token

consists of a random token identifer m chosen by the node and its blind signature σ signed by the collector. To obtain a credit token, the user chooses m, blinds it with a random blinding factor, and sends the blinded message m' to the collector. The collector signs on m' using a standard algorithm (e.g., Rivest, Shamir, and Adleman, RSA), and passes the signature σ' (i.e., pseudo-credit) back to the user. The user removes the blinding factor from σ' and obtains a valid signature σ on m. This process ensures that the collector cannot link $\langle m, \sigma \rangle$ to m' or σ'.

Request Token: Alice gets the request token when she communicates with the collector using her real ID, but when Alice uses a request token to request a task, it must be ensured that the collector does not know Alice is the requester. To achieve this goal, partially blind signature (PBS) [10] is used to construct request tokens. PBS is similar to blind signature, except that it allows the signer to put some common (not identifiable) information into the signature. A request token consists of a random token identifier τ, task index i (common information), and the collector's PBS over them. To get the token, Alice chooses τ, and then Alice and the collector run a PBS protocol during which Alice obtains the PBS. The property of PBS ensures that the collector cannot link the PBS or request token to Alice. Besides, the signature binds the token to a specific task, which means Alice cannot use it to process other tasks.

Report Token and Receipt: Multiple reports submitted by the same node should not be linkable. This means that the report tokens obtained in the same session should not be linkable to each other. Since a node's receipts for the same task are submitted together, we must also break the link between a report and its receipt. To achieve these goals, the report token and receipt are also constructed using PBS schemes in a similar way as a request token.

Mitigating Token Misuse: Tasks are indexed as 1, 2, 3, … in the order of their creation time, and grouped into *task windows* of size W. The first (second) W tasks belong to the first (second) window, and so on. Tokens are managed based on task windows. Let us consider just one window without loss of generality.

Before any task in this window is created, for each task i in the window, each node predetermines the C credit token identifiers $m_1, m_2, \ldots, m_C$ it will use for this task. The node commits to the collector that it will use these credit tokens for this task. To do so, it builds an extended Merkle tree [2] over $m_1, m_2, \ldots, m_C$, and then obtains a PBS from the collector over the root τ of the tree and task index i. Here, $\langle \tau, i, PBS \rangle$ is a commitment, and it is also used as the node's request token for task i. Given τ, the identifiers of the receipts the node will use in this task are also determined. The node also binds $m_1, \ldots, m_C$ to its own real ID using another extended Merkle tree rooted at α.

When a node submits the receipts of task i to redeem c credits, the collector verifies the relation between identifiers of the receipts and τ.

	Type I $n = 1$ $c = 1$	Type II $n = 1$ $c = 256$	Type III $n = 256$ $c = 256$	Type IV $n = 256$ $c = 1$
Node (Nexus S)	90 ms	95 ms	116 ms	112 ms
Collector (laptop)*	10 ms	14 ms	37 ms	33 ms

*The time needed to process a task for each node.

Table 1. The average running time of processing a task on a Nexus S phone and a laptop [2].

Then, through a blind signature scheme, the node obtains c blind signatures for $m_1, \ldots, m_c$ from the collector, and they form the c credit tokens. To prevent the node from abusing the unused identifiers $m_{c+1}, \ldots, m_C$, the node needs to reveal these unused identifiers to the collector, as well as the proof that they are included in the extended Merkle tree rooted at τ. Each node that does not submit data for a task also reveals its credit token identifiers for this task. When a node deposits a credit token $\langle m, SIG(m) \rangle$, it also sends the appropriate elements of the extended Merkle tree rooted at α to prove that m has been bound to its real ID in the token distribution phase.

This process ensures that, given a request token (τ) or receipt (β), the credit tokens (m) which can be obtained and who can deposit these tokens are determined. This means that even if an attacker can compromise other nodes, it cannot use their tokens/receipts to obtain credit tokens that can be deposited to its own account.

