Preserving Location Privacy for Outsourced Most-Frequent Item Query in Mobile Crowdsensing

Songnian Zhang, Suprio Ray, *Member, IEEE*, Rongxing Lu, *Fellow, IEEE*, Yandong Zheng, and Jun Shao

***Abstract*—The emergence of mobile crowdsensing (MCS) has provided us with unprecedented opportunities for both sensing coverage and data transmission. However, in many MCS applications, the MCS workers are usually required to report the location information of the assigned tasks, which inevitably reveals the workers' location information, even trajectories, and severely impedes the popularization of the MCS system. It is believed that the query on the most-frequent location, e.g., querying the most congested location over a period in a city, is one of the most popular statistics queries in the MCS system, but it may disclose workers' location information. To address the issue, in this article, we propose a location privacy-preserving scheme for outsourced most-frequent item query in the MCS system, where two noncollusive semi-trusted cloud servers cooperatively handle the most-frequent item query. Specifically, by employing our pseudonymization mechanism, transposition cipher, ciphertext packing technique, and order-preserving merge function, our proposed scheme can efficiently answer the most-frequent item query while ensuring the privacy of both workers' personal information and query results. Detailed security analysis shows that our proposed scheme is privacy-preserving. In addition, extensive experiments are conducted, and the results show that our proposed scheme outperforms alternative schemes in terms of computational costs and communication overhead.**

***Index Terms*—Location privacy, mobile crowdsensing (MCS), most-frequent item query, privacy preserving.**

I. Introduction

With the proliferation of mobile devices and wireless networks, the paradigm of mobile crowdsensing (MCS), which employs workers to sense physical data, has recently received considerable attention in both industry and academia [1]–[4]. In the past years, plenty of efforts have been devoted to developing various real-world MCS applications, such as noise pollution assessment [5], water levels monitoring [6], and traffic information sharing [7]. Within most of the MCS applications, abundant location data will be collected and further utilized for various location-based services. Among these location-based services, the most-frequent item query service over location data is very common, e.g., finding the most congested location in a city (hereafter, we will use "the most-frequent item" and "the most-frequent location" interchangeably). Specifically, given a set of location data $\{l_1, l_2, \ldots, l_n\}$, in which l_i has a frequency $\mathsf{freq}(l_i)$, a most-frequent location query is to find a location $l_{\max}$ that has the largest frequency, i.e., $\mathsf{freq}(l_{\max}) \geq \mathsf{freq}(l_i)$, for $i = 1, \ldots, n$. To clearly present the most-frequent location query in the MCS system, an example is given as follows.

TABLE I
Congestion Data Set Example

id	eventType	location	timeStamp
w_1	Traffic Congestion	(46.13, -73.2)	1593631555
w_2	Traffic Congestion	(46.11, -73.1)	1593621420
w_3	Traffic Congestion	(46.11, -73.1)	1593629413
w_2	Traffic Congestion	(46.12, -73.5)	1593542218

Example 1: Suppose the MCS platform accumulates a traffic congestion information data set from workers, and each data record includes an id (worker's identity information), an event type, a location, and a timestamp. Here, we list a few tuples of the data set in Table I.

In the table, w_i ($i = 1, 2, 3$) indicates the worker's identity and is used to trace a worker for incentives. An authorized query user would like to know which location is the most congested in a time window, so as to plan his/her route in advance. In order to obtain the most frequent location, the query user can launch an SQL query

```
SELECT location, count(*) AS count
FROM congestion
WHERE 1593610000<timeStamp<1593650000
GROUP BY location ORDER BY count DESC LIMIT 1.
```

Then, the MCS platform processes the query and returns the most-frequent location (46.11, −73.1) and the corresponding frequency value: 2.

Meanwhile, due to the rapid increase in the data volume, the MCS system tends to outsource the collected data and the corresponding most-frequent item query service to a cloud for the performance and cost considerations. However, the cloud is not fully trusted, and the leakage of the location data may

Manuscript received November 1, 2020; revised January 9, 2021; accepted January 27, 2021. Date of publication February 2, 2021; date of current version May 21, 2021. This work was supported in part by NSERC Discovery under Grant 04009 and Grant LMCRF-S-2020-03; in part by ZJNSF under Grant LZ18F020003; and in part by NSFC under Grant U1709217 and Grant 61972304. *(Corresponding author: Rongxing Lu.)*

Songnian Zhang, Suprio Ray, Rongxing Lu, and Yandong Zheng are with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: szhang17@unb.ca; sray@unb.ca; rlu1@unb.ca; yzheng8@unb.ca).

Jun Shao is with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China (e-mail: chn.junshao@gmail.com).

Digital Object Identifier 10.1109/JIOT.2021.3056442

incur a series of attacks, such as inferring movements, daily behaviors, and most likely appeared locations of a person, which can seriously threaten personal privacy even safety [8]. In the MCS system, to preserve location privacy over the cloud, privacy-preserving techniques are commonly involved in protecting the location data before reporting them, such as the clocking technique, adding dummy points, and the differential privacy technique [9]. However, the existing techniques, which can be used in our query scenario, sacrifice accuracy to protect the location privacy [10]. Therefore, it is still challenging to achieve the accurate query result for the most-frequent location query over the cloud while ensuring the privacy of the workers' location data $\{l_i\}_{i=1}^n$ and the query result $(l_{\max}, \mathsf{freq}(l_{\max}))$.

Aiming at the above challenge, in this article, we propose a privacy-preserving most-frequent location query scheme that can support the accurate query result. In our scheme, the cloud collects location data encrypted by the workers and conducts queries while preserving the privacy of the workers' location information and the query results. To prevent the cloud from linking the encrypted location data and further obtaining the query result over them directly, we employ the semantic-secure encryption to encrypt the location data, which undoubtedly creates difficulties for the cloud to answer the query. To tackle them, we design the privacy-preserving most-frequent location query scheme in a two-server setting [11]. In addition, our scheme protects the workers' real identities by a novel pseudonymization technique that can guarantee the unlinkable identities for one worker in different periods. Specifically, the main contributions of this work are threefold.

1) We propose a privacy-preserving scheme to answer the most-frequent location query in the MCS system, which can protect the workers' personal information and query results. In the scheme, we adopt the Paillier cryptosystem to encrypt the sensed location data and design a new pseudonymization mechanism to hide the workers' identities while ensuring the traceability. To preserve the privacy of query results, we utilize the transposition cipher [12] to protect the locations' frequency values and the self-blinding property of the Paillier cryptosystem to break the link between Paillier ciphertexts. To the best of our knowledge, we are the first to consider the privacy-preserving most-frequent location query over encrypted data and achieve the privacy in both the workers' personal information and query results.
2) We employ optimization techniques to improve the efficiency of the proposed scheme. In particular, we first introduce a ciphertext packing technique to reduce the computational and communication costs by compressing several ciphertexts into one. Besides, to further optimize our scheme, we propose an *order-preserving merge function* that can improve the efficiency of our scheme by merging two transferred data into one while maintaining the order relation on the first transferred data.
3) Finally, we conduct extensive experiments to evaluate the performance of the proposed scheme, and the results show that it outperforms the baseline scheme in terms of computational costs and communication overhead.

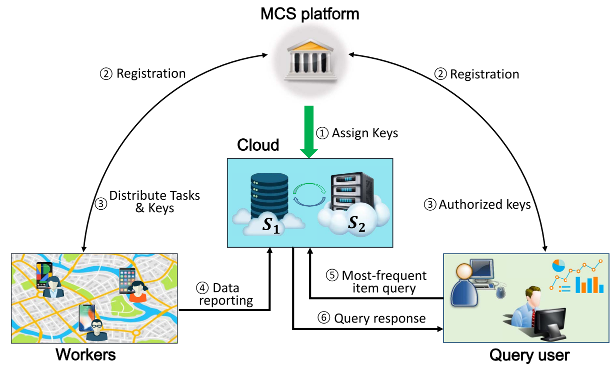

Fig. 1. System model under consideration.

The remainder of this article is organized as follows. In Section II, we introduce our system model, security model, and design goal. Then, we review our preliminaries in Section III. After that, we present our privacy-preserving query scheme in Section IV, followed security analysis and performance evaluation in Sections V and VI, respectively. Finally, we discuss some related works in Section VII and draw our conclusion in Section VIII.

II. Models and Design Goal

In this section, we formalize the system model, security model, and identify our design goal.

A. System Model

In our system model, we consider a typical cloud-based MCS system to answer the most-frequent item query on encrypted location data, which mainly consists of four types of entities, namely, an MCS platform $\mathcal{P}$, a set of workers $\mathcal{W} = \{w_1, w_2, w_3 \ldots\}$, the cloud $\mathcal{CS} = \{S_1, S_2\}$, and a query user $\mathcal{U}$, as shown in Fig. 1.

MCS Platform $\mathcal{P}$: In our system model, the MCS platform $\mathcal{P}$ is responsible for the whole MCS system. The tasks of $\mathcal{P}$ include managing the workers $\mathcal{W} = \{w_1, w_2, w_3 \ldots\}$ and the query user $\mathcal{U}$, initiating the MCS tasks to $\mathcal{W}$, and supporting the most-frequent item query from $\mathcal{U}$. However, since $\mathcal{P}$ may be not powerful in storage and computing, $\mathcal{P}$ tends to outsource MCS data management and the query services to the cloud $\mathcal{CS}$.

Cloud $\mathcal{CS} = \{S_1, S_2\}$: The MCS platform will employ two cloud servers $\mathcal{CS}=\{S_1, S_2\}$ from different cloud service providers, which are considered as powerful in both storage and computing. S_1 stores the reported data from the workers $\mathcal{W}$ while S_2 is authorized with the private key. They will cooperatively offer reliable most-frequent item query services to the query user $\mathcal{U}$.

