Privacy-preserving Data Aggregation in Mobile Phone Sensing

Yuan Zhang, Qingjun Chen, Sheng Zhong

Abstract—Mobile phone sensing provides a promising paradigm for collecting sensing data and has been receiving increasing attention in recent years. Different from most existing works, which protect participants' privacy by hiding the content of their data and allow the aggregator to compute some simple aggregation functions, we propose a new approach to protect participants' privacy by delinking data from its sources. This approach allows the aggregator to get the exact distribution of the data aggregation, and therefore enables the aggregator to efficiently compute arbitrary/complicated aggregation functions.

In particular, we first present an efficient protocol that allows an untrusted data aggregator to periodically collect sensed data from a group of mobile phone users without knowing which data belongs to which user. Assume there are n users in the group. Our protocol achieves "n-source anonymity" in the sense that the aggregator only learns that the source of a piece of data is one of the n users. Then, we consider a practical scenario where users may have different source anonymity requirements and provide a solution based on dividing users into groups. This solution optimizes the efficiency of data aggregation and meets all users' requirements at the same time.

Index Terms—Privacy, data aggregation, cloud computing, security, mobile sensing

1 Introduction

Mobile phone sensing provides a new paradigm for people to efficiently perform sensing tasks. In a typical mobile phone sensing application, a data aggregator recruits a group of mobile phone users to perform sensing tasks. With various kinds of sensors embedded in their mobile phones, these users perform the sensing task and then send the data back to the data aggregator through the communication network. Due to the outstanding sensing ability of mobile phones in recent years smartphones and the ubiquitousness of mobile phone users, mobile phone sensing is gaining increasing attention from both industry and academia. A number of mobile phone sensing based applications have been developed across areas such as healthcare [12, 34], transportation [31, 42], environment monitoring [28, 32], etc.

In these applications, data collected by the aggregator often contains users' private information. For example, most applications for traffic or environment monitoring collect the user's physical location in addition to their direct POI (points of interest) e.g. the traffic congestion level or the noise level; most healthcare applications collect information relating to a users health such as weight and blood pressure. Concerned about their privacy, mobile phone users may refuse to participate in the sensing especially when the aggregator is untrusted. Thus, protecting participants' privacy is extremely important to mobile phone sensing applications.

Realizing the importance of privacy protection, researchers began to investigate privacy issues in mobile phone sensing and a few works [22, 23, 26, 36, 38, 39] on protecting participants' privacy have been carried out in recent years. Take [22] for an example, an efficient secure protocol is designed for an untrusted aggregator to compute the sum of all participants' time-series data. Before sending its data to the aggregator, each mobile phone user encrypts it using an additively homomorphic cipher. The encryption keeps the content of a user's data private from the aggregator and other users. Different cryptographic schemes are used in [36, 38, 39] to implement the same summation functionality of the aggregator. Based on the secure summation, a few functions on the data aggregation other than the sum, such as the average, the Max/Min etc. can also be computed without knowing each user's data.

We notice that all works above choose to protect users' privacy by "hiding" their data's contents. Protocols proposed in these works are specifically designed to compute a certain aggregation function without revealing each data's value. If we want to compute more than one different aggregation function, we often need to apply one specific protocol for each function, which is very inefficient. Furthermore, most aggregation functions studied in these works are simple functions such as sum, average, Max/Min etc. Non-linear functions such as variance, z-test function, F-test function etc. are rarely

- *Yuan Zhang, Qingjun Chen and Sheng Zhong (Corresponding Author) are with the State Key Laboratory for Novel Software Technology, Nanjing University, and also with the Computer Science and Technology Department, Nanjing University, Nanjing, 210023, China.*
  Emails: zhangyuan05@gmail.com, knight.chan319@gmail.com, zhongsheng@nju.edu.cn

This work was supported in part by NSFC-61321491, NSFC-61425024, NSFC-61300235, NSFC-61402223, and Jiangsu Province Double Innovation Talent Program.

studied.

Unlike these works, in this paper, we protect users' privacy by delinking data from its sources. In particular, we aim to design protocols that allow the data aggregator to periodically collect a random permutation of all users' data without being able to identify the source of any particular piece of data. This approach allows the aggregator to get the exact distribution of the data aggregation, and therefore enables the aggregator to efficiently perform complicated statistic analyses that are difficult to perform using protocols that hide the data's contents. In addition, letting the aggregator know the data's contents (rather than keeping it private) is necessary for some mobile sensing applications. One possible example would be the users' location data in transportation sensing applications. Users' location data is often required to be made available in order to enable accurate analysis results.

In practice, delinking data from its sources also provides satisfactory privacy protection for participants in many mobile sensing applications. For example, consider a mobile sensing application that monitors the trend of an epidemic (e.g. avian flu) in a city. The aggregator may want to collect citizens' body temperatures. Disguising every citizen's body temperature value protects citizens' privacy, as does disguising the connection of a citizen to their body temperature value. Consider an application that guarantees the source of data is hidden amongst all citizens in a city. If all citizens send their body temperatures using this application to the aggregator, the aggregator only knows a random permutation of all citizen's body temperatures. Although it can easily spot abnormal body temperatures that indicate some citizens are likely infected, it cannot know which particular citizens are infected, nor the body temperature of a particular citizen. Such an application protects all participants' privacy well. Even an infected citizen would not refuse to send its abnormal body temperature in such an application.

Our paper consists mainly of two parts. In the first part, we study how to delink the data from its sources in a general mobile sensing application. Suppose there are n users and one untrusted aggregator. We propose an anonymous data aggregation protocol that allows the aggregator to collect all users' data. Our protocol achieves "n-source anonymity" in the sense that the aggregator only learns that the source of any particular piece of data is one of the n users. In the second part of this paper, we consider the situation where n is very large and different users may have different privacy requirements. In order to improve the efficiency, we propose dividing users into groups according to their privacy requirements and allowing users in each group to execute the anonymous data aggregation protocol together. We provide an optimal grouping algorithm which finds an optimal grouping that meets all users' privacy requirements and minimizes the total amount of data received by the aggregator at the same time.

Our contribution can be summarized as:

- Proposing a new privacy-preserving approach for mobile phone sensing data aggregation that can be applied to arbitrary aggregation functions. Our approach does not assume that this aggregator is trusted and does not require data transmissions among sensor nodes.
- Presenting an anonymous data aggregation protocol that allows the data aggregator to recieve a random permutation of all users' data without knowing the source of any particular piece of data.
- Formally proving the anonymous data aggregation protocol is n-source anonymous when there are n honest users in the application.
- When the total number of users is large and users have different minimal privacy level requirements, proposing a grouping algorithm that can be used to find an optimal grouping. Grouping users this way and allowing users within each group to execute our anonymous data aggregation protocol together would satisfy all users' privacy requirements and optimize the entire data aggregation's efficiency at the same time.
- Performing experiments to show the efficiency of our protocols.

The rest of this paper is organized as follows. In Section 2, we review the related work. After introducing the preliminary knowledge of our problem and solutions in Section 3, we present our anonymous data aggregation protocol in Section 4. In Section 5, we show how to deal with an extreme scenario in which the total number of users is very large. We evaluate our solutions' performance in Section 6 and then discuss possible extensions of our protocols in Section 7. Finally, we conclude our paper in Section 8.

2 Related Work

Many previous works (e.g., [5, 6, 11, 15, 25, 46]) on privacy-preserving data aggregation assume a trusted aggregator, which is different from our scenario. For example, in [11], a trusted "Report Service" anonymizes all users' data and then sends the anonymized data for end use.

There have been a few works [22, 23, 26, 36, 38, 39] which investigate privacy issues in mobile phone sensing applications with an untrusted aggregator. However, all of these works aim to allow the aggregator to compute some specific functions (e.g., sum, Min/Max, etc.) on data aggregation and protect participants' data content from the aggregator at the same time. For example, Li et al.[22, 26] propose an excellent sum aggregation protocol based on an additively homomorphic encryption, and also construct a highly efficient Min computation protocol based on their sum aggregation protocol. Besides the good efficiency and smart protocol designs, Li et al.'s protocols adopt a very delicate key system so that their protocols can thwart collusion attacks and are able to efficiently handle users' dynamic joining and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2515513, IEEE Transactions on Information Forensics and Security

leaving. Different from these works, we aim to let the aggregator know the data content and delink data from its sources instead to protect users' privacy. Furthermore, our work allows the aggregator to recieve a random permutation of the data aggregation and thus allowing the aggregator to efficiently compute any aggregation function. We note that theoretically the secure sum computation protocol implemented in these works could be extended to achieve our goal using an idea from [22], but the extension would need to traverse the space of users' data and run the secure sum computation protocol once for every possible data value, and so it would not be efficient for an application with large data space.

