Journal of Network Intelligence ©2023 ISSN 2414-8105 (Online)
Taiwan Ubiquitous Information Volume **8**, Number **2**, May **2023**

Secure Location Data Sharing Scheme with KN-tree in Mobile Online Social Network(SMOSNs)

Hong-Yang Liu
School of Computer Science, Minnan Normal University
Zhangzhou, 363000, China
975556702@qq.com

Hui Huang*
School of Computer Science, Minnan Normal University
Fujian Key Laboratory of Granular Computing and Application, Minnan Normal University
Zhangzhou, 363000, China
hhui323@163.com

Qun-Shan Chen
School of Computer Science, Minnan Normal University
Zhangzhou, 363000, China
xiamensam@163.com

Zhen-Jie Huang
Fujian Key Laboratory of Granular Computing and Application, Minnan Normal University
Zhangzhou, 363000, China
zjhuang@mnnu.edu.cn

Chen-Huang Wu
Key Laboratory of Applied Mathematics of Fujian Province University, Putian University
Putian, 351100, China
ptuwch@163.com

*Corresponding author: Hui Huang

Received November 22, 2022, revised January 5, 2023, accepted March 26, 2023.

Abstract. *Location data is a critical foundation for online social networks, and sharing location data is essential to improve the efficiency of online social services. However, location data often contains users' sensitive information. How to protect users' privacy is a crucial problem in the process of location data sharing. Many solutions have been proposed to preserve user privacy, such as deploying trusted base stations or using encryption. But the problem of privacy disclosure still exists. In particular, the existing schemes do not consider that malicious third-party nodes can obtain the user's location data through rainbow attacks. The location server can guess the social relationship of users through their frequent location requests. In this paper, we propose a brand-new scheme to ensure safety in the entire process. Salt encryption has been adopted to ensure data security in the location recording phase, and Bloom filter to ensure privacy security in the data sharing phase. Moreover, we proposed a KN-tree structure to simplify the computational process of location updates. Finally, the theoretical analysis shows that our scheme is superior to the existing methods in terms of security and practicality. We also realize and analyze the efficiency of our proposed scheme. And using the KN-tree structure to save users' location data can reduce the computing cost of the location server by about 27%.*

Keywords: Location sharing, Privacy protection, Bloom filter, KN-tree

1. **Introduction.**

With the rapid development of mobile internet [1] and mobile devices [2]. Mobile online social network [3](MOSNs) has becomes a major information sharing channel for people to share information. Such as QQ, Wechat, and Facebook [4]. MOSNs are not affected by time and space, and people can access the Internet through Wifi [5] or cellular network [6]. At the same time, people can communicate and share location information with family and friends through online social networks [7] no matter where they are.

Location data sharing has brought great convenience to people's lives. However, in MOSNs, data sharing faces two crucial challenges. First, the location data stored centrally cannot be verified. Second, data providers are getting increasingly concerned about data security and privacy issues. According to the user's location information and social relationship information, the external adversary can infer the user's home address, work location [8], physical condition, family members and other private information. Because the current MOSNs are centralized, once a malicious adversary [9] controls the server, the user's privacy will be destroyed. If the user's privacy is not securely protected, the user will be hesitant to share their location. Therefore, how to protect the privacy of users based on providing location services is a crucial issue that researchers need to consider at present.

In [10], the author adopts the order preserving encryption method to solve the problem of comparing the size of user's data without disclosing the user's actual coordinates. However, there are still limitations in the scope of use, which cannot adapt to fine-grained access scenarios. In [11], the authors exploite the K-anonymous scheme, which can solve the problem that users hide their identities in the process of sharing location data, thus making it difficult for external adversaries to crack the user's identity information. In [12], The author solves the problem of user identity hiding. The author replaces the user's real name with a pseudonym to avoid the user's identity privacy being cracked by an external adversary. At the same time, the location server can verify the existence of the pseudonym, but the adversary cannot distinguish the pseudonym.

Blockchain as a promising technology to provide distributed secure solutions [13]. It can solve the problems caused by MOSNs centralized storage. With the features such as tamper-proof, anonymity, and traceability, blockchain has attracted significant attention for enhancing security in areas such as Internet of Vehicles [14], Internet of Things [15], MOSNs [2]. Many works leveraging blockchain to share location data. For example, in [16], the author saves the credentials of location data through two transactions. Then, users can verify the integrity of the received location data. In [17], the author designs an APP architecture based on blockchain. Patients can share data without exposing their privacy. However, introducing blockchain technology into location data sharing brings other new challenges.

To further improve system security and provide fine-grained access for users. In [18], the author uses the Bloom filter to hide the user's social network. Then, in [19], the author proposes a multi-layer location data-sharing scheme to fine-grain the user's access strategy further. By setting the user's location data as a rectangular area, the leakage of the user's accurate location coordinates is avoided. To solve the problem of computational cost and easy-to-crack rectangular areas, Wang et al. [20] proposed an improved multi-layer location data-sharing scheme, which greatly reduced the computational cost of users by using offset coordinates. At the same time, a third-party node is introduced to help users verify data. However, the efficiency problem of location sharing in distributed scenarios remains unsolved.

Based on the above analysis, although these methods can meet security characteristics, they only realize the security of part of the process. The security of each stage in the whole sharing process is not fully considered. Namely, the security of the entire process cannot be realized. At the same time, the Merkle tree is used to verify the location data. When the location data is frequently updated, it has high computational and storage costs. At the same time, the user's location data are all hashed and stored in the blockchain. The adversary can crack the user's location data through violent attacks. Although the K-anonymity algorithm can solve the privacy problem of users, it will bring large computational overhead in the process of location query. In this paper, we propose a new whole-process privacy protection scheme. First, we use salt encryption to improve the security of user location data. Avoid using violent attacks to crack the user's location data when the adversary obtains the location information proof in the blockchain. Next, we use the method of K-anonymity and a fake identity to hide the user's location data in $k-1$ fake locations to protect the user's data security. We use fake identities in the data-sharing phase, making it impossible for the location server to link the user's real identity with location data. Thereby preventing curious location servers from inferring some helpful information. The main contributions of this paper can be summarized as follows.

- We adopt salt encryption to resist violent attacks from external adversaries. The hash value of the location information stored in the blockchain will not be cracked. Meanwhile, we use Bloom filter to reduce the computational overhead of the server.
- We propose a new Merkle tree structure, namely KN-tree. Each of its leaf nodes comprises the user's k anonymous location data. It dramatically reduces the computational cost of our Merkle tree generation, and in the dynamic update scenario, it has a smaller computational cost.
- We provide users with fine-grained access policies. During location sharing, we have considered two scenarios. First, when asking for a friend's location, if the requester fulfills the friend's requirements, requester can get the user's exact location. Second, suppose the requester met the requirements when asking about the stranger's location. we return the location coordinates near the stranger's actual location to protect the stranger's privacy under location sharing.

2. **Related Work.**

The traditional location data sharing scheme [21] saves the location data in the clouds, which can reduce the storage cost for users but also makes the data difficult to verify. Blockchain [22] is a decentralized, peer-to-peer network communication system that is traceable and tamper-proof. In 2008, Satoshi Nakamoto proposed a blockchain-based peer-to-peer electronic transaction system [23]. The introduction of blockchain can solve the double-spending problem. However, with the deepening of research, blockchain application scenarios are becoming more extensive. For example, in a data-sharing scenario [24], data credentials can be stored in the blockchain to achieve safe and efficient data sharing.

