sensors

MDPI

Article

Privacy-Enhancing *k*-Nearest Neighbors Search over Mobile Social Networks †

Yuxi Li [1,*,‡], Fucai Zhou [2], Yue Ge [2] and Zifeng Xu [3]

1 School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
2 Software College, Northeastern University, Shenyang 110819, China; fczhou@mail.neu.edu.cn (F.Z.); yggeyue@foxmail.com (Y.G.)
3 School of Cybergram, Hainan University, Haikou 570228, China; tnimdk@gmail.com
* Correspondence: liyuxi@cse.neu.edu.cn
† This paper is an extended version of our paper published in "Li, Y.; Zhou, F.; Xu, Z.; Ge, Y. PPFQ: Privacy-Preserving Friends Query over Online Social Networks. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; pp. 1348–1353, doi:10.1109/TrustCom50675.2020.00181.".
‡ Current address: No.195, Chuangxin Road, Hunnan District, Shenyang 116024, China.

Abstract: Focusing on the diversified demands of location privacy in mobile social networks (MSNs), we propose a privacy-enhancing *k*-nearest neighbors search scheme over MSNs. First, we construct a dual-server architecture that incorporates location privacy and fine-grained access control. Under the above architecture, we design a lightweight location encryption algorithm to achieve a minimal cost to the user. We also propose a location re-encryption protocol and an encrypted location search protocol based on secure multi-party computation and homomorphic encryption mechanism, which achieve accurate and secure *k*-nearest friends retrieval. Moreover, to satisfy fine-grained access control requirements, we propose a dynamic friends management mechanism based on public-key broadcast encryption. It enables users to grant/revoke others' search right without updating their friends' keys, realizing constant-time authentication. Security analysis shows that the proposed scheme satisfies adaptive $\mathcal{L}$-semantic security and revocation security under a random oracle model. In terms of performance, compared with the related works with single server architecture, the proposed scheme reduces the leakage of the location information, search pattern and the user–server communication cost. Our results show that a decentralized and end-to-end encrypted *k*-nearest neighbors search over MSNs is not only possible in theory, but also feasible in real-world MSNs collaboration deployment with resource-constrained mobile devices and highly iterative location update demands.

Keywords: mobile social networks; privacy-enhancing; collaboration architecture; location search; secure multi-party computation; homomorphic encryption

Citation: Li, Y.; Zhou, F.; Ge, Y.; Xu, Z. Privacy-Enhancing *k*-Nearest Neighbors Search over Mobile Social Networks. *Sensors* **2021**, *21*, 3994. https://doi.org/10.3390/s21123994

Academic Editor: Rongxing Lu

Received: 16 April 2021
Accepted: 4 June 2021
Published: 9 June 2021

1. Motivation

With the rapid development of 5G Wireless Communication, mobile social networks (MSNs), represented by instant messaging and location sharing, have become essential parts of people's everyday lives. According to [1], the number of enrolled users in MSNs worldwide reaches 862 million in 2020, and it is estimated to exceed 900 million by the end of 2021. In particular, the utilization rate of location-based MSN services reaches 96.9% based on the positioning system (e.g., GPS, WiFi, Bluetooth, etc.) embedded in mobile devices, such as Facebook's "Nearby friends", Foursquare's "Swarm", and Joyrun's "real-time running competition", and so forth. In these services, users can broadcast their locations among friends and send location-based queries for nearby friends. Therefore, the location-based services provide a profoundly mobile interface for users' real-life social networks.

Nevertheless, people are using the enormously popular MSNs services without realizing their privacy concerns: the MSNs services providers can observe and accumulate the geo-location that users transmit through the network. According to the Mobile APP Security Research [2], among the 50 MSN services surveyed, there are 35 services that leak users' location data to advertisers or data analysis services on purpose without any permission. In recent years, a lot of research also uses data analysis and machine learning technology to extract a large number of their sensitive information from users' location information over MSNs: by analyzing their search patterns, the matched friends and search similarities and so forth, it is easy to predict the location conversion patterns between users and their friends [3,4]. It is also turned out that Facebook's historical spatiotemporal trajectory leaks the geographical distance between each user and the spot that he frequently queries, and then the service provider can learn the access probability—whether and when the user will check-in the next time [5].

To avoid illegal access to users' locations and search patterns by unauthorized service providers and hackers, previous research aimed to encrypt the location information before uploading. However, traditional encryption methods limit the MSNs service provider's ability to provide location-based services for users. To achieve privacy-preserving location-based query, the straightforward mathematical methods deploy private information retrieval [6], searchable encryption [7] and other crypto primitives to make the encrypted location data searchable. However, these methods come at the huge cost of computation and communication overhead. Moreover, in a privacy-preserving setting, users have their keys embedded in their mobile devices. If a user is allowed to share his/her location encryption key with friends, he/she needs to launch a search request to the platform multiple times when he wants to retrieve his/her friends' locations. Moreover, when granting or revoking friends' search rights, a user and his/her friends should update their keys with synchronous locally, which are not suitable for MSNs platforms with highly extensible requirements. To the best of our knowledge, it is fair to say that achieving fine-grained location access control while providing an efficient, secure location-based neighbors search service has become one of the challenging research topics in the field of privacy-enhancing MSNs and still remains open.

In this work, we translate the high-level vision of the above issues and location privacy demand in MSNs into technical requirements and design a privacy-enhancing k-nearest neighbors search scheme containing cryptographic protocols that meet them. The purpose of this work is to protect users' location data and search patterns privacy, and make it available for users to query for their k-nearest neighbors based on current distances. In terms of technical contribution, our work presents an efficient construction so that the server can effectively compute and sort the encrypted distance between a user and his/her friends without any decryption operation, which is the first to tackle this problem through the lens of secure multi-party computation. It also achieves lightweight friend authentication and authority management by enabling users to grant/revoke their friends' search rights without updating others' keys. In terms of security, our scheme satisfies adaptive $\mathcal{L}$-semantic secure and revocation secure under random oracle model. We also undertook an extensive experiment that validates our work, showing that the proposed scheme is possible in theory and feasible in practice.

2. Related Works

2.1. MSNs Privacy

In recent decades, researchers have proposed many privacy-preserving approaches for MSNs. Encryption is the most common method for achieving privacy. For example, Flybynight [8] is a Facebook application for encrypting and decrypting sensitive data using client-side JavaScript. However, it is easy to be attacked by an adversary because the server holds users' keys and takes charge of key management. NOYB (short for none of your business) [9] offers privacy and preserves MSN services' functionality based on a secret dictionary's encryption. Besides, there have been many privacy-preserving

matching solutions over MSNs proposed with different techniques. Some schemes are based on private set intersection protocols [10,11] to allow two users to compute the intersection of the two private profile sets privately, but leak no useful information of both parties. For example, Niu et al. designed a spatiotemporal matching scheme for privacy-aware users in MSNs based on the profile's weight or level and the participant's social strength [10]. Zhang et al. proposed the concept of a fine-grained privacy information matching protocol by giving preference to each profile and using a similarity function to measure the matching degree [11]. To reduce computation cost, some works [12,13] designed non-encryption-based privacy-preserving matching protocols. Fu et al. proposed a privacy-preserving common-friend matching scheme based on a bloom filter [12]. It transmitted the common profiles of two users into an intersection of bloom filters, which ensures the privacy of friend lists against unknown users. However, it will not be able to resist brute-force attacks, resulting in privacy information leakage. Sun et al. [13] proposed a privacy-preserving spatiotemporal profiles matching scheme to let each user periodically record his locations by a geographic cell index among a large set of predefined ones, which can ensure spatiotemporal privacy at the cost of possibly huge communication and computation overhead.

2.2. Location Privacy

With the rapid development and enormous popularity of location-based services, scholars have paid more and more attention to location data's privacy and security. Many approaches focus on how to perform privacy-preserving location queries: Bamba et al. proposed a *k*-anonymity-based scheme that relies on a server to construct an anonymous set based on users' original queries to make query indistinguishable on the server-side [14]. Bordenabe et al. [15] and Shi et al. [16] both integrated differential privacy to realize nearby friends' queries. Differential privacy provides a rigorous privacy guarantee by adding noise (randomly to choose a set of fake locations) to make their data and query deferentially private. Jorgensen et al. incorporated a clustering procedure that groups users according to the social network's natural community structure and significantly reduced noise [17]. The above works [14–17] can achieve relatively high efficiency. However, the limitations of these works are that it is challenging to achieve provable security guarantees with formal security definitions, since they did not employ well-designed and provable encryption methods. Zhou et al. took advantage of private information retrieval (PIR) to realize nearby friends' queries [18]. It provides strong cryptographic guarantees but needs complex operations, and it only protects query privacy but not location privacy. Li et al. designed a private location information matching protocol over MSNs based on inner product similarity (IPS) [19], putting users' map locations into vectors and encrypting the vectors. The similarity function is used to measure the similarity degree of the encrypted vectors of different users. Schlegel et al. designed an encryption method of dynamic location grid index structure [20], achieving neighbor point search on the premise of not revealing location privacy to the third party. In the above encryption-based schemes, the computation efficiencies are not ideal, requiring multi-round interactions at the logarithmic level between user and server.

Many other works are based on higher security assumptions to achieve a trade-off between security and efficiency. For example, some works [21,22] assumed that the service provider is honest and that it has the authority to access the location plaintext without leaking any information to others. Some works [21,23,24] introduced a trusted third party (TTP) to achieve the trade-off between security and efficiency. Unfortunately, there may not exist such a TTP in real MSNs scenarios. Some non-TTP solutions [15,20] are based on approximate measurements (e.g., linear programming and dynamic grid) with no accurate result. Some works [18,25] need complex operations (e.g., sending fake queries or receiving redundant results) to achieve secure guarantees, which incur high communication and computation overhead at the user-side, making them unsuitable for resource-constrained mobile devices.

3. The Proposed Scheme

3.1. Overview

The privacy-enhancing k-nearest neighbors search scheme over MSNs can be viewed as a decentralized system of end-to-end encrypted social network databases, focusing on the diversified demands of location privacy in MSNs. Our design relies on various cryptographic building blocks, including pseudo-random function, homomorphic crypto mechanism, secure multi-party computation and broadcast encryption.

