Efficient and Scalable Multi-party Privacy-Preserving k-NN Classification

Xinglei Li and Haifeng Qian(✉)

East China Normal University, Shanghai, China
hfqian@admin.ecnu.edu.cn

Abstract. In recent years, storing data and mining valuable information among multi-party with the help of cloud servers has become popular. However, outsourcing sensitive information and potential information leakage during the process are severe issues. Additionally, the majority of existing privacy-preserving techniques can only be applied to the single-database scene, and as a result of the use of intricate homomorphic encryption, their overall efficiency is quite poor. In this paper, we proposed a Multi-party Privacy-Preserving k-Nearest-Neighbors (MPPkNN) classification scheme based on Multi-key Symmetric Homomorphic Encryption (MSHE). In the specific protocol design process, we innovatively apply the homomorphic encryption property of our MSHE to encrypt the query value in a way similar to public key encryption, which protects the confidentiality of secret keys. For privacy purposes, it is important to limit what a cloud server can infer about the encrypted data records. More particularly, we formally prove that for every single party, our Multi-Key SHE is semantically secure against chosen plaintext attack. As for the computational efficiency, our MPPkNN scheme achieves four orders of magnitude faster than the prior work under the same security parameters. Moreover, our scheme realizes addition and multiplication homomorphic operations under different secret keys, which theoretically supports the collaboration of any number of data owners.

Keywords: Privacy-preserving · Multi-party · KNN classification · Homomorphic encryption

1 Introduction

The manner in which data is processed is essential in the big data era. With the development of computer and communication technology, data is more integrated into machine learning and plays an irreplaceable role [26]. In recent years, due to its affordability, scalability and reliability, cloud service has gained more recognition in both the business and scientific communities. This makes users more inclined to submit their data and data processing to the cloud to reduce the requirements of their own computing performance.

H. Duan et al. (Eds.): SecureComm 2023, LNICST 568, pp. 266–286, 2025.
https://doi.org/10.1007/978-3-031-64954-7_14

However, although data outsourcing brings convenience, it will cause a series of security problems when the processed data involve user privacy, such as medical data, insurance data, or bank savings data. Therefore, our goal is to extract knowledge from the sensitive data while protecting the privacy of data owners, which is called privacy-preserving data mining [10,13]. In addition, due to the high requirements for data analysis results and the changes in the way of data distribution, the object of data mining has also expanded from a single database to a multi-party database. Collaboration among different users has become a common way. It enables them to create more accurate and trustworthy results in a privacy-preserving data mining method, i.e. more clouds can discover more knowledge than they can uncover on their own by combining their data [3]. Some research propose privacy-preserving solutions for horizontally-partitioned databases to increase the total number of data samples with the goal of creating more accurate data mining models [8]. In some cases, vertically-partitioned database solutions can be preferred to increase the number of attributes for the same instances [21]. Institutions such as hospitals operating in different parts of a country might prefer the first choice. Institutions like banks and insurance corporations, on the other hand, might aggregate data utilizing the second option.

To address above challenges, in this paper, we propose a multi-key symmetric homomorphic encryption (Multi-key SHE) based on [14], and use it to construct our multi-party privacy-preserving k-nearest neighbors classification (MPPkNN) scheme. Although these basic operations can be supported by fully homomorphic encryption techniques [4] in a non-interactive and privacy-preserving way, the consequent computational cost is too high. Specifically, the main contributions of this paper are summarized as follows.

- First, we design a multi-key symmetric homomorphic encryption that achieves homomorphic operations under a master key even though the data is encrypted by the different secret key pair. In addition, new data owners can participate in our scheme at any time without affecting other users, which means our scheme is scalable. We prove that our multi-key symmetric homomorphic encryption is chosen-plaintext attacks (CPA) secure, and it can provide a strong security guarantee for our scheme.
- Second, based on the multi-key SHE, we carefully devise a suit of privacy-preserving protocols for multi-party privacy-preserving k-NN classification scheme including secure query generating protocol (SQGP), secure distance computing protocol (SDCP), secure comparison protocol (SCP), and secure result returning protocol (SRRP). More specifically, the SDCP greatly reduces the communication overhead between servers and prevents secret keys from being leaked to users in a public way.
- Finally, we conduct extensive experiments to evaluate the performance of the proposed MPPkNN scheme. The experiments show that our scheme is four orders of magnitude faster than prior work under the same security parameters.

Table 1. Comparison for Key Features.

Schemes	Multiple DOs	Secret Key Hiding	Query Privacy	Intermediate Result Hiding	Results Privacy	Scalability
Wong [23]	✗	✗	✓	✓	✓	✗
Ren [18]	✗	✓	✓	✓	✓	✗
Samanthula [19]	✗	✓	✓	✓	✓	✗
Zhu [28]	✗	✓	✓	✓	✓	✗
Shaneck [20]	✓	✓	✓	✓	✓	✗
Ours	✓	✓	✓	✓	✓	✓

1.1 Related Work

Due to its efficiency and ease of use, the problem of how to securely compute the k-nearest neighbors of a query point has gained a lot of attention in recent years [15]. Before the maturity of cloud computing technology, the early studies mostly focused on how to implement a secure k-NN method between data owners and clients without cloud system. The work of [10] proposed a method that privately calculates the k-NN classification over horizontally partitioned data in the distributed database model. Additionally, Shaneck et al. [20] present a novel algorithm based on secure multiparty computation primitives to compute the nearest neighbors of records in horizontally distributed data and can extend to multiparty case naturally.

Moreover, by using the secure comparison protocol and the secure inner product protocol constructed by Yao's [25] garbled circuit, [17] implements the secure single-step k-NN with linear computation and communication complexity. Songhori et al. [22] proposes secure k-NN computation with sequential garbled circuit to further improve the efficiency. However, as the number of participating users increases, the garbled circuit in above work will become larger and more complex, which will greatly increase the communication cost. Compared with this, our MPPkNN scheme based on multi-key SHE has better scalability to achieve better multi-party cooperation.

Recent studies have mostly focused on solutions in cloud computing settings, which means the data owners are free of computing. Wong et al. [23] achieved a secure k-NN protocol based on an asymmetric scalar-product-preserving encryption(ASPE) scheme. Different from normal data encryption, [23] used an invertible matrix as a mask to ensure the security of data, which enables them to proceed distance comparison directly instead of calculating an exact distance. However, the secret key should be disclosed to the query user for the uniformity of data, which causes the insecurity of owner's data. Different from this work, Hu et al. [18] encrypts data through privacy homomorphism and devises secure protocols for processing k-NN queries on R-tree index, which is scalable to large datasets. Zhu et al. [28] introduced a secure k-NN query processing on encrypted data which extended form [23]. This work improved ASPE with Paillier Encryption which achieved key confidentiality, query controllability, data privacy, and query privacy. Nonetheless, [28] will leak data access pattern to the cloud. Compared to this work, [11] build their secure k-NN query scheme by utilizing a leveled fully homomorphic encryption and hid data access pattern as well as

preserving the query privacy and data privacy. However, it requires decryption keys to be given to the query users. Different from those works, our scheme will hide the secret key used to decrypt the outsourced dataset and requests even though the encryption is symmetric.

Samanthula [19] described the current state-of-art protocol for k-NN classification in the two-party federated cloud model which is extended from [6]. They also use Paillier encryption as the underlying cryptographic tool and design a new comparison protocol with high security level. However, the efficiency problem still existed in both works. In [5], an efficient scheme used the kd-tree and the order-preserving encryption to achieve privacy-preserving classification, but cannot guarantee semantic security and hide the access pattern. Other recent related work such as Lei et al. [12], proposed a SkNN scheme for 2-D data points using locality sensitive hashing(LSH) [1] to construct data index which improves efficiency without sacrificing security. However, the results returned from cloud contain false positives, which means they are not accurate values. Based on above works, our protocol achieves access pattern hiding and gets the result as same as operating on the plaintext. More details compared to other state-of-art protocols for k-NN classification had summarized in Table 1.

