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/JIOT.2016.2585592, IEEE Internet of Things Journal

Privacy-preserving Verifiable Set Operation in Big Data for Cloud-assisted Mobile Crowdsourcing

Gaoqiang Zhuo*, Qi Jia*, Linke Guo*, Ming Li†, and Pan Li‡
*Department of Electrical and Computer Engineering, Binghamton University,
State University of New York, Binghamton, NY 13902, USA
†Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557, USA
‡Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106, USA
Email: {gzhuo1, qjia1, lguo}@binghamton.edu, mingli@unr.edu, lipan@case.edu

***Abstract*—The ubiquity of smartphones makes the mobile crowdsourcing possible, where the requester (task owner) can crowdsource data from the workers (smartphone users) by using their sensor-rich mobile devices. However, data collection, data aggregation, and data analysis have become challenging problems for a resource constrained requester when data volume is extremely large, *i.e.*, big data. In particular to data analysis, set operations, including intersection, union, and complementation, exist in most big data analysis for filtering redundant data and preprocessing raw data. Facing challenges in terms of limited computation and storage resources, cloud-assisted approaches may serve as a promising way to tackle big data analysis issue. However, workers may not be willing to participate if the privacy of their sensing data and identity are not well preserved in the untrusted cloud. In this work, we propose to use cloud to compute set operation for the requester, at the same time workers' data privacy and identities privacy are well preserved. Besides, the requester can verify the correctness of set operation results. We also extend our scheme to support data preprocessing, with which invalid data can be excluded before data analysis. By using batch verification and data update methods, the proposed scheme greatly reduces the computational cost. Extensive performance analysis and experiment-based on real cloud system have shown both the feasibility and efficiency of our proposed scheme.**

***Index Terms*—Big Data, Mobile Crowdsourcing, Verifiable Computation, Privacy.**

I. Introduction

Mobile crowdsourcing enables a task owner to obtain data from a large number of smartphone users, and further perform data analysis on the aggregated data [1]. The task owner is also known as the requester, while the participating smartphone users are mobile workers who will collect and/or sense the data for the requester. With the development of the low cost sensing devices, many sensors have been embedded on mobile devices, such as GPS, accelerator, gyroscope, digital compass, temperature sensors, etc. More sensors measuring humidity, air quality, chemical, barometer, and biomedical information can be equipped into smartphones or connected via wireless technologies. These affordable sensor-rich smartphones make them capable of sensing the environment around people and people's physiological data as well. In mobile crowdsourcing, a requester can make use of the data crowdsourced from mobile workers to achieve certain tasks [2]–[7]. For example, a transportation management bureau can utilize the speed data reported from the commuters to analyze the traffic condition [8], [9]. Obviously, mobile crowdsourcing has many advantages: first, the ubiquitous smartphone users cover a large geographic area, which makes the data and information diverse and rich; second, the requester does not need to deploy specific sensor networks or employees to collect the targeted data; third, workers can receive rewards, such as reputation and revenue from the crowdsourcing participation.

In particular to the collected data, it might not be just a single value reported in a period of time [2]–[4], [6]. Instead, we consider a more general data type requested from the requester, which could be a range of data including multiple values or even a large set of elements without order. Set operations are often used in data processing. For example, a travel agency wants to know the most popular places that the tourists have visited during holidays. Here, the data from a worker (tourist) will be a set, and thus the requester (travel agency) needs to find the intersection of all sets. Set union may be used to merge different databases collected from different database owners. Set difference is useful when a requester wants to find the unique feature of one database compared to another. When the number of workers is very large, the requester requires a huge amount of storage space for storing the crowdsourced big data even if each worker's data is relatively small. As a result, a storage limited requester is not able to handle the above task. Taking a step further, even if the requester can store all collected "Big Data", the data processing and analysis may be another stumbling block when he/she lacks computation capability. Therefore, the set operation problem over the collected data might be overwhelming.

To tackle the above issue, we introduce the cloud into the architecture as in [10]–[14]. The cloud serves as the intermediate entity between the requester and the workers. When the requester wants to perform tasks over reported data sets, she delegates the task to the cloud and waits for the result. Then, the cloud helps the requester to collect all data sets

The work of P. Li was partially supported by the U.S. National Science Foundation under grants CNS-1343220, CNS-1149786. The work of M. Li was partially supported by National Science Foundation under grant CNS-1566634.

from the workers and computes the set operation. However, this solution may not work well because the public cloud is untrusted, and it may suffer severe attacks, e.g., hacked by an adversary [4], [15]–[19]. On the one hand, in mobile crowdsourcing, data privacy is a big concern for the workers, for which sensitive data should not be revealed directly to the cloud. In the above example, a worker is unwilling to expose her travel destinations to the cloud because this might breach her location privacy and cause physical attacks [12], [20], [21]. On the other hand, security issues also exist in cloud-assisted set operation for mobile crowdsourcing. Crowdsourced data might be modified by an untrusted cloud if it knows a data comes from a specific worker. An untrusted cloud may return a wrong set operation result to the requester. When computing set operations, the cloud may discard some data sets to reduce expense. Facing these challenges, we propose a verifiable set operation in big data for cloud-assisted mobile crowdsourcing. Our solution leverages the cloud to release computation burden of the requester while preventing all the above security and privacy issues. With our scheme, workers' data and identity privacy are well preserved. Meanwhile, the requester can verify the correctness of the result retrieved from the cloud. We also extend our scheme to support data preprocessing, batch verification, and efficient data update.

Related Works:

Private Set Intersection: Many works have been done to achieve private set intersection (PSI) [22]–[28]. PSI enables two parties to compute the intersection with private input and only the intersection is known to each party. The first protocol for PSI is proposed in [22]. Kissner *et al.* [23] use polynomial representations to solve set operations between two parties, and utilize Paillier Crypto system to protect the privacy of polynomials when trusted third party is not available. In [24]–[26], private set intersection with linear complexity is proposed. Dong *et al.* [27] make use of a new variant of bloom filter to achieve efficient PSI. In [28], bloom filter and homomorphic encryption are used to achieved outsourced private set intersection. All of these works can achieve private set intersection, however, none of them offers verifiability of the result. Thus, none of them can be applied in our work directly.

Verifiable Computation: Verifiable computation was introduced and formalized by Gennaro *et al.* in [29], which enables a resource-limited client to outsource the computation of a function to one or more workers. The workers return the result of function evaluation. The client should be able to efficiently verify the correctness of the results. After that, many works has been done to achieve verifiable computation [30]–[33]. In [30], they propose the first practical verifiable computation scheme for high degree polynomial functions. Fiore *et al.* [31] propose a solution for publicly verifiable computation of large polynomials and matrix computations, where anyone can verify the correctness of the results. Papamanthou *et al.* [32], [33] study the problem of cryptographically checking the correctness of outsourced set operations performed by an untrusted server, and the sets are dynamic. However, all of them [30]–[33] are designed for verifiable computation over plaintexts where data privacy is not considered. Verifiable computation for encrypted data is provided in [34]–[36]. Fiore *et al.* [34] use homomorphic encryption and homomorphic hashing to enable a client to query outsourced encrypted datasets, get encrypted result, and verify its correctness. In [35], Abadi *et al.* , design a delegated PSI on outsourced datasets based on a novel point-value polynomial representation. This protocol allows multiple clients to upload their datasets and obtain the intersection from the cloud. Guo *et al.* [36] propose a verifiable computation over encrypted data for mHealh systems, where a patient can ask the cloud to evaluate a polynomial over his encrypted personal health record, and verify the correctness of the evaluation result. Although [34]–[36] can achieve verifiable computation over encrypted data, they are all two party architecture, which is not suitable for our scenario.

Our Contributions: Generally speaking, we have made the following major contributions:

- We propose an efficient solution for the set operation in big data analysis based on the data collected from mobile crowdsourcing.
- We introduce the cloud as an intermediate entity to the traditional mobile crowdsourcing, where worker's data privacy and identity privacy are well protected.
- For requesters, they can verify the correctness of computation results retrieved from the cloud.
- We further extend the basic scheme to useful applications in big data analysis, such as data preprocessing, batch verification, and efficient data update.