Workers $\mathcal{W} = \{w_1, w_2, w_3 \cdots\}$: We consider the mobile users, who have registered in the MCS platform and participate in the MCS tasks, as the workers in our system. For each worker $w_i \in \mathcal{W}$, once sensing an event, he/she will report the event to the cloud $\mathcal{CS}$ in the following format:

$$\langle \mathtt{w_i}, \mathtt{eventType}, \mathtt{location}, \mathtt{timeStamp} \rangle$$

where `eventType`, `location`, and `timeStamp` are, respectively, denoted as the event's type, location, and reporting time.

Query User $\mathcal{U}$: In our system, the query user $\mathcal{U}$ may be a data analyst, who has registered to the MCS platform and will launch the most-frequent item query to the cloud for gaining the desirable results, e.g., the most frequent location item and its frequency in this work. Note that though our system model just considers one query user $\mathcal{U}$, it is natural to extend one to multiple query users.

B. Security Model

In our security model, we consider the MCS platform $\mathcal{P}$ is fully trusted, while the cloud $\mathcal{CS}$ is honest-but-curious, i.e., $\mathcal{CS}$ will faithfully follow the protocols by: 1) storing the sensed data from the workers $\mathcal{W}$ and 2) offering most-frequent item query services to the query user $\mathcal{U}$, but may be curious on the sensitive personal information, including the workers' identities and location, and the query results. For the cloud, we assume there is no collusion between S_1 and S_2, as well as no collusion between the cloud and other entities (the workers and the query user), which is reasonable since the cloud should maintain its reputation and interests. For the workers, they are considered honest. That is, they will honestly process the sensed data and report them to the cloud. However, since the cloud is not fully trusted, before reporting the sensed event, the real identity will be pseudonymized, and the location data will be encrypted in the following format:

$$\langle \mathtt{PID}_i, \mathtt{eventType}, \mathsf{Enc}(\mathtt{location})\mathtt{timeStamp} \rangle$$

where $\mathtt{PID}_i$ represents the pseudo-ID of the worker w_i, and $\mathsf{Enc}(\cdot)$ indicates employing a semantically secure encryption algorithm to encrypt the location data. For the query user, we consider the authorized user to be honest, who will faithfully follow the protocol to issue the most-frequent item query.

Note that external attackers may launch other active attacks, e.g., Denial-of-Service (DoS) attacks, to the MCS system. Since we focus on the privacy-preserving most-frequent item query, those attacks are beyond the scope of this article and will be discussed in our future work.

C. Design Goal

Under the aforementioned system model and security model, our design goal is to present a privacy-preserving most-frequent item query scheme for the MCS system. In particular, the following objectives should be attained.

1) *Privacy Preservation:* The fundamental requirement of the proposed scheme is the privacy preservation. On the one hand, the worker's personal information, including the identity and location data, should be kept secret from the cloud. On the other hand, our scheme guarantees that the cloud has no idea about the query result, which implies protecting the content of the query result, the frequency values $\{\mathsf{freq}(l_i)\}_{i=1}^{n}$, and the information about *which encrypted location is picked as the most frequent one.*
2) *Efficiency:* In order to achieve the above privacy requirements, it is inevitable to incur extra computational costs, i.e., processing the most-frequent item query over encrypted location will undoubtedly increase the computational costs compared with it doing over the plaintexts. In addition, since our scheme is deployed in a two-server setting, the communication overhead is another notable cost. Therefore, in the proposed scheme, we also aim to minimize the computational and communication costs of querying the most-frequent item.

III. Preliminaries

In this section, we briefly review the Paillier cryptosystem and transposition cipher, which will serve as the building blocks of our proposed scheme.

A. Paillier Cryptosystem

The Paillier cryptosystem [13] is a famous homomorphic encryption scheme that allows performing operations over the encrypted data and has been widely employed in various privacy-preserving computations [14]. Specifically, the Paillier cryptosystem includes three algorithms, namely: 1) key generation $\mathsf{P.KeyGen}(\kappa)$; 2) encryption $\mathsf{P.Enc}(pk, m)$; and 3) decryption $\mathsf{P.Dec}(sk, c)$, as follows.

1) $\mathsf{P.KeyGen}(\kappa)$*:* Given a security parameter κ, e.g., $\kappa = 512$, choose two large prime numbers $p = 2p' + 1$ and $q = 2q' + 1$, where $|p| = |q| = \kappa$, and p' and q' are also two primes. Let $n = pq$, $\lambda = \mathrm{lcm}(p-1, q-1) = 2p'q'$. After randomly choosing a generator $g \in \mathbb{Z}_{n^2}^*$ such that $\gcd(L(g^\lambda \bmod n^2), n) = 1$, $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$ is calculated, where $L(x) = (x - 1/n)$. The public key is $pk = (n, g)$, and the corresponding private key is $sk = (\lambda, \mu)$.
2) $\mathsf{P.Enc}(pk, m)$*:* Given a message $m \in \mathbb{Z}_n$, choose a random number $r \in \mathbb{Z}_n^*$, and the ciphertext c can be calculated as $c = \mathsf{P.Enc}(pk, m) = g^m \cdot r^n \bmod n^2$.
3) $\mathsf{P.Dec}(sk, c)$*:* Given the ciphertext c, with sk, the corresponding message m can be recovered as $m = \mathsf{P.Dec}(sk, c) = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$.

The Paillier cryptosystem is proved as semantically secure [13] and also enjoys the following three homomorphic properties. To save space, henceforth, $\mathsf{P.Enc}(pk, m)$ and $\mathsf{P.Dec}(sk, c)$ are abbreviated as $\mathsf{E}(m)$ and $\mathsf{D}(m)$, respectively.

1) *Addition:* Given two ciphertexts $\mathsf{E}(m_1)$ and $\mathsf{E}(m_2)$, we have $\mathsf{E}(m_1) \cdot \mathsf{E}(m_2) \rightarrow \mathsf{E}(m_1 + m_2)$.
2) *Multiplication:* Given a ciphertext $\mathsf{E}(m_1)$ and a plaintext $m_2 \in \mathbb{Z}_n$, we have $\mathsf{E}(m_1)^{m_2} \rightarrow \mathsf{E}(m_1 \cdot m_2)$.
3) *Self-Blinding:* Given a ciphertext $\mathsf{E}(m_1) = g^{m_1} \cdot r_1^n \bmod n^2$ and a random number $r_2 \in \mathbb{Z}_n^*$, we have $\mathsf{E}(m_1) \cdot r_2^n \bmod n^2 \rightarrow \mathsf{E}(m_1) = g^{m_1} \cdot (r_1 r_2)^n \bmod n^2$.

B. Transposition Cipher

As is known, the transposition cipher [12] is a classical encryption technique achieved by rearranging the order of letters according to the predetermined pattern. Given an $n \times m$ matrix R, where n is the row size and m is the column size, the transposition cipher, including key generation $\mathsf{T.KeyGen}()$, encryption $\mathsf{T.Enc}()$, and decryption $\mathsf{T.Dec}()$, can be applied on the matrix R as follows.

1) $\mathsf{T.KeyGen}(n, m)$*:* With the row size n and the column size m, two vectors

$$v_r = \{r_0, r_1, \ldots, r_{n-1} | r_0{=}0, r_1{=}1, .., r_{n-1}{=}n{-}1\}$$
$$v_c = \{c_0, c_1, \ldots, c_{m-1} | c_1{=}0, c_1{=}1, .., c_{m-1}{=}m{-}1\}$$

are initialized. Then, the transposition keys can be generated as two permuted vectors $\{P(v_r), P(v_c)\}$ by adopting the Durstenfeld version of the Fisher-Yates algorithm [15], i.e.,

$$P(v_r) = \left\{\text{swap}\left(r_i, r_j\right) |\ j \xleftarrow{\text{random}} [0, i] \text{ for } i{=}n{-}1 \text{ to } 0\right\}$$
$$P(v_c) = \left\{\text{swap}\left(c_i, c_j\right) |\ j \xleftarrow{\text{random}} [0, i] \text{ for } i{=}m{-}1 \text{ to } 0\right\}.$$

2) $\mathsf{T.Enc}(P(v_r), P(v_c), R)$*:* Given a matrix R, the transposition keys $P(v_r)$ and $P(v_c)$ can be applied to permute the matrix on its rows and columns, respectively, and then the encrypted matrix C_R is generated.
3) $\mathsf{T.Dec}(P(v_r), P(v_c), C_R)$*:* Given the encrypted matrix C_R, the corresponding matrix R can be recovered by swapping the rows and columns in C_R based on the mapping relations recorded in transposition keys: $P(v_r)$ and $P(v_c)$.

Obviously, for the matrix-based transposition cipher, the keyspace is $n! \times m!$, As the shuffle algorithm [15] can guarantee an unbiased permutation, i.e., every permutation is equally likely, the adversary can infer the original matrix R only with probability $(1/[n! \times m!])$.

IV. Our Proposed Scheme

In this section, we present our privacy-preserving most-frequent location query scheme. Before that, we would like to introduce the *order-preserving merge function*, which serves as an important component in our scheme.

A. Order-Preserving Merge Function

Suppose that there are two Paillier encrypted data sets $\mathsf{E}(x)$ and $\mathsf{E}(y)$ with the same size

$$\mathsf{E}(x) = \{(\mathsf{E}(x_1), \mathsf{E}(x_2), \ldots, \mathsf{E}(x_t)) | x_i \in \mathbb{Z}_n, 1 \le i \le t\}$$
$$\mathsf{E}(y) = \{(\mathsf{E}(y_1), \mathsf{E}(y_2), \ldots, \mathsf{E}(y_t)) | y_i \in \mathbb{Z}_n, 1 \le i \le t\}.$$

The order-preserving merge function is defined as follows.

Definition 1 (Order-Preserving Merge Function): An order-preserving merge function $f_{op}(\cdot)$ can map $\mathsf{E}(x_i)$ and $\mathsf{E}(y_i)$ to a new ciphertext $\mathsf{E}(z_i) = f_{op}(\mathsf{E}(x_i), \mathsf{E}(y_i))$, which has three properties.

