Hindawi
Mobile Information Systems
Volume 2017, Article ID 3715253, 11 pages
https://doi.org/10.1155/2017/3715253

Research Article

A Collusion-Resistant and Privacy-Preserving Data Aggregation Protocol in Crowdsensing System

Chang Xu, Xiaodong Shen, Liehuang Zhu, and Yan Zhang

Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Correspondence should be addressed to Liehuang Zhu; liehuangz@bit.edu.cn

Received 8 December 2016; Accepted 15 March 2017; Published 30 March 2017

Academic Editor: Rossana M. C. Andrade

With the pervasiveness and increasing capability of smart devices, mobile crowdsensing has been applied in more and more practical scenarios and provides a more convenient solution with low costs for existing problems. In this paper, we consider an untrusted aggregator collecting a group of users' data, in which personal private information may be contained. Most previous work either focuses on computing particular functions based on the sensing data or ignores the collusion attack between users and the aggregator. We design a new protocol to help the aggregator collect all the users' raw data while resisting collusion attacks. Specifically, the bitwise XOR homomorphic functions and aggregate signature are explored, and a novel key system is designed to achieve collusion resistance. In our system, only the aggregator can decrypt the ciphertext. Theoretical analysis shows that our protocol can capture k-source anonymity. In addition, extensive experiments are conducted to demonstrate the feasibility and efficiency of our algorithms.

1. Introduction

Recently, smart devices and wireless network have a rapid development. Smart devices, such as smart phone, pad, and smart watch, have become ubiquitous all over the world. They have not only strong and independent computational capability but also rich embedded sensors. The advance of wireless communication technology further makes them connect more tightly, which can be leveraged to develop more applications. People can use the rich embedded sensors to collect different kinds of data, including pictures, sounds, and videos. The strong ability and lower cost of this system derive a new popular paradigm named crowdsensing.

In a typical crowdsensing application, the server, or the aggregator, recruits a group of users to work for him. Having been informed about their sensing work by the aggregator, all the users use their devices to collect data with relevant sensors and upload them to the server through Wi-Fi or 3G/4G network. Recently, a myriad of crowdsensing applications have been developed in different areas such as transportation [1], environment monitoring [2], healthcare [3], and social network [4]. In this paper, we consider that an aggregator wants to periodically collect data and computes some functions based on them to obtain the desired information. For example, to monitor the health situation of a particular district, the aggregator recruits some users in this district to collect their body temperature or blood oxygen. The users need to contribute their data each hour by sensing relative data and uploading them.

However, most similar applications require users to upload their private data, which may breach individual's security and privacy. Concerned about these threats, users tend to refuse to participate in crowdsensing. Therefore, the user's security and privacy should be protected in a crowdsensing system. Lots of previous works [5–11] have focused on the challenge. Specifically, [5] allows the server to evaluate any multivariate polynomials. However, users need to communicate to generate their encryption key. In [7], the aggregator can only acquire the summation of all the raw data. Both of them did not consider the collusion between users and the aggregator. And [6] gives a solution by using more complex encryption keys. Unfortunately, the

protocol requires $k + 1$ rounds of key exchange when k colluding adversaries exist, and [8, 9] focus on multimedia data collection and require data interchange. Li et al. [10] proposed a novel key system to resist collusion attack, but it only supports sum and min aggregation. Zhang et al. [11] first proposed a scheme where the aggregator can acquire all the raw data; thus different functions can be computed in one round. Each user's privacy is protected by delinking data from its source. Thus, the only information the aggregator knows is that a particular data belongs to one of k users, which is called k-source anonymity. Nevertheless, there are still some problems in [11]. Specifically, each user owns half part of another user's secret keys and cannot resist collusion attack. The aggregator does not have decryption keys. Therefore, the outside adversary can decrypt the ciphertexts to get all the data if it can eavesdrop all the ciphertexts, which violates the aggregator's benefits.

In this paper, we propose a novel protocol to not only support different aggregation functions but also achieve collusion resistance. In each time period, we use the timestamp as a public parameter. The bitwise XOR homomorphic function is executed in the encryption phase. All the users take their encryption keys and the timestamp as the parameters of encryption functions to generate pseudorandom bit strings as a one-time pad to encrypt the raw data. To prevent the collusion attack, a novel encryption algorithm is designed. The aggregate signature is also taken to protect data integrity and achieve identity authentication.

Our contributions contain three parts:

(i) We propose a novel protocol to protect users' privacy and resist collusion attack when the aggregator can obtain all the users' raw data and compute any functions based on them. We assume that the aggregator and a fraction of users are not reliable and may collude with each other, and they still cannot obtain any valuable information.

(ii) We protect the aggregator's benefits by preventing outside adversary from decrypting the ciphertexts. All the ciphertexts are also guarded against being tampered by applying aggregate signature. If any abnormal data is found, the aggregator can require the trust party to get involved according to the signature.

(iii) We prove that our protocol can achieve k-source anonymity. Theoretical analysis shows the computational cost of our protocol. In addition, extensive experiments are also conducted to demonstrate that the protocol can be executed efficiently.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we present our system model, security model, and design goals. After the introduction of preliminary knowledge in Section 4, we elaborate our aggregation scheme in Section 5 and prove the security of the protocol in Section 6. Section 7 shows our experiment result. Finally, we conclude our paper in Section 8.

2. Related Work

The data aggregation issues are first discussed in wireless sensor networks (WSN) [12–16]. Although there are many differences between WSN and crowdsensing, the work about WSN still gives inspirations to solve problems in crowdsensing. Works in [17, 18] consider the user recruitment and incentive in crowdsensing. The papers [19–21] assume a trust server although they are devoted to protect users' privacy. References [5–9, 22] contribute to overcoming the challenge when the aggregator is untrusted. However, bidirectional communication between users is required in these schemes, which is a strong assumption in crowdsensing system. Jung et al. [5] first proposed their product protocol and sum protocol and combined them to evaluate any multivariate polynomials. In the product protocol, each user interchanges his/her public parameter with his/her left and right user while all the users are arranged in a circle. With two public parameters and the secret key, the user can derive a pseudorandom number to execute encryption operations. In the sum protocol, they use modular property to compute summation efficiently. However, the pseudorandom number can only be used once and may breach the privacy if being used in several rounds. Therefore, in each round the setup phase should be executed where users have to communicate with the other two. The collusion between users is also a security issue. Jung et al. [6] tried to solve the problem by using more complex ways to generate pseudorandom number, correspondingly more rounds of key interchange are needed, and Jung et al.'s scheme only supports particular aggregation functions.