This scheme has very low computation cost, as shown in Table 1.

Approaches to Aggregate-Data-Based Sensing

For aggregate-data-based sensing, when it is important to protect what sensing tasks a node has taken, we can first use the approach for raw-data-based sensing to collect the participants' data and then compute the aggregate. In the following, we consider the case where it is not sensitive if a node has taken a certain task, but it is important to protect the content of each node's data (e.g., the daily amount of exercise). In this case, if we can design an *aggregator oblivious* aggregation scheme (such a scheme ensures that the aggregator learns the desired aggregate statistics but nothing else, e.g., any individual node's data value), nodes can use their real ID to communicate with the collector (i.e., aggregator) without privacy leakage. Then credits can be directly paid to nodes, and the requirements on incentive can easily be satisfied. In the following, we focus on how to do aggregator oblivious data aggregation.

Previous Approaches and Their Limitations

Homomorphic encryption is widely used in privacy-preserving data aggregation. Homomorphic encryption allows certain computations over ciphertexts to be done in such a way that when

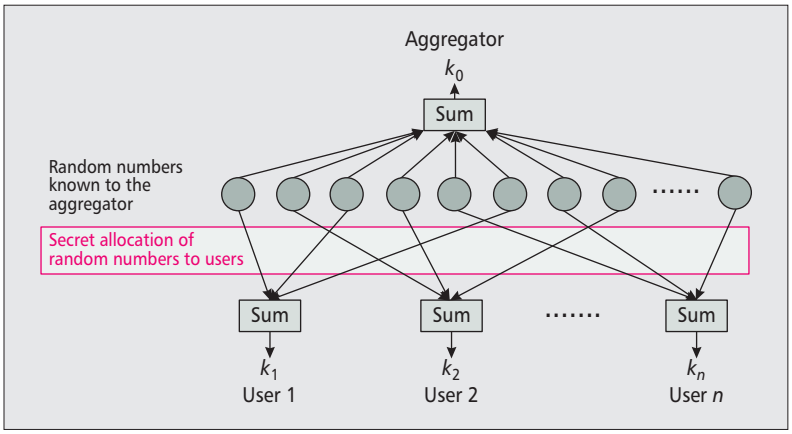

Figure 2. The intuition behind the basic encryption method for sum aggregation [3]. The aggregator computes the sum of a set of random numbers as the decryption key. These numbers are secretly allocated to the users, and each user computes the sum of its allocated numbers as the encryption key. The aggregator does not know which random numbers are allocated to each user, and thus does not know any user's key.

the encrypted result is decrypted, it matches the result of computations over plaintext. For example, suppose c_1 and c_2 are the ciphertexts of m_1 and m_2. Then one may decrypt the sum of ciphertext $c_1 + c_2$ to get the sum of plaintext $m_1 + m_2$. To use such encryption algorithms in aggregation, user A can encrypt his data and pass the result to user B. User B encrypts her own data, adds the ciphertext to A's ciphertext, and passes the result to user C. C does similar operations as B. Finally, the ciphertext is passed to the aggregator, which simply decrypts it to get the sum of all users' data. However, in most previous homomorphic encryption-based approaches, all users encrypt data with the same key, which is also known to the aggregator. If the communications between users can be intercepted, the aggregator will be able to decrypt each user's data value.

A similar but more complex approach is secure multi-party computation (SMC). It allows multiple parties to jointly compute a function (e.g., sum) over their inputs but keep these inputs private. However, most SMC algorithms are interactive and require parties to communicate with each other. This is impractical in participatory sensing.

Several recent constructions use the Paillier cryptosystem. It is a homomorphic encryption scheme, but users use different keys to encrypt their data, and the aggregator cannot decrypt any individual's data. The sum aggregation scheme in [11] divides the decryption key into portions and distributes them to nodes. The aggregator collects all nodes' ciphertexts, multiplies them together, and sends the aggregated ciphertext back to all nodes. Each node decrypts a share of the sum aggregate and sends it to the aggregator, which then obtains the final sum. However, this scheme requires multiple rounds of interaction between nodes and the aggregator. Other schemes based on the Paillier cryptosystem rely on inter-node communications, which is not practical in participatory sensing.

