Privacy-Aware and Trustworthy Data Aggregation in Mobile Sensing

Jingyao Fan*, Qinghua Li† and Guohong Cao*
*Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802
†Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72701
Email: {jfan,gcao}@cse.psu.edu, qinghual@uark.edu

***Abstract*—With the increasing capabilities of mobile devices such as smartphones and tablets, there are more and more mobile sensing applications such as air pollution monitoring and healthcare. These applications usually aggregate the data contributed by mobile users to infer about people's activities or surroundings. Mobile sensing can only work properly if the data provided by users is adequate and trustworthy. However, mobile users may not be willing to submit data due to privacy concerns, and they may be malicious and submit forged data to cause damage to the system. To address these problems, this paper proposes a novel privacy-aware and trustworthy data aggregation protocol for mobile sensing. Our protocol allows the server to aggregate the data submitted by mobile users without knowing the data of individual user. At the same time, if malicious users submit invalid data, they will be detected or the polluted aggregation result will be rejected by the server. In this way, the malicious users' effect on the aggregation result is effectively limited. The detection of invalid data works even if multiple malicious users collude. Security analysis shows that our scheme can achieve the trustworthy and privacy preserving goals, and experimental results show that our scheme has low computation cost and low power consumption.**

I. Introduction

Mobile devices, especially smart phones, are becoming more and more important in our daily life. With enhanced sensing capabilities such as camera, microphone, accelerometer, GPS, etc. and communication capabilities such as 4G, WiFi and Bluetooth, mobile devices can provide useful data to infer about our daily life (location, health, activity, etc.) as well as the environment (air pollution, noise, traffic, etc.), and hence help improve the quality of life.

There are many popular mobile sensing applications such as traffic monitoring [1], pollution monitoring [2], health monitoring [3] and survey applications that use participants as sensors [4]. In these applications, aggregation of the data submitted by mobile users are useful for inferring about the environment and people's life or identifying important phenomena in a certain area. For example, the average time people spend on social media can help us learn about people's social life pattern. The number of people who have caught a certain flu can help us learn about the flu and plan for vaccines. The maximum moving speed of road traffic during rush hours can provide important information about traffic jam and help people adjust their schedules and ways of transportation accordingly.

This work was supported in part by the National Science Foundation (NSF) under grant CNS-1320278 and CNS-1421578.

Although these mobile applications are very useful, in many cases, the collected data may raise privacy concerns. Participants may not trust the aggregation server or anyone else to see their data in plaintext. For example, to monitor the flu trend in a certain area, the server needs to collect information about the number of users infected with flu. However, the infection status of a user is sensitive and he may not be willing to participate in any sensing task if his privacy cannot be well protected.

Another important issue is whether the data provided by mobile users should be trusted or not. It is possible that a user's device malfunctions and provides wrong sensing results to the server, or even worse, the mobile user is malicious and intentionally forges data to mislead the server. In either case, the server ends up with wrong aggregation statistics and hence makes wrong inferences.

There are lots of existing works on preserving user privacy in data aggregation [5], [6], [7], [8], [9]. However, they assume that mobile users are trustworthy. As a result, a single forged data can make the aggregation result significantly deviate from the true value and become useless. On the other hand, there are also many existing works on trustworthy data aggregation in wireless sensor networks [10], [11], [12], [13]. However, they do not consider user privacy, and we are not aware of any data aggregation protocol that addresses both privacy and trustworthiness issues.

For general mobile sensing applications, some recent research [14], [15] addresses trust and privacy simultaneously. Their root of trust is that multiple users can measure the same data at the same time and location, and thus a measurement that deviates from the majority can be deemed as untrusted. However, this is not true for a broad range of aggregation applications, where each user generates his own data independently and the validity of his data cannot be judged from other users' data. For such applications, their schemes may mistakenly punish honest users with diverse data. Moreover, since they target general mobile sensing applications instead of data aggregation, only anonymity is provided for privacy protection. However, anonymity-based approaches are known to be vulnerable to tracking attacks [16], [17], [18].

In this paper, we address privacy and trustworthiness issues together for data aggregation in mobile sensing. We provide

data trustworthiness by ensuring that every user must submit a valid data value predefined by the aggregation server. For instance, if a server wants to count the number of users infected with a flu, it can predefine two values 1 (which means *infected*) and 0 (which means *uninfected*) for each user to choose from. The proposed scheme ensures that a malicious user who submits values other than 1 and 0 will be detected by the server, which means that the malicious user cannot submit 1,000 to convince the server that many people have caught this flu. Different from existing research on preserving privacy by submitting data in an anonymous way, the proposed protocol preserves privacy by hiding users' data from the server, i.e., user's data is not revealed to the server in clear-text.

The contributions of this paper are as follows:

- We propose a novel concept *data value vector* to restrict the behavior of malicious users. Instead of submitting any value he wants, a malicious user has to pick a value from the predefined data value vector. This vector can effectively limit the effect of malicious users on the final aggregation result.
- We propose a novel privacy-aware data validation technique which enables the aggregation server to validate if each user has submitted valid data from the data value vector, without knowing the user's data in clear-text. The validation technique works even if multiple malicious users collude.
- We design a Privacy-Aware and Trustworthy data Aggregation (PATA) protocol for mobile sensing. PATA nontrivially combines the aforementioned two techniques with blind signature and homomorphic encryption to prevent privacy leakage and mitigate invalid data injected by malicious users. If a malicious user submits invalid data, he will be detected or the polluted aggregation result will be rejected by the server. To the best of our knowledge, this is the first data aggregation protocol to address both privacy and trustworthiness at the same time.
- We have implemented PATA and evaluation results show that its computation cost and energy cost are very low.

The rest of the paper is organized as follows. Section II introduces related work. Section III describes our system model, threat model and cryptographic primitives and provides an overview of our protocol. Section IV explains our proposed protocol in detail. Section V and Section VI present security analysis and performance evaluation, respectively. Section VII and Section VIII present discussions and conclusions.

II. Related Work

Secure data aggregation schemes [12] [10] have been proposed to get a good approximation of the true aggregation result with the presence of compromised intermediate aggregation nodes. However, they assume raw data from sensors to be trustworthy and they reveal sensing data to the aggregators. Other research that considers the presence of untrusted mobile users focuses on how to identify malicious users to avoid future attacks [11], [13]. However, they reveal user's identities and their data to other parties.