In [10, 23], the privacy-preserving aggregation protocol is proposed while communication within users is not required. Zhang et al. [23] allowed the aggregator to obtain the minimum or the kth minimum value of all the data without knowing them, and they assumed a semihonest aggregator and only supported the single aggregation function. Li et al. [10] proposed a scheme to resist collusion by using a novel key management system. The key dealer generates a key set which contains hundreds or thousands of elements. Then, the set is divided and distributed to the users and aggregator, and each of them owns multiple secret keys. The ability to resist collusion attack relies on the adversary to guess an honest user's keys correctly from a big set. However, this scheme can only support sum and min aggregation.

Different from the schemes which protect privacy by hiding content, Zhang et al. [11] delink all the users' data with source, through which the aggregator can collect all the raw data while it remains unaware of their corresponding owners. Thus, the aggregator can compute any complex aggregation functions on them. Each user has two keys to encrypt data, and he/she shares each of them with the other two users. Therefore, the aggregator can decrypt all the ciphertexts without decryption keys when the bitwise XOR homomorphic functions are employed. However, this paper ignores the collusion attack and cannot protect the aggregator's benefits, because an honest user's secret key can easily be recovered and the aggregator has the same ability with outside adversary without decryption keys.

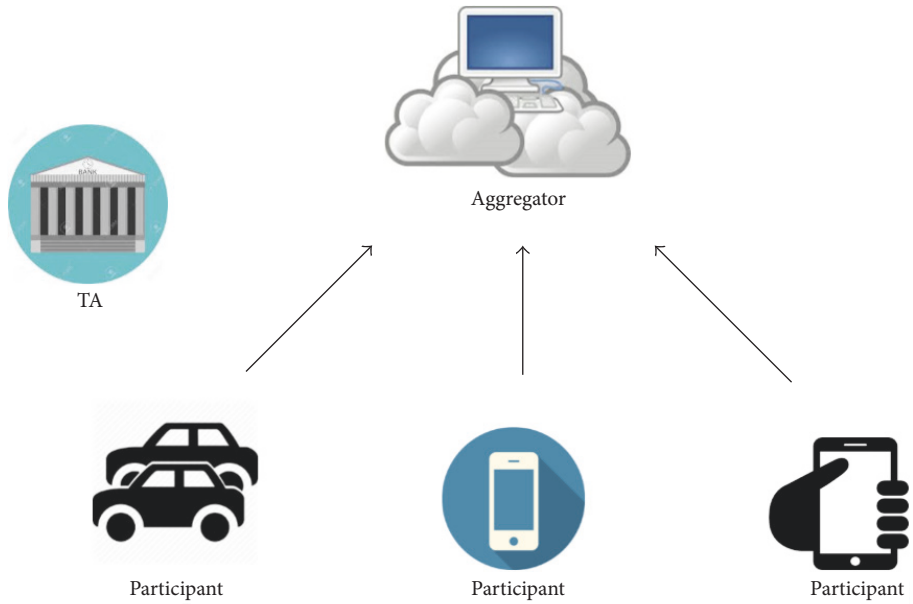

Figure 1: System model of data aggregation.

In this paper, we propose a novel protocol to solve the issues which are not dealt with in [11], while protecting the data integrity and achieving authentication and traceability.

3. Models and Design Goal

3.1. System Model. Our system is comprised of three parties: N participants, an aggregator, and a trust authority (TA). Assuming that there are N participants in this system who want to contribute data to the aggregator and get corresponding reward from the aggregator, the aggregator is willing to collect data and compute some functions on these data including addition and production aggregation. Only one-way communication channel is needed from participants to the aggregator, which could be 3G/4G, Wi-Fi, or other kind of channels supported by our system parties. We show our system model in Figure 1 and describe the details as follows:

(i) *TA*. The TA is responsible for initializing the whole system, which includes registering the aggregator and participants, generating and distributing keys, and revealing and revoking the malicious participants. Once the system initialization phase is finished, the TA is off-line in all the phases except for the occurrence of abnormal behavior.

(ii) *Participant*. The participants may be mobile users who hold smartphones with various sensors or vehicles with built-in sensors. They wish to sense data and upload them to the aggregator periodically to get reward. Assuming that there are n participants in our system, which can be numbered as $u_1, u_2, \ldots, u_n$, they collaborate to push data to the aggregator in each time period, for instance, fifteen minutes per time, which can be listed as $t_1, t_2, \ldots, t_n$. Peer-to-peer communication is not required among participants. In the remainder of this paper we will interchangeably use the same meaning for the user and the participant.

(iii) *Aggregator*. The aggregator periodically collects the participants' data and uses them to compute arbitrary aggregation functions. The aggregation result can be leveraged to get commercial benefits. The data can be time series data, location based data, or any other kind of predefined numerical data.

3.2. Security Model. Because the collected data may include users' sensitive information, we mainly focus on the participants' privacy in our security model. Any adversary should not link data with the real data owner, such that the users' privacy will not be compromised, even if any internal adversaries, namely, the malicious users and the aggregator, collude to snoop into the privacy. Meanwhile, if the abnormal data is detected, the TA is involved to reveal the malicious users' real identities and revoke them from the system.

(i) *TA*. In our system, we assume the TA is fully trusted and cannot be compromised. The communication channel between the TA and participants is secure or can be protected by cryptographic tools.

(ii) *Participant*. The participants honestly execute the protocol, but they are curious about other participants' data. This assumption is based on the fact that some users can leverage others' valid data to receive reward instead of collecting data by their own, which may consume computation resource, battery, and other resources. A fraction of malicious users may also collude to recover valid users' secret keys,

which compromises users' privacy. Another abnormal behavior is data pollution; that is, some users deliberately upload incorrect data to the aggregator, leading to wrong aggregation result. Many previous studies have focused on this issue but cannot solve the problem perfectly. Thus, in this paper, we provide traceability to identify the malicious users when the abnormal data is found in each round.

(iii) *Aggregator*. The aggregator is curious but honest. The aggregator can collect all the users' data and has more abilities to breach users' privacy. The untrusted aggregator can also collude with some malicious users to recover users' secret keys and thus link the data with its owner. Other parties may eavesdrop all the users' data to compute the aggregation, thus causing the aggregator's monetary loss.

