Privacy-preserving Verifiable Data Aggregation and Analysis 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*—Crowdsourcing is a crowd-based outsourcing, where a requester (task owner) can outsource tasks to workers (public crowd). Recently, mobile crowdsourcing, which can leverage workers' data from smartphones for data aggregation and analysis, has attracted much attention. However, when the data volume is getting large, it becomes a difficult problem for a requester to aggregate and analyze the incoming data, especially when the requester is an ordinary smartphone user or a start-up company with limited storage and computation resources. Besides, workers are concerned about their identity and data privacy. To tackle these issues, we introduce a three-party architecture for mobile crowdsourcing, where the cloud is implemented between workers and requesters to ease the storage and computation burden of the resource-limited requester. Identity privacy and data privacy are also achieved. With our scheme, a requester is able to verify the correctness of computation results from the cloud. We also provide several aggregated statistics in our work, together with efficient data update methods. Extensive simulation shows both the feasibility and efficiency of our proposed solution.**

***Index Terms*—Mobile Crowdsourcing, Cloud Computing, Security and Privacy, Verifiable Computation.**

I. Introduction

The pervasiveness of smartphones revolutionarily changes the traditional crowdsourcing to mobile crowdsourcing, where workers can perform tasks more freely for a requester by using their smartphones [1]–[7]. Nowadays, smartphones often come with a rich set of embedded sensors such as GPS, accelerator, gyroscope, digital compass, light, audio, and video sensors. More and more cheap sensors measuring temperature, humidity, barometer, chemical, and biomedical are also expected to be incorporated into smartphones. This indicates that smartphones are increasingly capable of sensing surrounding environments and providing all kinds of data desired by a requester. For example, a healthcare company may want to collect users' health-related information such as weights, blood glucose levels, heart rates with biomedical sensors for medical use. The transportation agencies can harness the collected sensed data for managing, scheduling, and maintaining urban transportation systems. The data may not only be the sensed data generated by smartphone sensors like in [1]–[6], it may also come from other sources, e.g., a worker's personal knowledge. Crowdsourcing data via smartphones brings a bunch of benefits to both requesters and workers. First of all, the ubiquitous penetration of smartphones into daily life implies sufficient geographic coverage and data diversity, which makes the data aggregation results more accurate. Second, it costs much less time and expense of the requester, because the requester does not need to dispatch data collector to go through well-planed routes. Also, the universal access of the Internet and cellular networks enables nearly real-time data collection. Besides, workers can also get many rewards (e.g., revenue and reputation) through participation.

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

No matter how promising the mobile crowdsourcing is, it will not be widely accepted unless the following issues are well addressed. First, the requester often needs to aggregate and analyze a huge amount of data samples from workers, which involves intensive storage, communication, and computation cost [8]. If the requester is an ordinary smartphone user or a start-up company, that cost is not affordable when the involved data volume is increasingly large [6], [9]. Second, a worker's data might contain private information like identity, age, weight, location, and so on. Revealing this information to the requester or other workers could incur serious privacy breach or even physical attacks [2], [5], [10]–[13]. As a result, workers might be reluctant to participate due to privacy concern.

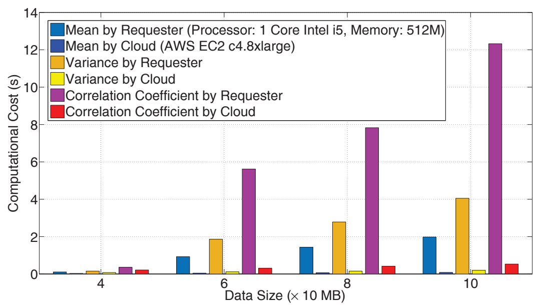


Fig. 1. Computation Comparison between the Requester and Cloud

Facing these challenges, we propose to take advantages the cloud to help the requester to aggregate, store, and process the collected data, while the requester only needs to send a request to the cloud and retrieve the corresponding results. Here, we use a simple example to show the benefits brought by using cloud. Given a task of computing mean, variance, and correlation coefficient over large integer datasets (all in plaintext), we show in Fig.1 the advantage of cloud over an ordinary resource-limited requester. As we can see from Fig.1, the advantage of cloud becomes larger as the number of data increases, and this is especially meaningful in big data aggregation and processing.

Besides, the involvement of cloud cuts down the direct data communication between requester and workers. With proper encryption schemes, workers' data privacy can be well protected from the requester. However, the introduction of the cloud brings new challenges. First, current cloud computing platforms cannot protect users' data privacy well [10], [14], [15]. Second, the cloud might encounter hardware/software errors and internal/external attacks, which may lead to incorrect computation. Therefore, how to verify the correctness of the computation results is a question. To address the above issues, we propose a privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing. In our scheme, the requester is able to delegate the data aggregation and analysis to the cloud, and verify the correctness of the retrieved results. Meanwhile, a worker's identity and data will not be revealed.

Related Works:

Privacy-Preserving Data Aggregation: Privacy-preserving data aggregation is useful in many areas such as mobile sensing [3]–[6], [16], eHealth [9], and smart grid [17]. In [3]–[5], [16], [17], different encryption schemes are used to achieve data privacy, but all data aggregation schemes are accomplished by the requester. An additive homomorphic encryption scheme is used to achieve Sum aggregate and Min aggregate of time-series data in [3]–[5]. The requester needs to store all the ciphertexts from the users and compute the Sum and Min by himself, which is obviously not applicable when the requester is constrained in storage and computation capability. Shi *et al.* [16] propose a privacy-preserving sum aggregation, where the decryption in their scheme requires brute-force to solve the DLP, which brings much computation burden to the requester. The collector in [17] can only obtain the summation of the data which is not enough in our proposed scenario. The in-network data aggregation will incur much delay in mobile crowdsourcing, because one worker's data needs to be routed by some other workers. In [6], [9], the aggregation is done by the cloud. A peer-to-peer based privacy-preserving data aggregation scheme is proposed for people-centric urban sensing in [6]. However, there is no mechanism for the requester to verify the correctness of the results retrieved from the service provider. Zhou *et al.* [9] propose a homomorphic data aggregation scheme in eHealth systems. In this scheme, the users' data privacy can be well protected and the computation of the aggregation statistics is delegated to the cloud. Unfortunately, in [9] they cannot achieve the verifiable computation which is required in our scenario. Our proposed solution cannot only protect data privacy but also achieve delegated computation and storage. The comparison between our solution and three other typical ones are given in Table. I.

Verifiable Computation: Gennaro *et al.* [18] introduce and formalize the concept of verifiable computation, which enables a resource-constrained client to outsource the computation of a function to one or more workers. The client should be able to efficiently verify the correctness of the results. In [19]–[21], verifiable computation over plaintext space is achieved. In [19], the first practical verifiable computation scheme for high degree polynomial functions is proposed. Dario *et al.* [20] propose a publicly verifiable computation of large polynomials and matrix computations, where anyone can verify the correctness of the results. Papamanthou *et al.* [21] propose a verifiable delegated computation of set operations, such as set union, set intersection and set difference. However, all these verifiable computation are over the plaintext space, which is not suitable for our scheme. Verifiable computation in ciphertext space is realized in [22]–[24]. Guo *et al.* [22] design a verifiable computation over the ciphertext space for mHealh systems, where a patient can ask the cloud to evaluate a polynomial over his encrypted personal health record. In [23], the accumulation tree is novelly used to verify the results of proximity test. Fiore *et al.* [24] achieve efficiently verifiable computation on encrypted data. All these solutions can only enable the workers to verify the results which cannot be used directly in our scheme. Based on [24], we extend their architecture from two-party model (user and cloud) to three-party model (worker, cloud and requester) so that we can achieve privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing.

