IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 20, 2016, accepted November 2, 2016, date of publication December 1, 2016, date of current version January 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2633544

Privacy-Preserving k-Nearest Neighbor Computation in Multiple Cloud Environments

HONG RONG, HUI-MEI WANG, JIAN LIU, AND MING XIAN
State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System, National University of Defense Technology, Changsha HN 731, China

Corresponding author: H. Rong (r.hong_nudt@hotmail.com)

ABSTRACT With the advent of big data era, clients lack of computational and storage resources tends to outsource data mining tasks to cloud computing providers in order to improve efficiency and save costs. Generally, different clients choose different cloud companies for the sake of security, business cooperation, location, and so on. However, due to the rise of privacy leakage issues, the data contributed by clients should be encrypted under their own keys. This paper focuses on privacy-preserving k-nearest neighbor (kNN) computation over the databases distributed among multiple cloud environments. Unfortunately, existing secure outsourcing protocols are either restricted to a single key setting or quite inefficient because of frequent client-to-server interactions, making it impractical for wide application. To address these issues, we propose a set of secure building blocks and outsourced collaborative kNN protocol. Theoretical analysis shows that our scheme not only preserves the privacy of distributed databases and kNN query but also hides access patterns in the semi-honest model. Experimental evaluation demonstrates its significant efficiency improvements compared with existing methods.

INDEX TERMS Big data, privacy-preserving data mining, k-nearest neighbor, multiple keys, multiple clouds.

I. INTRODUCTION

As the volume and variety of data captured by organizations or companies are growing more rapidly than ever, resource constrained clients tend to outsource both data and data mining tasks to cloud service providers (CSP) to improve efficiency and save costs. Generally speaking, different clients choose different CSPs because of various considerations on security level, geographical location, budget, etc. Nowadays, there's a growing trend for collaborative data mining, that is, clients who have a common goal to make use of shared information are likely to cooperate in computation over their databases vertically or horizontally partitioned and distributed among multiple clouds. Evidences have shown that this kind of cooperation can improve the accuracy of mining results [1]. For example, aggregating and analyzing e-health records contributed from different hospitals may improve mankind's understanding of certain disease and its treatment methods.

Unfortunately, due to the continuous occurrences of privacy breach in cloud computing [2], [3], concerns about security have significantly impeded the wide adoption of outsourced data mining. To protect confidentiality of data from unauthorized access, sensitive data are usually encrypted by clients before outsourcing. Generally, they tend to use their own keys for encryption, since there's little mutual trust among different parties, and applying independent keys enhances security protection.

In this paper, we focus on the problem of k-Nearest Neighbor (kNN) computation in the multi-cloud environments. The outsourcing protocol identifies the k points nearest to a given query over encrypted databases according to distance measurements like Minkowski or Euclidean distance, and returns an encrypted class based on the majority of the neighboring points. However, the cloud can still deduce sensitive information by observing the data access patterns even if the data are encrypted [4]. Therefore, the kNN protocol should not only guarantee the computation correctness, but also preserve privacy of clients' data, query, result, as well as access patterns.

This outsourced model is similar with distributed privacy-preserving data mining using SMC (Secure Multi-party Computation) technique [5]. But we claim that our problem cannot be solved in that SMC assumes data are not encrypted among multiple parties and results are also revealed. Most recent works [6]–[9] on outsouced kNN computation were based on a single data owner situation. The work in [10] considered multiple distrusted owners and used kernel density estimation instead of kNN to prevent distance-learning attacks. However, all participants employ the same key in this scheme. To conclude, the single key setting faces two problems in multi-cloud scenario: 1) The possible key leakage may jeopardize the privacy of all owners' data. For example, we assume that all data owners share the same key suite for encryption as well as decryption, and outsource the encrypted datasets to the cloud. Suppose one client is corrupted by an adversary, then the adversary is able to decrypt other owners' data by using the shared key if he has intercepted their uploading packets or compromised the storage server. 2) Data owners can't even decrypt their own data downloaded from cloud storage if they encrypt their dataset via the public key generated by cloud, as work in [6]. Hence, it's quite necessary for cloud clients to generate their own public/private keys for privacy protection.

A recent work [11] proposed general-purpose construction by leveraging the two independent decryption mechanisms of BCP cryptosystem to convert ciphertexts under different keys into ciphertexts under single key for arithmetic operations, which yet incurs heavy interactions between servers. To reduce the costs, two schemes were proposed for outsourced computation based on ElGamal encryption and its variation [12], [13]. But its additively homomorphic scheme needs solving discrete logarithm which is considered to be computationally intractable, while its multiplicatively homomorphic scheme may reveal partial privacy under the two-server model. Frameworks in [14] address the collaborative mining in multi-key setting considering different security requirements, but their schemes disclose data owners' symmetric keys explicitly and are not semantic secure. Consequently, current solutions are not secure and efficient enough for collaborative kNN tasks over distributed datasets from multiple clouds.

Main Contributions: In this paper, we propose a set of privacy-preserving building blocks and Outsourced Collaborative kNN (OCkNN) protocol that allows data owners to encrypt data with their own keys while the corresponding clouds can perform kNN over the distributed encrypted database. To the best of our knowledge, there's no prior work that addresses outsourced kNN problem under multi-cloud and multi-key setting. The main contributions of this work are three-fold:

- Our scheme is able to process ciphertexts under multiple keys by re-encryption technique, and returns the correct kNN class label for a given query. Compared with methods in [7], [8], and [14], our solution achieves a higher security level, for it not only ensures the privacy of database, kNN query, mining result, and access patterns not to be revealed to the cloud servers or other parties, but also reduces the risks of key leakage and snooping attacks by adopting separate encryption keys.
- This paper proposes efficient generic privacy-preserving building blocks, including addition, comparison, majority class computation, etc, under two-server model. The proposed secure comparison scheme does not reveal the access patterns (i.e., encrypted records) to cloud server, and it's designed without bit-decomposition, making it faster than the scheme in [6]. We emphasize that our solution does not require the data owners or querists to involve in any kNN computation after uploading their encrypted datasets or query to the cloud. They are allowed to retrieve the data from cloud storage and decrypt them with private keys, which is common for most cloud application. These building blocks are not just fit for this paper, but can be widely adopted in other privacy-preserving data mining protocols.
- The outsourced kNN protocol is constructed based on those building blocks, accelerated by parallel processing framework. We also propose special techniques to optimize the scheme. Theoretical analysis demonstrates that the proposed schemes execute kNN computation correctly, and they're secure under the standard semi-honest model with relatively low computational complexity. Additionally, the communication costs across clouds are minimized. Extensive experiments on real dataset also show significant improvements in efficiency and overhead of our schemes compared with similar works.

The rest of the paper is organized as follows. Our system model and threat model are described in Section II. In Section III, we briefly introduce proxy re-encryption technique. The design details of privacy-preserving building blocks and corresponding OCkNN protocol are presented in Section IV. Then, we evaluate the performance of our schemes in Section V, and we review the related work regarding outsourced privacy-preserving kNN in Section VI. Finally, we summarize the paper and outline future work in Section VII.

II. PROBLEM STATEMENT

In this section, we formally describe our system model, threat model and design objectives.

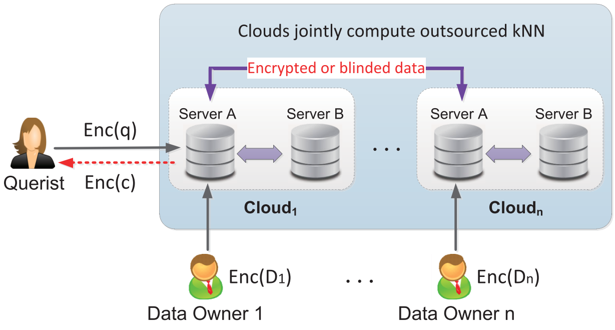

FIGURE 1. System Model.