To remove multi-round communications and inter-node communications, Shi *et al.* [12] proposed a sum aggregation scheme assuming that the decisional Diffie-Hellman problem is hard. In this approach, each node i ($i = 1, \ldots, n$) gets an encryption key k_i, and the collector gets a decryption key k_0, which satisfies that their sum is zero. Each node i encrypts its data x_i by computing $g^{x_i}H(t)^{k_i}$ and sends the ciphertext to the aggregator. The aggregator computes the product of all ciphertexts and multiplies it by $H(t)^{k_0}$, deriving $g_{\Sigma x_i}$. Then it solves the discrete log by trying every possible value of the sum until finding a matching one. This approach has a simple communication model, but the decryption needs to traverse the possible plaintext space of sum, which is inefficient for large plaintext space.

These approaches heavily rely on computationally expensive public key cryptography and hence are too expensive for resource-constrained mobile devices. Moreover, none of the previous schemes on stream data aggregation considers the Max/Min aggregate (i.e., the maximum/minimum value), which is also important.

Efficient Data Aggregation Schemes

To address the limitations, we proposed an efficient stream aggregation scheme for sum based on lightweight symmetric key cryptography, and extended it to support Max/Min [3, 4]. It is aggregator oblivious.

Based on a semantically secure additive homomorphic encryption algorithm [13], our basic encryption method for sum aggregation works as follows (see the basic idea in Fig. 2). Suppose each user's data value is an integer in range $[0, \Delta]$. Let n denote the number of users. In the initial setup phase, a key dealer generates a pool of nc different secrets, and divides them into n random disjoint subsets, with c secrets in each subset. It assigns all secrets to the aggregator, and assigns one subset to each user. Let $\mathcal{S}$ denote the set of all secrets, and $\mathcal{S}_i$ denote the subset assigned to user i. If no trusted key dealer is available, the setup phase can also be done through a standard secure multi-party protocol. In aggregation period t, user i computes $f(s, t)$ for each secret s in $\mathcal{S}_i$. Here f is a pseudorandom function that can be implemented using HMAC. Then user i sets his encryption key k_i^t as the sum of $f(s, t)$ modulo a large integer M. He encrypts his data x_i by computing the sum of x_i and k_i^t modulo M, and sends the ciphertext to the aggregator. The aggregator computes $f(s, t)$ for each secret in $\mathcal{S}$, and generates its decryption key k_0^t as the sum of $f(s, t)$ modulo M. It decrypts the sum of users' data by adding their ciphertexts and subtracting k_0^t from the result. Since $\mathcal{S}$ is the union of all users' secret set $\mathcal{S}_i$, the correct sum is derived.

The confidentiality of each user's data relies on the fact that the aggregator does not know the subset of secrets assigned to any specific user. If sufficient secrets are assigned to each user, it will be computationally infeasible for the aggregator to guess the secrets of any specific user in a brute force way. With standard combinatorial techniques, we can derive the number of secrets needed by each user to achieve a required security level (e.g., 80-bit). In most practical settings, this number is very small (less than 10), which means very low computation

cost at mobile devices. The computation cost of the aggregator can also be significantly reduced [3]. Measurements on Nexus S phones and a Windows laptop showed that this scheme runs at least one order of magnitude faster than existing schemes (Table 2). The scheme can easily achieve differential privacy [14] (which is secure against adversaries with arbitrary auxiliary information) with very low noise in the sum [15].