1) z_i preserves the order of x_i, i.e., if $x_i > x_j$, $z_i > z_j$.
2) z_i is probabilistic, i.e., if $x_i = x_j$ and $y_i = y_j$, in general, $z_i \neq z_j$.
3) z_i is reversible, i.e., (x_i, y_i) can be recovered from z_i where $1 \le i, j \le t$.

We construct an order-preserving merge function f_{op} by

$$\begin{aligned} \mathsf{E}(z_i) &= f_{op}(\mathsf{E}(x_i), \mathsf{E}(y_i)) = \mathsf{E}(x_i)^a \cdot \mathsf{E}(y_i)^b \cdot \mathsf{E}(r_{ci}) \\ &\Rightarrow z_i = a \cdot x_i + b \cdot y_i + r_{ci} \end{aligned} \tag{1}$$

where a and b are two integers, and r_{ci} is a random number. Moreover, we guarantee that $|a| > \max(|y_i|) + |b|$, and $|b| > |r_{ci}|$, where $|\cdot|$ represents the bit length of an integer. Next, we prove that the above construction satisfies Definition 1.

Theorem 1: The construction (1) can guarantee z_i satisfies all properties of the order-preserving merge function.

Proof: First, we assume that $\mathsf{E}(z_i) = f_{op}(\mathsf{E}(x_i), \mathsf{E}(y_i))$, $\mathsf{E}(z_j) = f_{op}(\mathsf{E}(x_j), \mathsf{E}(y_j))$ and $x_i > x_j$.

1) *z_i Preserves the Order of x_i:* Based on the above assumption, we have

$$\begin{aligned} z_i - z_j &= a \cdot \left(x_i - x_j\right) + b \cdot \left(y_i - y_j\right) + \left(r_{ci} - r_{cj}\right) \\ &> a \cdot (x_i - x_j) - (b \cdot y_i + r_{cj}) \xrightarrow{x_i > x_j} \\ &\ge a - \left(b \cdot y_j + r_{cj}\right) \xrightarrow{|b| > |r_{cj}|} \\ &> a - b \cdot \left(y_j + 1\right) \xrightarrow{|a| > \max(|y_i|) + |b|} \ge 0. \end{aligned}$$

Hence, z_i preserves the order of x_i.
2) *z_i is Probabilistic:* It is evident that z_i is probabilistic due to the existence of the random number r_{ci}.
3) *z_i is Reversible:* Given z_i, a, and b, since $a > b \cdot y_i + r_{ci}$, we have $v_i = z_i \mod a = b \cdot y_i + r_{ci}$. Consequently, $x_i = ([z_i - v_i]/a)$ and $y_i = ([v_i - (v_i \mod b)]/b)$. Hence, z_i is reversible.

B. Description of Our Proposed Scheme

In this section, we show the details of our proposed scheme, which mainly consists of five phases: 1) system initialization; 2) data report from workers; 3) query request from query user; 4) query response at cloud; and 5) response recovery at query user.

1) System Initialization: We consider the MCS platform $\mathcal{P}$ is a trustable entity and will bootstrap the whole system. To initialize the system, the MCS platform $\mathcal{P}$ needs to register and authenticate the workers and query user to guarantee that only the authenticated workers and query user can perform the MCS tasks and the most-frequent location query, respectively. The details of system initialization are described as follows.

1) *Parameter Generation:* Given the security parameters (κ_0, κ_1), the MCS platform first chooses two random numbers $s_0, s_1 \in \{0, 1\}^{\kappa_0}$ as the master secret keys and uses κ_1 to generate the Paillier public–private key pair $(pk, sk) = ((n, g), (\lambda, \mu))$ (see details in Section III-A). After that, the MCS platform chooses a secure symmetric key encryption $\mathsf{SE}()$, i.e., AES-128, and a secure hash function $\mathsf{H}()$, e.g., SHA-256. Finally, the MCS platform sets and publishes the system parameters $\{pk, \mathsf{SE}(), \mathsf{H}()\}$ and securely assigns s_1 to the server S_1 as well as sk to the server S_2.
2) *Registration:* Both the workers $\mathcal{W}$ and the query user $\mathcal{U}$ need to register to the system. For a worker $w_i \in \mathcal{W}$ with identity $\mathtt{ID_i}$, the MCS platform verifies the authenticity of $\mathtt{ID_i}$ and generates a set of pseudo-IDs together with the corresponding secret keys, i.e., $\mathtt{PID_i} = \{(\mathtt{PID_{ij}}, k_{ij}) | j = 1, 2, \ldots\}$ for w_i, where each $\mathtt{PID_{ij}} = \mathsf{SE}(\mathtt{ID_i} \| r_j, s_0)$ is generated by encrypting the identity $\mathtt{ID_i}$ and a random number r_j using the master

Algorithm 1 Compression

Input: An upper triangular matrix, R_u; the row or column size of R_u, n.
Output: A compressed matrix, R;

(row, col) $\leftarrow$ (n mod 2 == 0) ? ($n-1$, $\frac{n}{2}$):($\frac{n-1}{2}$, n)
$R \leftarrow$ new Matrix(row, col)
$cnt \leftarrow 0$
for $i = 0 \rightarrow n-1$ **do**
    for $j = i+1 \rightarrow n-1$ **do**
        $r \leftarrow cnt$ mod row
        $c \leftarrow cnt$ / row
        $R[r][c] \leftarrow R_u[i][j]$
        $cnt \leftarrow cnt + 1$
    end for
end for
return R

key s_0, and $k_{ij} = \mathsf{H}(\texttt{PID}_{\texttt{ij}}, s_1)$. Then, the MCS platform authorizes w_i with $\texttt{PID}_{\texttt{i}}$ via a security channel. With this setting, w_i can change his/her pseudo-ID at a regular interval, e.g., 1 h, by selecting different pseudo-IDs from $\texttt{PID}_{\texttt{i}}$. When necessary, the MCS platform can still easily recover the real identity $\texttt{ID}_{\texttt{i}}$ from a given pseudo-ID $\texttt{PID}_{\texttt{ij}}$ by decrypting it into $\texttt{ID}_{\texttt{i}} \| r_j$ with s_0. In order to ensure that w_i has different pseudo-IDs in different periods, w_i will remove the used $(\texttt{PID}_{\texttt{ij}}, k_{ij})$ from $\texttt{PID}_{\texttt{i}}$, and the MCS platform will regularly update $\texttt{PID}_{\texttt{i}}$ before it runs out. When the query user $\mathcal{U}$ with identity $\texttt{ID}_{\texttt{u}}$ registers him/herself to the most-frequent item query services provided by the MCS platform, the platform will authenticate the user, and authorize the private keys $\{k_u$=$\mathsf{H}(\texttt{ID}_{\texttt{u}}, s_1), sk\}$ to him/her, so that the latter can use the private keys to launch a query and recover the query results from the cloud.

3) *Task Generation:* The MCS platform needs to initialize a task that specifies the expected event and its format, and then distributes the task to all registered workers and the cloud. Since different location-based services usually require different precision values of the location data, the MCS platform also includes the precision requirement d_p, i.e., the decimal places reserved for latitude and longitude, in the task.

2) Data Report From Workers: Once a worker w_i accepts the task, he/she can choose a pair $(\texttt{PID}_{\texttt{ij}}, k_{ij})$ from $\texttt{PID}_{\texttt{i}}$ and report the sensed event by the following specified format:

$$\texttt{R}_{ij} = \left\langle \texttt{PID}_{ij}, \texttt{eventType}, \mathsf{E}(l_i), \texttt{timeStamp} \right\rangle$$

together with its hash value $\texttt{H}_{ij} = \mathsf{H}(\texttt{R}_{ij}, k_{ij})$, i.e., $\texttt{R}_{ij}||\texttt{H}_{ij}$, to the cloud server S_1, where $\texttt{PID}_{ij}$ is one pseudo-ID in $\texttt{PID}_i$, k_{ij} is the secret key corresponding to $\texttt{PID}_{ij}$, and $\mathsf{E}(l_i)$ indicates encrypting the encoded location data l_i with the Paillier cryptosystem. Note that the encoded location data l_i should be a positive integer. In our scheme, we adopt a simple location encoding algorithm, called SLE, which is more efficient than other existing techniques, such as the geohash [16] and the Z-order curve technique [17]. Suppose there is a location data, denoted as $\langle \texttt{lon}, \texttt{lat} \rangle$, with a precision d_p = 5, e.g., $\langle -73.98134, 40.75864 \rangle$. The encoding algorithm $\mathsf{SLE}(\texttt{lon}, \texttt{lat})$ works as follows.

Algorithm 2 Ciphertext Packing

function *packing*(R_c, $|m_l|$)
    $r \leftarrow$ getRowSize(R_c); $c \leftarrow$ getColSize(R_c)
    $nc \leftarrow$ (c mod 2 == 0) ? $\frac{c}{2}$:$\frac{c+1}{2}$
    $R_s \leftarrow$ new Matrix(r, nc)
    for $j = 0 \rightarrow nc - 1$ **do**
        for $i = 0 \rightarrow r - 1$ **do**
            $\mathsf{E}(v_1) \leftarrow R_c$.get($i, j*2$)
            if R_c.get($i, j*2+1$) == null **then**
                $res \leftarrow \mathsf{E}(v_1)$
            else
                $\mathsf{E}(v_2) \leftarrow R_c$.get($i, j*2+1$)
                $res \leftarrow \mathsf{E}(v_1)^{2^{|m_l|}} \cdot \mathsf{E}(v_2)$
            end if
            R_s.set(i, j, res)
        end for
    end for
    return R_s
end function

function *unpacking*(R_s, $|m_l|$, *col*)
    $r \leftarrow$ getRowSize(R_s); $c \leftarrow$ getColSize(R_s)
    $R_c \leftarrow$ new Matrix(r, col)
    for $j = 0 \rightarrow c - 1$ **do**
        for $i = 0 \rightarrow r - 1$ **do**
            $v \leftarrow \mathsf{D}(R_s$.get($i, j$))
            if $(2*j+1) > (col-1)$ **then**
                R_c.set($i, col-1, v$)
            else
                $v_1 \leftarrow v >> |m_l|$; $v_2 \leftarrow v$ & $(2^{|m_l|} - 1)$
                R_c.set($i, 2*j, v_1$); R_c.set($i, 2*j+1, v_2$)
            end if
        end for
    end for
    return R_c
end function

Step-1: Convert $\texttt{lon}$ and $\texttt{lat}$ into the range of [0, 360] and [0, 180], respectively. Denote the converted data as $\langle \texttt{plon}, \texttt{plat} \rangle$, where $\texttt{plon} = \texttt{lon} + 180$ and $\texttt{plat} = \texttt{lat} + 90$.