2 Preliminaries

In this section, we will introduce necessary preliminary knowledge, and it will be divided into two parts: (1) Multi-key Symmetric Homomorphic Encryption (MSHE) [14], (2) Multi-party Privacy-preserving k-NN Classification.

2.1 Description of MSHE

- **MSHE.Setup**(κ): Given a security parameters κ, set a series of public parameters $k_p, k_q, k_{\mathcal{M}}, k_{\mathcal{L}}$ and k_g, such that these parameters are satisfied $k_{\mathcal{M}} \ll (k_{\mathcal{L}} + k_g) < k_p/2 = k_q/2 = \kappa/2$. Specifically, $k_{\mathcal{M}}$ limits the message space and k_g limits the maximum number of secret keys. We set the $PP = (k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, k_g)$.
- **MSHE.KeyGen**(PP): On input public parameters, the key generation algorithm sets the message space to be $\mathcal{M} = \{m | m \in [-2^{k_{\mathcal{M}}-1}, 2^{k_{\mathcal{M}}-1}]\}$. Then, it randomly chooses a number $\mathcal{L}_0$ and two prime numbers p, q with the bit length $k_{\mathcal{L}}$ and k_p respectively. Besides, depending on the number of parties, this process will generate a different random number g_i for each party, in which the bit length of g_i is k_g. Finally, it will set $(p, \mathcal{L}_0)$ as the master key and $(p, g_i\mathcal{L}_0)$ as the secret key sent to the corresponding party. After adding $N = pq$ to public parameters, the algorithm is complete.
- **MSHE.Enc**(m, sk): A message m is encrypted by the secret key $sk = (p, g_i\mathcal{L}_0)$ as:

$$c = Enc(m) = (rg_i\mathcal{L}_0 + m)(1 + r'p) \mod \mathcal{N} \tag{1}$$

where $r \in_R \{0,1\}^{k_{\mathcal{L}}}$ and $r' \in_R \{0,1\}^{k_p}$.

- **MSHE.Dec**(c, sk): A ciphertext c can be decrypted with the secret key $sk = (p, g_i\mathcal{L}_0)$ as
$$m = Dec(c) = (c \mod p) \mod g_i\mathcal{L}_0. \tag{2}$$
the correctness of the decryption can be described as follows:
$$\begin{aligned} Dec(c) &= (c \mod p) \mod g_i\mathcal{L}_0 \\ &= (((rg_i\mathcal{L}_0 + m)\,(1 + r'p) \ mod\ \mathcal{N}) \ mod\ p) \ mod\ g_i\mathcal{L}_0 \\ &= (rg_i\mathcal{L}_0 + m) \ mod\ g_i\mathcal{L}_0 \quad (\because 2(k_{\mathcal{L}} + k_g) < k_p) \\ &= m \quad (\because k_{\mathcal{M}} \ll (k_{\mathcal{L}} + k_g)) \end{aligned}$$

- **MSHE.MDec**(c, msk): Since the second part of any secret keys is a multiple of master key, a ciphertext c can also be decrypted under the master secret key $msk = (p, \mathcal{L}_0)$ as
$$m = Dec(c) = (c \mod p) \mod \mathcal{L}_0. \tag{3}$$
Since the relationship of $k_p, k_{\mathcal{L}}$ and $k_{\mathcal{M}}$ is reminded, the correctness of the above decryption can also be guaranteed.

According to the above encryption and decryption process, we can find that the MSHE has the following homomorphic properties under any secret key:

- **HomAdd-I** Given two ciphertexts $c_1 = Enc(m_1) = (r_1g_i\mathcal{L}_0 + m_1)(1 + r'_1p) \ mod\ \mathcal{N}$ and $c_2 = Enc(m_2) = (r_2g_i\mathcal{L}_0 + m_2)(1 + r'_2p) \ mod\ \mathcal{N}$, we have $c_1 + c_2 = Enc(m_1) + Enc(m_2) = Enc(m_1 + m_2)$.
- **HomAdd-II**: Given a ciphertext $c_1 = Enc(m_1)$ and a plaintext m_2, we can calculate $c_1 + m_2 = Enc(m_1) + m_2 = Enc(m_1 + m_2)$.
- **HomMul-I**: Given $c_1 = Enc(m_1)$, $c_2 = Enc(m_2)$, we have $c_1 \cdot c_2 = Enc(m_1) \cdot Enc(m_2) = Enc(m_1 \cdot m_2)$
- **HomMul-II**: Given a ciphertext $c_1 = Enc(m_1)$ and a plaintext m_2, we can calculate $c_1 \cdot m_2 = Enc(m_1) \cdot m_2 = Enc(m_1 \cdot m_2)$.
In addition, the homomorphic properties of the MSHE still hold under the master secret key $\mathcal{L}_0$ even though the ciphertexts are encrypted by the different secret key pairs. More specifically,
- **MulKeyHomAdd-I** Given two ciphertexts $c_1 = Enc(m_1) = (r_1g_i\mathcal{L}_0 + m_1)(1 + r'_1p) \ mod\ \mathcal{N}$ and $c_2 = Enc(m_2) = (r_2g_j\mathcal{L}_0 + m_2)(1 + r'_2p) \ mod\ \mathcal{N}$, $i \neq j$. Then we have $c_1 + c_2 = Enc(m_1 + m_2)$ under msk because,
$$\begin{aligned} c_1 + c_2 &= (r_1g_i\mathcal{L}_0 + m_1)(1 + r'_1p) + (r_2g_j\mathcal{L}_0 + m_2)(1 + r'_2p) \ mod\ \mathcal{N} \\ &= (r_1g_i + r_2g_j)\mathcal{L}_0 + m_1 + m_2 + \alpha \cdot p \ \ mod\ \mathcal{N} \\ MDec(c_1 + c_2) &= ((c_1 + c_2) \ mod\ p) \ mod\ \mathcal{L}_0 \\ &= (r_1g_i + r_2g_j)\mathcal{L}_0 + m_1 + m_2 \ mod\ \mathcal{L}_0 = m_1 + m_2 \end{aligned}$$

- **MulKeyHomMul-I**:, we have $c_1 \cdot c_2 = Enc(m_1) \cdot Enc(m_2) = Enc(m_1 \cdot m_2)$ under msk because,

$$\begin{aligned} c_1 \cdot c_2 &= (r_1 g_i \mathcal{L}_0 + m_1)(1 + r_1' p) \times (r_2 g_j \mathcal{L} + m_2)(1 + r_2' p) \ mod \ \mathcal{N} \\ &= \beta \mathcal{L}_0 + m_1 \cdot m_2 + \alpha \cdot p \ mod \ \mathcal{N} \\ MDec(c_1 \cdot c_2) &= ((c_1 \cdot c_2) \ mod \ p) \ mod \ \mathcal{L}_0 \\ &= \beta \mathcal{L}_0 + m_1 \cdot m_2 \ mod \ \mathcal{L}_0 = m_1 \cdot m_2 \end{aligned}$$

  Besides the normal homomorphic properties between different ciphertexts, the homomorphic properties between ciphertext and plaintext is also remained.
- **MulKeyHomAdd-II**: Given a ciphertext $c_1 = Enc(m_1)$ and a plaintext m_2, we can calculate $c_1 + m_2 = Enc(m_1) + m_2 = Enc(m_1 + m_2)$.
- **MulKeyHomMul-II**: Given a ciphertext $c_1 = Enc(m_1)$ and a plaintext m_2, we can calculate $c_1 \cdot m_2 = Enc(m_1) \cdot m_2 = Enc(m_1 \cdot m_2)$.