The remainder of this paper is organized as follows. Section II introduces preliminaries, assumptions and problem formulation. Section III presents the system model, security model and design objectives. The proposed scheme is described in detail in Section IV, followed by extensions in Section V. Performance analysis is given in Section VI, and Section VII concludes the paper.

II. Preliminaries and Problem Formulation

A. Preliminaries

1) Bilinear Pairing: A bilinear pairing is a map $e : G \times G \rightarrow G_T$, where G and G_T are two multiplicative cyclic groups of the same prime order p and G is generated by g. The pairing e has the following properties [37], [38]:

- **Bilinearity :** $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in G$ and random numbers $a, b \in Z_p^*$;
- **Computability :** For all $u,\ v \in G$, $e(u, v)$ can be computed efficiently;
- **Non-degeneracy :** For $g \in G$, $e(g, g) \neq 1$.

2) Bilinear-map Accumulator: The bilinear-map accumulator is an efficient way to provide short proofs of membership for elements that belong to a set. Let $s \in Z_p^*$ be a randomly chosen value that constitutes the trapdoor in the scheme. The accumulator accumulates elements in $Z_p - \{s\}$, outputting a value that is an element in G. For a set X of elements in

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/JIOT.2016.2585592, IEEE Internet of Things Journal

$Z_p - \{s\}$, the accumulation value $\texttt{acc}(X)$ of set X is defined as

$$\texttt{acc}(X) = g^{\prod_{x \in X}(x+s)}.$$

With the help of $\texttt{acc}(X)$, each element in X has a unique membership proof. Specifically, the proof of subset containment of a set $S \subseteq X$ is the witness $W_{S,X}$ as,

$$W_{S,X} = g^{\prod_{x \in X-S}(x+s)}.$$

The subset containment of S in X can be checked through

$$e(W_{S,X}, g^{\prod_{x \in S}(x+s)}) \stackrel{?}{=} e(\texttt{acc}(X), g),$$

by any verifier with access to public information.

3) Polynomial interpolation with FFT: Let $\prod_{i=1}^{n}(x_i + s) = \sum_{i=0}^{n} b_i s^i$ be a degree-n polynomial. The coefficients $b_0, b_1, ..., b_{n-1}$, where $b_i \neq 0$ of the polynomial can be computed with $O(n \log n)$ complexity, given $x_1, x_2, ..., x_n$ [32].

B. Cryptograohic Assumptions

1) Discrete Logarithmic Problem (DLP): Let u, v be two elements in G. It is computationally intractable to find an integer a, such that $u = v^a$.

2) Computational Diffie-Hellman (CDH) Problem: Given (u, u^a, u^b) for $u \in G$ and unknown $a, b \in Z_p^*$, it is intractable to compute u^{ab} in polynomial time.

3) Decisional Diffie-Hellman (DDH) Problem: Given (u, u^a, u^b, u^c) for $u \in G$ and unknown $a, b, c \in Z_p^*$, it is easy to tell whether $c = ab \mod p$ by checking if $e(u^a, u^b) = e(u^c, g)$.

4) Bilinear q-strong Diffie-Hellman Assumption: Let k be the security parameter and (p, G, G_T, e, g) be a tuple of bilinear paring parameters. Given the elements $g, g^s, ..., g^{s^q} \in G$ for some s chosen at random from Z_p^*, no probabilistic polynomial-time algorithm can output a pair $(a, e(g,g)^{1/(a+s)}) \in Z_p \times G$, except with negligible probability.

C. Problem Formulation

When a requester wants to crowdsource data sets from the mobile workers and performs set operations based on the collected sets, the direct solution is to store all data sets locally and computes the result by himself. However, this solution does not work when the requester has limited storage and computation resources. Therefore, we introduce the cloud between the requester and the workers. The cloud can store the data sets and compute the result on behalf of the requester. In our work, we require that workers' data privacy and identity privacy should be protected. Specifically, the cloud should not know the plaintext of the data sets or the exact source of a data set. We formulate this problem as a privacy-preserving set operation. The data privacy is preserved through ElGamal encryption [39] and a keyed hash function [40]. While the identity privacy is achieved through ring signature [41]. The requester will get the computation result from the cloud together with a proof information. Therefore, we formulate this problem as a verifiable computation outsourcing problem. The correctness of the intersection set $I = S_1 \cap S_2 \cap ... \cap S_t$ is based on the following two conditions: [32]:

$$\textit{Subset condition}: \; I \subseteq S_1 \wedge I \subseteq S_2 \wedge ... \wedge I \subseteq S_t;$$
$$\textit{Completeness condition}: \; (S_1 - I) \cap (S_2 - I) \cap ... \cap (S_t - I) = \emptyset.$$

The subset condition is achieved by using bilinear map accumulator. The completeness condition is achieved by using the following property [32]: if polynomials $p_1, p_2, ..., p_t$ are co-prime to each other, then there exist polynomials $q_1, q_2, ..., q_t$ such that $q_1 p_1 + q_2 p_2 + \cdots + q_t p_t = 1$.

We use set properties to remove invalid data sets at the cloud. Supposing the range limit set defined by the requester is S_R, which means all valid data should be within S_R. Worker W_i has data set S_i. There are four possible relationships between S_R and S_i, as shown in Fig. 1. When the requester delegates the set intersection computation to the cloud, the cloud needs to excludes set S_i if the relationship between S_i and S_R is one of (1) - (3) in Fig. 1, which means S_i contains at least an element that's not in S_R. The set representation for this event is $S_R \cap S_i \neq S_i$.

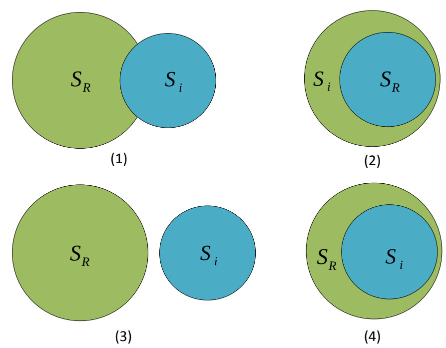


Fig. 1. Relationships between S_R and S_i

III. System Model

A. System Model

As shown in Fig. 2, our system model mainly consists of four entities, the mobile workers (*W*), the requester (*R*), the cloud (*C*), and the trusted authority (*TA*).

- *Trust Authority (TA): TA* is responsible for initializing the whole system which includes registering workers, requesters and the cloud, generating public parameters, and distributing keys, and maintaining the system. *TA* may be offline unless a dispute arises.
- *Requester:* The requester wants to obtain the intersection set of the workers' data sets. However, due to his/her limitation on the storage and computation capability, the requester will delegate storage and most of the computation tasks to the cloud.
- *Cloud:* The cloud receives the delegation requests from the requester and the encrypted data sets from mobile workers, then it computes the intersection set for the requester. The cloud also needs to provide some proof information to prove the correctness of the result.

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/JIOT.2016.2585592, IEEE Internet of Things Journal

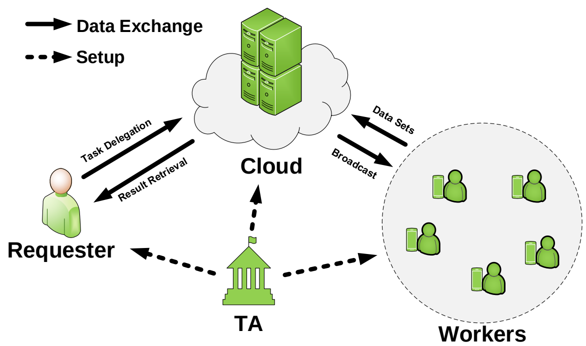


Fig. 2. System Architecture

- *Mobile Workers:* Mobile workers refer to those who have smartphones and are willing to contribute data to the requester's tasks. Each worker generates her own data set, and encrypts it before sending it to the cloud.