TABLE I
OUR SOLUTION VS. TYPICAL EXISTING SOLUTIONS

Schemes / Functionality	Ours	[5]	[6]	[16]
Data Privacy	✓	✓	✓	✓
Storage Delagation	✓	×	×	✓
Computation Delegation	✓	×	×	✓
Correctness Verification	✓	×	×	×
Sum/Mean	✓	✓	✓	✓
Variance/Correlation Coefficient	✓	×	×	✓

Our Contributions: We list our contributions as follows.

- The proposed architecture achieves privacy-preserving verifiable data aggregation and analysis for mobile crowdsourcing.
- With our scheme, each worker's identity privacy and data privacy are well protected. The cloud will not learn the exact identity of the workers or the content of aggregation results, and requester can only learn the final aggregation results.
- The requester is enabled to verify the correctness of computation results retrieved from the cloud.
- A variety of statistics, i.e., sum, mean, variance, and uncentered correlation coefficient are computed with our scheme and an efficient data update method is provided.

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 the protocol evaluation in Section V. Finally, Section VI concludes the paper.

II. Preliminaries and Problem Formulation

A. Preliminaries

1) Bilinear Pairing: A bilinear pairing is a map $e : G_1 \times G_2 \rightarrow G_T$, where G_1, G_2 and G_T are multiplicative cyclic groups of the same prime order q and G_1 is generated by g, G_2 is generated by h. The pairing e has the following properties [25], [26]:

- *Bilinearity:* $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1 \in G_1$, $g_2 \in G_2$ and random numbers $a, b \in Z_q^*$;

- *Computability:* For all $g_1 \in G_1$, $g_2 \in G_2$, $e(g_1, g_2)$ can be computed efficiently;
- *Non-degeneracy:* For $g \in G_1$, $h \in G_2$, $e(g, h) \neq 1$.

2) Cryptographic Assumptions:

- *Discrete Logarithmic Problem (DLP) [27]:* Let g_0, g_1 be two elements in G_1. It is computationally intractable to find an integer a, such that $g_1 = g_0^a$.
- *Computational Diffie-Hellman (CDH) Problem [25]:* Given (g_1, g_1^a, g_1^b) for $g_1 \in G_1$ and unknown $a, b \in Z_q^*$, it is intractable to compute g_1^{ab} in a polynomial time.
- *Decisional Diffie-Hellman (DDH) Problem [25]:* Given $(g_1, g_1^a, g_1^b, g_1^c)$ for $g_1 \in G_1$ and unknown $a, b, c \in Z_q^*$, it is easy to tell whether $c = ab \mod q$ by checking if $e(g_1^a, g_1^b) = e(g_1^c, g)$.
- *l-BDHI Assumption [24]:* Let $\mathcal{G}$ be a bilinear group generator, and $\mathcal{G}(1^\lambda) \to (q, G_1, G_2, G_T, e, g, h)$. Let $z \in Z_q$ be chosen uniformly at random. We say that the l-BDHI assumption holds for $\mathcal{G}$ if for every probabilistic polynomial time adversary $\mathcal{A}$ and any l = poly(λ) the probability $Pr[\mathcal{A}(q, G_1, G_2, G_T, e, g, h, g^z, h^z, ..., g^{z^l}, h^{z^l}) = e(g, h)^{1/z}]$ is negligible.

3) BGV Homomorphic Encryption: BGV homomorphic encryption is built upon the hardness of Learning with Errors (*LWE*) problem, and can fully support both additive and multiplicative homomorphisms [24], [28], [29]. In our scheme, we only use the somewhat homomorphic of [29], which includes four algorithms: $\texttt{ParamGen}$, $\texttt{KeyGen}$, $\texttt{Enc}_{\texttt{pk}}$ and $\texttt{Dec}_{\texttt{dk}}$. The details are given below [24].

- $\texttt{ParamGen}$: Given the security parameter λ, the algorithm generates the message space $R_p = Z_p[X]/\Phi_m(X)$ and the ciphertext space $R_q = Z/qZ[X]/\Phi_m(X)$, where p is a prime, and q is the same as above. $\Phi_m(X)$ is the mth cyclotomic polynomial in $Z_p[X]$ of degree n. Then, the algorithm defines two probability distributions, $D_{Z^n,\sigma}$ and ZO_n. The $D_{Z^n,\sigma}$ is a discrete Gaussian distribution with parameter σ over Z^n. The ZO_n is the distribution of random variable $x = (x_1, x_2, ..., x_n)$ with $x_i \in \{-1, 0, 1\}$ and $Pr[x_i = -1] = Pr[x_i = 1] = 1/4$ and $Pr[x_i = 0] = 1/2$.
- $\texttt{KeyGen}$: This algorithm chooses $a \in_R R_q$ and $s, e \in_R D_{Z^n,\sigma}$. Consider s and e as elements in R_q, and compute $b = a \cdot s + p \cdot e$, and set public key: $pk = (a, b)$ and decryption key: $dk = s$.
- $\texttt{Enc}_{\texttt{pk}}$: Given pk, $m \in R_p$, and $r \in_R (ZO_n, ZO_n, D_{Z^n,\sigma})$, the message m is parsed as an element in R_q with infinity norm bounded by $p/2$ and the random r is parsed as $r = (r_1, r_2, r_3) \in R_q^3$. The output is $c = c_0 + c_1 \cdot Y \in R_q[Y]$, where $c_0 = b \cdot r_2 + p \cdot r_3 + m$ and $c_1 = a \cdot r_2 + p \cdot r_1$.
- $\texttt{Dec}_{\texttt{dk}}$: This algorithm takes as input $c = c_0 + c_1 \cdot Y$ and outputs $m = c_0 - c_1 s$ mod p. The correctness is as follows:

$$\begin{aligned}
&c_0 - c_1 s \\
&= (br_2 + pr_3 + m) - (ar_2 + pr_1)s \\
&= (as + pe)r_2 + pr_3 + m - ar_2 s - pr_1 s \\
&= p(er_2 + r_3 + r_1 s) + m \\
&= m \mod p
\end{aligned}$$