3.3. Design Goals. Under our system model, our design goal is to develop a framework to hold the security properties. We not only protect the privacy of users but also resist against collusion attack launched by internal parties and other attacks such as message tampering launched by outside adversaries. Specifically, the following desirable goals should be achieved.

(i) *Protecting Participants' Privacy*. When the participants upload their sensing data to the aggregator, we should guarantee not only that outside adversary cannot eavesdrop and tamper the original data but also that other users cannot decrypt the ciphertext. Any tampered data can be recognized by the aggregator and a retransmission request is sent to the user. Any illegal party cannot forge a legal user's signature. The user's data should not be linked with its source by any party including the aggregator.

(ii) *Safeguarding the Aggregator's Benefits*. We assume any party including outside adversary has the ability to eavesdrop all the uploaded data, and if other people can recover the original data, they can compute any aggregation result and thus seriously damage the aggregator's benefits. Therefore, we design the protocol to prevent illegal parties from getting valuable data.

(iii) *Computation Efficiency and Accuracy*. The proposed framework should achieve computation efficiency and accuracy; in particular, (i) the users can efficiently encrypt the data and compute the corresponding signature on it; (ii) the aggregator can efficiently verify all the users' signatures and recover the original data accurately; (iii) with all the original data the aggregator can accurately compute any function on them with high efficiency.

4. Preliminary

4.1. k-Source Anonymity. Assuming that there is a group of k users, each user uploads his/her data to the aggregator. Although the aggregator can obtain the exact values of all data, it still cannot link each data with its owner, because every user's data is hidden in the dataset of k elements. The more users the group contains, the higher security level the system achieves. For a specific user, if the adversary can only know that the user's data belongs to one of the k users, we say this user's data holds k-source anonymity. If all data captures k-source anonymity in a data aggregation protocol, we say this protocol achieves k-source anonymity. Intuitively, the definition states if the aggregator cannot efficiently notice whether we switch two data items, the aggregation protocol with k users is k-source anonymous.

k-Source Anonymous. A data aggregation process or a protocol is *k-source anonymous* if $P(\ldots, u_i(d^i), \ldots, u_j(d^j), \ldots) \stackrel{c}{\equiv} P(\ldots, u_i(d^j), \ldots, u_j(d^i), \ldots)$ is satisfied, where k denotes the number of users for any group U, u_i and u_j are two users in U, $\{d^1, \ldots, d^k\} \in \{M\}^k$ is data aggregation sample, M denotes the message space, $P(\ldots, u_i(x_i), \ldots)$ denotes the data aggregator's view when running protocol with x_i ($x_i \in \{d^1, \ldots, d^k\}$) as u_i's input ($i = 1, 2, \ldots, k$), and $\stackrel{c}{\equiv}$ denotes computational indistinguishability of two random variable ensembles.

Our goal is to design efficient data aggregation protocols that can be used by an untrusted aggregator to collect all users data in a source-anonymous manner.

4.2. Bilinear Map and Aggregate Signature. Let G_1 and G_2 be two cyclic groups of prime order q, and their generators are g_1 and g_2, respectively. There exists an additional group G_T such that $|G_1| = |G_2| = |G_T|$. A bilinear map is a map $\hat{e} : G_1 \times G_2 \to G_T$ with the following properties.

(1) Bilinear. For all $u \in G_1$, $v \in G_2$, and $a, b \in \mathbb{Z}$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.

(2) Nondegenerate. $\hat{e}(g_1, g_2) \neq 1$. The aggregate signature scheme [24] employs a full-domain hash function $H : \{0,1\}^* \to G_1$ and comprises the following five phases.

Key Generation. For a particular user, pick random $x \stackrel{R}{\leftarrow} Z_p$, and compute $v \leftarrow g_2^x$. The user's public key is $v \in G_2$. The user's secret key is $x \in Z_p$.

Signing. For a particular user, given the secret key x and a message $M \in \{0,1\}^*$, compute $h \leftarrow H(M)$, where $h \in G_1$, and $\sigma \leftarrow h^x$. The signature is $\sigma \in G_1$.

Verification. Given a user's public key v, a message M, and a signature σ, compute $h \leftarrow H(M)$; accept the signature if $\hat{e}(\sigma, g_2) = \hat{e}(h, v)$ holds.

Aggregation. For the aggregating subset of users $U \subseteq \mathbb{U}$, an index i is assigned to each user, where $i \in [1, k]$ and $k = |U|$. Each user $u_i \in U$ provides a signature $\sigma_i \in G_1$ on a message $M_i \in \{0,1\}^*$ of his/her choice. The messages M_i must all be distinct. Compute $\sigma \leftarrow \prod_{i=1}^{k} \sigma_i$. The aggregate signature is $\sigma \in G_1$.

Aggregate Verification. Given an aggregate signature $\sigma \in G_1$ for an aggregating subset of users U, indexed as before, and the original messages $M_i \in \{0,1\}^*$ and public keys $v_i \in G_2$ for all users $u_i \in U$, to verify the aggregate signature σ,

(1) ensure that the messages M_i are all distinct, and reject them otherwise;

(2) compute $h_i \leftarrow H(M_i)$ for $1 \le i \le k = |U|$, and the aggregate signature is valid if $\widehat{e}(\sigma, g_2) = \sum_{i=1}^{k} \widehat{e}(h_i, v_i)$ is satisfied.

5. The Collusion-Resistant and Privacy-Preserving Data Aggregation Protocol

In this section, we present our privacy-preserving aggregation scheme to achieve the aforementioned design goals.

5.1. Overview. Our proposed scheme can achieve k-source anonymity while it prevents adversary from tampering users' uploaded messages and generates invalid signatures. During the system initialization phase, the TA generates users' public/secret key for message encryption and authentication and the aggregator's secret key for decryption. When the aggregator wants to collect some data from n users, it first confirms the time period t with other users. All the users sense the data and encrypt it with their own secret keys and sign the encrypted data and then collectively upload all the data to the aggregator. If any user does not have data to send, he/she can upload a predefined value, which helps others to decrypt data. After all the data are collected, the aggregator first aggregates all users' signatures and verifies them. If signatures are valid, all the users' original data can be recovered with the aggregator's secret key but cannot be linked with their owners' real identities.

Our scheme consists of three algorithms: *System Initialization* algorithm assigns keys to the users and the aggregator. *Enc&Sign* algorithm encrypts users' data and signs the ciphertext. *Verify&Dec* algorithm verifies and decrypts users' data.