We have to notice that, the MSHE scheme is a leveled homomorphic encryption scheme. [27] Its multiplicative depth over ciphertexts is limited by the security parameters and is up to $\left\lfloor \frac{k_p}{2k_{\mathcal{L}}} - 1 \right\rfloor$. The parameters we have chosen are able to satisfy the required number of multiplication operations in our scheme.

2.2 Description of MPP*k*-NN Classification

Without loss of generality, we sssume that there are N data owners, each has a dataset D which contains n data records $\boldsymbol{d_1}, \boldsymbol{d_2}, \ldots, \boldsymbol{d_n}$. Notice that every data record is an *m-dimension* vector and has a class label $l_i \in C$, where C denotes the label collection. All data is stored on cloud servers in encrypted form. The query owner QO holds an *m-dimension* vector $\boldsymbol{Q}$ and wants to get the classification result among the database. Specifically, the query owner sets a parameter k and finds the k nearest neighbors based on all databases according to a distance function, then obtains the result of query as l_q, where $l_q \in C$ is the major class label of those k nearest neighbors. In this paper, we adopt the Euclidean distance function as the distance function.

2.3 System Model

In this section, we present the main constituent entities in our system model, including data owner (DO), query user (QO), cloud computing server (CCS), key management server (KMS), and master key management server (MKMS).

- DO: Data owners have only limited computing power hence they need to outsource their encrypted data records to the cloud servers.
- QO: The query owner is an authorized party holding query value Q for classification, and can obtain the final result in plaintext.
- CCS: Cloud computing server stores the encrypted databases from the data owner and participates in the process of computing.

– KMS: Key management server keeps the secret key for DOs which will be used for outsourcing new data records.
– MKMS: In our scheme, we set a KMS for different use and call it master KMS (MKMS). MKMS is responsible for generating public parameters $(k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, k_{\}}, \mathcal{N})$ as described in MSHE scheme.

2.4 Threat Model

In this paper, we consider the computing cloud servers and key management servers are semi-honest. That means those cloud servers will act in an honest-but-curious way, faithfully following the designed protocols but curious about the private information under ciphertexts they received. Besides, there is no collusion between any of the servers.

3 Proposed Scheme

In this section, we describe basic security protocols and the whole scheme for our multi-party k-NN classification in detail.

3.1 Two-Party PPkNN Scheme

In order to simplify the description, we first introduce a two-party privacy-preserving collaborative k-NN classification scheme based on above Multi-key SHE, then we will show how to extend it to multi-party case. Generally, our scheme can be divided into the following five parts (Fig. 1).

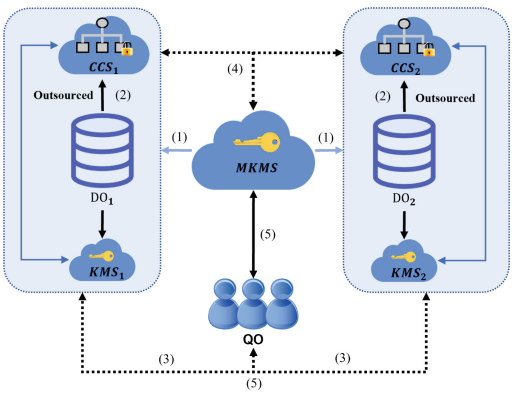


Fig. 1. System Model

System Initialization: In our scheme, the MKMS is considered to act as the authority to bootstrap the whole system. Specifically, it will run Setup algorithm

to generate common security parameters $(k_p, k_q, k_{\mathcal{M}}, k_{\mathcal{L}}, k_g)$. Then executes KeyGen to get the master secret key and secret key pairs. Finally, those secret key pairs will be distributed to the corresponding data owner while the master key will keep in MKMS.

Data Outsourcing: For the sake of the confidentiality of the dataset and subsequent operations, data processing and encryption are needed before outsourcing the dataset to the cloud servers. Hence, the data owners encrypt all dimensions including the class label of each data record in the database with their secret key pair, and outsource the encrypted database(ED) to the CCS. After that, the data owner will not participate in the subsequent scheme process. We suppose the amount of data is n_1, n_2 respectively, then we can denote the vector in ED_i as $\langle(\tilde{d_{j,1}^i}, \tilde{d_{j,2}^i}, \ldots, \tilde{d_{j,m}^i}), \tilde{l_j^i}\rangle$, $j \in [1, n_i]$.

Query Request Generating: Since the encryption scheme is symmetric, the query owner has no idea of any DOs' secret key. Hence we designed Secure Query Protocol based on Multi-key SHE to achieve encryption without secret key.

- **Secure Query Generating Protocol (SQGP):** Since the query owner does not have a secret key, he needs to encrypt the query value with the help of the MKMS in the subsequent query process. We design the secure query generating protocol to ensure the security of query.

At the beginning of the protocol, the query owner requests a query from the MKMS, and then the server generates an encrypted vector of zeros to the query party. Finally, the query owner sums the query value with the encrypted message and sends it to all CCSs. More details are shown in Algorithm 1.

The correctness of the above protocol is based on the **MulKeyHomAdd-II**. After that, the CCS can use the encrypted query to process local k-NN find.

Request Processing: This stage is mainly divided into three steps: Euclidean distance computing, local k-NN find, and across minimum comparison.

1. Euclidean distance computing: With encrypted query vector $\tilde{\boldsymbol{Q}} = (\tilde{q_1}, \tilde{q_2}, \ldots, \tilde{q_m})$, the $\mathrm{CCS}_1(\mathrm{CCS}_2)$ will compute the Euclidean distance between every data records of $\mathrm{ED}_1(\mathrm{ED}_2)$ and $\tilde{\boldsymbol{Q}}$ by following protocol.

- **Secure Distance Computing Protocol:** Since both additive and multiplicative homomorphic properties of Multi-key SHE are satisfied, the calculation of the distance does not require tedious steps. Notice that we used the square of the distance instead of the sqrt root as in the definition, but the magnitude relationship remains. Firstly, the $\mathrm{CCS}_1(\mathrm{CCS}_2)$ computes the value $(\mathcal{N}-1)\cdot\tilde{\boldsymbol{Q}}\ mod\ \mathcal{N}$, then adds it to each data records $\tilde{\boldsymbol{d_j}}$ in encrypted database. After that, the only thing left is multiplied by itself. Finally, by adding all dimensions together, we can get the set of Euclidean distance and we denote it as $\{Dis_{1,j}\}(\{Dis_{2,j}\})$. It is worth noting that these computations are able to be parallelized between CCS_1 and CCS_2. In addition, there is no communication overhead incurred during the entire distance computation.

Algorithm 1 Secure Query Generating Protocol

Input: $\boldsymbol{Q}$ of query owner.
Output: Encrypted $\boldsymbol{Q}$

1: **QO** : Request a query from the MKMS.
2: **MKMS** : After approving the query, the MKMS will generate a vector $\boldsymbol{C}$ consisting of m encrypted message $c_i, i = 1 \ldots m$.
$c_i = \textbf{MSHE.Enc}(0, msk) = (r_i \mathcal{L}_0 + 0)(1 + r'_i p) \ mod \ \mathcal{N}$
Send $\boldsymbol{C}$ to QO.
3: **QO** : After receiving the ciphertext $\boldsymbol{C}$, the query owner adds it to the query value:
$\tilde{\boldsymbol{Q}} = \boldsymbol{Q} + \boldsymbol{C}$
Send $\tilde{\boldsymbol{Q}}$ to all CCSs.