4) Homomorphic Hash Functions [24]: Define $H() : R_q[Y] \to G_1 \times G_2$ (or G_T) as a collision-resistant homomorphic hash. If the input is $\mu = \sum_{j=0}^{2} \mu_j Y^j \in R_q[Y]$, the algorithm computes $H_{\alpha,\beta}(\mu)$ as follows,

$$H_{\alpha,\beta}(\mu) = \sum_{j=0}^{2} \sum_{i=0}^{n-1} (\mu_j \alpha^j)_i \beta^i$$

where $\alpha \in_R R_q$ and $\beta \in_R Z_q$, and $H_{\alpha,\beta}()$ is a one-way homomorphic hash function. We use $deg_Y(\mu)$ to denote the degree of μ with respect to Y. For example, if $\mu = c_0 + c_1 Y$, then $deg_Y(\mu) = 1$. $H()$ has two possible outputs: if $deg_Y(\mu) < 2$, the algorithm outputs

$$H(\mu) = (T, U) = (g^{H_{\alpha,\beta}(\mu)}, h^{H_{\alpha,\beta}(\mu)}).$$

If $deg_Y(\mu) = 2$, the algorithm outputs

$$H(\mu) = e(g, h)^{H_{\alpha,\beta}(\mu)}.$$

5) Pseudorandom Functions [19], [24], [30]: $F_K() : \{0, 1\}^* \times \{0, 1\}^* \to G_1 \times G_2$ is a pseudorandom function with key $K = (K_1, K_2)$, and it consists of two other pseudorandom functions: $F_{K_1} : \{0, 1\}^* \to Z_q^2$ and $F_{K_2} : \{0, 1\}^* \to Z_q^2$. Given input tuple (I_1, I_2), $F_{K_1}(I_1) = (u_{I_1}, v_{I_1})$ and $F_{K_2}(I_2) = (u_{I_2}, v_{I_2})$, and $F_K(I_1, I_2) = (R, S) = (g^{u_{I_1} u_{I_2} + v_{I_1} v_{I_2}}, h^{u_{I_1} u_{I_2} + v_{I_1} v_{I_2}})$.

B. Problem Formulation

The fundamental problem is how to enable a requester to achieve data aggregation and analysis with the help of cloud in a privacy-preserving and verifiable way. Here, privacy preservation refers to workers' identity and data privacy, as well as the result privacy. Verifiability refers to requester's ability to verify the correctness of the results. Specifically, identity privacy is achieved by ring signature. *BGV* homomorphic encryption is used to protect workers' data privacy. Given a ciphertext $c = c_0 + c_1 \cdot Y$ encrypted with *BGV* homomorphic scheme, it is intractable to find out the plaintext without the decryption key. Data aggregation and analysis also rely on *BGV* homomorphic encryption due to its homomorphism. Given $H(\mu_1) = (T_1, U_1)$ and $H(\mu_2) = (T_2, U_2)$, and a constant $c \in Z_q$, the following homomorphic properties hold: $H(\mu_1 + \mu_2) = (T_1 \cdot T_2, U_1 \cdot U_2), H(c \cdot \mu) = (T^c, U^c)$, and $H(\mu_1 \times \mu_2) = e(T_1, U_2)$. With the above homomorphic properties, the cloud can do computations over the ciphertexts. Verifiability can be achieved by using the homomorphic hash functions. Due to the one-way and collision-resistant properties of the homomorphic hash functions, a requester can verify the correctness of cloud's computation results efficiently.

III. System Model

A. System Model

Our model mainly consists of four entities, the mobile workers (*MW*), the requester (*R*), the cloud (*C*), and the trusted authority (*TA*), as shown in Fig.2.

- *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* will be offline unless a dispute arises.

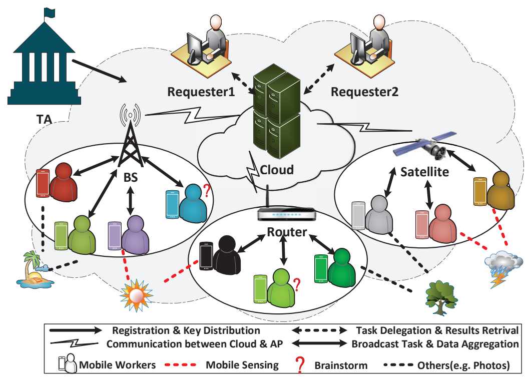

TA
Requester1
Requester2
BS
Cloud
Satellite
Router
Registration & Key Distribution
Communication between Cloud & AP
Task Delegation & Results Retrival
Broadcast Task & Data Aggregation
Mobile Workers
Mobile Sensing
Brainstorm
Others(e.g. Photos)

Fig. 2. System Architecture

- *Requester:* The requester wants to obtain aggregation statistics over the workers' data. However, due to his/her limitation on the storage and computation capability, the requester will delegate most of the computation to the cloud.
- *Cloud:* The cloud receives the delegation requests from the requester and the encrypted data from mobile workers, then it computes the results for the requester. The cloud is associated with some access points (*AP*s), which act as relays between cloud and the workers or requesters.
- *Mobile Workers:* Mobile workers refer to those who have smartphones and are willing to contribute data to the requester's tasks. *MW*s may move randomly and send the sensing data to the cloud after encryption.

B. Security Model

In our security model, the *TA* is fully trusted and will not be breached by any adversary. We assume that workers, requester and cloud are *honest-but-curious*, meaning that they will strictly follow the predefined protocol but may also dig out others' privacy based on available information.

- *Mobile Workers:* In our scheme, a worker's data should be kept confidential from others.
- *Requester:* The requester should be able to verify the correctness of the computation results received from the cloud. However, the requester cannot learn any individual worker's plaintext data.
- *Cloud:* The cloud will honestly compute the aggregation according to the requester's ask, and will provide the proof of correctness of the computation results to meet the security requirement. Besides, the cloud should only obtain the encrypted data from the workers and the encrypted aggregation results.

In addition, the collusion attacks among any entities are out of the scope of our paper.

C. Design Objectives

We have three main objectives for our privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing. First, a requester can securely delegate the computation of the aggregation statistics to the cloud so as to offload the burden of storage and computation. Second, workers can participate in the mobile crowdsourcing task without leaking identity and data privacy to anyone else. Finally, the requester can verify the correctness of the computation results from cloud in an efficient way.

IV. Our Proposed Scheme

In this section, we introduce our privacy-preserving verifiable data aggregation and analysis for cloud-assisted mobile crowdsourcing in detail. It mainly consists of the following subsections: system initialization, basic scheme, and extensions. Before diving into the details, we give a brief overview of our proposed scheme.

A. Overview

During the system initialization, *TA* assigns a public/private key pair to the requester, cloud and all workers. When the requester wants to obtain the statistics of a specific type of data, he sends his request and his homomorphic encryption key *pk* to the cloud and waits for the results. The cloud broadcasts the requester's task and homomorphic encryption key *pk* to all chosen workers asking for their data. The chosen workers prepare the required data m from embedded smartphone sensors or other data sources, and then encrypt m with requester's encryption key *pk*. Workers sign their data before sending it to cloud. After all chosen workers send their encrypted data $\mathtt{Enc}_{\mathtt{pk}}(m)$ and signature to the cloud via *AP*s. The cloud verifies the authenticity of each data, and then it computes the statistics over the ciphertexts and returns the results with corresponding proof information to the requester. Finally, if the requester successfully verifies the correctness of the results he/she continues to decrypt the results. Otherwise, the requester rejects the results.

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 initialization steps as follows.