- Aiming at the limited computation power of resource-constrained mobile devices, we design a lightweight end-to-end location encryption algorithm and a server-aid location re-encryption protocol based on Paillier homomorphic encryption to achieve further location sharing. The protocol allows the service provider to transfer friends' location ciphertexts into the query user's homomorphic ciphertexts without requiring them to be online to participate in the calculation.
- We build a secure dual-server architecture and design a secure k-nearest neighbors search protocol by secure multi-party computation and a homomorphic encryption mechanism under this architecture. The server can effectively compute and sort the distance between users and their friends without any decryption operation. Compared with the cloud-center model, where a single server holds complete knowledge, the dual-server architecture minimizes the leakage to the servers and reduces the cost of communication between the mobile user and the server.
- To achieve fine-grained access control, we design a dynamic friends management mechanism based on public-key broadcast encryption. It enables users to grant/revoke their friends' search rights without updating others' keys, achieving lightweight friend authentication and authority management. Moreover, this mechanism satisfies revocation secure that the adversary cannot obtain the user's location information through collusion with the server and the revoked friends, thus further improving the scheme's overall security.

3.2. Architecture and Syntax

Our scheme is designed to be executed among: **U**, $\mathcal{S}_1$, and $\mathcal{S}_2$. **U** is a set that contains n mobile users $\{\mathcal{U}_1, ..., \mathcal{U}_n\}$. Each user $\mathcal{U}_i \in \mathbf{U}$ can connect with others as his friends dynamically. $\mathcal{S}_1$ is the primary server that provides a mobile social network service to all users in **U**. Each user $\mathcal{U}_i \in \mathbf{U}$ can send a search request to $\mathcal{S}_1$ for k-nearest neighbors among friends based on current location. $\mathcal{S}_2$ is a collaborated server to conduct secure computation with $\mathcal{S}_1$ for k-nearest neighbors search.

The scheme's architecture is shown in Figure 1. At a high level, users' information and their relationships are modeled by a direct graph structure $\mathcal{G}$. To initialize the system, the primary server $\mathcal{S}_1$ executes **Initial** algorithm to output public parameter *params* and an empty $\mathcal{G}$. Any user $\mathcal{U}_i$ should use public parameter *params* to generate his symmetric key K_i and public/secret keys (PK_i, SK_i) locally by executing **KeyGen** algorithm and interacts with the primary server $\mathcal{S}_1$ for registration by **Join** protocol. Any enrolled user $\mathcal{U}_i \in \mathbf{U}$ can grant/revoke $\mathcal{U}_j \in \mathbf{U}$'s location search right by interacting with $\mathcal{S}_1$ in **Grant/Revoke** protocols. $\mathcal{U}_i$ holds a friends index F_i that records his granted friends. According to the real-life MSNs' location service architecture, we deploy trusted location infrastructure to provide tracing service by sending the current location l_i of each user $\mathcal{U}_i \in \mathbf{U}$ to his local mobile device periodically. $\mathcal{U}_i$ executes **LocUpdate** to encrypt his location data l_i by his symmetric key K_i at local and uploads the location ciphertext C_i to $\mathcal{S}_1$. $\mathcal{U}_i$ then can execute **Search** protocol with $\mathcal{S}_1$ by sending k-nearest neighbors search request. $\mathcal{S}_1$ performs encrypted search in $\mathcal{G}$ with the assistance of $\mathcal{S}_2$ and returns the search result to $\mathcal{U}_i$, without relying on the presence of any other user. The proposed scheme's syntax consists of seven polynomial-time algorithms and protocols, which is shown in **Syntax** below:

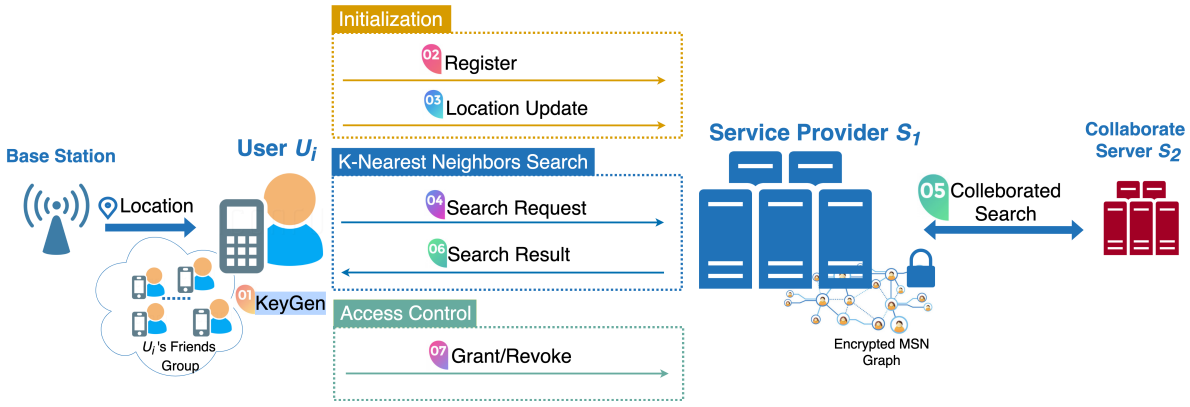


Figure 1. The architecture.

Syntax

- $(\mathcal{G}, params) \leftarrow$ **Initial**(1^k)
- $(K_i, PK_i, SK_i) \leftarrow$ **KeyGen**$(params)$
- $(\mathrm{F}_i; \mathcal{G}') \leftarrow$ **Join**$(\mathcal{U}_i(id_i, PK_i); \mathcal{S}_1(\mathcal{G}))$
- $(\perp; \mathcal{G}') \leftarrow$ **LocUpdate**$(\mathcal{U}_i(K_i, l_i); \mathcal{S}_1(\mathcal{G}))$
- $((\mathrm{F}'_i, K'_i); \mathcal{G}') \leftarrow$ **Grant**$(\mathcal{U}_i(id_j, K_i, PK_i, \mathrm{F}_i); \mathcal{S}_1(\mathcal{G}))$
- $(R_k; \perp; \perp) \leftarrow$ **Search**$(\mathcal{U}_i(K_i, \mathrm{F}_i); \mathcal{S}_1(\mathcal{G}); \mathcal{S}_2(SK_i))$
- $(\mathrm{F}'_i; \mathcal{G}') \leftarrow$ **Revoke**$(\mathcal{U}_i(id_j, PK_i, \mathrm{F}_i); \mathcal{S}_1(\mathcal{G}))$

Definition 1 (Correctness). *Correctness implies that, for all* 1^k*, all* $(\mathcal{G}, params)$ *generated by* ***Initial***(1^k)*, all* (K_i, PK_i, SK_i) *generated by* ***KeyGen***$(params)$*, all* $(\mathrm{F}_i; \mathcal{G}')$ *generated by* ***Join***$(\mathcal{U}_i(id_i, PK_i); \mathcal{S}_1(\mathcal{G}))$*, all* (K_i, PK_i, SK_i) *generated by* ***KeyGen***$(params)$*, and all sequences of* ***LocUpdate***, ***Grant*** *and* ***Revoke*** *protocols,* ***Search***$(\mathcal{U}_i(K_i, F_i); \mathcal{S}_1(\mathcal{G}); \mathcal{S}_2(SK_i))$ *will always output result* R_k *that:* R_k *satisfies* $D_1 < \cdots < D_K$*; and there does not exist* $\mathcal{U}_i \in R_k$ *such that* $\mathcal{U}_i \notin F_s$ *and* $\mathcal{U}_i \in \{F_s \backslash R_k\}$ *that* $D_i < max_{\mathcal{U}_j \in R_k}\{D_j\}$.

3.3. Security Definition

3.3.1. Adaptive $\mathcal{L}$-Semantic Secure

The security definition of adaptive $\mathcal{L}$-semantic secure is formalized by an ideal/real-world paradigm [7]. Roughly speaking, we require that the execution of the scheme in the real-world is indistinguishable from an ideal-world. In real-world Real(1^k), the protocols between the adversarial servers and the user execute just like in the real scheme. In ideal-world Ideal(1^k), there exist two simulators Sim_1 and Sim_2 that can obtain the leakage information from leakage functions and try to simulate the execution of $\mathcal{A}_1$ and $\mathcal{A}_2$ in Real(1^k).

Definition 2 (Adaptive $\mathcal{L}$-Semantic Secure). *Given the syntax in Section 3.2 and considering the following probabilistic paradigms, where* $\boldsymbol{U} = \{\mathcal{U}_1, ..., \mathcal{U}_n\}$ *is the users' set,* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are two non-colluding adversaries with pseudo-random polynomial time (PPT) computation ability,* Sim_1 *and* Sim_2 *are two PPT simulators and* $\mathcal{L}_1$ *to* $\mathcal{L}_4$ *are leakage functions.*

Real(1^k)*: It is run among the* $\mathcal{A}_1$*,* $\mathcal{A}_2$ *and* $\boldsymbol{U}$ *using the real scheme.*

- $\mathcal{A}_1$ *initializes a empty graph structure* $\mathcal{G}$ *by* $(\mathcal{G}, params) \leftarrow$ ***Initial***(1^k)*;*
- *Every* $\mathcal{U}_i \in \boldsymbol{U}$ *computes* $(K_i, PK_i, SK_i) \leftarrow$ ***KeyGen***$(params)$*;*
- *Every* $\mathcal{U}_i \in \boldsymbol{U}$ *interacts with* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *to execute* ***LocUpdate***, ***Grant*** *and* ***Revoke*** *protocols in any order;*
- $\boldsymbol{U}$ *send polynomial times queries* $(q_1, \ldots, q_t)$ *to* $\mathcal{A}_1$*;*
- *For each* q_i*:*
- $\mathcal{U}_s$ *and* $\mathcal{A}_1$ *run* ***Search*** *protocol with* $\mathcal{A}_2$*;*
- $\mathcal{A}_1$ *outputs* R_k^i *.*

Ideal(1^k)*: It is run by* Sim_1 *and* Sim_2 *and a challenger* $\mathcal{C}$*.*

- $\mathcal{A}_1$ *initializes a empty simulated* $\widetilde{\mathcal{G}}$ *and update it by* $\mathcal{L}_1$*;*
- Sim_1 *and* Sim_2 *simulate* ***LocUpdate***, ***Grant*** *and* ***Revoke*** *protocols in any order by* $\mathcal{L}_2$*;*
- $\mathcal{C}$ *sends polynomial times queries* $(q_1, \ldots, q_t)$ *to* Sim_1*;*
- *For each* q_i*:*
- Sim_1 *and* Sim_1 *simulate* ***Search***$(\mathcal{S}im_1^{\mathcal{L}_3,\mathcal{L}_4}(1^k), \mathcal{S}im_2^{\mathcal{L}_4}(1^k))$ *by* $\mathcal{L}_3$ *and* $\mathcal{L}_4$*;*
- Sim_1 *outputs the simulated* $\widetilde{R}_k^i$*.*