Step 2: Lift $\langle \texttt{plon}, \texttt{plat} \rangle$ to positive integers. In the example, as $\texttt{lon}$ and $\texttt{lat}$ are truncated to the first five decimal places, it is easy to obtain the positive integers by multiplying 10^5, so that $\texttt{plon}$ and $\texttt{plat}$, respectively, fall into the ranges $[0, 3.6 \times 10^7]$ and $[0, 1.8 \times 10^7]$.

Step 3: Integrate these two positive integers, $\texttt{plon}$ and $\texttt{plat}$, to a big integer l, i.e., $l = \texttt{plon} \cdot 10^{(d_p+3)} + \texttt{plat}$.

Upon receiving the report $\texttt{R}_{ij}||\texttt{H}_{ij}$, the cloud server S_1 first uses the secret key s_1 to compute $k_{ij} = \mathsf{H}(\texttt{PID}_{ij}, s_1)$, and then checks whether $\texttt{H}_{ij} \stackrel{?}{=} \mathsf{H}(\texttt{R}_{ij}, k_{ij})$. If yes, the report $\texttt{R}_{ij}$ will be accepted, and rejected otherwise.

3) Query Request From Query User: Assume the query user $\mathcal{U}$ launches the following most-frequent item query $\texttt{Q}$: *which location is the hottest for a specified event in a past time window, and what is the corresponding frequency value?* In other words, the query user wants to query the most-frequent location and its frequency value filtered by the event type and the time stamp. An example of the SQL statement is shown as

Algorithm 3 Decompression

Input: A compressed matrix, R; the row or column size of R_u, n; the number of added dummy row, d_r.
Output: An upper triangular matrix, R_u;

row $\leftarrow$ getRows(R) - d_r
$R_u \leftarrow$ new Matrix(n, n)
for $i = 0 \rightarrow n-1$ **do**
  for $j = i \rightarrow n-1$ **do**
    if i=j **then**
      $R_u[i][j] \leftarrow \mathsf{E}(1)$
    else if $i < j$ **then**
      $cnt \leftarrow i \cdot n + j$
      $r \leftarrow cnt$ mod row
      $c \leftarrow cnt$ / row
      $R_u[i][j] \leftarrow R[r][c]$
    end if
  end for
end for
return R_u

follows, which queries the most "Traffic congestion" location

```
SELECT location, count(*) AS count
FROM table
WHERE lowerBound < timeStamp < upperBound
AND eventType = "Traffic congestion"
GROUP BY location ORDER BY count DESC LIMIT 1.
```

The query user $\mathcal{U}$ will send the following query request:

$$\mathtt{ID_u}\|\mathtt{Q}\|\mathsf{H}(\mathtt{ID_u}\|\mathtt{Q}, k_u)$$

to the cloud server S_1. Upon receiving the query request $\mathtt{ID_u}||\mathtt{Q}||\mathsf{H}(\mathtt{ID_u}||\mathtt{Q}, k_u)$, the cloud server S_1 first uses the secret key s_1 to compute the key $k_u = \mathsf{H}(\mathtt{ID_u}, s_1)$, and then checks whether the received $\mathsf{H}(\mathtt{ID_u}||\mathtt{Q}, k_u)$ is correct. If yes, the query request will be further processed, and rejected otherwise.

4) Query Response at Cloud: In order to achieve the privacy preservation in responding to the query $\mathtt{Q}$, the cloud servers S_1 and S_2 will cooperatively provide the most-frequent location query services by running the following steps.

Step-1: Since S_1 receives and stores all reported data, the server will launch the query response by filtering the stored data and forming an encrypted location set $\mathsf{E}(L) = \{\mathsf{E}(l_1), \mathsf{E}(l_2), \ldots, \mathsf{E}(l_n)\}$, where $\mathsf{E}(l_i)$ ($1 \leq i \leq n$ and $n > 1$) indicates that the encoded location data l_i is encrypted by the Paillier cryptosystem. Fig. 2 shows an example of how to respond to the query $\mathtt{Q}$ over $\mathsf{E}(L) = \{\mathsf{E}(1), \mathsf{E}(2), \mathsf{E}(6), \mathsf{E}(6), \mathsf{E}(8)\}$.

Step-2: Based on the set $\mathsf{E}(L)$, S_1 first generates an $n \times n$ matrix R_u, which is an upper triangular matrix without the leading diagonal. In R_u, each element R_u^{ij} in the ith row and jth column is a test pair and is computed as $R_u^{ij} = \mathsf{E}(r_{ij}(l_i - l_j)) = ([\mathsf{E}(l_i)]/[\mathsf{E}(l_j)])^{r_{ij}}$ where $r_{ij} \in \{0, 1\}^{\alpha}$ ($\alpha \ll \kappa_1$) is a random number. In Fig. 2, we denote the element in R_u as $E_{l_i l_j}$ and ignore those elements $i \geq j$ for simplicity. Then, S_1 compresses R_u into a new $(n-1) \times (n/2)$ or $(n-1/2) \times n$ matrix by rearranging the valid elements in R_u [see the detailed compression in Algorithm 1 and an example of the compressed matrix, i.e., the matrix (b), in Fig. 2]. Next, S_1 adds dummy data into the compressed matrix, and the strategy is as follows.

a) Add at least one row and one column of dummy data to the compressed matrix.
b) Each dummy data are randomly generated by encrypting 0 or r, where $r \in [-\eta, \eta]$ is a random number, and $\eta \in \{0, 1\}^{\alpha}$.
c) Guarantee that there are at least one $\mathsf{E}(0)$ and one $\mathsf{E}(r)$ in the dummy data.

We denote the newly formed matrix as R_c. After adding dummy data, S_1 applies the transposition cipher on the matrix R_c to further enhance the privacy of our scheme. Specifically, S_1 generates two transposition keys, $P(v_r)$ and $P(v_c)$, and uses them to permute R_c. In Fig. 2, the transposition keys $P(v_r) = \{2, 3, 1\}$ and $P(v_c) = \{3, 1, 6, 4, 2, 5\}$ are used to encrypt the matrix (c), in which one row and one column dummy data are added, E_{00} indicates $\mathsf{E}(0)$, and E_{ri} means ith $\mathsf{E}(r)$. Before transferring the permuted matrix R_c to S_2, S_1 compresses it into a smaller matrix R_s by employing the ciphertext packing technique $packing(R_c, |m_l|)$, as shown in Algorithm 2, where $|m_l|$ is the bit length of a mask and can be determined and published by the MCS platform. (See an example on how to choose $|m_l|$ in Section VI-B.) In Fig. 2, the element in R_s is represented as $E_{l_i l_j}^{l_s l_t}$ that packs the underlying plaintext of $E_{l_i l_j}$ and $E_{l_s l_t}$ into one (for ease of presentation, E_{00} and E_{ri} are unified as $E_{l_i l_j}$ or $E_{l_s l_t}$). Finally, S_1 transfers the matrix R_s and the column size col of R_c to S_2.

Step-3: After receiving the matrix R_s, S_2 decrypts all elements in R_s and recovers underlying plaintexts of the elements in permuted R_c by the *unpacking* function in Algorithm 2. Then, S_2 tests if the recovered plaintexts $P_{l_i l_j}$ is 0, where $P_{l_i l_j} = r_{ij}(l_i - l_j)$. If yes, S_2 generates $\mathsf{E}(1)$ as the test result, and $\mathsf{E}(0)$ otherwise. Next, S_2 puts the test result to the corresponding position in the permuted R_c. In Fig. 2, the matrix (f) contains the recovered plaintexts $P_{l_i l_j}$, and the matrix (g) is the tested R_c, where E_0 and E_1 represent $\mathsf{E}(0)$ and $\mathsf{E}(1)$, respectively. Finally, S_2 sends the tested R_c back to S_1.