A. SYSTEM MODEL

In our system model depicted in Fig.1, there are n cloud environments $Cloud_1, \ldots, Cloud_n$, and n data owners $U_1, \ldots, U_n$ who hold their own databases $D_1, \ldots, D_n$ along with their respective public/private key pairs, denoted as $pk_{U_1}/sk_{U_1}, \ldots, pk_{U_n}/sk_{U_n}$. The database D_i has L_i records with $m+1$ attributes for $i \in [1, n]$, in which the $(m+1)^{th}$ attribute contains corresponding class label of that record. In reality, there may be more than one clients for a cloud provider. For simplicity, we assume each cloud has only one data owner, and L_i is equal to L. There's also an authorized querist Q with a kNN query $q =< q_1, \ldots, q_m >$ and its corresponding key pair pk_Q/sk_Q. Each cloud environment is composed of two servers, namely C_i^A and C_i^B. The database is stored on C_i^A who also performs homomorphic computation and acts as a delegate to interact with other clouds. C_i^B is used to assist C_i^A with complex operations over ciphertexts. Let $t_{j,h}^i$ denote the h^{th} attribute value of record t_j^i of D_i for $h \in [1, m+1]$ and $j \in [1, L]$. Initially, U_i encrypts each attribute $t_{j,h}^i$ with pk_{U_i} and acquires the encrypted database $\mathsf{Enc}_{pk_{U_i}}(D_i)$ denoted by D_i' as well, which is then uploaded to C_i^A for storage and kNN classification. Q uses pk_Q to compute $\mathsf{Enc}_{pk_Q}(q)$ denoted by q', which is then submitted to its preferred CSP server, e.g., C_1^A. After all clouds obtain the horizontally partitioned encrypted databases, they begin to evaluate the kNN function $f(D_1', \ldots, D_n', q') = c_q'$ together through cryptographic protocols, where c_q' denotes the encrypted class label for q. The final result is returned to Q under its public key.

Our system model is appropriate and applicable for the following two reasons. On one hand, to protect confidentiality of databases and query, it is essential for data owners and querists to encrypt their data before outsourcing. Besides, the encryptions are conducted by using their own keys, hence reducing the risks of secret key disclosure or being intercepted by other owners or compromised cloud servers (e.g., private key leakage in single-key model may endanger data privacy of all participants [15]). This is also consistent with the security demands of real-world application. On the other hand, the previous work [16] has illustrated that a non-interactive solution is impossible to implement under traditional single-server model, and adopting two non-colluding servers to perform privacy-preserving computation is commonly used to eliminate users' interactions [17]. Data owners take many aspects into account before data outsourcing, such as trustworthiness, economy, bandwidth, etc, so their CSPs are not likely to be the same. Furthermore, since different providers are generally driven by different business model and competing relationship (e.g., Google Compute Engine [23] and Amazon EC2 [24]), the possibility of launching collusion attacks is lowered down.

B. THREAT MODEL

As Fig.1 shows, our threat model primarily includes cloud servers, data owners and querist, communication channel.

1) Cloud Servers: All cloud servers are assumed to be semi-honest (i.e., honest-but-curious), which means that each server strictly follows the protocol, but may try to analyze user's inputs, intermediate results, as well as outputs in order to infer sensitive information. They have no background knowledge of clients' data distribution. In addition, there is no collusion between the two servers within one cloud or among different clouds.

2) Data owners and querist: They are assumed to be semi-honest clients of outsourced kNN service. They can cooperate with other participants for the sake of collaborative data mining, meanwhile they may attempt to gather others' private information. Besides, their online periods are relatively short and non-deterministic. We do not assume any data owner to collude with the cloud.

3) Channel: Communication channel is supposed to be open and insecure. Hence, all transmitted data between servers and clients are likely to be intercepted and analyzed by other parties.

C. DESIGN OBJECTIVES

Given the model above, our design should achieve the following objectives:

- **Correctness**. If the clients and cloud servers both follow the designed protocol, the returned result should be decrypted to a correct class label for the specific kNN query.
- **Confidentiality**. During the outsourcing process, nothing regarding the content of data owners' datasets, query record, mining result, or access patterns should be revealed and inferred by the cloud servers, or other parties. Access pattern is defined as the original encrypted input corresponding to the computed value, e.g., the minimum distance, the most frequent class label, etc.
- **Efficiency**. Due to the users' limited computational resources, their participation should be minimized, while the major workload should be managed by cloud servers efficiently. Meanwhile, the communication across clouds should also be reduced.

III. PRELIMINARIES

Proxy Re-Encryption (PRE) is a useful primitive introduced by Blaze, Bleumer and Strauss [19]. In a PRE system, a proxy is given a re-encryption key $rk_{i \rightarrow j}$ so that it can transform a ciphertext under public key pk_i into a ciphertext of the same plaintext under another user's public key pk_j. The proxy, however, learns nothing in terms of the corresponding plaintext. In this work, only by converting those ciphertexts into ones under a unified key can the cloud servers conduct homomorphic operations over the ciphertexts.

We use the classic bidirectional PRE scheme in [19] because ciphertext conversion from two directions (i.e., from user to server and from server to user) is required. The scheme is constructed on ElGamal cryptosystem, which is secure against chosen-plaintext attacks (CPA). It consists of the following five algorithms [20], [21]:

- $\mathsf{KeyGen}(\mathbb{G}, p, g) \rightarrow \{pk_i, sk_i\}$: Let $\mathbb{G}$ be a multiplicative cyclic group of an order of p, and g be a generator of $\mathbb{G}$. U_i uses this key generation algorithm to generate a key pair $sk_i = a \in Z_p^*$ and $pk_i = g^a \in \mathbb{G}$.

- ReKeyGen$(sk_i, sk_j) \rightarrow \{rk_{i\leftrightarrow j}\}$: The re-encryption key generation algorithm takes two private keys sk_i and sk_j as inputs, and outputs a re-encryption key $rk_{i\leftrightarrow j} = sk_j/sk_i \in \mathbb{Z}_p^*$. Here, it is required that $i \neq j$ in that there's no point to re-encrypt oneself's ciphertext.
- Enc$(pk_i, b) \rightarrow \{CT_i\}$: The encryption algorithm takes a public key pk_i and a message $b \in \mathcal{M}$ as inputs. It outputs an ciphertext $CT_i = (b \cdot g^r, pk_i^r)$ under pk_i. Here, $\mathcal{M}$ denotes the message space, and r is a random number generated from $\mathbb{Z}_p^*$. Let the notion $r \in_R \mathbb{Z}_p^*$ denote the random number generation hereafter.
- ReEnc$(rk_{i\leftrightarrow j}, CT_i) \rightarrow \{CT_j\}$: The re-encryption algorithm takes a re-encryption key $rk_{i\leftrightarrow j}$ and an original ciphertext CT_i as inputs, and outputs a transformed ciphertext $CT_j = (b \cdot g^r, (pk_i^r)^{rk_{i\leftrightarrow j}})$ under pk_j.
- Dec$(sk_i, CT_i) \rightarrow \{m\}$: The decryption algorithm takes a private key sk_i and an original or converted ciphertext CT_i under public key pk_i. It outputs a plaintext message $b \leftarrow b \cdot g^r/((pk_i^r)^{1/sk_i})$.

Moreover, ElGamal encryption has multiplicatively homomorphic property over ciphertexts. More specifically, we have

$$\mathsf{Enc}_{pk}(b_1) \times \mathsf{Enc}_{pk}(b_2) = \mathsf{Enc}_{pk}(b_1 \cdot b_2), \tag{1}$$

Here, "$\cdot$" denotes the multiplication operation in the plaintext domain while "$\times$" denotes the multiplication operation in the ciphertext domain.

IV. PROPOSED OUTSOURCED COLLABORATIVE kNN SCHEME

In this section, we first present the design details of secure building blocks. Then, the complete outsourcing protocol based on these build blocks is illustrated. The security and complexity of the proposed scheme are further analyzed.

A. OVERVIEW

With the assumption of semi-honest, but non-colluding cloud environments, C_i^A for $1 \leq i \leq n$ in all clouds starts to negotiate the public system parameters, based on which U_i, Q and C_i^B generate their respective public and private key pairs. After that, servers and clients jointly compute the re-encryption keys via key distribution protocol. C_i^A also generates re-encryption keys for B-servers from other clouds. After U_i's encrypted database under pk_i is uploaded to its corresponding CSP, C_i^A transforms all the ciphertexts into encryptions under a unified key (C_i^B's key) using the property of PRE scheme. Let D'_{dis} denote the distributed encrypted databases under multiple keys. After Q submits its encrypted query q' to its trustworthy cloud denoted by $Cloud_j$ ($j \in [1, n]$), this cloud performs as primary node temporarily in the distributed computing model while others act as slave nodes. $Cloud_j$ first delivers the query to other cloud servers, and then all clouds execute the kNN computation locally through a set of our proposed cryptographic building blocks. The intermediate results from all nodes are converged and processed at $Cloud_j$ to calculate the k globally nearest points over D'_{dis}. The final class label under the unified key should be converted back to the ciphertext under Q's public key. Finally, Q retrieves the encrypted output with its private key. Note that the outsourced kNN computation part is performed with no participations of data owners and querist whatsoever.