In addition, we notice that there are a number of works [10, 19, 20, 37, 45, 47] that study similar data aggregation problems in wireless sensor networks. For example, in [19], Groat et al. propose a secure data aggregation protocol that can be used to compute the sum/Max/Min function of all senor nodes' data in a tree-structured sensor network, and meanwhile guarantees that each sensor node's data achieves a similar k-indistinguishability security. Their solution is constructed by cleverly selecting the fake data that is used to disguise the real one and fake data's injecting positions, instead of utilizing any homomorphic encryptions. In [45], Yang et al. first propose a method to enable each pair of neighbouring nodes to secretly share a key, and then let each node encrypt its data with an additively homomorphic cipher that is established on the shared keys. Yang et al.'s protocol is able to compute the sum/average function, and is robust against collusion and data loss. Despite the difference in the ways how their security guarantees and aggregation goals are achieved, these works' solutions [10, 19, 20, 37, 45, 47] have one thing in common. All these solutions require sensor nodes to communicate with other and to help each other in the secure aggregation process (e.g. to establish key pairs, to encrypt/disguise the data, and/or to pass on the data). This is quite different from our scenario in which mobile phone users in a data aggregation seldom communicate with each other directly.

Besides above works which study privacy-preserving data aggregations, there are a few works which specifically focus on protecting source privacy in mobile environment. Most of these works assume there is a multihop path between the source node and destination node, while our paper does not make such an assumption. For example, in [27], Li and Ren study how to hide the source location information from the destination node in a wireless sensor network, and propose a solution based on randomly selecting intermediate nodes to forward the message for the source node. In [21], Huang et al. propose a pseudo normal distribution-based phantom routing protocol to protect the privacy of the source. Their solution adds phantom nodes into the communication path and performs random walks within it. In [2], Abuzneid et al. propose a new communication protocol for wireless sensor networks that uses disposable IDs to identify sensor nodes and protect their location privacy. In particular, Conti et al. [9] provide an excellent survey on source location privacy protection in wireless sensor networks. In their comprehensive survey, authors summarize key concepts in source location privacy protection, and give a complete overview on the existing solutions and their corresponding adversary settings. Among all solution categories listed in the survey, our solution is similar to those using dummy data sources. As commented by the authors in [9], a few dummy-source-based solutions (e.g. [29]) cannot deal with an adversary that has global traffic knowledge. Different from these solutions, our solution can deal with global adversaries. Others that can deal with global adversaries (e.g. [4, 30, 41]) mostly focus on making the generation of dummy data appear real, thus assuming the distribution of real events is known or can be approached with the help of neighboring nodes. In this paper, we do not make such assumptions.

Our anonymous data aggregation protocol follows the idea of Chaum's DC-Net [8] which provides an anonymous broadcast protocol. In DC-Net, every two users share a secret key and there are $n(n-1)/2$ keys in a n-user system. In this paper, we choose a simplified keying mechanism which only requires n keys. Most follow-up works on DC-Net [18, 43, 44] aim to efficiently add fault-tolerance or cheating-tolerance properties to the original DC-net, which differs from our objective.

There are a few other protocols that also provide anonymizing communication functionality such as Mix-Net [7], Crowds [35], and CliqueNet [40]. Systems built on these protocols (e.g. [24]) can also get the exact data distribution and thus are able to obtain arbitrary aggregation functions. However, these protocols require that bidirectional communication channels exist among sensor nodes, which is usually not the case in a mobile sensing scenario as nodes (smartphone users) participating in the sensing may not know each other in advance.

3 Preliminaries

3.1 System Model and Problem Setup

System Model. We consider a general mobile sensing system that consists of one aggregator and n mobile phone users. Let $U = \{u_1, \ldots, u_n\}$ be the entire user group. We assume that there is a communication channel between each user and the aggregator. The communication channel could be 3G/4G, wifi or other kinds of channels which are supported by the mobile phones and the aggregator. Also, we assume that the communication channel is secured by both the aggregator and mobile phone users using public-key infrastructure (PKI) such as [14, 33, 48], so that messages' integrity and authenticity are protected. In addition, we assume the job owner notifies every participating user of the job description and the total number of participating users in the aggregation via a bulletin-board or a broadcast. Note that

Table 1
Notations

Symbol	Description
U	the entire user group
u_i	the ith user
$E_\oplus$	the ciper system
$\oplus$	the bit-wise XOR operation
$\mathcal{M}$	the message space
$\mathcal{P}$	the data aggregators view (i.e. the messages she receives, her coin flips and her input)
$\overset{c}{\equiv}$	computationally indistinguishability of two random variable ensembles
$\mathcal{H}_{l,m,o}$	a pseudo-random function family $\mathcal{H}_{l,m,o} = \{h_s : \{0,1\}^m \to \{0,1\}^l\}_{s\in\{0,1\}^o}$
$h_s(t)$	a function indexed by s in $\mathcal{H}_{l,m,o}$
a_i	user i's minimal privacy level requirement
$\mathcal{G}$	a feasible grouping solution
G_j	a set that contains the index of all users in group G_j
$C(\mathcal{G})$	the cost of grouping solution $\mathcal{G}$
$\mathcal{G}^\star(x)$	a feasible grouping solution with the minimal cost when user group is $1,\ldots,x$
$f(x)$	the cost of $\mathcal{G}^\star(x)$
$g(x)$	the smallest index contained in the group that has user x inside in $\mathcal{G}^\star(x)$
n	the total user number
l	the data length (the bit number of data)
q	users' maximum privacy level

we do not allow the total number of participants to increase during the aggregation process or after the process starts. Furthermore, we assume that there is a trusted authority which can help all mobile users to establish the key system for once and similar secure communication channel exists between the authority and every mobile phone. The authority should NOT be selected by the aggregator. In practice, one possible authority could be the underlying communication service provider or some independent auditing company.

A user performs the sensing task periodically and sends time-series data to the aggregator through the secured channel. The aggregator collects all user data and performs further analysis on the data aggregation.

Privacy Definition. We define our privacy requirement in a standard adversary model called the *semi-honest* model [17]. In particular, the semi-honest model assumes that all involved parties will follow the protocol, but may attempt to derive extra knowledge about other parties' private inputs. A user's privacy is defined or evaluated as the anonymity level of its data. Specifically, if the best an adversary can learn is that the source of its data is one of k users ($k \in \{1,\ldots,n\}$), we say this user's data has *k-source anonymity* or this user has a privacy level of k. Naturally, the larger k's value is, the better the corresponding privacy is.

If all users' data has k-source anonymity in the data aggregation process/protocol, we say the process has k-source anonymity.

Formally, we use *computational indistinguishability* [17] from cryptography to model the anonymity of messages' sources and define our privacy as follows.Informally speaking, the definition states that if we switch any two users' data and the aggregator cannot efficiently notice any difference, the aggregation protocol with k-users is k-source anonymous.

Definition 1: A data aggregation process or a protocol is **k-source anonymous** if for any user group U that contains k users, any two users u_i and u_j in it, and for any data aggregation sample $\{d^1,\ldots,d^k\} \in \{\mathcal{M}\}^k$,

$$\mathcal{P}(\ldots,u_i(d^i),\ldots,u_j(d^j),\ldots) \overset{c}{\equiv} \mathcal{P}(\ldots,u_i(d^j),\ldots,u_j(d^i),\ldots),$$

where M denotes the message space, $\mathcal{P}(\ldots,u_i(x_i),\ldots)$ denotes the data aggregator's view (i.e. the messages she receives, her coin flips and her input) when running protocol with x_i ($x_i \in \{d^1,\ldots,d^k\}$) as u_i's input ($i = 1,\ldots,k$) and $\overset{c}{\equiv}$ denotes computationally 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.

3.2 A Bitwise-XOR Homomorphic Cipher

Our protocols are based on a cipher system that has the *bitwise XOR homomorphic property*. Given the ciphertexts of a group of plaintexts, the bitwise XOR of these plaintexts can be efficiently computed without decryption.

Below we describe a cipher system which has the bitwise XOR homomorphic property. After all users use their private keys to encrypt their data, the aggregator can get the bitwise XOR of all users' data simply by computing the bitwise XOR of all users' ciphertexts. The underlying idea of this cipher system is very similar to that of a stream cipher or an additively homomorphic cipher proposed in [6]. Specifically, every user uses their private key to generate a random bit string of the same length as its plaintext. Then, the encryption is done by performing a bit-wise XOR operation on the plaintext with the random bit string. All users' private keys are specifically designed so that the bitwise XOR of all random bit strings is equal to the bit string of zero. It is easy to verify that the bitwise XOR of all users' ciphertexts equals the bitwise XOR of their plaintexts.

Assume every user's data can be represented by an l-bit string and the nonce information that specifies the time period t can be represented by a m-bit string. Denote by $h_s(t)$ a function indexed by s in a pseudo-random function family $\mathcal{H}_{l,m,o} = \{h_s : \{0,1\}^m \to \{0,1\}^l\}_{s\in\{0,1\}^o}$. A possible efficient implementation of $\mathcal{H}_{l,m,o}$ and $h_s(t)$ can be found in [6].

Denote by $E_\oplus$ the cipher and by $\oplus$ the bit-wise XOR operation. $E_\oplus$ generates keys, encrypts, and decrypts as follows.

