applied sciences

MDPI

Article

A Security-Enhanced Query Result Verification Scheme for Outsourced Data in Location-Based Services

Guangcan Yang [1], Jiayang Li [1], Yunhua He [1], Ke Xiao [1,*], Yang Xin [2], Hongliang Zhu [2] and Chen Li [1]

[1] School of Information Science and Technology, North China University of Technology, Beijing 100144, China
[2] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[*] Correspondence: xiaoke@ncut.edu.cn

Abstract: Location-based services (LBSs) facilitate people's lives; location-based service providers (LBSPs) usually outsource services to third parties to provide better services. However, the third party is a dishonest entity that might return incorrect or incomplete query results under the consideration of saving storage space and computation resources. In this paper, we propose a security-enhanced query result verification scheme (SEQRVS) for the outsourced data in a LBS. Specifically, while retaining fine-grained query result verification, we improve the construction process of verification objects to enhance the security of the outsourced data. To prevent the third party from deducing the knowledge of the outsourced data stored in itself (statistically), our scheme designs a novel storage structure to enhance the ability of privacy preservation for the outsourced data. Furthermore, based on the secure keyword search and query result verification mode proposed in our scheme, the user cannot only verify the correctness and completeness of the query result but also achieve consistency verification by the blockchain. Finally, the security analysis and extensive simulation results show the security and practicality of the proposed scheme.

Keywords: location-based service (LBS); query result verification; privacy preservation; blockchain

check for updates

Citation: Yang, G.; Li, J.; He, Y.; Xiao, K.; Xin, Y.; Zhu, H.; Li, C. A Security-Enhanced Query Result Verification Scheme for Outsourced Data in Location-Based Services. *Appl. Sci.* **2022**, *12*, 8126. https://doi.org/10.3390/app12168126

Academic Editor: Arcangelo Castiglione

Received: 17 July 2022
Accepted: 8 August 2022
Published: 13 August 2022

1. Introduction

Location-based services (LBSs) are pervasive in people's social lives. With the increase in LBS applications, users can enjoy many convenient social services, such as map navigation, restaurant recommendations, and taxi reservations. One typical LBS is the point of interest (POI) information query system. By inputting locations and POI types, users can reach relevant POI information [1]. However, since the location-based service provider (LBSP) has to maintain exponentially-growing POI data to provide a better service, storing and computing data have been burdens for LBSP.

Outsourcing a service, as a prevalent service mode, has many advantages, such as cost saving, quick deployment, and flexible resource configuration [2]. In this mode, enterprises could migrate their service data to a third party, such as the cloud or fog side, and outsource their services to the cloud or fog server. Motivated by the rich benefits brought about by outsourcing services, the LBSP could reduce its burden by utilizing the computing and storage resources of a third party [3–5]. However, how to guarantee the confidentiality of outsourced data has become a key problem (due to the separation from the direct control of outsourced data) [6].

To address this problem, one common way is to encrypt the service data before outsourcing. Therefore, many research studies focus on how to search for encrypted data. Searchable encryption, as a method that enables the outsourced data to be searched without decrypting, has been adopted in many research studies [4,7,8]. For example, searchable encryption is used in [7] to store electronic health records, allowing different participating healthcare organizations and individuals (e.g., physicians, hospitals, medical laboratories, and insurance companies) to securely access electronic health records, enabling efficient data sharing. However, driven by illegal profits, such as saving storage costs, the third party may behave dishonestly.

Therefore, many researchers have designed query result verification mechanisms to guarantee the correctness and completeness of query results. For example, researchers in [9] proposed a fine-grained query result verification scheme. In this scheme, the verification object of the query result was constructed by the Bloom filter, and a user who received the query result can check the correctness and completeness by verifying the corresponding verification object. Although the scheme achieves a fine-grained query result verification, the construction process is not perfect. Specifically, the structure of the certain verification object has an exceptional layout, which will provide additional knowledge to the third party and further lead to the privacy disclosure of the outsourced data. For example, if the LBSP outsources its service data to the cloud side and adopts this query result verification mechanism, the cloud side could easily figure out which is the largest number of encrypted outsourced POI type data.

The schemes [10–12] also provide verification solutions to support the check of the query result. Although the confidentiality of the query index and the outsourced data can be guaranteed by the method of encryption, the relationship between the encrypted query index and corresponding encrypted query objects (e.g., the verification object) is one-to-one. That is, the above query result verification mechanisms will provide additional statistical information to the third party and further lead to the information leakage of the encrypted outsourced data. For example, if the LBSP outsources its service data to the cloud side and adopts the above query result verification mechanisms, the cloud side could infer the meaning of the encrypted POI type by counting the frequency of the query indexes and popularity of POI types.

Although current schemes have proposed various query result verification mechanisms to prevent the third party from returning erroneous or incomplete query results, there is little research on the non-repudiation of the returned data. In practice, the third party may blame the situation of returning erroneous or incomplete query result on the communication process. For example, when the user finds out the erroneous or incomplete query result by verifying the corresponding verification object, the third party may claim that it has returned the correct and complete query result and the reason for missing data in other aspects such as the network communication problem.

To address the above problems, we designed a Security Enhanced Query Result Verification Scheme (SEQRVS) for the outsourced data in LBS. Specifically, the contribution of our paper can be summarized as follows.

(1) Based on the outsourcing service of LBS, a secure keyword search and query result verification mode over encrypted outsourced data were constructed. In this mode, we improved the construction process of the verification object on the basis of analyzing the deficiency of the scheme [9]. Therefore, while retaining fine-grained query result verification, our scheme can effectively prevent the third party (i.e., the fog side) from obtaining additional knowledge from the structure of the verification object, and further enhance the security of the outsourced data.

(2) To prevent the third party (i.e., the fog side) from deducing the meaning of the outsourced data stored in itself by the way of statistics, we designed the one-to-n lookup table as the storage structure of the outsourced data. By implementing this storage structure, the fog side not only knows nothing about what data are requested by the user and which query object is returned to the user, but also cannot determine the correspondence between the query index and the corresponding query object, which further enhances the ability of privacy preservation for the outsourced data.

(3) To prevent the third party (i.e., the fog side) from attributing dishonest behaviors (e.g., storage errors) to the unreliability of network communication, we introduced the blockchain to guarantee the non-repudiation of the query result from the third party.

(4) A comprehensive security analysis is provided to show the security of the mode and the storage structure adopted in our scheme, and extensive simulation results also demonstrate the security and practicality of the proposed scheme.

The rest of this paper is organized as follows. We review the related work in Section 2. The background, including the system model, threat model, and preliminary techniques to be used in the paper are described in Section 3. We propose the process of data outsourcing, data retrieval, and data verification of the scheme in Section 4. Then, we analyze the security and evaluate the performance of our proposed scheme in Sections 5 and 6. In Section 7, we conclude the paper.

2. Related Works

In this section, we review some recent research work on privacy preservation, including secure storage, query, verifiable search, and methods to resist dishonest behavior.

2.1. Secure Storage and Query

Since the introduction of technologies, such as storage and computing in the cloud to LBS, many researchers have conducted work on how to ensure secure storage and query of outsourced data. Based on attribute encryption, linear encryption, and RSA encryption, Huang et al. [13] introduced a private-protected spatial–temporal LBS searchable framework, which effectively solved the problem of an expressive and practical search over encrypted LBS data. In [14], Wang et al. designed a secure dynamic spatial keyword query (SDSKQ) structure and proposed cryptographic text-signed quad-trees to improve search security, satisfying the requirements in practical applications, such as dynamic updates and diverse query type queries. Zhang et al. [15] also adopted the structure of quad-trees to build the index for the POI database, solving the secure problem of linear region search, and bridging the gap in research on linear region search. In [16], Guo et al. improved on the existing k-anonymity algorithm, compensating for the fact that the existing algorithm can leak some contents of POIs, providing a foundation for subsequent research on the security of the k-anonymity algorithm. Manju et al. [17] proposed a fog-assisted privacy protection scheme for LBS to process the users' query in the fog server, efficiently protecting privacy security, solving the problem of double identity attacks in mobile phones. All of the above studies have addressed the issue of secure data storage and query to some extent; however, few papers have considered the pitfalls of the cloud as a potential attacker, which may leak or forge the outsourced data to reach illegal benefits.

2.2. Verifiable Search

To ensure that LBS works without a hitch, verifying the outsourced data returned from the cloud is also an essential part of the process. Yin et al. [9] constructed a data verification object using the Bloom filter to verify the correctness and completeness of data, realizing fine-grained and efficient verification of data. In [18], Zhu et al. implemented fine-grained access control based on blind signatures and key policy attribute-based encryption (KPABE), and used function hidden inner product encryption (FHIPE) to encrypt Bloom filters for data file authentication, achieving secure and efficient data validation. In [19], Zhou et al. devised a lightweight and secure comparison protocol LSCP without interactions between the cloud and users. By using this protocol, users can easily eliminate duplicate and useless encrypted LBS messages during the authentication process, making the verification process in VANETs much faster. Benarous et al. [20] proposed a privacy-preserving scheme for verifying location data transmission for the Internet of Vehicles, improving the security of privacy for LBS of vehicles. All of the above studies on data verifiability have certain flaws. There is no guarantee that dishonest clouds will not obtain useful information from them, and they cannot be directly applied to the data verification process in LBS either.

2.3. Resistance to Dishonest Behavior

How to prevent dishonest behavior in the cloud is also a current area of research. Liao et al. [21] designed the continuous query KAT algorithm to prevent the cloud from analyzing the user's privacy based on the user's trajectory. By using this algorithm, the situation that the user's privacy may be compromised by the continuous query can be effectively solved. To further protect privacy, Kuang et al. [22] designed double-hidden regions in k-anonymity algorithms to prevent the cloud from accessing user privacy. In previous research, scholars have never really looked at the cloud as an attacker, to disrupt the entire LBS system, to examine the implementation options. However, in practical situations, especially when massive amounts of location data are increasingly and profoundly affecting people's lives, it is necessary to consider the cloud as an entity that threatens the security of the entire LBS system.

Therefore, in this work, we use the Paillier cryptosystem and lookup table technology to guarantee storage and query privacy. We use verification objects to provide a fine-grained and reliable verification of fog nodes and outsourced data. Moreover, we use blockchain technology to effectively prevent dishonest behaviors of fog nodes.

3. Background

In this section, the techniques used in the scheme are introduced firstly, then the system model and threat model are illustrated.

3.1. Preliminaries

To better illustrate our scheme, we briefly introduce several key techniques used in the scheme, including the Voronoi diagram used to divide the fog nodes, the counting Bloom filter used to construct the verification objects, and so on.

3.1.1. Voronoi Diagram

A Voronoi diagram [23] is composed of several geometries in a plane, and each geometric is known as the Voronoi polygon. These Voronoi polygons are generated by a set of generator points, and the generation process is as follows. Given a set of generator points specified beforehand, by making perpendicular bisectors for the straight lines composed of generator points, the whole plane can be divided into several regions. For each point x in the region R, the distance between x and generator point P_i is less than that between x and any other generator points, i.e., $\forall x \in R(P_i), dist(x, P_i) < dist(x, P_j)$, where $P_i \neq P_j$. According to the properties of the Voronoi diagram, if fog nodes can be seen as generator points, then a two-dimensional map can be divided into a Voronoi diagram. Figure 1 is an example of a Voronoi diagram based on fog nodes.