1) General Setup: Given the security parameter κ and λ, the *TA* generates the bilinear parameters $(q, G_1, G_2, G_T, e, g, h)$ and the public homomorphic encryption parameters (p, q, R_p, R_q) respectively. Then *TA* chooses a secure symmetric encryption algorithm $E()$, *e.g.*, *AES*, and a hash function: $H_1 : R_q[Y] \rightarrow Z_q$. Besides, TA chooses a master key $mk = \epsilon \in Z_q^*$ and computes a public key $P_{mk} = h^{\epsilon}$. Finally, *TA* keeps the master key and publishes the public parameters $pps = \{q, G_1, G_2, G_T, e, g, h, p, R_p, R_q, P_{mk}, H_1, E()\}$.

2) Entities Registration: Assume there are N mobile workers in the system: $MW = \{W_1, W_2, ..., W_N\}$. For each worker W_i, *TA* assigns him a private/public key pair (sk_i, pk_i), where $sk_i = x_i \in_R Z_q$ and $pk_i = h^{x_i}$. *TA* registers the cloud and the requester by sending the private/public key pairs $(sk_c, pk_c) = (x_c, h^{x_c})$ and $(sk_r, pk_r) = (x_r, h^{x^r})$ to the cloud and the requester respectively, where x_c and x_r are random number from Z_q. Besides, both requester and workers obtain encryption keys (α, β) for the homomorphic hash and $K = (K_1, K_2)$ for pesudorandom function $F_K()$ during this phase.

C. Basic Scheme

In this subsection, we introduce our basic scheme which can compute the sum of workers' data in a privacy-preserving and verifiable way. There are five steps.

Step 1: Task Generation

First, the requester generates the *BGV* homomorphic encryption key $pk = (a, b)$ and the decryption key $dk = s$ using the KeyGen algorithm described in the preliminaries. Then, the requester sends $\{\tau = \text{"}\textit{sum}\text{"}, t, pk\}$ to the cloud as an request, where $\tau = \text{"}\textit{sum}\text{"}$ is tag of the statistics to be computed, and $t \in Z$ is the total number of workers needed. For $i = 1$ to t, the requester computes $F_{K_1}(i) = (u_i, v_i)$, $\rho_i(x, y) = u_i x + v_i y$, where x and y are variables, and $\omega_(x, y) = \sum_{i=1}^{t} \rho_i(x, y)$ as the witness of $\tau = \text{"}\textit{sum}\text{"}$. The $\omega(x, y)$ will be used later in the correctness verification step.

Step 2: Task Forwarding

After receiving the delegated task $\{\tau = \text{"}\textit{sum}\text{"}, t, pk\}$ from requester, the cloud publishes pk to all chosen workers[1]. Without loss of generality, we assume the chosen workers are $\{W_1, W_2, ..., W_t\}$.

Step 3: Worker Participation

Assuming worker W_i has been chosen to participate. He prepares ready his data m_i, and encrypts it with requester's public encryption key pk to get

$$\mu_i = \mathtt{Enc}_{\mathtt{pk}}(m_i) = c_{i0} + c_{i1} \cdot Y.$$

Then, worker W_i needs to compute the following values for verification use. First, the homomorphic hash value of μ_i is computed,

$$H(\mu_i) = (T_i, U_i) = (g^{H_{\alpha,\beta}(\mu_i)}, h^{H_{\alpha,\beta}(\mu_i)}), \tag{1}$$

Then, he computes $F_{K_1}(i) = (u_i, v_i)$, $F_{K_2}(\tau) = (u, v)$, and

$$F_K(i, \tau) = (R_i, S_i) = (g^{u_i u + v_i v}, h^{u_i u + v_i v}), \tag{2}$$

followed by

$$\begin{aligned} X_i &= (R_i \cdot T_i^{-1})^{1/d} = (g^{u_i u' + v_i v' - H_{\alpha,\beta}(\mu_i)})^{1/d}, \\ Y_i &= (S_i \cdot U_i^{-1})^{1/d} = (h^{u_i u' + v_i v' - H_{\alpha,\beta}(\mu_i)})^{1/d}. \end{aligned} \tag{3}$$

Next, worker W_i generates $\sigma_i = (T_i, U_i, X_i, Y_i, \Lambda_i = 1)$. To achieve identity privacy, W_i uses ring signature [31] to sign μ_i. Given all chosen users' public keys $(pk_1, pk_2, ..., pk_t)$, μ_i, and private key sk_i, worker W_i randomly chooses $a_{ij} \in Z_p$ for all W_j, where $j \neq i$, and computes $s_j = g^{a_{ij}}$. Also, W_i computes $\delta_i = g^{H_1(\mu_i)}$, and

$$s_i = \left(\frac{\delta_i}{\phi(\prod_{j \neq i} pk_j^{a_{ij}})} \right)^{1/sk_i}.$$

The ring signature for μ_i is $s_{W_i} = (s_{i1}, s_{i2}, ..., s_{it})$. Finally, worker W_i sends $\{\mu_i, \sigma_i, s_{W_i}\}$ to cloud.

Step 4: Data Aggregation

When cloud receives $\{\mu_i, \sigma_i, s_{W_i}\}$ from all chosen workers $W_1, W_2, ..., W_t$, it first verifies if the received data really comes from the chosen workers by computing $\delta_i = g^{H_1(\mu_i)}$, and checking

$$e(\delta_i, h) \overset{?}{=} \prod_{j=1}^{t} e(s_{ij}, pk_i).$$

If the above equation holds, μ_i is signed by one of the t chosen workers. Otherwise, it is not.

[1]Sometimes, not all workers are chosen for a specific task. For examle, requester may only choose workers from points of interest, instead of all areas. How to choose workers is out of our scope.

***Proof of correctness:*.**

$$\begin{aligned} \prod_{j=1}^{t} e(s_{ij}, pk_i) &= e(s_{ii}, pk_i) \cdot \prod_{j \neq i} e(s_{ij}, pk_j) \\ &= e(\left(\frac{\delta_i}{\phi(\prod_{j \neq i} pk_j^{a_{ij}})} \right)^{1/sk_i}, h^{x_i}) \cdot \prod_{j \neq i} e(g^{a_{ij}}, h^{x_j}) \\ &= e(\frac{\delta_i}{\phi(\prod_{j \neq i} h^{x_j a_{ij}})}, h) \cdot \prod_{j \neq i} e(g^{x_j a_{ij}}, h) \\ &= e(\frac{\delta_i}{\prod_{j \neq i} g^{x_j a_{ij}}}, h) \cdot e(\prod_{j \neq i} g^{x_j a_{ij}}, h) \\ &= e(\delta_i, h) \end{aligned}$$

□

Then, cloud continues to compute $\mu = \sum_{i=1}^{t} \mu_i$ and $\sigma = (T, U, X, Y, \Lambda)$, where

$$T = \prod_{i=1}^{t} T_i,\ U = \prod_{i=1}^{t} U_i,\ X = \prod_{i=1}^{t} X_i,\ Y = \prod_{i=1}^{t} Y_i,\ \Lambda = \prod_{i=1}^{t} \Lambda_i.$$

Finally, the cloud sends a tuple of the computation result and its verification information $\{\mu, \sigma\}$ to the requester.