Step-4: S_1 receives the tested R_c and reverses it using the transposition keys: $P(v_r)$ and $P(v_c)$. Then, S_1 applies the decompression algorithm, as shown in Algorithm 3, to generate an upper triangular matrix that is depicted as matrix (i) in Fig. 2. Next, S_1 computes the encrypted frequency values for each item based on the upper triangular matrix with Algorithm 4, which uses the additive homomorphic property of the Paillier cryptosystem

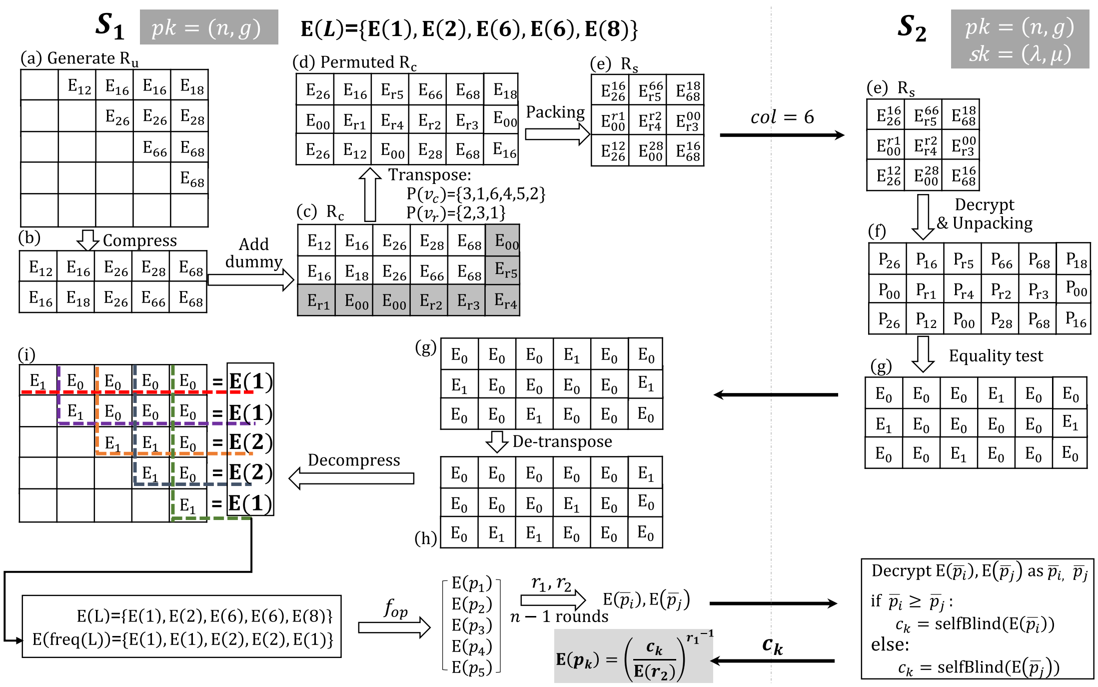

Fig. 2. Example for query response at cloud.

Algorithm 4 Frequency Counting

Input: A recovered upper triangular matrix filled with encrypted test results, R; the row or column size of R_u, n.
Output: A set for collecting encrypted frequency values, $\mathsf{E}(\mathsf{frep}(L))$.

```
for i = 0 → n − 1 do
    E_freq ← E(1)
    for j = 0 → n − 1 do
        if i > j then
            current ← R[j][i]
        else if i < j then
            current ← R[i][j];
        end if
        E_freq ← E_freq · current
    end for
    E(frep(L)).append(E_freq)
end for
return E(frep(L))
```

and is visually represented by calculating the test results on different color lines in the matrix (i).

Step-5: To date, S_1 holds two data sets with the same size

$$\mathsf{E}(L) = \{\mathsf{E}(l_1), \mathsf{E}(l_2), \cdots, \mathsf{E}(l_n)\}$$
$$\mathsf{E}\big(\mathsf{freq}(L)\big) = \big\{\mathsf{E}\big(\mathsf{freq}(l_1)\big), \mathsf{E}\big(\mathsf{freq}(l_2)\big) \cdots \mathsf{E}\big(\mathsf{freq}(l_n)\big)\big\}$$

where $\mathsf{freq}(l_i)$ $(1 \le i \le n)$ is the frequency value of l_i. In order to hide the privacy of the query result on *which item in* $\mathsf{E}(L)$ *is the most frequent* and enhance the performance, before picking out the most-frequent item, S_1 adopts the *order-preserving merge function* to merge $\mathsf{E}(l_i)$ and $\mathsf{E}(\mathsf{freq}(l_i))$ into one ciphertext

$$\mathsf{E}(p_i) = f_{op}\big(\mathsf{E}\big(\mathsf{freq}(l_i)\big), \mathsf{E}(l_i)\big) = \mathsf{E}\big(a \cdot \mathsf{freq}(l_i) + b \cdot l_i + r_{ci}\big)$$

in which p_i can preserve the order relation of $\mathsf{freq}(l_i)$, i.e., if $\mathsf{freq}(l_i) > \mathsf{freq}(l_j)$, $p_i > p_j$ holds. Then, S_1 selects two merged data, for example, $\mathsf{E}(p_1)$ and $\mathsf{E}(p_2)$, and chooses two random numbers $r_1, r_2 \in \{0, 1\}^\alpha$ to calculate $\mathsf{E}(\bar{p_1})$ and $\mathsf{E}(\bar{p_2})$, where

$$\mathsf{E}(\bar{p_1}) = \mathsf{E}(p_1)^{r_1} \cdot \mathsf{E}(r_2) = \mathsf{E}(p_1 \cdot r_1 + r_2)$$
$$\mathsf{E}(\bar{p_2}) = \mathsf{E}(p_2)^{r_1} \cdot \mathsf{E}(r_2) = \mathsf{E}(p_2 \cdot r_1 + r_2).$$

Next, S_1 sends $\mathsf{E}(\bar{p_1})$ and $\mathsf{E}(\bar{p_2})$ to S_2 for obtaining the ciphertext that has the larger plaintext.

Step-6: Upon receiving $\mathsf{E}(\bar{p_1})$ and $\mathsf{E}(\bar{p_2})$ from S_1, S_2 decrypts them using the private key sk and obtains two plaintexts: $\bar{p_1}$ and $\bar{p_2}$. Then, S_2 compares $\bar{p_1}$ and $\bar{p_2}$ and defines $\bar{p_k}$ as the larger one, i.e., $\bar{p_k} = \max(\bar{p_1}, \bar{p_2})$. After that, S_2 conducts the self-blinding operation on $\mathsf{E}(\bar{p_k})$, i.e., $c_k = \mathsf{selfBlind}(\mathsf{E}(\bar{p_k}))$, and returns c_k to S_1. In other words, the self-blinding operation will be employed on the ciphertext who has the larger underlying plaintext, seeing the example in Fig. 2.

Step-7: With the received c_k, S_1 recovers $\mathsf{E}(p_k)$ by $\mathsf{E}(p_k) = (c_k/[\mathsf{E}(r_2)])^{r_1^{-1}}$, where p_k contains the larger frequency value and the corresponding item. Since the self-blinding property can transform a ciphertext to another one without changing the corresponding plaintext, S_1 is unable to link c_k to $\mathsf{E}(\bar{p_1})$ or $\mathsf{E}(\bar{p_2})$.

After that, S_1 constructs $\mathsf{E}(p_j)$ by merging the unchecked $\{\mathsf{E}(\mathsf{freq}(l_j)), \mathsf{E}(l_j)\}$ and compares it with the recovered $\mathsf{E}(p_k)$. Repeat steps-5–7, until all items in $\mathsf{E}(L)$ are compared. Totally, there will be $n-1$ rounds, and the recovered ciphertext $\mathsf{E}(p_k)$ in the last round (gray background area in Fig. 2) will contain the largest frequency and its item. Note that if two or more items in $\mathsf{E}(L)$ have the largest frequency value, the larger item will be picked as the query result, which is guaranteed by the *order-preserving merge function*. In order to securely return $\mathsf{E}(p_k)$ and the parameters $\{a, b\}$ to the query user $\mathtt{ID_u}$, S_1 first generates the private key $k_u = \mathsf{H}(\mathtt{ID_u}, s_1)$ with authorized s_1 for the query user, then encrypts $\mathsf{E}(p_k)||a||b$ with k_u, i.e., $\mathtt{Res} = \mathsf{SE}(\mathsf{E}(p_k)||a||b, k_u)$. Eventually, S_1 sends $\mathtt{Res}$ to the query user $\mathtt{ID_u}$ for responding the most-frequent item query.

5) Response Recovery at Query User: Upon receiving the query response $\mathtt{Res}$ from S_1, with the authorized private keys $\{k_u, sk\}$, the query user $\mathtt{ID_u}$ can first recover $\mathsf{E}(p_k)||a||b$ with k_u, then decrypt $\mathsf{E}(p_k)$ with sk, i.e., $p_k = \mathsf{D}(sk, \mathsf{E}(p_k)) = a \cdot \mathsf{freq}(l_k) + b \cdot l_k + r_{ck}$. Finally, with a and b, the largest frequency value $\mathsf{freq}(l_{\max})$ and the most-frequent item $l_{\max}$ can be obtained as follows, where we let $v_k = p_k \bmod a$

$$\mathsf{freq}(l_{\max}) = \frac{p_k - v_k}{a}; \quad l_{\max} = \frac{v_k - (v_k \bmod b)}{b}. \tag{2}$$

V. Security Analysis

In this section, we discuss the security properties of our proposed most-frequent item query scheme. In particular, following our design goal, we will focus on how the proposed scheme can achieve privacy preservation for workers' personal information and the query result against two noncollusive cloud servers $\mathcal{CS} = \{S_1, S_2\}$.

A. Workers' Personal Information Is Privacy Preserving

In our scheme, the workers' personal information refers to their identities and locations, which are stored in the cloud. Specifically, S_1 stores the collected data, while S_2 holds the private key sk. In the following, we will show that the identity and location information are kept secret from S_1 and S_2.

Identity Information Is Privacy Preserving: A worker's real identity $\mathtt{ID_i}$ should be protected from S_1 and S_2. For S_1, it holds the collected data, i.e., $\langle \mathtt{PID}_{ij}, \mathtt{eventType}, \mathsf{E}(l_i), \mathtt{timeStamp}\rangle$, and it may attempt to infer the worker's real identity in two ways. First, S_1 may use the pseudo-ID $\mathtt{PID_{ij}}$ to infer $\mathtt{ID_i}$, where $\mathtt{PID_{ij}} = \mathsf{SE}(\mathtt{ID_i}\|r_j, s_0)$. Since S_1 does not have the secret key s_0, the security of the symmetric key encryption $\mathsf{SE}()$ ensures that S_1 cannot recover the real identity $\mathtt{ID_i}$ from $\mathtt{PID_{ij}}$. Second, S_1 may deduce $\mathtt{ID_i}$ using the remaining fields, i.e., $\mathtt{eventType}$, $\mathsf{E}(l_i)$, and $\mathtt{timeStamp}$. Since $\mathsf{E}(l_i)$ is generated by the Paillier cryptosystem, S_1 cannot recover l_i without the private key sk. Thus, S_1 is unable to infer $\mathtt{ID_i}$ using the location data. In this case, S_1 may infer $\mathtt{ID_i}$ with $\mathtt{eventType}$ and $\mathtt{timeStamp}$. That is, if a worker uses one fixed pseudo-ID, S_1 may learn workers' activity patterns from the accumulated $\mathtt{eventType}$ and $\mathtt{timeStamp}$, which can be used to infer the real identity $\mathtt{ID_i}$ [8]. Since a worker in our scheme will select a different $\mathtt{PID_{ij}}$ and change his/her identity at short intervals, S_1 is unable to obtain the worker's activity patterns. As a result, S_1 cannot infer the real identity $\mathtt{ID_i}$ with $\mathtt{eventType}$ and $\mathtt{timeStamp}$. Hence, S_1 has no idea about the worker's real identity.