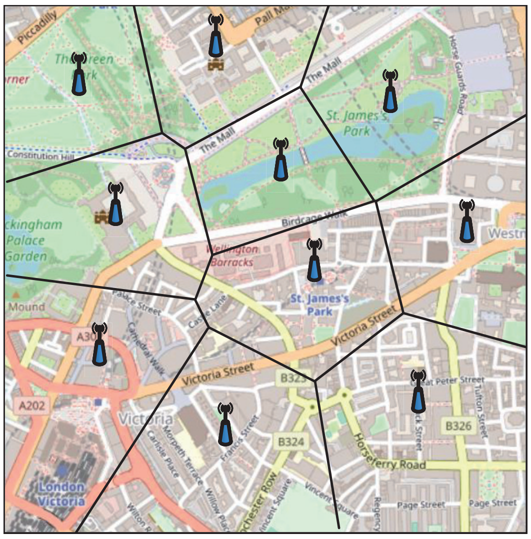


Figure 1. Voronoi diagram based on fog nodes (for an example in a UK location).

3.1.2. Counting Bloom Filter

The counting Bloom filter [24] is an improved version of the Bloom filter, which can support elemental addition and deletion operations. Before introducing the counting Bloom filter, it is necessary to explain the Bloom filter. The structure of a Bloom filter is a bit array of m bits, where each bit is set to 0 initially. The Bloom filter is usually used to determine whether an element is in or not in certain sets. Suppose that there is a set $S = \{s_1, s_2, \ldots, s_t\}$ of t elements. To enable an element, $s \in S$ can be represented in the Bloom filter; it needs to use l independent hash functions $\{h_1, h_2, \ldots, h_l\}$ to hash s to obtain l different positions in the Bloom filter, and these hash functions are with the same output range $[0, m-1]$. Then these l different positions in the structure of the Bloom filter are set to be 1. To decide whether $x \in S$, it needs to examine whether all the positions in the structure of the Bloom filter corresponding to $h_i(x)$ are equal to 1, where $1 \leq i \leq l$. Therefore, if one of these corresponding positions is 0, $x \notin S$. If all the corresponding positions are 1, then $x \in S$ or a false positive. To minimize the impact of the false positive, our scheme adopts the same way as the paper [9]. That is, the parameter l is set to be $\frac{m*ln2}{t}$, which will result in the minimum probability of the false positive (i.e., 2^{-l}). However, since each position of the standard Bloom filter only represents a single bit, it does not support element addition and deletion operations. Thus, the counting Bloom filter uses fixed size counters to represent an element instead of single bits. In this case, the corresponding counters are added by 1 when an element is inserted and the corresponding counters are decreased by 1 when an element is deleted.

3.1.3. Paillier Cryptosystem

The Paillier cryptosystem is a classic homomorphic encryption and is usually used to implement addition operations over the ciphertext domain [25]. In general, it consists of three polynomial-time algorithms (i.e., *Gen*, *Enc*, and *Dec*).

$Gen(1^n)$: Firstly, two independent large prime numbers p and q are randomly selected. Then we compute $N = p \cdot q$ and $\lambda = lcm(p-1, q-1)$, where $lcm(\ )$ is the least common multiple function. Finally, the public key $pk = N$ and private key $sk = (\lambda, \psi(N))$ can be obtained, where $\psi(N) = \lambda^{-1} mod\ N$.

$Enc(pk, m)$: Assume m is a plaintext to be encrypted. Firstly, a random number $r \in \mathbb{Z}_N^*$ is selected. Then the encrypted result c can be computed by Equation (1):

$$c = [(1+N)^m \cdot r^N mod\ N^2] \tag{1}$$

$Dec(sk, c)$: To obtain the plaintext m, the encrypted result c can be recovered with the private key sk by Equation (2):

$$m = \frac{(c^\lambda mod N^2) - 1}{N} \cdot \psi(N)\ mod\ N \tag{2}$$

3.2. System Model

As shown in Figure 2, the proposed scheme has four entities: the LBSP, the fog node, the user, and the blockchain, which describe the following scenario: due to the storage and computing advantages of fog computing, the LBSP outsources its private data to the fog nodes. However, to guarantee data security, the outsourced data need to be encrypted and the storage structures of outsourced data also need to be constructed to support the secure keyword search over the encrypted outsourced data. When a user requests a certain type of POI information in a specified query region, s/he can send a POI query request to the third party (i.e., the fog node) based on some parameters obtained from the LBSP. Similar to [9], the third party (i.e., the fog node) is considered to be a dishonest entity that could maliciously delete the stored outsourced data or tamper with the user's query result; the secure verification objects should be constructed and contained in the outsourced data. When a query ends, the query result along with the corresponding verification objects are returned to the user. Finally, the user can implement the correctness and completeness verification based on the received verification objects, taking a step towards consistency verification under the support of the blockchain.

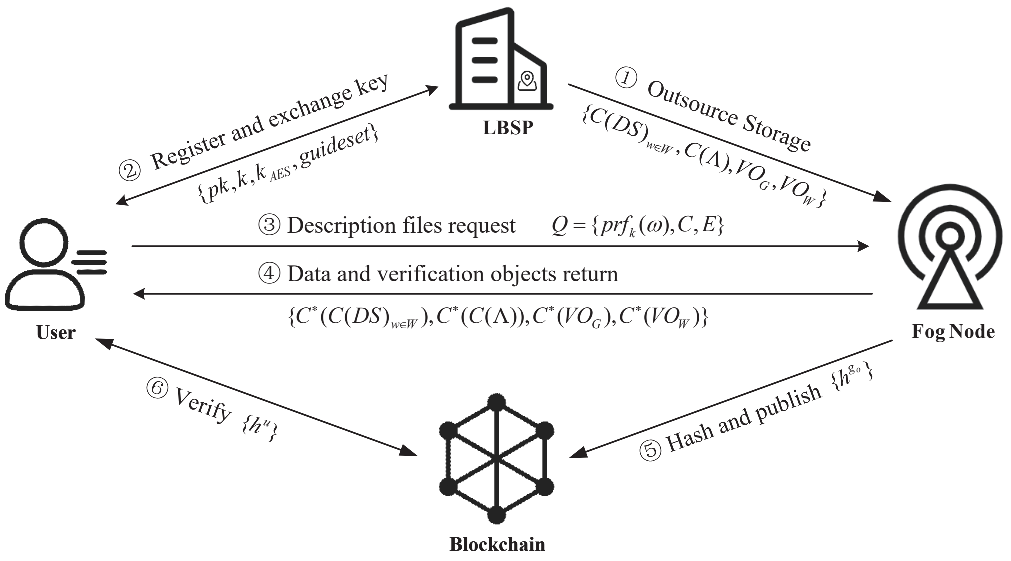

LBSP
User
Fog Node
Blockchain
① Outsource Storage
$\{C(DS)_{w\in W}, C(\Lambda), VO_G, VO_W\}$
② Register and exchange key
$\{pk, k, k_{AES}, guideset\}$
③ Description files request $Q=\{prf_k(\omega), C, E\}$
④ Data and verification objects return
$\{C^*(C(DS)_{w\in W}), C^*(C(\Lambda)), C^*(VO_G), C^*(VO_W)\}$
⑤ Hash and publish $\{h^{g_o}\}$
⑥ Verify $\{h^u\}$

Figure 2. System model.

3.3. Threat Model

Our threat model is mostly consistent with work in [9]. The LBSP and the blockchain are assumed to be honest; that is, they honestly behave similar to the scheme designed. To some extent, they can be regarded as reliable entities without leaking any privacy of users or colluding with the fog side. The fog side is assumed to be dishonest and curious; that is, they may maliciously discard or tamper with outsourced data to obtain improper benefits while attempting to analyze not only the users' queries themselves and their frequencies but also the outsourced encrypted data. The two threat models are described as follows:

Given two fog node verification objects VO_g, $VO_{g'}$ of two different fog node identifiers g, g', the attackers cannot tell the difference between the above two verification objects. The semantic security (i.e., fog node verification object indistinguishability security) of the fog node verification object is defined by a game between probabilistic polynomial-time adversary $\mathcal{A}$ and challenger $\mathcal{B}$.

($\widehat{Game}1$)

(1) Setup.
(2) Phase 1. $\mathcal{A}$ asks $\mathcal{B}$ to return the fog node verification objects after submitting different fog node identifiers many times.
(3) Challenge. $\mathcal{A}$ sends two challenge fog node identifiers g_1, g_2 to $\mathcal{B}$, which have not been sent in Phase 1. $\mathcal{A}$ asks $\mathcal{B}$ for the challenge fog node verification object. After receiving two challenge fog node identifiers g_1, g_2, $\mathcal{B}$ fairly chooses a bit $b \in \{1,2\}$ and returns the challenge fog node verification object VO_{g_b} of g_b to $\mathcal{A}$.
(4) Phase 2. $\mathcal{A}$ continues to ask $\mathcal{B}$ to return the fog node verification objects after submitting different fog node identifiers many times. The only restriction is that the fog node identifiers are different from the identifiers sent in Phase 1 and Challenge.
(5) Guess. The adversary $\mathcal{A}$ outputs a guess b' of b. If $b = b'$, $\mathcal{A}$ wins the game.

As the space of b is only two, $\mathcal{A}$ can choose the correct number with the probability of 50% if taking a random guess.

Definition 1. *The advantage of the probabilistic polynomial-time adversary $\mathcal{A}$ wins ($\widehat{Game1}$) is*

$$Adv_{\mathcal{A}} = \mid Pr[b = b^{'}] - \frac{1}{2} \mid . \tag{3}$$

If $Adv_{\mathcal{A}}$ is negligible in the game, the fog node verification object is semantically secure and achieves indistinguishability.

Given two description file verification objects $VO_w, VO_{w'}$ of set $c(ds)_w, c(ds)_{w'}$ for two different keywords $w, w^{'}$, the attackers cannot tell the difference between the above two verification objects. The semantic security (i.e., the description file verification object achieving indistinguishability) of the description file verification object is defined by a game between probabilistic polynomial-time adversary $\mathcal{A}$ and challenger $\mathcal{B}$.

($\widehat{Game2}$)

(1) Setup.
(2) Phase 1. $\mathcal{A}$ asks $\mathcal{B}$ to return the corresponding description file verification objects after submitting different keywords many times.
(3) Challenge. $\mathcal{A}$ sends two challenge keywords w_1, w_2 to $\mathcal{B}$, which have not been sent in Phase 1. $\mathcal{A}$ asks $\mathcal{B}$ for the challenge description file verification objects. After receiving two challenge keywords, w_1, w_2, $\mathcal{B}$ fairly chooses a bit $b \in \{1, 2\}$ and returns the challenge description file verification object VO_{w_b} of w_b to $\mathcal{A}$.
(4) Phase 2. $\mathcal{A}$ continues to ask $\mathcal{B}$ to return the corresponding location data file verification objects after submitting different keywords many times. The only restriction is that the keywords are different from the keywords sent in Phase 1 and Challenge.
(5) Guess. The adversary $\mathcal{A}$ outputs a guess $b^{'}$ of b. If $b = b^{'}$, $\mathcal{A}$ wins the game.

As the space of b is only two, $\mathcal{A}$ can choose the correct number with the probability of 50% if taking a random guess.