B. Security Model

In our security model, the *TA* is fully trusted and will not be breached by any adversary. The security requirements for other entities are given below.

- *Mobile Workers:* In our scheme, a worker's data set should be kept confidential from other workers and the cloud.
- *Requester:* The security requirement for the requester is that he should be able to verify the correctness of the computation result received from the cloud.
- *Cloud:* In our model, the cloud is curious but honest. It should not be able to know workers' data sets or the intersection set.

We exclude several types of attacks that are beyond the discussion of this paper. Our scheme may fail to fight against the denial-of-service attacks when numbers of malicious workers send requests to the cloud with dummy crowdsourcing data. Collusion attacks may not be thwarted when malicious users use brute force to compromise encrypted data. The insider attack and global observer attack are also not considered in this paper.

C. Design Objectives

We have three main objectives for our privacy-preserving verifiable computation of set intersection for mobile crowdsourcing. First, the cloud can compute the intersection set of the workers' data sets without knowing the content and source of the data sets, thus workers' data privacy and identity privacy are well preserved. Second, the requester can verify the correctness of the intersection set retrieved from the cloud. Third, to better adapt the privacy requirements for the collected big data, the proposed scheme should be scalable and efficient for processing huge volume of reported data.

TABLE I
NOTATION USED IN OUR SCHEME

Notation	Description
k	system security parameter
p	prime order of group G, G_T and Z_p
g	generator of group G
$H_0()$	hash function from a bit string to a field Z_p
$H(sk_h, \cdot)$	keyed hash function
S_i	data set of worker W_i
q	upper limit of $\|S_i\|$
C_i	ciphertext set of S_i
H_i	hashing set of S_i
(pk_i, sk_i)	public/private key pair of worker W_i
(pk, sk)	public/private key pair of the requester
I_S	intersection set of all data sets
I_H	intersection set of all hashing sets
I_C	set of ciphertexts whose plaintexts are elements in I_S

IV. OUR PROPOSED SCHEME

A. Overview

The trusted authority (*TA*) registers the requester, the cloud, and the workers by assigning a public/private key pair to each of them during the system initialization. Whenever the requester needs to compute the intersection set, he sends his request and public key pk to the cloud. Then, he waits for the results from the cloud. The cloud broadcasts the requester's task and public key pk to all the workers. Every worker W_i generates his data set S_i, and encrypts it with pk. The data will be signed with ring signature before sending to the cloud. After receiving encrypted data sets from all workers, the cloud verifies the authenticity of each of them, and computes the intersection set based on the encrypted data sets. Then the cloud sends the result together with its corresponding proof information to the requester. Finally, the requester decrypts the result and checks its correctness.

B. System Initialization

In this phase, *TA* first generates necessary parameters and keys for the system. Then, *TA* registers all workers, requesters and cloud into the system. We present the two steps as follows. Main notations are listed in Table. I.

1) General Setup: Given the security parameter k, *TA* generates the bilinear parameters (p, G, G_T, e, g). Also, a hash function $H_0() : [0,1]^* \to Z_p$ is defined. *TA* chooses a random value $s \in Z_p$, and computes $g^s, g^{s^2}, ..., g^{s^q}$. Then *TA* publishes $\{p, G, G_T, e, g, g^s, g^{s^2}, ..., g^{s^q}, H_0()\}$.

2) Entities Registration: Assume there are t mobile workers in the system: $\{W_1, W_2, ..., W_t\}$. For each worker W_i, *TA* assigns him a public/private key pair (pk_i, sk_i), where $sk_i = x_i \in_R Z_p$ and $pk_i = g^{x_i}$. *TA* registers the cloud and the requester by sending the private/public key pairs $(sk_c, pk_c) = (x_c, g^{x_c})$ and $(sk, pk) = (x, g^x)$ to the cloud and the requester respectively, where x_c and x are random number from Z_p. Besides, both requester and workers obtain the encryption key k_h for a private hash function $H(k_h, \cdot) : G \to Z_p$.

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/JIOT.2016.2585592, IEEE Internet of Things Journal

C. Mobile Crowdsourcing

In our scheme, the plaintext space is group G, while the data space could be of any type. Therefore, the requester needs to build a mapping table between the data space and the plaintext space for every task τ. The mapping table can be built as follows. The requester first defines a data space for the collected data. Then for every element in data space, a new random element in plaintext space G is chosen. The mapping table for the task is public to all. Then he sends τ and pk as a task to the cloud. After receiving τ and pk from the requester, the cloud broadcast the task to all the workers.

When a worker W_i receives the task from the cloud, she generates a data set based on task tag τ, and maps every element in data set to element in plaintext space according to the mapping table provided by the requester to get her plaintext set $S_i = \{m_{i,1}, m_{i,2}, ..., m_{i,n_i}\}$, where n_i is the cardinality of S_i, and $m_{i,j} \in G$, $j = 1, 2, ..., n_i$. In the following, we assume every worker will map her collected data set to plaintext set automatically, and use data set and plaintext set interchangeably. Then, a worker needs to perform the following steps.

1) Data Encryption: Given data set $S_i = \{m_{i,1}, m_{i,2}, ..., m_{i,n_i}\}$ and the requester's public key $pk = g^x$, the worker W_i chooses n_i random values $r_{i,j} \in_R Z_p$, $j = 1, 2, ..., n_i$, and computes ciphertext set $C_i = \{c_{i,1}, c_{i,2}, ..., c_{i,n_i}\}$, where $c_{i,j}$ is computed as follows,

$$c_{i,j} = (g^{r_{i,j}}, m_{i,j} \cdot pk^{r_{i,j}}).$$

2) Data Hashing: Given data set $S_i = \{m_{i,1}, m_{i,2}, ..., m_{i,n_i}\}$ and the shared secret hash key sk_h, the worker W_i computes the hashing set $H_i = \{h_{i,1}, h_{i,2}, ..., h_{i,n_i}\}$, where $h_{i,j}$ is computed as follows,

$$h_{i,j} = H(sk_h, m_{i,j}). \quad (1)$$

3) Data Accumulation: After obtaining the hashing set $H_i = \{h_{i,1}, h_{i,2}, ..., h_{i,n_i}\}$, worker W_i needs to compute the accumulation value of H_i:

$$\texttt{acc}(H_i) = g^{\prod_{j=1}^{n_i}(h_{i,j}+s)}.$$

Because s is a secret parameter known only to the requester, worker W_i cannot directly computes $\prod_{j=1}^{n_i}(h_{i,j} + s)$, she first finds out the coefficients $\{b_0, b_1, ..., b_{n_i}\}$, where

$$\sum_{j=0}^{n_i} b_j \cdot s^j = \prod_{j=1}^{n_i}(h_{i,j} + s).$$

This can be achieved by using polynomial interpolation with FFT. Then, worker W_i uses $\{b_0, b_1, ..., b_{n_i}\}$ and public parameters $\{g^s, g^{s^2}, ...g^{s^q}\}$ to compute $\texttt{acc}(H_i)$ as follows,

$$\begin{aligned} & g^{b_0} \cdot (g^s)^{b_1} \cdots (g^{s^{n_i}})^{b_{n_i}} \\ & = g^{\sum_{j=0}^{n_i} b_j \cdot s^j} = g^{\prod_{j=1}^{n_i}(h_{i,j}+s)} \\ & = \texttt{acc}(H_i). \end{aligned}$$

4) Signature Generation: When finishing the above three steps, worker W_i will compute her signature on $\texttt{acc}(H_i)$. The original ring signature scheme is described as follows [41]. Given all workers' public keys $(pk_1, pk_2, ..., pk_t)$, $\texttt{acc}(H_i)$, and her private key sk_i, worker W_i randomly chooses $b_{ij} \in Z_p$ for all the other workers W_j, where $j = 1, 2, ..., t, j \neq i$, and computes

$$\texttt{Sig}_{ij} = g^{b_{ij}}.$$

Then, W_i computes $\tau_i = g^{H_0(\texttt{acc}(H_i))}$, and

$$\texttt{Sig}_{ii} = \left(\frac{\tau_i}{\prod_{j \neq i} pk_j^{b_{ij}}}\right)^{1/sk_i}.$$