Guy et al. [16] proposed a blockchain-based personal data storage scheme. Two types of transactions are employed in their scheme. Namely, T_{access} and T_{data}. The data is stored off-chain, and the hash value of the data is stored on-chain. An automatic access controller records these two types of transactions on the blockchain. Cao et al. [17] proposed a blockchain-based APP architecture. Patients can securely share data without revealing their privacy.

In 2016, Nan et al. [18] proposed an efficient privacy-preserving location sharing scheme. In their scheme, a Bloom filter is used to strengthen the protection of the user's social relationship network. The location server cannot guess the user's social relationship through the user's location request. In 2018, Ji et al. [19] proposed a blockchain-based multi-layer privacy protection location sharing scheme. In their scheme, multi-level location sharing can be achieved by dividing geographic locations into multiple grids. However, their scheme has a significant security risk. Because the user's location coordinates are integers. At the same time, the hash value of the user's location information are saved in the blockchain. Then, for external adversary, they can obtain the accurate location of users through brute-force attack, thus causing the disclosure of the user's privacy.

In the previous scheme, to prevent the location server from knowing the accurate location information of the user, many schemes adopt the K-anonymity method [11]. Randomly generate $k-1$ location information and send these $k-1$ locations to the location server. Then the location server cannot distinguish which location is the accurate location of the user. However, there is a big problem with this scheme. The social network server randomly generates these $k-1$ locations. This location information must have a large gap from the accurate location. For example, the user's accurate location is in the south, but the virtual $k-1$ locations maybe the North Pole. To solve this problem and improve the credibility of users' false location. In 2019, Chen et al. [25] proposed a location privacy protection scheme under the mobile social network. In their scheme, they improved the way to generate k virtual locations. The virtual locations are no longer generated by the online social server but are composed of the user's historical location. However, this scheme will create new problems, which increase the network server's storage overhead.

In terms of user authentication. Wu et al. [26] proposed a multi-server key exchange protocol. Their scheme can guarantee perfect forward secrecy (PFS) and avoid privileged insider (PI) attacks. It improves the system's security and realizes a higher security standard. Yang et al. [27] proposed a key management scheme based on the multiserver architecture of the client-server mode in 5G networks, which adopted elliptic curves and bilinear pairs for encryption to improve the system's security. Wu et al. [28] proposed a UAV authentication protocol based on 5G network, which effectively solved the problem of key leakage in third-party authentication and improved the system's security. To make the authentication process more transparent. Mei et al. [29] proposed a privacy protection authentication scheme based on blockchain. It can perform batch verification of data and simplify key management. Chen et al. [30] designed an improved scheme VDERSc to achieve forward privacy. It effectively solved the problem of key leakage during the insertion process. It greatly improves the security of the system.

In terms of data integrity verification. Zhu et al. [31] proposed an improved Merkle tree structure. In the convolution Merkle tree scheme, the storage and computational overhead are significantly reduced with the same data input. OU et al. [32] proposed an online social network to protect location privacy. Their scheme adopts a single location server instead of setting up multiple location servers, reducing computational and communication overhead. Secondly, the scheme uses symmetric encryption instead of broadcast encryption. Finally, location sharing can be achieved without setting up cellular towers, significantly reducing communication overhead. However, their scheme still has the problem of single point of failure. At the same time, the locations of all users are under centralized management, which may cause data leakage when an external adversary controls the location server. Finally, when the location requester receives the location data, they cannot verify the authenticity of the data. Wang et al. [20] proposed a new blockchain-based multi-layer location data sharing scheme, which can meet users' different needs. However, there are still some problems with their scheme. In the location update phase, their scheme will frequently generate new encryption keys. When the number of users is large and updated frequently, key distribution is a big problem.

3. **Preliminaries.**

3.1. **Notations.** As shown in Table 1, we list the main symbols used in this scheme and give the corresponding meanings of the symbols.

TABLE 1. The notations and descriptions.

Notations	Descriptions
ID_A	Represent the identity of user A
Pub_l	Public key of the location server
Pub_s	Public key of online social network server
ds	The distance threshold set by the user that strangers can access
df	The distance threshold set by the user that friends can access
BF_{req}	The value of Bloom filter set by SNS
FID_A	The fake ID corresponding to user A
K_{ID_i}	Symmetric key shared between users and friends
K_{str}	Symmetric key shared between users and strangers
δ	Salt value of salt encryption

3.2. **K-anonymity.** To solve the link attack and prevent the adversary from directly obtaining user data, we usually adopt the method of data generalization. It is equivalent to an expansion of the dimension of personal information. The user's real data is hidden in some indistinguishable data items to form a data set of size k. We call this method of protecting user privacy K-anonymity, namely, obfuscating the real location/identity of the user by establishing a hidden area covering k anonymous records, which makes it impossible for an adversary to distinguish which is the real one. For example, Location-based services(LBS) have been rapidly developed and widely used in recent years. LBS has brought great convenience to people's lives, but on the other hand, it also poses a threat to individual privacy. Therefore, how to provide users with convenient location services and protect the privacy of individuals has become a research hotspot in the field of LBS. As an effective LBS privacy protection method, location K-anonymity has received more and more attention. When a user sends an LBS request, its location information is required. It is indistinguishable from the location information of other at least $k-1$ users. The adversary cannot determine the user's identity who issued the request, thus protecting individual privacy.

3.3. **KN-tree.** The traditional scheme uses the Merkle tree for integrity verification. We propose an enhanced Merkle tree structure, KN-tree, in the dynamic update scenario. The essence of the KN-tree is a hash binary tree composed of a series of leaf nodes. This leaf node is the hash value of the user's k location data. Finally, the server combines the leaf nodes generated by N different users into a Merkle tree. Each leaf node of the KN-tree stores two elements. One is the user's id ID_A, and the other is the hash value. This leaf node's hash value is composed of a user's k location data, and different leaf nodes represent the hash value of the k location data of different users. Compared with traditional MHT, our KN-tree has the following advantages. (1) KN-tree significantly reduces the computational overhead of simultaneous updates by multi-user. Assuming that the update cost is t when N users are updating simultaneously, the computational cost of MHT is $t*N$, and the computational cost of KN-tree is t. (2) The number of times the user uses the hash function is also reduced. For example, if a user has four locations, then 7 hashes are required to calculate the root node, but KN-tree only need to perform 5 hash operations. (3) KN-tree can save communication overhead and storage overhead. In the traditional MHT, eight transactions need to be sent if eight users wants to store the root node of the location data in the blockchain. When the KN-tree structure is adopted, it only needs to send one transaction, and the eight users' verification can be done through the same Merkle root.