B. PRIVACY-PRESERVING BUILDING BLOCKS

In this part, we propose a set of privacy-preserving building blocks, including key distribution, addition, comparison, and majority class computation. They serve as the basic constructions of the collaborative outsourced kNN solution.

1) THE KEY DISTRIBUTION (KD) PROTOCOL

At first, all clouds run together a setup process that initializes the ElGamal cryptosystem to generate system parameters $\{\mathbb{G}, p, g\}$, and distribute them to their clients. By computing KeyGen$(\mathbb{G}, p, g)$, all kNN parties generate their own key pairs, including U_i's $\{pk_{U_i}, sk_{U_i}\}$, Q's $\{pk_Q, sk_Q\}$ and unified key $\{pk_{C_i}, sk_{C_i}\}$ of server C_i^B.

There are three kinds of re-encryption keys in our system, the first is used for ciphertexts transformation between U_i and C_i^B for $i \in [1, n]$; the second is for conversion between Q and C_i^B; while the third is used between different clouds, namely, C_i^B and C_j^B, in which $i \neq j$. Let $rk_{U_i\leftrightarrow C_i}$, $rk_{Q\leftrightarrow C_i}$, $rk_{C_i\leftrightarrow C_j}$ denote these three keys, respectively. Even though the process of key generation may require interactions either within the local cloud or across distinct clouds, the methods to generate re-encryption keys are basically the same. Let's take re-encryption key between U_i and C_i^B as an example, C_i^A first generates $r_i \in_R \mathbb{Z}_p^*$, and distributes it to U_i; after that, U_i computes r_i/sk_{U_i} and sends it to C_i^B who obtains $sk_{C_i} \cdot r_i/sk_{U_i}$ and returns it to C_i^A; thus, C_i^A gets $rk_{U_i\leftrightarrow C_i}$ by computing $r_i^{-1} \cdot sk_{C_i} \cdot r_i/sk_{U_i}$. Note that C_i^A merely knows the re-encryption keys, and C_i^B knows nothing regarding the private keys of other parties during execution of this protocol, because intermediate results are blinded by random number r_i. Besides, the overall communication is protected by secure protocol like SSL.

2) THE SECURE ADDITION (SA) PROTOCOL

Assume that C_i^A holds private inputs $\{\mathsf{Enc}_{pk_{C_i}}(a), \mathsf{Enc}_{pk_{C_i}}(b)\}$, and C_i^B holds sk_{C_i}. The goal of this protocol is to compute the encrypted addition of a and b, i.e., $\mathsf{Enc}_{pk_{C_i}}(a+b)$ as output to C_i^A. As the encryption system is not additively homomorphic, it requires interactions between the two cloud servers. Nevertheless, it has been proven that the two-server setting may disclose private information about inputs [12]. If we send $\mathsf{Enc}_{pk_{C_i}}(ra)$ and $\mathsf{Enc}_{pk_{C_i}}(rb)$ to C_i^B, the blinding factor r can be removed by computing $ra/rb \rightarrow a/b$, even if the exact inputs are unknown to C_i^B. Thereby, the compromised C_i^B is able to identify the inputs if it possesses background knowledge of raw data distribution.

In this paper, we still consider the two semi-honest servers model while try to lower down the opportunities that the corrupted server can recover true proportional relationship of inputs to enhance data privacy. The overall steps of SA are presented in Algorithm 1.

```
Algorithm 1 SA($\mathsf{Enc}_{pk_{C_i}}(a)$, $\mathsf{Enc}_{pk_{C_i}}(b)$) $\rightarrow$ $\mathsf{Enc}_{pk_{C_i}}(a+b)$
Require: $C_i^A$ has $\mathsf{Enc}_{pk_{C_i}}(a)$ and $\mathsf{Enc}_{pk_{C_i}}(b)$; $C_i^B$ has $sk_{C_i}$,
    where $i \in [1, n]$. Initialize $k \leftarrow 1$.
    $C_i^A$:
      1) Generate random values $\alpha, \beta, r_1, r_2, \omega_1, \ldots, \omega_f \in_R \mathbb{G}$;
      2) Compute encrypted elements in a vector, denoted by $L_j$ $(j = 1, 2, \ldots, 4+f)$:
         • $\mathsf{Enc}_{pk_{C_i}}(r_1\beta a) \leftarrow \mathsf{Blind}(\mathsf{Enc}_{pk_{C_i}}(a), r_1\beta)$;
         • $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha) \leftarrow \mathsf{Enc}(pk_{C_i}, r_1\alpha)$;
         • $\mathsf{Enc}_{pk_{C_i}}(r_2\beta b) \leftarrow \mathsf{Blind}(\mathsf{Enc}_{pk_{C_i}}(b), r_2\beta)$;
         • $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha) \leftarrow \mathsf{Enc}(pk_{C_i}, -r_2\alpha)$;
         • $\mathsf{Enc}_{pk_{C_i}}(\omega_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(\omega_f)$;
         • $\Gamma \leftarrow \{L_j | j = 1, 2, \ldots, 4+f\}$;
         • $I \leftarrow \{1, 2, \ldots, 4+f\}$;
      3) Generate a random permutation function $\pi$;
         $\Gamma' \leftarrow \pi(\Gamma)$ and $I' \leftarrow \pi(I)$;
      4) Send $\Gamma'$ above to $C_i^B$;
    $C_i^B$:
      1) Receive encrypted results $\Gamma'$ from $C_i^A$;
      2) Decryption: $L_j' \leftarrow \mathsf{Dec}(sk_{C_i}, \Gamma'[j])$, for $1 \leq j \leq 4+f$;
      3) for $l = 1$ to $|\Gamma'| - 1$ do:
         • for $j = l + 1$ to $|\Gamma|$ do:
           – $S_k \leftarrow L_l' + L_j'$; $k \leftarrow k + 1$;
      4) Encryption: $S_j' \leftarrow \mathsf{Enc}(pk_{C_i}, S_j)$, for $1 \leq j \leq k$;
      5) Send $S'$ to $C_i^A$;
    $C_i^A$:
      1) Receive encrypted set $S'$ from $C_i^B$;
      2) $index_1 \leftarrow$ FindSumIndex($I'$, $\{1, 2\}$);
         $\gamma_1 \leftarrow \mathsf{Blind}(S'[index_1], r_1^{-1})$;
      3) $index_2 \leftarrow$ FindSumIndex($I'$, $\{3, 4\}$);
         $\gamma_2 \leftarrow \mathsf{Blind}(S'[index_2], r_2^{-1})$;
      4) Send $\gamma_1$, $\gamma_2$ to $C_i^B$;
    $C_i^B$:
      1) Receive $\gamma_1$, $\gamma_2$ from $C_i^A$;
      2) Decryption: $\gamma_j' \leftarrow \mathsf{Dec}(sk_{C_i}, \gamma_j)$, for $j = 1, 2$;
      3) $\lambda \leftarrow \gamma_1' + \gamma_2'$;
         Encryption: $\lambda' \leftarrow \mathsf{Enc}(pk_{C_i}, \lambda)$;
      4) Send $\lambda'$ to $C_i^A$;
    $C_i^A$:
      1) Receive $\lambda'$ from $C_i^B$;
      2) $\mathsf{Enc}_{pk_{C_i}}(a+b) \leftarrow \mathsf{Blind}(\lambda', \beta^{-1})$;
```

First, C_i^A generates random numbers: α, β, r_1, r_2, ω_f from $\mathbb{G}$, and computes multiplications: $r_1\beta$, $r_1\alpha$, $r_2\beta$, $-r_2\alpha$. Then, C_i^A computes a vector Γ containing $\mathsf{Enc}_{pk_{C_i}}(r_1\beta a)$, $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha)$, $\mathsf{Enc}_{pk_{C_i}}(r_2\beta b)$, $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha)$, as well as $\mathsf{Enc}_{pk_{C_i}}(\omega_f)$ by using ElGamal encryption and Blind operation. $\mathsf{Blind}(CT, r)$ is an operation that randomizes ct_1 of ciphertext CT by multiplying a random value r so that the plaintext b is randomized by r, where $CT = (ct_1, ct_2)$, $ct_1 = bg^{r'}$, $r, b \in \mathbb{G}$, and $r' \in Z_p^*$. This operation only requires one modular multiplication over $\mathbb{G}$.