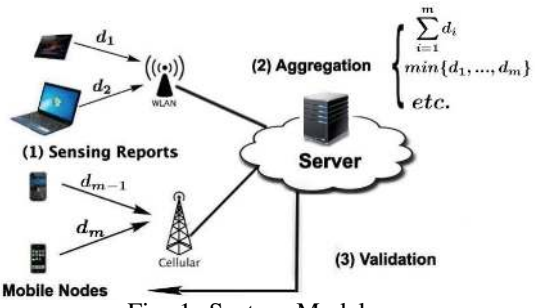

Fig. 1: System Model

Many privacy preserving techniques have been proposed [8], [5], [6], [7], [9]. Among them, [8] and [6] focus on sum aggregation and they use cryptography algorithms to ensure that the server can only decrypt the sum of participants' data but not any individual data. Shi *et al.* [5] propose a novel slicing and mixing technique and combines it with binary search to support count as well as a wide range of non-additive aggregation. Li *et al.* [9] achieve privacy for sum, count, min and max aggregation by using homomorphic encryption. Groat *et al.* [7] propose a non-cryptographic method that adds camouflage values to obfuscate users' data. However, all of the above schemes assume the users are trustworthy and these schemes will generate wrong aggregation results with the presence of malicious users.

ARTSense and TAPAS [14], [15] address data trustworthiness in an anonymous way. However, they focus on tasks with similar sensing factors (time, location, etc) and similar sensing results which limit the application of their works. Moreover, even with anonymity, users' privacy can still be violated if the data submitted is so sensitive that the server can infer the identity or location of the user. Different from their work, our solution identifies malicious users from the encrypted sensing data. VPA [19] uses homomorphic MAC to achieve integrity and let the participants compute the aggregation result collaboratively without revealing individual sensor readings. Different form their work, our scheme doesn't require peer-to-peer communication.

III. Preliminaries

A. System Model and Assumptions

Fig. 1 shows our system model, which consists of an aggregation server and multiple Mobile Nodes (MNs). MNs are mobile devices like smartphones and tablets. The server wants to get the aggregate statistics from m MNs. In the following, we focus on the sum aggregation, but our aggregation protocol for sum can be easily extended to other aggregate statistics such as count, average, histogram, and even maximum and minimum [9]. Before the server and MNs start any sensing task, the server first chooses the *data value vector* according to the application. The data value vector is a novel concept we propose to provide data trustworthiness and restrict the behavior of malicious users. It is a vector of all data values that are considered to be valid by the server, sorting in increasing order (More details about data value vector will be given in Section VII). Each sensing task is processed in the following

TABLE I: Notations

m	Number of mobile users in the system
n_i	The i^{th} MN
(e, N)	RSA public key of server
(d, N)	RSA private key of server
g	A small prime
h	A random number in $\mathbb{Z}_N$, coprime to N
k_0	Decryption key of the server for aggregation
k_i	Encryption key of n_i for aggregation
o_i	Original report of user i
s_i	Signed report of user i
c_i	Encrypted report of user i
t	The taskID of current task
$H()$	A cryptographic hash function
$D = [d_1, d_2, ..., d_w]$	*Data value vector*, a vector of all valid values of a sensing result, sorted in increasing order

three steps. First of all, each MN reports its sensing data to the aggregation server via WiFi or 3G (Step 1). On the server side, all data are collected to calculate the aggregate statistics (Step 2). Then the server validates each MN's report (Step 3). We focus on how to protect users' privacy throughout the process and how to achieve data trustworthiness in Step 3.

B. Threat Model

1) Threats from the Server: The server may be curious about some private information of MNs which can be inferred from the sensing reports. For example, the server can infer that the MN has been infected with flu from a report with high body temperature. On the other hand, the server's primary task is to obtain the aggregate statistics. Therefore, we assume that the server is honest but curious, and it will follow our protocol when processing sensing reports.

2) Threats from MNs: We cannot fully trust MNs. Mobile devices can malfunction sometimes and send invalid data, causing the server to have wrong aggregate statistics. Even worse, an MN may be malicious and sends invalid data to jeopardize the system intentionally.

Malicious MNs may collude with each other: a malicious MN sends an invalid report to the server during aggregation to cause disturbance to the aggregate statistics. Then during data validation, it can use a valid report that it obtains from another colluding MN who appears to be "honest". In this way, the malicious MN can affect the aggregation result without being discovered by the server.

3) Our Goal: First, we want to prevent the server from reading any individual's report, and in this way protect the privacy of MNs. Second, we aim to detect the injection of invalid data values (values not from the data value vector), and identify the malicious MNs who submit invalid data to derive wrong statistics.

C. Cryptographic Primitives

1) Blind Signature: Blind signature is a form of digital signature [20]. It can be implemented with common public key signing schemes, e.g., RSA and DSA. To get a blind signature on message x, the user combines it with a random "blinding factor". The blinded message x' is sent to the signer and the signer signs it using its private key. The resulting signature s' is sent back to the user. The user then removes the blinding factor from s' to get a signature s on x. s is identical to the signature generated by a normal signing protocol and thus can be verified with the signer's public key. This blind signature technique can ensure both unlinkability and unforgeability.

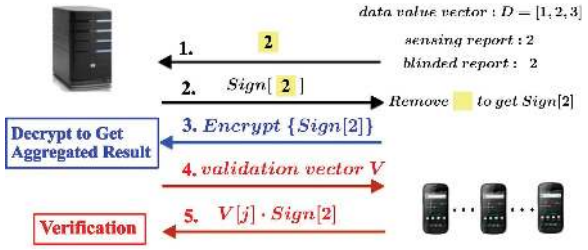

Fig. 2: An overview of PATA

D. Overview

Here we give an overview of our protocol to provide a general idea of how it works and the detailed solution will be presented in Section IV. Our approach has four phases: *Setup*, *Data Preprocessing*, *Data Aggregation*, and *Data Validation*.

Setup In this phase, a set of secrets are generated and assigned to the server and MNs, which will be used to generate encryption and decryption keys during the Data Aggregation phase (More details about secrets will be given in Section IV-A2). The secret distribution and key generation process ensure that the server can only decrypt the aggregated result but cannot decrypt any individual's report. Also, the server will generate system parameters, RSA keys and the data value vector and then broadcast them to all MNs.

Data Preprocessing (Step 1 and Step 2 in Fig. 2) Each MN picks a value $D[j]$ from the data value vector D based on its sensing result and uses $D[j]$ in its sensing report. For example, if the sensing result is 2.1 and the data value vector is $[1, 2, 3]$, the MN will use 2 as its data value in the report (More details about data value vector in Section VII). Then each MN gets its sensing report blindly signed by the server so that the server cannot see the data value. This signed report will be used in the following two phases by MNs. Since an MN cannot generate a signature itself, it will not be able to change the data value.