3.4. **Bloom filter.** Bloom proposed the Bloom filter in 1970. It's a long binary vector and a series of random mapping functions. Bloom filter can be used to retrieve whether an element is in a collection. Its advantage is that the space efficiency and query time are much better than the general algorithm. However, the disadvantage is that it has a false recognition rate and is difficult to delete. In our scheme, we use the false-positive rate feature to improve the system's security. We use Bloom filter to hide the user's friend relationship. Due to the misjudgment rate, the location server misjudges the stranger as the user's friend. Then, the location server cannot distinguish the user's friend. Thus, the security of the system is improved. The Bloom filter can be understood as a set collection. We can add elements to it by adding and using contains to determine whether an element is included, as shown in the Figure 3.

3.5. **Salt encryption.** For an encryption algorithm, the same input will get the same output. In this way, when the user performs authentication, the same hash encryption algorithm is applied to the plaintext password input by the user to obtain a hash value. Then the hash value is used to compare the previously-stored ciphertext value. If the two values are the same, The password authentication succeeds. Otherwise, the password authentication fails. Due to the collision characteristics of the hash function, when the background database is attacked and the hash value of the user password is obtained, the user password can still be cracked by a particular method (such as a rainbow table attack). Therefore, for security reasons, even if the two users enter the same password, they should be saved as different ciphertexts. Even if the user enters a weak password, it needs to be enhanced to increase the difficulty of breaking the password. Therefore, salt encryption was generated. In short, the original $H(p)$ has converted into $H(p+salt)$, equivalent to the hash function H change. The salt used in each hash calculation is random. The existing rainbow table will be completely unusable if H is changed. They must be regenerated for a specific H, which increases the difficulty of cracking.

4. **Problem Formulations.**

4.1. **System Model.** The system model is shown in Figure 1. Our scheme consists of four entities: user, social network server(SNS), location server(LS), and blockchain. The social network server stores the user's social relationship and identity. The location server stores the user's location information. Blockchain stores the hash value of the user's location information and the Merkle root. Our scheme will store users' location data and social relationships separately. It can improve the security of the system. Meanwhile, the social network server cannot obtain the user's location information, and the location server cannot obtain the user's social relationship.

Users: They are both providers and requestors of location data. Each user can register in SNS through the intelligent terminal device with embedded GPS to obtain the corresponding ID. Meanwhile, Users can also get various location sharing services.

SNS: SNS stores the ciphertext of the user's real identity ID, fake identity FID and a series of location information in a table. At the same time, SNS is also responsible for storing the user's relational network and generating Bloom filter.

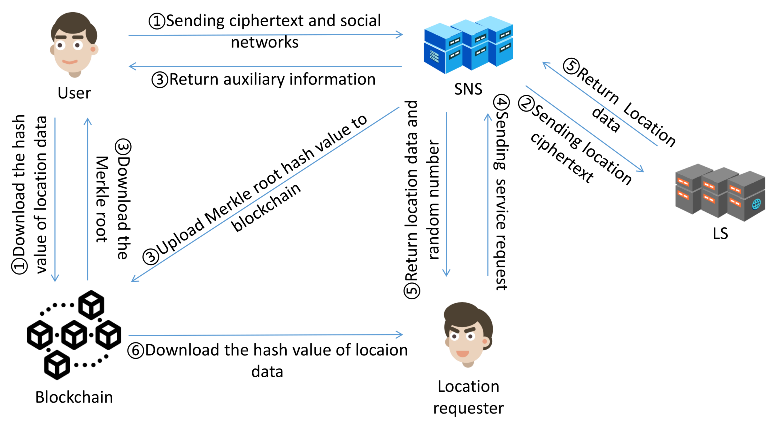
①Sending ciphertext and social networks
③Return auxiliary information
User
SNS
⑤Return Location data
②Sending location ciphertext
LS
①Download the hash value of location data
③Download the Merkle root
③Upload Merkle root hash value to blockchain
⑤Return location data and random number
④Sending service request
Blockchain
⑥Download the hash value of locaion data
Location requester

Figure 1. The system model of the proposed scheme.

LS: LS is responsible for storing the user's location data, retrieving the user's location information, and providing location services.

Blockchain: Blockchain is responsible for storing information credentials of user location data and providing credible proof in the verification phase.

4.2. **Treat model.** In this part, we mainly analyze the security threat of the system from two aspects: internal adversaries and external adversaries.

In our threat model, SNS is honest but curious. SNS will store users' social network information and forwards encrypted location data. Because the hash value of the data is stored in the blockchain, SNS cannot tamper with the user's location data. But SNS is curious about the location data in the server and judges which users are in the same location according to the ciphertext.

Location server is honest and curious. It has certain computing resources and can complete the calculation of location threshold value. But it is curious about users' friends' social networks. Use the information sent by LS to guess which data in the database is the user's actual location data. However, to protect information assets, LS will not actively collude with SNS to compromise users' privacy.

In theory, external adversaries do not collude with LS and SNS. It has strong computing power and can monitor users' transactions. And adversaries can download the location data receipt from the blockchain and then crack the user's location data through violent attacks. Therefore, we consider the violent attacks of external adversaries as the main external attacks.

4.3. **Design goals.** The goals of our proposed SMOSNs scheme are described as follows:

- **Privacy.** Although SNS can obtain the ciphertext of the user's location data and LS can obtain the value of the Bloom filter, SNS and LS cannot guess the user's actual location data and social network. In addition, LS inevitably knows some social relationship information of users, but we try to reduce such information leakage as practically as possible.
- **Resist brute-force attacks.** Suppose the external adversary has some background knowledge of plaintext space. At the same time, the data hash values of all users are saved in the blockchain. External adversaries can obtain the data hash value of any

user. However, it cannot obtain the plaintext corresponding to specific ciphertext through the violent attack.

5. **The Proposed Scheme.**

5.1. **Initialization.**

The scheme SMOSNs consists of four phases: Initialization, Location record, Location sharing, and Location verification and Update.

As shown in Algorithm 1, user A generates location data set C and then preprocesses the location data set to obtain the hash value after salt encryption. Finally, user A generates information sets Meg_{ID_A} and $Record_{ID_A}$ and sends them to SNS and the miner node of the blockchain, respectively. We describe the above process in detail below.

(1) The user ID_A first converts the geographic coordinate of its actual location into coordinates (X_0, Y_0) in the Cartesian coordinate system. Meanwhile, generates $k-1$ virtual position coordinate (X_1, Y_1), (X_2, Y_2), (X_3, Y_3),..., (X_{k-1}, Y_{k-1}), these $k-1$ coordinates are near the user's actual location. It is avoided that SNS generates some unrealistic position coordinates. The similarity of the $k-1$ location information is too low. For example, when the user is in the south, the virtual locations generated by SNS are all near the North Pole. Therefore LS can speculate the actual location information of the user. Then, ID_A defines access control policy df$_{ID_A}$ and ds$_{ID_A}$, which df$_{ID_A}$ is the distance threshold that needs to be satisfied when user A's friend wants to access its location. ds$_{ID_A}$ is the distance threshold that needs to be satisfied when a stranger wants to obtain user A's location.

(2) SNS generates public-private key pairs (pub$_s$, pri$_s$), and generates social relationship network G$_A$ of users.

(3) LS generates public-private key pairs (pub$_l$, pri$_l$).