After that, the vector Γ is permuted into Γ' using random permutation function π and sent to C_i^B. Upon receiving Γ', C_i^B begins to decrypt the vector elements by its private key sk_{C_i}. C_i^B then calculates the sums between any two blinded values according to a negotiated fashion, and encrypts each sum S_k under pk_{C_i} by a given order. The encrypted sum set S' is returned to C_i^A. After receiving S', C_i^A is able to locate the required encrypted sum, namely, $\mathsf{Enc}_{pk_{C_i}}(r_1\beta a + r_1\alpha)$ and $\mathsf{Enc}_{pk_{C_i}}(r_2\beta b - r_2\alpha)$ by FindSumIndex. As shown in Algorithm 2, the function "FindSumIndex" aims at finding the position of a specific sum in disordered dataset according to the permuted index set I' and subscripts of the required sum. The blinding values r_1, r_2 are removed from two sums by homomorphic multiplication, resulting in $\mathsf{Enc}_{pk_{C_i}}(\beta a+\alpha)$ and $\mathsf{Enc}_{pk_{C_i}}(\beta b - \alpha)$, denoted by γ_1, γ_2, respectively. They are transferred to C_i^B. After decrypting the received γ_1, γ_2, C_i^B computes $\beta a + \beta b$ by addition and delivers its encryption to C_i^A.

In the end, C_i^A can recover the encrypted value of $a + b$ by Blind with the multiplicative inverse of β.

```
Algorithm 2 FindSumIndex($I$, $\{a, b\}$) $\rightarrow$ $index$
Require: $C_i^A$ has permuted index set $I$ and original subscript
    set $\{a, b\}$, where $i \in [1, n]$.
    $C_i^A$, for $k = 1$ to $|I| - 1$ do:
      1) for $j = k + 1$ to $|I|$ do:
         • if $(I[k] = a \wedge I[j] = b) \vee (I[k] = b \wedge I[j] = a)$
           then:
           – $index \leftarrow (k-1)|I| - k(k-1)/2 + j - k$;
           – break;
```

Security Analysis of SA: The security of SA protocol is discussed based on "Real-vs.-Ideal" framework [22]. By this framework, we need to show that SA is secure against both adversary $\mathcal{A}_A$ corrupting C_i^A and $\mathcal{A}_B$ corrupting C_i^B in the real world under the semi-honest model.

1) Security Against Cloud Server C_i^A: During SA protocol, the view of semi-honest adversary $\mathcal{A}_A$ includes the inputs $\{\mathsf{Enc}_{pk_{C_i}}(a)$, $\mathsf{Enc}_{pk_{C_i}}(b)$, S', $\lambda'\}$ and output $\mathsf{Enc}_{pk_{C_i}}(a+b)$, all of which are encrypted under pk_{C_i}.

We build a simulator $\mathcal{F}$ computes $\mathsf{Enc}_{pk_{C_i}}(a')$, $\mathsf{Enc}_{pk_{C_i}}(b')$, assuming that $a' = 1$ and $b' = 2$ without loss of generality. Then, it generates random numbers α', β', r_1', r_2', $\omega_1', \ldots, \omega_f'$, and computes sums like the algorithm: $\mathsf{Enc}_{pk_{C_i}}(r_1'\beta' \cdot 1 + r_1'\alpha')$, $\mathsf{Enc}_{pk_{C_i}}(r_2'\beta' \cdot 2 - r_2'\alpha')$, etc. It also evaluates $\mathsf{Enc}_{pk_{C_i}}(\beta' \cdot 1 + \beta' \cdot 2)$ and $\mathsf{Enc}_{pk_{C_i}}(1+2)$. Finally, $\mathcal{F}$ returns those ciphertexts and random values to C_i^A.

Since C_i^A has no knowledge of sk_{C_i}, it's computationally hard for $\mathcal{A}_A$ to distinguish the ciphertexts in the views of real world and ideal world by the semantic security of the encryption scheme. Therefore, we have

$$\mathrm{Ideal}_{f,\mathcal{F}}(\mathsf{Enc}_{pk_{C_i}}(b_i)) \overset{c}{\approx} \mathrm{Real}_{\mathrm{SA},\mathcal{A}_A}(\mathsf{Enc}_{pk_{C_i}}(b_i)) \tag{2}$$

where $i \in \{1, 2\}$.

2) Security Against Cloud Server C_i^B: The computation task assigned to C_i^B can be divided into two rounds. In the first round, the view of $\mathcal{A}_B$ includes input $\{\mathsf{Enc}_{pk_{C_i}}(r_1\beta a)$, $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha)$, $\mathsf{Enc}_{pk_{C_i}}(r_2\beta b)$, $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha)$, $\mathsf{Enc}_{pk_{C_i}}(\omega_f)\}$, blinded messages $\{r_1\beta a, r_2\beta b, -r_2\alpha, r_1\alpha, \omega_f\}$, and output S'. Intuitively, $\mathcal{A}_B$ cannot obtain either a, b or the ratio a/b directly, because they are randomized by the C_i^A's generated random numbers $\alpha,\ \beta,\ r_1,\ r_2,\ \omega_f$. But there still exists some correlations among these blinded values, possibly leading to leakage of a/b, which can be computed by the following

$$\frac{a}{b} = -\frac{r_1\beta a \cdot (-r_2\alpha)}{r_2\beta b \cdot r_1\alpha}. \tag{3}$$

From Eq.(3), $\mathcal{A}_B$ can make a guess by picking random combinations from the blinded messages. During the second round, the view of $\mathcal{A}_B$ comprises input $\{\gamma_1, \gamma_2\}$, blinded messages $\{\beta a + \alpha, \beta b - \alpha\}$, output λ'. Although $\mathcal{A}_B$ is able to decrypt the ciphertexts in its view with sk_{C_i}, the decrypted messages are still blinded by random numbers.

Note that the ratio of inputs can only be deduced by $\mathcal{A}_B$ in an oblivious way. However, $\mathcal{A}_B$ is not able to distinguish the real world from the ideal world on condition that he does not know the input distribution, even if the ratio is obtained. We can build a simulator $\mathcal{F}$ in the ideal world that uses $a' = 1$ and $b' = 2$ as plaintexts, and generates randomized values similar with the algorithm, and returns them to $\mathcal{A}_B$. Therefore, we have the following

$$\mathrm{Ideal}_{f,\mathcal{F}}(\mathsf{Enc}_{pk_{C_i}}(r_ib_i)) \overset{c}{\approx} \mathrm{Real}_{\mathrm{SA},\mathcal{A}_B}(\mathsf{Enc}_{pk_{C_i}}(r_ib_i)) \tag{4}$$

where $i \in \{1, 2\}$, $r_i \in_R \mathbb{G}$, b_i is the plaintext.

Discussions: It's evident that the security level of SA depends on how many random numbers like ω_f are inserted in the set Γ. Suppose that C_i^A generates f random values from $\mathbb{G}$, then the possibility of a successful guess is

$$\varepsilon = \frac{1}{C_{f+4}^4 \cdot C_4^2} = \frac{4}{f^4 + 10f^3 + 35f^2 + 50f + 24} \tag{5}$$

based on Eq.(3). It can be easily seen that the difficulty for $\mathcal{A}_B$ to get the ratio corresponds to 4 power of f. However, along with the growing f, the costs to process these ciphertexts are also on the rise quadratically. Thus, it is critical to make a tradeoff between outsourcing efficiency and data security to suffice different objectives.

3) THE SECURE SQUARED DISTANCE (SSD) PROTOCOL

Given that C_i^A holds $A' =< \mathsf{Enc}_{pk_{C_i}}(a_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(a_m) >$ and $B' =< \mathsf{Enc}_{pk_{C_i}}(b_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(b_m) >$, while C_i^B holds the secret key sk_{C_i}, the output is the encryption of squared Euclidean distance $\mathsf{Enc}_{pk_{C_i}}(\sum_{l=1}^m (a_l - b_l)^2)$. First, the servers compute the differences between elements of the two encrypted vectors by SA protocol. Then the differences are squared by C_i^A invoking multiplicatively homomorphic scheme. The final result is calculated by adding the squared differences via SA. The detailed steps are shown in Algorithm 3. According to the Composition Theorem [22], this protocol does not reveal any privacy about inputs and outputs as long as SA protocol is secure, which has been illustrated in previous sub-protocol.