The proposed scheme achieves adaptive $\mathcal{L}$*-semantic security if, for all polynomial time* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*, there exists polynomial time simulators* Sim_1 *and* Sim_2 *such that the following two distribution ensembles are computationally indistinguishable:*

$$Output_{\mathcal{A}_{1/2}}^{Real(1^k)} \approx Output_{Sim_{1/2}}^{Ideal(1^k)}.$$

3.3.2. Revocation Security

Revocation security guarantees that the scheme satisfies that any user's revoked friend cannot provide a valid search for his location, even if an adversary illegally steals the revoked friend's key. We construct the experiment $\mathbf{Exp}_{\mathcal{A}_{rev}}^{\text{Revoke}}(1^k)$ to formalize the revocation security definition. $\mathbf{Exp}_{\mathcal{A}_{rev}}^{\text{Revoke}}(1^k)$ is interactively executed by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_{rev}$ who has the ability to add friends, perform a search and revoke friends in the real scheme. $\mathcal{C}$ deletes the user who has been added to the friends index by $\mathcal{A}_{rev}$. $\mathcal{A}_{rev}$ continues to generate a search token using the revoked friend's identity and makes a search request. After a polynomial number of queries, $\mathcal{C}$ revokes all users that are queried to the Grant oracle but are not subsequently queried to the Revoke oracle (i.e., all users for which $\mathcal{A}_{rev}$ holds their valid user keys).

The adversary $\mathcal{A}_{rev}$ must then produce a search token which, when used as an input to **Search** protocol, does not produce null, that is, $\mathcal{A}_{rev}$ must produce a valid search request even though it does not hold a non-revoked key. After several rounds of queries, if $\mathcal{A}_{rev}$'s probability of winning the revocation security experiment with PPT computation ability is negligible, then we can say that the proposed scheme satisfies revocation security.

Definition 3 (Revocation Secure)**.** *Given the syntax in Section 3.2 and considering* $\boldsymbol{Exp}_{\mathcal{A}_{rev}}^{Revoke}(1^k)$*, which is executed by a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}_{rev}$*:*

$\boldsymbol{Exp}^{Revoke}_{\mathcal{A}_{rev}}(1^k)$

$(\widetilde{\mathcal{G}}, params) \leftarrow \mathcal{S}im_1(1^k)$

$(K_i, PK_i, SK_i) \leftarrow_{\$} \boldsymbol{KeyGen}(1^k)$

$st \leftarrow \mathcal{A}_{rev}(1^k, PP)$

$\mathcal{G} \leftarrow_{\$} \boldsymbol{Initial}(1^k, K_{\mathcal{U}_i}, PP)$

$st \leftarrow_{\$} \mathcal{A}^{\mathcal{O}}_{rev}(st)$

For $id_j \in \boldsymbol{F}_i$

$(K_{\mathcal{U}_i}, PP) \leftarrow_{\$} \boldsymbol{Revoke}(\cdot, \mathcal{G}, id_j, PK_i, F_i)$

$\tau \leftarrow_{\$} \mathcal{A}^{\mathcal{O}}_{rev}(st)$

$R_k \leftarrow_{\$} \boldsymbol{Search}(\tau, k_{\mathcal{S}_1})$

If $R_k \neq \perp$ ***return*** 1

else return 0

*Specifically, $\mathcal{C}$ runs **Initial** to initialize $\mathcal{G}$, generates key (K_i, PK_i, SK_i) and state ciphertext cst_i by **KeyGen** and Join. $\mathcal{C}$ sends $\mathcal{G}$ and cst_i to $\mathcal{A}_{rev}$. $\mathcal{A}_{rev}$ can access to the following oracles, where $\cdot$ denotes the parameters that are provided by $\mathcal{A}_{rev}$ himself:*

- *$\mathcal{O}_{Grant}(\cdot, \mathcal{G}, id_j, PK_i, F_i)$: $\mathcal{A}_{rev}$ can send grant friend request to this oracle. If $id_j \notin F_i$, then the oracle $\mathcal{O}_{\boldsymbol{Grant}}$ runs Grant by the input provides by $\mathcal{A}_{rev}$. If $id_j \in F_i$, then the oracle $\mathcal{O}_{Revoke}$ outputs $\perp$.*
- *$\mathcal{O}_{Revoke}(\cdot, \mathcal{G}, id_j, PK_i, F_i)$: $\mathcal{A}_{rev}$ can send revoke friend request to this oracle. If $id_j \in F_i$, then the oracle $\mathcal{O}_{Revoke}$ runs **Revoke** by the input provides by $\mathcal{A}_{rev}$. If $id_j \notin F_i$, then the oracle $\mathcal{O}_{Revoke}$ outputs $\perp$.*
- *$\mathcal{O}_{Search}(\cdot, \mathcal{G}, PK_s, F_s)$: $\mathcal{A}_{rev}$ can send a search request in $\mathcal{G}$ to this oracle. $\mathcal{A}_{rev}$ generates a search token and sends it to $\mathcal{O}_{Search}$. Then, the oracle $\mathcal{O}_{Search}$ runs **Search** by the input provides by $\mathcal{A}_{rev}$, and outputs the search result to $\mathcal{A}_{rev}$.*

*After polynomial times rounds of queries, $\mathcal{C}$ revokes all the users that have access to $\mathcal{O}_{Grant}(\cdot, \mathcal{G}, id_j, PK_i, F_i)$ but not $\mathcal{O}_{Revoke}(\cdot, \mathcal{G}, id_j, PK_i, F_i)$. $\mathcal{A}_{rev}$ generates a search token τ in **Search** protocol. If the output of **Search** is not $\perp$, then returns 1, otherwise returns 0.*

The proposed scheme achieves revocation security if, for all $\mathcal{A}_{rev}$, all 1^k, the advantage of $\mathcal{A}_{rev}$ to win $\boldsymbol{Exp}^{Revoke}_{\mathcal{A}}(1^k)$ is negligible:

$$|Pr[\boldsymbol{Exp}^{Revoke}_{\mathcal{A}_{rev}}(1^k) = 1]| \leq negl(1^k).$$

3.4. The Detailed Construction

Let $\mathcal{BE} = \{\mathcal{BE}.\mathcal{K}ey\mathcal{G}en, \mathcal{BE}.\mathcal{J}oin, \mathcal{BE}.\mathcal{E}nc, \mathcal{BE}.\mathcal{D}ec\}$ be a broadcast encryption scheme that retains CPA secure against a coalition of revoked users [26], $\mathcal{P} = \{\mathcal{P}.\mathcal{K}ey\mathcal{G}en, \mathcal{P}.\mathcal{E}nc, \mathcal{P}.\mathcal{D}ec\}$ be the Pallier encryption scheme [27], $\mathcal{GM} = \{\mathcal{GM}.\mathcal{K}ey\mathcal{G}en, \mathcal{GM}.\mathcal{E}nc, \mathcal{GM}.\mathcal{D}ec\}$ be the Goldwasser-Micali encryption scheme [28], and $\mathcal{F} : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^k$ be a pseudo-random function. The detailed construction is given as follows:

3.4.1. Initialization

On input of the security parameter 1^k, $\mathcal{S}_1$ initializes the global social network graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and public parameters *params*. In graph $\mathcal{G}$, the maximal number of vertexes in $\mathcal{V}$ is n, that is $|\mathcal{V}| = n$, which represents the maximum amount of enrolled users. Each vertex $v_i \in \mathcal{V}$ should be attached with the information for an enrolled user $\mathcal{U}_i \in \mathbf{U}$ that $\mathcal{S}_1$ gathered. The existence of a non-zero edge $e_{ij} \in \mathcal{E}$ between $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ represents the friends relationship of $\mathcal{U}_i$ and $\mathcal{U}_j$. In other words, if $\mathcal{U}_i$ and $\mathcal{U}_j$ are strangers to each other, then $e_{ij} = 0$. $\mathcal{G}$ is empty at initialization.

3.4.2. Key Generation

If a user $\mathcal{U}_i$ is willing to join in the system, he should generate his own keys at local. $\mathcal{U}_i$'s keys consists of the following parts: the key for the pseudo-random function $\mathcal{F}$ to encrypt location data, the key pair for the broadcast encryption scheme $\mathcal{BE}$, the key pairs for the Pallier encryption scheme $\mathcal{P}$ and the Goldwasser-Micali encryption scheme $\mathcal{GM}$. $\mathcal{U}_i$ first takes as input the binary representation of the public parameters *params*, and randomly selects a k-bit string $k_i \in \{0,1\}^k$ for his key of $\mathcal{F}$. Then he generates (bpk_i, msk_i) by $\mathcal{BE}.\mathcal{KeyGen}$, (pk_i, sk_i) by $\mathcal{P}.\mathcal{KeyGen}$ and (pk'_i, sk'_i) by $\mathcal{GM}.\mathcal{KeyGen}$. Afterwards, he forms his symmetric key K_i as (msk_i, k_i), public key PK_i as (bpk_i, pk_i, pk'_i) and secret key SK_i as (sk_i, sk'_i). The lengths of the above keys are determined by the security parameter 1^k. Finally, $\mathcal{U}_i$ publishes his public key PK_i throughout the system.