Algorithm 1: Generation of location record

```
Input:
User $ID_A$'s location set $C = \{(X_0, Y_0), (X_1, Y_1), \ldots, (X_{k-1}, Y_{k-1})\}$
Output:
Location data $Record_{ID_A}, Meg_{ID_A}$;
1: $ID_A$ executes:
2: for $i = 0$; $i < k$; $i++$ do
3:     $LocData_i \leftarrow Enc\,(Pub_l, (X_i, Y_i))$;
4: end for
5: $secret_{ID_A} \leftarrow Enc\,(Pub_s, \delta)$;
6:$LocH_{ID_A} \leftarrow Hash\,(X_0||Y_0||\delta)$;
7: $C_{ID_A} \leftarrow Enc_{Pub_s}\,(ID_A, ts)$
8: $Record_{ID_A} \leftarrow PK_{ID_A}||LocH_{ID_A}||Timestamp||Sign_{ID_A}$;
9: $Meg_{ID_A} \leftarrow C_{ID_A}||secret_{ID_A}||LocData_0, ..., LocData_{k-1}||Sign_{ID_A}||ds||df$;
10: $ID_A$ sends $Record_{ID_A}$ to the Blockchain
11: $ID_A$ sends $Meg_{ID_A}$ to the SNS
```

5.2. **Location record.**

In the location recording phase, ID_A uses his private key to generate the signature Sig$_{ID_A}$=Sig$_{Pri_u}$(ID_A,ts), where ts represents the timestamp. Meanwhile, generates the confidential authentication information C_{ID_A}=Enc$_{Pub_s}$(ID_A,ts). Then encrypts location informations (X_0, Y_0), (X_1, Y_1), (X_2, Y_2),..., (X_{k-1}, Y_{k-1}) with the public key of LS. Get n encrypted location data LocData$_i$ (LocData$_i$=Enc(Pub$_l$,(X_i,Y_i))). ID_A calculates secret$_{ID_A}$=Enc(Pub$_s$,δ), LocH$_{ID_A}$=Hash(X_0||Y_0||δ), δ represents a random number. Finally, Record$_{ID_A}$ $\leftarrow$ PK$_{ID_A}$||LocH$_{ID_A}$||Timestamp||Sig$_{ID_A}$ is sent to the

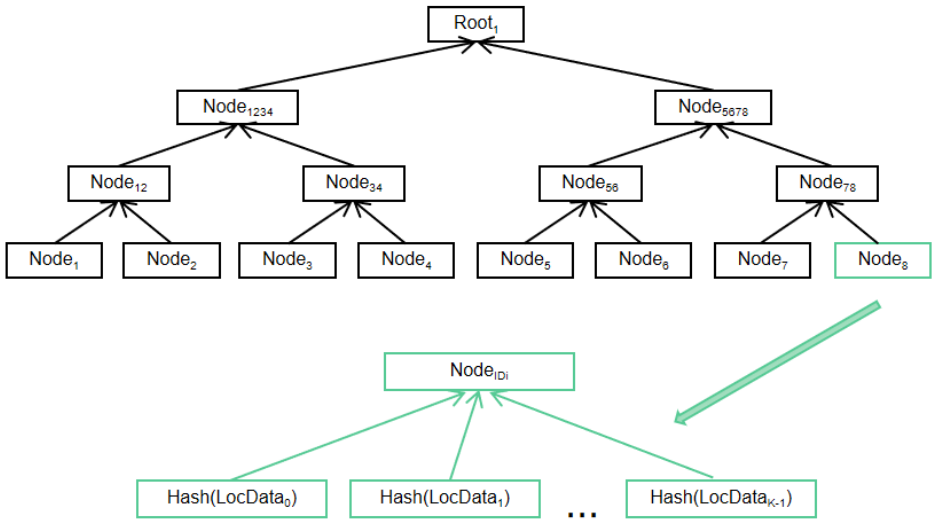

FIGURE 2. KN-tree.

TABLE 2. User and friend information.

User information					User's friend information	
User ID	User Pse—ID	ds_{ID}	df_{ID}	δ_{ID}	Friend ID	Friend Pse-ID
ID_1	PID_1	ds_{ID_1}	df_{ID_1}	δ_{ID_1}	$ID_{u,1}$	$\text{PID}_{u,1}$
ID_2	PID_2	ds_{ID_2}	df_{ID_2}	δ_{ID_2}	$ID_{u,2}$	$\text{PID}_{u,2}$
...	...	...	...	...	...	...
ID_n	PID_n	ds_{ID_n}	df_{ID_n}	δ_{ID_n}	$ID_{u,p}$	$\text{PID}_{u,p}$

blockchain, and then $Meg_{ID_A} \leftarrow (C_{ID_A}$,secret$_{ID_A}$,Sig$_{ID_A}$,LocData$_0$,...,LocData$_{k-1}$,ds,df) is sent to SNS.

When SNS receives data from different users, it first decrypts C_{ID_i} to get (ID_i,ts), then verifies whether the signature Sig$_{ID_i}$ is correct. If the signature is correct, SNS randomly generates k false identities FID$_i$, and stores (ID_i,FID$_i$,df$_i$,ds$_i$,PK$_i$) in the local server. As the table 2 shows. Finally, calculates Node$_{ID_i}$=Hash(Hash(LocData$_0$)||Hash(LocData$_1$) ||...||Hash(LocData$_{K-1}$)).

SNS uses the Node$_i$ generated by different users as the leaf nodes of the Merkle tree to create a KN-Tree(Figure 2). Meanwhile, the root of the KN-Tree tree is saved in the blockchain. Finally, SNS sends (FID$_i$,LocData$_i$,ds,df) to LS. When the LS receives the data sent by the SNS, it stores the decrypted (FID$_i$,(X_i,Y_i),ds,df) in local server.

5.3. **Location sharing.** At this phase, data sharing is mainly divided into two situations. The first case is that the user's friend i sends a location-sharing request, i encrypts the location data and symmetric key, and then sends the encrypted location data to SNS. Then, SNS will find the threshold value that meets the conditions and the friend's FID, insert the FID into the Bloom filter, and send it to LS. When LS receives the position data request, it calculates the position of i and the user. If the obtained value is less than the threshold value, LS will return the qualified FID to SNS. SNS returns the ID corresponding to FID to i. The second case is that a stranger sends a location request. In this process, SNS only needs to send the threshold value to LS, and then LS calculates based on the received threshold value and the stranger's location data, finds the qualified FID and returns it to SNS, and finally, SNS converts the FID into an ID and sends it to the applicant. We describe the above process in detail below.

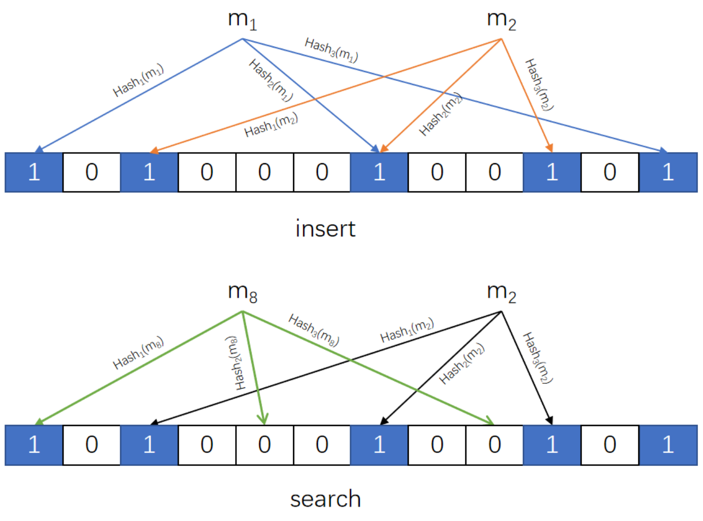

FIGURE 3. Bloom filter.

