Computer Communications 146 (2019) 85–94

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Optimizing rewards allocation for privacy-preserving spatial crowdsourcing

Ping Xiong [a], Danyang Zhu [a], Lefeng Zhang [a], Wei Ren [b], Tianqing Zhu [b,c,*]

[a] School of Information and Security Engineering, Zhongnan University of Economics and Law, Wuhan, 430073, China
[b] School of Computer Science, China University of Geosciences, Wuhan, 430074, China
[c] University of Technology Sydney, 2000, NSW, Australia

ARTICLE INFO

Keywords:
Spatial crowdsourcing
Privacy preservation
Homomorphic encryption

ABSTRACT

Rewards allocation in one of the key issues for ensuring a high task acceptance rate in spatial crowdsourcing applications. Generally, workers who participate in a crowdsourcing project are required to disclose their locations, which may lead to serious privacy threats. Unfortunately, providing a rigid privacy guarantee is incompatible with ensuring a high task acceptance rate in most existing crowdsourcing solutions. Hence, this paper proposes a crowdsourcing framework based on optimized reward allocation strategies. The key idea is to tune the reward for performing each task to the workers' preferences to attain a high acceptance rate. The first step in the framework is to interrogate the workers' preferences using a cryptographic protocol that fully preserves the location privacy of the workers. Based on those preferences, two different approaches to reward assignments have been proposed to ensure the rewards are distributed optimally. A theoretical analysis of the privacy protection inherent in the framework demonstrates that the proposed framework guarantee the worker's location privacy from adversaries including the requester and crowdsourcing server. Further, experiments based on real-world datasets show that the proposed strategies outperform existing solutions in terms of task acceptance rates.

1. Introduction

With the increasing use of smart devices equipped with multi-modal sensors, *spatial crowdsourcing* has become quite popular for location-based tasks, such as taking photos, recording videos, and reporting temperatures [1] at specified locations. A typical crowdsourcing framework generally involves three parties: a crowdsourcing server, requesters, and sets of workers. The server maintains a centralized platform, such as Mechanical Turk[1] and Freelancer,[2] on which various crowdsourcing projects can be performed. For each spatial crowdsourcing project, a requester posts the location-based tasks of the project on the server, a set of registered workers then physically travel to the specified locations to perform some of the tasks and receive rewards.

In the crowdsourcing landscape, the proportion of tasks are actually accepted by workers is an essential measurement for evaluating the efficiency of a crowdsourcing strategy. However, maintaining a high task acceptance rate while preserving the workers' location privacy is a critical issue. Two popular modes for accomplishing this goal are widely used in practice [2]: worker selected mode and server assigned mode [3]. In worker selected mode, the crowdsourcing server posts the tasks, and each worker selects their preferred tasks autonomously. This mode generally has a low computational cost but also has a lower task acceptance rate, as workers intuitively prefer nearby tasks. The "outlier" tasks with no close workers are typically not accepted. In server assigned mode, the workers upload their real locations to the crowdsourcing server, then each task is assigned to the most suitable worker according to each worker's location. Since the server assigned mode has a higher task acceptance rate, it has become the more popular mode of crowdsourcing. However, this mode does present great risks to location privacy. By exploiting user locations, an adversary may deduce sensitive information, such as a worker's home address, political views, or religious inclinations [4–6].

Formally, location privacy refers to people's sensitive information that related to their location data, such as sensitive location points and daily trajectories. In big data era where massive amount of data can be accessed, people's location privacy should not be revealed in the processes of data collecting, storing and analyzing. On the other hand, cyber security has already been developed to a severe threat to individuals, organizations and enterprises. A cyber security incident may cause destructive damage, such as data disclosure and financial losses [7]. As a booming mobile application which involves workers' location information, spatial crowdsourcing has to deal with this urgent threat.

* Corresponding author at: School of Computer Science, China University of Geosciences, Wuhan, 430074, China.
E-mail address: tianqing.zhu@uts.edu.au (T. Zhu).

[1] https://www.mturk.com.
[2] https://www.freelancer.com.

https://doi.org/10.1016/j.comcom.2019.07.020
Received 9 April 2019; Received in revised form 17 June 2019; Accepted 24 July 2019
Available online 1 August 2019

Several approaches have been proposed to address location privacy issues in server assigned mode in recent years. *Spatial cloaking* is the most common technique [8–11]. Through generalization or by introducing dummy points, each user's location is hidden within a cloaked region. However, a priori knowledge about the targets can be exploited to break privacy definitions [12].

Another category of methods relies on differential privacy (DP) [13] to randomize the distribution of worker locations [1,14,15]. A trusted third party is incorporated into the schema to generate a randomized distribution, which the crowdsourcing server uses to assign tasks. However, the usefulness of the data is decreased significantly when there is too much calibrated noise.

Cryptographics is the most effective way to retain both data utility and preserve privacy [16,17] and in recent years, many location-based services have introduced homomorphic encryption [18–20]. Carefully designing a protocol with acceptable computing costs is the main concern with these approaches, as most need to introduce a fourth party to handle the encrypted data, which increases the complexity of a crowdsourcing framework with extra computing and communication costs.

This paper proposes a secure framework for reward-based spatial crowdsourcing (SecRSC). Similar to worker selected mode, SecRSC allows workers to select their preferred tasks autonomously. Meanwhile, worker preferences for accepting tasks are interrogated privately to fully preserve location privacy, which is similar to server assigned mode. Moreover, task acceptance rates can be substantially improved by tuning the distribution of rewards according to the investigated preferences. Specifically, the requester publishes the required tasks on the crowdsourcing server. The workers then privately report their preferred tasks to the crowdsourcing server using the proposed cryptographic protocol. This generates a distribution showing, for each task, how many of the workers will potentially accept it. The reward for each task is finally calculated according to that distribution by the requester.

Given a fixed total reward budget, the goal is to obtain a higher task acceptance rate by offering additional rewards for the outlying tasks to increase their acceptance probability.

To achieve this objective, three challenges need to be addressed:

- Ensuring that the privacy of worker locations is preserved while investigating each worker's preference of selecting tasks. We aim to generate a worker distribution on tasks which can be further used for reward assignment by the requester. However, given the assumption that the crowdsourcing server and the requester are untrusted, how to protect the preferred tasks of each worker, which might imply the approximate location of the worker, from being deduced by both the crowdsourcing server and the requester, is one of the major challenges.
- Designing a cryptographic protocol for private data aggregation among multiple parties. Generally, ensuring low costs on computation and communication is the key for any secure multi-party computation problem. In spatial crowdsourcing, how to represent each worker's preference efficiently and transmit it among parties securely with high computational efficiency and low communication overhead is a challenge.
- Specifying adaptive rewards for each task given a fixed total reward budget. We aim to generate a vector of rewards that maximizes the task acceptance rate. However, how to define the condition which may lead to the maximal task acceptance rate is the key issue to be handled. In addition, for each task t, we need to specify the amount of reward, r, according to the number of workers who might accept the task, f, how to define a functional form which can properly characterize the relationship between r and f is also a challenge.

To address the first challenge, we use homomorphic encryption, which is particularly suitable for secure cloud-computing applications. In SecRSC, the information that workers submit to the crowdsourcing server is encrypted by homomorphic cryptosystem, and the server completes the required calculation without necessarily knowing the content of the data. To address the second challenge, we specify a prime number for each task. Hence, any given set of tasks can be represented as the product of the corresponding prime numbers, creating a unique number for each task set. Given very few bits are required for storing or transmitting a plain text, this approach results in low computation and communication costs. For the third challenge, we attempt to define a metric to evaluate the popularity degree of each task. By tuning the reward for each task, we try to ensure that each task is approximately the same attractive for each worker.

The contributions of this paper can be summarized as follows:

- We propose a novel framework for reward-based crowdsourcing to achieve a high task acceptance rate while preserving the location privacy of the workers.
- We present a cryptographic protocol for private data aggregation that leverages the advantages of both prime numbers and homomorphic encryption. This protocol could also be applied to other scenarios, such as private elections and private questionnaires.
- We introduce two different reward assignment approaches — one based on *identical rates of return*, and the other on an *optimized constraint function*. The experimental results demonstrate that either approach can significantly improve task acceptance rates.