Definition 2. *The advantage of the probabilistic polynomial-time adversary $\mathcal{A}$ winning ($\widehat{Game2}$) is*

$$Adv_{\mathcal{A}} = \mid Pr[b = b^{'}] - \frac{1}{2} \mid . \tag{4}$$

If $Adv_{\mathcal{A}}$ is negligible in the game, the description file verification object is semantically secure and achieves indistinguishability.

4. Proposed Scheme

In this section, we first describe the overview of our scheme. Then, based on three main processes, the proposed scheme is explained in detail. The summary of notations is presented in Table 1.

Table 1. Summary of notations.

Notation	Description
G	The identifier set of fog nodes.
g	The identifier of a fog node.
W	The keyword set related to POI types.
w	A keyword in set W.
DS_W	The set of description files containing all keywords.
$(ds)_w$	The set of description files containing keyword w.
$C(X)$	The corresponding ciphertext set of plaintext X.
prf_k	A pseudo-random function with the key k.
c	A secure encryption algorithm, e.g., AES.
(pk, sk)	Key pair generated by Paillier cryptosystem.
VO_G	The verification object set of all fog nodes.

Table 1. *Cont.*

Notation	Description
VO_g	The verification object of a fog node.
$vo_g[\,]$	The position in the verification object of a fog node.
VO_W	The verification object set of description files containing all keywords.
VO_w	The verification object of description files containing keyword w.
$vo_w[\,]$	The counter in the verification object of description files containing keyword w.
H	The set of l hash functions.
h	A hash function.

4.1. Overview

In brief, the design goal of our scheme was to support the security verification of query results on the basis of providing a secure query over the outsourced data. That is, when a query ends, both the query result and corresponding verification objects are returned to the user by the fog node. Upon receiving these returned data, the user can obtain the query result that corresponds to his/her query index as well as verify the completeness and correctness of the query result according to the verification objects and check the consistency of the query result based on the blockchain.

Our scheme is mainly composed of three processes: data outsourcing process, data retrieval process, and data verification process. (1) The data outsourcing process: given a flat map (e.g., a city), the LBSP constructs a Voronoi diagram mentioned in Section 3.1. Then, according to the fog node in each Voronoi cell, the LBSP forms the original database, shown in Table 2. Note that the fog nodes can vary based on the frequency layers or in case the user devices are connected to some local Wi-Fi hotspots. Based on the original database, the LBSP begins to construct the encrypted outsourced data, such as the retrieval index, the query result, and the corresponding verification objects. While introducing the construction process of outsourced data, Section 4.2 explains the improved construction process of verification objects to fix the deficiency of the scheme [9]. (2) The data retrieval process: based on encrypted outsourced data constructed in the data outsourcing process, the storage structure of the outsourced data (i.e, one-to-n lookup table) stored in each fog node is illustrated. When a user specifies a query region, the fog nodes contained in the query region will be in charge of the user's query service. Based on the one-to-n lookup table, Section 4.3 states how the user obtains the query service from the fog nodes and how to prevent the fog nodes from deducing the meaning of the outsourced data stored in themselves by the way of statistics. (3) The data verification process: after obtaining the returned data from the fog nodes, Section 4.4 shows how to verify the correctness and completeness of the query result according to the verification objects and check the consistency of the query result based on the blockchain.

Table 2. The original database structure of the LBSP.

G	W	DS_W
g_1	w_1	$(ds)_{w_1}$
	w_2	$(ds)_{w_2}$
	…	
	w_θ	$(ds)_{w_\theta}$
g_2	w_1	$(ds)_{w_1}$
	w_2	$(ds)_{w_2}$
	…	
	w_θ	$(ds)_{w_\theta}$

Table 2. *Cont.*

G	W	DS_W
	...	
g_s	w_1	$(ds)_{w_1}$
	w_2	$(ds)_{w_2}$
	...	
	w_θ	$(ds)_{w_\theta}$

4.2. Data Outsourcing Process

The process of constructing the encrypted outsourced data will be illustrated based on the original database structure. For each keyword w related to a certain POI type and each set of description files containing the keyword w, the LBSP uses the pseudo-random function with the key k (i.e., prf_k) and AES to form the encrypted retrieval index set $prf_k(W) = \{prf_k(w_1), prf_k(w_2), \ldots, prf_k(w_\theta)\}$ and the ciphertext set $C(DS_{w\in W}) = \{c(ds)_{w_1}, c(ds)_{w_2}, \ldots, c(ds)_{w_\theta}\}$. Moreover, to support the query result verification, the verification objects need to be constructed. Similar to the scheme proposed in [9], our scheme also adopts the Bloom filter to generate verification objects.

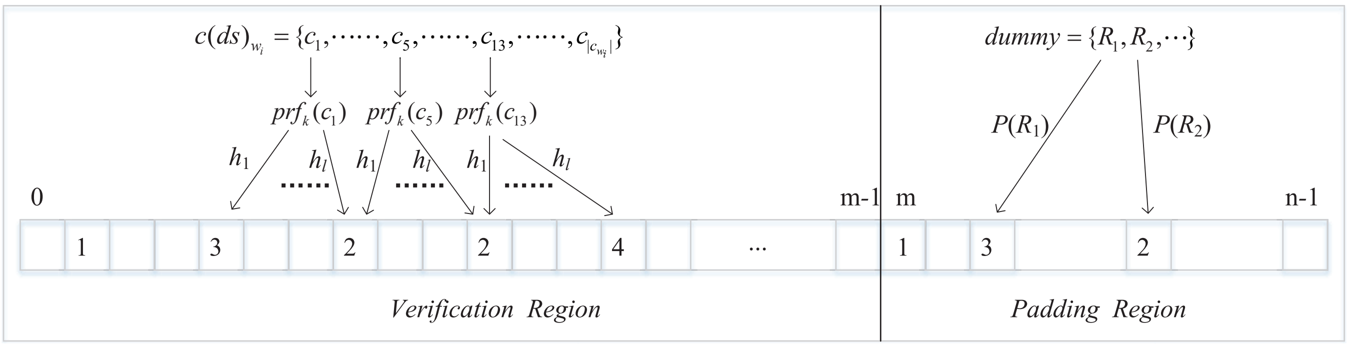
4.2.1. Design of the Fog Node Verification Object

For each fog node identifier g_i, the LBSP first uses AES to obtain the corresponding ciphertext $c(g_i)$, where $1 \le i \le s$ and s indicates the total number of fog nodes. Then, for each $c(g_i)$, the LBSP uses the pseudo-random function (i.e., prf_k) to obtain the secret value $prf_k(c(g_i))$ and prepares a standard Bloom filter VO_{g_i} with m bits, where the initial value of each bit is 0. Further, the LBSP utilizes set H that is composed of l hash functions to hash $prf_k(c(g_i))$ and obtains the set of hashed values $\{h_1(prf_k(c(g_i))) \in [0, m-1], \ldots, h_l(prf_k(c(g_i))) \in [0, m-1]\}$. Finally, the LBSP enabling these hashed values can be represented in the standard Bloom filter and sets the corresponding bit $vo_{g_i}[h_j(prf_k(c(g_i)))]$ to be 1, where $1 \le j \le l$.

4.2.2. Design of Description File Verification Objects

For each set of description files containing the keyword w_i (i.e., $c(ds)_{w_i}$), the LBSP prepares a counting Bloom filter VO_{w_i} with m counters, where the initial value of each counter is 0. Then, for each encrypted description file $c \in c(ds)_{w_i}$, the LBSP uses the pseudo-random function (i.e., prf_k) to obtain the secret value $prf_k(c)$ and utilizes the set H composed of l hash functions to hash $prf_k(c)$, obtaining the set of hashed values $\{h_1(prf_k(c)) \in [0, m-1], \ldots, h_l(prf_k(c)) \in [0, m-1]\}$. Finally, the LBSP enabling these hashed values can be represented in the counting Bloom filter, adding the corresponding counter $vo_{w_i}[h_j(prf_k(c))]$ by 1, where $1 \le j \le l$, $1 \le i \le \theta$, and θ indicate the total number of keywords.

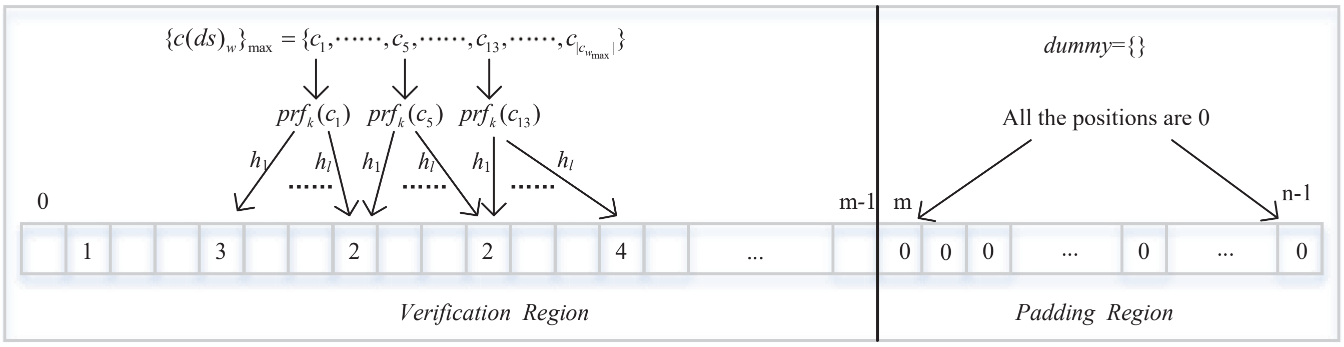
Problem statement: If the verification object VO_{w_i} is directly outsourced to the third party (i.e., the cloud side or the fog side), the number of description files represented in VO_{w_i} can be easily figured out by calculating $\frac{\sum_{j=0}^{m-1} vo_{w_i}[j]}{l}$, where l is the number of hash functions. Therefore, the scheme in [9] proposed a novel structure for the verification object to fix the above issue by adding a padding region, shown in Figure 3. In the structure, the number of counters is extended to n. The counters from 0 to $m-1$ are inserted corresponding to description files of $c(ds)_{w_i}$ by using prf_k and H, and the counters from m to n are inserted corresponding to $l \times |c(ds)_{w\in W}|_{max} - \sum_{j=0}^{m-1} vo_{w_i}[j]$ random strings $\{R_1, R_2, \ldots\}$ by using a pad function P with the range $[m, n-1]$, where $|c(ds)_{w\in W}|_{max}$ is the maximum number of description files containing the same keyword. The computed result $P(R) = m + prf_k(R)\ mod\ (n-m)$, $R \in \{0,1\}^*$ is not 0, the corresponding counter $vo_{w_i}[p(R)]$ is added by 1.



$c(ds)_{w_i} = \{c_1, \cdots\cdots, c_5, \cdots\cdots, c_{13}, \cdots\cdots, c_{|c_{w_i}|}\}$
$prf_k(c_1)$ $prf_k(c_5)$ $prf_k(c_{13})$
h_1 h_l h_1 h_l h_1 h_l
0
m-1
m
n-1
1 3 2 2 4 …
$dummy = \{R_1, R_2, \cdots\}$
$P(R_1)$
$P(R_2)$
1 3 2
Verification Region
Padding Region

Figure 3. The verification object with a padding region.