For S_2, since it cannot access the collected data, S_2 learns nothing about the worker's identity. It is worth noting that there is no collusion between S_1 and S_2. To sum up, the worker's identity information is privacy-preserving.

The Location Information Is Privacy-Preserving: The workers' location data $\{l_i\}_{i=1}^n$ and the linkage between locations (i.e., whether two encrypted locations refer to the same location) should be protected from S_1 and S_2.

For S_1, it stores the encrypted location data $\{\mathsf{E}(l_i)\}_{i=1}^n$ and processes these data through the Paillier homomorphic operations. When conducting queries, S_1 interacts with S_2 to obtain the encrypted results of location equality tests, i.e., $\mathsf{E}(1)$ or $\mathsf{E}(0)$, between any two $\mathsf{E}(l_i)$ and $\mathsf{E}(l_j)$, where the values $\{1, 0\}$ indicate whether these two encrypted locations refer to the same location or not. Since these values are encrypted, and S_1 does not have the private key sk, the security of the Paillier cryptosystem guarantees that S_1 cannot obtain the location data and their linkage information.

For S_2, as our scheme shows, it can use the authorized private key sk to recover: 1) the underlying plaintexts $P_{l_il_j}$ of elements in permuted R_c, where $P_{l_il_j} = r_{ij}(l_i - l_j)$, i.e.,

$$\begin{cases} \left.\begin{array}{l} P_{l_1l_2} = r_{12}(l_1 - l_2) \\ \quad\cdots\cdots \\ P_{l_1l_n} = r_{1n}(l_1 - l_n) \end{array}\right\}(n-1) \\ \left.\begin{array}{l} P_{l_2l_3} = r_{23}(l_2 - l_3) \\ \quad\cdots\cdots \\ P_{l_2l_n} = r_{2n}(l_2 - l_n) \end{array}\right\}(n-2) \\ \quad\cdots\cdots \qquad\qquad\qquad\qquad \cdots\cdots \\ \quad\cdots\cdots \qquad\qquad\qquad\qquad \cdots\cdots \\ P_{l_{n-1}l_n} = r_{(n-1)n}(l_{n-1} - l_n); \ \}1 \end{cases} \tag{3}$$

and 2) $\bar{p_i}$ and $\bar{p_j}$ in a comparison round, where

$$\begin{cases} \bar{p_i} = \big(a \cdot \mathsf{freq}(l_i) + b \cdot l_i + r_{ci}\big) \cdot r_1 + r_2 \\ \bar{p_j} = \big(a \cdot \mathsf{freq}(l_j) + b \cdot l_j + r_{cj}\big) \cdot r_1 + r_2. \end{cases} \tag{4}$$

For the plaintexts $P_{l_il_j}$, S_2 can construct a system of equations, as shown in (3). Obviously, there are n unknown locations $\{l_i\}_{i=1}^n$ and $([n(n-1)]/2)$ random numbers in the system. Since the system only has $([n(n-1)]/2)$ equations, S_2 cannot solve the system to obtain the location data $\{l_i\}_{i=1}^n$. For the plaintexts $\bar{p_i}$ and $\bar{p_j}$ in each comparison round, S_2 can construct a system of equations, as shown in (4). Since the system has at least four variables $\{r_{ci}, r_{cj}, r_1, r_2\}$ and only two equations, S_2 cannot solve this system to obtain the location data $\{l_i, l_j\}$. Besides, although S_2 can determine whether $r_{ij}(l_i - l_j) \stackrel{?}{=} 0$, it has no idea about which two locations are the same when the result is 0. As a result, S_2 can neither recover the location data nor infer the linkage information.

To sum up, the noncollusive S_1 and S_2 learn nothing about the workers' location data $\{l_i\}_{i=1}^n$ and their linkage information. Thus, the location information is privacy preserving.

B. Query Result Is Privacy Preserving

In this section, we first show the content of the query result $(l_{\max}, \mathsf{freq}(l_{\max}))$ is privacy preserving. Then, we prove the locations' frequency values, which can be used to infer the query result, are privacy preserving. In addition, we demonstrate the query result on *which encrypted location* $\mathsf{E}(l_i)$ *in* $\mathsf{E}(L)$ *is picked as the most frequent one* is privacy preserving.

Content of the Query Result Is Privacy Preserving: In response to the most-frequent location query, S_1 can obtain $\mathsf{E}(p_k)$, where $p_k = a \cdot \mathsf{freq}(l_{\max}) + b \cdot l_{\max} + r_{ck}$, and S_2 can obtain the largest $\bar{p_k} = p_k \cdot r_1 + r_2$ in the last round. For S_1, it cannot even access p_k from $\mathsf{E}(p_k)$ without sk. For S_2, p_k is also kept secret due to the existence of the random numbers, r_1 and r_2. Additionally, in our scheme, S_1 encrypts the query response $\{\mathsf{E}(p_k), a, b\}$ by the symmetric key encryption $\mathsf{SE}()$, i.e., $\texttt{Res} = \mathsf{SE}(\mathsf{E}(p_k)||a||b, k_u)$, before transferring it to the query user. Therefore, S_2 cannot recover $\{\mathsf{E}(p_k), a, b\}$ without k_u, which can prevent S_2 from obtaining p_k by the authorized sk. Note that p_k contains the query result $(l_{\max}, \mathsf{freq}(l_{\max}))$. Hence, in our scheme, the content of the query result is privacy preserving.

Locations' Frequency Values Are Privacy Preserving: Next, we will show that the frequency values are privacy preserving in our scheme.

For S_1, it can obtain the encrypted test results and further use them to homomorphically compute the encrypted frequency values. As all of them are encrypted with the Paillier cryptosystem, S_1 is unable to know their underlying plaintexts without sk. As a result, S_1 has no idea about the locations' frequency values.

As our scheme shows, S_2 can know the equality test results and recover $\bar{p_i}$ and $\bar{p_j}$ in comparison rounds [see details in (4)]. First, with the equality test results, if S_2 can correctly put them into the original positions in R_u, the server can easily count the frequency values using the similar way in Algorithm 4. However, before transferring R_u to S_2, two techniques are adopted: 1) adding dummies and 2) transposition cipher. If we assume n is an even number and there are d_r rows and d_c columns dummy data, the keyspace of the transposition cipher will be $((n-1)+d_r)! \times (n/2+d_c)!$. Since the transposition cipher is applied to the Paillier ciphertexts, the existing attacks for transposition cipher, such as the frequency analysis, do not work in our scheme. As a result, S_2 can only recover the original matrix R_u with the probability $P_r(S_2)$, where

$$P_r(S_2) = \left\{1/[((n-1)+d_r)! \times \left(\frac{n}{2}+d_c\right)]!|n > 1, d_r, d_c \geq 1\right\}.$$

However, since each guessed R_u is equally likely, S_2 cannot verify whether the guessed R_u is correct or not. Thus, S_2 is unable to obtain the frequency values by counting the test results. Second, S_2 can recover $\bar{p_i}$ and $\bar{p_j}$ in comparison rounds. Since our *order-preserving merge function* has the probabilistic property, i.e., $l_i = l_j \Rightarrow p_i \neq p_j$, we have $\bar{p_i} \neq \bar{p_j}$ even if $l_i = l_j$, which prevents S_2 from inferring the frequency values by observing $\bar{p_i} = \bar{p_j}$. To sum up, the locations' frequency values are privacy preserving.

Query Result on Which $\mathsf{E}(l_i)$ *in* $\mathsf{E}(L)$ *Is Picked as the Most-Frequent One Is Privacy-Preserving:* For S_1, in the last round, it knows the received ciphertext $\mathsf{E}(p_k)$ that has the largest frequency. However, S_2 applies the self-blinding operation on the returned ciphertext before transferring it to S_1. Thus, S_1 only knows the received ciphertext $\mathsf{E}(p_k)$ has the largest frequency but cannot link it to the encrypted location data in $\mathsf{E}(L)$. For S_2, it receives $\mathsf{E}(\bar{p_i})$ and $\mathsf{E}(\bar{p_j})$ from S_1 and can recover $\bar{p_i}$ and $\bar{p_j}$ using the authorized sk. In the last round, S_2 can even know the largest $\bar{p_k}$ and the corresponding $\mathsf{E}(\bar{p_k})$. However, since $\mathsf{E}(\bar{p_k})$ is generated in S_1 by $\mathsf{E}(\bar{p_k}) = \mathsf{E}(p_k)^{r_1} \cdot \mathsf{E}(r_2)$, and there is no collusion between these two servers, S_2 cannot even link $\mathsf{E}(\bar{p_k})$ to $\mathsf{E}(p_k)$. Hence, S_2 cannot infer the query result on which encrypted location in $\mathsf{E}(L)$ is the most frequent.

From the above analysis, we can see that in our scheme, both the worker's personal information and the query result are privacy preserving, and we achieve our design goal in terms of privacy preservation.