The rest of this paper is organized as follows. We introduce the background and related work in Section 2. Then we propose the framework, *SecRSC*, in Section 3. In Section 4, we present the privacy and performance analysis of the proposed framework. Section 5 shows the experimental results, followed by the conclusion in Section 6.

2. Background

2.1. Definitions and notations

Let $T = \{t_1, t_2, \ldots, t_{|T|}\}$ be a spatial task set that includes $|T|$ tasks uploaded by a requester to a crowdsourcing server. Each task t_i consists of a tag and a location, $t_i = \langle tag_i, l_i \rangle$, where tag_i is a nonce, i.e., a number used once, and uniquely identifies t_i; l_i is the location where t_i needs to be performed. $W = \{w_1, w_2, \ldots, w_{|W|}\}$ is the set of workers involved in the crowdsourcing project, where $|W|$ is the number of workers.

In the proposed SecRSC, the requester needs to investigate each task may be accepted by any number of workers. The goal is to obtain the worker distribution for the tasks, denoted as F, which is further used for reward allocation. Therefore, each worker needs to submit tags for their preferred tasks to the crowdsourcing server. Given tasks with identical rewards, workers are assumed to prefer tasks nearer to their exact location. Let $T_w \subset T$ be the tasks that worker w prefers to perform. We regard T_w as w's location privacy because it implies that w's exact location might be close to these task points. Thus, we need to prevent T_w for each worker w from being disclosed to the requester and the crowdsourcing server, as well as other adversaries. The formal definition of location privacy in SecRSC is presented in Section 4.

In SecRSC, we use homomorphic encryption [21,22] to protect the messages submitted by the workers. Homomorphic encryption allows computations to be performed on ciphertext and, once decrypted, the results are the same as operations performed on plain text. Partial homomorphic cryptosystems [22–24] support some operations on ciphertext, such as addition, multiplication, and quadratic functions, while fully homomorphic cryptosystems [25,26] allow any required computation to be performed on ciphertexts.

In our proposed protocol, we employ the ElGamal cryptosystem [23] with its multiplicative homomorphic property. ElGamal cryptosystem is a triple of algorithms: the key generator G, the encryption algorithm E, and the decryption algorithm D. Given a cyclic group $\mathbb{G}$ of order q with generator g, the private key $sk = x$ is randomly sampled from

$\{1, \ldots, q-1\}$, the public key is $pk = (\mathbb{G}, q, g, h)$ where $h = g^x$. Given a plaintext m, the encryption of m is:

$$c = E(m) = (c_1, c_2) = (g^r, m \cdot h^r) \tag{1}$$

where r is a random from $\{1, \ldots, q-1\}$. And the decryption of c is:

$$D(c) = c_2 \cdot (c_1^x)^{-1} = m \cdot g^{xr} \cdot (g^{xr})^{-1} = m \tag{2}$$

ElGamal has a multiplicative homomorphic property as follows:

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2}) \\ &= (g^{(r_1+r_2)}, m_1 \cdot m_2 \cdot h^{r_1+r_2}) \\ &= E(m_1 \cdot m_2) \end{aligned} \tag{3}$$

2.2. Related work

There have been many studies on privacy preservation for spatial crowdsourcing in recent years. The existing approaches can be classified into three categories, spatial cloaking [8–11], DP-based methods [1,14,15,27], and cryptographic methods [18–20,28,29].

Spatial cloaking generalizes an exact location within a cloaked region, such as Pournajaf et al.'s [10] twofold approach. Cloaked areas are reported to the crowdsourcing server rather than exact locations, and the global task assignments are solved with uncertain locations. Workers then refine the results of the first stage according to their exact location. Agir et al. [8] presented a location-obfuscation solution to preserve privacy. Obfuscation areas are generated dynamically, to replace exact locations, according to the parameters of personalized privacy requirements. The main shortcoming of spatial cloaking is that its reliability is sensitive to an adversary's background knowledge. In addition, spatial cloaking cannot completely hide the information about a location because an adversary can still confirm that the location belongs to a certain region.

Differential privacy exploits randomization mechanisms to generate and release an aggregated report that cannot be interpreted by an adversary to determine whether or not any particular user is included in the dataset [30]. To et al. [1] proposed a framework for protecting worker locations by including the cellular service provider as a trusted third party. The service provider generates a sanitized distribution of workers in the crowdsourcing region using a differentially private spatial decomposition algorithm [31]. Xiong et al. [15] proposed a differentially private method for reward-based crowdsourcing. Worker distribution is represented by a contour plot with a DP guarantee. The drawback of DP-based methods is that the utility of the aggregated result might be dramatically decreased, especially when the privacy budget is relatively small, due to the large volume of noise. In addition, these methods generally involve a trusted third party, which is a potential vulnerability. The trusted third party might become a single point of attack if it is compromised by adversaries [20]. In addition, user information may be abused by the trusted third party for a range of reasons. For example, government agencies, using public security as a rationale, may demand that enterprises provide client information as reported in [32].

Cryptographic methods aim to completely protect location privacy through cryptographic protocols. Shen et al. [29] proposed a secure privacy framework for crowdsourcing using additive homomorphic encryption. A privacy service provider (PSP), as a semi-honest third party, is introduced to provide the privacy functionality and collect encrypted private data. Liu et al. [20] proposed a secure crowdsourcing framework, HESI, based on homomorphic encryption. A logistic server is introduced to construct an SKD-tree for indexing encrypted worker locations. However, these methods expands the typical crowdsourcing framework by introducing other parties (a PSP or a server), which increases the complexity of the framework and imposes an extra computing cost.

We focus on reward-based crowdsourcing in this paper and propose a novel framework, called SecRSC. The unique differences compared with the existing works lie on two aspects: Firstly, unlike existing privacy preserving approaches with worker selected mode or server assigned mode, SecRSC combines the advantages of the both modes. SecRSC allows workers select tasks autonomously so that the framework can be performed efficiently. It also collects information from workers which is further used as prior knowledge to increase the task acceptance rate. Secondly, unlike other homomorphic encryption-based approaches, such as the works in [29] and [20], which have to introduce an extra server, a fourth party, for performing complicated protocols and expensive communications, SecRSC maintains a third-party framework while the crowdsourcing server undertakes few extra computation according to the proposed protocol.

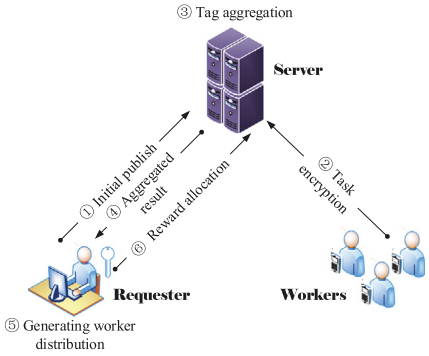


Fig. 1. The framework of SecRSC.

3. The SecRSC framework

3.1. Overview of SecRSC

Rewards allure men to brave danger, an ancient Chinese say, means that, given enough reward, any hard task will be accepted by some brave men. To increase the acceptance probability of those "undesirable" tasks, offering attractive reward is a straightforward and effective solution. Therefore, given a fixed amount of budget, the core idea of SecRSC is to decrease the reward for the coveted tasks so that the savings can be used to supplement those "undesirable" ones.

SecRSC is performed by two stages.

- Stage I: The requester investigates the popularity degree of each task without knowing any worker's individual preference. With the proposed cryptographic protocol in Section 3.2, the requester can obtain a worker distribution on the tasks.
- Stage II: According to the distribution obtained in Stage I, the requester computes the reward for each task using the reward assignment approaches proposed in Section 3.3.

We assume that the requester and the crowdsourcing server are semi-honest. They are curious about worker locations but will follow the protocol without collusion. Therefore, the goal of SecRSC is to achieve a high task acceptance rate with a fixed reward budget while preserving worker locations from the requester and the crowdsourcing server.