Although the padding region can efficiently prevent the third party from calculating how many description files are contained in $c(ds)_{w_i}$, the structure of the verification object is not perfect since this construction will leak some important information. For example, since each VO_{w_i} satisfies $\sum_{j=0}^{n-1} vo_{w_i}[j] = l \times |c(ds)_{w \in W}|_{max}$. Then, the structure of the verification object VO_{max} is an exceptional layout since all of the values of counters in the padding region are 0, where VO_{max} represents the verification object of $\{c(ds)_{w \in W}\}_{max}$. In other words, the value in the last m to n counters are all 0, as shown in Figure 4. Due to the layout shown in Figure 4, the third party can easily lock the verification object VO_{max} and further figure out the corresponding encrypted query. Consequently, the layout will support the third party to infer the meaning of the encrypted query index by investigating the data number of the POI type related to keyword w.



$\{c(ds)_w\}_{\max} = \{c_1, \cdots\cdots, c_5, \cdots\cdots, c_{13}, \cdots\cdots, c_{|c_{w_{\max}}|}\}$
$prf_k(c_1)$ $prf_k(c_5)$ $prf_k(c_{13})$
h_1 h_l h_1 h_l h_1 h_l
0
m-1
m
n-1
1 3 2 2 4 …
dummy={}
All the positions are 0
0 0 0 … 0 … 0
Verification Region
Padding Region

Figure 4. The structure of the special verification object.

Improved construction process of verification objects: The main idea of our improved method is to set the sum of genuine description files and dummy description files of each keyword $w \in W$ to be a fixed value $S = |c(ds)_{w \in W}|_{max} + r$, where $|c(ds)_{w \in W}|_{max}$ is the maximum number of genuine description files and r is a random value. $|c(ds)_{w \in W}|_{max} - |c(ds)_w| + r$ indicates the number of dummy description files. Suppose that $|c(ds)_{w \in W}|$ denotes the number of genuine description files for each keyword $w \in W$, then the total number of description files that need to be represented (i.e., the number that needs to be inserted into the structure of the verification object) can be set as $Sum = l \times (|c(ds)_{w \in W}|_{max} + r)$. Specifically, in the structure of the verification object VO_{w_i}, the verification region is inserted $l \times |c(ds)_{w_i}|$ times and the padding region is inserted $n_{dummy} = l \times (S - |c(ds)_{w_i}|)$ times. Finally, the sum of the inserted times (i.e., the sum of numbers in all counters) for each verification object is $l \times S$. An example of our verification object is shown in Figure 5. The improved construction process of the verification object is shown in Algorithm 1.

Algorithm 1 Improved construction process of verification objects.

```
Input: The ciphertext files sets $C(G)$ and $C(DS_{w \in W})$
Output: The verification object sets $VO_G$ and $VO_W$
1:  for each $c(g_i) \in C(G)$ do
2:      Generate a standard Bloom filter with $m$ bits;
3:      Calculate $prf_k(c(g_i))$;
4:      Calculate $\{h_1(prf_k(c(g_i))), \ldots, h_l(prf_k(c(g_i)))\}$ by using the set $H$ that composed of
        $l$ hash functions;
5:      for $1 \le j \le l$ do
6:          Set the counter $vo_{g_i}[h_j(prf_k(c(g_i)))]$ to be 1;
7:      end for
8:      return $VO_{g_i}$;
9:  end for
10: return The verification object set $VO_G$;
11: for each $c(ds)_{w_i} \in C(DS_{w \in W})$ do
12:     Generate a counting Bloom filter with $n$ counters;
13:     for each $c \in c(ds)_{w_i}$ do
14:         Calculate $prf_k(c)$;
15:         Calculate $\{h_1(prf_k(c)), \ldots, h_l(prf_k(c))\}$ using $H$;
16:         for $1 \le j \le l$ do
17:             Add the counter $vo_{w_i}[h_j(prf_k(c))]$ by 1 in the verification region;
18:         end for
19:         Generate $n_{dummy} = S - |c(ds)_{w_i}|$ dummy description files;
20:         for each $c \in c(dummy)_{w_i}$ do
21:             Calculate $prf_k(c)$;
22:             Calculate $\{h_1(prf_k(c)), \ldots, h_l(prf_k(c))\}$ using $H$;
23:             for $1 \le j \le l$ do
24:                 Add the counter $vo_{w_i}[h_j(prf_k(c))]$ by 1 in the padding region;
25:             end for
26:         end for
27:     end for
28:     return $VO_{w_i}$;
29: end for
30: return The verification object set $VO_W$;
```

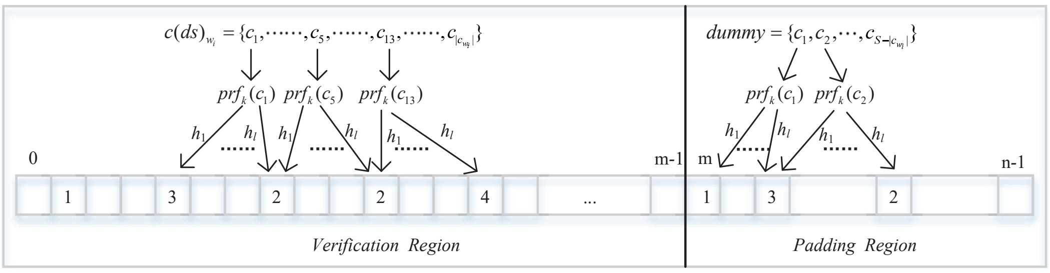


Figure 5. The structure of the verification object in our scheme.

4.2.3. Design of the Lookup Table

After the above settings, the encrypted retrieval index set (i.e., $prf_k(W)$), the encrypted query result set containing keywords (i.e., $C(DS)_{w \in W}$), the verification object set of fog nodes (i.e., VO_G), and the description file verification object set corresponding to the encrypted query result set (i.e., VO_W) are obtained. Based on the above-encrypted data, the LBSP can outsource these data to a third party (i.e., the fog side). However, how to use the user's encrypted query index to obtain the corresponding query result and the verification objects without leaking any useful information to the third party is an important problem that needs to be solved.

Problem statement. To overcome the disclosure of query privacy, it is common to use the user's encrypted query index to support the secure query over the outsourced data. Although this way can prevent the third party from obtaining the plaintext information of the user's query index (e.g., the plaintext information about the query keyword w in our scheme), it still exposes important information, such as the frequency or rarity of requests for outsourced data corresponding to the user's encrypted query index, and accurate statistical information on the frequency of all encrypted outsourced data requested. Upon the above-exposed information, the third party may carry out some dishonest behaviors, such as deleting the outsourced data that are rarely requested or inferring the meaning of the encrypted query index (e.g., $prf_k(w)$ in our scheme).

Improved storage structure of the outsourced data. In brief, the reason that a third party can obtain the exposed information is that the relationship between the encrypted query index and the corresponding query objects is designed to be one-to-one in the storage structure. Therefore, to prevent the third party from implementing the inference attack based on the above one-to-one correspondence, our scheme designs the one-to-n lookup table as the storage structure of the outsourced data to disturb the corresponding relationship between the encrypted query index and the corresponding query objects. For example, Table 3 shows the improved storage structure with the one-to-two correspondence (i.e., the one-to-two lookup table). By using this storage structure, the relationship between the encrypted query index (i.e., $prf_k(w)$) and the corresponding query objects (e.g., VO_w) stored in the fog node can achieve the goal of one-to-two. Note that each fog node only owns the encrypted outsourced data within its Voronoi cell (i.e., the LBSP outsources the lookup table $T(g)$ that contains the ciphertext data to the corresponding fog node). In the lookup Table 3, a and b are two random numbers, where $1 \leq a \leq \theta$ and $1 \leq b \leq s$. If $i + a > \theta$, then $i + a = (i + a)\ mod\ \theta$. If $i + b > s$, then $i + b = (i + b)\ mod\ s$. Moreover, ID is the identifier set of data items and each id indicates the identifier of a data item. $c(\Lambda)$ is the encrypted set of the number of genuine description files and each $c(\lambda_{w \in W})$ denotes the encrypted number of genuine description files corresponding to the set of encrypted description files (i.e.,$c(ds)_{w \in W}$), where $\lambda_{w \in W}$ indicates the number of genuine description files and it is obtained by encrypting the genuine number with AES. Based on the improved storage structure, the next subsection will show how to remedy the proposed problems with our scheme.

Table 3. Improved storage structure with one-to-two correspondence.

$T(G)$	ID	$prf_k(W)$	$C(DS)_{w \in W}$	$C(\Lambda)$	VO_G	VO_W
$T(g_1)$	id_1	$prf_k(w_1), prf_k(w_{(1+a)})$	$c(ds)_{w_1}, c(ds)_{w_{(1+a)}}$	$c(\lambda_{w_1}), c(\lambda_{w_{(1+a)}})$	$VO_{g_1}, VO_{g_{(1+b)}}$	$VO_{w_1}, VO_{w_{(1+a)}}$
	id_2	$prf_k(w_2), prf_k(w_{(2+a)})$	$c(ds)_{w_2}, c(ds)_{w_{(2+a)}}$	$c(\lambda_{w_2}), c(\lambda_{w_{(2+a)}})$	$VO_{g_1}, VO_{g_{(1+b)}}$	$VO_{w_2}, VO_{w_{(2+a)}}$
			...			
	id_θ	$prf_k(w_\theta), prf_k(w_{(\theta+a)})$	$c(ds)_{w_\theta}, c(ds)_{w_{(\theta+a)}}$	$c(\lambda_{w_\theta}), c(\lambda_{w_{(\theta+a)}})$	$VO_{g_1}, VO_{g_{(1+b)}}$	$VO_{w_\theta}, VO_{w_{(\theta+a)}}$
$T(g_2)$	id_1	$prf_k(w_1), prf_k(w_{(1+a)})$	$c(ds)_{w_1}, c(ds)_{w_{(1+a)}}$	$c(\lambda_{w_1}), c(\lambda_{w_{(1+a)}})$	$VO_{g_2}, VO_{g_{(2+b)}}$	$VO_{w_1}, VO_{w_{(1+a)}}$
	id_2	$prf_k(w_2), prf_k(w_{(2+a)})$	$c(ds)_{w_2}, c(ds)_{w_{(2+a)}}$	$c(\lambda_{w_2}), c(\lambda_{w_{(2+a)}})$	$VO_{g_2}, VO_{g_{(2+b)}}$	$VO_{w_2}, VO_{w_{(2+a)}}$
			...			
	id_θ	$prf_k(w_\theta), prf_k(w_{(\theta+a)})$	$c(ds)_{w_\theta}, c(ds)_{w_{(\theta+a)}}$	$c(\lambda_{w_\theta}), c(\lambda_{w_{(\theta+a)}})$	$VO_{g_2}, VO_{g_{(2+b)}}$	$VO_{w_\theta}, VO_{w_{(\theta+a)}}$
			...			
$T(g_s)$	id_1	$prf_k(w_1), prf_k(w_{(1+a)})$	$c(ds)_{w_1}, c(ds)_{w_{(1+a)}}$	$c(\lambda_{w_1}), c(\lambda_{w_{(1+a)}})$	$VO_{g_s}, VO_{g_{(s+b)}}$	$VO_{w_1}, VO_{w_{(1+a)}}$
	id_2	$prf_k(w_2), prf_k(w_{(2+a)})$	$c(ds)_{w_2}, c(ds)_{w_{(2+a)}}$	$c(\lambda_{w_2}), c(\lambda_{w_{(2+a)}})$	$VO_{g_s}, VO_{g_{(s+b)}}$	$VO_{w_2}, VO_{w_{(2+a)}}$
			...			
	id_θ	$prf_k(w_\theta), prf_k(w_{(\theta+a)})$	$c(ds)_{w_\theta}, c(ds)_{w_{(\theta+a)}}$	$c(\lambda_{w_\theta}), c(\lambda_{w_{(\theta+a)}})$	$VO_{g_s}, VO_{g_{(s+b)}}$	$VO_{w_\theta}, VO_{w_{(\theta+a)}}$

4.3. Data Retrieval Process

Since the process of the user registration in the LBSP is not the focal point of our scheme, we explain the data retrieval process based on the registered user who has obtained the key k of the pseudo-random function, the key k_{AES} of AES, the set of l hash functions H from the LBSP, and a hash function h_v that used to verify the consistency of the query result in the registration process. When an authenticated user wants to request a query service, s/he could specify a query region by the client-side installed on his/her mobile device (e.g., an app) and send the query region to the LBSP. Subsequently, the LBSP sends the identifier set of fog nodes contained in the query region and a guide set that is used to fix the proposed problems, such as preventing the fog nodes from obtaining accurate statistical information about the frequency of the encrypted outsourced data. Upon receiving the identifiers of fog nodes and the guide set, the user begins to communicate with the corresponding fog nodes and enjoys the outsourced data retrieval service. Herein, since the interaction process between the user side and the fog side is the same, we focus on explaining the interaction process between one user and one fog node g_o. In what follows, we first introduce how the LBSP designs a guide set based on the one-to-n lookup table and then shows how the user uses the guide set to request the encrypted outsourced data.