Data Aggregation (Step 3 in Fig. 2) In this phase, each MN encrypts the signed report it obtained during the Data Preprocessing phase using its encryption key and then sends it to the server. On receiving all ciphertexts, the server aggregates them together and then decrypts it using its decryption key to get the aggregated result. The server will not be able to decrypt any individual MN's report because their keys satisfy certain conditions.

Data Validation (Step 4 and Step 5 in Fig. 2) In this phase, the server validates if each MN has submitted a report with valid data from the data value vector and has followed our protocol honestly. Firstly, the server generates a *validation vector* V based on the data value vector and the taskID, and sends V to each MN. On receiving this validation vector, the MN multiplies its own signed report with the corresponding element in the validation vector. The result is sent to the server

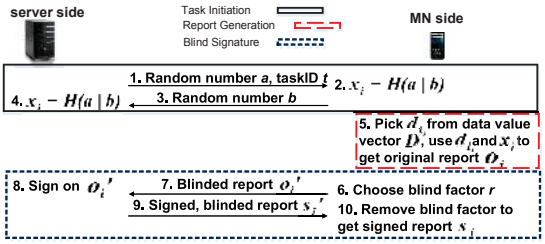

Fig. 3: Data Preprocessing

who can verify if the MN has cheated or not based on this result. The whole process is done with encryption, and thus the server is able to validate each MN's report without knowing the data value. More details will be shown in the next section.

IV. PROPOSED SCHEME

In this section, we present each phase of our protocol in detail, based on the notations shown in Table I.

A. Setup

This phase runs before the server and MNs start processing sensing tasks. It only needs to be done once.

1) System Parameters Generation: The server first generates the RSA public and private key pair (e, N) and (d, N), a small prime g and a random number $h \in \mathbb{Z}_N$ that is coprime to N. According to the application's sensing task, the server will generate a data value vector $D = [d_1, d_2, ..., d_l, ..., d_w]$, which includes all possible sensing values that are considered to be "valid" by the server, sorted in an increasing order. The public key (e, N), small prime g, random number h, and the data value vector are sent to all MNs.

2) Secret Distribution: To preserve privacy, the server's decryption key and MNs' encryption keys must satisfy certain conditions (For example, for sum aggregation, the sum of server's key and all MNs' keys should be 0). To achieve this goal, we adapt the scheme proposed in [9]. A trusted authority is responsible for generating and assigning secrets to the server and all MNs. For each sensing task, the server and MNs will use these secrets to generate their keys.

Let n_i $(i = 1, 2, ..., m)$ denote node i in the system. The trusted authority first generates mc random and different secrets and divides all the secrets evenly into m disjoint subsets. Let S denote the set formed by all mc secrets and S_i denote the i^{th} subset. Then the authority randomly selects q secrets from S to form a subset $\hat{S}$ and sends all secrets in $\hat{S}$ to the server. Then it divides secrets in $S - \hat{S}$ evenly into m disjoint subsets $\bar{S}_i$ and sends S_i and $\bar{S}_i$ to n_i. Obviously, $S = \cup_{i=1}^{m} S_i = (\cup_{i=1}^{m} \bar{S}_i) \cup \hat{S}$. Using these secrets, the server and MNs can generate their keys for each sensing task. (How keys are calculated from the secrets is shown in Section IV-C.)

B. Data Preprocessing

In this phase, a new round of sensing is initiated by the server and each MN generates its sensing report and obtains a blind signature for it. For n_i and the server, this phase proceeds in the following steps as shown in Fig. 3

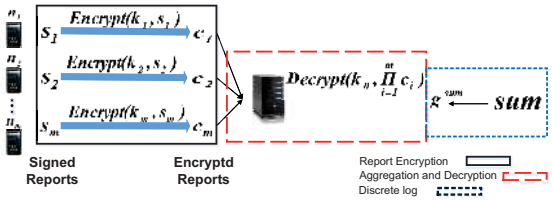

Fig. 4: Data Aggregation

Task Initiation When the server wants to collect sensing data for a new task, it generates a random number a and a unique taskID t and sends them to all MNs. On receiving a and t, n_i generates a random number b and sends it to the server. Both the server and n_i will compute x_i as $x_i = H(a|b)$. The server ensures that all x_i are different. (x_i for different MNs are different so that collusion can be prevented.)

Report Generation n_i does the sensing and examines the data value vector to pick a value d_{l_i} (suppose l_i is the index of d_{l_i} in D) that is closest to its true sensing data. Then it uses this value (d_{l_i}) in its original sensing report o_i: $o_i = g^{d_{l_i}} x_i \bmod N$.

Blind Signature n_i randomly chooses a number r which is relatively prime to N as a blinding factor to get a blinded report: $o_i' = g^{d_{l_i}} x_i r^e \bmod N$. o_i' is sent to the server and the server then signs it using its RSA private key: $s_i' = (o_i')^d \bmod N = (g^{d_{l_i}} x_i r^e)^d \bmod N$. The server sends this signature s_i' back to n_i and n_i removes the blinding factor to get the signed report s_i: $s_i = (s_i' r^{-1}) \bmod N = (g^{d_{l_i}} x_i)^d \bmod N$. The unlinkability property ensures that the server cannot know o_i and the unforgeability property ensures n_i cannot change the value d_{l_i} in its report.

C. Data Aggregation

In this phase, the server will aggregate all sensing reports without knowing each MN's data value. For n_i and the server, this phase is done in the following steps as shown in Fig. 4.

Report Encryption First, n_i calculates its encryption key for the current task t: $k_i = (\sum_{\sigma \in \bar{S}_i} H(f_\sigma(t)) - \sum_{\sigma \in S_i} H(f_\sigma(t))) \bmod N$. Here, f_σ is a member of the pseudo-random function family $\mathbb{F}_\lambda = \{f_\sigma : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{\sigma \in \{0,1\}^\lambda}$ indexed by σ. Then n_i encrypts the signed report s_i with k_i: $c_i = (g^{d_{l_i}} x_i)^d \cdot h^{k_i} \bmod N$. The encrypted report c_i is sent to the server.

Aggregation and Decryption First, the server generates its decryption key for task t: $k_0 = \sum_{\sigma \in \hat{S}} H(f_\sigma(t)) \bmod N$. We can easily verify that: $\sum_{i=0}^{m} k_i = 0$. After collecting all MNs' encrypted reports, the server multiplies them together and then decrypts the product using k_0 and all x_i chosen during Task Initiation of Data Preprocessing:

$$S = h^{k_0} \prod_{i=1}^{m} [(g^{d_{l_i}} x_i)^d \cdot h^{k_i}] \prod_{i=1}^{m} x_i^{-d} \bmod N$$
$$= \prod_{i=0}^{m} h^{k_i} (\prod_{i=1}^{m} g^{d_{l_i}})^d \bmod N = (g^{\sum_{i=1}^{m} d_{l_i}})^d \bmod N$$

Using its own RSA public key (e, N), the server can get: $S' = S^e \bmod N = g^{\sum_{i=1}^{m} d_{l_i}}$.