The SecRSC framework is shown in Fig. 1. Each party in the framework implements several steps in succession:

- Step ① *Initial publish*. The requester publishes the task set T on the crowdsourcing server. At this stage, the reward for each task has not yet been specified.
- Step ② *Task encryption*. Each worker, w, browses the published tasks and selects their preferred tasks. The tags for the selected tasks, represented as unique prime numbers, are multiplied. The product is then encrypted by a homomorphic encryption algorithm. Thus, the ciphertext, C_w, is submitted to the crowdsourcing server.

Table 1
Information access table.

Party	Task locations	Worker's location	Worker's distribution
Server	Yes	No	No
Requester	Yes	No	Yes
Worker	Yes	Yes	No

- Step ③ *Tag aggregation*. The crowdsourcing server calculates $\tilde{C}$ by multiplying all the C_ws from the workers.
- Step ④ *Aggregated result transfer*. The crowdsourcing server forwards the $\tilde{C}$ to the requester.
- Step ⑤ *Generating worker distribution*. Given $\tilde{C}$, the requester extracts the worker distribution of the tasks by decryption and factorization.
- Step ⑥ *Reward allocation*. Using the worker distribution as the input, the reword for each task is calculated by a pricing function and further published on the crowdsourcing server.

There are three algorithms involved in the proposed framework. In algorithm 1, each worker calculates the product of selected task tags and encrypts it with the public key of the requester, the process is demonstrated in step ②. The algorithm 2 allows crowdsourcing server to aggregate task tags by directly multiplying ciphertexts, which is shown step ③. Finally, the requester obtains the distribution of workers' preferred tasks through decryption and factorization process in algorithm 3, as shown in the step ⑤.

Note that computing reward is undertaken in Step ⑥ by the requester, rather than the server. Because the fundamental function of the server is to maintain the crowdsourcing platform and serve for various of crowdsourcing project simultaneously, vast of extra computation may incur substantial burdens. On the other hand, the computation is affordable by the requester. The capacity of terminal devices has been dramatically increased in recent years which makes fog computing [33] and federated learning [34] practicable.

Finally, workers autonomously select tasks and travel to the specified locations to perform them. Once a task has been completed, the reward for the task is transferred to the worker's account through an automated payment system, such as the centralized gateway of a bank, or a de-centralized payment service based on blockchain.

Three types of information can be accessed by the three parties in SecRSC, *task locations*, *worker locations* and *the worker distribution*, as shown in Table 1. Task locations are published openly so all parties can freely access this information. Worker locations are regarded as private; hence, each worker keeps their location a secret. SecRSC ensures that only the requester can obtain the worker distribution. In addition, the requester and the crowdsourcing server are unable to deduce any information about the worker locations using the information that can be accessed. The table demonstrates that each worker, crowdsourcing server and the requester have limited information, which is enough for their own process but inadequate to infer any information about others. The SecRSC keeps sensitive location information separated for each party participating in the framework. Therefore, workers' location privacy can be preserved.

The details of each step are presented in the remaining content of this section.

3.2. The homomorphic encryption-based protocol

In this subsection, we present the homomorphic encryption-based protocol, which aims to privately collect information from workers and generate the worker distribution. It involves steps ①∼⑤ of SecRSC. Then, we provide a case study to clearly demonstrate the workflow of the proposed protocol.

Table 2
Published tasks.

Task	*Location*	*Tag*
t_1	l_1	2
t_2	l_2	3
t_3	l_3	5
...	...	...
$t_{\|T\|}$	$l_{\|T\|}$	...

3.2.1. The cryptographic protocol

We use a public key encryption algorithm, ElGamal, in SecRSC and assume that the workers have access to the requester's public key pk.

In Step ①, the requester publishes the task set $T = \{t_1, t_2, \ldots, t_{|T|}\}$. To exploit the multiplicative homomorphic property of ElGamal, we specify a prime nonce to represent a tag for each task, with each task obviously having a different tag. Note that, any pairwise relatively prime positive integers are available to be used as the tags. Therefore, the published tasks might appear as Table 2.

In Step ②, each worker, w, generates their preferred task set T_w, and multiplies the tags of the tasks in T_w. The product is denoted by Tag_w,

$$Tag_w = \prod_{t_i \to tag_i, t_i \in T_w} tag_i \tag{4}$$

Because Tag_w can be uniquely factored to the product of the tags that reveal the task set T_w, it must be encrypted before it is submitted to the crowdsourcing server. Using the requester's public key, worker w encrypts Tag_w with ElGamal cryptosystem as:

$$C_w = E(pk, Tag_w) = (c_{1w}, c_{2w}) \tag{5}$$

The ciphertext, C_w, is finally submitted to the crowdsourcing server by worker w. Without the requester's private key, the crowdsourcing server is unable to decrypt C_w. The details of tag encryption are described in Algorithm 1.

Algorithm 1 Task Encryption (Worker)

Require: $pk = (\mathbb{G}, q, g, h)$, T_w
Ensure: Encrypted tags C_w
1: Calculates the product of the tags Tag_w with Equation (4);
2: Selects a random r from $\{1, 2, ..., q-1\}$, calculates $c_{1w} = g^r \bmod q$;
3: Calculates $c_{2w} = Tag_w \cdot h^r \bmod q$;
4: Outputs $C_w = (c_{1w}, c_{2w})$;

In Step ③, having collected C_w from each worker, the crowdsourcing server extracts c_{1w} and c_{2w} from each C_w, multiplies all the c_{1w}s and all the c_{2w}s respectively:

$$\tilde{C}_1 = \prod_{w \in W} c_{1w} \tag{6}$$

$$\tilde{C}_2 = \prod_{w \in W} c_{2w} \tag{7}$$

The combination of $\tilde{C}_1$ and $\tilde{C}_2$, $\tilde{C} = (\tilde{C}_1, \tilde{C}_2)$, is therefore transformed to the requester in step ④.

Algorithm 2 shows the tag aggregation performed by the crowdsourcing server.

In Step ⑤, the requester decrypts $\tilde{C}$ with the private key sk. According to the homomorphic encryption property of ElGamal, the requester can obtain the product of all the Tag_ws, TAG,

$$TAG = D(sk, \tilde{C}) = \prod_{w \in W} Tag_w \tag{8}$$

Because TAG is actually a product of tags represented by prime numbers, it can be uniquely factored in the form

$$TAG = \prod_{p_i \in P} p_i^{f_i} \tag{9}$$

Algorithm 2 Tag Aggregation (crowdsourcing server)

Require: C_ws from a set of workers
Ensure: $\tilde{C}$
1: **for** each C_w **do**
2: extracts c_{1w} and c_{2w}
3: **end for**
4: Computes $\tilde{C}_1$ and $\tilde{C}_2$ with Equation (6) and (7) respectively;
5: Outputs $\tilde{C} = (\tilde{C}_1, \tilde{C}_2)$;

Table 3
Worker distribution.

Task	t_1	t_2	...	t_i	...	$t_{\|T\|}$
Frequency	f_1	f_2	...	f_i	...	$f_{\|T\|}$

Table 4
Case study.

$Worker$	$Selectedtasks$	$Tags$	Tag_w
w_1	t_2, t_3	3, 5	15
w_2	t_1, t_2	2, 3	6
w_3	t_2	3	3
w_4	t_2, t_3	3, 5	15
w_5	t_2, t_3	3, 5	15

where P is the pre-defined tag set and each $f_i \geq 0$. Therefore, f_i is the number of workers who select the task with tag p_i. By factoring TAG, the requester can finally obtain the worker distribution of the tasks, F, as shown in Table 3.

Algorithm 3 shows the decryption and factorization performed by the crowdsourcing server.