4.3.1. Design of Guide Set

Based on the one-to-n lookup table, the LBSP can set a guide set in the form of $F = \{f_d | 1 \leq d \leq n\}$, where n is the redundancy of the one-to-n lookup table. Specifically, a guide set F consists of a series of binary digits and each element f_d is a binary number. Herein, note that there is only one binary 'number 1' in each guide set. For example, Table 3 is a one-to-two lookup table, then the guide set can be designed in the form of {1, 0} or {0, 1}. Note here that the form of the guide set sent from the LBSP to the user is random, i.e., the guide set received by the user may be {1, 0} or {0, 1}.

4.3.2. Query Request Submission

When a user wants to request a query service, the user should submit the query request in the form of $Q = \{prf_k(\omega), C, E\}$ to the fog node g_o. In the query request Q, $\omega \in W$ indicates a keyword of interest related to a POI type and $prf_k(\omega)$ is gained by encrypting ω with the shared key k of the pseudo-random function. Moreover, C is a ciphertext set composed of a series of encrypted data in the form of $\{c_d | 1 \leq d \leq n\}$, in which c_d represents encrypted data obtained by the public key pk and an element f_d in the guide set F. Specifically, to obtain c_d, the user encrypts the f_d using the Paillier cryptosystem under the public key $pk = N$ and a random number $r_d \in \mathbb{Z}_N^*$ as follows:

$$c_d = [(1+N)^{f_d} \cdot r_d^N mod N^2] \quad . \tag{5}$$

Moreover, E represents encrypted data obtained by the public key pk and encrypted query index $prf_k(\omega)$. Specifically, to obtain E, the user encrypts the $prf_k(\omega)$ using the Paillier cryptosystem under the public key $pk = N$ and the set $\{r_d | 1 \leq d \leq n\}$ as follows:

$$E = [(1+N)^{prf_k(\omega)} \cdot (\prod_{d=1}^{n} r_d)^N mod N^2] \quad . \tag{6}$$

4.3.3. Data Retrieval

Upon receiving the user's query request $Q = \{prf_k(\omega), E, C\}$, the fog node g_o first scans the column $prf_k(W)$ of the lookup table $T(g_o)$ and finds the corresponding data items that contain $prf_k(\omega)$. According to the one-to-n lookup table, n data items can be found. For example, since Table 3 is a one-to-two lookup table, then two data items id_i and id_y can be found, where the column $prf_k(W)$ of id_i is $\{prf_k(w_i), prf_k(w_x)\}$ and the column

$prf_k(W)$ of id_y is $\{prf_k(w_y), prf_k(w_i)\}$, where $prf_k(\omega) = prf_k(w_i)$. For each found data item, the fog node performs the computation as follows:

$$\begin{aligned} C^*(prf_k(W)) &= [\prod_{d=1}^{n} c_d^{prf_k(W)} modN^2] \\ &= [(1+N)^{\sum_{d=1}^{n} f_d \cdot prf_k(w_d)} \\ &\cdot (\prod_{d=1}^{n} r^{prf_k(w_d)})^N modN^2] \\ &= [(1+N)^{1 \cdot prf_k(w_i)} \cdot (\prod_{d=1}^{n} r_d)^N modN^2] \\ &= [(1+N)^{prf_k(w_i)} \cdot (\prod_{d=1}^{n} r_d)^N modN^2] \quad , \end{aligned} \tag{7}$$

where $prf_k(w_d) \in prf_k(W)$.

Then, the fog node g_o can find the target data item id_t by checking whether $C^*(prf_k(W))$ is equal to E, i.e., when $C^*(prf_k(W)) = E$, the corresponding data item is the target data item id_t. After finding the target data item, the fog node g_o further computes:

$$C^*(C(DS)_{w \in W}) = [\prod_{d=1}^{n} c_d^{C(DS)_{w \in W}} modN^2] \quad , \tag{8}$$

$$C^*(C(\Lambda)) = [\prod_{d=1}^{n} c_d^{C(\Lambda)} modN^2] \quad , \tag{9}$$

$$C^*(VO_G) = [\prod_{d=1}^{n} c_d^{VO_G} modN^2] \quad , \tag{10}$$

$$C^*(VO_W) = [\prod_{d=1}^{n} c_d^{VO_W} modN^2] \quad . \tag{11}$$

After completing the above calculations, the fog node g_o returns $C^*(C(DS)_{w \in W})$, $C^*(C(\Lambda))$, $C^*(VO_G)$, and $C^*(VO_W)$ back to the user. Moreover, the fog uses h_v sent from the LBSP to hash $C^*(C(DS)_{w \in W})$ and sends $h^{g_o} = h_v(C^*(C(DS)_{w \in W}))$ to the blockchain. Upon receiving the returned data, the user can obtain $c(ds)_\omega$, $c(\lambda_\omega)$, VO_{g_o}, and VO_ω with the private key sk and further obtain the available result set ds_ω and the number of genuine description files λ_ω with the shared key k_{AES}.

4.4. Data Verification Process

According to the returned data, the correctness, completeness, and consistency of the query result can be verified by the user.

4.4.1. Correctness Verification

To check the correctness of the fog node g_o, the user first encrypts g_o to obtain $c(g_o)$ by using AES and further obtains $prf_k(c(g_o))$ with k. Then, with the set of l hash functions H, the user begins to calculate $\{h_1(prf_k(c(g_o))), \ldots, h_l(prf_k(c(g_o)))\}$ and checks the corresponding positions in VO_{g_o}. According to the above comparison, the user can confirm whether the returned data are sent by the fog node g_o. To check the correctness of the query result $c(ds)_\omega$, the user first calculates $\{h_1(prf_k(c)), \ldots, h_l(prf_k(c))\}$ with the set of l hash functions H for each encrypted description file c. Then, the user makes a comparison between $\{h_1(prf_k(c)), \ldots, h_l(prf_k(c))\}$ and vo_ω. If one of the counters in VO_ω is 0, this c is incorrect. The process of correctness verification is shown in Algorithm 2.

Algorithm 2 Correctness verification.

```
Input: $g_o$, $VO_{g_o}$, $k$, $H$, $c(ds)_\omega$, $VO_\omega$
Output: The correctness of $g_o$ and $c(ds)_\omega$
1: Calculate $prf_k(c(g_o))$;
2: Calculate $\{h_1(prf_k(c(g_o))), \dots, h_l(prf_k(c(g_o)))\}$ using $H$;
3: Check all positions $h_1(prf_k(c(g_o))), \dots, h_l(prf_k(c(g_o)))$ in $VO_{g_o}$. If one of them is equal
   to 0, $g_o$ is incorrect;
4: for each $c \in c(ds)_\omega$ do
5:    Calculate $prf_k(c)$;
6:    Calculate $\{h_1(prf_k(c)), \dots, h_l(prf_k(c))\}$ using $H$;
7:    Check all counters $\{h_1(prf_k(c)), \dots, h_l(prf_k(c))\}$ in $VO_\omega$. If one of them is equal to
      0, $c$ is incorrect;
8: end for
```

4.4.2. Completeness Verification

To check the completeness of the query result $c(ds)_\omega$, the user first finds out the number Sum_ω which indicates the available description files from the available result set ds_ω. Then, if Sum_ω is not equal to the number of genuine description files λ_ω, the query result $c(ds)_\omega$ can be directly judged as incomplete. Otherwise, the user finds out the corresponding available encrypted description files based on ds_ω. Further, for each available encrypted description file $c \in c(ds)_\omega$, the user calculates $\{h_1(prf_k(c)), \dots, h_l(prf_k(c))\}$ with the set of l hash functions H and the corresponding counters $vo_\omega[h_1(prf_k(c))], \dots, vo_\omega[h_l(prf_k(c))]$ in VO_ω are decreased by 1. Finally, the user can confirm whether the completeness of the query result $c(ds)_\omega$ by judging whether R_ω is equal to 0, where $R_\omega = \frac{\sum_{j=0}^{m-1} vo_\omega[j]}{l}$. The process of completeness verification is shown in Algorithm 3.

Algorithm 3 Completeness verification.

```
Input: $\lambda_\omega$, $VO_\omega$, $c(ds)_\omega$, $ds_\omega$
Output: The completeness of $c(ds)_\omega$
1: $Sum_\omega \leftarrow |ds_\omega|$
2: if $Sum_\omega \neq \lambda_\omega$ then
3:    $c(ds)_\omega$ is incomplete;
4: else
5:    for each available $c \in c(ds)_\omega$ do
6:       Calculate $prf_k(c)$;
7:       Calculate $\{h_1(prf_k(c)), \dots, h_l(prf_k(c))\}$ using $H$;
8:       The corresponding counters $\{h_1(prf_k(c)), \dots, h_l(prf_k(c))\}$ in $VO_\omega$ are decreased
         by 1;
9:    end for
10:   Calculate $R_\omega = \frac{\sum_{j=0}^{m-1} vo_\omega[j]}{l}$
11:   if $R_\omega \neq 0$ then
12:      $c(ds)_\omega$ is incomplete;
13:   end if
14: end if
```

4.4.3. Consistency Verification

To check the consistency of the query result $c(ds)_\omega$, the user directly uses the hash function h_v to hash $C^*(C(DS)_{w \in W})$ sent from the fog node and further sends $h^u = h_v(C^*(C(DS)_{w \in W}))$ to the blockchain for the comparison between h^{g_o} and h^u. Since the data stored on the blockchain are obtained based on the consensus mechanisms, such as PBFT or Raft, if the comparison result shows that the hash value of h^{g_o} is equal to the hash value of h^u, the user can confirm that the received query result is consistent with the query result sent from the fog node g_o. Moreover, the fog node g_o cannot repudiate the incorrectness and incompleteness of the query result due to network problems. The process of consistency verification is shown in Algorithm 4.