2. Local k-NN find: Since the operation is parallel across computing servers, without loss of generality, we just gave the actions in CCS_1 with the set $\{Dis_{1,j}\}$. To make the comparison more efficient, we use the complete binary tree to implement the process [24]. More specifically, a $(2n_1 - 1)$-length list L is used as our complete binary tree and the last n_1 nodes are assigned with the order number j of elements in $\{Dis_{1,j}\}$. We then set each parent's value to the minimum of its two children from the bottom up by using the following secure comparison protocol, and get the first minimum value from the root. For finding the next $k-1$ nearest neighbors, we need to run algorithm List Adjust (LA) [24]. The main idea of LA is to first record the path of the current minimum in the binary tree from top to bottom, set the value of each node in the path to 0, and then generate the next minimum in the same way from bottom to top.

- **Secure Comparison Protocol:** This protocol is to compute the minimum value of any two encrypted elements Dis_{1,j_1}, $Dis_{1,j_2}$of $\{Dis_{1,j}\}$ with the help of master key. Firstly, the CCS_1 randomly chooses three numbers r_1, r_2 and r_3, where $r_1, r_2 \in \{0,1\}^{k_{\mathcal{M}}}$ satisfying $r_1 > r_2 > 0$ and $r_3 \in \{-1, 1\}$. If $r_3 = 1$, the CCS_1 computes $t = Dis_{1,j_1} - Dis_{1,j_2}$, else $t = Dis_{1,j_2} - Dis_{1,j_1}$. Then it will calculate $c = r_1 * t + r_2$ and send this to MKMS. On receiving c, the MKMS decrypts it with the master key and set a value $s = 1$ if $D_{msk}(c) < 0$ and $s = -1$ otherwise, then returns that value to CCS_1. Finally, if $s \times r_3 = 1$, the CCS will set the minimum value is Dis_{1,j_2}, otherwise, the minimum number is Dis_{1,j_1}
- **Correctness:** According to the homomorphic properties of multi-party SHE, even if the encrypted key of query value and data records is different, Dis_{1,j_1} still presents the right encrypted value of distance between them under the master key. Hence we denote Dis_{1,j_1} as $E_{msk}(D_{j_1})$, the same to Dis_{1,j_2}. Then, when $D_{j_1} < D_{j_2}$, if we choose $r_3 = 1$, we have $c = r_1 * E_{msk}(D_{j_1} - D_{j_2}) + r_2 = E_{msk}(r_1 * (D_{j_1} - D_{j_2}) + r_2) = E_{msk}(x)$, where $x < 0$. Then $s = -1$, $min\{Dis_{1,j_1}, Dis_{1,j_2}\} = Dis_{1,j_1}$. If we choose $r_3 = -1$, $c = r_1 * E_{msk}(D_{j_2} - D_{j_1}) = E_{msk}(x)$, where $x > 0$. Hence we will get $s = 1, s \times r_3 = -1$, the $min\{Dis_{1,j_1}, Dis_{1,j_2}\}$ still is Dis_{1,j_1}. After calculus, the correctness of the other two cases can also be guaranteed.

Algorithm 2 Local k-NN find

Input: Encrypted distances $\{Dis_{1,j}\}$, Cooresponding label set $\{l_j^1\}$, Constant k
Output: k encrypted labels $\{l_s^1\}$

```
Generete a (2n_1 − 1)-length list L.
for all t ∈ [2n_1 − 1, n_1] do
    L[t] = t − n_1 + 1
end for
for all t ∈ [1, n_1 − 1] do
    if SCP(Dis_{1,L[2t]}, Dis_{1,L[2t+1]}) = 1 then
        L[t] = L[2t + 1]
    else
        L[t] = L[2t]
    end if
end for
for all s ∈ [1, k] do
    l_s^1 = l~_{L[1]}^1
    LA(L, {Dis_{1,j}})
end for
return {l_s^1}
```

3. Across Minimum Comparison: After processing local k-NN find, CCS_1 and CCS_2 maintain their k nearest neighbors list based on local data records, the left thing is to choose the final k minimum value across those lists. We run algorithm List Merge (LM) to achieve it. The basic operation is to compare two elements of lists in order until we find k minimum results.

Result Returning: After all the computation is completed, the cloud computing server needs to return the results to the query party safely. Since the encryption is implemented by the owner's private key, the decryption of the final results needs to be completed by the key management server. Hence we propose the following protocol.

- **Secure Result Returning Protocol:** If we send the final k results directly to the MKMS for decryption, the plaintext will reveal the information of classification. Instead of that, the query owner randomly chooses k encrypted ciphertexts of zero in the query request generating step(if the m is less than k, the query owner is able to request more ciphertexts from MKMS) and generates a set of random numbers $r_\alpha^1, \ldots, r_\alpha^k$. Constructing a vector $\boldsymbol{V} = (v_1, \ldots, v_k), v_i = E_{msk}(0)_i + r_\alpha^i$ and sending it to CCS_1.

After calculating $\boldsymbol{L'} = \boldsymbol{V} + \boldsymbol{L}$, the CCS_1 sends it to MKMS for decryption, and the final results will return to query owner directly.

3.2 Extension to the Multiparty Case

So far, we have concentrated on the case where two data owners are involved in the protocol. In many applications, however, this algorithm would be more

Algorithm 3 Secure Comparison Protocol

Input: Two ciphertexts $E(a), E(b)$
Output: $E(min\{a, b\})$

CCS$_1$: Choose $r_1, r_2 \in \{0,1\}^{k_{\mathcal{M}}}$ randomly where $r_1 > r_2$. Choose $r_3 \in \{-1, 1\}$ randomly.
if $r_3 = 1$ **then**
  Compute $t = E(a) - E(b) = E(a - b)$
  Compute $E(x) = r_1 \times t + r_2 = E(r_1 \times (a - b) + r_2)$.
else
  Compute $t = E(b) - E(a) = E(b - a)$
  Compute $E(x) = r_1 \times t + r_2 = E(r_1 \times (b - a) + r_2)$
end if
Send $E(x)$ to MKMS.

MKMS: Decrypt $E(x)$ with master key $(p, \mathcal{L}_0)$.
if $x > 0$ **then**
  $s = 1$
else
  $s = -1$
end if
Return s to CCS$_1$.

CCS$_1$:
if $s \times r_3 = 1$ **then**
  $E(min\{a, b\}) = E(b)$
else
  $E(min\{a, b\}) = E(a)$
end if

useful if it was extended to the case when more than two parties participate. In this section, we will show how to extend our scheme to multiple parties.

More specifically, if there is a new data owner who wants to participate in classification, then he first needs to submit an application to MKMS. After judging whether the user limit is exceeded, the MKMS will generate and distribute the corresponding secret key pair to the request if the bit length of data owners' number is less then the selected parameter k_g. The process of outsourcing is as same as in the case of two parties.

Then, each CCS will compute the distance and execute k-NN finding to get local k nearest neighbors list. In order to obtain the final results, we need to compare the list of CCS$_1$ to others by operating LM algorithm $N - 1$ times. Finally, the CCS$_1$ executes secure result returning protocol to return the classification.

4 Security Analysis

In this section, we analyze the security of our proposed multi-party PPkNN scheme. Since our scheme is based on the Multi-key SHE, we first prove that the

Multi-key SHE is semantically secure against CPA, and then prove the security of the whole scheme.

4.1 Security of Multi-key SHE

We prove that our Multi-Key SHE is semantically secure against CPA under the $(\mathcal{L}_0, p)$-based decision assumption defined in [27]. According to the Mulit-Key SHE, let k_g be the user boundary parameter. Let k_p and $k_\mathcal{L}$ be two security parameters satisfying $k_\mathcal{L} + k_g < k_p/2$. Let p and q be two large primes with bit length k_p. Let $\mathcal{N} = pq$. Let $\mathcal{L}_0, g_i$ be two random number with $|\mathcal{L}_0| = k_\mathcal{L}$ and $|g_i| = k_g$ respectively. Let $\mathcal{L}_i = g_i\mathcal{L}_0$ and the secret key pair is $(\mathcal{L}_i, p)$ and the master key pair is $(\mathcal{L}_0, p)$.