Case 1: Friend inquiry phase

The description of the location sharing phase is shown in Algorithm 2. In this phase, the location requester can apply for location sharing with his friends. Firstly, the requester generates a one-time symmetric key K_{ID_i}, then uses the public key of LS to encrypt location information (x, y) and K_{ID_i} to get the ciphertext C, C=Enc$_{Pub_l}(x, y, K_{ID_i})$. Meanwhile, the requester sends the query request (ID_i, F, dis) to the SNS. Thus, C_{req}=(ID_i, F, dis) is obtained, which F represents the friend inquiry, and dis represents the distance threshold of the requester. Finally, (C_{req}, C) is sent to SNS together.

After receiving the information from the requester, the SNS decrypts the C_{req} to get (ID_i, F, dis). At the same time, the SNS looks for the data items about ID_i in the local server and compares the distance threshold of ID_i's friends. SNS gets the minimum value dis$_{min}$=min(dis, df$_{ID_j}$), df$_{ID_j}$ represents the distance threshold of ID_i's friends. Then SNS sets up a Bloom filter, the size of this Bloom filter is predefined, and it is larger than the user's relationship network. SNS inserts each FID$_j$ that satisfies the condition into the Bloom filter. The detail is shown in the Figure 3. Finally send (C, BF$_{req}$, dis$_{min}$) to LS, BF$_{req}$ represents Bloom filter.

When LS receives (C, BF$_{req}$, dis$_{min}$), it first decrypts C to get the requester's location data (X, Y), then inputs the data items from the database into the Bloom filter. LS can find all FID$_m$ that meet the conditions. Then performs vector calculation on the position informations (X_m, Y_m) and (X, Y) to check whether dist((X, Y),(X_m, Y_m))= $\sqrt{(X_m - X)^2 + (Y_m - Y)^2} \leq$ dis$_{min}$, assuming that there are n satisfied, LS returns the data set FID$_i$, Enc(K_{ID_i},(X_i,Y_i) (i ∈1,2,...,n) containing these n data items to SNS.

When SNS receives this data set, it first deletes all location datas that are not friends of the requester according to Table 1. Supposing there are n' satisfying conditions, then SNS converts the n' FID$_i$ into ID_i, and sends ID_i, Enc(K_{IDi},(X_i,Y_i), which i ∈1,2,...,n' and Enc(Pub$_{req}$, δ) to the requester.

Case 2: Stranger inquiry phase

The stranger location inquiry phase is similar to the friend location inquiry phase, but SNS does not need to find friend information and set Bloom filter in this phase. It only needs to find location information that satisfies the location policy. The most important

Algorithm 2: Location sharing

```
Input:
Symmetric key $K_{ID_i}$; LS's public key $Pub_l$; SNS's public key $Pub_s$; requester's ID $ID_i$;
Output:
Req$_{ID_i}$;
Case 1:
1: Requester $ID_i$ executes:
2: $C \leftarrow Enc_{Pub_l}(x, y, K_{ID_i})$;
3: $C_{req} \leftarrow (ID_i, F, dis)$;
4: $ID_i$ send $(C_{req}, C)$ to SNS:
5: SNS executes:
6: dis$_{min}$←min(dis,df$_{ID_j}$);
7: SNS set up a Bloom filter BF$_{req}$;
8: SNS sends $(C, BF_{req}, dis_{min})$ to LS;
9: LS execute:
10: $(x, y)$←$Dec_{Pub_l}(C)$;
11: LS find all FID$_m$ that meet the conditions;
12: for $j = 0$; $j < m$; $j + +$ do
13:       if dist$((x, y), (x_j, y_j)) \leq dis_{min}$
14:             return $FID_j$;
15: Assuming that there are n satisfied;
16: LS send $\{FID_j, Enc(K_{ID_i}, (X_j, Y_j))(j \in 1, 2, ..., n)\}$ to SNS;
17: SNS execute:
18: $ID_j \leftarrow FID_j$;
19: for $j$ = 0; $j$ < $n'$+1; $j$ ++ do
20:       Enc$(K_{ID_i}, (X_j, Y_j))$
21: Enc$(Pub_{req}, \delta)$;
22:SNS send $\{ID_i, Enc(K_{ID_i}, (X_i, Y_i)), Enc(Pub_{req}, \delta)\}$ to requester;
Case 2:
1: Requester ID$_i$ executes:
2: $C \leftarrow Enc_{Pub_l}(x, y, K_{str})$;
3: $C_{req} \leftarrow (ID_i, S, dis)$;
4: ID$_i$ send $(C_{req}, C)$ to SNS:
5: SNS executes:
6: FID$_i \leftarrow ID_i$
7: SNS send $(FID_i, C, dis)$ to LS;
8: for $j = 0$; $j < m$; $j + +$ do
9:       if dist$((x, y), (x_j, y_j)) \leq dis_{min}$
10:             return $FID_j$;
12: Assuming that there are n satisfied;
13: LS send $\{FID_j, Enc(K_{str}, (X_j, Y_j))(j \in 1, 2, ..., n)\}$ to SNS;
14: SNS executes:
15: Assuming that there are n' satisfied;
16: SNS send $\{ID_j, Enc(K_{str}, (X_j, Y_j)), Enc(Pub_{req}, \delta)(j \in 1, 2, ..., n')\}$ to requester;
```

thing is that at this phase, the location data returned to the stranger is no longer the user's actual location but the user's approximate location.

The requester generates a one-time symmetric key K_{str}. Meanwhile, it uses the public key of LS to encrypt its location information (x, y) and K_{str} to get the ciphertext C, C=Enc$_{Pub_L}(x, y, K_{str})$. Thus, C_{req}=(ID_i,S,dis) is obtained, S represents the stranger's query, and dis represents the distance threshold of the requester. Finally, (C_{req},C) is sent to SNS together.

After receiving the information from the requester, the SNS decrypts the C_{req} to obtain (ID_i,S,dis). Then the SNS searches the local server for the data item about ID_i and converts ID_i into FID$_i$. Finally, send (FID_i,C,dis) to LS.

When LS receives (FID_i, C, dis), it decrypts C with his private key to obtain the requester's location data (X, Y), then performs vector calculation to see whether dist((X, Y) , (X_i,Y_i)) $\leq$ dis$_{min}$(dis,dis_{ID_i}). Assuming that there are n satisfied. LS returns the data containing the n data items. Finally, the data sets FID_i, Enc(K_{str},(X_i,Y_i)) ($i \in$1,2,...,n) is given to SNS.

Finally, SNS receives the data set. It returns a false location of each qualified ID_i, because this false location is the nearby coordinates of the requestee. Then our scheme can ensure the security of the requestee's data. Assuming that there are n' satisfying conditions, then SNS converts these n' FID$_i$ into ID_i, but the location shared by the user is any one of the $k-1$ false coordinates that satisfies the condition. Then ID_i,Enc(K_{str},(X_i,Y_i), where $i \in$(1,2,...,n') and Enc(Pub$_{req}$,δ) are given to the requester.