Algorithm 3 Decryption and Factorization (Requester)

Require: $\tilde{C} = (\tilde{C}_1, \tilde{C}_2)$, $sk = x$
Ensure: F
1: Calculates $s = (\tilde{C}_1)^x \bmod q$
2: Calculates $TAG = \tilde{C}_2 \cdot s^{-1} \bmod q$;
3: Factorizes TAG to generate F;
4: Outputs F;

In conclusion, the cryptographic protocol can be summarized as follows:

i. Worker $w \rightarrow server : ID_w \parallel C_w$
ii. Server $\rightarrow requester : \tilde{C}$

3.2.2. A case study

Here we present an example to demonstrate the workflow of the proposed cryptographic protocol. Suppose there are five workers, w_1, $w_2, \ldots, w_5$, and four tasks, $t_1, t_2, \ldots, t_4$, in a crowdsourcing application. The tags for the tasks are 2, 3, 5, and 7. In Step ②, each worker selects their preferred tasks, as shown in Table 4, multiplies the tags for the selected tasks, and performs encryption with ElGamal. Take the worker w_1 as an example, $Tag_{w_1} = 3 \cdot 5 = 15$, and $C_{w_1} = E(p_k, 15) = (c_{1w_1}, c_{2w_1})$. C_{w_1}, C_{w_2}, ..., C_{w_5} are then submitted to the crowdsourcing server. In Step ③, the crowdsourcing server calculates $\tilde{C} = (\prod_{i=1}^{5} c_{1w_i}, \prod_{i=1}^{5} c_{2w_i})$, and $\tilde{C}$ is further transferred to the requester in Step ④. Finally, in Step ⑤, the requester decrypts the $\tilde{C}$ and obtains $TAG = D(sk, \tilde{C})$. TAG can be factored in the form $TAG = 2^1 \cdot 3^5 \cdot 5^3 \cdot 7^0$, which implies that one worker is potentially interested in performing task t_1, while t_2 has been selected by five workers, t_3 by three workers, and no one is willing to perform t_4.

Intuitively, the more workers that select a task, the higher the probability that the task will be accepted. However, the acceptance probability for any task selected by a few, or even no workers, can be improved by increasing the reward. Therefore, in Step ⑥, SecRSC uses the proposed pricing mechanism to adaptively specify a reward for each of the tasks, which is presented in the next subsection.

3.2.3. Implementation of the protocol

We set the size of the modulus q of ElGamal algorithm to be 1024 bits in this paper. When exploiting the multiplicative homomorphic property of ElGamal,

$$\prod_i E(m_i) = E(\prod_i m_i) \tag{10}$$

we must ensure that $\prod_i m_i < 2^{1024}$.

Therefore, in Step ③, having collected the cipher text, C_w, from each worker, the server first randomly divides all the cipher texts into $|G|$ groups with each group containing no more than k ciphertexts to ensure the product of the plaintexts corresponding to the ciphertexts is less than 2^{1024}, where k can be estimated given the number of tasks $|T|$ as follows.

According to the prime number theorem [35]:

$$\pi(n) \sim n/log(n) \tag{11}$$

where $\pi(n)$ is the number of primes not exceeding n, n can be deduced given $\pi(n)$. For example, when $|T| = \pi(n) = 5000$, we have more than 5000 primes given $n = 60000$. Suppose each worker, w, can select no more than l tasks, then $Tag_w < 60000 \cdot l$. Thus, the product of k plaintexts is:

$$\prod_{i=1}^{k} Tag_{w_i} < (60000 \cdot l)^k. \tag{12}$$

Let $(60000 \cdot l)^k \leq 2^{1024}$ and suppose $l = 10$, we have $k \leq 53$. Apparently, k also needs to be larger than 1. To implement the protocol efficiently, k should be as large as possible, say, $k = 50$ in this case.

Then each group is processed following the protocol independently so that the requester will finally obtain $|G|$ TAGs in Step ⑤. The global TAG can be aggregated by multiplying the $|G|$ TAGs. For example, suppose $TAG_1 = 2^1 \cdot 3^5 \cdot 5^3 \cdot 7^0$ and $TAG_2 = 2^2 \cdot 3^2 \cdot 5^1 \cdot 7^1$, then they can be aggregated to: $TAG = 2^3 \cdot 3^7 \cdot 5^4 \cdot 7^1$.

3.3. Reward assignment

In this subsection, we present two reward assignment approaches: *identical rate of return* (IRR) and an *optimized constraint function* (OCF), to adaptively specify rewards for each of the tasks.

3.3.1. Reward assignment with identical rate of return (IRR)

Reward allocation strategy can be various according to the specialty of different applications. For example, Liu et al. [36] proposed an accumulative reward allocation method based on modified ROC curve. Intuitively, in spatial crowdsourcing, a worker is more likely to accept nearby tasks with high returns. Thus, reward and distance are the key factors for a user when considering whether or not to accept a task. We use the *rate of return* (RoR) to evaluate the preference of a worker accepting a task.

Definition 1 (*Rate of Return*)**.** Given a task t with reward r, if the distance to perform t is d, the rate of return, $RoR = r/d$, is the average return for traveling unit distance to perform t.

The higher the RoR of a task, the higher the probability that a worker will accept the task. Let RoR_E be the average expectation of the workers on RoR, the tasks with RoR larger than RoR_E will be accepted. We aim to assign an appropriate reward for each task so that all the tasks have the identical rate of return, RoR_I. When the total budget R is larger than the lower bound which ensures $RoR_I = RoR_E$, each task has a RoR greater than RoR_E, leading to the highest task acceptance rate. RoR_E can be estimated by market investigation according to economic principles.

Given a task t with a frequency f, we suppose the f workers are scattered around the task point following either a Gaussian distribution or uniform distribution. A higher f implies a higher probability that a user is closer to the task point. In other words, to some extent, f is inversely proportional to the minimal distance of the workers to the task point. We use $\hat{d} = 1/ln(f+2)$ to simulate the proportion of f to the distance d. Therefore, the RoR of task t is proportional to

$$RoR_t = r \cdot ln(f+2) \tag{13}$$

We aim to tune the reward distribution to achieve an identical RoR for all the tasks, namely, for any pair of tasks t_i and t_j, $r_i \cdot ln(f_i+2) = r_j \cdot ln(f_j+2)$.

Given the total budget R, we denote the proportion of r_i to R as $x_i = r_i/R$, $i = 1, 2, \ldots, |T|$. Thus, we have the following equation set

$$\begin{cases} x_1 + x_2 + \cdots + x_{|T|} = 1 \\ \frac{x_1}{\hat{d}_1} = \frac{x_2}{\hat{d}_2} = \cdots = \frac{x_{|T|}}{\hat{d}_{|T|}}, \end{cases} \tag{14}$$

which can be represented by

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \frac{1}{\hat{d}_1} & -\frac{1}{\hat{d}_2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\hat{d}_2} & -\frac{1}{\hat{d}_3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \frac{1}{\hat{d}_{|T|-1}} & -\frac{1}{\hat{d}_{|T|}} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{|T|} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix}. \tag{15}$$

Solving Eq. (15), we get

$$x_i = \frac{1}{\frac{1}{\hat{d}_i} \cdot \sum_{j=1}^{|T|} \hat{d}_j}. \tag{16}$$

Therefore, the reward of each task t_i can be calculated by

$$r_i = \frac{R}{ln(f_i+2) \cdot \sum_{i=1}^{|T|} \frac{1}{ln(f_i+2)}} \tag{17}$$

3.3.2. Reward assignment with optimized constraint function (OCF)

In the aforementioned IRR, the reward of each task is computed directly according to the worker distribution of tasks without any constraint. In most practical applications, a requester generally tends to limit the reward for each task within a fixed interval. We present an OCF to adaptively compute the reward for each task. The function should meet two requirements: (1) the reward r output by the function should be limited by a pre-specified interval, $[r_{min}, r_{max}]$; (2) the reward r should decrease with an increase in the frequency f.