Step 5: Result Retrieval and Verification

Based on the tag $\tau = \text{"}\textit{sum}\text{"}$ of the task, the requester computes $F_{K_2}(\tau) = (u, v)$, $\omega = \omega(u, v) = \sum_{i=1}^{t}(u_i u + v_i v)$ and $W = e(g, h)^{\omega}$. After requester receives (μ, σ) from the cloud, he computes $H(\mu) = (T', U')$. Then, he checks if the following equations hold or not.

$$\begin{aligned} &(T, U) \overset{?}{=} (T', U'),\ e(T, h) \overset{?}{=} e(g, U), \\ &e(X, h) \overset{?}{=} e(g, Y),\ W \overset{?}{=} e(T, h) \cdot e(X, h)^d. \end{aligned} \tag{4}$$

If any of the above equations does not hold, the requester rejects the results. Otherwise, he accepts the results. If the results are correct, those equations hold as follows,

***Proof of correctness:*.**

$$\begin{aligned} (T, U) &= (\prod_{i=1}^{t} T_i, \prod_{i=1}^{t} U_i) \\ &= (g^{\sum_{i=1}^{t} H_{\alpha,\beta}(\mu_i)}, h^{\sum_{i=1}^{t} H_{\alpha,\beta}(\mu_i)}) \\ &= (g^{H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i)}, h^{H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i)}) \\ &= (T', U'), \\ e(T, h) &= e(g^{H_{\alpha,\beta}(\mu)}, h) = e(g, h^{H_{\alpha,\beta}(\mu)}) \\ &= e(g, U), \\ e(X, h) &= e(g^{\sum_{i=1}^{t}(u_i u + v_i v) - H_{\alpha,\beta}(\mu)}, h)^{1/d} \\ &= e(g, h^{\sum_{i=1}^{t}(u_i u + v_i v) - H_{\alpha,\beta}(\mu)})^{1/d} \\ &= e(g, Y), \\ e(T, h) \cdot e(X, h)^d &= e(g^{H_{\alpha,\beta}(\mu)}, h) \\ &\quad \cdot e(g^{\sum_{i=1}^{t}(u_i u + v_i v) - H_{\alpha,\beta}(\mu)}, h) \\ &= e(g, h)^{\sum_{i=1}^{t}(u_i u + v_i v)} = W. \end{aligned}$$

□

Then, the requester continues to derive the summation of all chosen workers' data $\sum_{i=1}^{t} m_i$ by decrypting μ as follows,

$$\texttt{Dec}_{\texttt{dk}}(\mu) = \texttt{Dec}_{\texttt{dk}}(\sum_{i=1}^{t} \mu_i) = \texttt{Dec}_{\texttt{dk}}(\sum_{i=1}^{t} c_{i0} + \sum_{i=1}^{t} c_{i1} Y)$$
$$= \sum_{i=1}^{t} c_{i0} - s \cdot \sum_{i=1}^{t} c_{i1} = \sum_{i=1}^{t} m_i.$$

D. Extensions

In addition to the computation of the summation, our also show the efficient verifiable computation of a few other basic aggregation statistics. Efficient data update is also provided.

1) Statistics: Here, we demonstrate the computation of the mean, the variance, and the uncentered correlation coefficient.

Mean: The computation of mean is very useful in many applications. One example is that the transportation bureau can estimate the traffic condition in one area by computing the average speed of the cars. The speed values are sensed by the workers' smartphones. Assume a requester wants to know the mean value of t data $\vec{m} = (m_1, m_2, ..., m_t)$, where m_i is the data contributed by worker W_i. The computation of the mean is straightforward. The requester can use the above method to get the verifiable computation results of the summation $S_{\vec{m}} = \sum_{i=1}^{t} m_i$ of all data from t workers and divides it by t to get the mean

$$\mu_{\vec{m}} = \frac{S_{\vec{m}}}{t}.$$

Variance: The computation of aggregation variance is also very meaningful in many applications. The variance of t data values $\vec{m} = (m_1, m_2, ..., m_t)$ is computed as follows,

$$\texttt{Var}(\vec{m}) = \frac{\sum_{i=1}^{t}(m_i - \mu_{\vec{m}})^2}{t} = \frac{\sum_{i=1}^{t} m_i^2}{t} - \mu_{\vec{m}}^2.$$

Apparently, the computation can be split into two parts, the first part is the mean of m_i^2, and the other part is the square of the mean $\mu_{\vec{m}}$. Since we have already introduced how to compute $\mu_{\vec{m}}$, then here we only need to find the computation of $\mu_{\vec{m}^2} = (\sum_{i=1}^{t} m_i^2)/t$. Let $S_{\vec{m}^2} = \sum_{i=1}^{t} m_i^2$, then the computation of Var($\vec{m}$) is reduced to the computation of $S_{\vec{m}^2}$.

Let $f_1 = \sum_{i=1}^{t} x_i^2$, the requester precomputes $\omega_{f_1}(z_1, z_2) = f_1(\rho_1(z_1, z_2), \rho_2(z_1, z_2), ..., \rho_t(z_1, z_2))$, $\hat{\omega} = \omega_{f_1}(u, v) = \sum_{i=1}^{t}(u_i u + v_i v)^2$, and $\hat{W} = e(g, h)^{\hat{\omega}}$. Requester sends $(\tau = \text{``}\textit{sum of squares}\text{''}, t, pk)$ to the cloud as a request. Cloud collects (μ_i, σ_i) from workers in the same way as above, and it computes $\hat{\mu} = \sum_{i=1}^{t} \mu_i^2$ and $\hat{\sigma} = (\hat{T}, \hat{X}, \hat{Y}, \hat{\Lambda})$, where

$$\hat{T} = \prod_{i=1}^{t} e(T_i, U_i),\ \hat{X} = \prod_{i=1}^{t} e(X_i, U_i)^2,$$
$$\hat{Y} = \prod_{i=1}^{t} e(T_i, Y_i)^2,\ \hat{\Lambda} = \prod_{i=1}^{t} e(X_i, Y_i).$$

Then, cloud sends $(\hat{\mu}, \hat{\sigma})$ to requester who computes $H(\hat{\mu}) = e(g, h)^{H_{\alpha,\beta}(\hat{\mu})}$ and checks if the following equations hold.

$$\hat{T} \overset{?}{=} H(\hat{\mu}),\ \hat{X} \overset{?}{=} \hat{Y},\ \hat{W} \overset{?}{=} \hat{T} \cdot \hat{X}^d \cdot \hat{\Lambda}^{d^2}.$$

If any of the above equation does not hold, the requester rejects the results. Otherwise, the requester accepts the results. If the results are correct, the equations hold as follows,

Proof of correctness:.