3.4.3. Join

Before joining in, $\mathcal{U}_i$ should generate his friends index F_i with d entries, where d represents the maximum amount of $\mathcal{U}_i$'s friends. F_i is a key-value data structure, which is empty at first. The key part of F_i will be attached with the granted friends' identities, the corresponding value part will be attached with the granted friends' session keys. More precisely, if $\mathcal{U}_j$ is a friend of $\mathcal{U}_i$, then $\mathrm{F}_i[id_j]$ stores the session key k^i_j that $\mathcal{U}_j$ has shared with $\mathcal{U}_i$, where id_i represents $\mathcal{U}_i$'s identity: $\mathrm{F}_i[id_j] = k^i_j$, where id_j represents $\mathcal{U}_j$'s identity. To register, $\mathcal{U}_i$ should also add the server $\mathcal{S}_1$ in F_i by generating $\mathcal{S}_1$'s session key $k^i_{\mathcal{S}_1}$ by $\mathcal{BE}.\mathcal{J}oin_{msk_i}(\mathcal{S}_1)$ and setting $\mathrm{F}_i[\mathcal{S}_1] = k^i_{\mathcal{S}_1}$. Afterwards, $\mathcal{U}_i$ randomly selects a k-bit string st_i as his current state value and encrypts st_i to cst_i by $\mathcal{BE}.\mathcal{E}nc_{bpk_i}(\mathcal{S}_1, st_i)$. Then $\mathcal{U}_i$ sends $\mathcal{S}_1$ a registration request $Re_i = (id_i||cst_i||k^i_{\mathcal{S}_1})$. $\mathcal{S}_1$ selects an empty vertex $v_i \in \mathcal{V}$ in $\mathcal{G}$ and attaches v_i with Re_i.

3.4.4. Location Update

An enrolled user $\mathcal{U}_i \in \mathbf{U}$ can interact with $\mathcal{S}_1$ to update his location by **LocUpdate** protocol. First, $\mathcal{U}_i$ obtains his current geo-location l_i from the trusted location infrastructure that sends $\mathcal{U}_i$'s geo-location to his local mobile device periodically. $\mathcal{U}_i$ maps l_i into an integer x_i from $\mathbb{Z}_k$ and computes its square ${x_i}^2$. To hide l_i from $\mathcal{S}_1$, $\mathcal{U}_i$ needs to encrypt x_i and ${x_i}^2$ at local: he chooses two random values r_1 and r_2 from $\mathbb{Z}_k$, uses his key k_i to generate $p_1 = \mathcal{F}_{k_i}(r_1)$ and $p_2 = \mathcal{F}_i(r_2)$ by pseudo-random function $\mathcal{F}$, and hides x_i and ${x_i}^2$ into $c_{x_i} = (x_i + p_1, r_1)$ and $c_{{x_i}^2} = ({x_i}^2 + p_2, r_2)$ by (p_1, p_2) and (r_1, r_2). Finally he forms his current location ciphertext L_i as $L_i = (c_{x_i}, c_{x_i^2})$ and sends L_i to $\mathcal{S}_1$. $\mathcal{S}_1$ updates the information embedded in vertex v_i in $\mathcal{G}$ as $v_i \leftarrow v_i||\{L_i\}$.

3.4.5. Grant

When $\mathcal{U}_i$ connects $\mathcal{U}_j$ as his friend, he should grant $\mathcal{U}_j$'s right to search his location by conducting **Grant** protocol with $\mathcal{S}_1$. First, $\mathcal{U}_i$ adds $\mathcal{U}_j$'s identity id_j as an entry in $\mathcal{U}_i$'s friends index F_i, generates $\mathcal{U}_j$'s session key k^i_j by $\mathcal{BE}.\mathcal{J}oin_{msk_i}(id_j)$, sends k^i_j to $\mathcal{U}_j$ in secure channel. $\mathcal{U}_i$ then selects a k-bit string st'_i as his updated state value, encrypts st'_i to cst'_i for the updated friends group in F_i that contains $\mathcal{U}_j$ by $\mathcal{BE}.\mathcal{E}nc_{(bpk_i)}(st'_i, \mathrm{F}_i)$, and boardcasts cst'_i to the system. After receiving his session key k^j_i from $\mathcal{U}_j$, he attaches $\mathrm{F}_i[id_j]$ with k^j_i: $\mathrm{F}_i[id_j] = k^j_i$. Afterwards, he sends grant request $(cst'_i||id_j)$ to $\mathcal{S}_1$. $\mathcal{S}_1$ first checks whether there is a non-zero direct edge e_{ij} in $\mathcal{G}$. If not, it sets $e_{ij} = 1$ and update v_i in $\mathcal{G}$ with new cst'_i: $v_i \leftarrow v_i \backslash \{cst_i\} \cup \{cst'_i\}$.

3.4.6. K-Nearest Neighbors Search

Each enrolled user $\mathcal{U}_s \in \mathbf{U}$ can send a search request to $\mathcal{S}_1$ for retrieving his k-nearest neighbors sorted by distances, shown in Protocol 1. First of all, $\mathcal{U}_s$ retrieves his friends' identities $\{id_1, \ldots, id_d\}$ from his friends index F_i, downloads the state ciphertexts $\{cst_1, \ldots, cst_d\}$ for all his friends $\{\mathcal{U}_1, \ldots, \mathcal{U}_d\}$ from the system. For each $cst_i \in$

$\{cst_1, \ldots, cst_d\}$, $\mathcal{U}_s$ decrypts it to st'_i by $\mathcal{BE}.\mathcal{D}ec_{k_s^i}(st'_i)$. Afterwards, $\mathcal{U}_s$ consolidates the decryption results into search token $\tau = (st'_1, \ldots, st'_d)$ and sends τ to $\mathcal{S}_1$. After receiving τ, $\mathcal{S}_1$ extracts $\{cst_1, \ldots, cst_d\}$ from v_j's all adjacents $\{v_1, \ldots, v_d\}$ in $\mathcal{G}$. For each $cst_i \in \{cst_1, \ldots, cst_d\}$, $\mathcal{S}_1$ decrypts it to st_i by $\mathcal{BE}.\mathcal{D}ec_{k_{\mathcal{S}_1}^i}(cst_i)$. It compares each st_i in $\{st_1, \ldots, st_d\}$ with st'_i in τ: if st'_j is equal to st_i, then $\mathcal{U}_s$ has been granted the right to search for $\mathcal{U}_i$'s location. Afterwards, for each granted $\mathcal{U}_i$, $\mathcal{S}_1$ retrieves the encrypted location L_i attached in v_i. Then, $\mathcal{S}_1$ and $\mathcal{S}_2$ conduct the following protocols:

1. Location re-encryption protocol $\mathcal{P}_1$:

1. $\mathcal{S}_1$: selects a random value p^* from $\{0,1\}^k$;
2. computes $c^*_{x_i} = x_i + p_1 + p^*$ to statistically hide $x_i + p_1$;
3. sends $(c^*_{x_i}, r_1)$ to $\mathcal{S}_2$.
4. $\mathcal{S}_2$: computes $c^*_{x_i} - \mathcal{F}_{k_i}(r_1)$;
5. computes $[c^*_{x_i} - \mathcal{F}_{k_i}(r_i)] \leftarrow \mathcal{P}.\mathcal{E}nc_{pk_s}(c^*_{x_i} - \mathcal{F}_{k_i}(r_i))$;
6. sends $[c^*_{x_i} - \mathcal{F}_{k_i}(r_i)]$ back to $\mathcal{S}_1$.
7. $\mathcal{S}_1$: computes $[p^*]^{-1}$;
8. computes $[x_i] = [c^*_i - \mathcal{F}_{k_i}(r_i)][p^*]^{-1}$ to remove p^* from $[c^*_{x_i} - \mathcal{F}_{k_i}(r_1)]$;
9. $\mathcal{S}_1$ and $\mathcal{S}_2$ generate $[x_i^2]$ by the same procedures in 1-8.
10. $\mathcal{S}_1$: gets $\mathcal{U}_i$'s location ciphertext $L_i^s = ([x_i], [x_i^2])$ under pk_s.

2. Encrypted distance computation protocol $\mathcal{P}_2$:

1. $\mathcal{S}_1$: selects a random value r_i from $\{0,1\}^k$;
2. generates $x'_i = [x_i] * [r_i]$;
3. sends x'_i to $\mathcal{S}_2$.
4. $\mathcal{S}_2$: decrypts $[x'_i]$ to x_i: $x'_i \leftarrow \mathcal{P}.\mathcal{D}ec_{sk_s}[x'_i])$;
5. computes $h \leftarrow h_i * h_s \ mod \ n$;
6. encrypts h under pk_s: $[h] \leftarrow \mathcal{P}.\mathcal{E}nc_{pk_s}(h)$;
7. sends $[h]$ to $\mathcal{S}_1$.
8. $\mathcal{S}_1$: computes $s = [h] * [x_i]^{n-r_s} \ mod \ n$;
9. gets $[D_i] = [x_i^2] * [n-2][x_i x_s] * [x_s^2]$.

After conducting $\mathcal{P}_1$ and $\mathcal{P}_2$ for all $\mathcal{U}_s$'s friends, $\mathcal{S}_1$ forms a key-value set $I = \{(id_1, [D_1]), \ldots, (id_d, [D_d])\}$ that contains all pairs of the encrypted distances between $\mathcal{U}_s$ and his friends along with their identities. $\mathcal{S}_1$ encrypts each $id_i \in I$ to $[id_i]$ by $\mathcal{P}.\mathcal{E}nc_{pk_s}(id_i)$, and generates $\tilde{R} = \{([id_1], [D_1]), \ldots, ([id_d], [D_d])\}$. $\mathcal{S}_1$ and $\mathcal{S}_2$ perform a secure comparison protocol $\mathcal{P}_3$ [29] for $\mathcal{S}_1$ and $\mathcal{S}_2$ to compare each pair $([id_x], [D_x])$ and $([id_y], [D_y])$ in $\tilde{R}$ based on the distance D_x and D_y. We use $\mathcal{P}_3$ as a black-box building block for **Search** protocol, and pick Batcher's sorting [30] for performing efficient parallel multi-time comparisons.

Finally, $\mathcal{S}_1$ obtains the sorted final result $R = \{([id_1], [D_1]), \ldots, ([id_d], [D_d])\}$, and sends it back to $\mathcal{U}_s$. $\mathcal{U}_s$ can decrypt each $[id_i]$ to id_i by $\mathcal{P}.\mathcal{D}ec_{sk_s}[id_i]$, then obtain his k-nearest neighbors identities $R_k = (id_1, \ldots, id_k)$ that were sorted by distance.

3.4.7. Revoke

When $\mathcal{U}_i$ wants to revoke $\mathcal{U}_j$'s search right, he should conduct **Revoke** protocol with $\mathcal{S}_1$. $\mathcal{U}_i$ first deletes $\mathrm{F}_i[id_j]$ locally, selects a k-bit string st'_i as his updated state value, encrypts st'_i to cst'_i by $\mathcal{BE}.\mathcal{E}nc_{bpk_i}(st'_i, \mathrm{F}_i)$ for the updated group in F_i that excludes $\mathcal{U}_j$. Afterwards, he sends revoke request $(cst'_i || id_j)$ to $\mathcal{S}_1$. $\mathcal{S}_1$ first checks whether there is a non-zero direct edge e_{ij} in $\mathcal{G}$. If true, it set $e_{ij} = 0$ and update v_i in $\mathcal{G}$ with new cst'_i: $v_i \leftarrow v_i \backslash \{cst_i\} \cup \{cst'_i\}$.