The ring signature for $\texttt{acc}(H_i)$ is $\texttt{Sig}_{W_i} = \{\texttt{Sig}_{i1}, \texttt{Sig}_{i2}, ..., \texttt{Sig}_{it}\}$. However, in real life, when the number of workers t is large and workers are distributed over a wide area, it's very time-consuming or impossible for a worker to communicate with all the other workers to get their public keys. Therefore, we cannot directly apply the above ring signature. Instead, we assume every worker belongs to a ring signature group, and all workers in the same group are in proximity with each other. We use L_i to denote the index set of workers who are in the same signature group as W_i, and $K_{min}|L_i| \leq K_{max}$, where K_{min} and K_{max} are the minimum and maximum number of workers in any signature group. Then, W_i's ring signature is $\texttt{Sig}_{W_i} = \{\texttt{Sig}_{ij}\}, j \in L_i$, where $\texttt{Sig}_{ii} = \left(\frac{\tau_i}{\prod_{j \neq i} pk_j^{b_{ij}}}\right)^{1/sk_i}$, and $\texttt{Sig}_{ij} = g^{b_{ij}}$, $j \in L_i - \{i\}$. Finally, worker W_i sends $\{C_i, H_i, \texttt{acc}(H_i), \texttt{Sig}_{W_i}\}$ to the cloud.

D. Intersection Computation

After receiving $\{C_i, H_i, \texttt{acc}(H_i), \texttt{Sig}_{W_i}, L_i\}$, $i = 1, 2, ..., t$, from all workers $\{W_1, W_2, ..., W_t\}$, the cloud will compute the intersection set for the requester. Before performing the computation, the cloud first verifies if the received data really comes from valid workers by computing $\tau_i = g^{H_0(\texttt{acc}(H_i))}$, and checking

$$e(\tau_i, g) \stackrel{?}{=} \prod_{j \in L_i} e(\texttt{Sig}_{ij}, pk_j),$$

where $i = 1, 2, ..., t$. If the above equation holds, the cloud knows that the data comes from one of the valid workers. Otherwise, the cloud refuses the data.

Proof of correctness.

$$\begin{aligned} & \prod_{j \in L_i} e(\texttt{Sig}_{ij}, pk_j) = e(\texttt{Sig}_{ii}, pk_i) \cdot \prod_{j \in L_i - \{i\}} e(\texttt{Sig}_{ij}, pk_j) \\ & = e\left(\left(\frac{\tau_i}{\prod_{j \in L_i - \{i\}} pk_j^{b_{ij}}}\right)^{1/sk_i}, g^{x_i}\right) \cdot \prod_{j \in L_i - \{i\}} e(g^{b_{ij}}, g^{x_j}) \\ & = e\left(\frac{\tau_i}{\prod_{j \in L_i - \{i\}} g^{x_j b_{ij}}}, g\right) \cdot \prod_{j \in L_i - \{i\}} e(g^{x_j b_{ij}}, g) \\ & = e(\tau_i, g). \end{aligned}$$

□

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/JIOT.2016.2585592, IEEE Internet of Things Journal

After successful verification of the ring signatures, the cloud computes the intersection set for the requester. Define I_S as the intersection set of the original data sets $S_1, S_2, ..., S_t$, *i.e.*,

$$I_S = S_1 \cap S_2 \cap \cdots \cap S_t.$$

Because all data sets $S_1, S_2, ..., S_t$ are encrypted by workers before being sent to the cloud, the cloud is unable to find I_S for the requester based on the ciphertexts. Instead, the cloud needs to find all the ciphertexts whose plaintexts correspond to the intersection set I_S. Assuming $m_{i,j} \in I_S$, for some i's and j's, then we define I_C as the set of ciphertexts $c_{i,j}$ of all elements $m_{i,j} \in I_S$,

$$I_C = \{c_{i,j}\}_{m_{i,j} \in I_S}.$$

The cloud derives I_C based on hashing sets $H_1, H_2, ..., H_t$, because $m_{i,j}$ and $h_{i,j}$ are one-to-one mapping, I_S is equivalent to I_H, where

$$I_H = H_1 \cap H_2 \cap \cdots \cap H_t.$$

Take H_1 and H_2 as an example, where $H_1 = \{h_{1,1}, h_{1,2}, ..., h_{1,n_1}\}$ and $H_2 = \{h_{2,1}, h_{2,2}, ..., h_{2,n_2}\}$. If $h_{1,u} = h_{2,v}$, $1 \leq u \leq n_1, 1 \leq v \leq n_2$, then $m_{1,u} = m_{2,v}$, $m_{1,u} \in S_1$ and $m_{2,v} \in S_2$. This means that $m_{1,u}$ (or $m_{2,v}$) $\in S_1 \cap S_2$. If $m_{1,u} \in S_1 \cap S_i$ for all $i = 1, 2, ..., t$, then the cloud knows that $c_{1,u} \in I_C$. After comparing every pair of elements $h_{i,u} \in H_i$ and $h_{j,v} \in H_j$, the cloud finally obtains I_H and I_C.

E. Proof Generation

After obtaining I_H and I_C, the cloud continues to generate proof information for the correctness of I_C. First, the cloud computes

$$p_i(s) = \prod_{h \in H_i - I_H} (h + s),$$

using polynomial interpolation with FFT. After that, the cloud computes δ as follows,

$$\delta = \{\delta_i\} = \{\mathtt{acc}(H_i - I_H)\} = \{g^{p_i(s)}\},$$

where $i = 1, 2, ..., t$. The accumulation values of the difference sets $H_i - I_H$ will be used by the requester to verify the subset condition. Then, the cloud finds a coefficient set $\{q_1(s), q_2(s), ..., q_t(s)\}$ [32], such that

$$q_1(s)p_1(s) + q_2(s)p_2(s) + ... + q_t(s)p_t(s) = 1,$$

and $\{g^{q_1(s)}, g^{q_2(s)}, ..., g^{q_t(s)}\}$ are computed accordingly. The set of values $\{g^{q_1(s)}, g^{q_2(s)}, ..., g^{q_t(s)}\}$ will be used by the requester to verify the completeness condition. Finally, the cloud sends $\{I_C, \delta, \{g^{q_i(s)}\}, \mathtt{acc}(H_i)\}$ to the requester.

F. Result Retrieval and Verification

When the requester receives the result and corresponding proof information $\{I_C, \delta, \{g^{q_i(s)}\}, \mathtt{acc}(H_i)\}$ from the cloud, she first decrypts I_C with her private key $sk = x$ to get I'_S as follows,

$$I'_S = Dec_{sk}(I_C) = \{Dec_{sk}(c_{i,j})\}$$
$$= \left\{ \frac{m_{i,j} \cdot pk^{r_{i,j}}}{(g^{r_{i,j}})^{sk}} \right\} = \{m_{i,j}\}.$$

Then, the requester computes I'_H with the private hash key sk_h based on I'_S,

$$I'_H = \{H(sk_h, m_{i,j})\} = \{h_{i,j}\}.$$

Next, the requester computes accumulation value of I'_H,

$$\mathtt{acc}(I'_H) = g^{\prod_{h \in I'_H}(h+s)}.$$

Finally, the requester checks if the following equations hold.

$$e(\mathtt{acc}(I'_H), \delta_i) \stackrel{?}{=} e(\mathtt{acc}(H_i), g), \quad (2)$$

$$\prod_{i=1}^{t} e\left(\delta_i, g^{q_i(s)}\right) \stackrel{?}{=} e(g, g). \quad (3)$$

Eq.(2) can verify the subset condition, and Eq.(3) can verify the completeness condition. If all the above two equations hold, the requester accepts the result. Otherwise, the requester discards it. If the returned result I_C is correct, then $I'_S = I_S$ and $I'_H = I_H$. The two checking equations hold as follows.