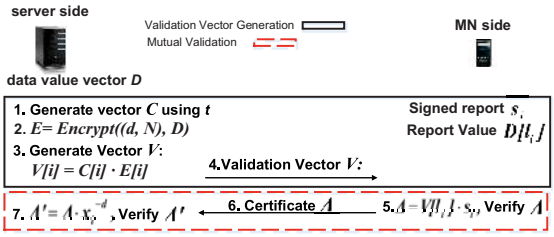

Fig. 5: Data Validation

Discrete log The server needs to compute the discrete log of S' base g to get the sum. For fast decryption, we use the low-cost algorithm proposed by Galbraith emphet al. [21]. The method requires average decryption time $(1.714+o(1))\sqrt{mP}$ (P is the plaintext space , which is roughly $(d_w - d_1)$).

D. Data Validation

In this phase, the server validates if each MN's report is valid in the following two steps as shown in Fig. 5.

Validation Vector Generation First, using the taskID t and its own RSA private key, the server generates a *certificate vector* C of the same length as the data value vector:

$$C = [(H(t))^d \bmod N, ..., (H(t))^d \bmod N]$$

The server encrypts the data value vector D with g and its RSA private key and gets an *encrypted value vector* E:

$$E = [(g^{-d_1})^d \bmod N, ..., (g^{-d_w})^d \bmod N]$$

The server then multiplies each item in C with the corresponding item in E and gets a *validation vector* V:

$$V = [(g^{-d_1} H(t))^d \bmod N, ..., (g^{-d_w} H(t))^d \bmod N]$$

Then V is sent to each MN.

Validation n_i knows d_{l_i} is the ${l_i}^{th}$ value in the data value vector. It multiplies the signed report s with the ${l_i}^{th}$ value in V to get a *validation certificate* Δ:

$$\Delta = (g^{d_{l_i}} x_i)^d \cdot (g^{-d_{l_i}} H(t))^d \bmod N = (H(t)x_i)^d \bmod N$$

Using the server's public key (e, N), the MN verifies that Δ is a valid signature over $H(t)x_i$. (Without this verification, the server can put different elements in vector C. Then it can map an MN's Δ to the corresponding element in C, thus knowing the MN's data value.) Then it sends Δ back to the server. The server knows the x_i assigned to n_i. It computes:

$$\Delta' = \Delta \cdot x_i^{-d} \bmod N = (H(t))^d \bmod N$$

Since the server knows $H(t)$, it can easily verify if the result Δ' equals to $(H(t))^d \bmod N$ or not. If Δ' does not equal to $(H(t))^d \bmod N$, the server detects that the MN did not follow our protocol (e.g., submitting an invalid report) and thus can be removed from the system.

E. Node Management

MNs may join or leave the system or the server may want to revoke a malicious MN's participation in future sensing tasks. To deal with leave, join and revoke, the *trusted authority* needs to update secrets for the server and MNs. However, for a large system with many MNs, the communication cost may be high. To reduce the communication cost, we adapt the ring-based interleaved grouping technique proposed in [22] on top of our protocol. All MNs are divided into q interleaved groups, with each node belonging to two groups. For each group of nodes, the authority assigns secrets to the server and MNs as shown in the Secret Distribution process. Let $S_{iA}, \bar{S}_{iA}$ and $S_{iB}, \bar{S}_{iB}$ denote the sets of secrets that n_i receives from group A and B (the two groups it belongs to), respectively. n_i merges the sets like this: $S_i = S_{iA} \cup S_{iB}, \bar{S}_i = \bar{S}_{iA} \cup \bar{S}_{iB}$. Suppose the sets of secrets the server gets from q groups are $S_1, S_2, ..., S_q$. The server merges them together: $\hat{S} = \cup_{i=1}^{q} \hat{S}_i$.

1) Join: The joining MN will be inserted into a random position in the ring. After insertion, the two groups that cover this position may violate the group size property, thus regrouping is needed. Then the authority will assign new sets of secrets to the adjusted groups and the server. Li *et al.* [22] proved that at most $4u$ nodes need to update keys (u is the size of the smallest group).

2) Leave: The leaving MN is removed from the ring. After deletion, the two groups that cover this position may violate the group size property and the overlap property. Thus, the trusted authority need to regroup. Then it will assign new secrets to the changed groups and the server. It has been verified in [22] that at most $6u$ MNs need to update their keys.

3) Revoke: If the server discovers an MN has cheated, it will revoke the MN from the system. To do revocation, the server sends this MN's id to the authority which will update secrets as the "Leave" process shown above.

V. Security Analysis

A. Privacy

To analyze how well our PATA can protect each MN's privacy, we first define a privacy attack game between an adversary and a challenger.

- The challenger generates a random number $y \in \{0, 1\}$. Then it runs the System Parameters Generation process and generates secret keys $k_0, k_1, ..., k_m$ such that $\sum_{i=0}^{m} k_i = 0$. It sends the system parameters and k_0 to the adversary.
- The adversary chooses a set of data values $\{d_1, ..., d_m\}$ and sends them to the challenger. The challenger chooses a corresponding set of data values $\{d'_1, ..., d'_m\}$ that satisfies $\sum_{i=1}^{m} d_i = \sum_{i=1}^{m} d'_i$. Then the adversary makes a sequence of the following queries to the challenger:
  Signing Query: The challenger picks a random number r relatively prime to N. If $y = 0$, the challenger sends tuple $(g^{d_1} r^e, ..., g^{d_m} r^e)$ to the adversary. If $y = 1$, the challenger sends $(g^{d'_1} r^e, ..., g^{d'_m} r^e)$ to the adversary.
  Encryption Query: If $y = 0$, the challenger returns $(g^{d_1} h^{k_1}, ..., g^{d_m} h^{k_m})$ to the adversary. If $y = 1$, the challenger returns $(g^{d'_1} h^{k_1}, ..., g^{d'_m} h^{k_m})$ to the adversary.
- The adversary outputs $\hat{y} \in \{0, 1\}$

We do not include the validation in the query because the validation certificates that the server gets from MNs do not contain any information about MNs' data. Let S denote the event that $\hat{y} = y$ and $Pr[S]$ denote the probability that event S occurs. We define the advantage of the adversary to be $|Pr[S] - 1/2|$.