Protocol 1 K-Nearest Neighbors Search

$\mathcal{U}_s(K_s, \mathbf{F}_s)$ $\mathcal{S}_1(PK_s, \mathcal{G})$

1: **For** $1 \le i \le d$ **do**

2: $st'_i \leftarrow \mathcal{BE}.\mathcal{D}ec_{k_s^i}(cst_i)$

3: $\tau \leftarrow (st'_1, \ldots, st'_d)$

4: $\xrightarrow{\tau}$

Protocol 2 Secure Sort

$\mathcal{S}_1(\tau, PK_s, \mathcal{G})$ $\mathcal{S}_2(SK_s)$

1: $(st'_1, \ldots, st'_d) \leftarrow \tau$

2: **For** $1 \le i \le d$ **do**

3: $st_i \leftarrow \mathcal{BE}.\mathcal{D}ec_{k_{\mathcal{S}_1}^i}(cst_i)$

4: **If** $st'_i = st_i$

5: $L_i \leftarrow v_i$

6: $L_i^s \longleftarrow \mathcal{P}_1(\mathcal{S}_1(L_i, PK_s);\ \mathcal{S}_2(SK_s)) \longrightarrow \perp$

7: $D_i \longleftarrow \mathcal{P}_2(\mathcal{S}_1(L_i, L_i^s, PK_s);\ \mathcal{S}_2(SK_s)) \longrightarrow \perp$

8: $I \leftarrow \{(id_1, [D_1]), \ldots, (id_d, [D_d])\}$

9: **For** $1 \le i \le d$ **do**

10: $[u_i] \leftarrow \mathcal{P}.\mathcal{E}nc_{pk_s}(u_i)$

11: $\tilde{R} \leftarrow \{([id_1], [D_1]), \ldots, ([id_d], [D_d])\}$

12: $\tilde{R}^i \leftarrow \tilde{R}$

13: **For** $1 \le j \le ((\log d)^2)$**do**

14: $\tilde{R}^{j+1} \leftarrow \tilde{R}^j$

15: **For** $1 \le i \le d$ **do**

16: $(([id_x]^i, [D_x]^i), ([id_y]^i, [D_y]^i)) \leftarrow P_i$

17: $P_{i+1} \longleftarrow \mathcal{P}_3(\mathcal{S}_1(P_i.next), PK_s);\mathcal{S}_2(SK_s)) \longrightarrow \perp$

18: $R \leftarrow \tilde{R}$

5: $\xleftarrow{R}$

$\mathcal{U}_s$ **output:**R $\mathcal{S}_1$ **output:**$\perp$

4. Security Analysis

4.1. Adaptive $\mathcal{L}$-Semantic Secure

Theorem 1. *If $\mathcal{F}$ is a pseudo-random function, $\mathcal{P}$, $\mathcal{GM}$ and $\mathcal{BE}$ are CPA secure, and the DGK protocol [31] is proved to be semantically secure in the random oracle model, then the proposed scheme satisfies adaptive $\mathcal{L}$-semantic security, which is defined in Definition 2.*

Proof. We construct two simulators Sim_1, Sim_2 that can generate the simulated values in Ideal(1^k) using the information given in the leakage functions $\mathcal{L}_1$ to $\mathcal{L}_4$, and prove that Ideal(1^k) is indistinguishable with Real(1^k) by any PPT adversary.

Given the information leaked from $\mathcal{L}_1$, Sim_1 can learn $|cst_i^j|$ and $\{|cst_i^1|, \ldots, |cst_i^q|\}$. Afterwords, it can choose random value $\widetilde{cst}_i^j$ with lengths $|cst_i^j|$ to simulate cst_i^j. Due to the CPA secure of $\mathcal{BE}$, cst_i^j is indistinguishable from $\widetilde{cst}_i^j$ by any PPT adversary. Therefore, Sim_1 cannot learn extra information from $\{|cst_i^1|, \ldots, |cst_i^q|\}$, which satisfies:

$$\text{Output}_{\mathcal{A}_1}^{\text{Real}}(\{cst_i^1, \ldots, cst_i^q\}) \approx \text{Output}_{Sim_1}^{\text{Ideal}}(\{\widetilde{cst}_i^1, \ldots, \widetilde{cst}_i^q\}).$$

Given the information leaked from $\mathcal{L}_2$, Sim_1 can learn $|c_{x_i}^j|$ and $|c_{x_i^2}^j|$ in $L_i^j = (c_{x_i}^j, c_{x_i^2}^j)$. Afterwards, it can choose two random values in lengths $|c_{x_i}^j|$ and $|c_{x_i^2}^j|$ to output the

simulated $\tilde{L}_i^j = (\tilde{c}_{x_i}^j, \tilde{c}_{x_i^2}^j)$. Since L_i^j is generated by $\mathcal{F}$, $\tilde{L}_i^j$ and L_i^j are indistinguishable by any PPT adversary due to the randomness of $\mathcal{F}$. Therefore, Sim_1 cannot learn extra information from the update history $\{L_i^1, \ldots, L_i^q\}$, which satisfies:

$$\text{Output}_{\mathcal{A}_1}^{\text{Real}}(\{L_i^1, \ldots, L_i^q\}) \approx \text{Output}_{Sim_1}^{\text{Ideal}}(\{\tilde{L}_i^1, \ldots, \tilde{L}_i^q\}).$$

Given the information leaked from $\mathcal{L}_3$, Sim_1 can obtain search tokens $\{\tau_1, \ldots, \tau_q\}$. Afterwards, it can choose random value $\tilde{\tau}_i$ in size $|\tau_i|$ to simulate each τ_i. Moreover, since $\{st_1', \ldots, st_d')$ is generated by $\mathcal{BE}.\mathcal{D}ec$ by decrypting $\{cst_1, \ldots, cst_d\}$ using keys $\{k_s^1, \ldots, k_s^d\}$), and each k_s^i in $\{k_s^1, \ldots, k_s^d\}$ is a k-bit random string, each st_j' in τ_i is indistinguishable from $\tilde{\tau}_i$ by any PPT adversary. Therefore, Sim_1 cannot learn extra information from $\tau_1, \ldots, \tau_q$, which satisfies:

$$\text{Output}_{\mathcal{A}_1}^{\text{Real}}(\{\tau_1, \ldots, \tau_q\}) \approx \text{Output}_{Sim_1}^{\text{Ideal}}(\{\tilde{\tau}_1, \ldots, \tilde{\tau}_q\}).$$

The sorting network between $\mathcal{A}_1$ and $\mathcal{A}_2$ contains $(\log d)^2$ levels, and each level contains $(\log d)^2$ times of $\mathcal{P}_3$ protocols. Therefore, the simulation of the sorting network can be reduced to prove Sim_1 and Sim_2 can simulate the secure comparison protocol $\mathcal{P}_1$ with leakage functions. Given the information leaked from $\mathcal{L}_4$, Sim_2 can learn the leaked information $([\![z_i]\!], [\lambda_i])$ from each round of $\mathcal{P}_3$ and the rounds number $(\log d)^2$. In each round, Sim_2 can learn $([D_x], [D_y], l)$. Sim_1 and Sim_2 should simulate $\mathcal{A}_1$ and $\mathcal{A}_2$ with $\mathcal{L}_4$ by all pairs with $(\log d)^2$ times in sorting protocol to get the final simulation value. At every pairs i, $\mathcal{A}_1$'s view can be denoted as $view_{\mathcal{A}_1} = (sk_s, [\![z]\!], \|\lambda\|)$. Given $(sk_s, [\![z]\!], [\lambda])$, we can build Sim_1 in the following phases:

- Randomly choose $\tilde{\lambda}$, compute $||\tilde{\lambda}||$ as $x_x \leq x_y$;
- Randomly choose $\tilde{z} \leftarrow (0, 2^{\lambda+l}) \bigcup Z$;
- Encrypt $\tilde{z}$: $[\tilde{z}] \leftarrow \mathcal{P}.\mathcal{E}nc_{pk_s}(z)$;
- Output $view_{Sim_1} = (sk_s, l, [\tilde{z}], ||\tilde{\lambda}||)$.

Since $z = x + r$, where x is a l-bits integer and r is a l+λ-bits integer, the distribution of $\tilde{z}$ is indistinguishable from z. We can get $(sk_s, [\tilde{z}]) \approx (sk_s, [z])$. Besides, since the distribution of $\tilde{z}$ and z are independent of t, we can get $(sk_s, l, [\tilde{z}]||\tilde{\lambda}||) \approx (sk_s, l, [z], ||\tilde{\lambda}||)$. In a similar way, at every pairs i, $\mathcal{A}_2$'s view can be denoted as $view_{\mathcal{A}_2} = (([D_x]_i, [D_y]_i, l, pk_s, r, \|\lambda\|, [z_l])$. We can build Sim_2 to simulate $\mathcal{A}_2$ in the following phases:

- Choose $\tilde{r} \leftarrow (0, 2^{\lambda+l}) \bigcup Z$;
- Choose two random values $\tilde{\lambda}, \tilde{z}_l$, computes $||\tilde{\lambda}||, \|\tilde{z}\|$;
- Output $view_{Sim_2} = ([D_x], [D_y], l, pk_s, \tilde{r}, [\tilde{z}_l])$.

In both $view_{\mathcal{A}_2}$ and $view_{Sim_2}$, r is extracted from uniform distribution $(0, 2^{\lambda+l}) \bigcup \mathbb{Z}$, $[\tilde{z}_l]$ is the ciphertext of $\mathcal{P}$ which is randomness, so $([D_x], [D_y], l, pk_s) \approx ([D_x], [D_y], l, pk_s, r, [\tilde{z}_l])$. We can obtain: $view_{\mathcal{A}_2}$ and $view_{Sim_2}$ are computational indistinguishable. What is more, since $||\tilde{\lambda}|| \approx \|[D_x] \leq [D_y]\|$, $(sk_s, l, [z], ||\tilde{\lambda}||) \approx (sk_s, l, [z], \|[x_x] \leq [y_y]\|)$. Due to the semantic security of DGK, Sim_1 and Sim_2 can obtain d ciphertexts that are unsorted from the leakage function $\mathcal{L}_4$. Then, Sim_1 and Sim_2 can simulate Bathcer's sorting protocols in $(\log d)^2$ times.