Proof of correctness.

$$\begin{aligned} & e(\mathtt{acc}(I'_H), \delta_i) \\ & = e(g^{\prod_{h \in I_H}(h+s)}, g^{\prod_{h \in H_i - I_H}(h+s)}) \\ & = e(g, g)^{\prod_{h \in H_i}(h+s)} \\ & = e(\mathtt{acc}(H_i), g) \end{aligned}$$

$$\begin{aligned} & \prod_{i=1}^{t} e\left(\delta_i, g^{q_i(s)}\right) = \prod_{i=1}^{t} e\left(g^{p_i(s)}, g^{q_i(s)}\right) \\ & = e(g, g)^{\sum_{i=1}^{t} p_i(s) q_i(s)} \\ & = e(g, g). \end{aligned}$$

□

V. Extensions

Although the basic scheme satisfies all the security and privacy requirements for the cloud-assisted mobile crowdsourcing, it is still challenging on integrating certain meaningful designs for the big data analysis. Due to the large volume crowdsourcing data, the efficiency on deriving the intersection set, verification on identities, and data update are not as good as expected. Therefore, we continue to use the same methodology to extend the above basic scheme to satisfy the big data analysis for mobile crowdsourcing.

A. Verifiable Data Preprocessing

First of all, to reduce the cost on processing the operation on collected data, we need to carefully exam the reported data. Normally, the requester has a specific range requirements on the data set. In the previous example, the requester may determine that only sets of a specific range of tourist sites are eligible for the computation of intersection. This is especially

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/JIOT.2016.2585592, IEEE Internet of Things Journal

useful for improving efficiency and accuracy in big data analysis, because it will greatly reduce the unnecessary raw data for data processing. Suppose the range limit set defined by the requester is S_R, and worker W_i has data set S_i. As we mentioned in the problem formulation, there are four possible relationships between S_R and S_i, as shown in Fig. 1.

Three steps are needed to achieve verifiable data preprocessing. First, the requester needs to compute a hashing set $H_R = \{h_{R,1}, h_{R,2}, ..., h_{R,z}\}$ for $S_R = \{m_{R,1}, ..., m_{R,z}\}$, z is the size of S_R, and $h_{R,i}$ is computed in the same way as in Eq. (1). Second, based on relationships (1) - (3) in Fig. 1 the cloud finds out all sets S_i which satisfies,

$$S_R \cap S_i \neq S_i.$$

and removes S_i. Because S_i is not within the range limit defined by the requester. At this step, the cloud also needs to prove that S_i is not an eligible set while other sets are by computing $\mathtt{acc}(H_R - H_i)$, and sends $\{\mathtt{acc}(H_R - H_i), \mathtt{acc}(H_i)\}$ to the requester. Here, $\mathtt{acc}(H_i)$ comes from worker W_i who can use *sign-and-encrypt* before sending it to the cloud, where the encryption key is shared between workers and the requester. Finally, the requester checks if the following equation holds,

$$e(\mathtt{acc}(H_R), g) \stackrel{?}{=} e(\mathtt{acc}(H_R - H_i), \mathtt{acc}(H_i)).$$

The equation holds if and only if S_i is a subset of S_R.

Proof of correctness.

$$\begin{aligned} RHS &= e(\mathtt{acc}(H_R - H_i), \mathtt{acc}(H_i)) \\ &= e(g^{\prod_{h \in H_R - H_i}(h+s)}, g^{\prod_{h \in H_i}(h+s)}) \\ &= e(g, g)^{\prod_{h \in H_R}(h+s)} \\ &= LHS. \end{aligned}$$

□

B. Batch Verification

To reduce the computational costs at both the cloud and the requester, we use batch verification for ring signature verification at the cloud, and for correctness verification at the requester.

1) Ring Signature Verification: When the cloud receives $\{C_i, H_i, \mathtt{acc}(H_i), \mathtt{Sig}_{W_i}\}$, $i = 1, 2, ..., t$, from all workers, it computes $\tau_i = g^{H_0(\mathtt{acc}(H_i))}$, $i = 1, 2, ..., t$. Then, instead of checking each worker's ring signature one by one, cloud only checks if the following equation holds.

$$e(\prod_{i=1}^{t} \tau_i, g) \stackrel{?}{=} \prod_{i=1}^{t} \prod_{j \in L_i}^{t} e(\mathtt{Sig}_{i,j}, pk_j).$$

If the above equation holds, the cloud knows that the data comes from valid workers. Otherwise, the cloud refuses the data.

Proof of correctness.

$$\begin{aligned} &\prod_{i=1}^{t} \prod_{j \in L_i} e(\mathtt{Sig}_{ij}, pk_j) = \prod_{i=1}^{t} \left(e(\mathtt{Sig}_{ii}, pk_i) \cdot \prod_{j \in L_i - \{i\}} e(\mathtt{Sig}_{ij}, pk_j) \right) \\ &= \prod_{i=1}^{t} \left(e\left(\left(\frac{\tau_i}{\prod_{j \in L_i - \{i\}} pk_j^{b_{ij}}} \right)^{1/sk_i}, g^{x_i} \right) \cdot \prod_{j \in L_i - \{i\}} e(g^{b_{ij}}, g^{x_j}) \right) \\ &= \prod_{i=1}^{t} \left(e\left(\frac{\tau_i}{\prod_{j \in L_i - \{i\}} g^{x_j b_{ij}}}, g \right) \cdot \prod_{j \in L_i - \{i\}} e(g^{x_j b_{ij}}, g) \right) \\ &= \prod_{i=1}^{t} e(\tau_i, g) = e(\prod_{i=1}^{t} \tau_i, g). \end{aligned}$$

□

2) Result Correctness Verification: When the requester receives set intersection result from the cloud, she needs to verify the correctness of result. The verification involves checking if both subset containment condition and completeness condition are satisfied. Checking the subset containment condition is computation intensive, because its complexity depends on the number of workers. With batch verification, the requester only needs to check one equation for subset condition, by changing Eq. (2) to the following.

$$e(\mathtt{acc}(I'_H), \prod_{i=1}^{t} \delta_i) \stackrel{?}{=} e(\prod_{i=1}^{t} \mathtt{acc}(H_i), g)$$

If all the above equation and Eq. (3) hold, the requester accepts the result. Otherwise, the requester discards it. If the returned result I_C is correct, then $I'_S = I_S$ and $I'_H = I_H$. The proof of correctness is given below.

Proof of correctness.

$$\begin{aligned} &e\left(\mathtt{acc}(I'_H), \prod_{i=1}^{t} \delta_i \right) = e\left(g^{\prod_{h \in I_H}(h+s)}, \prod_{i=1}^{t} \left(g^{\prod_{h \in H_i - I_H}(h+s)} \right) \right) \\ &= e\left(g, g\right)^{\sum_{i=1}^{t} \prod_{h \in H_i}(h+s)} = e\left(g^{\sum_{i=1}^{t} \prod_{h \in H_i}(h+s)}, g \right) \\ &= e\left(\prod_{i=1}^{t} \mathtt{acc}(H_i), g \right). \end{aligned}$$

□

C. Data Update

When a worker W_i wants to update $U = \{m_{i,j}\} \subset S_i$ to $U' = \{m'_{i,j}\} \subset S'_i$, where S'_i is the new data set, she does not need to compute C_i and H_i from scratch. The computation can be delegated to the cloud. To update the ciphertext set from C_i to C'_i, worker W_i computes a set $\{m'_{i,j}/m_{i,j}\} \mod p$ and sends $(\{m'_{i,j}/m_{i,j}\}, I)$ to the cloud, where I is the index set telling the cloud which data to update. After receiving $(\{m'_{i,j}/m_{i,j}\}, I)$, the cloud updates $c_{i,j}$, where $m_{i,j} \in U$ to $c'_{i,j}$ as follows,

$$\begin{aligned} c'_{i,j} &= (g^{r_{i,j}}, \frac{m'_{i,j}}{m_{i,j}} \cdot m_{i,j} \cdot pk^{r_{i,j}}) \\ &= (g^{r_{i,j}}, m'_{i,j} \cdot pk^{r_{i,j}}) \end{aligned}$$

The computation of $\mathtt{acc}(H'_i)$ can also be delegated to the cloud, where H'_i is the new hashing set. First, W_i computes H'_i

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/JIOT.2016.2585592, IEEE Internet of Things Journal

based on S_i', and sends H_i' to the cloud. Then cloud computes $\mathtt{acc}(H_i')$ using public parameters $\{g^s, ..., g^{s^q}\}$, and sends $\mathtt{acc}(H_i')$ back to W_i. However, since the cloud is untrusted, W_i needs to verify the correctness of $\mathtt{acc}(H_i')$ by checking

$$e\left(g^{\prod_{h\in H_i}(h+s)}, g^{1/\prod_{h\in H_{i,U}}(h+s)}\right) \stackrel{?}{=} e\left(g^{\prod_{h\in H_i'}(h+s)}, g^{1/\prod_{h\in H_{i,U'}}(h+s)}\right),$$

where $H_{i,U}$ is the hashing set of U, and $H_{i,U'}$ is the hashing set of U'.