Definition 1. *(Server Obliviousness). A scheme is server oblivious if no probabilistic polynomial-time adversary has more than negligible advantage in winning the above game.*

Server obliviousness is a strong privacy guarantee since it ensures that the server knows nothing about each individual user's data.

Theorem 1. *Our PATA protocol is server oblivious assuming that decisional Diffie-Hellman (DDH) is hard.*

Proof: Let $x \xleftarrow{R} X$ denote the action of assigning x a value sampled from the uniform distribution on set X. Let $\mathcal{P}()$ denote the system parameters generation oracle, $\mathcal{K}()$ denote the aggregation key generation oracle, $\mathcal{X}()$ denote the oracle generating random numbers $x_1, ..., x_j$ that satisfy $\prod_{i=1}^{j} x_i = \prod_{i=1}^{j} g^{d_i} r^e$ and $\mathcal{Z}()$ denote the oracle generating random numbers $z_1, ..., z_j$ that satisfy $\prod_{i=1}^{j} z_i = \prod_{i=1}^{j} g^{d_i} h^{h_i}$. Let Q_s and Q_e denote the tuples returned by the challenger in Signing Query and Encryption Query, respectively. We formally define a sequence of games $Game\ j$ $(j = 0, 1, 2..., m)$ based on the above game:

Game j. We model the adversary as a deterministic algorithm A and the attack game is described algorithmically as follows:

$$y \xleftarrow{R} \{0,1\}, (g, h, N) \leftarrow \mathcal{P}(), (k_0, ..., k_m) \leftarrow \mathcal{K}()$$
$$(d_1, ..., d_m) \leftarrow A(), r \xleftarrow{R} \{0,1\}^N, (x_1, ..., x_j) \leftarrow \mathcal{X}(g, r, d_1, ..., d_j)$$
$$(z_1, ..., z_j) \leftarrow \mathcal{Z}(g, h, k_1, ..., k_j, d_1, ..., d_j)$$
$$if\ y = 0 : Q_s \leftarrow (g^{d_1} r^e, ..., g^{d_m} r^e), Q_e \leftarrow (g^{d_1} h^{k_1}, ..., g^{d_m} h^{k_m})$$
$$else : Q_s \leftarrow (x_1, ..., x_j, g^{d_{j+1}} r^e, ..., g^{d_m} r^e)$$
$$Q_e \leftarrow (z_1, ..., z_j, g^{d_{j+1}} h^{k_{j+1}}, ..., g^{d_m} h^{k_m})$$
$$\hat{y} \leftarrow A(g, h, N, k_0, (d_1, ..., d_m), Q_s, Q_e)$$

Obviously, Game m is equivalent to the original game. Thus it is sufficient to prove that any polynomial-time adversary's advantage ($|Pr[S_m] - 1/2|$) in the above game is negligible. From Lemma 1 and Lemma 2 (See the Appendix), we have that $|Pr[S_m] - 1/2| = |Pr[S_m] - Pr[S_0]|$ is negligible. □

B. Trustworthiness

Definition 2. *(Pollution Attack). Malicious users submit invalid data that is not within the data value vector which can result in wrong aggregation statistics without being detected.*

To analyze how well PATA can protect against pollution attacks, we first define a pollution attack game between an adversary and a challenger.

- The challenger generates valid data values $\{d_1, ..., d_w\}$, RSA key pairs, a random value γ and sends $\{d_1, ..., d_w\}$ and (e, N) to the adversary.
- The adversary chooses a data value $\alpha \notin \{d_1, ..., d_w\}$ and makes the following queries to the challenger:
  Signing Query: The challenger returns $\alpha^d \bmod N$.
  Validation Query: The challenger returns $\{(\frac{\gamma}{d_1})^d \bmod N, ..., (\frac{\gamma}{d_w})^d \bmod N\}$
- The adversary outputs δ

Let S denote the event that $\delta = \gamma^d \bmod N$. We define the advantage of the adversary in winning the above game to be $Pr[S]$. It is not hard to see that S corresponds to the event that a malicious MN passes data validation after submitting invalid data.

Definition 3. *(Pollution Resistance). A scheme is pollution resistant if no probabilistic polynomial-time adversary has more than negligible advantage in winning the above game.*

Pollution resistance ensures that pollution attacks will be detected by the server through data validation.

Theorem 2. *Our PATA protocol is pollution resistant assuming that RSA signature is secure.*

Proof: We prove that if the adversary can win the above game, he can break RSA signature. Let $A()$ denote the adversary and $\mathcal{P}()$ denote the parameters generation oracle of the challenger. Let Q_s and Q_v denote the tuples returned by the challenger in Signing Query and Validation Query, respectively. We construct an algorithm $B(\theta)$ based on the pollution attack game:

$$(\{d_1, ..., d_w\}, d, e, N, \gamma) \leftarrow \mathcal{P}()$$
$$Q_s \leftarrow (\frac{\gamma}{\theta})^d \bmod N, Q_v \leftarrow \{(\frac{\gamma}{d_1})^d \bmod N, ..., (\frac{\gamma}{d_w})^d \bmod N\}$$
$$\delta \leftarrow A(\{d_1, ..., d_w\}, e, N, \gamma, Q_s, Q_v), output\ \delta$$

Breaking RSA signature means that we can generate $a^d \bmod N$ given $a \in \mathbb{Z}_N$. If the input to B is a, we can get the signature like this: $B(a)/Q_s = \gamma^d/(\frac{\gamma}{a})^d \bmod N = a^d \bmod N$. Therefore, winning the above game means breaking RSA signature. □

C. Collusion Attacks

Definition 4. *(Collusion Resistance). A scheme is collusion resistant if: 1) even when the server colludes with a set of MNs, server obliviousness still holds for the uncompromised MNs; 2) even when a set of MNs collude with each other, pollution resistance still holds.*

Theorem 3. *Our PATA protocol is collusion resistant.*

Proof: First we prove part 1) of collusion resistance. We add a new query to the query part of the original attack game:

Collusion Query: The adversary chooses a set of compromised users U and for all $n_i \in U$, the challenger returns the corresponding k_i and d_i' to the adversary.

Server obliviousness holds for the uncompromised nodes if no probabilistic polynomial-time adversary has more than negligible advantage in winning this revised game. The proof that this game is hard to win is similar to the proof of Theorem 1.