5.4. **Location verification and update.** When the requester obtains the location data, requester can get ID_i, Enc(K_{str},(X_i, Y_i)), and Enc(Pub$_{req}$, δ). Meanwhile, requester decrypts Enc(K_{str},(X_i, Y_i)) and Enc(Pub$_{req}$, δ) to get (X_i, Y_i) and δ, further calculates LocH'$_{ID_i}$=Hash($X_i||Y_i||\delta$). Then downloads LocH$_{ID_i}$, which is saved in the blockchain. If they are equal, it proves that the data is correct.

During the location data update phase, the user generates k new location data, random salt value δ, and calculates LocH$'_{ID_A}$=Hash ($X'_0||Y'_0||\delta$). Then, as shown in line 8 of algorithm 1, Record$_{ID_A}$ is sent to the miner, and the miner packs and uploads it to the blockchain. At the same time, the user sends the k new location data to the SNS. SNS calculates Node$'_{ID_i}$. Then SNS obtains the leaf node of the KN-tree that needs to be updated. Finally, it reconstructs the KN-tree. For example, when there are 8 users for a location update and k=16, our scheme can reduce 48 hash calculations, significantly improving the system's computing efficiency. At the same time, the storage overhead of the blockchain is reduced. In the traditional scheme, 8 users get 8 Merkle root values, and our scheme gets one Merkle root value, thus reducing storage and communication overhead.

6. Analysis Of Scheme.

In this section, we demonstrate that our scheme satisfes all the required properties.

6.1. **Privacy analysis.** We analyze that our scheme can protect users' privacy form disclosure. In order to avoid the SNS obtaining the user's location data, we encrypt the user's location data with the public key Pub$_l$ of LS. Get LocData$_i$ = Enc(Pub$_l$,(X_i,Y_i)). At the same time, the user will encrypt the one-time key $Enc_{Pub_l}(x, y, K_{ID_i})$ through the LS public key. Then, when the LS returns the location data to the requester, the SNS can only get the ciphertext about the user's location data. This prevents the user's location data from being leaked to LS. At the same time, because LS stores a lot of location data, to avoid LS guessing the user's social network information, we insert the applicant's friend FID into the Bloom filter BF and then send the BF to LS. When LS receives the BF, it inputs all FIDs into BF to find the FID that meets the conditions. Because the Bloom filter has a false positive rate, LS will misjudge the friend who does not belong to the applicant, thus increasing the difficulty of LS in guessing the user relationship network and improving the security of the system.

6.2. **Brute-force attack resilience.** As mentioned in Section 4.2, an external adversary can download all data in the blockchain, such as $Hash\,(X_0||Y_0||\delta)$. Then it uses brute force algorithm to collide with $Hash\,(X_0||Y_0||\delta)$ to obtain $Hash\,(X'||Y')$, However $Hash\,(X'||Y')$ is not the user's location data. Because all users use salt encryption before uploading location data to the blockchain, and the salt values are different. At the same time, the hash values obtained are different due to different δ for the same location data sent by different users. external adversaries cannot guess which users have the same location through rainbow attacks. Therefore, our scheme can resist the violence of external adversaries.

6.3. **Formal analysis based on ROR model.** Now, we adopt the ROR model to prove the proposed scheme formally. ROR model is generally used to demonstrate the security of authentication protocols [28]. Next, we also use the ROR model to prove that the authentication part of this protocol is secure.

Theorem 1. Suppose U_A is an adversary running in polynomial time t for scheme P in the random oracle model. Assuming that the SNS is not broken, the advantages obtained by breaking the semantic security of the scheme P are:

$$Adv_p(A) \leq \frac{q_h^2}{|Hash|}$$

Where q_h, $|Hash|$ represents the number of hash queries and the scope space of the hash.

Proof: The detailed description of defined three games, $G_i (i = 0, 1, 2)$. Let $Succ_i$ represents that U_A successfully predicts the event of C_i in game G_i.

G_0: In game G_0, the actual attack of adversary U_A against P in the random oracle model. In addition, C_i is randomly selected before the game. By definition:

$$Adv_p(U_A) = 2Pr[Succ_0] - 1$$

G_1: This game is modelled that U_A can eavesdrop on messages. Firstly, U_A performa an *Execute* request and *Test* request. Then, U_A needs to verify whether the result returned after the *Test* request is the real location information or a random number. In this scheme, it is assumed that U_A intercepts all parameters $secret_{ID_A}$, $LocH_{ID_A}$ and C_{ID_A} transmitted in the location-sharing phase. They are not transmitted over the channel in plaintext information. U_A cannot determine the private key pri$_s$ from the records. Therefore, eavesdropping will not increase the chances of the adversary's success in this game. We then obtain,

$$Pr[Succ_1] = Pr[Succ_0]$$

G_2: This game simulates all hash collisions in location-sharing. The difference between G_1 and G_2 is added to the Hash oracles. Meanwhile, in this game, Hash and Send queries are performed to model it calls an "active attack". Adversary U_A tries to deceive the LR to complete the location-sharing. U_A repeatedly queries the Hash oracle for collisions, however, each message is associated with the security level n set by the LR. According to the birthday paradox, the probability of a hash collision is $\frac{q_h^2}{|Hash|}$. Finally, We can obtain:

$$|Pr[Succ_1] - Pr[Succ_2]| \leq \frac{q_h^2}{|Hash|}$$

TABLE 3. Comparison with related works.

Scheme	BMobishare [18]	BMPLS [19]	TPPLS [25]	LSPP [27]	Our scheme
Confidentiality	√	√	√	×	√
Tamper-proof	√	√	×	√	√
Decentralization	×	√	×	√	√
Traceability	×	√	×	−	√
Batch verifiability	×	×	×	×	√

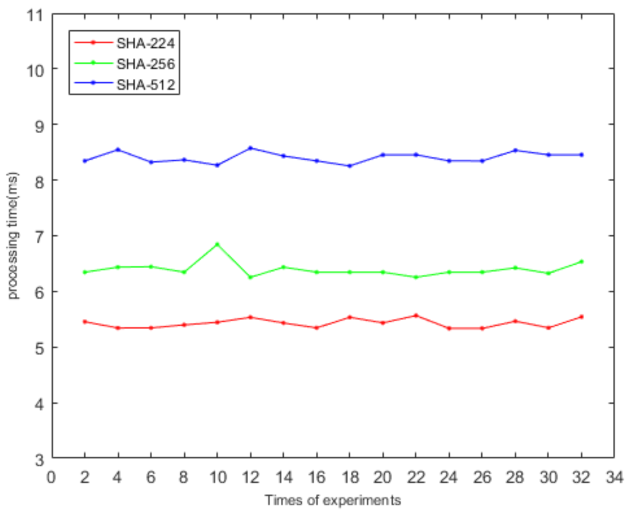

FIGURE 4. Comparison of encryption algorithms

In all cases there are $Pr[Succ_1] = Pr[Succ_0]$, $|Pr[Succ_1] - Pr[Succ_2]| \leq \frac{q_h^2}{|Hash|}$. We can obtain the following result

$$Adv_p(A) \leq \frac{q_h^2}{|Hash|}$$

From this equation, it can be seen that when the range space of the hash function is large, scheme P is safe.