Therefore, for all polynomial time $\mathcal{A}_1$ and $\mathcal{A}_2$, there exists polynomial time simulators Sim_1 and Sim_2 such that:

We can demonstrate that the proposed scheme satisfies adaptive $\mathcal{L}$-semantic security in the random oracle model, which is defined in Definition 2. Theorem 1 proved. □

4.2. Revocation Secure

Theorem 2. *If $\mathcal{BE}$ is CPA secure, then the proposed scheme satisfies revocation secure, which is defined in Definition 3.*

Proof. Assuming the advantage of $\mathcal{A}_{rev}$ to win $\mathbf{Exp}^{\text{Revoke}}_{\mathcal{A}_{rev}}(1^k)$ is negligible, we can construct an adversary $\mathcal{A}_{be}$, who can break the CPA secure of $\mathcal{BE}$ with assist of $\mathcal{A}_{rev}$. We will show that if $\mathcal{A}_{rev}$ has a non-negligible advantage in $\mathbf{Exp}^{\text{Revoke}}_{\mathcal{A}_{rev}}(1^k)$, then we can construct an adversary $\mathcal{A}_{be}$ that uses $\mathcal{A}_{rev}$ as a subroutine to break the CPA secure of $\mathcal{BE}$.

To make the output of $\mathbf{Exp}^{\text{Revoke}}_{\mathcal{A}_{rev}}(1^k)$ as 1, $\mathcal{A}_{rev}$ needs to provide a valid search token. To achieve that, $\mathcal{A}_{rev}$ must know st_i. A new value of st_i is randomly selected and encrypted by $\mathcal{BE}.\mathcal{E}nc_{bpk_i}(st_i, \mathrm{F}_i \backslash u_j)$ at each time a user is revoked from the system, where $\mathrm{F}_i \backslash u_j$ is the new friends index. $\mathcal{A}_{rev}$ then broadcast this encrypted value to all users. $\mathcal{BE}$'s security ensures that only a non-revoked friend of $\mathcal{U}_i$ can decrypt this ciphertext to obtain st_i with overwhelming probability. Hence, the adversary can only create a valid search token if he is a valid friend of $\mathcal{U}_i$, or he will break the security of $\mathcal{BE}$. That is, the probability that a random bit string is valid is 2^{-k}. It means that the adversary will not be able to produce a valid token with non-negligible probability.

Let $\mathcal{C}$ be the challenger for the adversary $\mathcal{A}_{be}$ against $\mathcal{BE}$, $\mathcal{A}_{be}$ will act as the challenger for $\mathcal{A}_{rev}$:

1. $\mathcal{C}$ runs $\mathcal{BE}.\mathcal{K}ey\mathcal{G}en(1^k)$ to generate keys (msk_{be}, bpk_i). $\mathcal{A}_{be}$ initializes F_i, randomly chooses a k-bit string st_i, and sends (st_i, F_i) to $\mathcal{C}$. $\mathcal{C}$ runs $\mathcal{BE}.\mathcal{E}\mathrm{nc}_{bpk_i}(st_i, \mathrm{F}_i)$ to generate $st_{\mathcal{S}_1}$, and sends it to $\mathcal{A}_{be}$. $\mathcal{A}_{be}$ runs **KeyGen** to generate K_i, runs **Join** to generate $k^i_{\mathcal{S}_1}$, where K_i does not include k_{be}.
2. $\mathcal{A}_{be}$ issues a query to $\mathcal{C}$ for the secret key of $\mathcal{A}_{rev}$. $\mathcal{C}$ runs $\mathcal{BE}.\mathcal{J}\mathrm{oin}_{msk_i}(\mathcal{A}_{rev})$ to generate $k_{\mathcal{A}_{rev}}$, sends $k_{\mathcal{A}_{rev}}$ to $\mathcal{A}_{be}$. To fully enroll $\mathcal{A}_{rev}$ as a valid friend, the state ciphertext also needs to be updated by $\mathcal{A}_{be}$. $\mathcal{A}_{be}$ send F_i and a newly generated st_i to $\mathcal{C}$, $\mathcal{C}$ runs $\mathcal{BE}.\mathcal{E}nc_{bpk_i}(st_i, \mathrm{F}_i)$ to generate new cst_i. $\mathcal{A}_{be}$ runs **Grant** to generate the key $k^i_{\mathcal{A}_{rev}}$ of $\mathcal{A}_{rev}$.
3. $\mathcal{A}_{be}$ runs **Initial** to generate graph $\mathcal{G}$, and sends $k^i_{\mathcal{A}_{rev}}$ and $\mathcal{G}$ to $\mathcal{A}_{rev}$. $\mathcal{A}_{rev}$ can access to oracles $\mathcal{O}_{\text{Grant}}$ and $\mathcal{O}_{\text{Revoke}}$.
4. $\mathcal{A}_{be}$ revokes $\mathcal{A}_{rev}$ by running **Revoke**, $\mathcal{A}_{be}$ runs **Revoke** a second time in order to produce two values $st_{i0} \leftarrow \{0,1\}^k$ and $st_{i1} \leftarrow \{0,1\}^k$ for st_i, and sends st_{i0} and st_{i1} to $\mathcal{C}$ as the challenge value for $\mathcal{A}_{be}$, along with a set of no revoked friends F_i of $\mathcal{A}_{rev}$.
5. $\mathcal{C}$ selects a bit $b \in \{0,1\}$, uses $\mathcal{BE}.\mathcal{E}nc_{bpk_i}(st_{ib}, \mathrm{F}_i)$ to encrypt st_{ib} and generates cst_{ib}, sends cst_{ib} to $\mathcal{A}_{be}$ as the challenge ciphertext for the CPA secure of $\mathcal{BE}$. $\mathcal{A}_{be}$ sends cst_{ib} to $\mathcal{A}_{rev}$ as the challenge ciphertext of $\mathbf{Exp}^{\text{Revoke}}_{\mathcal{A}_{rev}}(1^k)$.
6. $\mathcal{A}_{rev}$ generates token τ, and sends τ to $\mathcal{A}_{be}$. Since the advantage for $\mathcal{A}_{rev}$ to win $\mathbf{Exp}^{\text{Revoke}}_{\mathcal{A}_{rev}}(1^k)$ is non-negligible, the probability of validity of τ is non-negligible.
7. If $t_0 \neq \perp$, then **Search** stops. According to the following situations, $\mathcal{A}_{be}$ outputs its guess for b:
   - If $t_0 \neq \perp$, this tells $\mathcal{A}_{be}$ that st_{i0} was used to generate the token, $\mathcal{A}_{be}$ outputs its guess for b as $b' = 0$;
   - Of $t_1 \neq \perp$, this tells $\mathcal{A}_{be}$ that $st_{i1}$was used to generate the token, $\mathcal{A}_{be}$ outputs its guess for b as $b' = 1$.

From the above analysis, the advantage of $\mathcal{A}_{be}$ to break the CPA secure of $\mathcal{BE}$ can be computed as $\mathrm{Adv}^{\mathcal{BE}}_{\mathcal{A}_{be}}(1^k)$:

$$\begin{aligned}\mathrm{Adv}^{\mathcal{BE}}_{\mathcal{A}_{be}}(1^k) &= |[(\Pr[(t_0 \vee t_1) \neq \perp] \cdot 1 - \frac{1}{2}) + (\Pr[((t_0 \wedge t_1) \neq \perp] \cdot \frac{1}{2} - \frac{1}{2})| \\ &= |\delta \cdot 1 + (1-\delta) \cdot \frac{1}{2} - \frac{1}{2}| \\ &= |\frac{(\delta+1)}{2} - \frac{1}{2}| \\ &= \frac{\delta}{2}.\end{aligned}$$

Since the advantage δ of $\mathcal{A}_{rev}$ to win $\mathbf{Exp}^{\mathrm{Revoke}}_{\mathcal{A}_{rev}}(1^k)$ is non-negligible, the advantage $\frac{\delta}{2}$ of $\mathcal{A}_{be}$ to break the CPA security of $\mathcal{BE}$ is non-negligible, which contradicts the CPA security of $\mathcal{BE}$. Therefore, there exists no $\mathcal{A}_{rev}$, who can win $\mathbf{Exp}^{\mathrm{Revoke}}_{\mathcal{A}_{rev}}(1^k)$ with non-negligible probability, and the proposed scheme satisfies revocation security as defined in Definition 3. Theorem 2 proved. □

5. Theoretical Analysis

The complexity analysis is shown in Table 1, where n is the maximum amount of enrolled users and d is the maximum amount of each user's friends. We compare our scheme with the related privacy-preserving location-based query schemes [15,18,20] in Table 2. Due to the significant differences among the existing schemes in application scenarios, secure models, evaluation indicators and other factors, we focus on comparing characteristics and security.

Table 1. Complexity analysis.

	$Stor_u$	$Stor_{\mathcal{S}_1}$	$Stor_{\mathcal{S}_2}$	$Comp_u$	$Comp_{\mathcal{S}_1}$	$Comp_{\mathcal{S}_2}$	$Comm_u$	$Comm_{\mathcal{S}_1}$	$Comm_{\mathcal{S}_2}$
Register	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	–	–	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Grant	$\mathcal{O}(d)$	$\mathcal{O}(nd)$	–	$\mathcal{O}(d)$	–	–	$\mathcal{O}(d)$	$\mathcal{O}(nd)$	–
Revoke	–	–	–	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–	$\mathcal{O}(1)$	$\mathcal{O}(1)$	–
LocUpdate	–	$\mathcal{O}(n)$	–	$\mathcal{O}(1)$	–	–	$\mathcal{O}(1)$	$\mathcal{O}(n)$	–
Search	–	–	–	$\mathcal{O}(d)$	$\mathcal{O}(d+(\log d)^2)$	$\mathcal{O}(d+(\log d)^2)$	$\mathcal{O}(k)$	$\mathcal{O}((\log d)^2)$	$\mathcal{O}((\log d)^2)$

Stor: storage complexity; *Comp*: computation complexity; *Comm*: communication complexity.

Table 2. Properties comparison.