Next we prove part 2) of collusion resistance. We revise the original pollution attack game like this:

- The challenger generates valid data values $\{d_1, ..., d_w\}$, a set of different random numbers $\{x_1, ..., x_m\}$, RSA key pairs, a random value γ and sends $\{d_1, ..., d_w\}$, x_i and (e, N) to the adversary.
- The adversary chooses a data value $\alpha \notin \{d_1, ..., d_w\}$ and makes the following queries to the challenger:
  Signing Query: The challenger returns $(\alpha x_i)^d \bmod N$.
  Validation Query: The challenger returns $\{(\frac{\gamma}{d_1})^d \bmod N, ..., (\frac{\gamma}{d_w})^d \bmod N\}$
  Collusion Query: The adversary chooses a set of colluding users U ($n_i \notin U$) and the challenger generates a data

TABLE II: Cost Analysis per Task

	Server	MNs
Storage (bits)	$1024(3m+w+1)+$ $w\log_2 P$	$1024(w+4)+$ $w\log_2 P$
Computation (modular exp.)	$m+2w+\sqrt{mP}$	3
Communication (bits)	$1024m(w+6)$	$1024(w+6)$

value β_j for each $n_j \in U$. The challenger then returns x_j, β_j, $(\beta_j x_j)^d \bmod N$ and $(\gamma x_j)^d \bmod N$ of all colluders to the adversary.

- The adversary outputs δ

Let S denote the event that $\delta = (\gamma x_i)^d \bmod N$. We define the advantage of the adversary in winning the above game to be $Pr[S]$. It is not hard to see that S corresponds to the event that a colluding MN passes data validation after submitting invalid data. Pollution resistance still holds if no probabilistic polynomial-time adversary has more than negligible advantage in winning this revised pollution attack game. The proof that this game is hard to win is similar to the proof of Theorem 2. In fact, we can see from the above game that collusion is prevented because the server ensures x_i for different MNs are different during Data Preprocessing (See Section IV-B). □

VI. Evaluations

A. Cost Analysis

Suppose each secret is 256 bits and each node is assigned with four secrets (As shown in [22], with four secrets per node, 80-bit security can be achieved even for a system with 10^4 MNs). MNs' encryption keys, the server's decryption key and RSA keys are 1024 bits long. The plaintext space of possible data values is P and the data value vector is of size w. The range of sum is R ($R = mP$). The storage cost, computation cost and communication cost of the server and MNs are shown in Table II.

1) Storage Cost: Each MN needs to store its secrets, the signed report, the server's public key, the data value vector, and the validation vector. The server needs to store its RSA key pairs, its secrets, the data value vector, the validation vector and all MNs' encrypted reports. When $m = 1,000$, $P = 10^6$, and $w = 100$, the storage cost is about 200K bits on MNs and 3M bits on the server. For modern computer and smartphones, the storage cost is not an issue.

2) Computation Cost: According to existing benchmark data [23], modular exponentiation dominates the computation cost. Thus we mainly consider modular exponentiations here.

On the MN side, three modular exponentiations are needed per task. This cost is very low and is not an issue.

On the server side, in Setup, Data Preprocessing and Data Validation phases $m+2w$ modular exponentiations are needed. Such computation overhead is not an issue for modern server machines. However, in the Data Aggregation phase, the server needs to compute discrete log which is generally difficult. To speed up the decryption, we use a parallel version of Pollard's kangaroo method, and the decryption cost can be reduced to about $1.714\sqrt{R}$ modular exponentiations.

TABLE III: Running Time per Task on the Server

(a) Effect of m

m	Setup	Preprocess	Validation
10	169 ms	2 ms	14 ms
100		8 ms	29 ms
1000		15 ms	51 ms

(b) Effect of R

R	Aggregation
10^2	2.7 ms
10^4	15 ms
10^6	85 ms

TABLE IV: Running Time per Task on Mobile Nodes

Setup	Preprocessing	Aggregation	Validation
0.16 ms	48.5 ms	1.4 ms	0.6 ms

3) Communication Cost: For the server and n_i, the messages exchanged during one task are as follows: random number a and b, blinded report o_i' and signed blinded report s_i' during Data Preprocessing; encrypted report c_i during Data Aggregation; the validation vector and Δ during Data Validation. Even with large m and w, the communication cost is still small and not an issue at all since data aggregation applications are not time-sensitive.

B. Implementation and Cost Evaluations

To measure the computation cost and power consumption, we build a prototype and use Java to implement our protocol. The prototype includes several MNs and a server. The smartphones we use to implement MNs are Android Nexus S Phones with 1GHz CPU, 512MB RAM and Android OS 4.4.2. The laptop we use to implement the server is a MacBook Air Laptop with 1.3GHz CPU, 8GB RAM and MAC OS 10.9.2. MNs communicate with the server through WiFi. We use RSA blind signature as the blind signature scheme and SHA-256 as the hash function.

1) Computation Cost: Based on the analysis in Section VI-A, on the server side, we measure the running time of Setup, Data Preprocessing, and Data Validation phases for different m and measure the running time of the Data Aggregation phase for different R. The results are shown in Table III. For most applications, a plaintext space of 10^6 is more than enough. In this case, with $1,000$ mobile nodes, the running time per task on the server is less than 400 ms. On the MN side, none of the four phases' are affected by m or R. The running time of each phase is shown in Table IV. We can see that, on the MN side, the running time is about 50 ms.

2) Power Consumption: We use Monsoon Power Monitor to measure power consumption as in [24]. Since each MN's computation complexity is not affected by other MNs, we evaluate the power consumption with different R when $m = 100$. The results are shown in Table V. We can see the power consumption of our scheme is low, even when the sum range is vary large. To further understand the energy cost of our protocol, we also use an Android application PowerTutor to estimate the total amount of energy consumed per task as well as the amount of energy consumed by LCD display, WiFi, and CPU, respectively. The results are shown in Table VI. We can see that most of the energy (more than 90 percent) is consumed by LCD and the amount of energy consumed by WiFi and CPU is small. Even with $R = 10^6$, an MN can complete more than $200,000$ tasks before running out of battery.

TABLE V: Power Consumption on Smartphones with m=100

R	10^2	10^4	10^6
Power Consumtion (mW)	182.67	204.82	224.65

TABLE VI: Energy Cost on Smartphones with m=100

R		10^2	10^4	10^6
Energy Cost per Task (mJ)	LCD	55.9	65.1	85.9
	WiFi	2.7	3.0	3.8
	CPU	2.1	2.5	3.5
	Total	60.7	70.6	93.2
# Tasks per Battery (3.7 V,1500 mAh)		329160	283002	214378

C. Degree of Privacy

The goal of our PATA scheme is to ensure that the server does not know the value of the data that each MN submits, and we want to measure how well this privacy goal is achieved. We define a metric to measure the data privacy provided by PATA: $DP(n_i) = \sum_{l=1}^{w} \hat{p}(d_l)||d_l - d_i||$. $DP(n_i)$ is the data privacy of n_i. $\hat{p}(d_l)$ is the probability for the server to guess if n_i submits d_l (the l^{th} value in the data value vector). If $d_l = d_i$, then $||d_l - d_i|| = 1$; otherwise, $||d_l - d_i|| = 0$.