We state the basic idea of the encryption and decryption here. Consider the following equation:

$$s_1 \oplus s_2 \oplus \cdots \oplus s_n = s_1 \oplus s_2 \oplus \cdots \oplus s_n. \tag{1}$$

Then we use s_k $(k \in [1,n])$ as the key of the pseudorandom hash function h; thus

$$\begin{aligned} &h_{s_1}(t) \oplus h_{s_2}(t) \oplus \cdots \oplus h_{s_n}(t) \\ &\quad = h_{s_1}(t) \oplus h_{s_2}(t) \oplus \cdots \oplus h_{s_n}(t). \end{aligned} \tag{2}$$

We assign the left part of (1) to all users and the right part to the aggregator. They use the same t as the parameter of h and $h_{s_k}(t)$ as the pad to encrypt or decrypt the data, and thus the aggregator can eliminate all the pads with its keys.

However, although all users can collectively compute n hash functions, the aggregator has to compute n hash functions every time by himself. Therefore, we move some elements in the right part of (1) to the left:

$$\begin{aligned} &s_1 \oplus s_2 \oplus \cdots \oplus s_n \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{n-q} \\ &\quad = s_{n-q+1} \oplus \cdots \oplus s_n. \end{aligned} \tag{3}$$

Thus the aggregator knows fewer secret keys and computes much fewer hash functions, and each user only needs to compute a few more functions. The notations in our scheme are listed in the Notations.

5.2. System Initialization. Given the security parameter λ, the TA generates the bilinear parameters $(q, G_1, G_2, G_T, \widehat{e}, g_1, g_2)$ and chooses two hash functions: $H : \{0,1\}^* \to G_1$ and $h_s(x)$ which is a function indexed by s in a pseudorandom function family $\mathscr{H}_{l,m+\lceil \log_2 n \rceil, l} = \{h_s : \{0,1\}^{m+\lceil \log_2 n \rceil} \to \{0,1\}^l\}_{s \in \{0,1\}^l}$.

Then the TA randomly generates $s_1, s_2, \ldots, s_{nc} \in \{0,1\}^l$ as secret keys, where n is the number of the users. As the idea described in previous subsection, we distribute these secret keys as follows:

(i) The keys are randomly divided into n disjoint subset, $\dot{S}_1, \dot{S}_2, \ldots, \dot{S}_n$. We use $\dot{S}_i$ to denote the user i's additive secret key set, where $|\dot{S}_i| = c$, and $\dot{S}$ to denote the universal additive set, where $\dot{S} = \bigcup_{i=1}^{n} \dot{S}_i$.

(ii) The TA randomly chooses q secret keys from S to generate a subset S_a and divides the remaining $nc - q$ secret keys into n random disjoint parts, denoted as $\widehat{S}_i$, $i = 1, 2, \ldots, n$, which is called subtractive secret key set. Among them, there are $nc - q - n \times \lfloor (nc-q)/n \rfloor$ subtractive sets containing $\lfloor (nc-q)/n \rfloor + 1$ keys, and the other $n(1 + \lfloor (nc-q)/n \rfloor) - nc + q$ sets contain $\lfloor (nc-q)/n \rfloor$ keys. We also use $\widehat{S}$ to denote the universal subtractive set, where $\widehat{S} = \bigcup_{i=1}^{n} \widehat{S}_i$. It is clear that $\dot{S} = \widehat{S} \bigcup S_a$.

(iii) Let $S_i = \dot{S}_i \bigcup \widehat{S}_i$ for $i = 1, 2, \ldots, n$, and each S_i is sent to the user i as encryption key set. Also, all the keys in S_a are sent to the aggregator as decryption keys.

The signing key is also generated in this phase. For each user i $(i = 1, 2, \ldots, n)$, the TA generates a pseudo ID $Uid_i = \{0,1\}^{\lceil \log_2 n \rceil}$ for him in each period, a secret signing key $sk_i = x_i \xleftarrow{R} \mathbb{Z}$, and a public signing key $pk_i = g_2^{x_i}$ for the pseudo ID.

5.3. Enc&Sign. In each period t, before the users upload their data to the aggregator, a sequence number $\text{seq}(i) \in [1,n]$ is generated for user i, where $i = 1, 2, \ldots, n$. $\{\text{seq}(i)\}_{i=1,2,\ldots,n}$ is a permutation of $\{1, 2, \ldots, n\}$, which is used to scramble the order of users' data. Each user i encrypts his/her data according to $\text{seq}(i)$, and the aggregator does not know the owner of the jth data after decryption, where $j = 1, 2, \ldots, n$, because $\{\text{seq}(i)\}_{i=1,2,\ldots,n}$ is unknown for him and thus cannot get i so that $\text{seq}(i) = j$. Considering security issues, the sequence number should be changed randomly. We emphasize that the sequence number can be generated by the TA or through communication among the users.

For user i, although he/she only owns l-bit data, he/she has to upload $n * l$ bits' ciphertext to hide his/her data; otherwise the connection between his/her identity and d^i can be easily found. Therefore, each user has to compute extra $(n-1)l$ bits' ciphertext.

Input:
For each user i $(i = 1, 2, \ldots, n)$, input his/her pseudo ID Uid_i, secret encryption key set S_i, and secret signing key $sk_i = x_i$.
Each user uploads his/her data $d^i \in \{0,1\}^l$ in the time period t.
The symbol $a \mid b$ represents the concatenation of a and b and $\oplus h_{s\in S_i}(x)$ denotes the exclusive-or of all the results of function h for each element s in S_i.
Output:
The user i outputs e^i and σ^i as follows:
begin
(1) Generate n random l-bit strings k_j^i $(j = 1, 2, \ldots, n)$ using
$$k_j^i = \oplus h_{s\in S_i}(t \mid j)$$
(2) The data d^i is encrypted as:
$$e^i = \left(\{0\}^l \oplus k_1^i\right) \mid \left(\{0\}^l \oplus k_2^i\right) \mid \cdots \mid \left(d^i \oplus k_{\mathrm{seq}(i)}^i\right) \mid \cdots \mid \left(\{0\}^l \oplus k_n^i\right)$$
(3) The signature is computed as:
$$\sigma^i = H^{x_i}\left(Uid \mid e^i\right) \in G_1$$
(4) Output e^i and σ^i.
end

Algorithm 1: Enc&Sign.