Algorithm 3 SSD$(A', B') \rightarrow E'_{A,B}$

```
Require: $C_i^A$ has $A' =< \mathsf{Enc}_{pk_{C_i}}(a_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(a_m) >$
    and $B' =< \mathsf{Enc}_{pk_{C_i}}(b_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(b_m) >$; $C_i^B$ has $sk_{C_i}$,
    where $i \in [1, n]$.
    $C_i^A$ and $C_i^B$:
      1) for $l = 1$ to $m$ do:
           • $b'_l \leftarrow \mathsf{Blind}(\mathsf{Enc}_{pk_{C_i}}(b_l), -1)$;
           • $\mathsf{Enc}_{pk_{C_i}}(a_l - b_l) \leftarrow \mathrm{SA}(\mathsf{Enc}_{pk_{C_i}}(a_l), b'_l)$;
    $C_i^A$:
      1) for $l = 1$ to $m$ do:
           • $\lambda_l \leftarrow \mathsf{Enc}_{pk_{C_i}}(a_l - b_l) \times \mathsf{Enc}_{pk_{C_i}}(a_l - b_l)$;
    $C_i^A$ and $C_i^B$:
      1) Initialize $E'_{A,B} \leftarrow \mathsf{Enc}_{pk_{C_i}}(0)$;
      2) for $l = 1$ to $m$ do:
           • $E'_{A,B} \leftarrow \mathrm{SA}(E'_{A,B}, \lambda_l)$;
```

4) THE SECURE MINIMUM BETWEEN TWO NUMBERS (SM2N) PROTOCOL

Given that C_i^A holds two private input $\{A', B'\}$, and C_i^B holds the secret key sk_{C_i}, where A', B' are encrypted pairs like $A' = (\mathsf{Enc}_{pk_{C_i}}(a), \mathsf{Enc}_{pk_{C_i}}(s_a))$, $B' = (\mathsf{Enc}_{pk_{C_i}}(b), \mathsf{Enc}_{pk_{C_i}}(s_b))$, the output is the encryption of the encrypted minimum between input values. Here, s_a and s_a denote the secrets associated with a and b which are to be compared. A good case in point is the encrypted class label. The main idea of this protocol is to utilize oblivious mechanisms to judge the sign of difference between inputs. At the end of the execution, C_i^A obtains the ciphertext of the smallest value and its secret while two servers learn nothing about input privacy, and access patterns, including the difference.

The complete steps are presented in Algorithm 4. At first, C_i^A generates a random number λ and computes the encrypted input difference $\mathsf{Enc}_{pk_{C_i}}(a - b)$, denoted by α, if λ is odd; otherwise, it computes $\mathsf{Enc}_{pk_{C_i}}(b - a)$. Then, α is further blinded with a positive random value r which is not so large to make integer overflow, and α' is sent to C_i^B for judgement. C_i^B initially decrypts α' with its private key. If the decrypted value is positive, which means the minuend is larger, it returns $\mathsf{Enc}_{pk_{C_i}}(1)$ as comparison outcome denoted by σ; and returns $\mathsf{Enc}_{pk_{C_i}}(2)$ otherwise. Depending on σ and λ, C_i^A is able to compute encrypted minimum as follows: If λ is odd, compute the encrypted minimum denoted by $\min'(a, b)$, which is

$$\begin{aligned}&\mathsf{Enc}_{pk_{C_i}}((\varphi - 1) \cdot a + (1 - (\varphi - 1)) \cdot b)\\ &\quad = \mathsf{Enc}_{pk_{C_i}}(\varphi \cdot a - \varphi \cdot b + 2 \cdot b - a);\end{aligned} \tag{6}$$

Algorithm 4 SM2N$(A', B') \rightarrow (\min'(a,b), s'_{\min(a,b)})$

```
Require: $C_i^A$ has $A' = (\mathsf{Enc}_{pk_{C_i}}(a), \mathsf{Enc}_{pk_{C_i}}(s_a))$, $B' = (\mathsf{Enc}_{pk_{C_i}}(b), \mathsf{Enc}_{pk_{C_i}}(s_b))$; $C_i^B$ has $sk_{C_i}$, where $i \in [1, n]$.
  $C_i^A$:
    1) Generate a random number $\lambda \in_R \mathbb{G}$;
    2) if $\lambda$ is odd then:
         • $\alpha \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(a), \mathsf{Enc}_{pk_{C_i}}(-b)$) with $C_i^B$;
    3) else
         • $\alpha \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(b), \mathsf{Enc}_{pk_{C_i}}(-a)$) with $C_i^B$;
    4) Generate a small random number $r > 0$, $r \in_R \mathbb{G}$;
    5) $\alpha' \leftarrow \mathbf{Blind}(\alpha, r)$; Send $\alpha'$ to $C_i^B$;
  $C_i^B$:
    1) Receive $\alpha'$ from $C_i^A$, and initialize $\sigma' \leftarrow \mathsf{Enc}_{pk_{C_i}}(2)$;
    2) if $\mathbf{Dec}(sk_{C_i}, \alpha') > 0$ then $\sigma' \leftarrow \mathsf{Enc}_{pk_{C_i}}(1)$;
       Send $\sigma'$ to $C_i^A$;
  $C_i^A$:
    1) Receive $\sigma$ from $C_i^B$;
    2) if $\lambda$ is odd then:
         • $\min'(a,b) \leftarrow$ ComputeMinValue($a', b', \sigma'$);
         • $s'_{\min(a,b)} \leftarrow$ ComputeMinValue($s'_a, s'_b, \sigma'$);
    3) else
         • $\min'(a,b) \leftarrow$ ComputeMinValue($b', a', \sigma'$);
         • $s'_{\min(a,b)} \leftarrow$ ComputeMinValue($s'_b, s'_a, \sigma'$);

  ComputeMinValue ($\mathsf{Enc}_{pk_{C_i}}(u), \mathsf{Enc}_{pk_{C_i}}(v), \mathsf{Enc}_{pk_{C_i}}(\varphi)$)
  $C_i^A$ and $C_i^B$:
    • $\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot u) \leftarrow \mathsf{Enc}_{pk_{C_i}}(\varphi) \times \mathsf{Enc}_{pk_{C_i}}(u)$;
    • $\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot v) \leftarrow \mathsf{Enc}_{pk_{C_i}}(\varphi) \times \mathsf{Enc}_{pk_{C_i}}(v)$;
    • $\mathsf{Enc}_{pk_{C_i}}(\varphi u - \varphi v) \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot u), \mathsf{Enc}_{pk_{C_i}}(-\varphi \cdot v)$);
    • $\mathsf{Enc}_{pk_{C_i}}(2v - u) \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(2v), \mathsf{Enc}_{pk_{C_i}}(-u)$);
    • $\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot u - \varphi \cdot v + 2v - u) \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot u - \varphi \cdot v), \mathsf{Enc}_{pk_{C_i}}(2v - u)$);
    • Return $\mathsf{Enc}_{pk_{C_i}}(\varphi \cdot u - \varphi \cdot v + 2v - u)$;
```

Otherwise, $\min'(a,b) \leftarrow \mathsf{Enc}_{pk_{C_i}}((\varphi - 1) \cdot b + (1 - (\varphi - 1)) \cdot a)$. Here, φ is the plaintext of σ. The above computation is expressed as a function called Compute MinValue, shown at the bottom of Algorithm 4. For example, if $\varphi = 1$ and $\lambda\%2 = 1$, $\min'(a,b) = \mathsf{Enc}_{pk_{C_i}}(0 \cdot a + 1 \cdot b) = \mathsf{Enc}_{pk_{C_i}}(b)$. It's obvious that the observation is correct, because $\varphi = 1$ indicates b is the minimum value. Let $s'_{\min(a,b)}$ be the encrypted secret. Similarly, $s'_{\min(a,b)}$ corresponding to the smaller input can also be computed following this manner.

Security Analysis of SM2N: This protocol utilizes SA scheme to compute the difference of inputs, which is blinded by a positive random number r. Hence, C_i^B cannot know the exact difference of inputs except its sign. As the order of subtraction inputs varies with λ's parity, it's rather difficult for C_i^B to speculate the relationship between inputs. Remark that C_i^A cannot distinguish $\mathsf{Enc}_{pk_{C_i}}(1)$ and $\mathsf{Enc}_{pk_{C_i}}(2)$ because of probabilistic property of the encryption scheme. Other intermediate results and outputs are also preserved as long as SA and multiplicative homomorphism are secure in the semi-honest model. Additionally, C_i^A does not know which encrypted data corresponds to the smallest value, since $\min'(a,b)$, $s'_{\min(a,b)}$ are freshly computed ciphertexts, differing from encrypted inputs for every comparison operation. Therefore, combined with security discussion mentioned earlier, SM2N does not disclose any privacy of user's data or access pattern.