$$\hat{T} = \prod_{i=1}^{t} e(T_i, U_i) = \prod_{i=1}^{t} e(g^{H_{\alpha,\beta}(\mu_i)}, h^{H_{\alpha,\beta}(\mu_i)})$$
$$= e(g, h)^{H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i^2)} = H(\hat{\mu}),$$
$$\hat{X} = \prod_{i=1}^{t} e(X_i, U_i)^2 = \prod_{i=1}^{t} e(R_i \cdot T_i^{-1}, U_i)^2$$
$$= \prod_{i=1}^{t} e(g^{(u_i u' + v_i v') - H_{\alpha,\beta}(\mu_i)}, h^{H_{\alpha,\beta}(\mu_i)})^{2/d}$$
$$= \prod_{i=1}^{t} e(g^{H_{\alpha,\beta}(\mu_i)}, h^{(u_i u' + v_i v') - H_{\alpha,\beta}(\mu_i)})^{2/d}$$
$$= \hat{Y},$$
$$\hat{T} \cdot \hat{X}^d \cdot \hat{\Lambda}^{d^2} = e(g, h)^{H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i^2)}$$
$$\cdot\, e(g, h)^{2\sum_{i=1}^{t}(u_i u' + v_i v')H_{\alpha,\beta}(\mu_i) - 2H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i^2)}$$
$$\cdot\, e(g, h)^{\sum_{i=1}^{t}(u_i u' + v_i v')^2 - 2\sum_{i=1}^{t}(u_i u' + v_i v')H_{\alpha,\beta}(\mu_i)}$$
$$\cdot\, e(g, h)^{H_{\alpha,\beta}(\sum_{i=1}^{t} \mu_i^2)} = e(g, h)^{\sum_{i=1}^{t}(u_i u' + v_i v')^2}$$
$$= \hat{W}.$$

□

Then, requester continues to derive $S_{\vec{m}^2} = \sum_{i=1}^{t} m_i^2$ by simply decrypting $\hat{\mu}$. Finally, the requester derives the variance:

$$\texttt{Var}(\vec{m}) = S_{\vec{m}^2}/t - S_{\vec{m}}^2/t^2.$$

Uncentered correlation coefficient: The uncentered correlation coefficient is also an important statistics, which is computed using the following formula.

$$r_{\vec{m},\vec{m}'} = \frac{\sum_{i=1}^{t} m_i m_i'}{\sqrt{\sum_{i=1}^{t} m_i^2} \cdot \sqrt{\sum_{i=1}^{t} m_i'^2}}$$

Since $S_{\vec{m}^2} = \sum_{i=1}^{t} m_i^2$ and $S_{\vec{m}'^2} = \sum_{i=1}^{t} m_i'^2$ can be computed using the same method in the Variance part, where $\vec{m}^2 = (m_1^2, m_2^2, ..., m_t^2)$ and $\vec{m}'^2 = (m_1'^2, m_2'^2, ..., m_t'^2)$, the requester only needs to obtain the scalar product of $\vec{m}$ and $\vec{m}'$: $S_p = \sum_{i=1}^{t} m_i m_i'$.

Let $f_2(\vec{x}, \vec{y}) = \sum_{i=1}^{t} x_i y_i$, and assume there are two tags τ and $\tilde{\tau}$, where τ is the same as in variance and summation parts but $\tilde{\tau}$ is a new tag. $F'_{K_2}(\tilde{\tau}) = (u'', v'')$ The requester precomputes $\omega_{f_2}(z_1, z_2) = f_2((\rho_1(z_1, z_2), ..., \rho_t(z_1, z_2)), (\tilde{\rho}_1(z_1, z_2), ..., \tilde{\rho}_t(z_1, z_2)))$, $\tilde{\omega} = \omega_{f_2}(u', v') = \sum_{i=1}^{t}(u_i u' + v_i v')(u_i u'' + v_i v'')$, and $\tilde{W} = e(g, h)^{\tilde{\omega}}$.

The requester sends $(f_2 = \sum_{i=1}^{t} x_i y_i, t, \tau, \tilde{\tau}, pk)$ to the cloud and waits for the results. Cloud collects (μ_i, σ_i) and (μ_i', σ_i') from workers, where $\mu_i' = H(m_i'), \sigma_i' = (T_i', U_i', X_i', Y_i', \Lambda_i')$, for $i = 1, 2, ..., t$. Then, cloud computes $\tilde{\mu} = \sum_{i=1}^{t} \mu_i \mu_i'$ and $\tilde{\sigma} = (\tilde{T}, \tilde{U}, \tilde{X}, \tilde{Y}, \tilde{\Lambda})$ as follows,

$$\tilde{T} = \prod_{i=1}^{t} e(T_i, U_i'),\ \tilde{U} = \prod_{i=1}^{t} e(T_i', U_i),\ \tilde{\Lambda} = \prod_{i=1}^{t} e(X_i, Y_i'),$$
$$\tilde{X} = \prod_{i=1}^{t} e(X_i, U_i')e(X_i', U_i),\ \tilde{Y} = \prod_{i=1}^{t} e(T_i, Y_i')e(T_i', Y_i)$$

Then, cloud sends $\tilde{\mu}$ and $\tilde{\sigma} = (\tilde{T}, \tilde{U}, \tilde{X}, \tilde{Y}, \tilde{\Lambda})$ to the requester. When the requester receives the results from the cloud, he computes the hash value $H(\tilde{\mu}) = e(e, h)^{H_{\alpha,\beta}(\tilde{\mu})}$ and checks if the following equations hold.

$$\tilde{T} \overset{?}{=} \tilde{U} \overset{?}{=} H(\tilde{\mu}),\ \tilde{X} \overset{?}{=} \tilde{Y},\ \tilde{W} \overset{?}{=} \tilde{T} \cdot \tilde{X}^d \cdot \Lambda^{d^2}.$$

If all the equations hold, the requester accepts the results and decrypts $\tilde{\mu}$ to get $S_p = \sum_{i=1}^{t} m_i m_i'$. Finally, the uncentered correlation coefficient is

$$r_{\vec{m},\vec{m}'} = \frac{S_p}{\sqrt{S_{\vec{m}^2}} \cdot \sqrt{S_{\vec{m}'^2}}}.$$

2) Efficient Data Update: The requester may need to update the aggregated statistics periodically, which brings much burden to workers if they have to participate from scratch. We consider data update in two cases.

Case 1: The chosen workers stay the same, i.e., $\{W_1, ..., W_t\}$, but a subset of them have new data to update. We assume the subset is $\{W_1, ..., W_n\}$, where $n \leq t$. In this case, only workers with new data need to participate in the data aggregation, while the rest of workers do not need to do anything. This reduces a lot of computation and communication load of the whole system. Take computation of sum as an example, $W_i \in \{W_1, ..., W_n\}$ has new data m_i' and old data m_i. W_i computes

$$\mu_i' = Enc_{pk}(m_i') - Enc_{pk}(m_i)$$

and $\delta_i' = (T_i'/T_i, U_i'/U_i, X_i'/X_i, Y_i'/Y_i, \Lambda_i'/\Lambda_i)$ for m_i', where $(T_i', U_i', X_i', Y_i', \Lambda_i')$ is computed in the same way as Equations (1)-(3). Then, W_i sends (μ_i', δ_i') to cloud. Cloud computes $\mu' = \mu + \mu_i'$ and $\delta' = (T', U', X', Y', \Lambda')$, where

$$T' = T \cdot \frac{T_i'}{T_i}, U' = U \cdot \frac{U_i'}{U_i}, X' = X \cdot \frac{X_i'}{X_i}, Y' = Y \cdot \frac{Y_i'}{Y_i}, \Lambda' = \Lambda \cdot \frac{\Lambda_i'}{\Lambda_i}.$$

Finally, cloud sends (μ', δ') to requester. Requester verifies the correctness of results in the same way as Equation (4). Data update is very efficient in this way, if only a subset of workers have new data. Similar process can be applied to the computation of mean, variance and correlation coefficient.