Theorem 1. *For every single party in Multi-Key SHE with secret key pair $sk_i = (\mathcal{L}_i, p)$, their ciphertext is semantically secure against CPA under the $(\mathcal{L}_0, p)$-based decision assumption.*

Before proving the above theorem, we introduce three lemmas that are essential to support our proof.

Lemma 1. *Any valid ciphertext in the Multi-Key SHE is in the form of $E_{msk}(m) = (m + \alpha\mathcal{L}_0 + \beta p) \bmod \mathcal{N}$, where $\alpha, \beta \in \mathbf{Z}_\mathcal{N}$ and $\alpha\mathcal{L}_0 < p$.*

Lemma 2. *For any $x \in \mathbb{S}(\mathbb{S} : \{x = (\alpha\mathcal{L}_0 + \beta p) \bmod \mathcal{N} | \alpha, \beta \in \mathbf{Z}_\mathcal{N}, \alpha\mathcal{L}_0 < p\}$, the representation $x = (\alpha\mathcal{L}_0 + \beta p) \bmod \mathcal{N}$ is unique.*

Lemma 3. *For any $x \in \mathbf{Z}_\mathcal{N}$, there exist $\alpha, \beta \in \mathbf{Z}_\mathcal{N}$ such that $x = (\alpha\mathcal{L}_0 + \beta p)$ mod $\mathcal{N}$.*

The details of above three lemmas' proof can be found in [27]. Based on the definition of set $\mathbb{S}$, we can see that the $\mathbf{Z}_\mathcal{N}$ can be divided into two sets

$$\begin{cases} \mathbb{S} : \{x = (\alpha\mathcal{L}_0 + \beta p) \bmod \mathcal{N} \mid \alpha, \beta \in Z_\mathcal{N}, \alpha\mathcal{L}_0 < p\} \\ \overline{\mathbb{S}} : \{x = (\alpha\mathcal{L}_0 + \beta p) \bmod \mathcal{N} \mid \alpha, \beta \in Z_\mathcal{N}, \alpha\mathcal{L}_0 \geq p\} \end{cases}$$

Remark that $\mathbb{S} \cup \overline{\mathbb{S}} = \mathbf{Z}_\mathcal{N}$. According to the two sets, we can give the following definition of the $(\mathcal{L}_0, p)$-based decision problem.

Definition 1. *Given $(k_p, k_\mathcal{L}, \mathcal{N})$ the $(\mathcal{L}_0, p)$-based decision problem is to determine that a random number $x \in \mathbf{Z}_\mathcal{N}$ is in $\mathbb{S}$ or $\overline{\mathbb{S}}$ without knowing $(p, q, \mathcal{L}_0)$.*

From [27], we know that there are only two ways to solve the $(\mathcal{L}_0, p)$-based decision problem. One is to factorize $\mathcal{N}$ to obtain p, q, which is an admittedly difficult problem. The other is to exhaust the values of $\{\alpha, \mathcal{L}_0\}$ and compute $gcd(x - \alpha\mathcal{L}_0, \mathcal{N})$. However, the $\mathcal{L}_0$ is a $k_\mathcal{L}$-bit number, which means the computational cost is larger than $2^{k_\mathcal{L}}$. Hence, when the security parameters k_p and $k_\mathcal{L}$ are large enough, the $(\mathcal{L}_0, p)$-based decision problem is intractable.

Definition 2 ($(\mathcal{L}_0, p)$-based Decision Assumption). *$(\mathcal{L}_0, p)$ based decision problem satisfies $(\mathcal{L}_0, p)$-based decision assumption if for any polynomial time algorithm, its advantage in solving the $(\mathcal{L}_0, p)$-based decision problem is a negligible function in k_p and $k_{\mathcal{L}}$.*

Finally, we can prove the Theorem 1.

For a single party with key pair $sk_i = (\mathcal{L}_i, p)$, we assume that there exists a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ that has a non-negligible advantage ε to break the semantic security of the encryption. We can construct a challenger $\mathcal{B}$, which has a non-negligible advantage to break the $(\mathcal{L}_0, p)$-based decision problem by accessing to $\mathcal{A}$. Let $z \in \{0, 1\}$ be a random bit and $(k_0, k_2, \mathcal{N}, x)$ be a $(\mathcal{L}_0, p)$-based decision instance which is given to $\mathcal{B}$. Note that x is randomly chosen from $\mathbb{S}$ if $z = 0$, and x is randomly chosen from $\overline{\mathbb{S}}$ if $z = 1$. Then, the $(\mathcal{L}_0, p)$-based decision problem is to guess z.

With the four tuples $(k_p, k_{\mathcal{L}}, \mathcal{N}, x)$, $\mathcal{B}$ first chooses $k_{\mathcal{M}}$ such that $k_{\mathcal{M}} \ll k_{\mathcal{L}}$ and sets $\mathcal{M} = \left\{m \mid m \in \left(-2^{k_1}, 2^{k_1}\right)\right\}$ as the message space. Then, $\mathcal{B}$ sends $(k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, \mathcal{N})$ to $\mathcal{A}$.

After receiving $(k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, \mathcal{N})$, $\mathcal{A}$ chooses two messages $m_0, m_1 \in \mathcal{M}$ and send them to $\mathcal{B}$. At this time, $\mathcal{B}$ flips a bit $b \in \{0, 1\}$, computes $c = E_{sk_i}(m_b) + x$ and returns c as a ciphertext back to $\mathcal{A}$.

Upon receiving c, $\mathcal{A}$ returns $\mathcal{B}$ a bit $b' \in \{0, 1\}$ as the guess of b. $\mathcal{B}$ then guesses $z = 0$ if $b' = b$. We can see that, when $z = 0$, $x \in \mathbb{S}$, we have $c = E_{sk_i}(m_b) + x = m_b + \alpha_i \mathcal{L}_i + \beta_i p + \alpha \mathcal{L}_0 + \beta p \bmod \mathcal{N} = m_b + (\alpha_i g_i + \alpha)\mathcal{L}_0 + (\beta_i + \beta)p \bmod \mathcal{N}$. Since the bit length of g_i is k_g which satisfies $k_{\mathcal{L}} + k_g < k_p/2$, the part $(\alpha_i g_i + \alpha)\mathcal{L}_0$ is still less then p. Therefore, the c is a valid ciphertext under the master key $\mathcal{L}_0$. In this case, $\mathcal{A}$ can exert his capability and will guess b correctly with the probability $\frac{1}{2} + \varepsilon$. Then, $\Pr[\mathcal{B} success \mid z = 0] = \frac{1}{2} + \varepsilon$. On the other hand, when $z = 1$, i.e., $x \in \overline{\mathbb{S}}$, $c = E_{sk_i}(m_b) + x = m_b + (\alpha_i g_i + \alpha)\mathcal{L}_0 + (\beta_i + \beta)p$ is no longer a valid ciphertext because $(\alpha_i g_i + \alpha)\mathcal{L}_0 > \alpha \mathcal{L}_0 \geq p$. Then, the probability that $\mathcal{A}$ can guess b correctly is only $\frac{1}{2}$. Then, $\Pr[\mathcal{B}\ success \mid z = 1] = \frac{1}{2}$. Summarizing the above two cases, we have $\Pr[\mathcal{B}\ success] = \frac{1}{2}\left(\frac{1}{2} + \varepsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{2}$. Since ε is non-negligible, the above result contradicts the $(\mathcal{L}_0, p)$-based decision assumption. Thus, the Multi-Key SHE is semantically secure against CPA for every single party.