Key Generation and Distribution: we assume there is a trusted authority TA who helps the users to establish the key system. TA first picks $S_0,\ldots,S_{n-1} \in \{0,1\}^l$ uniformly and independently. For each user i ($i = 1,\ldots,n$), TA computes $s_a^i = S_{i-1}$ and $s_b^i = S_{(i \bmod n)}$ and sends them to user i. User i keeps (s_a^i, s_b^i) private and uses it as its private key.

Encryption: to encrypt a bit-string $x_i \in \{0,1\}^l$ in time period t, user i chooses pseudo-random functions $h_{s_a^i}$ and $h_{s_b^i}$ from $\mathcal{H}_{l,m,l}$, generates a random bit string

$k_i(t)$ that equals $h_{s_a^i}(t) \oplus h_{s_b^i}(t)$, and then computes the ciphertext as

$$\overline{x_i} = x_i \oplus k_i(t).$$

Decryption: the aggregator decrypts the bitwise XOR of all users' data by computing

$$\overline{x_1} \oplus \ldots \oplus \overline{x_n}.$$

4 The Anonymous Data Aggregation Protocol

In this section, we present our anonymous data aggregation protocol. Our protocol achieves n-source anonymity in the sense that for any particular piece of data the aggregator only learns that the source is one of the n users.

Denote our protocol by $P_{E_\oplus}$. $P_{E_\oplus}$ consists of only one round. In this round, every user reports an encrypted nl-bit string to the aggregator. And the aggregator can easily compute all user's data based on all ciphertexts it receives.

We assume that every user holds an unique sequence number before the protocol starts.[1] A user's sequence number determines the positions of its real data in the encrypted bit string. Denote by $Seq(i) \in [1, n]$ the sequence number of user i ($i = 1, \ldots, n$). Note that $Seq(i)$ ($i = 1, \ldots, n$) is actually a permutation of $\{1, 2, \ldots, n\}$.

First, each user i encrypts the bit string of its data d^i using $E_\oplus$ to get a nl-bit string and sends it to the aggregator. Specifically, user i chooses pseudo-random functions $h_{s_a^i}$ and $h_{s_b^i}$ from $\mathcal{H}_{l,m+\lceil \log_2 n \rceil, l}$, and generates n random l-bit strings $k_j^i (j = 1, \ldots, n)$ using

$$k_j^i = h_{s_a^i}(t|j) \oplus h_{s_b^i}(t|j) \qquad (1)$$

where $t|j$ is the concatenation of t and j. After all k_j^i ($j = 1, \ldots, n$) are constructed, user i uses $k_{Seq(i)}^i$ to encrypt its real data d^i and uses k_j^i ($j \neq Seq(i)$) to encrypt dummy data 0^l. Obviously, user i gets n encrypted l-bit strings: $\{0\}^l \oplus k_1^i, \ldots, \{0\}^l \oplus k_{Seq(i)-1}^i, d^i \oplus k_{Seq(i)}^i, \{0\}^l \oplus k_{Seq(i)+1}^i, \ldots, \{0\}^l \oplus k_n^i$. To get the encrypted nl-bit string, user i concatenates the n encrypted l-bit strings one by one.

Second, the aggregator computes all users' data based on all ciphertexts it receives. Regard the received nl-bit string as n parts and each part l-bit. Every part inside user i's ciphertext is an encryption of the bit string $\{0\}^l$ except that the $Seq(i)$-th part is the encryption of user i's real data d^i. Since $Seq(i)$ ($i = 1, \ldots, n$) is a permutation of $\{1, 2, \ldots, n\}$ and $E_\oplus$ is bitwise XOR homomorphic, it is easy to see that the bitwise XOR of all bit strings encrypted by all users equals the concatenation of all users' real data. Denote the bit string received from user i by $\overline{e^i}$. The aggregator performs the decryption by computing $m = \overline{e^1} \oplus \ldots \oplus \overline{e^n}$. As we have mentioned, m is the concatenated bit string of all users' real data. So the aggregator can get all users' data by breaking m into n parts with equal length.

Figure 1 illustrates the main aggregation procedure (in one time period) using a simple diagram. In this example, the bit strings that users send and that the aggregation receives consists of 3 parts as there are 3 users. For each user, it fills one part of the bit string with their real data while filling the other two with dummy data. For instance, for user 1, as its sequence number is 3, the 3rd part of its bit string should be filled with encrypted real data 11, while the other two parts should be filled with encrypted dummy data 0. Then, all users send their ciphertexts to the aggregator. After the aggregator receives all three ciphertexts, it performs the decryption by XORing three ciphertexts and gets all users' data by breaking the decrypted bit string into 3 parts with equal length, where each part stands for the real data of one user. In this example, the decrypted bit string is 110011011011 and the aggregator can get $1100_2 = 12_{10}$, $1101_2 = 13_{10}$ and $1011_2 = 11_{10}$ which are the real data of all users.

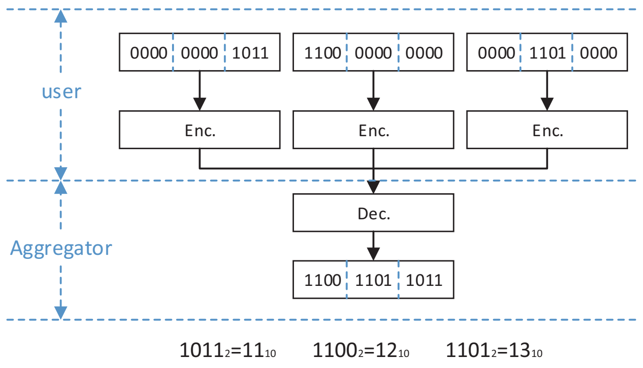


Figure 1. an example of our protocol when $Seq = \{3, 1, 2\}$ and $userdata = \{11, 12, 13\}$

We summarize our protocol in Algorithm 1.

Complexity Analysis. Our protocol has very good computational efficiency: Once the keys have been established, on average each user mainly needs $2n$ l-bit XOR operations, $2n$ hashing operations during each time period; the aggregator mainly needs $n - 1$ nl-bit XOR operations during each time period.

In terms of communication efficiency, our protocol requires each user to send nl bits on average during each time period. Meanwhile, the aggregator needs to receive $n^2 l$ bits.

Security Analysis. Regarding the security of our anonymous data aggregation protocol, we have the following result.

Theorem 2: The anonymous data aggregation protocol

1. The unique sequence numbers could be assigned by a trusted authority or be a result of the users' communications. Note that when the sequence numbers are determined through users' communications, users have to know each other in advance. Thus, this method can only be applied in cases that users are able and willing to communicate with each other.

Algorithm 1 Anonymous Data Aggregation Protocol

```
Input:
    a group of $n$ users and an aggregator;
    user $i$ $(i = 1, \ldots, n)$ has an unique sequence number $Seq(i) \in [1, n]$ and a private key $(s_a^i, s_b^i)$;
    in each time period $t$, user $i$ $(i = 1, \ldots, n)$ has an $l$-bit string $d^i$.
    In the protocol, $x|y$ is the concatenation of $x$ and $y$.
Output:
    the aggregator outputs $D$, the set of all users' bit strings;
1: the aggregator sets $D$ to the empty set $\emptyset$.
2: user $i$ $(i = 1, \ldots, n)$ chooses pseudo-random functions $h_{s_a^i}$ and $h_{s_b^i}$ from $\mathcal{H}_{l,m+\lceil \log_2 n \rceil, l}$, and generates $n$ random $l$-bit string $k_j^i (j = 1, \ldots, n)$ using
       $k_j^i = h_{s_a^i}(t|j) \oplus h_{s_b^i}(t|j)$
3: user $i$ $(i = 1, \ldots, n)$ encrypts $d^i$ as
       $\overline{e^i} = (\{0\}^l \oplus k_1^i) | \ldots | (\{0\}^l \oplus k_{Seq(i)-1}^i) | (d^i \oplus k_{Seq(i)}^i) | (\{0\}^l \oplus k_{Seq(i)+1}^i) | \ldots | (\{0\}^l \oplus k_n^i)$
   and sends the ciphertext $\overline{e^i}$ to the aggregator.
4: the aggregator computes $m = \overline{e^1} \oplus \ldots \oplus \overline{e^n}$ and set the result set as
       $D = \{m[1, l], m[l+1, 2l], \ldots, m[(n-1)l+1, nl]\}$
   where $m[x, y]$ stands for bits of $m$ from $x$-th bit to $y$-th bit.
5: return the set of all users' data numbers $D$;
```

is n-source anonymous in the semi-honest model.

Proof: Let $M = \{0, 1\}^l$. Given a set of semi-honest users, I={1,…,n}, a sequence of inputs $D = (d^1, \ldots, d^n) \in M^l$, the view of the aggregator in the anonymous data aggregation protocol is a sequence of random variables:

$$E = (\overline{e_1^1}, \ldots, \overline{e_1^n}, \overline{e_2^1}, \ldots, \overline{e_2^n}, \ldots, \overline{e_n^1}, \ldots, \overline{e_n^n}),$$

where $e_j^i \in \{0, 1\}$ $(i, j \in I)$ is the ciphertext sent by user i in the jth round.