Dealing with Dynamic Joins and Leaves: When a user leaves, her secrets should be removed from the aggregator's secret pool or redistributed to other users, such that the aggregator can still get the correct sum. Some of the secrets may be redistributed to users compromised by the aggregator, and the aggregator knows that these secrets belonged to the leaving user. This reduces the aggregator's uncertainty about the secrets used by the leaving user and other users, and decreases the security level of all users. To maintain the required security level, when a user leaves, a new set of secrets should be distributed to all remaining users, which means high communication cost. Similar issues exist when a user joins.

Redundancy in Security: We proposed to address this problem through maintaining redundancy in security [4]. The basic idea is to let each user maintain higher than l-bit security by using more secrets than required by l-bit security. When a users leaves, this user's secrets together with part of several other users' secrets are removed from the aggregator's secret pool, such that the aggregator does not know which of them belong to the leaving user. Those affected users are notified to remove the corresponding secrets from their secret pool. This is carefully done such that each remaining user still maintains sufficient secrets for the required security level. Join is dealt with similarly.

Overlapped Grouping: Intuitively, grouping can be used to efficiently deal with dynamic joins and leaves. In a naïve solution, the key dealer divides users into small disjoint groups, and applies the basic encryption method to each group independently. The aggregator can decrypt the sum of each group, and add them together to obtain the sum of all users' data. When a user joins or leaves a group, only the users in this group and the aggregator are redistributed secrets, which means low communication cost. However, since the aggregator knows the sum of each group, the solution is not aggregator oblivious. Also, if differential privacy is needed, sufficient noise should be added to the sum of each group, which means a large noise accumulated in the final sum. To address these problems, an overlapped grouping technique is proposed in [15] that guarantees aggregator obliviousness.

	n	10^3	10^4	10^5	10^6
Encryption (Nexus S)	Our scheme	2.4 ms	2.1 ms	1.9 ms	1.4 ms
	EXP	90 ms	90 ms	90 ms	90 ms
Decryption (laptop)	Our scheme	24 μs	18 μs	15 μs	12 μs
	EXP($\Delta = 10^2$)	1.8 s	5.6 s	18 s	56 s
	EXP($\Delta = 10^3$)	5.6 s	18 s	56 s	177 s
	EXP($\Delta = 10^4$)	18 s	56 s	177 s	560 s
	EXP($\Delta = 10^5$)	56 s	177 s	560 s	1770 s

Table 2. The running time of our Sum protocol and EXP [12] on a Nexus S phone and a laptop [4].

Conclusions and Open Challenges

Although participatory sensing has many useful applications, privacy concerns and lack of incentives prevent its large-scale deployment. In this article, we address how to protect privacy and provide incentives to users simultaneously. Focusing on raw-data-based sensing and aggregate-data-based sensing, we review previous approaches and point out their limitations. We also propose novel privacy-aware incentive schemes for raw-data-based sensing and efficient approaches for aggregate-data-based sensing.

Open Challenges: The data collector relies on truthful sensing data collected from nodes to identify important patterns and derive useful statistics. However, dishonest nodes may manipulate their sensing data for benefits. A selfish node may report faked data without doing the actual sensing; a malicious node may manipulate its data values to "pollute" the aggregate data collection. To ensure the usefulness of collected data, it is crucial to mitigate data forgery attacks. How to thwart data forgery under the framework of privacy-aware incentive is an open challenge.

For raw-data-based sensing, one new attack on privacy is a credit-based inference attack. Specifically, the collector may infer if a node has submitted a data report for a task from the number of credits the node has earned. For example, suppose there are three tasks paid at a rate of 1, 2, and 5 credits. If a node has earned 3 credits, the collector can infer that it has taken the first two tasks. For another example, suppose the collector has published 10 tasks, 5 of which require a data report from Central Park, and each task is paid for one credit. If a node Bob has earned 6 credits, the collector can infer that Bob has submitted data for at least one of those 5 tasks. Thus, the collector knows that Bob has been to Central Park. How to address such attacks is also an open challenge.