Algorithm 4 Consistency verification.

```
Input: $h^{g_o}$, $h^u$, ChainGroup $STORAGE$
Output: The consistency of $c(ds)_\omega$
1: $chain \leftarrow ChainFactory.openChain(STORAGE)$;
2: $table \leftarrow Chain.EQ(chain, num)$;
3: $list[\,] \leftarrow table.Entry(time)$;
4: Store $h^{g_o}$ sent from $g_o$;
5: Store $h^u$ sent from the user;
6: if $h^{g_o}$ is equal to $h^u$ then
7:     The query result is consistent;
8: else
9:     May be something wrong with network problems;
10:    return
11: end if
```

Moreover, the scheme divides the cost the LBSP used to pay for the third-party storage into two parts, one part remains—the original storage cost—and the other part is given by the user incentivized by verifying the hash value of the data published by the fog node on the blockchain. The more times fog nodes return data honestly, the more incentive rewards they receive.

5. Security Analysis

In this section, we provide a comprehensive security analysis of the SEQRVS scheme, including the semantic security of verification objects and lookup table.

5.1. Security of Verification Objects

The purpose of the scheme containing the fog node verification object and the description file verification object proposed in this paper is the same as [9]; once the verification objects are constructed, for security's sake, they reveal nothing about the characteristics of the fog node or the contents of the description files. That is to say, both the outside eavesdroppers and the inside data 'leakers' can hardly acquire useful information from the verification objects. To reach such a goal, the proposed scheme focuses on two aspects: the meaninglessness of the verification objects themselves and the indistinguishability of the verification objects from each other.

According to Section 4.2, the fog node verification object consists of m bits with binary numbers and the description file verification objects consist of n counters with natural numbers. Due to the security of the hash function mapping process (i.e., the secrecy of the hash function), an attacker cannot obtain information related to the data itself from these sequences of numbers. Therefore, the verification objects themselves are meaningless. It is clear to see that guaranteeing the indistinguishability of verification objects means that their sizes and formats cannot be distinguished. The improved construction process mentioned in Section 4.2 set the length of the fog node verification object to m and the description file verification objects to n, solving the size problem. Inserting indistinguishable elements into the counters of the Bloom filter can further guarantee the indistinguishability of the inserted elements. In Algorithm 1, we used the pseudo-random function to guarantee the indistinguishability of the inserted elements.

Before proving the security of the verification objects, the definition of the pseudo-random function is given first.

Definition 3. *For a probabilistic polynomial time distinguisher $\mathcal{D}$ with the advantage to distinguish $F_k(x)$ from a string r of length s, where $F : \{0,1\}^* \times \{0,1\}^\tau \rightarrow \{0,1\}^S$ is a keyed function,*

$x \in \{0,1\}^*$ is a random length string and $k \in \{0,1\}^\tau$ and r are chosen uniformly and randomly from $\{0,1\}^S$. This advantage can be defined as

$$Adv_{\mathcal{D}}^{F} = |Pr[\mathcal{D}(r) = 1] - Pr[\mathcal{D}(F_k(x)) = 1]|. \tag{12}$$

Definition 3 means that no polynomial time algorithm can distinguish the output of a pseudo-random function from the output of a real random function [26]. If F is a pseudo-random function, then the advantage $Adv_{\mathcal{D}}^{F}$ is negligible under the randomly chosen key k from $\{0,1\}^\tau$.

The formal security proof is given as follows.

Theorem 1. *If prf is a pseudo-random function, then the fog node and description file verification objects are semantically secure and achieve indistinguishability in the random oracle model.*

Proof. For ($\widehat{Game1}$) and ($\widehat{Game2}$) given in Section 3.3, we define that adversary $\mathcal{A}$ has a non-negligible advantage $\epsilon(\epsilon < 1)$ to win these two games. In the meantime, a distinguisher $\mathcal{D}$ with a non-negligible advantage to distinguish the output between the pseudo-random function and the real random function can be constructed by $\mathcal{A}$.

As the function used in Algorithm 1 is prf, we introduce another algorithm Algorithm 1^*, which uses a random function $rrf : \{0,1\}^* \to \{0,1\}^S$ to replace the pseudo-random function prf. We denote Algorithm 1 as f_0 and Algorithm 1^* as f_1. In essence, the prf and rrf are modeled as $\mathcal{D}$-accessible random oracles. $\mathcal{D}$ emulates the game ($\widehat{Game1}$) and ($\widehat{Game2}$) for $\mathcal{A}$. According to whether adversary $\mathcal{A}$ succeeds in the game, $\mathcal{D}$ determines whether $x = 0$ or $x = 1$, so that it accepts the algorithm f_x, $x \in \{0,1\}$. Specifically, if $\mathcal{A}$ succeeds, then $\mathcal{D}$ determines $x = 0$; otherwise, $x = 1$.

$\mathcal{A}$ chooses two keywords w_1 and w_2 to $\mathcal{D}$. $\mathcal{D}$ uses the algorithm f_x and chooses a bit $b \in \{0,1\}$ randomly. The choice $\mathcal{D}$ made influenced whether $\mathcal{A}$ can succeed. However, the chances of $b = 0$ and $b = 1$ are uniformly $\frac{1}{2}$. Then $\mathcal{D}$ returns VO_{w_b} to $\mathcal{A}$. $\mathcal{A}$ outputs a bit b' and $\mathcal{D}$ outputs a guess x' for x. As defined above, $\mathcal{A}$ has a non-negligible advantage ϵ to succeed in games ($\widehat{Game1}$) and ($\widehat{Game2}$) (i.e., $b' = b$), we can easily conclude that $\mathcal{D}$ also has a non-negligible advantage ϵ to determine the guess $x' = x = 0$. This conclusion can also prove that $\mathcal{D}$ can distinguish the output VO_{w_b} of $f_0(w_b)$ (using the pseudo-random function prf under the key k) from $f_1(w_b)$ (using the real random function rrf) with the non-negligible advantage ϵ. Since VO_{w_b} contains $S \times l$ elements after padding, for each element c, assume that the advantage that $\mathcal{D}$ distinguishes $prf_k(c)$ from $rrf(c)$ is ϵ', we have

$$\epsilon = \prod_{S*l} \epsilon' \Rightarrow \epsilon' = \sqrt[(S\times l)]{\epsilon} \tag{13}$$

Since ϵ is non-negligible, the advantage $\sqrt[(S\times l)]{\epsilon}$ is also non-negligible. Therefore, if $\mathcal{A}$ wins games ($\widehat{Game1}$) and ($\widehat{Game2}$) with a non-negligible advantage ϵ, then $\mathcal{D}$ has the non-negligible advantage $\sqrt[(S\times l)]{\epsilon}$ to distinguish the output between the pseudo-random function prf and the real random function rrf, which contradicts Definition 3. So that the security of verification objects can be proofed. □

5.2. Security of Lookup Table

Our scheme also adopts a secure outsourced data storage and query approach by designing a one-to-n lookup table so that multiple queries to the same POI by different users or by the same user at different times correspond to different contents of the fog node storage. By doing so, we completely break the traditional one-to-one storage structure of the lookup index and lookup content. To ensure the security of the proposed scheme, we will illustrate the security of the lookup table in the following ways: the security of stored content in the lookup table, the indistinguishability of individual data items in the lookup table, the security of the data request process.

Firstly, the files themselves are secure, because LBSP encrypts the description files before outsourcing them to the fog node with the keys that attackers cannot achieve. Secondly, as shown in Table 3, the data items in the lookup table are all composed of ID, $prf_k(W)$, $C(DS)_{w \in W}$, $C(\Lambda)$, VO_G, and VO_W. Moreover, the data in the data items are all n-dimension (n = 2 in Table 3), attackers—even the fog node itself—cannot tell the difference between the different data items. Thirdly, when the same POI is queried, the description files and verification object returned by the fog node each time will likely be different data items (i.e., in different positions on the server) and that item will be unique in that query. As the generation of the guide set is random, the returned data from querying the same POI will be randomly assigned to multiple data entries in the lookup table, and the fog side cannot obtain the complete frequency information of the query index through traditional statistical analysis methods, enabling a secure query process. Moreover, queries against different POIs may return data from the same data item, which helps to confuse the one-to-one correspondence between the query index and the returned data.

6. Evaluation

To verify that our proposed query scheme for the description files in LBS is practical and feasible, we experimentally evaluated the scheme in the following ways.

6.1. Experimental Settings

Because of the large number of description files and the number of queries, we chose AES as the encryption algorithm for the description files. In addition, we used HMAC-SM3 with 256 bits key to instantiate the pseudo-random hash function $prf_k()$. For the Hash function used to validate data on the blockchain, we used SHA-256, the same as Ethereum.

In our experiments, we used Java language to implement all programs. The client-side was an Inter i7-6700HQ 2.6 GHz computer with 16 GB RAM running Windows 10. The fog node was simulated by using the Linux CentosOS 7. As for the blockchain, we used the FISCO BCOS [27] consortium chain as the backbone of the blockchain and the system environment was also CentOS7. The compiler was the WeBASE IDE and the language was Solidity 0.4.24 (0.4.24 is upward compatible up to version 0.5).

6.2. Performance of Verification Objects

As a key technique in our query solution, the time to construct the verification objects greatly affects how well the solution works. Therefore, we first conducted experiments on the construction time of two types of verification objects.

To evaluate the performances of verification objects, we chose five representative types of POIs for this experiment: $Hotel$s, $Restaurant$s, $Supermarket$s, $Gas\ station$s, and $Attraction$s. Table 4 shows the amounts of these POIs in the regions. Moreover, we set the total inserted times S to 500, which is much more than any number of POIs in our experiments. The time required to construct the verification objects are shown in Figure 6. The graph should have consisted of the constructing time of the fog node verification object and the constructing time of the description file verification object. However, due to the difference in the construction method of the verification objects (i.e., the difference in the number of numbers inserted into the verification objects), the construction time of the fog node verification object was much less than that of the description file verification object and was negligible in the construction process for both types of verification objects.

Table 4. The numbers and distribution of the POIs.

Fog Node	Hotel	Restaurant	Supermarket	Gas Station	Attraction	Total
g_1	29	41	18	7	5	100
g_2	33	32	22	2	11	100
g_3	30	38	16	4	12	100
g_4	24	38	10	10	18	100
g_5	29	36	18	6	11	100
g_6	29	31	24	11	5	100
g_7	22	43	11	8	16	100
g_8	23	42	12	5	18	100
g_9	24	41	13	10	12	100
total	243	342	144	63	108	900

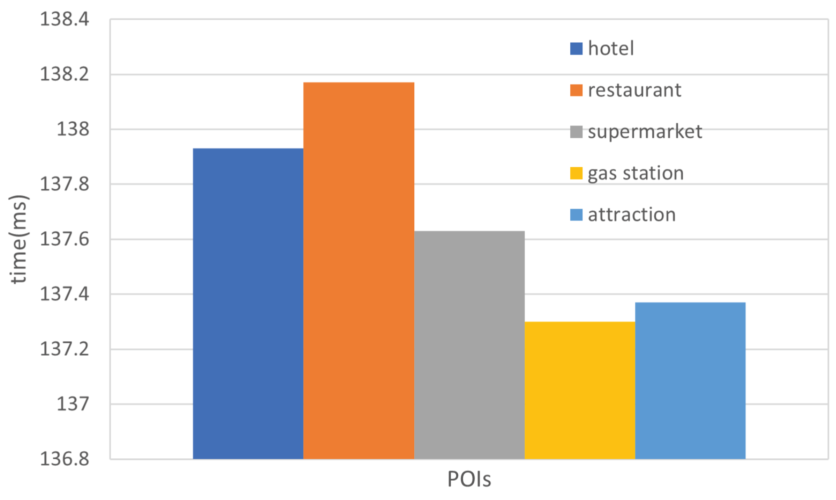


Figure 6. Time costs of constructing the verification objects.