Proof of correctness.

$$LHS = e\left(g^{\prod_{h\in H_i}(h+s)}, g^{1/\prod_{h\in H_{i,U}}(h+s)}\right) = e\left(g, g\right)^{\prod_{h\in H_i - H_{i,U}}(h+s)},$$

$$RHS = e\left(g^{\prod_{h\in H_i'}(h+s)}, g^{1/\prod_{h\in H_{i,U'}}(h+s)}\right) = e\left(g, g\right)^{\prod_{h\in H_i' - H_{i,U'}}(h+s)}.$$

□

Because $H_i - H_{i,U} = H_i' - H_{i,U'}$, then $LHS = RHS$.

VI. Protocol Evaluation

In this section, we first analyze how the security and privacy goals are achieved. Then, we show the feasibility and efficiency of our proposed scheme with extensive simulation.

A. Security and Privacy Analysis

In our scheme, workers' data privacy and identity privacy are protected from the cloud. The security refers to the correctness of the result, and privacy refers to workers' data privacy and identity privacy.

1) Data Privacy: When a worker W_i participates in the data crowdsourcing, she generates the data set S_i and then encrypts every element $m_{i,j} \in S_i$ with the requester's public key $pk = g^x$ to get $c_{i,j} = (g^{r_{i,j}}, m_{i,j} \cdot pk^{r_{i,j}})$. According to the discrete logarithmic problem (DLP) assumption, the cloud will not find $r_{i,j}$ given $g^{r_{i,j}}$ and the modulo p. Furthermore, the private key x is kept to the requester, so the cloud cannot compute $m_{i,j}$ in polynomial time based on the ciphertext set. Worker W_i also sends a hashing set H_i to the cloud. Every element in H_i is computed through a keyed hash function, and the key sk_h is only shared between the requester and workers. Without sk_h the cloud is not able to find $m_{i,j}$ due to the one-way and collision resistance of the underlying hash function.

2) Identity Privacy: In our scheme, ring signature is used by the workers to protect their exact identities. According to [41], if there are t_0 workers in a signature group, and every worker signs her data with ring signature, then the probability of identifying the owner of the signature by the cloud is at most $1/t_0$, which is equal to random guessing.

TABLE II
Simulation Setup

Parameters	Setting
Number of workers t	10000 to 50000
Size of a data set n_i	1000 to 5000
Size of intersection set	50 to 250
Group order	128 bit

3) Correctness of Result: The requester is able to check the correctness of the result retrieved from the cloud. If the result I_C is not correct, then the requester will get the wrong intersection set I_S' and wrong hashing set I_H'. Followed by incorrect accumulation value $\mathtt{acc}(I_H')$. Based on the Computational Diffie-Hellman (CDH) problem and Bilinear q-strong Diffie-Hellman Assumption, the cloud cannot figure out δ_i given $\mathtt{acc}(I_H')$ and $\mathtt{acc}(H_i)$ to make $e(\mathtt{acc}(I_H'), \delta_i) = e(\mathtt{acc}(H_i), g)$ hold. As a result, the requester verifies if I_S' satisfies subset containment condition by checking $e(\mathtt{acc}(I_H'), \delta_i) \stackrel{?}{=} e(\mathtt{acc}(H_i), g)$. If the subset condition is satisfied, then δ_i is the valid witness of I_H' in H_i. By checking Eq. (3), the requester knows if the completeness condition is satisfied or not. If Eq. (3) holds, then according to DDH assumption, $q_1(s)p_1(s) + q_2(s)q_2(s) + ... + q_t(s)p_t(s) = 1$ must be true. This means $\cap_{i=1}^{t}(H_i - I_H) = \emptyset$. When both subset and completeness conditions are satisfied, the requester verifies the correctness of the result.

B. Simulation-based Analysis

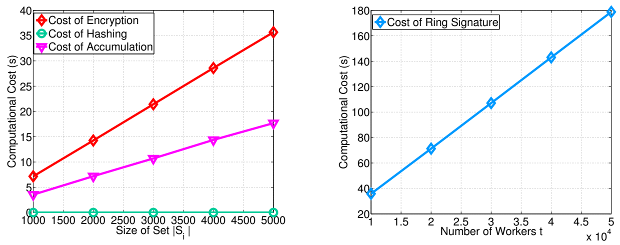
1) Simulation Setup: We simulate our protocol based on a cryptographic library: Paring-Based Cryptography (PBC) [40]. In particular, in *PBC*, we use the Type A elliptic curve. The program is written in C and implemented in CentOS 6.7 with GCC version 4.4.7. The desktop has 4.0 GHz Intel(R) Core(TM) i7-4790K CPU and 32 GB memory. The number of workers is from 10^4 to 5×10^4, and size of data set is from 1000 to 5000. We assume the size of intersection set ranges from 50 to 250. Simulation parameters are also listed in Table. II.

2) Simulation Results: We present the simulation results of computational costs at worker and at requester in the following, together with the performance of the batch verification and data update method. All computational costs are time consumption of CPU. We use random values from G as set elements throughout our simulation.

Computational Costs at Worker: The computational costs of encryption, hashing and accumulation at the worker are given in Fig. 3(a). For worker W_i the cost of encryption increases with the size of data set S_i. When $|S_i|$ is 1000, the computation cost is 7.1s and when $|S_i|$ increases to 5000, the cost increases to 35.6s. The cost of computing accumulation value is also linearly increasing with size of data set. When size of data set is 1000, the cost is 3.5s and when the size is 5000 the cost is 17.7s. The cost of hashing is quite efficient. When $|S_i|$ is 1000 and 5000, the costs are 1.19ms and 6.13ms

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/JIOT.2016.2585592, IEEE Internet of Things Journal

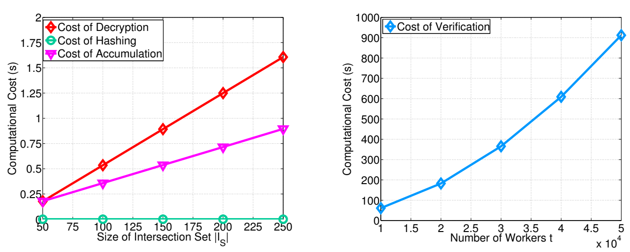
respectively. The computational cost of ring signature at each worker is given in Fig. 3(b), and it is decided by the total number of workers. When there are 5×10^4 workers, the cost of ring signature is 178s. There is a tradeoff between protecting identity privacy and computational cost. We will try to find more efficient ring signature scheme in the future work.