5) THE SECURE MINIMUM IN n NUMBERS (SMnN) PROTOCOL

Assume that C_i^A with inputs $\{A'_1, A'_2, \ldots, A'_n\}$ interacts with C_i^B securely to compute minimum from n inputs, where $A_j = (\mathsf{Enc}_{pk_{C_i}}(a_j), \mathsf{Enc}_{pk_{C_i}}(s_j))$ and $j \in [1, n]$. The main goal of the SMnN protocol is to compute the encryption of the minimum value and its associated secret, i.e., $\min'(a_1, \ldots, a_n)$ and $s'_{\min(a_1,\ldots,a_n)}$, without revealing any information about a_j and s_j to C_i^A and C_i^B. SMnN is designed based on SM2N as the secure building block, and any generic sort algorithms can be applied to SMnN. We leverage heap sort algorithm in comparing operations to improve efficiency, the complexity of which is $O(\log n)$. Due to space limitations, the complete presentation of SMnN is omitted. Furthermore, no privacy or access patterns of input data is revealed to any two cloud servers during execution of SMnN, since the security of SMnN relies on SM2N, which has been proven secure.

6) THE SECURE MAJORITY CLASS COMPUTATION (SMCC) PROTOCOL

C_i^A and C_i^B jointly compute encrypted majority class label based on the encrypted kNN class sets, where $\Theta' =< \mathsf{Enc}_{pk_{C_i}}(c'_1), \mathsf{Enc}_{pk_{C_i}}(c'_2), \ldots, \mathsf{Enc}_{pk_{C_i}}(c'_k) >$. We also assume that C_i^A knows the encrypted vector of each class label as background knowledge, i.e., $\Theta =< \mathsf{Enc}_{pk_{C_i}}(c_1), \ldots, \mathsf{Enc}_{pk_{C_i}}(c_\theta) >$ in advance. Here, c'_l denotes the class label of l^{th} closest neighbor to query q, for $1 \leq l \leq k$ while c_j denotes the unique class label in the joint database for $1 \leq j \leq \theta$. For simplicity, c'_l and c_j are numeric values. To avoid encryption of zero in frequency counting phase, c_j is added with an offset ϕ beforehand which is merely known to C_i^B. Obviously, $(c'_l + \phi) \in \{c_1, \ldots, c_\theta\}$. During the SMCC protocol, the encrypted class label as output denoted by c'_{maj} is revealed only to C_i^A whereas neither c'_l nor c_j is revealed to C_i^A and C_i^B. Besides, C_i^A does not know which data record corresponds to c'_{maj} in order to conceal access patterns.

The overall steps involved in the SMCC protocol are shown in Algorithm 5. In the beginning, C_i^A generates a permutation function π, which is used to disorder the arrangement of Θ into another vector Λ. Then, C_i^A and C_i^B cooperate to compute such matrix S that each element is encrypted difference between Λ and Θ', i.e., $S_{l,j} = \mathsf{Enc}_{pk_{C_i}}(c'_l - c_j)$, for $1 \leq l \leq k$ and $1 \leq j \leq \theta$. The matrix S is then sent to C_i^B . After receiving S, C_i^B decrypts every component of S and computes the appearance frequency for each class.

```
Algorithm 5 SMCC($\Theta, \Theta'$) $\rightarrow c'_{maj}$
Require: $C_i^A$ has $\Theta = < \mathbf{Enc}_{pk_{C_i}}(c_1), \ldots, \mathbf{Enc}_{pk_{C_i}}(c_\theta) >$
    and $\Theta' = < \mathbf{Enc}_{pk_{C_i}}(c'_1), \ldots, \mathbf{Enc}_{pk_{C_i}}(c'_k) >$; $C_i^B$ has $sk_{C_i}$,
    where $i \in [1, n]$. Initialize $\eta_1, \eta_2, fr \leftarrow 0$.
  $C_i^A$:
    1) for $l = 1$ to $\theta$ do:
       • $\Theta_l \leftarrow$ SA($\Theta_l$, $\mathbf{Enc}_{pk_{C_i}}(\phi)$) with $C_i^B$;
    2) Generate a permutation function $\pi$;
    3) $\Lambda \leftarrow \pi(\Theta)$;
    4) for $l = 1$ to $k$ do:
       • $\varepsilon \leftarrow$ Blind($\mathbf{Enc}_{pk_{C_i}}(c'_l)$, $-1$);
       • for $j = 1$ to $\theta$ do:
         – $S_{l,j} \leftarrow$ SA($\Lambda_j, \varepsilon$) with $C_i^B$;
    5) Send $S$ to $C_i^B$;
  $C_i^B$:
    1) for $j = 1$ to $\theta$ do:
       • for $l = 1$ to $k$ do:
         – if Dec($sk_{C_i}, S_{l,j}$) $= \phi$ then $fr_j \leftarrow fr_j + 1$;
    2) for $j = 2$ to $\theta$ do:
       • if $fr_{max} < fr_j$ then:
         – $\rho \leftarrow j$; $fr_{max} \leftarrow fr_j$;
    3) $V_l \leftarrow \mathbf{Enc}_{pk_{C_i}}(2)$, for $l = \rho$; otherwise, $V_l \leftarrow$
       $\mathbf{Enc}_{pk_{C_i}}(1)$, for $l \neq \rho$ and $1 \leq l \leq \theta$;
    4) Send $V$ to $C_i^A$;
  $C_i^A$:
    1) $V' \leftarrow \pi^{-1}(V)$, where $V' = \{\mathbf{Enc}_{pk_{C_i}}(\tau_l) | 1 \leq l \leq \theta\}$;
       $\mathbf{Enc}_{pk_{C_i}}(\tau_l \cdot c_l) \leftarrow \mathbf{Enc}_{pk_{C_i}}(\tau_l) \times \mathbf{Enc}_{pk_{C_i}}(c_l)$, for $1 \leq$
       $l \leq \theta$;
    2) $\mathbf{Enc}_{pk_{C_i}}(\eta_1) \leftarrow$ SA($\mathbf{Enc}_{pk_{C_i}}(\tau_l \cdot c_l)$, $\mathbf{Enc}_{pk_{C_i}}(\eta_1)$) with
       $C_i^B$, for $1 \leq l \leq \theta$;
    3) $\mathbf{Enc}_{pk_{C_i}}(\eta_2) \leftarrow$ SA($\mathbf{Enc}_{pk_{C_i}}(c_l)$, $\mathbf{Enc}_{pk_{C_i}}(\eta_2)$) with
       $C_i^B$, for $1 \leq l \leq \theta$;
    4) $c'_{maj} \leftarrow$ SA($\mathbf{Enc}_{pk_{C_i}}(\eta_1)$, $\mathbf{Enc}_{pk_{C_i}}(-\eta_2)$) with $C_i^B$;
```

It can be easily observed that each row of S_i must contains merely one encryption of ϕ, and $\theta - 1$ encryptions of random values based on the fact $(c'_l + \phi) \in \{c_1, \ldots, c_\theta\}$. Therefore, C_i^B is able to calculate a frequency vector fr, the element of which corresponds to the frequency of that class. After that, the index of most frequent class in Λ is computed, and C_i^B returns C_i^A an encrypted vector V in which the frequent index is the ciphertext of 2 and the rest are ciphertexts of 1. Upon receiving V, C_i^A reorders it through the inverse of π and computes the following formula:

$$\begin{aligned}\mathbf{Enc}_{pk_{C_i}}(c_{maj}) &= \mathbf{Enc}_{pk_{C_i}}(\sum_{l=1}^{\theta} c_l \cdot (\tau_l - 1)) \\ &= \mathbf{Enc}_{pk_{C_i}}(\sum_{l=1}^{\theta} c_l \cdot \tau_l - \sum_{l=1}^{\theta} c_l). \qquad (7)\end{aligned}$$

The scalar product between $< c_1, \ldots, c_\theta >$ and $< \tau_1 - 1, \ldots, \tau_\theta - 1 >$ is the majority class label, because there's only one position of 1 in $< \tau_1 - 1, \ldots, \tau_\theta - 1 >$ which indicates the most frequent class and the rest is 0. So the multiplication preserves the majority class. Eq.(7) can be divided into three parts, i.e., $\Sigma_{l=1}^{\theta} c_l \cdot \tau_l$, $\Sigma_{l=1}^{\theta} c_l$, and their subtraction, which are computed separately to prevent encryption of zero.