The highest bar in the graph represents the construction time corresponding to the *Restaurant*s, which is positively correlated with the number of POIs in the area; that is, the more POIs of the same type in the area, the more time it took to construct, which is in line with the common perception. However, the differences between the different categories were not very large, especially during the whole process of location-based services. Thus, it is not a concern for an attacker to infer from the construction time of a verification object on what it represents.

For the description file verification objects, although the POI numbers were distinguishably different, the *Restaurant* numbers were five times more the *Gas station* numbers, and the time cost of *Restaurant*s was only 6.3% (138.17 and 137.30 ms) more than the *Gas station*s; hence, not very different. The reason for this result is that the verification object construction time is mainly determined by the selected pseudo-random function. Moreover, as shown in Figure 7, we compare our verification object construction scheme with [9]. It is clear that our construction time is slightly higher, also because the pseudo-random function chosen is different. Compared to the HMAC-MD5 used in [9], our pseudo-random function key grew from 128 to 256 bits, addressing the potential security concerns associated with the use of an insecure cryptographic algorithm in the former, and the increase in construction time is within acceptable limits.

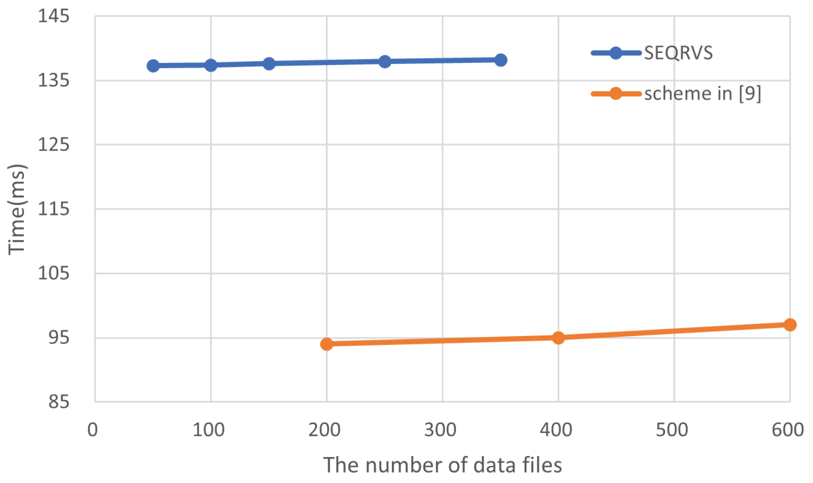


Figure 7. The comparison of time costs in constructing verification objects.

The construction time of the verification object affects the efficiency of the outsourced storage of description files; it is the verification time of the verification object that is critical for the user. Low latency is an important indicator to verify the feasibility of our solution. We made experiments on the verification time of the verification object.

Figure 8 shows the time cost increased with the number of data queried in our scheme. As our solution is similar to the data authenticity verification method in [9], we only made a comparison of the data completeness verification process. Figure 8 also shows the difference between our method and [9]. Since our solution stores the true number of description files corresponding to the verification objects in advance when constructing the verification object, we have a huge advantage in the data completeness verification process. As the grey line shows, if the data sent from the third party is incomplete, we can identify it in a short time and determine the amount of missing data, enabling the description file query solution to work more efficiently. Compared to the data query scheme in [9], the other main reason for our scheme, apart from the higher time consumption due to the introduction of pre-judgment, is a large amount of data we had to pad. The data verification objects in [9] were all padded to the data verification object corresponding to the keyword with the most data files, whereas our scheme was padded to T, which is greater than the maximum value of the former. The increased time cost was solely due to the security of the verification object. As a result, our verification time was slightly higher than [9], but still within acceptable limits.

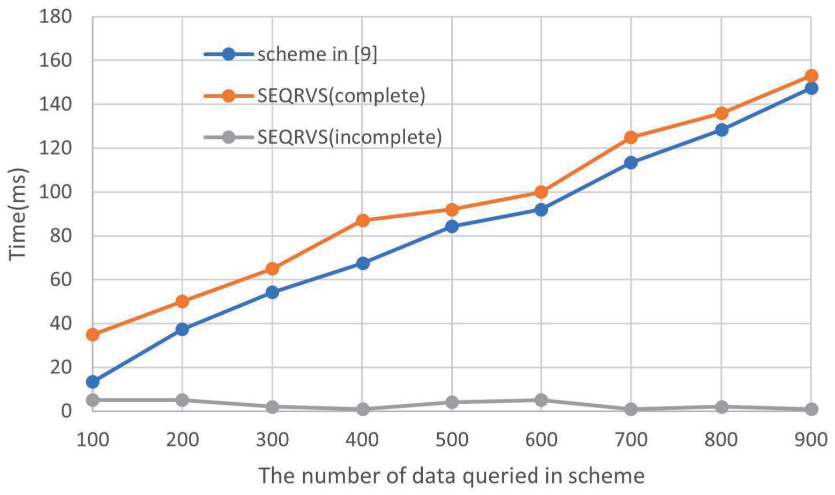


Figure 8. The comparison of time costs of data completeness verifications.

6.3. Performance of Lookup Table

To better analyze the effectiveness of our proposed scheme, we also needed to further analyze the lookup table, and the data storage structure in the fog nodes, to ensure that our scheme was practical and feasible. We conducted experiments on the query time of the lookup table. With the increase in querying data files, the query time can be seen in Figure 9.

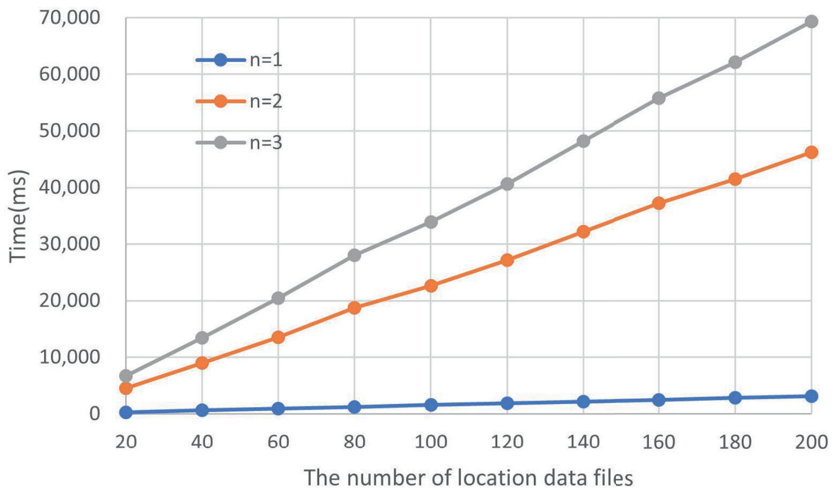


Figure 9. Query time.

The blue line in Figure 9 represents the one-to-one lookup table method, and the red and grey lines represent the lookup tables for *n* = 2 and *n* = 3, respectively. It can be seen that the search time increases with the increase of search times. For *n* = 1, because the number of data files in the table is only one, the fog node can easily determine the important priority of data according to the access query times of each data and infer the meaning of the data representation (and even the encrypted data). At the same time, compared with *n* = 2 and *n* = 3, since the number of searches is determined, once a piece of datum is found, the search process will end immediately, which leads to a significant decrease in the average search time. When *n* = 2, it takes about 46 s to query the description files 200 times and the average time for each query is 0.23 s, which is within a reasonable limit. It can also be seen from the graph that the time spent on the look-up table corresponding to *n* = 3 has increased by approximately 50% compared to *n* = 2, but there is no improvement in terms of functional safety. Therefore, the one-to-two look-up table was ultimately chosen as the optimal choice.

6.4. Performance of Blockchain Incentive

We also experimented with the incentive benefits obtained by using blockchain in the fog node as part of the verification process, as shown in Figure 10.

The test results show that—with the increase of query times—if the fog node returns the description files required by users without error, its incentive income will also increase. The fog node can reach the service fee as discussed. Our plan is feasible.

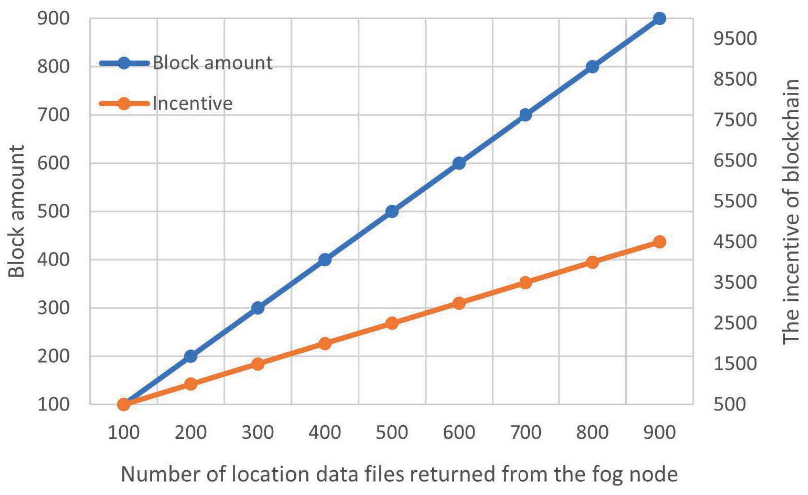


Figure 10. Blockchain incentive.

6.5. Performance of Query Scheme

To verify the feasibility of the whole scheme and to simplify the whole process of querying description files, we visualized the area responsible for each fog node as a 10 × 10 matrix area, with one POI in each small area. Considering the practical situation, we designed 3 × 3 of matrix regions, which means that there were 9 fog nodes and 900 different small regions (for simplicity, we specify that each small region has only one POI attribute). The regions are shown in Figure 11.

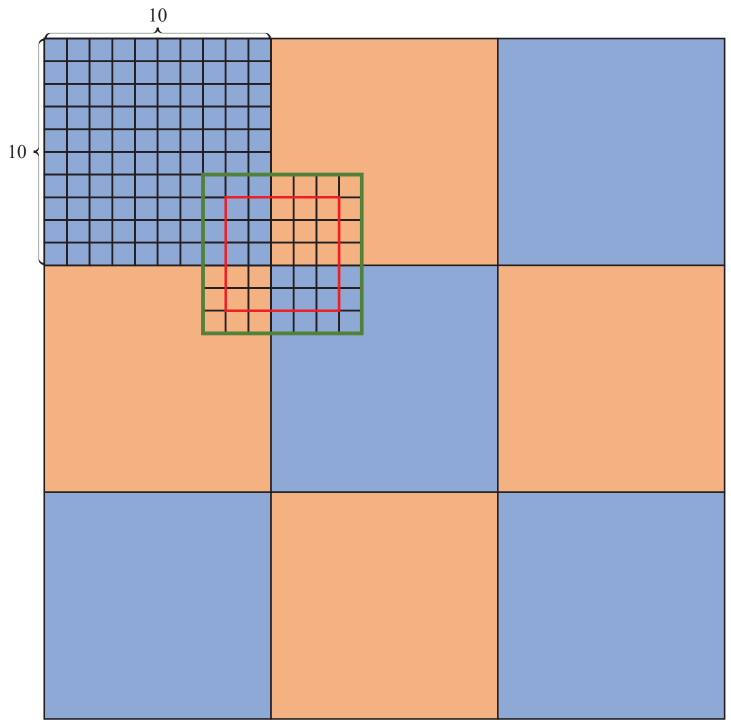
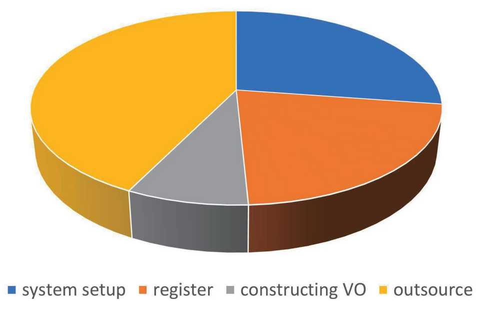


Figure 11. Region area.