Therefore, we propose the following function to assign the reward r_i for each task t_i:

$$r_i = (r_{max} - r_{min}) \cdot e^{-k \cdot f_i} + r_{min} \tag{18}$$

where $k \geq 0$ is the parameter to be optimized. Given a fixed k, when $f_i = 0$, r_i has the maximum value, $r_i = r_{max}$. With the increase of f_i, $e^{-k \cdot f_i}$ decreases so that r_i decreases accordingly. When $f_i \to +\infty$, we have $r_i \to r_{min}$.

Given the worker distribution shown in Table 3 and the total reward R, we need to find an optimized k to calculate each r_i that meets the constraint $\sum_{i=1}^{|T|} r_i = R$.

Let $r_{max} = a \cdot \frac{R}{|T|}$ and $r_{min} = b \cdot \frac{R}{|T|}$, apparently, $a > 1 > b$, we have

$$(a \cdot \frac{R}{|T|} - b \cdot \frac{R}{|T|}) \sum_{i=1}^{|T|} e^{-k \cdot f_i} + b \cdot R = R \tag{19}$$

namely,

$$\sum_{i=1}^{|T|} e^{-k \cdot f_i} = \frac{1-b}{a-b} \cdot |T| \tag{20}$$

Consider the following function

$$g(k) = \sum_{i=1}^{n} e^{-k \cdot f_i} - \frac{1-b}{a-b} \cdot |T| \tag{21}$$

because $e^{-k \cdot f_i}$ decreases with the increase of k when $k \geq 0$, the function $g(k)$ is a monotonically decreasing and continuous function. In addition, $g(k)$ is greater than 0 when $k = 0$. Therefore, there must exist a $k > 0$ that satisfies $g(k) = 0$. We can use the bisection method to efficiently search for the optimized k. The pseudocode for this search algorithm is shown in Algorithm 4.

Algorithm 4

Require: $k_1 = 0$, $error$, the worker distribution F, r_{min}, r_{max}, total reward R
Ensure: The optimized k
1: Find a k_2 which satisfies $g(k_2) < 0$;
2: **while** $k_2 - k_1 > error$ **do**
3: $k_m = \frac{k_1+k_2}{2}$;
4: **if** $g(k_1) \cdot g(k_m) > 0$ **then**
5: $k_1 = k_m$;
6: **else**
7: $k_2 = k_m$;
8: **end if**
9: **end while**
10: Output k_1;

4. Analysis of SecRSC

4.1. Privacy analysis

The proposed SecRSC can completely preserve a worker's location privacy. It is computationally infeasible for either the crowdsourcing server or the requester to deduce the tasks selected by a worker.

We formally define location privacy with the following game. Given a spatial task set T, the game between an adversary $\mathcal{A}$ and a challenger C consists of the following steps:

(1) Adversary $\mathcal{A}$ randomly chooses two distinct groups of workers with their preferred tasks, $W_0 = \{T_{u_1}, T_{u_2}, \ldots, T_{u_j}\}$, $W_1 = \{T_{v_1}, T_{v_2}, \ldots, T_{v_k}\}$, such that the TAGs of W_0 and W_1 are same, where T_{u_j} is the preferred tasks of the worker u_j, T_{v_k} is the preferred tasks of the worker v_k. Then, Adversary $\mathcal{A}$ sends W_0 and W_1, as well as her public key pk, to Challenger C.
(2) Challenger C tosses a coin and decides $b = 0$ if the toss is heads, $b = 1$ otherwise. Then, challenger C encrypts each task set in W_b and calculates the product $\tilde{C}$. $\tilde{C}$ is then sent back to Adversary $\mathcal{A}$.
(3) Adversary $\mathcal{A}$ experiments with the received $\tilde{C}$ and finally determines $b' \in \{0, 1\}$ where $W_{b'} \in \{W_0, W_1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. The advantage for Adversary $\mathcal{A}$ in this game is defined as

$$Adv(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

Definition 2 (*Location Privacy*)**.** In the SecRSC Framework, each worker has location privacy if, for any probabilistic polynomial time adversary $\mathcal{A}$, $Adv(\mathcal{A})$ is a negligible function, where the probability space is over the coin flips of the challenger and the adversary.

Location privacy ensures that the requester and the crowdsourcing server cannot deduce any information about the location of workers.

Suppose the crowdsourcing server is Adversary $\mathcal{A}$. Without colluding with the requester, it cannot obtain the requester' private key and further decrypt any ciphertext submitted by a worker. Let the advantage for breaking the ElGamal cryptosystem be ϵ, the $\Pr[b' = b]$ will be $1/2+\epsilon$. Thus the advantage $Adv(\mathcal{A}) = |1/2 + \epsilon - 1/2| = \epsilon$. If the ElGamal cryptosystem is semantically secure, ϵ will be negligible, and the SecRSC has location privacy according to Definition 2.

Suppose the requester is the adversary $\mathcal{A}$, she also cannot win the game without colluding with the server. Given $\tilde{C}$, although the

requester can decrypt $\tilde{C}$ by her private key and obtain the TAG, she cannot determine whether the TAG is generated from W_0 or W_1 because W_0 and W_1 have the same TAG. There is a $50 - -50$ chance that the requester gets the right b', namely, $\Pr[b' = b] = 1/2$. The advantage for the requester in this game is $Adv(\mathcal{A}) = |1/2 - 1/2| = 0$. Thus, the SecRSC also has location privacy for a semi-honest requester according to Definition 2. On the other hand, if the requester colludes with the server, she can ask the server for the ciphertext of each worker, and further obtain each entry in W_0 and W_1. Then the requester will definitely win the game.

Note that the homomorphic encryption-based method will be invalid in the extreme case that only one worker is involved in a spacial crowdsourcing project. Because there is only one encrypted message from the worker, the requester will finally obtain the message by decrypting the cipher text. However, this extreme case should be ignored because, in real-world, a special crowdsourcing project generally involves hundreds of tasks and widely distributed workers.

4.2. Performance analysis

There are two pairs of computations in SecRSC: (1) encryption and decryption; (2) multiplication and factorization.

Encryption and decryption are performed in Steps ② and ⑤. Each worker needs to compute one ElGamal encryption in Algorithm 1 (line $2 \sim 3$). The requester needs to compute one ElGamal decryption in Algorithm 3 (line $1 \sim 2$). Given pre-computed keys, the computation of one encryption and decryption consumes negligible time, as presented in Section 5.2.

Two multiplications are performed in Steps ② and ③. Specifically, each worker multiplies all the selected tags in Algorithm 1 (line 1), while the server multiplies all the ciphertext from the workers in Algorithm 2 (line 4). Factorization is performed in Algorithm 3 (line 3) by the requester in Step ⑤. Generally, when the number of workers and tasks is large, it leads to relatively expensive computing costs. For example, in our experiment, given 10,000 workers and 10,000 tasks, factorization consumes approximately $20s$.

Therefore, we consider the computation cost of multiplication and factorization while ignoring encryption and decryption because the latter is much cheaper than the former. The process of key generation for the ElGamal cryptosystem is also ignored because it can be pre-computed by a requester before publishing tasks.

In the most extreme case, we assume that each worker selects all the tasks $|T|$ as their preferred tasks. Therefore, for a worker, the computation complexity of multiplication is $\mathcal{O}(|T|)$. For the crowdsourcing server, the computation complexity of multiplication is $\mathcal{O}(|W|)$. For the requester, the computation complexity of factorization is $\mathcal{O}(|T| \cdot |W|)$. While in practice, the number of tasks selected by a worker is generally much less than $|T|$, say, $3 \sim 5$.

5. Experiments

In this section, we evaluate the performance of SecRSC and the algorithms by answering the following questions:

- *How does SecRSC perform in terms of task acceptance rates?* We investigated the performance of SecRSC using the worker selected mode as a benchmark, which applies a uniform strategy in which the rewards are allocated equally among all the tasks. We demonstrate that the proposed reward assignment strategies, IRR and OCF, can significantly improve task acceptance rates.
- *How do the parameters of SecRSC impact the proposed framework?* We conducted the experiments in different settings with various parameters, such as the RoR threshold and the number of tasks and workers, to evaluate the robustness of SecRSC. We also investigate the variations in execution time and communication costs as the scale of the crowdsourcing application increases.