(a) Enc., Hashing and Accumulation (b) Ring Signature

Fig. 3. Computational Costs at Worker

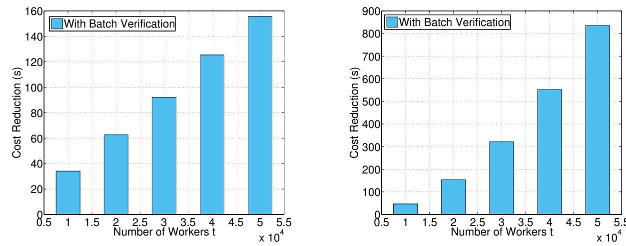
Computational Costs at Requester: Since most of the computation is delegated to the cloud, the requester only needs to deal with the cost related to the intersection set and verification. First, the requester needs to decrypt the ciphertext of intersection set. As we can see from Fig. 4(a), the cost of decryption is linearly increasing with the size of intersection I_S. The cost is 1.61s when the size of intersection is 250. Then, the requester computes the hash values for the newly derived intersection set. The cost of hashing is 2.84×10^{-4}s when $|I_S|$ is 250. The cost of accumulation increases with $|I_S|$ as well. When $|I_S|$ is 50 the cost of accumulation is 0.18s, and increases to 0.89s when $|I_S|$ is 250. The cost of verification at the requester is very high, as shown in Fig. 4(b). It is proportional to $|S_i| \times t$. When the number of workers is 5×10^4, the verification cost is 912s. We will apply batch verification to reduce verification cost in the next part.



(a) Dec., Hashing and Accumulation (b) Verification

Fig. 4. Computational Costs at Requester

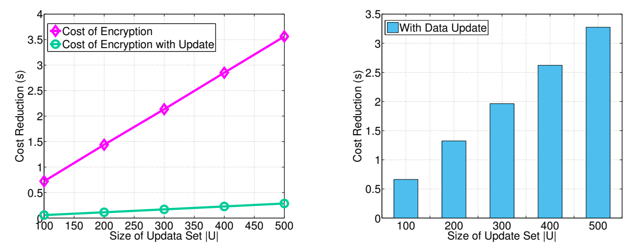
Batch Verification: To make verification more efficient, we propose batch verification. When there are 10^4 workers and each worker has 1000 elements in the data set, the data volume is at least 10^8 Bytes. The computational cost is high even for the cloud. We show the cost reduction at the cloud when batch verification is used in Fig. 5(a). The cost reduction at cloud is 34.1s when there are 10^4 workers, and 155.8s when there are 5×10^4 workers. The cost reduction at the requester is also very obvious, as shown in Fig. 5(b). When there are 5×10^4 workers, the cost reduction for verification can be 840s, which is a great improvement compared with original cost of 912s.



(a) At the Cloud (b) At the Requester

Fig. 5. Cost Reduction with Batch Verification

Data Update: Finally, we show the benefits of our data updating scheme in Fig. 6(a) and Fig. 6(b). Costs of encryption with and without data update are given in Fig. 6(a). The cost reduction is shown in Fig. 6(b). When there are 100 elements to be updated, a worker can save 0.6s computational cost. When there are 500 elements to be updated, a worker can save 3.2s computational cost. For mobile workers, less computational cost means longer battery usage, which is very crucial for smartphone users.



(a) Costs of Encryption (b) Cost Reduction

Fig. 6. Performance of Data Update

C. Experiment-based Analysis

We implement our scheme in real mobile-cloud system. In specific, we use HTC Nexus 9 as the requester and the workers. Nexus 9 has Android 5.0 Lollipop operating system, NVIDIA Tegra K1 2.3GHz x64 processor, 16GB flash memory and 2GB RAM. We use Amazon EC2 instance of type m4.10xlarge, 40 vCPU, 160 GB memory, as the cloud. All codes are written in Java for the experiment-based analysis. In this experiment, we set $K_{min} = 10$, $K_{max} = 50$, the size of a data set $|S|$ is from 1 to 10, and the total number of workers t is still from 10000 to 50000. These settings can better reflect real-life applications.

Workers: We first test the cost of encryption, hashing and accumulation on the mobile end. When the size of a worker's data set $|S|$ is from 1 to 10, these costs are given in Table. III. As we can see from Table. III, the costs of hashing and accumulation are quite low, while the cost of encryption is

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/JIOT.2016.2585592, IEEE Internet of Things Journal

TABLE III
COST OF ENCRYPTION, HASHING, AND ACCUMULATION AT WORKER

$\|S\|$	1	2	3	4	5	6	7	8	9	10
Enc. (ms)	46.8	91.09	180.06	198.84	240.85	364.56	448.02	553.88	574.62	690
Hashing (ms)	0.07	0.11	0.13	0.12	0.19	0.18	0.24	0.22	0.26	0.24
Acc. (ms)	22.97	22.68	26.71	24.16	38.19	23.33	32.78	26.75	22.46	44.86

not. There is a tradeoff between the user experience and data privacy protection. Then, we test the cost of ring signature at a worker, and the cost of ring signature is given in Table. IV.

TABLE IV
COST OF RING SIGNATURE AT WORKER

$\|L\|$	10	20	30	40	50
`Sig.` (ms)	653.42	1312.17	2057.87	2724.45	3523.04

Cloud: When cloud receives data from workers, it verifies the ring signatures first. We show the costs of signature verification with and without using batch in Table.V. The number of workers t ranges from 10000 to 50000. As we can see from Table.V, the verification cost is reduced a lot after using batch verification. The sum cost of set operation and proof generation at cloud is given in Table. VI, where we set the size of all data sets as 10.

TABLE V
COST OF SIGNATURE VERIFICATION AT CLOUD

t	10000	20000	30000	40000	50000
Without batch (s)	40.03	72.35	138.69	383.87	736.72
With batch (s)	22.25	61.62	116.40	361.05	696.73

TABLE VI
COST OF SET OPERATION AND PROOF GENERATION AT CLOUD

t	10000	20000	30000	40000	50000
Set operation + Proof (s)	2.83	4.10	5.62	8.81	14.55

Requester: In our implementation, the resource-limited requester is also represented by HTC Nexus 9. Since set operation is delegated to the cloud, requester needs to decrypt the received ciphertext set and verify its correctness. Since in the experiment, a worker's data set size is from 1 to 10, we assume theThe cost of decryption is given in Table. VII. The cost of correctness verification is given in Table. VIII. The cost of verification increases with the number of workers. After using batch verification, more than half of the cost is reduced, which is very important for battery-limited devices.

TABLE VII
COST OF DECRYPTION AT REQUESTER

$\|I_S\|$	1	2	3	4	5	6	7	8	9	10
Dec. (ms)	24.37	53.07	71.58	94.40	116.17	156.49	223.76	262.04	291.82	339.42

TABLE VIII
COST OF CORRECTNESS VERIFICATION AT REQUESTER

t	10000	20000	30000	40000	50000
Without batch (s)	72.03	169.70	260.91	372.72	414.73
With batch (s)	25.55	53.20	92.07	155.99	159.91

VII. CONCLUSION

In this paper, we propose a scheme to enable the requester to delegate set operations over crowdsourced big data to the cloud. Meanwhile, worker's data and identity privacy are preserved, and the requester can verify the correctness of the set operation result. We extend our scheme to achieve data preprocessing, batch verification and data update are also proposed to reduce computational costs of the system.

REFERENCES

[1] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, vol. 2. IEEE, 2011, pp. 3–6.
[2] Q. Li and G. Cao, "Privacy-preserving participatory sensing."
[3] ——, "Efficient and privacy-preserving data aggregation in mobile sensing," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, Oct 2012, pp. 1–10.
[4] Q. Li, G. Cao, and T. La Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 11, no. 2, pp. 115–129, March 2014.
[5] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: privacy-aware people-centric sensing," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 211–224.
[6] R. Zhang, J. Shi, Y. Zhang, and C. Zhang, "Verifiable privacy-preserving aggregation in people-centric urban sensing systems," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 9, pp. 268–278, September 2013.
[7] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," in *Distributed computing and internet technology*. Springer, 2013, pp. 19–26.
[8] H. Yue, L. Guo, R. Li, H. Asaeda, and Y. Fang, "Dataclouds: Enabling community-based data-centric services over the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 472–482, Oct 2014.
[9] K. Hara, S. Azenkot, M. Campbell, C. L. Bennett, V. Le, S. Pannella, R. Moore, K. Minckler, R. H. Ng, and J. E. Froehlich, "Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 6, no. 2, p. 5, 2015.
[10] B. Liu, Y. Jiang, F. Sha, and R. Govindan, "Cloud-enabled privacy-preserving collaborative learning for mobile sensing," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM, 2012, pp. 57–70.