	Accuracy	Evaluation Method	Dynamic	Cryptography tool	SP	LP	AC	Rank Model
[16]	✓	Euclidean distance/Anchor points	×	PIR/$\mathcal{P}$	✓	✓	×	–
[17]	×	Linear Programming	✓	HMAC	✓	✓	×	–
[21]	×	Dynamic Grid	×	HMAC	✓	×	×	User
Ours	✓	Squared Euclidean distance	✓	$\mathcal{P}/\mathcal{GM}/\mathcal{BE}$	✓	✓	✓	2 servers

SP: Search Privacy; LP: Location Privacy; AC: Access Control.

For result accuracy, [15] achieves differential privacy for location information using linear programming techniques. It is specifically designed for simple computation that cannot provide accurate encrypted distance sorting. Ref. [20] uses a dynamic location grid structure to cluster users close to each other. However, the search results in [15,20] have a specific rate of false positives, which are suitable for similarity search. Our scheme and [18] use Euclidean distance to calculate the encrypted distance to achieve precise secure sorting. Ref. [18] focuses on searching the number of points of interest in a specific location area; our scheme sorts the distances based on the proven-secure comparison protocol. In terms of security, Ref. [18] protects location search privacy by way of private information retrieval (PIR). Although it adopts the anchor technology to improve search efficiency, it still has a certain communication overhead. Ref. [20] achieves sort privacy by assuming the server

only performs the search, and the user performs the result sorting. As a result, the above methods each sort privacy but lead to high computation or communication costs.

Besides, compared with other schemes, our scheme also has a flexible access control mechanism. Moreover, our scheme achieves a constant-time computation cost and communication cost when updating friends and encrypting locations, and a user only needs to store key-related information locally. Therefore, we can demonstrate that our proposed scheme has both a very light user workload and a moderate server workload while being secure against the honest-but-curious adversary. In nowadays's mobile social networking environment, the user-side lightweight device's storage and computation cost should be minimized as much as possible. As a consequence, the proposed scheme is more suitable for the real-life thin clients MSNs deployment scenario.

6. Implementation

We implement and analyze the performance of our scheme. The experiments were run on several computers with Linux Ubuntu 18.04.2 64-Bit Version with Inter(R) Core(TM) I7-2600 quad-core processor (3.4 ghz) and 8 GB memory, which were installed on VMware Workstation in the LAN in C++ language. One of the computers acted as the server-end and the others acted as user-ends, respectively. We implemented a job allocation mechanism in the server-end that the computer acted as the master server and used threads to simulate the collaborated server that performed the assistant job. Each user-end stored the user's keys locally and interacted with the server-end. To submit a search request, a user-end only communicated with the master server.

In the simulation experiments, the security parameter k was set to 256 bits. We chose SHA256 in the OpenSSL library [32] for the pseudo-randomness function, and used the Relic library [33] to implement Paillier and GM homomorphic encryption. To implement the scheme more securely, we improved the modulus n of the Paillier and GM to 1024 bits. Besides, we used BGW2 [26] to implement public-key broadcast encryption. The key length in the above public encryption methods was set to be 1024 bits.

We conducted data simulations based on real-world data sets, which came from the newest version of the Enron email dataset [34], where we randomly selected 1000 accounts as the total users set. We represented users' friendships in the form of linked contacts. We selected a random integer in $(10, 50)$ to simulate the user's location' value, which was updated periodically. Moreover, we initialized the social network graph structure $\mathcal{G}$ with 1000 vertexes and 3831 edges that contained the above data and used a unique value to identify each vertex (user) in $\mathbb{Z}_k$. We did not record the network communication time during all the experiments since it depends on the user-end and the server-end's network connection. Each data point in the experiments was obtained after being repeated 50 times to generate the average value.

6.1. Storage Analysis

We first analyzed the storage overhead of our scheme. Table 3 shows the comparison between the encrypted $\mathcal{G}$ and unencrypted $\mathcal{G}$ of the generation time and the server's storage cost in the trend of the number of users increases. It can be seen that the server's storage cost increased almost linearly with the increase of the number of users. Since we used symmetric encryption to encrypt location, compared with the Paillier homomorphism ciphertext, the inflation rate of the symmetric ciphertext of the location decreased significantly, which is consistent with the theoretical analysis. Therefore, the proposed scheme achieves the trade-off of users' location confidentiality and search privacy with the acceptable additional storage cost.

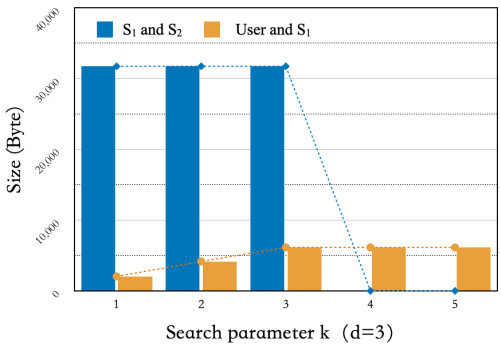
Table 3. Storage cost.

	Unencrypted $\mathcal{G}$		Encrypted $\mathcal{G}$		
Vertex	**Storage (kb)**	**GenTime (s)**	**Storage (kb)**	**GenTime (s)**	**Inflation Rate**
200	18.752	0.423	57.506	2.359	306.665%
400	38.101	0.477	115.302	2.941	302.622%
600	57.460	0.514	174.379	3.316	303.478%
800	74.677	0.538	225.039	3.770	301.349%
1000	95.988	0.575	289.864	4.113	301.979%

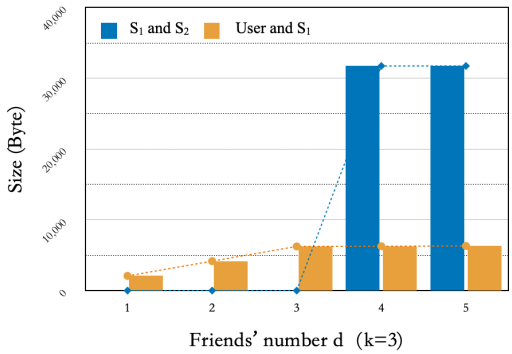
6.2. Communication

In terms of communication, we mainly analyzed the amount of data transformed between (1) $\mathcal{U}_i$ and $\mathcal{S}_1$ and (2) $\mathcal{S}_1$ and $\mathcal{S}_2$ in **Search** protocol. Theoretically, when $\mathcal{U}_i$ requests to search k-nearest neighbors among his d friends, $\mathcal{U}_i$'s communication overhead increases almost linearly with k. When $\mathcal{S}_1$ and $\mathcal{S}_2$ interact with each other to compute the distance from the total of d friends' location ciphertexts, the data size of the communication between them is $\mathcal{O}((\log d)^2)$.

Figure 2a,b shows the relationship between the two types of communication overhead in the experiment with the increasing trends of the friends' number d and the search parameter k, respectively. In general, the amount of data transmission required by the user in **Search** protocol is positively related to k. When k increases to a particular value (greater than d), the data transmission volume tends to be stable. The communication overhead between $\mathcal{S}_1$ and $\mathcal{S}_2$ is mainly positively related to d, but independent of the increase of k. Moreover, the distance computation sub-procedure requires several rounds of interactions, so the amount of communication overhead between servers is relatively large, which is consistent with the theoretical analysis.



(a) Communication overhead affected by search parameter k



(b) Communication overhead affected by friends' number k

Figure 2. Communication overhead.

6.3. Search Time

We also analyzed the primary source of the search time overhead for Search protocol. First, we divided the Search protocol at the server-end into two sub-procedures of location search and distance sort. Figure 3 shows the relationship between search time and the number of friends d. In Figure 3, the total time overhead of Search protocol is shown in the blue curve, the time overhead to extract and re-encrypt location ciphertext is shown in the yellow curve, and the time overhead to compute and sort the encrypted distance is shown in the red curve.

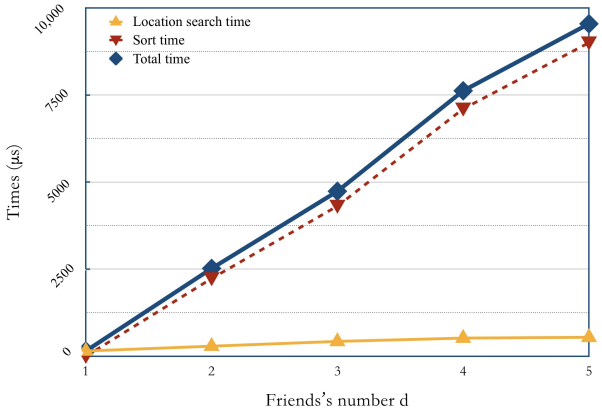


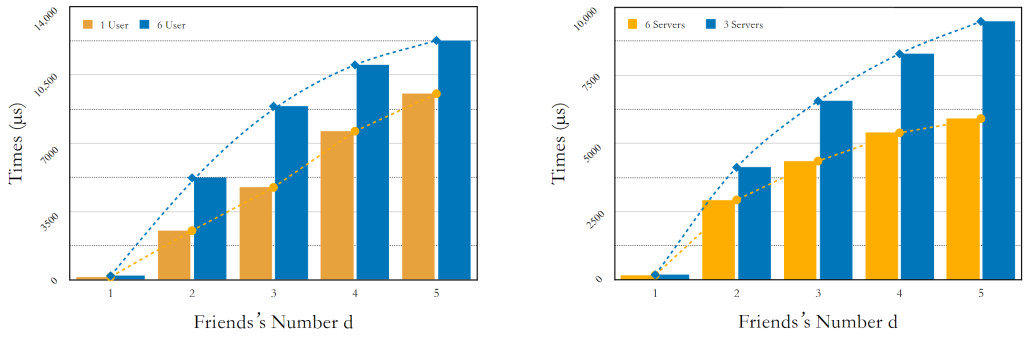
Figure 3. Search time overhead.

From Figure 3, we can see that the time overhead of the two sub-procedures in the Search protocol generally increases with the increasing trend of d. Specifically, the location search time is far lower than the distance sort time, and with d increases to 4, the curve growth is slowing down. The distance sort time has a stable approximate linear relation with d. Therefore, it can be concluded that the computation and comparison of encrypted distances are two primary time-overhead sources of the Search protocol, which is consistent with the theoretical analysis.