For $\forall (i, j) \in I \times I$ where $i < j$, suppose user i and j switch their data, the sequence of inputs changes to

$$D' = (d^1, \ldots, d^{i-1}, d^j, d^{i+1}, \ldots, d^{j-1}, d^i, d^{j+1}, \ldots, d^n),$$

and the view of the aggregator in the anonymous data aggregation protocol becomes

$$\begin{aligned} E'(i, j) = (&\overline{e_1^1}, \ldots, \overline{e_1^{i-1}}, \overline{e_1^j}, \overline{e_1^{i+1}} \ldots, \overline{e_1^{j-1}}, \overline{e_1^i}, \overline{e_1^{j+1}} \ldots, \overline{e_1^n}, \\ &\overline{e_2^1}, \ldots, \overline{e_2^{i-1}}, \overline{e_2^j}, \overline{e_2^{i+1}} \ldots, \overline{e_2^{j-1}}, \overline{e_2^i}, \overline{e_2^{j+1}} \ldots, \overline{e_2^n}, \\ &\ldots, \\ &\overline{e_n^1}, \ldots, \overline{e_n^{i-1}}, \overline{e_n^j}, \overline{e_n^{i+1}} \ldots, \overline{e_n^{j-1}}, \overline{e_n^i}, \overline{e_n^{j+1}} \ldots, \overline{e_n^n}). \end{aligned}$$

According to Definition 1, to prove $\mathcal{P}$ is n-source anonymous, it is equivalent to prove

$$E \overset{c}{\equiv} E'(i, j)$$

holds for $\forall (i, j) \in I \times I$, and $\forall D \in M^l$.

To show this, we construct a simulator $\mathcal{S}$ that takes $\{\vec{d^i}\}_{i=1,\ldots,n}$ as inputs and outputs a view that is computationally indistinguishable to both E and $E'(i, j)$. Let

$$D_a = (d_a^1, d_a^2, \ldots, d_a^n), \text{where for each } id_a^i = \overline{e_1^i} \oplus \ldots \oplus \overline{e_n^i}.$$

In particular, given the received $(d^1, \ldots, d^n)$ as input, $\mathcal{S}$ generates a view E'' as follows.

First, $\mathcal{S}$ generates a random permutation function $\pi : [1, n] \to [1, n]$, and runs the same aggregation protocol with user i's data being $d_a^{\pi(i)}$. Let E'' be the view of the aggregator at this time. Let

$$D'_a = (d_a^{\pi(1)}, d_a^{\pi(2)}, \ldots, d_a^{\pi(n)})$$

We know $E \overset{c}{\equiv} E''$, otherwise we can distinguish D with its pseudo-random permutation D'_a in polynomial time which is impossible. Also we know $E'(i, j) \overset{c}{\equiv} E''$, otherwise we can distinguish D' with its pseudo-random permutation D'_a in polynomial time which is impossible also.

Therefore, we have

$$E \overset{c}{\equiv} E'(i, j)$$

□

5 Protecting Privacy When User Number Is Large

In some scenarios, the total number of the participants could be very large, e.g. an epidemic monitoring application collecting body temperature of citizens in a big city. Allowing all users to execute our anonymous data aggregation protocol together may put a heavy burden to the aggregator, as the complexity of our protocol is $O(n^2)$ for the aggregator. In this section, we study how to optimize the efficiency of the secure data aggregation process in a scenario where the total number of users is large. In particular, we let the aggregator divide users into groups according to users' privacy requirements before it runs our anonymous data aggregation protocol within each user group. With the optimal grouping solution, the efficiency of the entire data aggregation process can be optimized.

5.1 Dividing Users Into Groups

When n is large, the bottleneck of the data aggregation process's efficiency is on the aggregator's side. Since the total number of bit strings received by the aggregator directly affects the time taken by the aggregator in receiving and decrypting data, here we optimize the aggregation efficiency by minimizing the total number of bit strings received by the aggregator.

Notice that all n users achieve a privacy level of n when they form one big group to run this anonymous data aggregation protocol together. This is achieved by allowing all $n-1$ users to send a "dummy" bit string every time a user sends its bit string. To receive n "true" bit strings, the aggregator needs to receive $n(n-1)$ dummy bit strings also. If we divide n users into several groups, and allow users inside each group to execute our protocol together, then every time a users sends its bit string, only the other users in the same group need to send a dummy string. This can lead to the aggregator receiving significant fewer bit strings.

When applying our protocol on users inside a group, every user in the group has the same privacy level, which is equal to the size of the group. In practice, users may have different privacy level requirements. Making all groups' sizes equal to the lowest privacy requirement apparently will not satisfy users who have higher privacy requirements. Nevertheless, making all groups' sizes equal to the highest privacy level requirements of the users could lead to a relatively low efficiency. In the next section, we show how to find an optimal grouping which satisfies all users' privacy requirements and meanwhile optimize the efficiency of the data aggregation process.

5.2 Optimal Grouping

Assume users have different minimal privacy level requirements. Denote by a_i user i's minimal privacy level requirement ($i \in [1, n]$). This means user i can only be put into a group of a size that is no less than a_i. Since the best privacy level in a n-user scenario is n, we only consider the case that all a_i is less than or equal to n. Note that this means that there is at least one grouping solution (e.g., putting all users in one group) that satisfies all users' privacy requirements. Also, assume users are sorted in a non-decreasing order according to their minimal privacy level requirements. Thus, we have

$$1 \le a_1 \le a_2 \le \ldots \le a_n \le n. \tag{2}$$

We are interested in finding an optimal grouping which satisfies all users' privacy level requirements and minimizes the total bit strings sent to the aggregator at the same time.

Denote by $\mathcal{G} = \{G_j\}_{j=1,\ldots,|\mathcal{G}|}$ a grouping solution where G_j is a set that contains the index of all users in group G_j. We refer to a grouping solution which satisfies all users' privacy level requirement as a feasible grouping solution, and the total number of bit strings sent to the aggregator when this solution is adopted as the grouping solution's cost C. It is straightforward to see that our problem is equivalent to finding a grouping solution which is of minimal cost to all feasible grouping solutions.

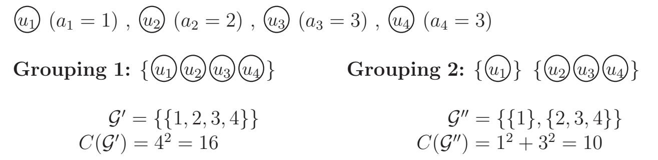
Specifically, if for any user i and any group G_j in $\mathcal{G}$,

$$i \in G_j \Rightarrow |G_j| \ge a_i, \tag{3}$$

we say $\mathcal{G}$ is feasible. And it is easy to verify that $\mathcal{G}$'s cost

$$C(\mathcal{G}) = \Sigma_{j=1,\ldots,|\mathcal{G}|}(|G_j|)^2. \tag{4}$$

Figure 2 shows a 4-user example which has only two feasible grouping solutions.



u_1 $(a_1 = 1)$, u_2 $(a_2 = 2)$, u_3 $(a_3 = 3)$, u_4 $(a_4 = 3)$
Grouping 1: $\{u_1 u_2 u_3 u_4\}$
Grouping 2: $\{u_1\}$ $\{u_2 u_3 u_4\}$
$\mathcal{G}' = \{\{1,2,3,4\}\}$
$C(\mathcal{G}') = 4^2 = 16$
$\mathcal{G}'' = \{\{1\},\{2,3,4\}\}$
$C(\mathcal{G}'') = 1^2 + 3^2 = 10$

Figure 2. Two feasible grouping solutions and their corresponding costs in an example of 4 users.

If a group contains only users with consecutive indexes, we say this group is *consecutive*. For example, $G = \{1, 2, 3\}$ is consecutive and $G' = \{1, 3\}$ is not. If all groups inside a grouping solution are consecutive, we say this solution is a *consecutive grouping solution*. The following proposition allows us to narrow down our searching space from all feasible grouping solutions to all feasible consecutive grouping solutions.

Proposition 3: For every feasible grouping solution $\mathcal{G}$, there exists a feasible consecutive grouping solution $\mathcal{G}^c$ which has the same cost.

Proof: Without loss of generality let $\mathcal{G} = \{G_1, G_2, \ldots, G_{|\mathcal{G}|}\}$, where $|G_1| \le |G_2| \le \ldots \le |G_{|\mathcal{P}|}|$. We can prove the proposition by constructing a valid $\mathcal{G}^c$ as follows.

1) Generate $|\mathcal{G}|$ "empty" groups $G_1^c, G_2^c, \ldots, G_{|\mathcal{G}|}^c$. The size of group G_j^c is the same as size of G_j ($j = 1, \ldots, |\mathcal{G}|$).
2) Put user $1, 2, \ldots, n$ one by one into group G_1^c, and then into G_2^c after G_1^c is full, ..., and finally into $G_{|\mathcal{G}|}^c$.
3) Let $\mathcal{G}^c = \{G_1^c, G_2^c, \ldots, G_{|\mathcal{G}|}^c\}$.