Existing game-theory-based incentive schemes for participatory sensing have studied methods for setting the number of credits paid to each reporting node. How to integrate them into the framework of privacy-preserving incentive is another open problem.

Acknowledgment

This work was supported in part by the U.S. National Science Foundation (NSF) under grant number CNS-1320278.

Existing game-theory-based incentive schemes for participatory sensing have studied methods for setting the number of credits paid to each reporting node. How to integrate them into the framework of privacy-preserving incentive is another open problem.

References

[1] Q. Li and G. Cao, "Providing Privacy-Aware Incentives for Mobile Sensing," *Proc. IEEE PerCom*, 2013.
[2] Q. Li and G. Cao, "Providing Efficient Privacy-Aware Incentives for Mobile Sensing," *Proc. ICDCS*, 2014.
[3] Q. Li and G. Cao, "Efficient and Privacy-Preserving Data Aggregation in Mobile Sensing," *Proc. IEEE ICNP*, 2012.
[4] Q. Li, G. Cao, and T. F. Porta, "Efficient and Privacy-Aware Data Aggregation in Mobile Sensing," *IEEE Trans. Dependable Secure Computing*, vol. 11, no. 2, 2014, pp. 115–29.
[5] J. Krumm, "Inference Attacks on Location Tracks," *Proc. 5th Int'l. Conf. Pervasive Computing*, 2007, pp. 127–43.
[6] C. Cornelius *et al.*, "Anonysense: Privacy-Aware People-Centric Sensing," *Proc. ACM MobiSys*, 2008, pp. 211–24.
[7] E. De Cristofaro and R. Di Pietro, "Preserving Query Privacy in Urban Sensing Systems," *Proc. ICDCN*, 2012, pp. 218–33.
[8] D. Yang *et al.*, "Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing," *Proc. ACM MobiCom*, 2012.
[9] D. Chaum, "Blind Signatures For Untraceable Payments," *Proc. CRYPTO '82*, 1982.
[10] M. Abe and T. Okamoto, "Provably Secure Partially Blind Signatures," *Proc. CRYPTO*, 2000, pp. 271–86.
[11] V. Rastogi and S. Nath, "Differentially Private Aggregation of Distributed Time-Series with Transformation And Encryption," *Proc. ACM SIGMOD*, 2010.
[12] E. Shi *et al.*, "Privacypreserving Aggregation of Time-Series Data," *Proc. Network and Distrib. Sys. Security Symp.*, 2011.
[13] C. Castelluccia *et al.*, "Efficient and Provably Secure Aggregation of Encrypted Data in Wireless Sensor Networks," *ACM Trans. Sensor Networks*, vol. 5, no. 36, 2009, pp. 20:1–20:3.
[14] C. Dwork *et al.*, "Calibrating Noise to Sensitivity in Private Data Analysis," TCC, 2006.
[15] Q. Li and G. Cao, "Efficient Privacy-Preserving Stream Aggregation in Mobile Sensing with Low Aggregation Error," *Proc. 13th Privacy Enhancing Technologies Symp.*, 2013.

Biographies

QINGHUA LI [M] received a B.E. degree from Xian Jiaotong University, an M.S. degree from Tsinghua University, and a Ph.D. degree from Pennsylvania State University. In 2013, he joined the University of Arkansas, where he is currently an assistant professor in the Department of Computer Science and Computer Engineering. His research interests are security and privacy in networked and mobile systems including mobile sensing, smart grid, and mobile cloud computing.

GUOHONG CAO [F] received a B.S. degree in computer science from Xian Jiaotong University and a Ph.D. degree in computer science from Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at Pennsylvania State University, where he is currently a professor. His research interests include wireless networks, wireless security, smartphones, vehicular networks, wireless sensor networks, and distributed fault-tolerant computing. He has served on the Editorial Boards of *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Wireless Communications*, and *IEEE Transactions on Vehicular Technology*, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS '09, MASS '10, and INFOCOM '13. He was a recipient of the NSF CAREER award in 2001.