If the server has no knowledge of MNs' data at all, $\hat{p}(d_l)$ follows uniform distribution. In our data aggregation, the server knows the average of all collected reports, and thus we use a normal distribution with μ to be the average, and $\sigma = (d_w - d_1)/4$. Note that using range/4 as an estimate of the standard deviation is widely used in statistics, and using the range of our data value vector as the range of collected data is also reasonable in our model. Since the server guesses values based on the data value vector, the probability calculated from the normal distribution should be unified so that $\sum_{l=1}^{w} \hat{p}(d_l) = 1$: $\hat{p}(d_l) = \frac{f(d_l,\mu,\sigma)}{\sum_{l=1}^{w} f(d_l,\mu,\sigma)}$. Here $f(d_l, \mu, \sigma)$ is the probability density function of $\mathcal{N}(\mu, \sigma)$. In our experiments, each mobile node submits a random value in the range of the data value vector and the server guesses based on the average of the collected reports as shown above. The results are shown in Fig. 6. We can see that PATA can provide a high level of privacy.

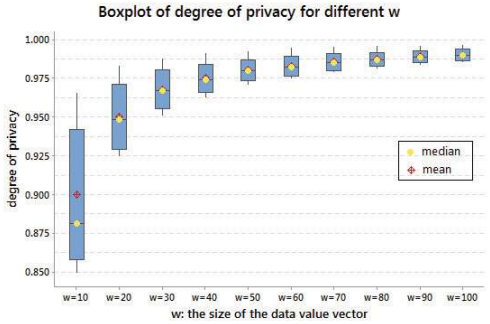

Fig. 6: Degree of Privacy

VII. Discussions

A. Data Value Vector

The data value vector may be linear or non-linear. For example, to monitor the flu trend by collecting how many users are infected, the server can set the data value vector as $[0, 1]$, where 0 means that a user is fine and 1 means that a user has caught this flu. Even when the data range is large, we can still select a small number of representative values to meet the application requirement. For example, to collect how many hours people spend on social media (Facebook, Twitter, etc.) every week, the server can pick a data value vector like $[0, 2, 5, 10, 20, 30, 50, 100]$. This short vector covers the range of possible hours people spend on social media and includes values at different levels. Certainly, based on the application requirement, more fine grained data value vector can be used, at the cost of more communication bandwidth.

B. Trustworthiness

Our PATA scheme considers data inside the data value vector to be valid. Further validation within the data value vector is not the focus of this paper. Our solution works especially well for those applications that aggregate binary sensing results (0 or 1, "yes" or "no", etc.), and there are many such applications: survey applications that use participants as sensors to learn about people's habits and life patterns [4] (e.g., whether a user smokes or not, whether a user watches TV every night, etc.) and user context detection applications that use sensors on mobile phones to learn about people's surroundings and current activities [25], [26] (e.g., using light sensors to detect whether a user is at a club or not, using microphones to detect whether a user is talking or not, using accelerometers to detect whether a user is driving on the highway or not, etc.). For these applications, PATA can prevent malicious users from submitting invalid data to trick the server and at the same time protects participants' privacy well. For other applications that aggregate data with a larger range (e.g., the blood pressure of users, the time users spend on Facebook, etc.), PATA can still provide certain degree of trustworthiness while protecting the privacy. Also, by properly setting the data range in the data value vector, the negative effects of the malicious users can be minimized.

C. Trusted Authority

In Section IV, we assume that a trusted authority is responsible for generating secrets for the server and MNs. The assumption can be relaxed to an honest-but-curious key dealer which does not collude with the aggregation server. The key dealer follows our protocol to generate secrets for all parties but it may try to infer MNs' data value by eavesdropping communications between the server and MNs. Under this relaxed assumption, we only need to add an encryption and decryption step to the Data Aggregation phase: each MN encrypts the encrypted report c_i with a pre-shared key with the server and sends the final result to the server. To get the aggregate statistics, the server first decrypts each MN's ciphertext using the pre-shared key and then does the Aggregation and Decryption described in Section IV.

D. Fault Tolerance

Some nodes may fail to submit reports for a task. In order to get the aggregate statistics for the remaining MNs, the server can ask the key dealer to submit encrypted reports for the failed nodes. Suppose n_j is a failed node. First the key dealer generates an original report $o_i = g^{0+\delta}$ (δ is a noise which

can be chosen as small values from the data value vector) and has the server sign it blindly. Since the key dealer knows n_j's encryption key k_j, it can submit an encrypted report $c_j = (g^\delta)^d \cdot h^{k_j} \ mod\ N$ to the server. On receiving all reports, the server can decrypt as follows:

$$S = h^{k_0}(g^\delta)^d h^{k_j} \prod_{i\neq j}^{m}[(g^{d_{l_i}} x_i)^d \cdot h^{k_i}] \prod_{i\neq j}^{m} x_i^{-d} \bmod N$$

$$= (g^{\delta + \sum_{0<i\neq j\leq m} d_{l_i}})^d \bmod N$$

The server can then get a noisy sum for the remaining nodes following the PATA protocol. With some appropriate noise, differential privacy can even be achieved [22].

VIII. Conclusions

We proposed a novel privacy-aware and trustworthy data aggregation protocol (PATA). Our protocol utilizes blind signature, homomorphic encryption, as well as a novel encrypted vector-based data validation technique. The server can aggregate users' data without actually knowing any individual's data. Our protocol also enables the server to validate users' trustworthiness in a privacy-preserving manner based on whether they submit valid data report following our protocol.

Security analysis shows that our scheme can protect the privacy of mobile users from a curious server and at the same time can protect the system from malicious users. Experimental results based on a prototype show that our protocol runs very fast even when the plaintext space is large and many mobile users exist in the system. The power consumption of the smartphone is also very low. These results show that our protocol is feasible for a wide range of mobile sensing applications with various plaintext spaces and resource constraints.

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. of ACM SenSys*, 2009.

[2] E. Kanjo, "NoiseSPY: A Real-Time Mobile Phone Platform for Urban Noise Monitoring and Mapping," *MONET*, vol. 15, no. 4, 2010.