It is straightforward to see that $\mathcal{G}^c$ is consecutive and $\mathcal{G}^c$'s cost is equal to $\mathcal{G}$'s cost. We only need to prove that $\mathcal{G}^c$ is feasible or the size of every group in $\mathcal{G}^c$ is no less than the highest privacy levels of all users inside this group.

Consider an arbitrary group G_j^c ($j = 1, \ldots, |\mathcal{G}|$). According to inequality (2), the user who has the highest privacy level requirement inside G_j^c is the last user put into this group. Denote by i this user's index ($i = |G_1| + \ldots + |G_j|$). Thus, proving $|G_j^c| \ge a_i$ would validate G_j^c's feasibility.

Let G_k be the group that contains user i in solution $\mathcal{G}$. Since $\mathcal{G}$ is feasible, we know $|G_k| \ge a_i$ according to condition (3).

- If $k = j$, we know $|G_j^c| = |G_j| = |G_k| \ge a_i$.
- If $k < j$, we know $|G_j^c| = |G_j| \ge |G_k| \ge a_i$.
- If $k > j$, this means in solution $\mathcal{G}$ user i is not in any group among $G_1, \ldots, G_j$. Now assume $G_1, \ldots, G_j$ only contain users whose index is smaller than i. There are $i - 1$ users whose index is smaller than i.

However, $G_1, \ldots, G_j$ contains $|G_1| + \ldots + |G_j| = i$ users. Therefore we know that the assumption cannot be true and that there is at least one user i' whose index i' is greater than i contained in one group among $G_1, \ldots, G_j$. Since G_j has the largest size among $G_1, \ldots, G_j$, we know $|G_j^c| = |G_j| \geq a_{i'} \geq a_i$.

Therefore, $\mathcal{G}^c = \{G_j^c\}$ is feasible. □

By Proposition 3, we can find an optimal grouping solution $\mathcal{G}^\star$ by searching the solution that has the minimal cost in all feasible consecutive grouping solutions only. In the rest of this section, we discuss only consecutive grouping solutions. For ease of presentation, we omit "consecutive" and refer to them simply as grouping solutions.

To find a feasible grouping solution that groups users $1, \ldots, n$ with minimal cost, we use the dynamic programming method. The main idea is to divide the original problem of grouping users $1, \ldots, n$ into smaller sub-problems. The original problem's solution can be found based on the sub-problems' solutions.

Specifically, the sub-problem is to find a feasible grouping solution $\mathcal{G}^\star(x)$ that groups user $1, \ldots, x$ ($x = 1, 2, \ldots, n$) with minimal cost. ($\mathcal{G}^\star = \mathcal{G}^\star(n)$.)

Define two functions:

$$f(x) : [0, n] \to \mathbb{R} \tag{5}$$

and

$$\begin{cases} g(x) : [1, n] \to [1, n] \\ g(x) \leq x \end{cases} . \tag{6}$$

Function $f(x)$ returns the $\mathcal{G}^\star(x)$'s cost,[2] and $g(x)$ returns the smallest index contained in the group that has user x inside in grouping solution $\mathcal{G}^\star(x)$.

Consider the sub-problem of finding $\mathcal{G}^\star(x)$ that groups user 1 to x. Recall $\mathcal{G}^\star(x)$ groups users $g(x), g(x)+1, \ldots, x$ in one group. $\mathcal{G}^\star(x)$'s cost is equal to $(x - g(x) + 1)^2$ adds the quadratic sum of the remaining groups' sizes according to equation (4). To minimize $\mathcal{G}^\star(x)$'s cost, the quadratic sum of the remaining groups' sizes has to be minimal. Notice that the remaining groups form a grouping solution that groups users $1, \ldots, g(x) - 1$, the quadratic sum of these groups' sizes is the cost of the grouping solution. Since the minimal cost of a grouping solution that groups users $1, \ldots, g(x)-1$ is $f(g(x)-1)$, the cost of $\mathcal{G}^\star(g(x) - 1)$, we know $\mathcal{G}^\star(x)$ can be constructed using its sub-problem's solution $\mathcal{G}^\star(g(x) - 1)$ as follows.

$$\mathcal{G}^\star(x) = \mathcal{G}^\star(g(x) - 1) \bigcup \{\{g(x), \ldots, x\}\}, \tag{7}$$

and

$$f(x) = f(g(x) - 1) + (x - g(x) + 1)^2. \tag{8}$$

Based on these two equations, the Dynamic Programming (DP) solves the problem in a bottom-up way. Specifically, the DP solves $(\mathcal{G}^\star(1), g(1), f(1))$ first, and then $(\mathcal{G}^\star(2), g(2), f(2))$, and then ..., and finally $(\mathcal{G}^\star(n), g(n), f(n))$. When $(\mathcal{G}^\star(1), g(1), f(1))$ to $(\mathcal{G}^\star(x-1), g(x-1), f(x-1))$ are known, to determine $(\mathcal{G}^\star(x), g(x), f(x))$ we only need to know $g(x)$'s value. ($\mathcal{G}^\star(x)$ and $f(x)$ can be computed using equations (7) and (8).)

In particular, $g(x)$'s value can be determined by traversing all possible values and finding the value that minimizes the corresponding $f(x)$ as follows. Let $i = g(x)$. Since $\mathcal{G}^\star(x)$ puts user $i, i+1, \ldots, x$ in one group and $\mathcal{G}^\star(x)$ is feasible, the group's size must be no less than the maximum privacy level requirement of users inside this group, i.e. $i \leq x - a_x + 1$. By traversing i in $[1, x - a_x + 1]$, we can get the value of $g(x)$ as follows.[3]

$$g(x) = \arg \min_{i \in [1, x - a_x + 1]} f(i-1) + (x - i + 1)^2. \tag{9}$$

Table 2 shows the corresponding $(\mathcal{G}^\star(x), g(x), f(x))$ values in the Figure 2's example. We formally present our optimal grouping algorithm as Algorithm 2.

Table 2
A walk-through example of using Dynamic Programming to find the optimal grouping in Figure 2's example

	g(x)	f(x)	$\mathcal{G}^\star(x)$
x=0	0	∞	$\emptyset$
x=1	1	1	{{1}}
x=2	1	4	{{1,2}}
x=3	1	9	{{1,2,3}}
x=4	2	f(2-1)+$(4-2+1)^2$=10	$\mathcal{G}^\star(1) \bigcup \{\{2,3,4\}\} = \{\{1\},\{2,3,4\}\}$

By our earlier derivations, we can easily get the following result.

Theorem 4: The optimal grouping algorithm returns an optimal grouping for all users.

Complexity Analysis. Our optimal grouping algorithm needs to sort n users according to their minimal privacy requirements first, and then traverse all possible values of j to compute $(g(1), f(1)), \ldots, (g(n), f(n))$. The sorting can be done using an $O(n \log n)$ sorting algorithm such as merge sort. To compute $(g(i), f(i))$, at most i possible values need to be traversed. Thus the complexity for computing $(g(n), f(n))$ is $O(1 + 2 + \ldots + n)$ which is $O(n^2)$. Overall, the computation complexity is $O(n^2)$.

One might wonder why we want to spend $O(n^2)$ grouping users just in order to expedite another algorithm (our anonymous data aggregation protocol) with computation complexity $O(n^2)$ (per time period). Note that this is worthwhile, because: 1) The grouping algorithm is much more efficient compared with the aggregation protocol, despite both of them having a computation complexity of $O(n^2)$. This is due to the data aggregation process involving encryptions and decryptions which are relatively time-consuming, while the grouping algorithm involves only simple arithmetic computations. (For example, according to our evaluation results (see

2. If there is no feasible solution to group user 1 to x, $f(x)$ returns infinity. When $x = 0$, there is no user to group, thus let the cost f(0) = 0 and $\mathcal{G}^\star(0) = \emptyset$.

3. If $x - a_x + 1 < 1$, user x's minimal privacy level is too large and there is no feasible grouping solution. In this case, we let $f(x) = \infty$ and $g(x) = 0$.

Algorithm 2 Optimal Packing Algorithm

```
Input:
    a group of $n$ users (already sorted in an increasing order according to their privacy requirements;
    user $i$ ($i = 1, \dots, n$)'s minimal privacy level requirement is $a_i$.
Output:
    an optimal grouping $\mathcal{G}^\star$;
 1: sets $f(0)$ to 0.
 2: sets $f(i)$ ($i = 1, \dots, n$) to $\infty$.
 3: sets $g(i)$ ($i = 1, \dots, n$) to 0.
 4: for $i = 1, \dots, n$ do
 5:     if $a_i > i$ then
 6:         continue;
 7:     end if
 8:     for $j = 1, \dots, i + 1 - a_i$ do
 9:         if $f(i) > f(j-1) + (i-j+1)^2$ then
10:             $f(i) = f(j-1) + (i-j+1)^2$.
11:             $g(i) = j$.
12:         end if
13:     end for
14: end for
15: set $k$ to $n$.
16: set $\mathcal{G}^\star$ to the empty set.
17: repeat
18:     $\mathcal{G}^\star \leftarrow \mathcal{G}^\star \bigcup \{\{g(k), g(k)+1, \dots, k\}\}$
19:     $k \leftarrow g(k) - 1$
20: until $g(k) = 1$
21: return $\mathcal{G}^\star$;
```

Tables 5,6 and 7), the efficiency of the grouping algorithm is approximately 100 times of the efficiency of the aggregation process when the sensing data is only 10-bits long.) 2) In a static environment, we only need to perform grouping once, after which we can run the anonymous data aggregation protocol many times. In addition, a mobile sensing application for aggregating time-series data generally needs to run the aggregation protocol for many time periods.