6.4. Scalability

In terms of scalability, we first analyzed the impact of the search users' number who submit search requests in parallel on the time overhead of the Search protocol. To be specific, we deploy one host to simulate one user to execute the Search protocol and record the total time overhead. Then we deploy six hosts to simulate six users to repeat the same experiment and compare the results. It is worth mentioning that, when recording the time of multi-user search, multiple user-ends simultaneously send the search requests to the server-end. We record the start and end time when the server-end receives the search request until it completes each user's search. Figure 4 shows the relationship between the parallel search users' number and Search protocol's total time. It can be seen that one user's search time is slightly lower than six parallel users' search times. The former is approximately in a stable linear relation with d, and the latter slows down to a constant level with the increase of d. From the trend it can be concluded that, with the number of search users d increasing, its impact on search time overhead is weakened, and it further weakens the influence of the increasing number of friends on the search time. Therefore, the multi-user parallelism has a weak impact on search time overhead, which helps the scheme to achieve a certain level of scalability.

Besides, we analyzed the influence of the expandable number of remote servers on the search time overhead. First, we deployed three servers to execute the Search protocol for six users simultaneously and recorded the total time overhead. Then we deployed six servers to repeat the same experiment and compare the results. Figure 4 shows the relationship between the number of servers and the search time. It can be seen that the search time of 6-server deployment is significantly lower than the running time with 3-server deployment, and the former's growth was slowed down to a constant level after d reaches 4, but the latter's growth takes an approximately linear relationship with the number of friends steadily. Therefore, it can be concluded that deploying multiple servers to perform parallel searches can reduce the search time overhead and further weaken the influence of the increasing number of friends on the search time.



(a) Search time affected by the querying users' number (b) Search time affected by the servers' number

Figure 4. Scalability.

Remark 1. *It is worth pointing out that the search process's main computation cost is the homomorphic encryption/decryption operation and broadcast decryption operation. The computation efficiency is closely related to the selected parameters of the underlying algorithms. The server-end implementation can also be optimized to reduce the search time by using multiple threads for distance sorting and using approximate sorting algorithms, and so forth. In our experiment, we did not adopt any optimization method. The server was allowed to complete all the computation steps in a single thread in each phase to reflect the scheme's original execution efficiency faithfully.*

7. Conclusions

Aiming at the problem of location privacy disclosure in MSNs, we propose a privacy-enhancing k-nearest neighbors search scheme over MSNs. We deploy a dual-server collaborative architecture and design an encrypted location-oriented k-neighbor search protocol based on secure multi-party computation and homomorphic encryption. Our scheme achieves accurate nearby friends retrieval while protecting the geo-location and the distance order from revealing them to the servers. We propose a lightweight dynamic friends management mechanism based on public-key broadcast encryption to satisfy the fine-grained access control requirement. It enables users to grant/revoke a friend's location search right without updating others' keys and achieves constant-time identity authentication. The scheme satisfies adaptive $\mathcal{L}$-semantic security and revocation security under the random oracle model. Compared with the works on single server architecture, the proposed scheme reduces the communication cost between users and the server and prevents location information leakage, which achieves a trade-off of the location availability and privacy.

Author Contributions: Conceptualization, Y.L. and F.Z.; Data curation, Y.L.; Formal analysis, Y.L.; Funding acquisition, F.Z.; Methodology, Y.L.; Project administration, F.Z.; Validation, Y.G. and Z.X.; Writing–original draft, Y.L.; Writing–review & editing, Y.L., Y.G. and Z.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Northeastern University Annual Basic Scientific Research Funding under Grant 02190022121006 and the Natural Science Foundation of China under Grant 61772127, Grant 61532007 and Grant 61472184.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Global Social Media Stats. Available online: https://datareportal.com/social-media-users (accessed on 10 April 2021).
2. Weichbroth, P.; Łysik, Ł. Mobile Security: Threats and Best Practices. *Mob. Inf. Syst.* **2020**. [CrossRef]
3. Anastasios, N.; Salvatore, S.; Mascolo, C.; Pontil, M. An empirical study of geographic user activity patterns in foursquare. In Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, Barcelona, Spain, 17–21 July 2011.
4. Cheng, Z.; Caverlee, J.; Lee, K.; Sui, D. Exploring millions of footprints in location sharing services. In Proceedings of the International Conference on Weblogs and Social Media, Barcelona, Spain, 17–21 July 2011; pp. 81–88.

5. Preotiuc, P.D.; Cohn, T. Mining user behaviors: A study of check-in patterns in location based social network. In Proceedings of the Conference on ACM Web Science, Paris, France, 2–4 May 2013; pp. 306–315.
6. Chor, B.; Goldreich, O.; Kushilevitz, E.; Sudan, M. Private information retrieval. In Proceedings of the IEEE 36th Annual Foundations of Computer Science, Milwaukee, Wisconsin, 23–25 October 1995; pp. 41–50.
7. Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. Comput. Secur.* **2016**, *19*, 895–934. [CrossRef]
8. Lucas, M.M.; Nikita, B. Flybynight: Mitigating the privacy risks of social networking. In Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society, Alexandria, VA, USA, 27 October 2008; pp. 1–8.
9. Guha, S.; Kevin, T.; Paul, F. NOYB: Privacy in online social networks. In Proceedings of the First Workshop on Online Social Networks, Seattle, WA, USA, 18 August 2008; pp. 49–54.
10. Niu, B.; Li, X.; Zhu, X.; Li, X.; Li, H. Are you really my friend? Exactly spatiotemporal matching scheme in Privacy-Preserving mobile social networks. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Beijing, China, 24–26 September 2014; pp. 33–40.
11. Zhang, R.; Zhang, Y.; Sun, J.; Yan, G. Fine-grained private matching for proximity-based mobile social networking. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 1969–1977.
12. Fu, Y.; Wang, Y. BCE: A privacy-preserving common-friend estimation method for distributed online social networks without cryptography. In Proceedings of the 7th International Conference on Communications and Networking, Kunming, China, 8–10 August 2012; pp. 212–217.
13. Sun, J.; Zhang, R.; Zhang, Y. Privacy-preserving spatiotemporal matching. In Proceedings of the IEEE INFOCOM, Turin, Italy, 14–19 April 2013; pp. 800–808.
14. Bamba, B.; Liu, L.; Pesti, P.; Wang, T. Supporting anonymous location queries in mobile environments with PrivacyGrid. In Proceedings of the 17th international conference on World Wide Web, New York, NY, USA, 21–25 April 2008; pp. 237–246.
15. Bordenabe, N.E.; Chatzikokolakis, K.; Palamidessi, C. Optimal Geo-Indistinguishable Mechanisms for Location Privacy. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14), Scottsdale, AZ, USA, 3–7 November 2014; pp. 251–262.
16. Elaine, S.; Richard, C.; Hubert, C. Privacy-preserving aggregation of time-series data. In Proceedings of the Network and Distributed System Security Symposium (NDSS 2011), San Diego, CA, USA, 6–9 February 2011; pp. 1–17.
17. Jorgensen, Z.; Yu, T.; Cormode, G. Publishing attributed social graphs with formal privacy guarantees. In Proceedings of the 2016 International Conference on Management of Data, San Francisco, CA, USA, 26 June–1 July 2016; pp. 107–122.
18. Zhou, C.L.; Chen, Y.H.; Tian, H.; Cai, S.B. Location Privacy and Query Privacy Preserving Method for K-nearest Neighbor Query in Road Networks. *J. Softw.* **2020**, *31*, 471–492. [CrossRef]
19. Li, Z.; Wang, C.; Yang, S.; Jiang, C.; Li, X. Lass: Local-activity and social-similarity based data forwarding in mobile social networks. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 174–184. [CrossRef]
20. Schlegel, R.; Chow, C.; Huang, Q.; Wong, D. User-defined privacy grid system for continuous location-based services. *IEEE Trans. Mob. Comput.* **2015**, *14*, 2158–2172. [CrossRef]
21. Han, M.; Li, L.; Xie, Y.; Wang, J.; Duan, Z.; Li, J.; Yan, M. Cognitive approach for location privacy protection. *IEEE Access* **2018**, *6*, 13466–13477. [CrossRef]
22. Siddula, M.; Li, Y.; Cheng, X.; Tian, Z.; Cai, Z. Privacy-enhancing preferential lbs query for mobile social network users. *Wirel. Commun. Mob. Comput.* **2020**. [CrossRef]
23. Yang, X.; Yang, M.; Yang, P.; Leng, Q. A multi-authority attribute-based encryption access control for social network. In Proceedings of the 2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE), Beijing, China, 17–19 August 2017; pp. 671–674.
24. Luo, E.; Liu, Q.; Wang, G. Hierarchical multi-authority and attribute-based encryption friend discovery scheme in mobile social networks. *IEEE Commun. Lett.* **2016**, *20*, 1772–1775. [CrossRef]
25. Alanwar, A.; Shoukry, Y.; Chakraborty, S.; Martin, P.; Tabuada, P.; Srivastava, M. PrOLoc: Resilient localization with private observers using partial homomorphic encryption. In Proceedings of the 2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Pittsburgh, PA, USA, 18–21 April 2017; pp. 41–52.
26. Boneh, D.; Craig, G.; Brent, W. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Proceedings of the Advances in Cryptology—CRYPTO 2005, Santa Barbara, CA, USA, 14–18 August 2005; pp. 258–275.
27. 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; pp. 223–238.
28. Goldwasser, S.; Micali, S. Probabilistic Encryption. *J. Comput. Syst. Sci.* **1984**, *28*, 270–299. [CrossRef]
29. Li, Y.; Zhou, F.; Xu, Z. PPFQ: Privacy-Preserving Friends Query over Online Social Networks. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December–1 January 2021; pp. 1348–1353.
30. Batcher, K.E. Sorting networks and their applications. In Proceedings of the spring joint computer conference, New York, NY, USA, 30 April–2 May 1968; pp. 307–314.
31. Veugen, T. Improving the DGK comparison protocol. In Proceedings of the International Workshop on Information Forensics and Security, Tenerife, Spain, 2 December 2012; pp. 49–54.

32. The OpenSSL Project. OpenSSL: The Open Source Toolkit for SSL/TLS. 2015. Available online: http://www.openssl.org/ (accessed on 26 May 2021).
33. Relic-Toolkit. Available online: https://github.com/relic-toolkit (accessed on 26 May 2021).
34. Cohen, W.W. Enron Email Dataset. 2015. Available online: https://www.cs.cmu.edu/~enron/ (accessed on 26 May 2021).