To generate $n * l$ bits' ciphertext, user i first uses his/her encryption keys as the secret key of $h_s(x)$ to generate $n * l$ bits' one-time pad, $k_1^i, k_2^i, \ldots, k_n^i$. Notice that all k_j^i are different, because each of them is generated by different parameter $t \mid j$. To scramble the order of d^i, we encrypt d^i with $k_{\mathrm{seq}(i)}^i$ instead of k_i^i. Furthermore, l-bit zero string is encrypted with k_j^i $(j \neq \mathrm{seq}(i))$. Thus, n encrypted l-bit strings are obtained: $\{0\}^l \oplus k_1^i, \{0\}^l \oplus k_2^i, \ldots, d^i \oplus k_{\mathrm{seq}(i)}^i, \ldots, \{0\}^l \oplus k_n^i$. Then all of them are concatenated one by one to generate e^i. If any user does not have data to upload, he/she can simply set his/her data to 0.

Finally, the signature σ^i is generated. Each user executes Algorithm 1 and then sends e^i and σ^i to the aggregator with his/her pseudo ID Uid_i.

5.4. Verify&Dec. The aggregator runs Algorithm 2 to fetch the data. After receiving all the ciphertexts from n users, the aggregator first leverages users' public keys to verify their signatures. If the algorithm outputs −1, which means some signatures are invalid, the aggregator discards the invalid data and asks for a retransmission. Otherwise all the users' original data can be recovered with the aggregator's secret key.

To decrypt all the users' data, the aggregator needs to take exclusive-OR on all the ciphertexts. Let $e^a = e^1 \oplus e^2 \oplus \cdots \oplus e^n$. Divide e^a into n parts as $e^a = e_1^a \mid e_2^a \mid \cdots \mid e_n^a$, and each part is a l-bit string. We know that $e_{\mathrm{seq}(i)}^a$ is the ciphertext of d^i, and:

$$\begin{aligned} e_{\mathrm{seq}(i)}^a &= k_{\mathrm{seq}(i)}^1 \oplus k_{\mathrm{seq}(i)}^2 \oplus \cdots \oplus k_{\mathrm{seq}(i)}^n \oplus d^i \\ &= \left(\oplus h_{s\in\hat{S}}(t \mid \mathrm{seq}(i))\right) \oplus \left(\oplus h_{s\in\dot{S}}(t \mid \mathrm{seq}(i))\right) \oplus d^i \quad (4) \\ &= \left(\oplus h_{s\in S_a}(t \mid \mathrm{seq}(i))\right) \oplus d^i = c_{\mathrm{seq}(i)} \oplus d^i \end{aligned}$$

Therefore, the aggregator first computes C in Step (2), and uses C to decrypt all the ciphertexts. The original data D is output in Step (3), which can be divided as $D = \{m[1, l], m[l+1, 2l], \ldots, m[(n-1)l + 1, nl]\}$, where each $m[x, y]$ is a user's original data. However, the aggregator cannot link each data with its owner, because $\mathrm{seq}(i)$ is unknown for him.

If the aggregator finds any abnormal data in D, for example, $m[x, y]$, it can request the TA to recover the identity of the malicious user. The aggregator sends all the data containing e^i and σ^i as well as x and y to the TA. If the $\mathrm{seq}(i)$ is known by the TA, it can directly find the malicious users. Otherwise, the TA recovers the corresponding secret encryption keys and real identities from the pseudo IDs, decrypts all the data and reveals the malicious users' identities.

6. Security Analysis

In this section, we analyze our framework and elaborate how our protocol can achieve the design goals under the security model. Specifically, we mainly focus on the following three aspects: why our protocol can hold *k-source anonymity* so that the only knowledge the server can get is that the data owner is one of the k participants in the system, why the collusion attack cannot help adversaries to recover the users secret keys, and why the data integrity can be protected and the identity authentication can be guaranteed.

6.1. Our Protocol Is k-Source Anonymous. As the definition of *k-source anonymity* listed in Section 4, we want to prove that if we interchange any two users' data in the same interval, the adversary including the aggregator cannot efficiently tell the difference.

Given a group of participants $U = \{u_1, u_2, \ldots, u_n\}$, and their corresponding sensing data $D = \{d^1, d^2, \ldots, d^n\} \in M^n$, where $M = \{0,1\}^l$ is the message space of d^i, $i = 1, 2, \ldots, n$. Each user u_i runs Algorithm 1 to encrypt his/her data d^i and sends the generated ciphertext e^i to the aggregator. Note that e_i has been defined in Step (2) of Algorithm 1. It is obvious

Input:
For each user i ($i = 1, 2, \ldots, n$), input his/her pseudo ID Uid_i, public signing key $pk_i = g_2^{x_i}$ and uploaded data e^i and σ^i.
The time period t and the aggregator's decryption keys are requested.
The symbol | and $\oplus$ have the same meaning in Algorithm 1.
Output:
The aggregator outputs all the users' original data $D = d^1 \mid d^2 \mid \cdots \mid d^n$ as follows:
begin
(1) Aggregate all the signatures and verify them:

$$\widehat{e}\left(\prod_{i=1}^{n}\sigma_i, g_2\right) \stackrel{?}{=} \prod_{i=1}^{n}\widehat{e}\left(H\left(Uid \mid e^i\right), pk_i\right)$$

(2) If the equation in Step (1) does not hold, the algorithm outputs −1, otherwise continue to compute:

$$c_j = \oplus h_{s\in S_a}(t \mid j) \quad \text{for } j = 1, 2, \ldots, n$$
$$C = c_1 \mid c_2 \mid \cdots \mid c_n$$

(3) The aggregator calculates the final result as:

$$D = e^1 \oplus e^2 \oplus \cdots \oplus e^n \oplus C.$$

(4) Output D.
end

Algorithm 2: Verify&Dec.

that the length of e^i is $n * l$ bits. Here we divide e^i into n parts as follows:

$$e^i = e_1^i \mid e_2^i \mid \cdots \mid e_n^i, \tag{5}$$

where $e_j^i = \{0\}^l \oplus k_j^i$ for $j = 1, 2, \ldots, n$, $j \neq \text{Seq}(i)$, and $e_j^i = d^i \oplus k_j^i$ for $j = \text{Seq}(i)$. After all the users' data have been collected in this interval, the knowledge what the aggregator learns can be represented as:

$$V = \left(e_1^1 \mid e_2^1 \mid \cdots \mid e_n^1, e_1^2 \mid e_2^2 \mid \cdots \mid e_n^2, \ldots, e_1^n \mid e_2^n \mid \cdots \mid e_n^n\right), \tag{6}$$