6 Performance Evaluation

In this section, we perform theoretical analysis and experiments to evaluate the performance of our anonymous data aggregation protocol and the optimal grouping algorithm.

We implement our algorithms using Microsoft Visual Studio 2012 and the Crypto++ library [13]. As suggested in [6], we use HMAC<SHA512> as our pseudo-random function family and adopt the length-matching hash functions construction in [6]. In particular, HMAC<SHA512> generates a 512-bit output. In case we need a shorter output of length l, we truncate the output into short bit strings of length l and then use the exclusive-OR of all these strings as the final output. In the case that l is greater than 512, we first break the input message into several 512-bit strings and one string of length l' ($l' < 512$), then generate several 512-bit output strings and one l'-bit output string, and finally use the concatenation of these output strings as the final output. All experiments are performed on a laptop running the 64-bit Windows 7 Professional operating system with Intel Core i7 3520M CPU and 8GB memory.

Users' data is uniformly sampled from their data space $[0, 2^l - 1]$. Users' minimal privacy level requirements are randomly chosen from $[1, q]$ according to a modified normal distribution[4] with a mean of $0.1q$ and a variance of $0.05q$, where q denotes the maximum privacy level requirement. Results of experiments on our anonymous data aggregation protocol are averaged by 10 runs, and results of experiments on our optimal grouping algorithm are averaged by 1000 runs.

6.1 Efficiency Comparisons with Previous Content-Hiding-Based Protocols

Since our protocol requires users to send fake data to make real data anonymous, our protocol's communication cost is correspondingly increased. In this section, we perform theoretical analysis to compare the efficiency of our protocol and other previous content-hiding-based protocols in terms of the amount of data transmitted.

In particular, we compare the amounts of data transmitted in our protocol and two recent content-hiding-based protocols [26, 39] when they are used to securely compute two different aggregation functions sum and Max/Min. Before we present our comparison results, we want to make a few clarifications. First of all, as we have mentioned, the two protocols proposed in [26, 39] are designed for a different purpose (i.e. to compute the sum securely) compared with our protocol. Therefore, we perform the efficiency comparisons only to provide a straightforward idea about these protocols' performance of conducting different computations, and the comparison results do not suggest one protocol is actually better than another. As a matter of fact, these protocols and our protocol are complementary to each other. Furthermore, to be fair, we note that the protocol proposed by Li et al. in [26] enjoys excellent enhanced security to deal with users' collusion and dynamic user joins/leaves compared with ours.

Table 3
Transmission Overheads Comparison

	to compute sum	to compute max/min
[39]	$nn'l + nl$	$nl^2 + (nn' + n)\lceil\log(n+1)\rceil$
[26]	nl	$n2^l\lceil\log(n+1)\rceil$
Our protocol	n^2l	n^2l

In [39], a slicing technique is used to protect each node's data. When computing sum, each node slices its data into $n'+1$ random shares ($1 \le n' \le n-1$), keeps one

4. Since users' privacy level requirements are integers in $[1, q]$, we adopt a modified normal distribution which uses a regular normal distribution to generate samples, and then discretizes samples to their nearest integers in [1,q].

share and sends other n' shares to n' randomly selected nodes (called "cover nodes"). Every node sums its own share and the shares of other nodes which it receives, and sends this sum to the aggregator. By summing the data received from all nodes, the aggregator knows the sum. The amount of data transmitted equals $nn'l + nl$ (To be fair, here we neglect the traffic generated in the selection of cover nodes). To compute Max/Min, the aggregator runs a binary with the help of all nodes. It needs l queries and in each query the aggregator sends all nodes an l-bit data and counts the number of positive answers with the same slicing technique. If each user's answer is sliced into $n+1$ slices and the answer is $log(\lceil n+1 \rceil)$ bit, The amount of data transmitted here equals $nl^2 + (nn'+n)\lceil \log(n+1) \rceil$.

In [26], zero-sum noise has been added to protect users' data. To achieve privacy-preserving sum aggregation, each user only needs to send the encrypted data to the aggregator in a time period. Thus, the amount of data transmitted in a period is nl. To achieve privacy-preserving max/min, each user needs to generates a new data of length $2^l \lceil \log(n+1) \rceil$. Thus, the amount of data transmitted in a period is $n2^l \lceil \log(n+1) \rceil$.

Table 3 shows comparison results. We can see that as the complexity of the aggregation function increases, the communication overhead increases in both [39] and [26] while the overhead remains the same in our protocol. In addition, although our protocol has a $O(n^2)$ communication overhead, the overhead could still be less compared with existing privacy preserving data aggregation protocols in some cases, for example, l is very large in [39] or n' is chosen to be $n-1$ in [26].

6.2 Efficiency of the Anonymous Data Aggregation Protocol

The efficiency of our protocol is closely related to the total user number n and the data length l. Therefore we test our protocol's efficiency under different n and l values respectively.

Table 4
Efficiency of our protocol when l changes (n = 1000)

	l=5	l=15	l=20	l=30	l=40	l=50
Enc.	$7.3ms$	$7.3ms$	$7.3ms$	$7.3ms$	$7.3ms$	$7.3ms$
Dec.	$197.3ms$	$198.0ms$	$198.1ms$	$199.3ms$	$200.8ms$	$202.2ms$
	l=100	l=200	l=500	l=1000	l=2000	l=5000
Enc.	$7.3ms$	$7.4ms$	$7.5ms$	$15.1ms$	$35.4ms$	$115.4ms$
Dec.	$209.3ms$	$222.0ms$	$275.0ms$	$348.7ms$	$486.7ms$	$897.4ms$

First, we fix the total user number to 1000 and test our protocol under twelve different data lengths from $l = 5$ to $l = 5000$. The results are shown in Table 4. We can see that the encryption/decryption time increases slowly as l grows. Even when l reaches 5000 (which means a quite large data space $[0, 2^{5000} - 1]$ for general mobile phone sensing applications), the encryption time is only 115.4 milliseconds and the decryption time is only 897.4 milliseconds.

Table 5
Efficiency of our protocol when n changes (l=10)

	n=100	n=200	n=500	n=1000	n=2000	n=5000
Enc.	$0.8ms$	$1.5ms$	$3.7ms$	$7.3ms$	$14.7ms$	$36.6ms$
Dec.	$2.1ms$	$7.9ms$	$49.6ms$	$197.5ms$	$792.1ms$	$4.9s$

Table 6
Efficiency of our protocol when n changes (l=100)

	n=100	n=200	n=500	n=1000	n=2000	n=5000
Enc.	$0.7ms$	$1.5ms$	$3.7ms$	$7.3ms$	$14.7ms$	$36.5ms$
Dec.	$2.1ms$	$8.6ms$	$52.4ms$	$209.3ms$	$836.4ms$	$5.2s$

Also, we fix l to 10 and 100, and test our protocol under different values of total user number n: 100, 200, 500, 1000, 2000 and 5000. The results are shown in Tables 5 and 6. We can see that the encryption time increases slowly as n grows. Due to the $O(n^2)$ complexity, the decryption time's growth is a little faster compared with the encryption time, but the efficiency is still high. When n reaches 5000, the corresponding decryption time is only 4.9 seconds ($l = 10$) and 5.2 seconds ($l = 100$).

6.3 Efficiency and Effectiveness of the Optimal Grouping Algorithm

The efficiency of our optimal grouping algorithm is closely related to the total user number n. Therefore we test our algorithm's efficiency under different values of n: 100, 200, 500, 1000, 2000, 5000, 10000 and 50000. We set users' maximum privacy level q to a half of n in this test.

Table 7 shows the results. We can see that the efficiency of our optimal grouping algorithm is very high. When n equals 10000, the time it takes our algorithm to find an optimal grouping is only 143.7 milliseconds. When n reaches 50000, the time is only 2.98 seconds.