[3] M. Rostami, A. Juels, and F. Koushanfar, "Heart-to-Heart (H2H): Authentication for Implanted Medical Devices," in *ACM SIGSAC*, 2013.

[4] P. Vicente, E. Reis, and M. Santos, "Using Mobile Phones for Survey Research," *IJMN*, vol. 51, no. 5, 2009.

[5] J. Shi, Y. Zhang, and Y. Liu, "Prisense: Privacy-Preserving Data Aggregation in People-Centric Urban Sensing Systems," in *INFOCOM*, 2010.

[6] V. Rastogi and S. Nath, "Differentially Private Aggregation of Distributed Time-Series with Transformation and Encryption," in *Proc. of ACM SIGMOD*, 2010.

[7] M. M. Groat, W. He, and S. Forrest, "KIPDA: k-Indistinguishable Privacy-preserving Data Aggregation in Wireless Sensor Networks," in *Proc. of IEEE INFOCOM*, 2011.

[8] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-Preserving Data Aggregation in Wireless Sensor Networks," in *Proc. of IEEE INFOCOM*, 2007.

[9] Q. Li and G. Cao, "Efficient and Privacy-Preserving Data Aggregation in Mobile Sensing," in *Proc. of IEEE ICNP*, 2012.

[10] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," in *Proc. of ACM SenSys*, 2003.

[11] W. Zhang, S. K. Das, and Y. Liu, "A Trust Based Framework for Secure Data Aggregation in Wireless Sensor Networks," in *Proc. of IEEE SECON*, 2006.

[12] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," in *Proc. of ACM MobiHoc*, 2008.

[13] X. Xu, Q. Wang, J. Cao, P.-J. Wan, K. Ren, and Y. Chen, "Locating Malicious Nodes for Data Aggregation in Wireless Networks," in *Proc. of IEEE INFOCOM*, 2012.

[14] X. Oscar Wang, W. Cheng, P. Mohapatra, and T. Abdelzaher, "ARTSense : Anonymous Reputation and Trust in Participatory Sensing," in *Proc. of IEEE INFOCOM*, 2013.

[15] L. Kazemi and C. Shahabi, "TAPAS: Trustworthy Privacy-Aware Participatory Sensing," *KAIS*, vol. 37, no. 1, 2013.

[16] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-Based Identity Inference in Anonymous Spatial Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 12, 2007.

[17] H. Zang and J. Bolot, "Anonymization of Location Data Does Not Work: A Large-Scale Measurement Study," in *Proc. of ACM MobiCom*, 2011.

[18] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao, "Privacy Vulnerability of Published Anonymous Mobility Traces," in *ACM MobiCom*, 2010.

[19] R. Zhang, J. Shi, Y. Zhang, and C. Zhang, "Verifiable Privacy-Preserving Aggregation in People-Centric Urban Sensing Systems," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 9, 2013.

[20] D. Chaum, "Blind Signatures for Untraceable Payments," in *Proc. of CRYPTO*, 1983.

[21] S. Galbraith, J. Pollard, and R. Ruprai, "Computing Discrete Logarithms in an Interval," *Mathematics of Computation*, vol. 82, no. 282, 2013.

[22] Q. Li and G. Cao, "Efficient Privacy-Preserving Stream Aggregation in Mobile Sensing with Low Aggregation Error," in *PETS*, 2013.

[23] D. J. Bernstein and T. L. (editors), "eBACS: ECRYPT Benchmarking of Cryptographic Systems," http://bench.cr.yp.to, accessed 10 May 2014.

[24] W. Hu and G. Cao, "Energy-Aware Video Streaming on Smartphones," in *Proc. of INFOCOM*, 2015.

[25] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz, "Sensing Techniques for Mobile Interaction," in *Proc. of ACM UIST*, 2000.

[26] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones," in *Proc. of ACM MobiSys*, 2009.

Appendix
Lemmas and Proofs

Lemma 1. $|Pr[S_0]| = 1/2$

Proof: In Game 0, Q_s and Q_e are the same for $y = 0$ and $y = 1$. Thus the probability that the adversary correctly guesses y is the same as a random guess. □

Lemma 2. $|Pr[S_{j+1}] - Pr[S_j]| = \epsilon_{ddh}$, *where* ϵ_{ddh} *is the DDH-advantage of a polynomial-time adversary.*

Proof: We prove that if the adversary can distinguish between Game j and Game $j-1$. We construct an algorithm $D(\alpha, \beta, \delta)$ as follows:

$y \xleftarrow{R} \{0,1\}, (g, N) \leftarrow \mathcal{P}(), (k_0, ...k_{j-1}, k_{j+2}, ..., k_m) \leftarrow \mathcal{K}()$

$(d_1, ..., d_m) \leftarrow A(), r \xleftarrow{R} \{0,1\}^N, (x_1, ..., x_j) \leftarrow \mathcal{X}(g, r, d_1, ..., d_j)$

$(z_1, ..., z_j) \leftarrow \mathcal{Z}(g, h, k_1, ..., k_j, d_1, ..., d_j)$

$if\ y = 0 : Q_s \leftarrow (g^{d_1} r^e, ..., g^{d_m} r^e), Q_e \leftarrow (g^{d_1} h^{k_1}, ..., g^{d_m} h^{k_m})$

$else : Q_s \leftarrow (x_1, ..., x_j, g^{d_{j+1}} r^e, ..., g^{d_m} r^e), Q_e \leftarrow (z_1, ..., z_{j-1},$

$$\frac{\prod_{i=1}^{j} g^{d_j}}{\prod_{i=1}^{j-1} z_i \cdot \delta \cdot \prod_{i=j+2}^{m} \beta^{k_i}}, g^{d_{j+1}}\delta, g^{d_{j+2}}\beta^{k_{j+2}}, ..., g^{d_m}\beta^{k_m})$$

$\hat{y} \leftarrow A(g, \beta, N, k_0, (d_1, ..., d_m), Q_s, Q_e)$

$if\ y = \hat{y},\ output\ 1;\ else\ output\ 0$

Suppose $a, b, c \in \mathbb{Z}_N$ are three different random numbers. The DDH problem is to distinguish between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c). If the input to D is (g^a, g^b, g^{ab}), the computation proceeds as in Game j, and $Pr[D(g^a, g^b, g^{ab}) = 1] = Pr[S_j]$. If the input to D is (g^a, g^b, g^c), the computation proceeds as in Game $j+1$, and $Pr[D(g^a, g^b, g^c) = 1] = Pr[S_{j+1}]$. Therefore, distinguishing between Game j and Game $j+1$ means solving DDH problem. □