Case 2: Some of the chosen workers leave while some new workers join, and the total number of chosen workers is unchanged. In this case, a leaving worker W_i sends (μ_i, δ_i) to a joining worker W_j, and W_j computes

$$\mu_j = Enc_{pk}(m_j) - Enc_{pk}(m_i)$$

and $\delta_j = (T_j/T_i, U_j/U_i, X_j/X_i, Y_j/Y_i, \Lambda_j/\Lambda_i)$ for m_j, where $(T_j, U_j, X_j, Y_j, \Lambda_j)$ is computed in the same way as Equations (1)-(3). Then, W_j sends (μ_j, δ_j) to cloud. Cloud computes $\mu' = \mu + \mu_j$ and $\delta' = (T', U', X', Y', \Lambda')$, where

$$T' = T \cdot \frac{T_j}{T_i}, U' = U \cdot \frac{U_j}{U_i}, X' = X \cdot \frac{X_j}{X_i}, Y' = Y \cdot \frac{Y_j}{Y_i}, \Lambda' = \Lambda \cdot \frac{\Lambda_j}{\Lambda_i}.$$

Finally, cloud sends (μ', δ') to requester. The rest is the same as in **Case 1**.

V. Protocol Evaluation

In this section, we first discuss how the security and privacy requirements are achieved, then we demonstrate the efficiency and feasibility of our scheme through extensive simulation.

A. Security Analysis

1) Data Privacy: In our scheme, a worker's data m_i is encrypted by the *BGV* Homomorphic Encryption with the requester's public encryption key $pk = (a, b)$ to a ciphertext $\mu_i = c_0 + c_1 \cdot Y$, where $c_0 = br_2 + pr_3 + m$ and $c_1 = ar_2 + pr_1$. When the workers participate in the aggregation task, they will send their encrypted data μ_i to the cloud. The cloud will not be able to derive m_i from μ_i because there are only two equations but with four unknown factors. After cloud computes the summation, variance or uncentered correlation coefficient, these values are still in encrypted form due to the homomorphic property of the encryption method. Therefore, only the requester who has the decryption key dk can get the plaintext of the encrypted data. In our scheme, the workers send their encrypted data to the cloud using *IBE*, so that even if the message from a worker to the cloud is intercepted by the requester or others, the message will not be decrypted because it is encrypted using the cloud's public key.

2) Correctness of Verification: Here, we analyze the correctness of verification for the summation computation. First of all, when the requester receives μ and $\sigma = (T, U, X, Y, \Lambda)$ from the cloud, he first computes $W = e(g, h)^{\omega_f(u', v')} = e(g, h)^{\sum_{i=1}^{t}(u_i u' + v_i v')}$. Then, he checks if $W = e(T, h) \cdot e(X, h)^d$. According to the *l-BDHI* Assumption, this equation hold only when X contains $\sum_{i=1}^{t}(u_i u' + v_i v')$ in the exponent of g. If this is true, the requester knows that $X = \prod_{i=1}^{t} X_i = g^{\sum_{i=1}^{t}(u_i u' + v_i v') - H_{\alpha,\beta}(\mu)}$. Therefore, T must be equal to $g^{H_{\alpha,\beta}(\mu)}$. Then, according to *DDH*, if both $e(T, h) = e(g, U)$ and $e(X, h) = e(g, Y)$ hold, U and Y are also correctly computed by the cloud. Until here, the requester has verified that (T, U) is correct. Finally, if $H(\mu) = (T, U)$ also hold, the requester knows that μ is indeed the summation $\sum_{i=1}^{t} m_i$. The correctness of verification for the variance and uncentered correlation coefficient computation can be proved in the same way.

3) Identity Privacy: According to [31], if there are t chosen workers, and every worker signs his data with ring signature, then the probability of identifying the owner of the signature is at most $1/t$, which is equal to random guessing.

B. Performance Analysis

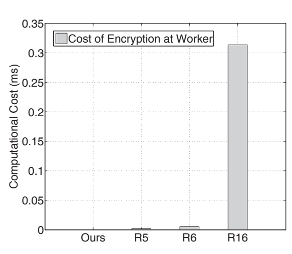
1) Simulation Setup: We conduct our simulation based on two libraries: *PBC* [32] and *HElib* [33]. In particular, in *PBC*, we use the Type A elliptic curve, which has the form of $y^2 = x^3 + x$. We use VMWare Workstation 10 Ubuntu 14.04 in a desktop with Intel Core i5 processor and 4GB RAM for the simulation. Our simulation dataset is an open dataset from [34], and we use the data of weights here.

2) Simulation Results: We denote the schemes in [5], [6], [16] as R5, R6 and R16 respectively.

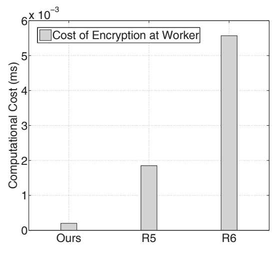
- **Cost of Encryption at Worker:**

First, we compare the encryption cost of a worker in our scheme with the other three schemes R5, R6, and R16. All the encryption costs are averaged over 100 users. The comparison is shown in Fig.3(a) and Fig.3(b). As we can see from Fig.3(a) and Fig.3(b), the average encryption cost of each worker in our scheme is the lowest. Low encryption cost is beneficial to mobile workers, because it means longer battery usage.

- **Cost of Sum at Requester:**

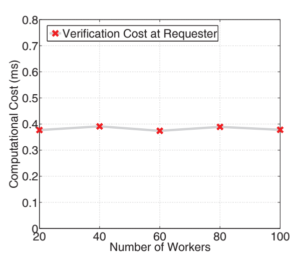


(a) Four Schemes

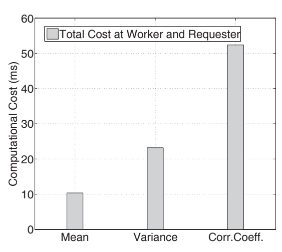


(b) Without R16

Fig. 3. Cost of Encryption at Worker



(a) Cost of Verification



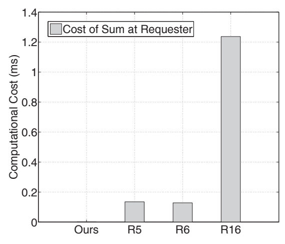
(b) Cost of Three Statistics

Fig. 5. Cost of Verification at Requester and Cost of Three Statistics

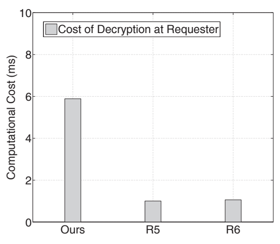
Next, we simulate the cost of sum at requester when there are 100 chosen workers. The simulation results is given in Fig.4(a). Since in our scheme the computation of sum is outsourced to the cloud, there is no cost at the requester. However, the requester in R5, R6, R16 needs to compute the sum by herself.