[11] G. Zhuo, Q. Jia, L. Guo, M. Li, and Y. Fang, "Privacy-preserving verifiable proximity test for location-based services," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[12] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, "Privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing," in *INFOCOM, 2016 Proceedings IEEE*. IEEE, 2016.

[13] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: when qoe meets qop," *Wireless Communications, IEEE*, vol. 22, no. 4, pp. 74–80, 2015.

[14] H. Li, Y. Yang, T. Luan, X. Liang, L. Zhou, and X. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[15] X. Chen, X. Wu, X.-Y. Li, Y. He, and Y. Liu, "Privacy-preserving high-quality map generation with participatory sensing," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 2310–2318.

[16] L. Guo, C. Zhang, and Y. Fang, "Privacy-preserving revocable content sharing in geosocial networks," in *Communications and Network Security (CNS), 2013 IEEE Conference on*, Oct 2013, pp. 118–126.

[17] L. Guo, C. Zhang, J. Sun, and Y. Fang, "Paas: A privacy-preserving attribute-based authentication system for ehealth networks," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 224–233.

[18] ——, "A privacy-preserving attribute-based authentication system for mobile health networks," *Mobile Computing, IEEE Transactions on*, vol. 13, no. 9, pp. 1927–1941, 2014.

[19] L. Guo, X. Zhu, C. Zhang, and Y. Fang, "Privacy-preserving attribute-based friend search in geosocial networks with untrusted servers," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 629–634.

[20] J. Shao, R. Lu, and X. Lin, "Fine: A fine-grained privacy-preserving location-based service framework for mobile devices," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 244–252.

[21] R. Lu, X. Lin, Z. Shi, and J. Shao, "Plam: A privacy-preserving framework for local-area mobile social networks," in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 763–771.

[22] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology-EUROCRYPT 2004*. Springer, 2004, pp. 1–19.

[23] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in Cryptology–CRYPTO 2005*. Springer, 2005, pp. 241–257.

[24] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security*. Springer, 2010, pp. 143–159.

[25] E. De Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," in *Advances in Cryptology-ASIACRYPT 2010*. Springer, 2010, pp. 213–231.

[26] E. De Cristofaro and G. Tsudik, *Experimenting with fast private set intersection*. Springer, 2012.

[27] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 789–800.

[28] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012, pp. 85–86.

[29] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Advances in Cryptology–CRYPTO 2010*. Springer, 2010, pp. 465–482.

[30] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 111–131.

[31] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 501–512.

[32] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Optimal verification of operations on dynamic sets," in *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 91–110.

[33] R. Canetti, O. Paneth, D. Papadopoulos, and N. Triandopoulos, "Verifiable set operations over outsourced databases," in *Public-Key Cryptography–PKC 2014*. Springer, 2014, pp. 113–130.

[34] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 844–855.

[35] A. Abadi, S. Terzis, and C. Dong, "O-psi: delegated private set intersection on outsourced datasets," in *ICT Systems Security and Privacy Protection*. Springer, 2015, pp. 3–17.

[36] L. Guo, Y. Fang, M. Li, and P. Li, "Verifiable privacy-preserving monitoring for cloud-assisted mhealth systems," in *INFOCOM, 2015 Proceedings IEEE*. IEEE, 2015.

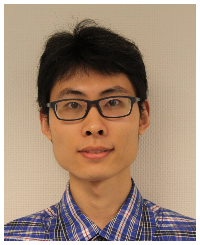
[37] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in CryptologyCRYPTO 2001*. Springer, 2001, pp. 213–229.

[38] E.-J. Goh, *Encryption schemes from bilinear maps*. Stanford University, 2007.

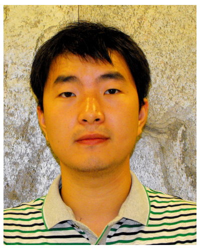
[39] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in cryptology*. Springer, 1984, pp. 10–18.

[40] L. Ben, "The pairing-based cryptography, https://crypto.stanford.edu/pbc/," access June 20, 2015.

[41] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 295–302.

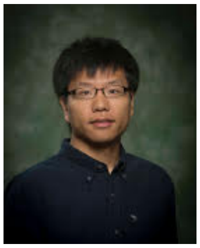


Gaoqiang Zhuo received his B.E. degree in Communication Engineering from Beijing University of Posts and Telecommunications in 2012 and M.S. degree in Information and Communication Technology from University of Agder in 2014. He is currently working towards his Ph.D. degree in Binghamton University. His research interests include security and privacy issues in location-based service and mobile crowdsourcing. He is a recipient of Best Paper Award of Globecom 2015, Symposium on Communication and Information System Security. He is a student member of the IEEE.



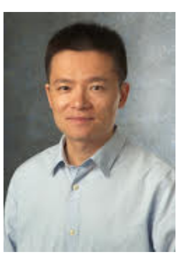
Qi Jia received his B.E. degree in Communication Engineering from Beihang University (Beijing University of Aeronautics and Astronautics) in 2014. He is currently working towards his PhD. degree in Binghamton University. His research interests include security and privacy issues in machine learning and data mining. He is a co-recipient of Best Paper Award of Globecom 2015, Symposium on Communication and Information System Security. He is a student member of the IEEE.

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/JIOT.2016.2585592, IEEE Internet of Things Journal



Linke Guo received the BE degree in electronic information science and technology from the Beijing University of Posts and Telecommunications in 2008. He received the MS and PhD degrees in electrical and computer engineering from the University of Florida in 2011 and 2014, respectively. Since August 2014, he has been an assistant professor in the Department of Electrical and Computer Engineering, Binghamton University, State University of New York. His research interests include network security and privacy, social networks, and applied cryptography. He serves as the publication chair of IEEE Conference on Communications and Network Security (CNS) 2016. He was the symposium co-chair of Network Algorithms and Performance Evaluation Symposium, ICNC 2016. He has served as the Technical Program Committee (TPC) members for several conferences including IEEE INFOCOM, ICC, GLOBECOM, and WCNC. He is the co-recipient of Best Paper Award of Globecom 2015, Symposium on Communication and Information System Security. He is a member of the IEEE and ACM.

Ming Li received the B.E. degree in Electrical Engineering from Sun Yatsen University, China, in 2007, the M.E. degree in Electrical Engineering from Beijing University of Posts and Communications, China, in 2010, and the Ph.D. degree in Electrical and Computer Engineering from Mississippi State University, Starkville, in 2014, respectively. She is currently an Assistant Professor in the Department of Computer Science and Engineering, University of Nevada, Reno. Her research interests include cybersecurity, privacy-preserving data analysis, resource management and network optimization in cyber-physical systems, cloud computing, mobile computing, wireless networks, smart grid, and big data.



Pan Li received the B.E. degree in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in Electrical and Computer Engineering from University of Florida, Gainesville, in 2009, respectively. Since Fall 2015, he has been with the Department of Electrical Engineering and Computer Science at Case Western Reserve University. He was an Assistant Professor in the Department of Electrical and Computer Engineering at Mississippi State University between August 2009 and August 2015. His research interests include network science and economics, energy systems, security and privacy, and big data. He has been serving as an Editor for IEEE Journal on Selected Areas in Communications – Cognitive Radio Series and IEEE Communications Surveys and Tutorials, a Feature Editor for IEEE Wireless Communications, and a Technical Program Committee (TPC) Co-Chair for Ad-hoc, Mesh, Machine-to-Machine and Sensor Networks Track, IEE VTC 2014, Physical Layer Track, Wireless Communications Symposium, WTS 2014, and Wireless Networking Symposium, IEEE ICC 2013. He received the NSF CAREER Award in 2012 and is a member of the IEEE and the ACM.