After that, we also have to prove that the ciphertext between any two parties is computationally indistinguishable if they are non-collusive.

Theorem 2. *For any two parties which are non-collusive, their ciphertext is computationally indistinguishable under the $(\mathcal{L}_0, p)$-based decision assumption.*

We first consider that if these parties have colluded, then they can calculate the greatest common divisor among their secret key $\mathcal{L}_i$. Since the parameter k_g is relatively small compared to $k_{\mathcal{L}}$, it is computationally feasible to obtain the master key $\mathcal{L}_0$.

If they are relatively independent, then we can show that any ciphertexts from those parties are contained in the subspace of master key pair $(\mathcal{L}_0, p)$. Since the existence of random numbers during the encryption process and the $(\mathcal{L}_0, p)$-based decision assumption, the ciphertext is computationally indistinguishable.

4.2 Security of SCP

We use $V_{CCS_i}(SCP) = \{r_1, r_2, r_3, t, c, s\}$ to denote the execution view of CCS_i in SCP protocol, and describe the simulated view of CCS_i by $V^S_{CCS_i}(SCP) = \{r'_1, r'_2, r'_3, t', c', s'\}$, where r'_1 and r'_2 are two random integers with bit length $k_{\mathcal{M}}$ and r'_3 can randomly set as -1 or 1. Furthermore, t', c' are two random integers in $\mathcal{Z}_{\mathcal{N}}$. As the returned result, the s' is also selected from $\{-1, 1\}$ such that $s' \times r'_3 = s \times r_3$. Since the MSHE is secure against CPA, the $\{c, t\}$ is computationally indistinguishable from $\{c', t'\}$. Except that, $\{r_1, r_2\}$ is also computationally indistinguishable from $\{r'_1, r'_2\}$ because all of them are random numbers. As for r_3 and s, there are four possible combinations. Therefore, $V_{CCS_i}(SCP)$ is computationally indistinguishable from $V^S_{CCS_i}(SCP)$.

From the perspective of MKMS, the received message is c and output s, hence we simulate this view in the following way: if $s = 1$, we randomly choose a positive number in $(0, 2^{\mathcal{M}-1}]$ and encrypt it with master key $(p, \mathcal{L}_0)$. Otherwise, we select a negative number in $[2^{\mathcal{M}-1}, 0)$ and do the same operation. Since the plaintexts are random values with the same sign, hence the view of the simulator is also computationally indistinguishable.

4.3 Security of Other Protocols

Since the SDC protocol only operates locally without data exchange, the security is based on Multi-key SHE which has been proven secure against CPA. The rest of the protocols, including TBKF, LA, and LM, consist of some local computation and communication between CCS and MKMS using our secure comparison protocol. Therefore these protocols are secure as long as the secure comparison protocol is secure.

4.4 Security of MPP*k*NN Classification Scheme

In general, our scheme can be regard as a kind of searchable encryption scheme [2], hence we prove its security with real/ideal world model. In the multi-party case, there are N data owners, each of whom owns two servers. For simplicity, we select only one DO of these data owners as the entity for analyzing and treat the others as a whole entity DO$'$. Since they are non-collusive, the above operation will not affect the final result. Then we can define the leakage function of our scheme from the perspective of $CCS, KMS, MKMS$, and DO$'$.

- L(CCS): The information leaked to CCS includes the public parameters pp, encrypted dataset ED, encrypted query value $\tilde{Q}$, and local minimum distance list $\{MDis\}$.
- L(KMS): The information leaked to KMS includes the public parameters pp, secret key pair $(\mathcal{L}_i, p)$. However, it does not participate in any subsequent operations, but only holds the key for data updating. So we do not need to analyze its security.

- L(MKMS): The information leaked to MKSM includes the public parameters pp, master key pair $(\mathcal{L}_0, p)$, final blinding encrypted results $\tilde{L}$ and encrypted data during the execution of the SCP.
- L(DO′): The information leaked to DO′ includes the public parameters pp, and final encrypted results $L = \{\tilde{l_1}, \tilde{l_2}, \cdots, \tilde{l_k}\}$

Based on L(CCS), L(KMS), and L(MKMS), we can define the real/ideal world as follows.

Real World: In the real world, there are four probabilistic polynomial time (PPT) adversaries, denoted by $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and $\mathcal{A}_4$. Notice that these adversaries are non-collusive. There is a challenger who interacts with adversaries as follows.

- ***System initialization phase:*** The challenger runs the system initialization algorithm to generate public parameters $pp = (k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, k_g, \mathcal{N})$, master key pair $(\mathcal{L}_0, p)$, and DO's secret key pair $(\mathcal{L}_i, p)$. Then, the challenger publishes pp and sends msk and sk to $\mathcal{A}_3$ and $\mathcal{A}_2$ respectively.
- ***Outsourcing phase:*** $\mathcal{A}_1$ sends dataset $\{(d^i_{j,1}, \ldots, d^i_{j,m}, l^i_j)|j \in [1, n_i]\}$ to challenger. Then the challenger encrypts dataset and sends ED to $\mathcal{A}_1$.
- ***Query generating phase:*** After receiving a blinding query value, the challenger sends $\tilde{Q}$ to $\mathcal{A}_1$.
- ***kNN finding phase:*** When $\mathcal{A}_1$ obtains ED and $\tilde{Q}$, $\mathcal{A}_1$, $\mathcal{A}_3$ and $\mathcal{A}_4$ cooperatively calculate on ED to get the final classification results.
  In real world, the view of $\mathcal{A}_1$ is $View_{\mathcal{A}_1,Real} = \{ED, \tilde{Q}, \{MDis\}\}$ and the comparison results from the secure comparison protocol. The view of $\mathcal{A}_3$ is $View_{\mathcal{A}_3,Real} = \{msk, \tilde{L}\}$ and some SHE ciphertexts received in SCP. The view of $\mathcal{A}_4$ is $View_{\mathcal{A}_4,Real} = \{L\}$.

Ideal World: The ideal world involves a simulator with leakage L(CCS), L(KMS), L(MKMS) and L(DO′). The simulator can construct the view as follow.

- For the view of $\mathcal{A}_1$, the simulator uses the $sk \in$ L(KMS) to encrypt a set of random numbers with bit-length k_1, which has the same amount as the data in ED, and denote it as ED′. Then, the simulator chooses a k_1-bit random number $\tilde{Q}'$ and encrypts the list $\{1, 2, \cdots, k\}$ with sk, which denoted as $\tilde{L}'$. We denote the view of $\mathcal{A}_1$ is $View_{\mathcal{A}_1,Ideal} = \{ED', \tilde{Q}', \tilde{L}'\}$. Since the multi-key SHE is CPA-secure, the ciphertexts are indistinguishable. Therefore, $\mathcal{A}_1$ can not distinguish the ED and ED′. Besides, the two lists $\{MDis\}$ and $\tilde{L}'\}$ are in order and also encrypted by sk. So we have that the $View_{\mathcal{A}_1,Real}$ and $View_{\mathcal{A}_1,Ideal}$ are indistinguishable.

As for other adversaries, we can build their view in the ideal world in a similar way. Since the Multi-key SHE and those privacy-preserving protocols are secure, we can assert that our scheme is selectively secure with the leakage L(CCS), L(KMS), L(MKMS), and L(DO′).