6.4. **Comparison With Related Work.** In this section, we theoretically analyse our scheme and proved that it satisfies all the above characteristics. We also compare our scheme with other related works, including BMobishare [18], BMPLS [19], TPPLS [25] , LSPP [27]. The results in Table 3 show that our scheme has better characteristics in all aspects, where "√ " denotes satisfied, " × " denotes dissatisfied, and " − " denotes uninvolved.

7. **Performance Evaluation.** In this section, we evaluate the computational overhead of our proposed scheme. The experiment consists of three stages according to SMOSNs , ie., initialization, location sharing, location update. The experiments are implemented on a personal computer with a PC(CPU:Intel(R) Core(TM) i5-10300H CPU @2.50GHz 2.50 GHz, RAM:16G, OS:Windows 10) using python-3.7.

7.1. **Initialization phase.** In the initialization phase. Users encrypt the location data through salt encryption to avoid the rainbow attack. DO need to complete Hash(X_0,Y_0,δ), Hash(X_1,Y_1,δ), ... , Hash(X_{K-1},Y_{K-2},δ). In Figure 4, we use a hash algorithm with

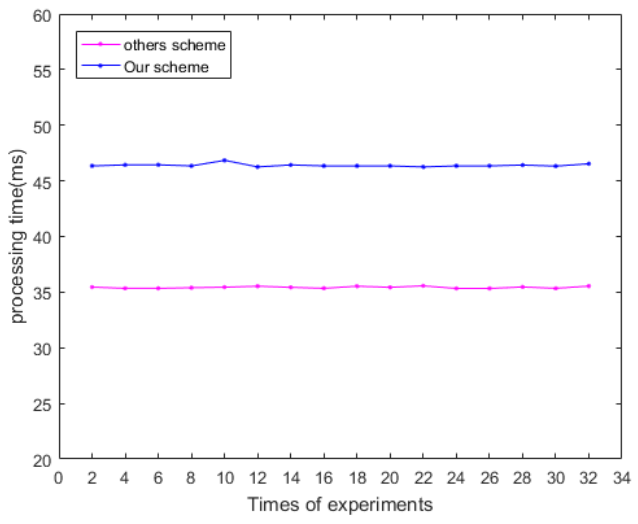

FIGURE 5. Time cost of DO

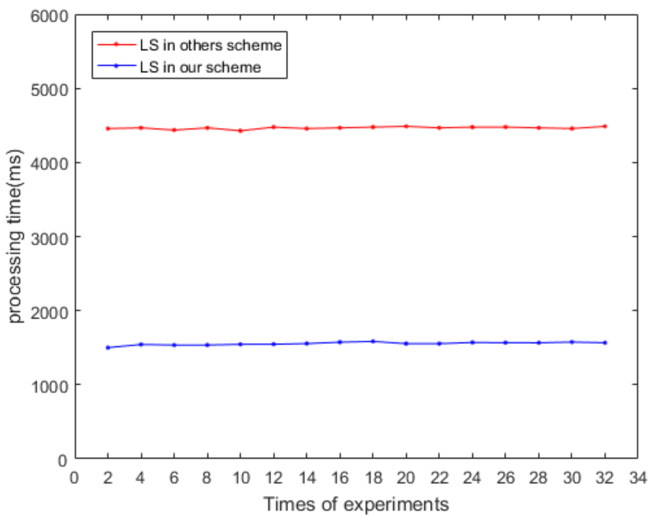

FIGURE 6. Time cost of LS

different parameters to calculate the user's computing cost. From Figure 4, We can note that the hash algorithm with different parameters will lead to slightly different computational costs. When we use SHA-224, SHA-256, and SHA-512, the computing cost is still less than 10ms. Therefore, the computation cost of initialization is minimal for users.

7.2. **Locaiton sharing phase.** In this phase. This computational cost is mainly reflected in the construction of the KN-tree. Its construction process is more straightforward than the traditional Merkle tree and requires fewer hashes. Each leaf node represents different user location data. k location data of the same user can be used as the leaf node of the KN-tree after k times of hash operation, which has a relatively minor update cost in this scenario that needs frequent updates. In Figure 5, Our scheme requires less computing overhead with an equal amount of data because our leaf node is not composed of K-position data. When k=16, our scheme reduces 6 hash operations compared with the traditional Merkle tree. In this way, there is less computational overhead in the user

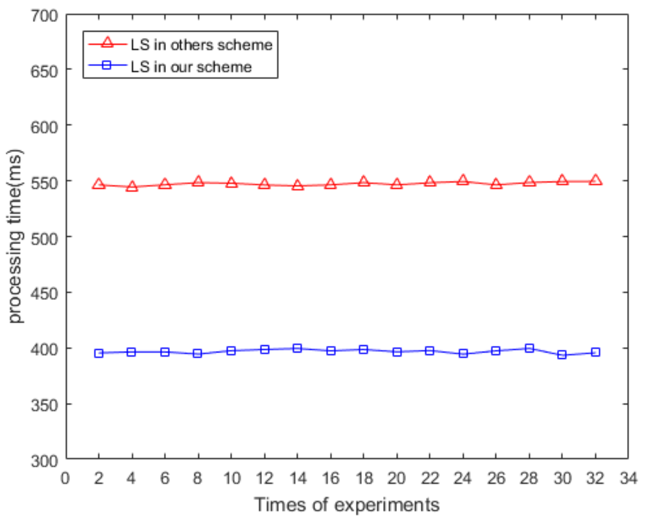

Figure 7. Time cost of LS

construction and update process. KN-tree can verify multiple user data at the same time, which significantly improves the system's efficiency.

At the same time, we use an accumulator to reduce the computing cost of the server. We set the location data to 200, including 10 locations that meet the user's needs. We calculate the computing cost of the traditional scheme and our scheme. From Figure 6, we can see that the time required for our scheme to filter out the correct demand location data is less than the traditional scheme to calculate all the location data which meet the conditions. When the number of location data is large, our scheme reduces computational overhead and dramatically improves the system's efficiency.

7.3. **Locaiton update phase.** At this phase, users generates KN-tree nodes. As shown in the Figure 7, in this process, when k=16, n=8, the user's computations in our scheme are 6 times less than in the traditional scheme, reducing the generation of intermediate nodes. Meanwhile, the location server will reduce 48 intermediate node calculations. When the user's k is relatively large, the computation overhead of the user and location server are significantly reduced.

8. **Conclusion.** Secure sharing of location data is the basis of online social networks. However, there is an awkward problem in the process of sharing. Location data involves the privacy of users. It is natural to enable data owners to own and control their data. Currently, most of the schemes have the risk of location leakage and the problem of low efficiency. In this paper, we propose a new KN-tree to verify the integrity of location data. Since we form a Merkle tree with the locations of multiple users, batch verification of user data can be realized, significantly reducing the computational and communication overhead. At the same time, the blockchain is used as a trusted third party to replace the traditional centralized storage and help complete data sharing. The decentralized and tamper-proof features of blockchain make our scheme more robust. Salted encryption can effectively prevent malicious adversaries from brute-forcing the user's location data through rainbow attacks, further enhancing the scheme's security. Therefore, our scheme is superior to the existing schemes. Finally, in future work, we will focus on further reducing the computational overhead of users and servers while exploring more efficient encryption schemes.