VI. Performance Evaluation

In this section, we evaluate the performance of our proposed scheme. Specifically, on the worker side, we will compare our location encoding algorithm SLE with geohash and Z-order curve technique in computational costs. On the cloud side, since we are the first to consider the security and privacy of the most-frequent location query, the proposed scheme is compared with the *baseline* scheme, which neither applies the ciphertext packing technique nor the *order-preserving merge function*, in terms of computational costs and communication overhead. Besides, in order to facilitate the presentation and discussion of the performance advantages of the ciphertext packing technique and the *order-preserving merge function*, we divide the process of the query response at the cloud into two protocols: 1) *frequency count protocol* (from step-1 to step-4) and 2) *frequency comparison protocol* (from step-5 to step-7). All of the experiments are conducted on an Intel CORE i5-3317U CPU@1.70 GHz Windows Platform with 8-GB RAM.

Implementation: We implement all of the schemes, including our proposed scheme and the alternative schemes, in Java. For geohash, we directly use the package from the maven repository [18], while the Z-order curve algorithm is an optimized version of [19] that can support the codable range from 1 to 2^{32}.

Data Set: In our experiments, we adopt a real-world data set from *New York Motor Vehicle Collisions* [20], denoted as *NYMVC*. In particular, we first extract the fields of *date*, *longitude*, and *latitude* in the data set. Then, we filter out the missing-location items and the abnormal items, in which the location data are beyond the scope of New York City. Eventually, there are 792 288 items in our data set.

A. Computational Costs of Encoding Algorithms

In Fig. 3(a), we apply the encoding algorithm SLE, geohash, and Z-order curve technique to encode the location data in *NYMVC* and compare the total computational costs varying the decimal places from 3 to 7. From the figure, we can see, at all precision levels, our encoding algorithm SLE outperforms the geohash significantly and the Z-order curve technique slightly. However, when encoding the location data, the Z-order curve technique adopts a table with 256 items to optimize the computational costs, which requires more

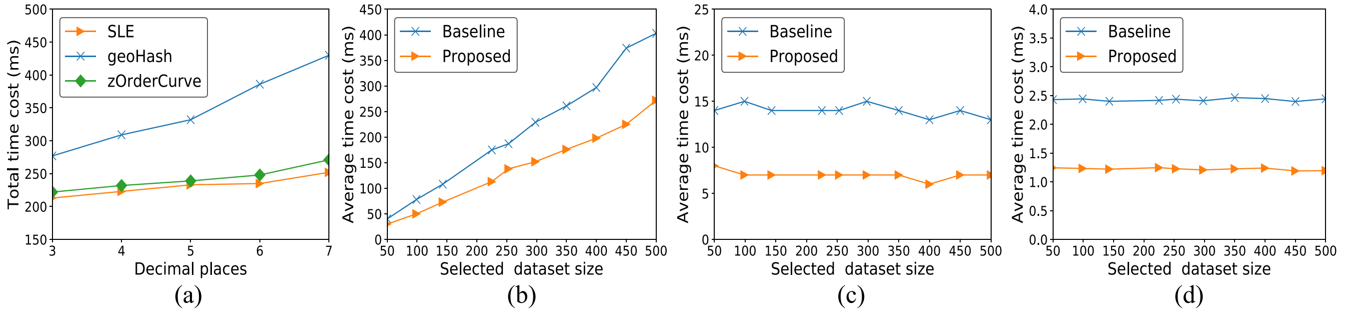

Fig. 3. Computational costs comparisons. (a) Total time cost for different encoding algorithms. (b) Average time cost of *frequency count protocol*. (c) Average time cost of *frequency comparison protocol*. (d) Average time cost of query user (repeat 1000 times).

memory space than our algorithm. Hence, the SLE algorithm has superiority in resource-constrained mobile devices over geohash and the Z-order curve technique.

B. Computational Costs of the Cloud and Query User

In this section, we compare computational costs between our proposed scheme and the baseline scheme varying the selected data set size from 50 to 500. To simulate the real-world scenario, we aggregate the location data in *NYMVC* by the *date* (YY-MM-DD) field and generate 1715 subdata sets with the size from 31 to 959. In the following evaluations, we will conduct our experiments on these subdata sets with different sizes. For setting parameters, both in the proposed scheme and the baseline scheme, we set the security parameter $\kappa_0 = 256$, $\kappa_1 = 512$. As shown in [21], a value in decimal degrees of 5 decimal places is precise to about 1 m at the equator. Therefore, it is reasonable to truncate the location data with the decimal places $d_p = 5$. Consequently, the encoded location data will have the maximum bit length $\max(|l|) = 52 (2^{52} > 3.6 \times 10^{15})$. Also, we can choose the bit length of the mask $|m_l| = 256$ as we would like to pack two plaintexts into one. Regarding $|a|$, $|b|$, and $|r_c|$, we let $|a| = 128$, $|b| = 64$, and $|r_c| = 32$, which can guarantee that $2^{|r_c|} >> 792,288$, $|b| > |r_c|$, and $|a| > \max(|l|) + |b| = 52 + 64 = 116$.

From Fig. 3(b), we can see that the *frequency count protocol* in our proposed scheme has a lower average time cost than that in the baseline scheme who does not apply the ciphertext packing technique. The reason is that in the baseline scheme, S_2 needs to decrypt $r \times c$ ciphertexts for equality test, where r is the row size, and c is the column size of the permuted matrix R_c. However, in the proposed scheme, S_2 only decrypts around $([r \times c]/2)$ ciphertexts. Although S_1 will take extra computational costs in packing ciphertexts and S_2 will take some costs in unpacking, the experimental results show that the ciphertext packing technique can improve the average execution time by up to around 1.5×.

Fig. 3(c) illustrates the computational costs of *frequency comparison protocol* in the proposed scheme and baseline scheme. In our proposed scheme, with the *order-preserving merge function*, S_1 can merge $\mathsf{E}(l_i)$ and $\mathsf{E}(\mathsf{freq}(l_i))$ into one while ensuring privacy. However, in the baseline scheme, S_1 needs to transfer a pair $\langle \mathsf{E}(l_i), \mathsf{E}(\mathsf{freq}(l_i)) \rangle$ to S_2, which means S_1 has to generate two pairs $\langle r_1, r_2 \rangle$ and $\langle r_3, r_4 \rangle$ to hide l_i and $\mathsf{freq}(l_i)$, respectively, where r_1 to r_4 are random numbers. Also, S_2 has to employ the self-blinding operation both in $\mathsf{E}(l_i)$ and $\mathsf{E}(\mathsf{freq}(l_i))$ to break the link. Therefore, although this function will take extra computational costs in merging data, the proposed scheme can sharply reduce the average time cost compared to that of the baseline scheme in *frequency comparison protocol*.

Besides, if the *order-preserving merge function* is not employed, S_1 has to send two ciphertexts, $\mathsf{E}(l_{\max})$ and $\mathsf{E}(\mathsf{freq}(l_{\max}))$, to the query user. Consequently, in the baseline scheme, the query user has to decrypt one more ciphertexts instead of executing (2) in Section IV-B5. Fig. 3(d) shows the time cost for recovering $l_{\max}$ and $\mathsf{freq}(l_{\max})$ on the query user side, which indicates that the Paillier decryption is much more expensive than calculating (2).

C. Communication Overhead

One of the novelties of our work is that we make use of the ciphertext packing technique and *order-preserving merge function* to achieve communication efficiency. Employing the same parameters discussed in Section VI-B, we compare the communication overhead of *frequency count protocol* and *frequency comparison protocol* between the proposed scheme and the baseline scheme. Fig. 4(a) plots the communication overhead of *frequency count protocol* in both schemes with the selected data set size varying from 50 to 500. From the figure, it is evident that the communication overhead of *frequency count protocol* in the proposed scheme is halved by employing the ciphertext packing technique to compress the permuted matrix R_c. In Fig. 4(b), we compare the communication overhead of *frequency comparison protocol*. Since there are $n-1$ rounds of interaction between S_1 and S_2, and a fixed number of ciphertexts are sent per round, the communication overhead in both schemes increases linearly with the selected data set size. In the proposed scheme, S_1 only needs to transfer two ciphertexts to S_2, while four ciphertexts are transferred in the baseline scheme. Hence, the communication overhead of *frequency comparison protocol* in the proposed scheme is reduced to half by using the *order-preserving merge function*.

VII. Related Work

Finding Frequent Items: In the realm of data mining, finding frequent items, i.e., heavy hitters, can be defined

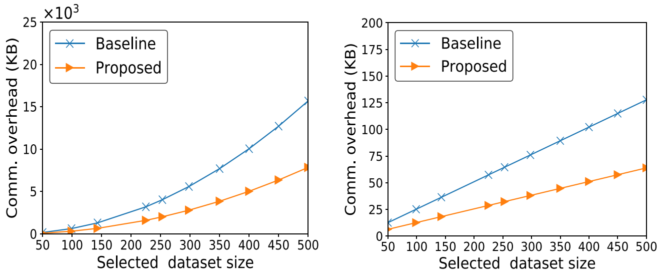

Fig. 4. Communication overhead comparisons between the proposed scheme and the baseline scheme varying selected data set size. (a) Frequency count protocol. (b) Frequency comparison protocol.

as identifying the items either with top-k frequency values or the values that occur more than a certain threshold. Cormode and Muthukrishnan [22] presented methods for dynamically determining the hot items in a relation. In the scheme, a small data structure is maintained to monitor the transactions on the relation. It can be used to return all hot items without scanning the relation when finding frequent items. In the survey [23], many schemes are investigated in addressing the problem of finding frequent items. In the study, the existing algorithms can be classified into three categories: 1) sampling-based; 2) counting-based; and 3) hashing-based algorithm. Recently, Song *et al.* [24] proposed a novel approach to find the top-k frequent items in a window of any specified size within an upper bound, which are estimated by the k items/groups with top aggregate values. However, all of the above schemes only focus on improving the performance in time and space and do not consider the security and privacy in the outsourcing scenario. Although, in 2019, Wang *et al.* [10] proposed a solution for finding the most-frequent values in a privacy-preserving manner by using a local differential privacy technique, it can neither output the accurate most-frequent values nor fully preserve the privacy of query results.