The experiments were conducted on a machine with a 3.6 GHz Intel Core i7 CPU and 8GB RAM in Windows 7. The ElGamal cryptosystem was implemented using the Python cryptography toolkit PyCrypto[3] with the size of the modulus 1024 bits.

5.1. Datasets and measurement

5.1.1. Datasets

We used the real-world datasets Gowalla[4] and T-Drive[5] in our experiments. Gowalla is a social network dataset containing users and their check-in location information. T-Drive includes location information collected by taxi drivers in Beijing, China.

5.1.2. Evaluation metrics

Task Acceptance Rate (TAR) is the most widely used metric to evaluate the performance of a crowdsourcing application. It measures the extent to which a crowdsourcing application can completely fulfill its tasks [1,15]. *TAR* is the ratio of the number of accepted tasks to the total number of tasks in a crowdsourcing project,

$$TAR = \frac{number\ of\ accepted\ tasks}{total\ number\ of\ tasks}. \tag{22}$$

In our experiment, we investigated the performance of SecRSC with different values of *minimal RoR*, denoted by Min_{RoR} on the assumption that a worker will refuse a task if the RoR is lower than Min_{RoR}. This assumption is similar to the definition of *maximum travel distance* (MTD) in [1] that states a task will be refused if the distance is larger than the MTD. For simplicity, we also assume that each worker can select no more than 10 tasks. We successively set the Min_{RoR} to the 5, 7.5, 10, 12.5, 15 quantile values of the RoRs in worker selected mode with a uniform reward assignment strategy. In each case, the *TAR*s of the worker selected mode were 95%, 92.5%, 90%, 87.5% and 85%, respectively.

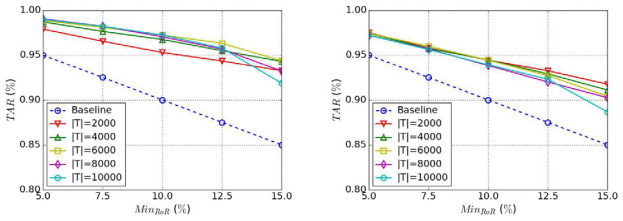
5.2. Experiment results and comparisons

5.2.1. Performance on TAR

To evaluate the performance of SecRSC with the proposed reward assignment approaches, IRR and OCF, we first fixed the ratio of the number of workers to the number of tasks $|W|/|T| = 2$, and investigated the variations in *TAR* with different Min_{RoR}, as well as increasing numbers of workers and tasks.

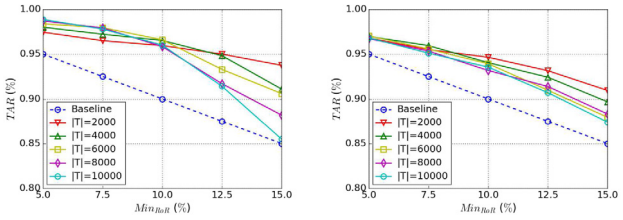
Fig. 2 shows the results using the Gowalla dataset. Compared to the uniform reward assignment approach used in worker selected mode, denoted by *Uniform*, both IRR and OCF achieved a much higher *TAR* than that of *Uniform*. For example, Fig. 2(a) illustrates that, when $Min_{RoR} = 10\%$ and $|T| = 2000, 4000, 6000, 8000, 10000$ with $|W| = 4000, 8000, 12000, 16000, 20000$ correspondingly, the *TAR*s of IRR were 95.3%, 96.8%, 97.3%, 97.1%, and 97.3%, while the *TAR* of *Uniform*, as the benchmark, was 90%. Fig. 2(b) shows a similar trend for *TAR*s when OCF is applied. For example, in the same scenarios above, the *TAR*s of OCF were 94.5%, 94.4%, 94.5%, 93.8%, and 93.9%. We also observed that both IRR and OCF are robust enough that the *TAR*s were insensitive to the number of workers and tasks. For example, when $Min_{RoR} = 10\%$ and $|W|/|T| = 2$, the variations in *TAR* achieved by IRR and OCF were 5.38×10^{-5} and 8.13×10^{-6}, respectively, with the increase of $|T|$ from 2000 to 10000. It turns out that the proposed SecRSC is efficient for crowdsourcing applications with various scales.

We obtained similar results from the T-drive dataset. As shown in Fig. 3, both IRR and OCF significantly outperformed the *Uniform* strategy, and all the results for *TAR* were above the baseline. For example, when $Min_{RoR} = 10\%$, $|T| = 2000, 4000, 6000, 8000, 10000$ and $|W|/|T| = 2$, the *TAR*s of IRR were 96.0%, 96.5%, 96.6%, 95.8%, and



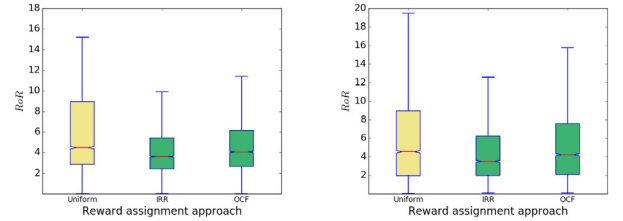
(a) IRR (b) OCF

Fig. 2. *TAR* over varying Min_{RoR} on Gowalla.



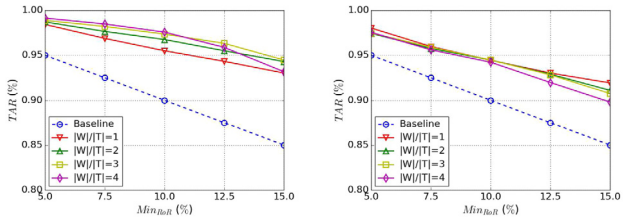
(a) IRR (b) OCF

Fig. 3. *TAR* over varying Min_{RoR} on T-Drive.



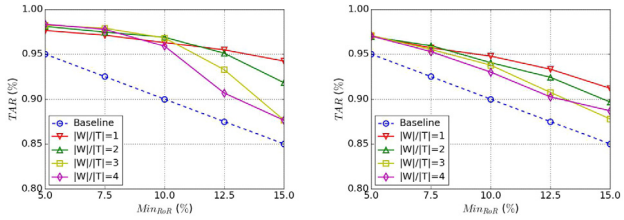
(a) Gowalla (b) T-Drive

Fig. 4. Distribution of *RoR*.



(a) IRR (b) OCF

Fig. 5. *TAR* over varying Min_{RoR} and $|W|/|T|$ on Gowalla.



(a) IRR (b) OCF

Fig. 6. *TAR* over varying Min_{RoR} and $|W|/|T|$ on T-Drive.

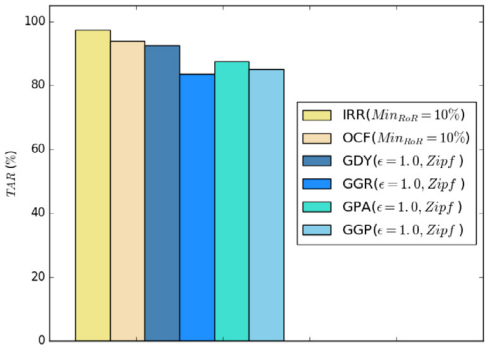


Fig. 7. *TAR* comparisons with existing methods.

96.0%, and the *TAR*s of OCF were 94.7%, 94.1%, 93.9%, 93.2%, 93.6%, which are all much higher than the 90% benchmark.

To look inside IRR and OCF and analyze how each allocation scheme works, we studied the distribution of *RoR* with the three reward assignment approaches. It turns out that, compared to the *Uniform*, both IRR and OCF compress the distribution of *RoR* into a more narrow interval so that the number of tasks with a lower *RoR* than Min_{RoR} decreases significantly. For instance, the *RoR* distribution generated by the three reward assignment methods when $|T| = 8000$ and $|W| = 16000$ are presented in the boxplot in Fig. 4. Take the T-drive dataset as an example, given the threshold $Min_{RoR} = 10\%$, for *Uniform*, IRR, and OCF, the number of tasks with RoR less than Min_{RoR} were 800, 322, and 546, respectively, so that IRR and OCF achieve higher *TAR*s than that of *Uniform*.