Table 7
Efficiency of the optimal grouping algorithm (q = $n/2$)

	n=100	n=200	n=500	n=1000	n=2000	n=5000	n=10000	n=50000
time	$15.9\mu s$	$58.0\mu s$	$381.2\mu s$	$1.5ms$	$6.0ms$	$36.9ms$	$143.7ms$	$2.98s$

To evaluate the effectiveness of the optimal grouping algorithm, we compare the total amount of data received by the aggregator in two cases: 1)letting all users apply the anonymous data aggregation protocol according to a naive grouping. The naive grouping divides users into groups with the same size that equals the highest privacy level requirement of all users. 2)letting users apply the protocol according to the optimal grouping computed by our grouping algorithm. In particular, we set the total user number n to 10000 and test the two cases under different values of q ranging from 1000 to 10000. Note that there might be some "remnant users" when the highest privacy level requirement of users cannot divide n in the first case, we assume the naive grouping

Table 8
Total amount of data received by the aggregator with/without applying the optimal grouping algorithm (n = 10000)

q	With optimal grouping	With naive grouping	Percentage
1000	$1,055,742$	$2,980,569$	35.42%
2000	$2,191,421$	$6,105,841$	35.89%
3000	$3,408,804$	$9,374,218$	36.36%
4000	$4,720,393$	$12,853,078$	36.73%
5000	$6,120,561$	$16,398,246$	37.32%
6000	$7,611,881$	$20,468,094$	37.19%
7000	$9,197,760$	$24,337,719$	37.79%
8000	$10,892,837$	$26,494,114$	41.11%
9000	$12,693,283$	$34,446,455$	36.85%
10000	$14,538,832$	$34,543,026$	42.09%

generates one big group by adding these users into a normal group.

Table 7 shows the results. We can see that the effectiveness of our optimal grouping algorithm is very excellent. When q equals 1000, the total amount of data received by the aggregator after grouping using our algorithm is only 35.42% of the total amount of data received by the aggregator with a naive grouping. When q equals 10000, the percentage is only 42.09%. Applying our grouping algorithm greatly improves the entire data aggregation process's efficiency.

7 Discussion

In this section, we discuss several interesting topics and possible extensions of our protocols.

7.1 Dealing with the Dynamic Change of Users

In this section, we extend our anonymous data aggregation protocol to efficiently handle users' dynamic change.

Intuitively, when a participating user leaves, or a new user joins in the aggregation, our protocol can be re-established by letting the trusted authority (TA) re-issue new keys to all current participants and allow them to regenerate new random bit strings to encrypt their messages with. However, applying the above method in highly dynamic scenarios with a very large number of users may incur very high computation and communication overheads for all remaining users. To deal with this issue, we propose alternative efficient approaches which significantly reduce the overhead introduced due to the dynamic change of users.

7.1.1 When a User Leaves

To efficiently handle users which leave, we propose to let the TA help the aggregator. Specifically, when a participating user i leaves the system, all remaining users continue to run the original protocol without any change. After the aggregator notices user i's leaving, it notifies the TA of the leaving user, and asks for its help.

Recall that the TA knows every user's keys, including user i's two keys s_a^i and s_b^i. To help the aggregator, the TA simulates user i to compute its random bits k_j^i $(j = 1, \ldots, n)$ as specified in (1), and uses them to encrypt n dummy strings of $\{0\}^l$. The TA sends these encrypted bit strings to the aggregator. With the bit strings generated by the TA and the remaining users, the aggregator can still compute the aggregation of the remaining users' data (and a dummy data of $\{0\}^l$).

We point out that this approach does not require the remaining users to make any changes when any user leaves. In addition, they do not need to keep track of the current number of users in the aggregation (except the original total number n). Therefore, the computation and communication overhead due to the users leave is significantly reduced compared with the intuitive method.

7.1.2 When a New User Joins

When a new user joins the aggregation, we propose an efficient approach that requires only two of the original participants to update their keys and their sequence numbers, thus the additional overhead due to the new users' joining is also reduced significantly compared with the intuitive method.

Specifically, recall that each original user owns two of the n keys $S_0, \ldots, S_{n-1} \in \{0,1\}^l$, the TA generates one new random key $S_n \in \{0,1\}^l$ when a new user joins the system. Also, the TA randomly selects two original users i and i'. Let (s_a^i, s_b^i) and $(s_a^{i'}, s_b^{i'})$ be users i's and i''s keys respectively. Then, the TA sends S_n to user i and user i', and sends s_a^i and $s_a^{i'}$ to the new user. Now, to participate in the aggregation protocol, user i and user i' would use (S_n, s_b^i) and $(S_n, s_b^{i'})$ as their new keys respectively, and the new user would use $(s_a^i, s_a^{i'})$ as its own keys. It is easy to verify that the bitwise XOR homomorphism still holds, thus the aggregation still holds correctly.

In addition, the three users would also need to update their sequence numbers to protect the new user's privacy. To do so, the TA randomly shuffles their original sequence numbers (letting the newly joined users sequence number be n+1), and notifies them of the shuffled ones.

With the new keys and sequence numbers, all users could continue to jointly apply our protocol to let the aggregator compute the data aggregation securely.

7.2 Grouping by Users' Views

We have proposed a grouping method from the aggregator's point of view. One might also be interested in designing the grouping method from each user's point of view.

Let $\mathcal{G} = \{G_j\}_{j=1,\ldots,|\mathcal{G}|}$ denote the grouping solution, where G_j is the set that contains the index of all users in the j-th group. For each i, let $g(i) \in [1, |\mathcal{G}|]$ denote the group index of user i in the grouping solution. According to the complexity analysis of our aggregation protocol, user i's computation overhead and communication overhead are determined by its group size $|G_{g(i)}|$, and are $O(|G_{g(i)}|)$.

As user groups may have different sizes, their computation and communication overhead may also be different. Therefore, we use users' average computation overhead and communication overhead as our optimizing objectives. Specifically, both overheads equal:

$$\begin{aligned} & \sum_{i=1,\ldots,n} |G_{g(i)}|/n & (10)\\ = & \sum_{j=1,\ldots,|\mathcal{G}|} (\sum_{i\in G_j} |G_j|)/n & (11)\\ = & \sum_{j=1,\ldots,|\mathcal{G}|} (|G_j|)^2/n. & (12)\end{aligned}$$

Given a fixed total number of users, to minimize users' average computation overhead and communication overhead specified in the above equations, is equivalent to minimize the aggregator's total computation overhead which is specified in equation (4). Therefore, our optimal packing algorithm also optimizes users' average overheads.

7.3 Regrouping

When users' required privacy level changes, or a new user joins the system, or an existing user leaves, the aggregator may need to re-run our optimal packing algorithm to compute a new optimal grouping solution. Here, we provide a possible alternative method to further expedite the regrouping process.

Instead of re-running the optimal packing algorithm every time a change occurs, we propose to use a composite updating policy to adjust the existing grouping solution. Specifically, we let the aggregator perform two kinds of re-grouping operations: the "*complete re-grouping*" which is required in order to re-run the optimal packing algorithm, and the "*quick re-grouping*" which adjusts the previous grouping solution to guarantee that users' privacy requirements are not violated.

In cases where a user decreases its required privacy level, quick re-grouping makes no change to the existing grouping solution; In cases where a user increases its privacy level from a to a', quick re-grouping searches for the smallest existing group, other than the this user's group, with the size that is no less than $a' - a$, and merges this group and the user's group together in the new grouping solution.

In cases where a new user whose required privacy level equals a joins the system, quick re-grouping adds this user to the smallest existing group with the size that is no less than $a - 1$ if such group exists. If there is no such a group, quick re-grouping continuously merges the largest group with small groups, starting with the smallest one, until the merged group's size is no less than $a - 1$, and then adds the user into this group.

In cases that an existing user leaves, quick re-grouping simply merges this user's group with the smallest if they are not of the same group. Otherwise, quick re-grouping simply merges this user's group with the second smallest group.

It is easy to see that the new grouping solution generated by quick re-grouping satisfies all users' privacy requirements. In addition, by sacrificing some optimality, quick re-grouping is much more efficient compared with complete-regrouping. To expedite the regrouping process, the aggregator can run complete re-grouping at the start, then perform quick-regrouping operations to handle every change. Meanwhile, to maintain a good optimality at all time, once a certain number of quick re-groupings are performed, the aggregator performs a complete re-grouping. Here, the number of times can be determined empirically based on the total number of current users and the magnitude of the changes.

7.4 Sensing Data Authenticity

Other than the problems we have studied, another very interesting problem is how to guarantee the data collected from the users is trustworthy, or authentic.

To guarantee the authenticity of sensing data generally requires that the sensing data is collected in the right way, at the right place, and that the data has not been tampered with. This could be a highly challenging task depending on the specific requirements of the sensing job. Recently, there have been a few works which propose different approaches to secure the authenticity of the sensing data in different sensing applications. Some of them are based on non-technical economic and/or legal frameworks. For example, freelance photographers could receive payments by sending their photos which conform to the guidelines on integrity [1] put in place by the New York Times newspaper. Some of them are based on cryptographic and/or trusted hardware techniques. For example, in [16], authors use Trusted Platform Module (TPM) hardware to generate cryptographic signatures on the sensing data to guarantee that the data is not tampered with. Some of them are based on reputation systems. For example, in [3], authors build a trust and reputation system in a wireless sensor network to differentiate trustworthy nodes from malicious nodes.

In our humble opinion, solving the authenticity issue in mobile sensing applications requires a hybrid framework that is based on all techniques mentioned above. Firstly, since mobile phone users generally incur costs when they do perform sensing tasks, proper economic incentives can be introduced to compensate their costs, and incentivize non-malicious users to contribute correct sensing data. Secondly, cryptographic and trusted hardware techniques can be introduced to detect false data which is sent by a malicious user. For example, trusted hardwares (e.g. the TPM) can be used to monitor the software configurations or status by generating cryptographic signatures on the hash values of the configurations/status. By verifying the signatures and examining the hash values, the data aggregator knows whether the data is collected or processed following the correct procedure. Finally, a reputation system enables analysis based on empirical data, thus can help the aggregator to avoid malicious users in future data aggregations.