Location Privacy in MCS: The location information plays a critical role in the MCS system, and most of the MCS applications require the workers to report tasks' locations, which is bound to expose the personal location information of workers. Therefore, there are many researches [25]–[27] for preserving the location privacy while providing necessary services in the MCS system. The very popular solutions are clocking [28] and adding dummy points. Pournajaf *et al.* [29] exploited hiding the worker's real location inside a cloaked region, whereas Kido *et al.* [30] studied on transferring the real location with false location data (dummies), which have temporal consistency, to the service provider. Another solution is to use k-anonymity to protect the worker's real location from k locations [31]–[33]. Although these schemes studied the k-anonymity technique in the LBS scenario, they can be easily employed in the MCS scenario as well [26]. Recently, in the MCS system, using the differential privacy technique to protect the location information has attracted a lot of attention. Wang *et al.* [34] proposed a location privacy-preserving scheme to protect workers' location information in the task allocation stage. In their solution, the reported location is obfuscated under the guarantee of differential privacy. Yan *et al.* [35] introduced the differential privacy technique to preserve the location privacy for task selection in the MCS system. Wang *et al.* [36] applied differential location privacy to the sparse MCS. In this work, a probabilistic obfuscation matrix, which satisfies ϵ-differential privacy, is generated to obfuscate the real location to another one. However, the existing location protection schemes either lose the accuracy of the original location or add noises to the reported location. Therefore, they are unavailable for offering the accurate result of the most-frequent location query in the MCS system.

VIII. Conclusion

In this article, we have proposed the first location privacy-preserving scheme that can offer the accurate most-frequent item query in the MCS system. The proposed scheme is characterized by employing our pseudonymization mechanism, location encoding algorithm SLE, transposition cipher, ciphertext packing technique, and *order-preserving merge function* to not only preserve the privacy but also ensure efficiency. Security analysis shows that our proposed scheme is indeed privacy-preserving under the defined security model. In addition, extensive performance experiments are conducted, and the results indicate the proposed scheme is really efficient in terms of computational costs and communication overhead compared to the baseline scheme. In our future work, we plan to evaluate our proposed scheme in a real platform, also would like to extend our work to support efficient and privacy-preserving top-k frequent items queries in MCS.

References

[1] X. Kong, X. Liu, B. Jedari, M. Li, L. Wan, and F. Xia, "Mobile crowdsourcing in smart cities: Technologies, applications, and future challenges," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8095–8113, Oct. 2019.

[2] D. Zhang, L. Wang, H. Xiong, and B. Guo, "4W1H in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 42–48, Aug. 2014.

[3] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2419–2465, 3rd Quart., 2019.

[4] X. Zhang, R. Lu, J. Shao, H. Zhu, and A. A. Ghorbani, "Secure and efficient probabilistic skyline computation for worker selection in mcs," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11524–11535, Dec. 2020.

[5] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels, "Noisetube: Measuring and mapping noise pollution with mobile phones," in *Information Technologies in Environmental Engineering*. Berlin, Germany: Springer, 2009, pp. 215–228.

[6] S. Kim, C. Robson, T. Zimmerman, J. Pierce, and E. M. Haber, "Creek watch: Pairing usefulness and usability for successful citizen science," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2125–2134.

[7] H. Ma, D. Zhao, and P. Yuan, "Opportunities in mobile crowd sensing," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 29–35, Aug. 2014.

[8] Q. Zhao, C. Zuo, G. Pellegrino, and Z. Lin, "Geo-locating drivers: A study of sensitive data leakage in ride-hailing services," in *Proc. Annu. Netw. Distrib. Syst. Security Symp.*, Feb. 2019, pp. 1–15.

[9] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2971–2992, Aug. 2018.

[10] T. Wang, N. Li, and S. Jha, "Locally differentially private heavy hitter identification," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 9, 2019, doi: 10.1109/TDSC.2019.2927695.

[11] Y. Zheng, R. Lu, and M. Mamun, "Privacy-preserving computation offloading for time-series activities classification in ehealthcare," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[12] G. Lasry, *A Methodology for the Cryptanalysis of Classical Ciphers with Search Metaheuristics*. Kassel, Germany: Kassel Univ. Press GmbH, 2018.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1999, pp. 223–238.

[14] Y. Zheng, R. Lu, and J. Shao, "Achieving efficient and privacy-preserving k-NN query for outsourced ehealthcare data," *J. Med. Syst.*, vol. 43, p. 123, Mar. 2019.

[15] R. Durstenfeld, "Algorithm 235: Random permutation," *Commun. ACM*, vol. 7, no. 7, p. 420, 1964.

[16] G. Niemeyer. (2008). *Geohash*. Accessed: Feb. 18, 2021. [Online]. Available: http://geohash.org/site/tips.html

[17] G. M. Morton, *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. Ottawa, ON, Canada: Int. Bus. Mach. Company, 1966.

[18] *Geohash Maven Repository*. Accessed: Feb. 18, 2021. [Online]. Available: https://search.maven.org/search?q=g:ch.hsr

[19] *Z-Order Curve Implementation*. Accessed: Feb. 18, 2021. [Online]. Available: https://github.com/eren-ck/MortonLib

[20] JohnSnowLabs. (2018). *Nypd Motor Vehicle Collisions*. Accessed: Feb. 18, 2021. [Online]. Available: https://datahub.io/JohnSnowLabs/nypd-motor-vehicle-collisions

[21] ISO 6709. (2008). *Standard Representation of Geographic Point Location by Coordinates*. Accessed: Feb. 18, 2021. [Online]. Available: https://www.iso.org/obp/ui/iso:std:iso:6709:ed-2:v1:en

[22] G. Cormode and S. Muthukrishnan, "What's hot and what's not: Tracking most frequent items dynamically," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 249–278, 2005.

[23] H. Liu, Y. Lin, and J. Han, "Methods for mining frequent items in data streams: An overview," *Knowl. Inf. Syst.*, vol. 26, no. 1, pp. 1–30, 2011.

[24] C. Song, X. Liu, T. Ge, and Y. Ge, "Top-k frequent items and item frequency tracking over sliding windows of any size," *Inf. Sci.*, vol. 475, pp. 100–120, Feb. 2019.

[25] Z. Wang *et al.*, "When mobile crowdsensing meets privacy," *IEEE Commun. Mag.*, vol. 57, no. 9, pp. 72–78, Sep. 2019.

[26] L. Pournajaf, D. A. Garcia-Ulloa, L. Xiong, and V. Sunderam, "Participant privacy in mobile crowd sensing task management: A survey of methods and challenges," *ACM SIGMOD Rec.*, vol. 44, no. 4, pp. 23–34, 2016.

[27] Y. Wang, Z. Yan, W. Feng, and S. Liu, "Privacy protection in mobile crowd sensing: A survey," *World Wide Web*, vol. 23, no. 1, pp. 421–452, 2020.

[28] C.-Y. Chow, M. F. Mokbel, and X. Liu, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *GeoInformatica*, vol. 15, no. 2, pp. 351–380, 2011.

[29] L. Pournajaf, L. Xiong, V. Sunderam, and S. Goryczka, "Spatial task assignment for crowd sensing with cloaked locations," in *Proc. IEEE 15th Int. Conf. Mobile Data Manag.*, vol. 1, 2014, pp. 73–82.

[30] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Proc. Int. Conf. Pervasive Services*, 2005, pp. 88–97.

[31] X. Liu, K. Liu, L. Guo, X. Li, and Y. Fang, "A game-theoretic approach for achieving *k*-anonymity in location based services," in *Proc. IEEE INFOCOM*, 2013, pp. 2985–2993.

[32] J. Cui, J. Wen, S. Han, and H. Zhong, "Efficient privacy-preserving scheme for real-time location data in vehicular ad-hoc network," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3491–3498, Oct. 2018.

[33] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving k-anonymity in privacy-aware location-based services," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2014, pp. 754–762.

[34] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, "Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 627–636.

[35] K. Yan, G. Luo, X. Zheng, L. Tian, and A. M. V. V. Sai, "A comprehensive location-privacy-awareness task selection mechanism in mobile crowd-sensing," *IEEE Access*, vol. 7, pp. 77541–77554, 2019.

[36] L. Wang, D. Zhang, D. Yang, B. Y. Lim, and X. Ma, "Differential location privacy for sparse mobile crowdsensing," in *Proc. IEEE 16th Int. Conf. Data Min. (ICDM)*, 2016, pp. 1257–1262.

Songnian Zhang received the M.S. degree from Xidian University, Xi'an, China, in 2016. He is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.

His research interests include cloud computing security, big data query, and query privacy.

Suprio Ray (Member, IEEE) received the Ph.D. degree from the Department of Computer Science, University of Toronto, Toronto, ON, Canada, in 2015.

He is an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. His research interests include big data and database management systems, run-time systems for scalable data science, provenance and privacy issues in big data, and data management for the Internet of Things.

Rongxing Lu (Fellow, IEEE) is currently an Associate Professor with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada. He has published extensively in his areas of expertise. His research interests include applied cryptography, privacy enhancing technologies, and IoT-big data security and privacy.

Dr. Lu was a recipient of the nine best (student) paper awards from some reputable journals and conferences. He is the Winner of Excellence in Teaching Award from 2016 to 2017, FCS, UNB. He currently serves as the Vice-Chair (Conferences) of IEEE ComSoc CIS-TC (Communications and Information Security Technical Committee).

Yandong Zheng received the M.S. degree from the Department of Computer Science, Beihang University, Beijing, China, in 2017. She is currently pursuing the Ph.D. degree with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.

Her research interests include cloud computing security, big data privacy, and applied privacy.

Jun Shao received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2008.

He was a Postdoctoral Fellow with the School of Information Sciences and Technology, Pennsylvania State University, Pennsylvania, PA, USA, from 2008 to 2010. He is currently a Professor with the School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current research interests include network security and applied cryptography.