Acknowledgements. This work was supported in part by the National Social Science Fund of China (No.21XTQ015), the Natural Science Foundation of Fujian Province of China under Grant (No.2019J01752, No.2020J01905), the Research Project of Fujian Provincial Department of Education of China under Grant (No.JAT200293), Science and Technology Project of Putian City (No.2021R4001-10) and the Presidential Research Fund of Minnan Normal University (No.KJ18024).

REFERENCES

[1] K. Shafique, B. A. Khawaja, F. Sabir, M. Mustaqim, "Internet of things (iot) for next-generation smart systems: a review of current challenges, future trends and prospects for emerging 5g-iot scenarios", *IEEE Access*, vol. 8, pp. 23022–23040, 2020.

[2] M. Salehan, A. Negahban, "Social networking on smartphones: When mobile phones become addictive", *Computers in Human Behavior 29*, vol. 29, pp. 2632-2639, 2013.

[3] Y. Y. Ahn, S. Han, H. Kwak, S. Moon and H. Jeong, "Analysis of topological characteristics of huge online social networking services", *The 16th International Conference on World Wide Web*, pp. 835–844, 2007.

[4] R. Wilson, D. G. Samuel, "A review of facebook research in the social sciences", *Perspectives on Psychological Science*. vol. 7, pp. 203–220, 2012.

[5] D. Pralhad, H. X. Xiao, "Performance comparison of 3G and metro-scale WiFi for vehicular network access", *The 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 301–307, 2010.

[6] W. Wei, W. Yue, "Oscillation resolution for mobile phone cellular tower data to enable mobility modelling", *2014 IEEE 15th International Conference on Mobile Data Management*, pp. 321–328, 2014.

[7] D. T. Hoang, L. Chonho, N. Dusit, W. Ping, "A survey of mobile cloud computing: architecture, applications, and approaches", *Wireless Communications and Mobile Computing*, vol. 13, pp. 1587–1611, 2013.

[8] L. Ling, "From data privacy to location privacy: Models and algorithms", *International Conference on Very Large Data Bases*, pp. 1429–1430, 2007.

[9] W. Wei, X. J. Zhu, Q. Li, "Lbsnsim: Analyzing and modeling location based social networks", *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 1680–1688, 2014.

[10] A. Boldyreva, N. Chenette and A. O′Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions", *Annual Cryptology Conference*, pp. 578-595, 2011.

[11] L. A. Sweeney, "K-anonymity", *International Journal of Uncertainty*, vol. 10, pp. 557-570, 2002.

[12] J. Chen, S. U. Shen and X. Wang, "Towards privacy-preserving location sharing over mobile online social networks", *IEICE TRANSACTIONS on Information and Systems*, vol. 102, pp. 133–146, 2019.

[13] K. Christidis, M. Devetsikiotis, "Devetsikiotis, Blockchains and smart contracts for the internet of things", *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[14] J. Li, S. L. Xie, "Joint transaction relaying and block verification optimization for blockchain empowered d2d communication", *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 828–841, 2019.

[15] K. Zhang, Y. Zhu, S. Maharjan, Y. Zhang, "Edge intelligence and blockchain empowered 5g beyond for the industrial internet of things", *IEEE Network*, vol. 33, pp. 12–19, 2019.

[16] G. Zyskind, O. Nathan, "Decentralizing privacy: Using blockchain to protect personal data", *IEEE Symposium on Security and Privacy*, pp. 180–184, 2015.

[17] S. Cao, G. X. Zhang, P. F. Liu, X. S. Zhang and F. Neri, "Cloud-assisted secure ehealth systems for tamper-proofing ehr via blockchain", *Information Sciences*, vol. 485, pp. 427–440, 2019.

[18] N. Shen, J. Yang, K. Yuan, C. Fu and C. F. Jia, "An efficient and privacy-preserving location sharing mechanism", *Computer Standards Interfaces*, vol. 44, pp. 102–109, 2016.

[19] Y. X. Ji, J. W. Zang, J. F. Ma, C. Yang, X. Yao, "BMPLS: Blockchain-based multi-level privacy-preserving location sharing scheme for telecare medical information systems", *Journal of Medical Systems*, vol. 42, pp. 1–13, 2018.

[20] Q. H. Wang, T. Y. xia, Y. Z. Ren, L. F. Yuan and G. X. Miao, "A new blockchain-based multi-level location secure sharing scheme", *Applied Sciences*, vol. 11, pp. 2260, 2021.

[21] M. Ali, R. Dhamotharan, E. Khan, "SeDaSC: Secure data sharing in clouds", *Applied Sciences*, vol. 11, pp. 395-404, 2015.

[22] M. Nofer, P. Gomber, O. Hinz and D. Schiereck, “Blockchain”, *Business Information Systems Engineering*, vol. 59, pp. 183–187, 2017.

[23] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”, *Decentralized Business Review*, pp. 21260, 2008.

[24] H. Wang, “Privacy-preserving data sharing in cloud computing”, *Journal of Computer Science and Technology*, vol. 25, pp. 401–414, 2010.

[25] J. Chen, S. Su and X. Z. Wang, “Towards privacy-preserving location sharing over mobile online social networks”, *IEICE TRANSACTIONS on Information and Systems*, vol. 102, pp. 133–146, 2019.

[26] T. Wu, X. Guo, Y. Chen, “Amassing the security: An enhanced authentication protocol for drone communications over 5G networks”, *Drones*, vol. 6, pp. 10, 2022.

[27] L. Yang, Y. C. Chen, T. Y. Wu, “Provably secure client-server key management scheme in 5G networks”, *Wireless Communications and Mobile Computing*, vol. 2021, pp. 4083199, 2021.

[28] T. Y. Wu, Z. Lee, “An authenticated key exchange protocol for multi-server architecture in 5G networks”, *IEEE Access*, vol. 8, pp. 28096-28108, 2020.

[29] Q. Mei, H. Xiong, Y. C. Chen, “Blockchain-enabled privacy-preserving authentication mechanism for transportation cps with cloud-edge computing”, *IEEE Transactions on Engineering Management*, pp. 1-12, 2022.

[30] C. M. Chen, T. Z. Wang, E. K. Khan, “Verifiable dynamic ranked search with forward privacy over encrypted cloud data”, *Peer-To-Peer Networking and Applications*, vol. 14, pp. 2977-2991, 2021.

[31] H. Zhu, Y. J. Guo and L. B. Zhang, “An improved convolution merkle tree based blockchain electronic medical record secure storage scheme”, *Journal of Information Security and Applications*, vol. 61, pp. 102952, 2021.

[32] O. Ruan, L. X. Zhang and Y. Y. Zhang, “Location-sharing protocol for privacy protection in mobile online social networks”, *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, pp. 1–14, 2021.

[33] M. Abdalla, P. A. Fouque, D. Pointcheval, “Password-based authenticated key exchange in the three-party setting”, *IEE Proceedings Information Security*, vol. 153, pp. 27-39, 2006.