8 Conclusion

In this paper, we first propose an anonymous data aggregation protocol that allows an untrusted aggregator to collect participants' data without being able to identify the source of any particular piece of data in a mobile sensing scenario. To improve the efficiency, especially in cases where the total number of participants is very large, we propose to divide users into several groups and let users inside one group execute the anonymous data aggregation protocol together. We study how to find an optimal grouping which minimizes the total amount of data sent to the aggregator and give an optimal grouping algorithm.

Different from exist privacy-preserving mobile sensing works which only support secure computation of single simple aggregation function, our protocols allow the aggregator to efficiently compute arbitrary/complicated aggregation functions and protect users' privacy at the same time.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable time and effort in reviewing this paper. Their insightful suggestions helped in improving the quality of this paper significantly.

References

[1] New york times: Guidelines on integrity.

[2] Abdel-shakour Abuzneid, Tarek Sobh, and Miad Faezipour. An enhanced communication protocol for anonymity and location privacy in wsn. In *Proceedings of the IEEE Wireless Communications and Networking Conference, New Orleans, LA, USA, country*, volume 912, 2015.

[3] Efthimia Aivaloglou and Stefanos Gritzalis. Hybrid trust and reputation management for sensor networks. *Wireless Networks*, 16(5):1493–1510, 2010.

[4] Kemal Bicakci, Hakan Gultekin, Bulent Tavli, and Ibrahim Ethem Bagci. Maximizing lifetime of event-unobservable wireless sensor networks. *Computer Standards & Interfaces*, 33(4):401–410, 2011.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 325–341, 2005.

[6] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, pages 1–36, 2009.

[7] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[8] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptology*, 1(1):65–75, 1988.

[9] Marco Conti, Johnny Willemsen, and Bruno Crispo. Providing source location privacy in wireless sensor networks: a survey. *Communications Surveys & Tutorials, IEEE*, 15(3):1238–1280, 2013.

[10] Mauro Conti, Lei Zhang, Sankardas Roy, Roberto Di Pietro, Sushil Jajodia, and Luigi Vincenzo Mancini. Privacy-preserving robust data aggregation in wireless sensor networks. *Security and Communication Networks*, 2(2):195–213, 2009.

[11] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *ACM MobiSys*, 2008.

[12] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. Mobile phone-based pervasive fall detection. *Personal and Ubiquitous Computing*, 14(7):633–643, 2010.

[13] W. Dai. Crypto++ library. 2010.

[14] J Dankers, T Garefalakis, R Schaffelhofer, and T Wright. Public key infrastructure in mobile systems. *Electronics & Communication engineering journal*, 14(5):180–190, 2002.

[15] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[16] Peter Gilbert, Landon P Cox, Jaeyeon Jung, and David Wetherall. Toward trustworthy mobile sensing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, pages 31–36. ACM, 2010.

[17] Oded Goldreich. *Foundations of cryptography: Basic applications*, volume 2. Cambridge Univ Press, 2004.

[18] Philippe Golle and Ari Juels. Dining cryptographers revisited. In *EUROCRYPT*, pages 456–473, 2004.

[19] Michael M Groat, Wenbo He, and Stephanie Forrest. Kipda: k-indistinguishable privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2024–2032. IEEE, 2011.

[20] Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. Pda: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2045–2053. IEEE, 2007.

[21] Jun Huang, Meisong Sun, Shitong Zhu, Yi Sun, Cong-cong Xing, and Qiang Duan. A source-location privacy protection strategy via pseudo normal distribution-based phantom routing in wsns. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 688–694. ACM, 2015.

[22] Qinghua Li and Guohong Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In *IEEE ICNP*, 2012.

[23] Qinghua Li and Guohong Cao. Efficient privacy-preserving stream aggregation in mobile sensing

with low aggregation error. In *Privacy Enhancing Technologies*, pages 60–81. Springer, 2013.

[24] Qinghua Li and Guohong Cao. Providing privacy-aware incentives for mobile sensing. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, volume 18, page 22, 2013.

[25] Qinghua Li and Guohong Cao. Providing efficient privacy-aware incentives for mobile sensing. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pages 208–217, 2014.

[26] Qinghua Li, Guohong Cao, and Thomas F. La Porta. Efficient and privacy-aware data aggregation in mobile sensing. *IEEE Trans. Dependable Sec. Comput.*, 11(2):115–129, 2014.

[27] Yun Li and Jian Ren. Providing source-location privacy in wireless sensor networks. In *Wireless Algorithms, Systems, and Applications*, pages 338–347. Springer, 2009.

[28] Nicolas Maisonneuve, Matthias Stevens, Maria E. Niessen, and Luc Steels. Noisetube: Measuring and mapping noise pollution with mobile phones. In *ITEE*, pages 215–228, 2009.

[29] Adnan Majeed, Ke Liu, and Nael Abu-Ghazaleh. Tarp: Timing analysis resilient protocol for wireless sensor networks. In *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*, pages 85–90. IEEE, 2009.

[30] Kiran Mehta, Donggang Liu, and Matthew Wright. Location privacy in sensor networks against a global eavesdropper. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 314–323. IEEE, 2007.

[31] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: using mobile smartphones for rich monitoring of road and traffic conditions. In *SenSys*, pages 357–358, 2008.

[32] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark H. Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *MobiSys*, pages 55–68, 2009.

[33] Balachandra Muniyal, S Sharma, et al. Wireless public key infrastructure for mobile phones. *International Journal of Network Security & Its Applications*, 4(6):111–117, 2012.

[34] Mashfiqui Rabbi, Shahid Ali, Tanzeem Choudhury, and Ethan Berke. Passive and in-situ assessment of mental and physical well-being using mobile sensors. In *Ubicomp*, pages 385–394, 2011.

[35] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[36] Eleanor G. Rieffel, Jacob T. Biehl, William van Melle, and Adam J. Lee. Secured histories: computing group statistics on encrypted data while preserving individual privacy. *CoRR*, abs/1012.2152, 2010.

[37] Sandip Roy, Marco Conti, Sanjeev Setia, and Sushil Jajodia. Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact. *Information Forensics and Security, IEEE Transactions on*, 9(4):681–694, 2014.

[38] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*. The Internet Society, 2011.

[39] Jing Shi, Rui Zhang, Yunzhong Liu, and Yanchao Zhang. Prisense: Privacy-preserving data aggregation in people-centric urban sensing systems. In *INFOCOM*, pages 758–766. IEEE, 2010.

[40] EG Sirer, M. Polte, and M. Robson. Cliquenet: A self-organizing, scalable, peer-to-peer anonymous communication substrate, december 2001. *Available from http://www. cs. cornell. edu/People/egs/papers/cliquenet-iptp. pdf*, 2001.

[41] Guillermo Suarez-Tangil, Esther Palomar, Benjamin Ramos, and Arturo Ribagorda. An experimental comparison of source location privacy methods for power optimization in wsns. In *Proceedings of the 3rd WSEAS international conference on Advances in sensors, signals and materials*, pages 79–84, 2010.

[42] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, pages 85–98, 2009.

[43] Luis von Ahn, Andrew Bortz, and Nicholas J. Hopper. k-anonymous message transmission. In *ACM Conference on Computer and Communications Security*, pages 122–130, 2003.

[44] Michael Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *EUROCRYPT*, volume 434 of *Lecture Notes in Computer Science*, pages 302–319. Springer, 1989.

[45] Piyi Yang, Zhenfu Cao, Xiaolei Dong, Tanveer Zia, et al. An efficient privacy preserving data aggregation scheme with constant communication overheads for wireless sensor networks. *Communications Letters, IEEE*, 15(11):1205–1207, 2011.

[46] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(4), 2008.

[47] Xiaoying Zhang, Hong Chen, Ke Wang, Hui Peng, Yongjian Fan, and Deying Li. Rotation-based privacy-preserving data aggregation in wireless sensor networks. In *Communications (ICC), 2014 IEEE International Conference on*, pages 4184–4189. IEEE, 2014.

[48] Yanchao Zhang, Wei Liu, Wenjing Lou, Yuguang Fang, and Younggoo Kwon. Ac-pki: anonymous and certificateless public-key infrastructure for mobile ad hoc networks. In *ICC*, pages 3515–3519, 2005.

Yuan Zhang received his BS (2005) in Automation from Tianjin University, MSE (2009) in Software Engineering from Tsinghua University, and PhD (2013) in Computer Science from State University of New York at Buffalo. He is interested in security, privacy, and economic incentives.

Qingjun Chen received his BS (2013) in computer science from Nanjing University. He is currently working toward the MS degree with the department of computer science and technology, Nanjing University. He is interested in security, privacy, and economic incentives.

Sheng Zhong received his BS (1996), MS (1999) from Nanjing University, and PhD (2004) from Yale University, all in computer science. He is interested in security, privacy, and economic incentives.