We also investigated the performance of SecRSC on *TAR*s with a varying ratio $|W|/|T|$ to simulate various crowdsourcing applications. Figs. 5 and 6 show the trend of *TAR*s on the two datasets when the ratio $|W|/|T|$ was varied from 1 to 4 with a fixed $|T| = 4000$. The results show that the proposed IRR and OCF outperformed the *Uniform* strategy in all cases. For example, when the Min_{RoR} was 10%, the *TAR* of IRR and OCF on the *Gowalla* dataset were 96.8% and 94.4% on average, improvements of 6.8% and 4.4%, respectively, compared to the *Uniform* strategy. For the T-Drive dataset, the improvements were 6.4% and 3.7%, respectively. It turns out that SecRSC is viable for a variety of crowdsourcing applications.

Comparing the two proposed reward assignment approaches, we observed that, on average, IRR outperformed OCF by 1.6% in terms of *TAR*. However, without any constraint on reward, the reward assigned to some tasks by IRR might be extremely small, even close to zero. OCF produced a suboptimal reward distribution with a slight compromise to the *TAR* due to the dual minimal and maximal reward constraints. It is more logical and practical for real reward-based crowdsourcing applications to only impose a lower bound on the reward for each task.

While the proposed SecRSC outperformed the worker selected mode in terms of *TAR*, SecRSC achieved a slightly better performance than the solutions based on the server assigned mode, such as GDY, G-PA, G-GR, and G-GP proposed in [1] and [37]. Fig. 7 illustrates a comparison of these approaches in terms of *TAR* on the Gowalla dataset. Further, SecRSC provides a stronger privacy guarantee than these solutions by completely preserving worker location privacy. Due to the cryptographic protocol, an adversary is unable to deduce any information about the tasks selected by a worker, or the location of workers.

5.2.2. Execution time and communication overhead

The execution time, denoted by T_{exe}, consists of four parts,

$$T_{exe} = T_{enc} + T_{dec} + T_{aggr} + T_{fac} \tag{23}$$

[3] https://www.dlitz.net/software/pycrypto/.
[4] http://snap.stanford.edu/data/loc-gowalla.html.
[5] http://research.microsoft.com/en-us/projects/tdrive/.

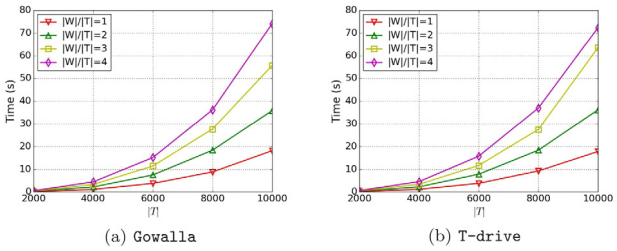


Fig. 8. Runtime estimation of prime factorization.

Table 5
Communication overhead on Gowalla dataset.

$\|T\|$	Communication cost (kB)			
	$\frac{\|W\|}{\|T\|}=1$	$\frac{\|W\|}{\|T\|}=2$	$\frac{\|W\|}{\|T\|}=3$	$\frac{\|W\|}{\|T\|}=4$
2000	309.4	611.7	943.9	1233.4
4000	942.8	1879.1	2797.7	3760.2
6000	1772.4	3509.0	5250.3	6934.5
8000	2610.4	5195.7	7735.5	10481.6
10000	3559.6	7034.0	10589.6	14165.0

where T_{enc} is the time of encryption by each worker, T_{aggr} is the time of tag aggregation by the crowdsourcing server, T_{dec} and T_{fac} are the time of decryption and factorization by the requester.

In practice, T_{enc}, T_{dec} and T_{aggr} are negligible while T_{fac} is the most time-consuming part. Specifically, encrypting and decrypting a number of about 2^{1000} consumed 0.06 ms and 3.5 ms, respectively, in our experiments. And the tag aggregation for 10,000 ciphertexts was completed within 25 ms. Thus, we only further investigates T_{fac} and ignored the other parts of T_{exe}.

Fig. 8 shows the trend for T_{fac} with an increasing $|T|$ while varying $|W|/|T|$. It can be observed that the factorization time increased quadratically with an increase in $|T|$ for both the Gowalla and T-Drive datasets. Given a fixed $|T|$, a larger $|W|/|T|$ always led to a longer factorization time. However, the runtime was no more than 75s when the number of tasks and workers was 10,000 and 40,000. It turns out that the proposed SecRSC is efficient for most crowdsourcing applications.

We investigated the communication overhead in SecRSC by recording all of the messages transmitted among the workers, the crowdsourcing server, and the requester. The results show that the proposed cryptographic protocol incurs an adequately low communication overhead. For example, Table 5 shows the results on the *Gowalla* dataset with respect to different numbers of tasks $|T|$ and an increasing $|W|/|T|$.

It can be observed that the communication cost varied from 309.4kB to 14165.0kB when $|T|$ was changed from 2000 to 10,000 and $|W|/|T|$ from 1 to 4. The maximum cost was no more than 15MB with 40,000 workers and 10,000 tasks.

Compared to other cryptography-based approaches, such as HESI proposed in [20], SecRSC achieves significantly better performance in terms of computation and communication costs.

6. Conclusions

Spatial crowdsourcing has become a promising paradigm for collecting and analyzing spatial–temporal data. However, the existing server assigned and worker selected modes for implementing crowdsourcing are unable to provide an adequately rigid privacy guarantee while ensuring a high task acceptance rate. Concurrently, concerns about personal privacy when participating in crowdsourcing schemes are increasing.

In this paper, we propose a novel framework, SecRSC, for reward-based crowdsourcing. A cryptographic protocol based on homomorphic encryption is designed to privately investigate worker preferences for performing specific crowdsourcing tasks. We use pairwise relatively prime numbers as a tag for each task to take advantage of the unique products and factors prime numbers provide to fully preserve the privacy of worker locations. We believe the proposed cryptographic protocol could also be applied to other scenarios, such as private elections or private questionnaires. Two reward assignment approaches, IRR and OCF, are thus presented for adaptively specifying rewards for each task. Given a fixed budget, the 'hot' tasks are given a relatively lower reward, while the 'outliers' that might not be accepted are given attractive returns. Experimental results on real-world datasets show that, by adaptively tuning the reward for each of the tasks, the proposed reward assignment approaches significantly outperform the uniform assignment approach which is used in worker selected mode to determine the *TAR*.

Unlike existing privacy-preserving solutions in server assigned mode, the proposed SecRSC aims to protect the tasks selected by a worker instead of the worker's real location. In other words, a worker never even needs to disclose his/her real location in SecRSC. Further, SecRSC provides a sufficient privacy guarantee with low computing and communication overhead for various crowdsourcing applications.

Future work will extend the proposed framework to scenarios where tasks have spatiotemporal relationships. Frameworks for crowdsourcing with private trajectories will also be explored.

Acknowledgments

This work is supported by the Ministry of Education, Humanities, and Social Science Project of China under Grant No. 19A10520035, the Fundamental Research Funds for the Central Universities, Zhongnan University of Economics and law under Grant No. 201911414.

References

[1] H. To, G. Ghinita, C. Shahabi, A framework for protecting worker location privacy in spatial crowdsourcing, Proc. VLDB Endow. 7 (10) (2014) 919–930, http://dx.doi.org/10.14778/2732951.2732966.

[2] L. Pournajaf, D.A. Garcia-Ulloa, L. Xiong, V. Sunderam, Participant privacy in mobile crowd sensing task management: a survey of methods and challenges, SIGMOD Rec. 44 (4) (2016) 23–34, http://dx.doi.org/10.1145/2935694.2935700, http://doi.acm.org/10.1145/2935694.2935700.