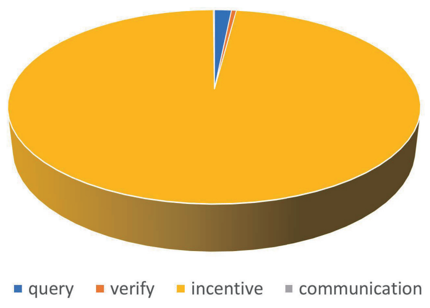
We set the range of areas that the user is allowed to query to be a rectangular area of 5 × 5, locations beyond this range are not considered part of the query result. Since we used the k-anonymity algorithm to protect the privacy of the querying user, the query area was expanded accordingly. We set k to 9, which expanded the corresponding query area to a rectangular range of 7 × 7. The region surrounded by the red line in Figure 11 was the true area of the LBS user, and the region surrounded by the green line was the expanded area.

Regarding the size of the verification objects, only when the number of counters was more than 9585, the false positive was less than 0.01 [9]. So we set the number of counters in the fog node verification object to 10,000. To verify the impact of the verification and padding regions of the description file verification objects on the description file query, we designed two sets of comparison tests, i.e., the verification regions of the counting Bloom filter of the description file verification objects were designed to be 10,000 and 20,000, and the padding regions were designed to be 2500 and 5000. As mentioned above, the size of the fog node verification object was approximately 5 KB (10,000 $\times$ 4 bits) and the sizes of the description file verification objects were approximately 6 to 12 KB (12,500 $\times$ 4 bits to 25,000 $\times$ 4 bits).

We conducted 100 experiments for each POI type data query, with each query location randomly selected. The results of the experiments are shown in Figure 12.



(**a**) Time cost of preparation works.



(**b**) Time cost of one query.

Figure 12. The time of our SEQRVS scheme.

Figure 12a shows the time proportion of the preparation works of our scheme, including the construction of verification objects and the storage process. The time is about 1828 ms (100 description files). Figure 12b shows the time proportion of the one query. For most of them, it involves the incentive time for the fog node to reach the storage rewards. It takes about 14 s, which was determined by the time the blockchain generated a block. Indeed, the time the user obtained the required description files after querying and verifying was only about 307 ms.

From the above experimental results, it can be seen that the average query time for each description file is within acceptable limits. The experiments show that the scheme proposed in this paper is practical and feasible.

7. Conclusions

In this paper, we propose a security-enhanced query result verification scheme for outsourced data in location-based services. We corrected the mistake of the work in [9] to make the whole description file verification process unobstructed without missing the advantages of being 'fine-granted' and secure. We used the one-to-n lookup table to confuse the corresponding relationship between the query index and the description files. Moreover, we solved the current problem of fog nodes exploiting vulnerabilities to deny dishonest behavior and optimize the current cost structure for third-party storage through blockchain technology. Performance and accuracy experiments demonstrate the validity and efficiency of our proposed scheme.

However, there are still some defects in our scheme. Compared with the previous works, our work has a certain improvement in security, but due to the complexity of using algorithms and technologies, the time consumption has increased slightly (still within a reasonable range). In addition, this scheme only considers the possible privacy disclosure caused by the fog node as a third-party storage object, but does not consider that LBSP may also disclose the user's private information or analyze user queries to obtain improper benefits.

In the future, one research direction will be how to protect the user's privacy under the consideration of taking LBSP as a dishonest entity. Moreover, the advantages of blockchain technology have not been fully brought into play, so another research direction will be to further use its characteristics to improve the security and feasibility of the scheme.

Author Contributions: Conceptualization, G.Y.; data curation, H.Z.; formal analysis, Y.H.; methodology, J.L.; software, C.L.; writing—original draft, J.L. and K.X.; writing—review and editing, G.Y. and Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially supported in part by the National Key R&D Program of China under grant 2020YFB1805403, the R&D Program of Beijing Municipal Education Commission under grant KM202010009010, the Yunnan Key Laboratory of Blockchain Application Technology (202105AG070005), the Beijing Municipal Natural Science Foundation under grant M21029, the National Key Research and Development Program of China under grant 2018YFB1800302, the Foundation of Guizhou Provincial Key Laboratory of Public Big Data under grant 2018BDKFJJ021, and the Talent special project (XN083).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the finding of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare that there are no conflict of interest regarding the publication of this article.

References

1. Adem, B.A.; Alrashdan, M.; Abdulnabi, M.; Jaradat, A.; Tubishat, M.; Ghanem, W.A.; Yusof, Y. A General Review on Location Based Services (LBS) Privacy Protection Using Centralized and Decentralized Approaches with Potential of Having a Hybrid Approach. *Int. J. Future Gener. Commun. Netw.* **2021**, *14*, 3057–3079.
2. A Almusaylim, Z.; Jhanjhi, N. Comprehensive review: Privacy protection of user in location-aware services of mobile cloud computing. *Wirel. Pers. Commun.* **2020**, *111*, 541–564. [CrossRef]
3. Yang, G.; He, Y.; Xiao, K.; Tang, Q.; Xin, Y.; Zhu, H. Privacy-Preserving Query Scheme (PPQS) for Location-Based Services in Outsourced Cloud. *Secur. Commun. Netw.* **2022**, *2022*, 9360899. [CrossRef]
4. Li, D.; Wu, J.; Le, J.; Liao, X.; Xiang, T. A novel privacy-preserving location-based services search scheme in outsourced cloud. *IEEE Trans. Cloud Comput.* **2021**. [CrossRef]
5. Huang, H.; Gartner, G.; Krisp, J.M.; Raubal, M.; Van de Weghe, N. Location based services: ongoing evolution and research agenda. *J. Locat. Based Serv.* **2018**, *12*, 63–93. [CrossRef]
6. Yang, P.; Xiong, N.; Ren, J. Data security and privacy protection for cloud storage: A survey. *IEEE Access* **2020**, *8*, 131723–131740. [CrossRef]
7. Chen, L.; Lee, W.K.; Chang, C.C.; Choo, K.K.R.; Zhang, N. Blockchain based searchable encryption for electronic health record sharing. *Future Gener. Comput. Syst.* **2019**, *95*, 420–429. [CrossRef]
8. Wang, Y.; Sun, S.F.; Wang, J.; Liu, J.K.; Chen, X. Achieving searchable encryption scheme with search pattern hidden. *IEEE Trans. Serv. Comput.* **2020**, *15*, 1012–1025. [CrossRef]
9. Yin, H.; Qin, Z.; Zhang, J.; Ou, L.; Li, K. Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data. *IEEE Trans. Cloud Comput.* **2017**, *9*, 27–39. [CrossRef]
10. Wu, Z.; Wang, R.; Li, Q.; Lian, X.; Xu, G.; Chen, E.; Liu, X. A location privacy-preserving system based on query range cover-up or location-based services. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5244–5254. [CrossRef]
11. Liu, Z.; Wu, L.; Meng, W.; Wang, H.; Wang, W. Accurate Range Query With Privacy Preservation for Outsourced Location-Based Service in IoT. *IEEE Internet Things J.* **2021**, *8*, 14322–14337. [CrossRef]
12. Yadav, V.K.; Verma, S.; Venkatesan, S. Efficient and secure location-based services scheme in VANET. *IEEE Trans. Veh. Technol.* **2020**, *69*, 13567–13578. [CrossRef]
13. Huang, Q.; Du, J.; Yan, G.; Yang, Y.; Wei, Q. Privacy-Preserving Spatio-Temporal Keyword Search for Outsourced Location-Based Services. *IEEE Trans. Serv. Comput.* **2021**, *99*, 1. [CrossRef]
14. Wang, X.; Ma, J.; Miao, Y.; Liu, X.; Zhu, D.; Deng, R.H. Fast and Secure Location-based Services in Smart Cities on Outsourced Data. *IEEE Internet Things J.* **2021**, *8*, 17639–17654. [CrossRef]
15. Zhang, H.; Guo, Z.; Zhao, S.; Wen, Q. Privacy-preserving linear region search service. *IEEE Trans. Serv. Comput.* **2017**, *14*, 207–221. [CrossRef]
16. Guo, J.; Sun, J. Secure and Practical Group Nearest Neighbor Query for Location-Based Services in Cloud Computing. *Secur. Commun. Netw.* **2021**, *2021*, 5686506. [CrossRef]

17. Manju, A.; Subramanian, S. Fog-Assisted Privacy Preservation Scheme for Location-Based Services Based on Trust Relationship. *Int. J. Grid High Perform. Comput.* **2020**, *12*, 48–62. [CrossRef]
18. Zhu, X.; Ayday, E.; Vitenberg, R. A privacy-preserving framework for outsourcing location-based services to the cloud. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 384–399. [CrossRef]
19. Zhou, J.; Cao, Z.; Qin, Z.; Dong, X.; Ren, K. LPPA: Lightweight privacy-preserving authentication from efficient multi-key secure outsourced computation for location-based services in VANETs. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 420–434. [CrossRef]
20. Benarous, L.; Kadri, B. A novel privacy preserving scheme for cloud-enabled internet of vehicles users. In *Security, Privacy and Trust in the IoT Environment*; Springer: New York, NY, USA, 2019; pp. 227–254.
21. Liao, D.; Sun, G.; Li, H.; Yu, H.; Chang, V. The framework and algorithm for preserving user trajectory while using location-based services in IoT-cloud systems. *Clust. Comput.* **2017**, *20*, 2283–2297. [CrossRef]
22. Kuang, L.; Wang, Y.; Ma, P.; Yu, L.; Li, C.; Huang, L.; Zhu, M. An improved privacy-preserving framework for location-based services based on double cloaking regions with supplementary information constraints. *Secur. Commun. Netw.* **2017**, *2017*, 7495974. [CrossRef]
23. Wan, S.; Zhao, Y.; Wang, T.; Gu, Z.; Abbasi, Q.H.; Choo, K.K.R. Multi-dimensional data indexing and range query processing via Voronoi diagram for internet of things. *Future Gener. Comput. Syst.* **2019**, *91*, 382–391. [CrossRef]
24. Yang, C.; Tao, X.; Zhao, F.; Wang, Y. Secure data transfer and deletion from counting bloom filter in cloud computing. *Chin. J. Electron.* **2020**, *29*, 273–280. [CrossRef]
25. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: New York, NY, USA, 1999, pp. 223–238.
26. Bellare, M.; Rogaway, P. *Introduction to Modern Cryptography*; UCSD CSE: La Jolla, CA, USA, 2005; Volume 207, p. 207.
27. FISCO BCOS Blockchain. Available online: https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/introduction.html (accessed on 15 July 2022).