5 Experimental Evaluation

5.1 Computational Complexity

The first thing to consider is the complexity of encryption. Compared to other FHE techniques, Multi-key SHE only involves addition and multiplication operations, which results in less expence while encrypting data. Moreover, since the computational cost of an addition operation is far less than a multiplication operation, we discard addition operation from our analysis. The complexity required for each multiplication operation is the square of the length of the data bits. In addition, the modulo operation in the encryption and decryption process is relative to the division operation, which has the same complexity as the multiplication operation. Therefore, each encryption of Multi-key SHE takes nearly $k_{\mathcal{L}}^2 + k_p^2 + 2 \times (2k_p)^2 = k_{\mathcal{L}}^2 + 9k_p^2$ and each decryption needs $k_p^2 + k_{\mathcal{L}}^2$. More specifically, we suppose that the total amount of data is $m \times n_{sum} = \sum_i^N n_i$, then for data outsourcing, it needs n_{sum} encryption operations. Since the Euclidean distances calculating phase executes square operations on ciphertext, the cost is $mn_{sum} \times (k_p)^2$. When executing k-NN finding, it runs nearly n_i secure compare protocol for the first nearest neighbor and $log_2^{n_i}$ for the next $k-1$ LA operations in each CCS. Since the secure compare protocol needs to transform the ciphertext once and decrypt it, the complexity of k-NN finding sums to $[n_i + (k-1)log_2^{n_i}] \times (10k_p^2 + 2k_{\mathcal{L}}^2)$. After the local finding, we execute $N-1$ across compare protocol, which executes at most $k \times (N-1)$ SCP. In summary, the complexity of whole k-NN scheme is $\mathcal{O}(kNm(k_p^2 + k_{\mathcal{L}}^2))$.

5.2 Communication Complexity

For MSHE, the bit length of one ciphertext is $2k_p$. Therefore, in the stage of data outsourcing, it needs to send $2mn_{sum}k_p$ bits to according CCSs in total. When computing the Euclidean distance, all the ciphertexts are computed locally without the participation of the key management server, so a large part of the communication overhead is saved. For k-NN finding stage, the local computation invokes secure comparison protocol $\sum_i^N [n_i + (k-1)log_2^{n_i}]$ times which needs to exchange $\sum_i^N [n_i + (k-1)log_2^{n_i}] \times (2k_p + 1)$ bits and the across compare stage will run it at most $k \times (N-1)$ times which exchange $k(N-1) \times (2k_p + 1)$ bits. Finally, the return of the generated results exchanges $2k$ ciphertext and k final results between QO, MKMS and CCS, which sums to $k(4k_p + k_{\mathcal{M}})$ bits.

5.3 Experiment Performance

In this section, we evaluate the proposed MPPKNN scheme by conducting experiments on both synthetic and real-world datasets. These experiments were performed on a laptop with i7-11700 CPU 2.50GHz eight-core processor, 32-GB memory and Windows 10 operating system using the programming language Java. In order to facilitate the subsequent comparison with other experiments, we set the number of participants to two. However, it should be noted that due

to the limitations of equipment conditions, we were unable to consider the scenario of local parallel computing of all parties in this paper. For a comprehensive understanding of the security aspects, we have provided detailed security parameter settings for the entire scheme in Table 2. These settings play a crucial role in ensuring the robustness and reliability of our proposed MPPKNN scheme.

We now evaluate the influence of the parameters on the performance. For this, we design several sets of comparison experiments to simulate the proposed scheme on three synthetic datasets with 10000 data records, whose dimensions are 10, 20, and 30, respectively. The results are summarized in Fig. 2

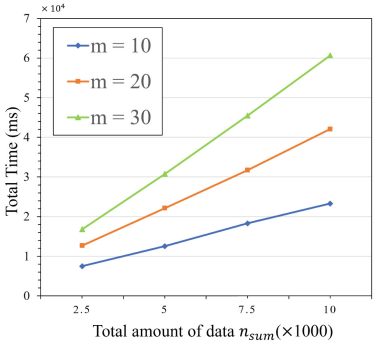
- *The parameter n_{sum} and m:* We first evaluated the computation costs of the whole scheme in different dimensions as well as the total amount of data. The parameters are set as $k = 50, n_{sum} = \{2500, 5000, 7500, 10000\}, m = \{10, 20, 30\}$. The results are shown in Fig. 2(a). Specifically, the computation cost grows linearly with the amount of data under certain dimensions and increases obviously with dimension for a certain amount of data. Moreover, the experiment result shows that our scheme is pretty efficient. For example, when $n_{sum} = 10000$ and $m = 30$, the whole time taken is only 6067 ms.
- *The parameter n_{sum} and k:* In Fig. 2(b), we plot the computational cost of the our k-NN classification varying with k and n_{sum}. In this experiment, we set $m = 20, k = \{10, 50.150\}$. From this chart, we can see that the cost is not sensitive to changes in k. For example, when $n_{sum} = 10000$ and $k = 150$, the computational cost is about 5833 ms while the cost is 5694 ms when $n_{sum} = 10000$ and $k = 50$, which only reduced 139 ms. The reason is that the proportion of TBKF operation in the whole scheme is not high, the operation is directly operated on the ciphertext, and through the construction of a binary tree, the update complexity for the next nearest neighbor is reduced to logarithmic level.
- *The parameter N:* Finally, we extend the number of DOs to multiple parties, set $N = \{10, 50, 100\}$, and the final result is shown in Fig. 2(c). It is worth noting that the total amount of data is still 10000 despite the increase in the number of DOs, which means the amount of data per data owner is an average. It can be seen from Fig. 2(c) that the computation cost increases significantly with the number of DOs, which means that when the number of DOs increases, local parallel computing can save a lot of unnecessary overhead. At the same time, it can be predicted that when the parties can compute in parallel, the imbalance of the local data volume will also bring a certain impact on the efficiency (Table 3).

Table 2. Parameter settings

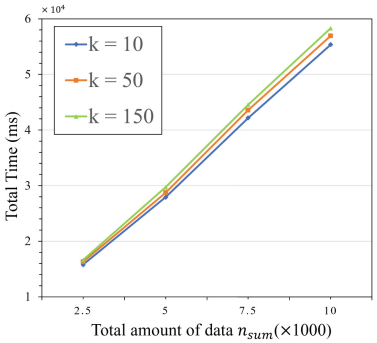
Parameter	Value
N	10, 50, 100
$k_p, k_{\mathcal{M}}, k_{\mathcal{L}}, k_g$	$k_p = 1024, k_{\mathcal{M}} = 60, k_{\mathcal{L}} = 200, k_g = 15$
$p, q, \mathcal{L}_0, \mathcal{N}$	$\|p\| = \|q\| = k_p, \|\mathcal{L}_0\| = k_{\mathcal{L}}, \mathcal{N} = pq$
$\mathcal{L}_i$	$\|\mathcal{L}_i\| = k_{\mathcal{L}} + k_g$

Table 3. Time cost comparison

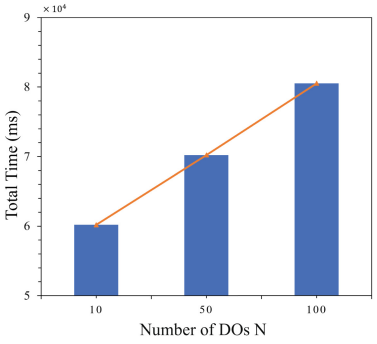
	N = 2500	N = 5000	N = 7500	N = 10000
Scheme 1 [24]	324.996(s)	653.74(s)	969.146(s)	1282.675(s)
Scheme 2 [16]	487.137(s)	963.643(s)	1467.591(s)	1950.875(s)
Scheme 3 [23]	0.213(s)	0.425(s)	0.640(s)	0.851(s)
Our	0.748(s)	1.254(s)	1.829(s)	2.328(s)



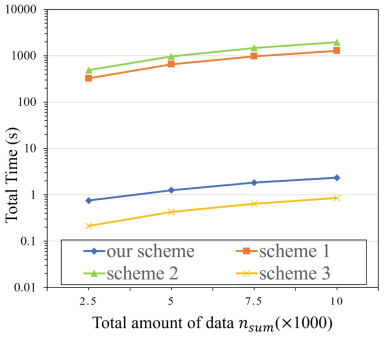
(a) Fixed $N = 2, k = 50$.