Let any two participants switch their data, denoted as u_i and u_j, where $1 \le i \le j \le n$, so all the users' original dataset is: $D' = \{d^1, \ldots, d^{i-1}, d^j, d^{i+1}, \ldots, d^{j-1}, d^i, d^{j+1}, \ldots, d^n\}$. Then the aggregator's knowledge is changed to:

$$\begin{aligned} V' = \big(&e_1^1 \mid \cdots \mid e_{i-1}^1 \mid e_j^1 \mid e_{i+1}^1 \mid \cdots \mid e_{j-1}^1 \mid e_i^1 \mid e_{j+1}^1 \mid \cdots \mid e_n^1, e_1^2 \mid \cdots \mid \\ &e_{i-1}^2 \mid e_j^2 \mid e_{i+1}^2 \mid \cdots \mid e_{j-1}^2 \mid e_i^2 \mid e_{j+1}^2 \mid \cdots \mid e_n^2, \ldots, e_1^n \mid \cdots \mid e_{i-1}^n \mid e_j^n \mid \\ &e_{i+1}^n \mid \cdots \mid e_{j-1}^n \mid e_i^n \mid e_{j+1}^n \mid \cdots \mid e_n^n\big) \end{aligned} \tag{7}$$

According to the definition, if we want to prove that our protocol is *k-source anonymous*, it is equivalent to prove:

$$V \stackrel{c}{\equiv} V' \tag{8}$$

holds for any i, j where $1 \le i \le j \le n$, and $\forall D \in M^n$.

To prove the correctness of this equation, we construct a simulator S to run the same protocol. First, the S generates a pseudorandom permutation function $f_p : [1, n] \to [1, n]$, and produces a permutation of $[1, n]$ as $[f_p(1), f_p(2), \ldots, f_p(n)]$. Then, S permutes the original dataset D to be:

$$D'' = \left\{d^{f_p(1)}, d^{f_p(2)}, \ldots, d^{f_p(n)}\right\} \tag{9}$$

The protocol is executed with the input of D'', and outputs the corresponding result:

$$\begin{aligned} V'' = \big(&e_1^{f_p(1)} \mid e_2^{f_p(1)} \mid \cdots \mid e_n^{f_p(1)}, e_1^{f_p(2)} \mid e_2^{f_p(2)} \mid \cdots \mid e_n^{f_p(2)}, \ldots, \\ &e_1^{f_p(n)} \mid e_2^{f_p(n)} \mid \cdots \mid e_n^{f_p(n)}\big). \end{aligned} \tag{10}$$

Because we cannot distinguish D and its pseudorandom permutation D'' in polynomial time, the V and V'' are computationally indistinguishable. Therefore, the following equation holds:

$$V \stackrel{c}{\equiv} V''. \tag{11}$$

Similarly, we can draw the conclusion:

$$V' \stackrel{c}{\equiv} V''. \tag{12}$$

Otherwise, D' and its pseudorandom permutation D'' can be distinguished in polynomial time. Therefore, we have:

$$V \stackrel{c}{\equiv} V''. \tag{13}$$

6.2. Our Protocol Is Collusion-Resilient. The ability to resist collusion attack depends on the size of each user's encryption keys and the aggregator's decryption keys. If we increase the size of S_i, or c, and the size of S_a, or q, the security level can be enhanced. Furthermore, when more users participate in the aggregation, our protocol can achieve better security.

In our scheme, the malicious users may collude to recover the aggregator's decryption keys, or collude with the aggregator to recover the other honest users' encryption keys. Let p_u denote the probability with which an honest user's key can be guessed successfully in a single trial, p_a denote the probability to recover the aggregator's key in a single trial, and

γ denote the proportion of malicious users who collude with the aggregator. As we can see in [10], we have:

$$p_u \leq \frac{1}{\binom{(1-\gamma)nc}{c} \cdot \binom{(1-\gamma)n\lfloor (nc-q)/n \rfloor}{\lfloor (nc-q)/n \rfloor}}, \quad p_a \leq \frac{1}{\binom{(1-\gamma)nc}{q}} \tag{14}$$

Assuming that there are at most 30% malicious users in the group, and the security requirements are $p_u \leq 2^{-80}$ and $p_a \leq 2^{-80}$. When the number of participants is 10^2, c and q can be set as 7 and 13 respectively, which means that the aggregator owns 13 decryption keys and every user owns 14 encryption keys at most. When the number of participants reaches 10^3, we set the $c = 5$ and $q = 9$.

Therefore, we can see when we set $n = 100, c = 7$, and $q = 13$, or $n = 1000, c = 5$, and $q = 9$, $p_u \leq 2^{-80}$ and $p_a \leq 2^{-80}$ are satisfied. Because the probability to recover secret keys in a single trial is not bigger than 2^{-80}, our protocol is collusion-resilient. Even if the number of users changes, we can adjust c and q to resist collusion attack.

6.3. Our Protocol Achieves Authentication and Data Integrity. Our protocol leverages aggregate signature as a built-in block to achieve authentication and the users' data integrity. The adversary can break authentication and data integrity if and only if it can forge the aggregate signature. However, the aggregate signature described in Section 4 is proven to be secure under the aggregate chosen-key security model [24]. We know the probability with which the adversary can generate valid aggregate signature is negligible. Furthermore, each user's data is bound with a pseudo ID. Therefore, authentication and data integrity are achieved.

7. Performance Evaluation

In this section, we implement our system and evaluate the performance of each instance of progress in our protocol, which demonstrates the efficiency and feasibility of our system. The comparison with other existing aggregation protocols is also performed with experiments and theoretical analysis.

As we can see, the main cost for the participants in our protocol is to encrypt and sign the uploaded data. We simulate participants' steps to examine the computational cost and elaborate the comparison with previous work. The aggregator's efficiency is affected by the verification and decryption, which is also shown to be accepted in the experiment.

7.1. Implementation and Experimental Settings. We implement our protocol in a desktop with Intel Core i7-4790 3.60 GHz CPU and 8 GB memory. The compilation environment is Visual Studio 2013 in Windows 10. And the cryptographic functions in the algorithm are provided by the MIRACL library. We use the same hash function as that in [25], which uses HMAC-SHA512 as the pseudorandom function. For each l-bit raw data, if $l < 512$, the HMAC-SHA512 outputs a pseudorandom 512-bit string, which is truncated into l-bit substrings and taken exclusive-OR on all these substrings. When $l > 512$, we take several HMAC-SHA512 functions and concatenate their output, while the remaining part can use the same method as the condition $l < 512$.