- **Cost of Decryption at Requester:**

We also simulate the cost of decryption at requester when there are 100 chosen workers. The simulation results is given in Fig.4(b). The cost of decryption in our scheme is 5.89ms, while the costs in R5, R6, and R16 are 1.005ms, 1.061ms, and 4706ms respectively. To make figure look clear, the cost of R16 is not given in Fig.4(b). It's obvious that our decryption cost is lower than R16, but higher than R5 and R6. The comparatively high decryption cost in our scheme is a sacrifice to achieve verifiability.



(a) Cost of Sum



(b) Cost of Decryption

Fig. 4. Cost at the Requester

- **Cost of Verification at Requester:**

The best feature of our scheme is the verifiability of correctness of outsourced computation. Intuitively, the cost of verification at requester in our scheme is a constant regardless of the number of workers. The reason is that requester always verifies a single result retrieved from cloud whatever the number of workers is. The simulation result is shown in Fig.5(a).

- **Cost of Three Statistics:**

We simulate the total computational cost of our proposed scheme regarding the computation of mean, variance and uncentered correlation coefficient based on 100 workers. The simulation results are shown in Fig.5(b). Here, the total computational cost refers to the sum of costs at both workers and requester. Because in our scheme the cloud is introduced to ease computation burden of the requester, we are more interested in the total cost at workers and requester. This total cost is crucial in reflecting the efficiency and feasibility of our scheme. When there are 100 chosen workers, the costs for mean, variance, and correlation coefficient are 10.366ms, 23.188ms, and 52.371ms respectively.

- **Correctness Verification:**

To provide a simple presentation for the correctness verification, we assume that there are 150 workers and each reports one datum encrypted with *BGV* homomorphic encryption to the cloud. All data are generated according to the normal distribution $\mathcal{N}(50, 20)$. The data are shown as the brown dots and $\mathcal{N}(50, 20)$ the green curve in Fig.6. Since the data are discrete values, when there are only limited number of them, the real data distribution will slightly deviate from the original ideal distribution. The real distribution in this simulation is $\mathcal{N}(51.34, 19.38)$, which is shown as the red curve in Fig.6. Based on the data from workers, cloud computes the mean and variance for the requester. If the computation is correct, the mean and variance should be the same as in $\mathcal{N}(51.34, 19.38)$. As we can see from Fig.6, the retrieved data distribution

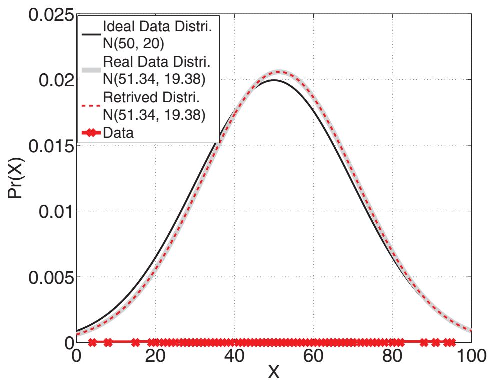


Fig. 6. Correctness Verification

is exactly overlapping with the real data distribution, which means the retrieved distribution is statistically identical to the original one.

- **Update Efficiency:**

Finally, we show the efficiency of our scheme in data updates. We assume there are $t = 6 \times 10^3$ workers, and simulate the computational cost of the workers using our update scheme and the computational cost of total recomputation. The simulation results are shown in Fig.7. The blue line shows the computational cost of workers with total recomputation. The

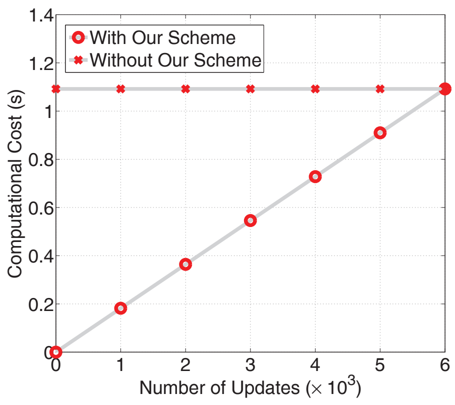


Fig. 7. Comparison of Update Efficiency

cost is high because all workers need to recompute everything whatever the number of real updates is. The red line shows the computational cost when our update scheme is used. It is much more efficient when using our update scheme, because only the workers who has a new data (in **Case 1**) and new joining workers (in **Case 2**) need to compute again. While the rest of the workers have no computational costs.

VI. Conclusion

In this paper, we propose a privacy-preserving and verifiable data aggregation scheme for cloud-assisted mobile crowdsourcing. The proposed scheme enables a requester to delegate data aggregation and analysis task to the cloud. In this process, workers' identity and data privacy are well protected. Meanwhile, requester can verify the correctness of the retrieved results. Simulation results show the efficiency and feasibility of our scheme.

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] 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.

[3] Q. Li and G. Cao, "Privacy-preserving participatory sensing."

[4] ——, "Efficient and privacy-preserving data aggregation in mobile sensing," in *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, Oct 2012, pp. 1–10.

[5] 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.

[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] M. Li and P. Li, "Crowdsourcing in cyber-physical systems: Stochastic optimization with strong stability," *Emerging Topics in Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 218–231, 2013.

[8] 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.

[9] J. Zhou, Z. Cao, X. Dong, and X. Lin, "Ppdm: Privacy-preserving protocol for dynamic medical text mining and image feature extraction from secure data aggregation in cloud-assisted e-healthcare systems," *Selected Topics in Signal Processing, IEEE Journal of*, vol. PP, no. 99, pp. 1–1, 2015.

[10] 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.

[11] 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.

[12] 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.

[13] 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*. IEEE, 2014, pp. 244–252.

[14] 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.

[15] A. Kapadia, D. Kotz, and N. Triandopoulos, "Opportunistic sensing: Security challenges for the new paradigm," in *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. IEEE, 2009, pp. 1–10.

[16] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data." in *NDSS*, vol. 2, no. 3, 2011, p. 4.

[17] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 327–332.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] G. Zhuo, Q. Jia, L. Guo, M. Li, and Y. Fang, "Privacy-preserving verifiable proximity test for location-based services," in *GC' 15 - Security*, San Diego, USA, Dec. 2015.

[24] 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.

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

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

[27] A. J. Menezes, T. Okamoto, S. Vanstone *et al.*, "Reducing elliptic curve logarithms to logarithms in a finite field," *Information Theory, IEEE Transactions on*, vol. 39, no. 5, pp. 1639–1646, 1993.

[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 309–325. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090262

[29] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Advances in Cryptology–CRYPTO 2011*. Springer, 2011, pp. 505–524.

[30] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 863–874.

[31] 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.

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

[33] S. Halevi and V. Shoup, "Helib, http://shaih.github.io/HElib/," access June 20, 2015.

[34] UCLA, "Socr data mlb heightsweights, http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights," access December 31, 2015.