(b) Fixed $N = 2, m = 20$.



(c) Fixed $m = 30, k = 50$.



(d) Comparison. Fixed $m = 10, k = 10$

Fig. 2. Computation costs of MPPkNN for varying number of dimension m, nearest neighbors k and Data Owner N

5.4 Comparison

In this section, we make a comparison with three state-of-the-art privacy-preserving k-NN classification schemes on a real-world dataset from [9], which possesses 10545 records and 29 attributes. Specifically, the PPkNN scheme proposed in [16] uses the Paillier encryption system as the basic security tool and constructed the scheme based on two non-colluding servers.

In [24], they built secure k-NN scheme classification with the Elgamal variant technique, which ensures the privacy of other users and the outsourced dataset

will not be leaked even if there exist colluded users. Besides, the schemes in [23] used matrices to encrypt the database and user query, which achieve higher efficiency on computation.

We set the module length is 1024 bits for scheme 1 [24], scheme 2 [16] and our proposed scheme, and the bit length for scheme 3 [23] is 64. The other parameters are set as $N = 2, k = 10, m = 10, n_{sum} = \{2500, 5000, 7500, 10000\}$ and the final comparison results are summarized in Fig. 2(d).

When compared to scheme 1 and scheme 2, our proposed encryption scheme demonstrates a significant reduction in time overhead by several orders of magnitude. For instance, when n_{sum} is set to 10000, scheme 1 requires approximately 1282.675 s, whereas our scheme only takes 2.328 s. This notable improvement in time efficiency showcases the effectiveness of our proposed scheme. Although the computational efficiency is slightly lower compared to scheme 3, our scheme has higher security in encryption and communication and will not leak any information about the secret key.

6 Conclusion

Using cloud services to operate the k nearest neighbors classification on data can not only improve efficiency but also reduce the local overhead. In order to avoid disclosure of privacy, we wrap the data with cryptographic tools while still meet the computational requirements. At the same time, in order to cope with the trend of multi-party collaborative data mining, we proposed the MPPkNN scheme which theoretically supports the collaboration of any number of data owners. In addition, our scheme is more efficient due to the use of the multi-key symmetric homomorphic encryption, which reduces the whole computational cost of our system. However, if more and more data owners participate with geometrically increased amount of data and data dimension, the tedious calculation of Euclidean distance between query input and all dataset records will still limits the improvement of the efficiency. In the future, we will focus on developing more secure protocols that can be executed in parallel, while applying methods such as LSH [1] to reduce unnecessary computations to achieve higher efficiency. Moreover, we will consider implementing bootstrapping operation as described in [7] so that the MSHE can support infinite numbers of multiplications.

References

1. Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., Schmidt, L.: Practical and optimal LSH for angular distance. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
2. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
3. Demir, S., Tugrul, B.: Privacy-preserving trend surface analysis on partitioned data. Knowl.-Based Syst. **144**, 16–20 (2018)

4. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_2
5. Du, J., Bian, F.: A privacy-preserving and efficient k-nearest neighbor query and classification scheme based on k-dimensional tree for outsourced data. IEEE Access **8**, 69333–69345 (2020)
6. Elmehdwi, Y., Samanthula, B.K., Jiang, W.: Secure k-nearest neighbor query over encrypted data in outsourced environments. In: International Conference on Data Engineering (2013)
7. Guan, Y., Lu, R., Zheng, Y., Zhang, S., Shao, J., Wei, G.: Toward privacy-preserving cybertwin-based spatiotemporal keyword query for its in 6G era. IEEE Internet Things J. **8**(22), 16243–16255 (2021)
8. Inan, A., Kaya, S.V., Saygın, Y., Savaş, E., Hintoğlu, A.A., Levi, A.: Privacy preserving clustering on horizontally partitioned data. Data Knowl. Eng. **63**(3), 646–666 (2007)
9. Johnson, B.A., Iizuka, K.: Integrating OpenStreetMap crowdsourced data and Landsat time-series imagery for rapid land use/land cover (LULC) mapping: case study of the Laguna de bay area of the Philippines. Appl. Geogr. **67**, 140–149 (2016)
10. Kantarcıoğlu, M., Clifton, C.: Privately computing a distributed k-nn classifier. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 279–290. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30116-5_27
11. Kesarwani, M., et al.: Efficient secure k-nearest neighbours over encrypted data. In: EDBT, vol. 2018, pp. 564–575 (2018)
12. Lei, X., Liu, A.X., Li, R.: Secure KNN queries over encrypted data: dimensionality is not always a curse. In: International Conference on Data Engineering (2017)
13. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–54. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_3
14. Mahdikhani, H., Lu, R., Zheng, Y., Shao, J., Ghorbani, A.A.: Achieving o $(\log^3 n)$ communication-efficient privacy-preserving range query in fog-based IoT. IEEE Internet Things J. **7**(6), 5220–5232 (2020)
15. Osmanoglu, M., Demir, S., Tugrul, B.: Privacy-preserving k-NN interpolation over two encrypted databases. PeerJ Comput. Sci. **8**, e965 (2022)
16. Park, J., Lee, D.H.: Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments. IEEE Access **8**, 64617–64633 (2020)
17. Qi, Y., Atallah, M.J.: Efficient privacy-preserving k-nearest neighbor search. In: 2008 the 28th International Conference on Distributed Computing Systems, pp. 311–319. IEEE (2008)
18. Ren, C.R., Hu, H., Xu, J., Choi, B.: Processing private queries over untrusted data cloud through privacy homomorphism. In: International Conference on Data Engineering (2011)
19. Samanthula, B.K., Elmehdwi, Y., Jiang, W.: K-nearest neighbor classification over semantically secure encrypted relational data. IEEE Trans. Knowl. Data Eng. **27**(5), 1261–1273 (2014)
20. Shaneck, M., Kim, Y., Kumar, V.: Privacy preserving nearest neighbor search. In: Yu, P.S., Tsai, J.J.P. (eds.) Machine Learning in Cyber Trust, pp. 247–276. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-88735-7_10

21. Skillicorn, D.B., McConnell, S.M.: Distributed prediction from vertically partitioned data. J. Parallel Distrib. Comput. **68**(1), 16–36 (2008)
22. Songhori, E.M., Hussain, S.U., Sadeghi, A.R., Koushanfar, F.: Compacting privacy-preserving k-nearest neighbor search using logic synthesis. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE (2015)
23. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure KNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139–152 (2009)
24. Xie, B., Xiang, T., Liao, X.: Access-oblivious and privacy-preserving k nearest neighbors classification in dual clouds. Comput. Commun. **184**, 12–23 (2022)
25. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (SFCS 1986), pp. 162–167. IEEE (1986)
26. Zantalis, F., Koulouras, G., Karabetsos, S., Kandris, D.: A review of machine learning and IoT in smart transportation. Future Internet **11**(4), 94 (2019)
27. Zheng, Y., Lu, R., Guan, Y., Shao, J., Zhu, H.: Efficient and privacy-preserving similarity range query over encrypted time series data. IEEE Trans. Dependable Secure Comput. **19**(4), 2501–2516 (2021)
28. Zhu, Y., Huang, Z., Takagi, T.: Secure and controllable k-NN query over encrypted cloud data with key confidentiality. J. Parallel Distrib. Comput. **89**, 1–12 (2016)