Security Analysis of SMCC: During the first round computation of C_i^A, its inputs $\{\Theta, \Theta'\}$ and processed results $\{\Lambda, S\}$ are all encrypted under C_i^B's private key. During the second round computation of C_i^A, the input V and final result $\mathbf{Enc}_{pk_{C_i}}(c_{maj})$ are also encrypted values. Due to security of ElGamal and SA scheme, these ciphertexts are computationally indistinguishable from random numbers in $\mathbb{G}$. As for security against corrupted C_i^B, even though C_i^B can attain the frequencies of each class label, it infers neither the exact class label values nor the index corresponds to that class, in that the original order of Θ is altered by π. Therefore, we can build a simulator in the ideal world that is computationally indistinguishable from real world based on [22]. In other words, SMCC does not disclose anything about the class labels. Besides, since fr is only known to C_i^B, the record of actual majority class is oblivious to C_i^A. That's how the query access patterns are hidden.

C. THE COMPLETE PROTOCOLS OF OUTSOURCED COLLABORATIVE kNN

In this part, we present our privacy-preserving kNN outsourcing protocols in multi-cloud environments using secure schemes of addition, comparison, etc. proposed in Section IV-*B* as building blocks.

1) MAIN IDEA

There're many ways to compute kNN over the distributed databases. A straightforward solution is to converge the datasets from all other clouds at one spot so that the primary cloud can complete kNN over the unified database. No doubt that this method incurs significant communication overhead (i.e., $O(nmL)$, where n, m, L denote the number of clouds, database dimension, and database size, respectively) and cloud resources are not fully utilized. Another solution requires each cloud to compute the distances between query and data tuples. After that, all the distances are transmitted to a single cloud for comparison. However, the communication complexity is $O(nL)$, which is still heavy when L is large. It's a commonsense that the latency within the cloud provider is normally low due to using high performance switchers and relatively simple network, while the network connected to different clouds may be unstable and slow for many reasons, e.g., Internet traffic jam. So the interactions between clouds should be as few as possible.

To solve these issues, we propose a set of protocols by leveraging parallel property of distributed cloud environments. The process includes two parts: one is kNN computation for intra-cloud; the other is kNN classification for inter-clouds. In the following, we name the protocol for intra-cloud as Local-OCkNN and the protocol for inter-cloud as Global-OCkNN. Local-OCkNN is mainly responsible for the locally top k tuples computation, while Global-OCkNN locates the globally nearest points. More specifically, during Local-OCkNN execution, with given q', C_i^A and C_i^B computes the k nearest encrypted records together over its database D_i'. Then, the encrypted distances together with class labels of top k are transmitted to the primary cloud, say $Cloud_p$, who performs Global-OCkNN to compare kn distances via SMnN and returns the encrypted class to the querist. In this way, the complexity for communication across clouds is $O(nk)$, which is far smaller than above two schemes for $n \ll L$ and $k \ll L$.

2) LOCAL-OCkNN

This protocol requires each CSP to handle kNN queries independently in a privacy-preserving manner while still guaranteeing its data owner U_i's rights to retrieve and decrypt encrypted data locally like using cloud storage service. Specifically, the entire process of Local-OCkNN comprises of three phases, that is, Data Uploading Phase, kNN Query Phase, Outsourced kNN Computation Phase, the major steps of which are shown in Algorithm 6.

During Data Uploading Phase, U_i encrypts its database D_i with pk_{U_i} by running $D_i' = \{\mathsf{Enc}(pk_{U_i}, t_{j,h}^i)|j \in [1, L], h \in [1, m+1]\}$, where $t_{j,m+1}^i$ denotes the class label for j^{th} record, and uploads them to C_i^A's storage. Upon receiving D_i', C_i^A re-encrypts the ciphertexts under owners' keys by leveraging PRE technique long with $rk_{U_i \leftrightarrow C_i}$. At the end of the execution, C_i^A gets the re-encrypted database D_{re}' under pk_{C_i}.

During Query Authentication Phase, C_i^A first checks whether Q is an authorized party. After successful identification, q' is then transformed into encryption under pk_{C_i} using its re-encryption key. Here, $q' = \mathsf{Enc}_{pk_Q}(q)$, and q'_{C_i} denote the transformed ciphertexts for q under pk_{C_i}. Afterwards, C_i^A activates the local kNN computation process.

During kNN Computation Phase, first of all, C_i^A and C_i^B compute the encryptions of squared distances between all records of D_{re} and q via SSD protocol. Let M be an encrypted vector, where $M_j =< ed_j', t_{j,m+1}', \mathsf{Enc}_{pk_{C_i}}(j) >$, $ed_j' = \mathsf{Enc}_{pk_{C_i}}(||t_j - q||)$, for $1 \leq j \leq L$, and the last two are encryptions of class label and index, respectively. After that, the record who has the smallest distance is selected from M by conducting SMnN. The outputs are the encrypted minimum distance, its corresponding secrets: class label and index, denoted by $ed_{\min}'$, $c_{\min}'$, and $I_{\min}'$ respectively. Since SMnN computes the minimum in an oblivious way, the index of the minimum record is unknown, which prevents from calculating successive smallest distances. To address this problem, we design a function called ConvertMinToMax to make the current smallest distance become the largest in the encrypted set so as to avoid getting the repeated value.

Algorithm 6 Local-OCkNN(D_i, q', k) → $\{\tau_{i,1}, \ldots, \tau_{i,k}\}$

```
Require: $U_i$ holds its dataset $D_i$ and key pair $pk_{U_i}/sk_{U_i}$; $C_i^A$
  knows the count of neighbors ($k$), $rk_{U_i \leftrightarrow C_i}$, $rk_{Q \leftrightarrow C_i}$, and
  encrypted query $q'$; $C_i^B$ has key pair $pk_{C_i}/sk_{C_i}$.
  {Data Uploading Phase}
  $U_i$: $D_i' \leftarrow \{\mathsf{Enc}_{pk_{U_i}}(t_{j,h}^i)|j \in [1, l], h \in [1, m+1]\}$; Upload
  $D_i'$ to $C_i^A$;
  $C_i^A$: $D_{re}' \leftarrow \{\mathsf{ReEnc}(rk_{U_i \leftrightarrow C_i}, D_i')|i \in [1, n]\}$;
  {Query Authentication Phase}
  $C_i^A$:
    1) Authenticate $Q$'s identity, if $Q$ is not authorized then
       abort;
    2) $q'_{C_i} \leftarrow \mathsf{ReEnc}(rk_{Q \leftrightarrow C_i}, q')$, where $q' = \mathsf{Enc}_{pk_Q}(q)$;
  {Outsourced kNN Computation Phase}
  $C_i^A$ and $C_i^B$:
    1) for $j = 1$ to $L$ do:
       • $ed_j' \leftarrow$ SSD($D_{re}'[j], q'_{C_i}$);
       • $M_j \leftarrow < ed_j', t_{j,m+1}', \mathsf{Enc}_{pk_{C_i}}(j) >$;
    2) for $j = 1$ to $k$ do:
       • $\{ed_{\min}', c_{\min}', I_{\min}'\} \leftarrow$ SMnN($M$);
       • $\tau_{i,j} \leftarrow \{ed_{\min}', c_{\min}'\}$;
       • ConvertMinToMax($ed', ed_{\min}', I_{\min}', d_{\max}'$);

  ConvertMinToMax ($ed', d_{\min}', I_{\min}', d_{\max}'$)
  $C_i^A$, for $l = 1$ to $L$ do:
    • $\mu_l \leftarrow$ SA($\mathsf{Enc}_{pk_{C_i}}(l), \mathsf{Enc}_{pk_{C_i}}(\phi)$);
    • $\mu_l \leftarrow$ SA($\mu_l, \mathsf{Enc}_{pk_{C_i}}(-I_{\min})$);
    • $\delta \leftarrow$ SA($d_{\max}', \mathsf{Enc}_{pk_{C_i}}(-d_{\min})$);
    • $\delta' \leftarrow \mathsf{Blind}(\delta, -1)$;
    • Generate a permutation function $\pi$; $\mu' \leftarrow \pi(\vec{\mu})$;
    • Send $\mu'$ to $C_i^B$;
  $C_i^B$, for $l = 1$ to $L$ do:
    • if $\mathsf{Dec}(sk_{C_i}, \mu_l') = \phi$, then $\nu_l' \leftarrow \mathsf{Enc}_{pk_{C_i}}(2)$;
      else, $\nu_l' \leftarrow \mathsf{Enc}_{pk_{C_i}}(1)$;
    • Send $\nu'$ to $C_i^A$;
  $C_i^A$:
    • $\psi \leftarrow \pi^{-1}(\nu')$;
    • for $l = 1$ to $L$ do:
      – $\chi_l \leftarrow \psi_l \times \delta$;
      – $ed_l' \leftarrow$ SA($ed_l', \delta'$);
      – $ed_l' \leftarrow$ SA($\chi_l, ed_l'$);
```