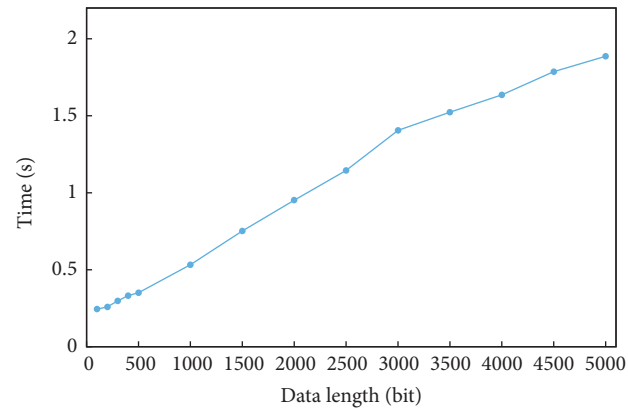

Figure 2: Computational cost of encryption at user side.

All the users' data is generated by taking a uniform sample from $[0, 2^l - 1]$. The reason why we do not adopt the real-life data is that the value of the user's data does not affect the efficiency of our protocol. We only take experiments on the computational cost in both sides but do not consider the communication cost between them, because each user's data size is $n * l$ bits, generally tens of or several kilobytes, which can be transmitted in short time. All the algorithms in our protocol are executed for 10 times. We take the average running time as the final output.

7.2. Computational Cost at User Side. All the computational cost at user side depends on Algorithm 1. In our scheme, if a user wants to encrypt the raw data, he/she needs to compute $n * k$ pseudorandom functions, where k is the number of the user's keys; namely, $k = |S_i|$, for $i = 1, 2, \ldots, n$.

Figure 2 shows the result of our computation time. We set the group size as $n = 1000$, $\gamma = 0.3$, $p_u < 2^{-80}$, and $p_a < 2^{-80}$. According to [10], we let $c = 5$, which means that each user owns nearly 10 encryption keys. During each time period, each user computes 10 pseudorandom functions and takes 10000 exclusive-OR operations on l-bit data. We can see that our algorithm is efficient and can be applied in real environment. In this figure, the encryption time increases linearly with the data length. If the data length is smaller than 2000 bits, the encryption time is no more than one second. When the data length reaches 5000 bits, we can finish the encryption within two seconds.

Figure 3 shows the relationship between the encryption time and the number of users when the data length is 1000 bits. Obviously, as the number of users increases, each user needs to compute more ciphertexts. However, it only

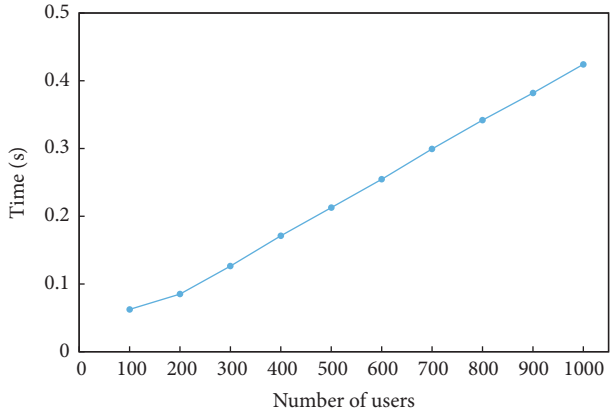

Figure 3: Computational cost of encryption at user side.

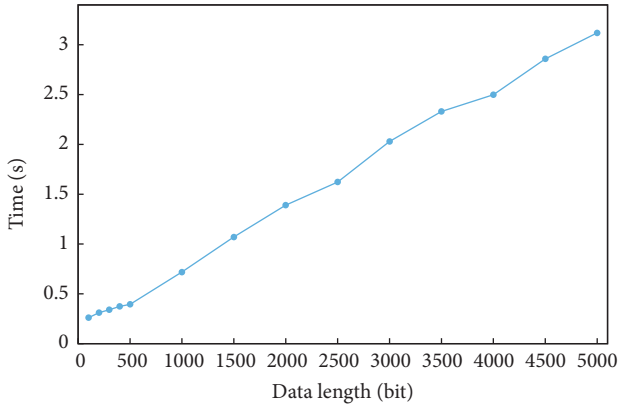

Figure 4: Computational cost of decryption at aggregator side.

takes 0.21 s for each user to take the encryption when the number of users increases to 500.

Here we compare our protocol with that in [11], to encrypt the raw data they have to compute 2 pseudorandom functions and take $2n$ exclusive-OR operations on l bits' data. Recall that although our computational cost is five times as much as that in [11], we can achieve more security properties. When the data length is 100 bits, our protocol costs 0.24 s and the scheme proposed in [11] costs 0.05 s. When the data length is 1000 bits, 0.58 s and 0.12 s are needed, respectively.

We emphasize that we do not list the details of the comparison of signature time and verification time here, because the signature time depends much on the aggregate signature scheme. In fact, we only need to take 30 ms to sign the final ciphertext, when the data length is 5000 bits.

7.3. Computational Cost at Aggregator Side. Different from [11], the aggregator owns decryption keys to prevent others from obtaining all the raw data. In our scheme, the aggregator needs to compute nq pseudorandom functions and take $n * (n + q)$ exclusive-OR operations on l bits' data. However, it takes $n * (n+q)$ exclusive-OR operations on l bits' data in [11].

If we set $n = 1000$, $\gamma = 0.3$, $p_u < 2^{-80}$, and $p_a < 2^{-80}$, the result is shown in Figure 4. The aggregator only needs to take several seconds to decrypt all the data when the data length is 5000 bits, and it only takes nearly one more second than [11] to resist the collusion attack. When the times of exclusive-OR operations increase, the time to compute pseudorandom functions does not dominate the whole running time any more. Experiment shows that it takes 0.26 s for decryption when l equals 100, while 0.031 s is needed in [11]. When l reaches 1000, our protocol and [11] consume 0.72 s and 0.23 s, respectively.