[3] L. Kazemi, C. Shahabi, Geocrowd: enabling query answering with spatial crowdsourcing, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, in: SIGSPATIAL '12, ACM, New York, NY, USA, 2012, pp. 189–198, http://dx.doi.org/10.1145/2424321.2424346, http://doi.acm.org/10.1145/2424321.2424346.

[4] M.E. Andrés, N.E. Bordenabe, K. Chatzikokolakis, C. Palamidessi, Geo-indistinguishability: differential privacy for location-based systems, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer; Communications Security, in: CCS '13, ACM, New York, NY, USA, 2013, pp. 901–914, http://dx.doi.org/10.1145/2508859.2516735, http://doi.acm.org/10.1145/2508859.2516735.

[5] B. Hoh, M. Gruteser, H. Xiong, A. Alrabady, Enhancing security and privacy in traffic-monitoring systems, IEEE Pervas. Comput. 5 (4) (2006) 38–46, http://dx.doi.org/10.1109/MPRV.2006.69.

[6] Y. Matsuo, N. Okazaki, K. Izumi, Y. Nakamura, T. Nishimura, K. Hasida, H. Nakashima, Inferring long-term user properties based on users' location history, in: Proceedings of the 20th International Joint Conference on Artifical Intelligence, in: IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2159–2165, http://dl.acm.org/citation.cfm?id=1625275.1625624.

[7] N. Sun, J. Zhang, P. Rimba, S. Gao, L.Y. Zhang, Y. Xiang, Data-driven cybersecurity incident prediction: a survey, IEEE Commun. Surv. Tutor. 21 (2) (2019) 1744–1772, http://dx.doi.org/10.1109/COMST.2018.2885561.

[8] B. Agir, T.G. Papaioannou, R. Narendula, K. Aberer, J.-P. Hubaux, User-side adaptive protection of location privacy in participatory sensing, Geoinformatica 18 (1) (2014) 165–191.

[9] M.F. Mokbel, C.-Y. Chow, W.G. Aref, The new casper: query processing for location services without compromising privacy, in: Proceedings of the 32nd International Conference on Very large Data Bases, VLDB Endowment, 2006, pp. 763–774.

[10] L. Pournajaf, L. Xiong, V. Sunderam, X. Xu, Stac: spatial task assignment for crowd sensing with cloaked participant locations, in: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, in: SIGSPATIAL '15, ACM, New York, NY, USA, 2015, pp. 90:1–90:4, http://dx.doi.org/10.1145/2820783.2820788, http://doi.acm.org/10.1145/2820783.2820788.

[11] M. Gruteser, D. Grunwald, Anonymous usage of location-based services through spatial and temporal cloaking, in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, 2003, pp. 31–42.
[12] R. Dewri, Local differential perturbations: location privacy under approximate knowledge attackers, IEEE Trans. Mob. Comput. 12 (12) (2013) 2360–2372, http://dx.doi.org/10.1109/TMC.2012.208.
[13] C. Dwork, Differential privacy, in: ICALP'06: Proceedings of the 33rd International Conference on Automata, Languages and Programming, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 1–12, http://dx.doi.org/10.1007/11787006_1.
[14] H. To, G. Ghinita, L. Fan, C. Shahabi, Differentially private location protection for worker datasets in spatial crowdsourcing, IEEE Trans. Mob. Comput. 16 (4) (2017) 934–949.
[15] P. Xiong, L. Zhang, T. Zhu, Reward-based spatial crowdsourcing with differential privacy preservation, Enterprise Inf. Syst. 11 (10) (2017) 1500–1517.
[16] A. Khoshgozaran, C. Shahabi, Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy, Adv. Spatial Temporal Databases (2007) 239–257.
[17] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, K.-L. Tan, Private queries in location based services: anonymizers are not necessary, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ACM, 2008, pp. 121–132.
[18] T. Shu, Y. Chen, J. Yang, Protecting multi-lateral localization privacy in pervasive environments, IEEE/ACM Trans. Netw. 23 (5) (2015) 1688–1701, http://dx.doi.org/10.1109/TNET.2015.2478881.
[19] S. Tian, Y. Cai, Q. Zheng, A hybrid approach for privacy-preserving processing of knn queries in mobile database systems, in: Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, in: CIKM '13, ACM, New York, NY, USA, 2013, pp. 1161–1164, http://dx.doi.org/10.1145/2505515.2507814, http://doi.acm.org/10.1145/2505515.2507814.
[20] B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, W. Qiu, Protecting location privacy in spatial crowdsourcing using encrypted data., in: EDBT, 2017, pp. 478–481.
[21] I. Damgård, M. Jurik, A generalisation, a simpli.cation and some applications of paillier's probabilistic public-key system, in: K. Kim (Ed.), Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 119–136.
[22] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Advances in Cryptology — EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 223–238, http://dx.doi.org/10.1007/3-540-48910-X_16.
[23] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Proceedings of CRYPTO 84 on Advances in Cryptology, Springer-Verlag New York, Inc., New York, NY, USA, 1985, pp. 10–18, http://dl.acm.org/citation.cfm?id=19478.19480.
[24] S. Goldwasser, S. Micali, Probabilistic encryption & how to play mental poker keeping secret all partial information, in: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, in: STOC '82, ACM, New York, NY, USA, 1982, pp. 365–377, http://dx.doi.org/10.1145/800070.802212, http://doi.acm.org/10.1145/800070.802212.
[25] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, in: STOC '09, ACM, New York, NY, USA, 2009, pp. 169–178, http://dx.doi.org/10.1145/1536414.1536440, http://doi.acm.org/10.1145/1536414.1536440.
[26] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping, ACM Trans. Comput. Theory 6 (3) (2014) 13:1–13:36, http://dx.doi.org/10.1145/2633600, http://doi.acm.org/10.1145/2633600.
[27] H. To, G. Ghinita, C. Shahabi, PrivGeoCrowd: a toolbox for studying private spatial crowdsourcing, in: Proceedings of the 31st IEEE International Conference on Data Engineering.
[28] E. Magkos, Cryptographic approaches for privacy preservation in location-based services: a survey, in: IJITSA, vol. 4, 2011, pp. 48–69.
[29] Y. Shen, L. Huang, L. Li, X. Lu, S. Wang, W. Yang, Towards preserving worker location privacy in spatial crowdsourcing, in: 2015 IEEE Global Communications Conference, GLOBECOM, 2015, pp. 1–6, http://dx.doi.org/10.1109/GLOCOM.2015.7416965.
[30] C. Dwork, A firm foundation for private data analysis, Commun. ACM 54 (1) (2011) 86–95, http://dx.doi.org/10.1145/1866739.1866758, http://doi.acm.org/10.1145/1866739.1866758.
[31] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, T. Yu, Differentially private spatial decompositions, in: 2012 IEEE 28th International Conference on Data Engineering, 2012, pp. 20–31, http://dx.doi.org/10.1109/ICDE.2012.16.
[32] D. Lee,
[33] R. Lu, K. Heung, A.H. Lashkari, A.A. Ghorbani, A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot, IEEE Access 5 (2017) 3302–3312, http://dx.doi.org/10.1109/ACCESS.2017.2677520.
[34] V. Smith, C.-K. Chiang, M. Sanjabi, A.S. Talwalkar, Federated multi-task learning, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 4424–4434, http://papers.nips.cc/paper/7029-federated-multi-task-learning.pdf.
[35] R. Crandall, C. Pomerance, Prime Numbers: A Computational Perspective, Springer, 2005.
[36] W. Liu, J. Cai, Y. Wang, Q. Chen, D. Tang, Mix-flow scheduling using deep reinforcement learning for software-defined data-center networks, Internet Technol. Lett. (2019) http://dx.doi.org/10.1002/itl2.99.
[37] N. Li, W. Yang, W. Qardaji, Differentially private grids for geospatial data, in: Proceedings of the 2013 IEEE International Conference on Data Engineering, ICDE 2013, in: ICDE '13, IEEE Computer Society, Washington, DC, USA, 2013, pp. 757–768, http://dx.doi.org/10.1109/ICDE.2013.6544872.