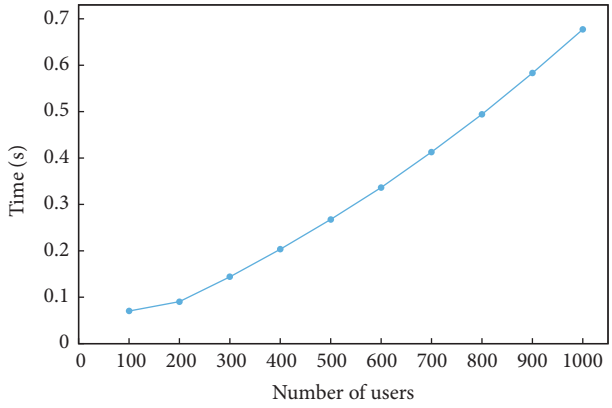

Figure 5: Computational cost of decryption at aggregator side.

When the data length is set as 1000 bits, the computational cost at aggregator side is shown in Figure 5. The decryption time is proportional to the number of users. If the group includes 100 users, the decryption can be finished in 0.1 s. Even though the number of users grows up to 1000, the decryption time does not exceed one second.

8. Conclusion

In this paper, we propose a novel protocol to allow an untrusted aggregator to compute any aggregation functions based on all users' data. Collusion resistance is also achieved even if part of malicious users collude with the untrusted aggregator. The data collection can be finished in one-round communication, so bidirectional communication channel is not needed in our protocol. We also protect all users' data integrity by leveraging the aggregate signature. Security analysis shows that k-source anonymity is achieved. Through extensive performance evaluations, we have demonstrated that the proposed scheme is efficient at user/aggregator side. In our scheme, dynamic joining and exit of users have not been discussed. In the future, we will continue our efforts to address this issue.

Notations

n: The number of users in the scheme
$\dot{S}$: The universal additive secret key set
$\dot{S}_i$: The user i's additive secret key set
$\widehat{S}$: The universal subtractive secret key set
$\widehat{S}_i$: The user i's subtractive secret key set
S_i: The user i's encryption key set, $S_i = \dot{S}_i \bigcup \widehat{S}_i$

S_a: The aggregator's decryption key set
c: The size of $\dot{S}_i$
q: The size of S_a
d^i: The user i's data
l: The data length
t: The time period
H: A hash function, $H : \{0,1\}^* \rightarrow G_1$
$h_s(x)$: A function indexed by s in a pseudorandom function family $\mathscr{H}_{l,m+\lceil \log_2 n \rceil,l} = \{h_s : \{0,1\}^{m+\lceil \log_2 n \rceil} \rightarrow \{0,1\}^l\}_{s \in \{0,1\}^l}$
Uid_i: The user i's pseudo ID
pk_i: The user i's public signing key for Uid_i
sk_i: The user i's secret signing key for Uid_i.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant nos. 61402037, 61272512, and 61300172), National Key Research and Development Program 2016YFB0800301, and DNSLAB, China Internet Network Information Center, Beijing 100190.

References

[1] A. Thiagarajan, L. Ravindranath, K. LaCurts et al., "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pp. 85–98, ACM, November 2009.

[2] M. Mun, S. Reddy, K. Shilton et al., "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 55–68, ACM, Kraków, Poland, June 2009.

[3] S. Consolvo, D. W. McDonald, T. Toscos et al., "Activity sensing in the wild: a field trial of ubifit garden," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1797–1806, ACM, Florence, Italy, April 2008.

[4] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 174–186, ACM, Breckenridge, Colo, USA, June 2008.

[5] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation," in *Proceedings of the IEEE INFOCOM*, pp. 2634–2642, IEEE, Turin, Italy, April 2013.

[6] T. Jung, X.-Y. Li, and M. Wan, "Collusion-tolerable privacy-preserving sum and product calculation without secure channel," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 45–57, 2015.

[7] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 111–125, Springer, 2013.

[8] F. Qiu, F. Wu, and G. Chen, "SLICER: a slicing-based K-anonymous privacy preserving scheme for participatory sensing," in *Proceedings of the 10th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '13)*, pp. 113–121, October 2013.

[9] F. Qiu, F. Wu, and G. Chen, "Privacy and quality preserving multimedia data aggregation for participatory sensing systems," *IEEE Transactions on Mobile Computing*, vol. 14, no. 6, pp. 1287–1300, 2015.

[10] Q. Li, G. Cao, and T. F. La Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 2, pp. 115–129, 2014.

[11] Y. Zhang, Q. Chen, and S. Zhong, "Privacy-preserving data aggregation in mobile phone sensing," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 980–992, 2016.

[12] C. R. Perez, *Reputation-based resilient data aggregation in sensor network [M.S. thesis]*, Department of Electrical and Computer Engineering, Purdue University, 2007.

[13] C. R. Perez-Toro, R. K. Panta, and S. Bagchi, "RDAS: reputation-based resilient data aggregation in sensor network," in *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '10)*, pp. 1–9, June 2010.

[14] M. M. Groat, W. Hey, and S. Forrest, "KIPDA: K-indistinguishable privacy-preserving data aggregation in wireless sensor networks," in *Proceedings of the IEEE INFOCOM*, pp. 2024–2032, IEEE, April 2011.

[15] C. Li and Y. Liu, "SRDA: smart reputation-based data aggregation protocol for wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 105364, 10 pages, 2015.

[16] X. Du, M. Guizani, Y. Xiao, and H.-H. Chen, "Defending DoS attacks on broadcast authentication in wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pp. 1653–1657, Beijing, China, May 2008.

[17] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos, "User recruitment for mobile crowdsensing over opportunistic networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 2254–2262, IEEE, 2015.

[18] L. Gao, F. Hou, and J. Huang, "Providing long-term participation incentive in participatory sensing," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks (IEEE INFOCOM '15)*, pp. 2803–2811, Hong Kong, China, May 2015.

[19] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proceedings of the Theory of Cryptography Conference*, pp. 325–341, Springer, 2005.

[20] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: a secure hop-by-hop data aggregation protocol for sensor networks," *ACM Transactions on Information and System Security*, vol. 11, no. 4, article 18, 2008.

[21] 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, pp. 211–224, ACM, June 2008.

[22] I. Boutsis and V. Kalogeraki, "Privacy preservation for participatory sensing data," in *Proceedings of the 11th IEEE International Conference on Pervasive Computing and Communications (PerCom '13)*, pp. 103–113, San Diego, Calif, USA, March 2013.

[23] Y. Zhang, Q. Chen, and S. Zhong, "Efficient and privacy-preserving min and k th min computations in mobile sensing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 9–21, 2017.

[24] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 416–432, Springer, 2003.

[25] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 3, pp. 1–36, 2009.