Let $d_{\max}'$ be the ciphertext of the largest signed integer and ϕ be the non-zero offset known only to C_i^B. First, C_i^A applies SA to compute the difference (denoted as μ_l) between the index of minimum and each offset index, as well as encryption of $d_{\max} - d_{\min}$, denoted by δ. The order of vector μ is further permuted before transmitted. C_i^B decrypts μ' and returns a vector encrypted ν' as follows. If the decrypted element is ϕ, $\nu_l' = \mathsf{Enc}_{pk_{C_i}}(2)$; otherwise, $\nu_l' = \mathsf{Enc}_{pk_{C_i}}(1)$. Obviously, there's only one 2 in ν, whose location indicates the minimum distance position. C_i^A recovers the original order of ν', and then updates every encrypted distance by evaluating

$$\begin{aligned} ed_l' &= \mathsf{Enc}_{pk_{C_i}}(\Delta \cdot (\nu_l - 1) + ed_l') \\ &= \mathsf{Enc}_{pk_{C_i}}(\Delta \cdot \nu_l + ed_l' - \Delta), \end{aligned} \tag{8}$$

where $\nu_l' = \mathsf{Enc}_{pk_{C_i}}(\nu_l)$, $\Delta = d_{\max} - d_{\min}$. Based on Eq.(8), C_i^A computes encryption of $\Delta \nu_l + ed_l'$ first, and then subtracts δ. For $\nu_l = 1$, the value remains unchanged, whereas the encrypted smallest distance of M is converted to maximum when $\nu_l = 2$. At the end of Local-OCkNN execution, the output is a collection of k tuples, in which each tuple contains the ciphertexts of Euclidean distance and its corresponding label, meanwhile no privacy regarding client's database and query is revealed to cloud servers.

3) GLOBAL-OCkNN

This protocol enables multiple clouds to cooperate in outsourced kNN classification. It computes the globally k nearest points over the distributed databases. There are four steps of Global-OCkNN, namely, Key Distribution Phase, kNN Query Phase, Collaborative kNN Classification Phase, and Result Retrieval Phase. Algorithm 7 presents the design details.

During Key Distribution Phase, all participating parties, including data owners, querists, and cloud servers, run KD protocol interactively to generate their own key pairs. In the end, U_i has pk_{U_i}/sk_{U_i}, and Q has pk_Q/sk_Q, while cloud server C_i^B holds pk_{C_i}/sk_{C_i} with C_i^A holding corresponding re-encryption keys.

During kNN Query Phase, Q encrypts q component-wise and submits to the preferred cloud server, denoted by C_p^A. The chosen cloud acts as master node by undertaking the most workload and controlling the entire procedures while other clouds work as slave nodes. Upon receiving q', C_p^A replicates and distributes it to other cloud servers.

During Collaborative kNN Classification Phase, each cloud invokes Local-OCkNN protocol to obtain locally closest points to q, but it does not mean they're globally nearest. So C_i^A sends its local results to C_p^A for further processing. Here, T_i contains $Cloud_i$'s k encrypted nearest tuples and class labels. Once receiving vector T, C_p^A transforms all ciphertexts into ones under pk_{C_p}. Another vector E' is calculated by combining T' and encrypted indices, the element of which is $(\tau'[j/k+1, j\%k], \mathsf{Enc}_{pk_{C_p}}(j))$, for $j \in [1, nk]$. "%" denotes modular operation. As has been mentioned before, τ' contains the ciphertexts of distance and class label as secret. With E', the primary node is able to perform SMnN to get the current smallest tuple. We also use ConvertMinToMax function to update distances obliviously. After k iterations, C_p^A gets the encrypted vector of class labels, denoted by $\Theta' =< \mathsf{Enc}_{pk_{C_i}}(c_1'), \ldots, \mathsf{Enc}_{pk_{C_i}}(c_k') >$. Without loss of generality, we assume that the distributed database consists of θ unique classes, denoted by $< c_1, \ldots, c_\theta >$. C_p^A has Θ as background knowledge. At last, cloud servers compute the encrypted majority class via SMCC protocol, and the result is converted into c_q' under pk_Q.

Algorithm 7 Global-OCkNN(D_i, q, k, Θ) → c_q

```
Require: $U_i$ holds its database $D_i$; $Q$ holds its query $q$; Cloud
    servers know the count of neighbors ($k$) and encryptions of
    class labels $\Theta$.
  {Key Distribution Phase}
  $Cloud_i$, $U_i$, and $Q$:
    1) Jointly compute $\Omega \leftarrow$ KD, where $i, j \in [1, n]$, $\Omega$ is
       the set of key pairs for kNN participants;
  {kNN Query Phase}
  $Q$:
    1) $q' \leftarrow \{\mathsf{Enc}_{pk_Q}(q_j) | j \in [1, m]\}$;
    2) Submit $q'$ to $C_p^A$;
  $C_p^A$:
    1) Replicate $q'$ and distribute it to other federal clouds
       $C_i^A$, where $i \neq p$ and $i, p \in [1, n]$;
  {Collaborative kNN Classification Phase}
  $Cloud_i$, for $i = 1$ to $n$ do:
    1) Compute $T_i \leftarrow$ Local-OCkNN($D_i, q', k$), where
       $T_i = \{\tau_{i,1}, \ldots, \tau_{i,k}\}$;
    2) Send $T_i$ to $C_p^A$;
  $C_p^A$ and $C_p^B$:
    1) $T_i' \leftarrow \mathsf{ReEnc}(rk_{C_i \leftrightarrow C_p}, \mathsf{Enc}_{pk_{C_p}}(T_i))$, where $i \neq p$
       and $i, p \in [1, n]$;
    2) $E_j' \leftarrow \{\tau'[j/k+1, j\%k], \mathsf{Enc}_{pk_{C_p}}(j)\}$, where $j \in [1, n*$
       $k]$, $\tau'[j/k+1, j\%k] \in \{T_1', \ldots, T_n'\}$;
    3) for $j = 1$ to $k$ do:
       • $(gd_{\min}', gc_{\min}', gI_{\min}') \leftarrow$ SMnN($E'$);
       • $\Theta_j' \leftarrow gc_{\min}'$;
       • ConvertMinToMax($E', gd_{\min}', gI_{\min}', d_{\max}'$);
    4) $\mathsf{Enc}_{pk_{C_p}}(c_q) \leftarrow$ SMCC($\Theta, \Theta'$);
    5) $c_q' \leftarrow \mathsf{ReEnc}(rk_{Q \leftrightarrow C_p}^{-1}, \mathsf{Enc}_{pk_{C_p}}(c_q))$;
    6) Send $c_q'$ to $Q$;
  {Result Retrieval Phase}
  $Q$:
    1) Receive $c_q'$ from $C_p^A$;
    2) $c_q \leftarrow \mathsf{Dec}(sk_Q, c_q')$;
```

During Result Retrieval Phase, Q retrieves the desired class label by decrypting c_q' using its private key sk_Q.

4) SECURITY ANALYSIS OF OCkNN

According to Composition Theorem [22], if every step of Global-OCkNN is secure, then we can prove that the complete protocol is secure. The security of OCkNN under semi-honest model is defined as follows.

Theorem 1: During the execution of the OCkNN protocol, no privacy regarding the inputs of data owners and querist, the final output or the access patterns are revealed to cloud servers or other participants as long as ElGamal cryptosystem is semantically secure, and blinding factors are randomly selected.

Proof: To prove the security, we need to demonstrate each phase of Global-OCkNN is secure. First, the KD protocol ensures every party has its own key without being known by anyone else. Due to the semantic security of the ElGamal cryptosystem, q' and D_i' are unknown to all the other parities during the outsourcing period. $Cloud_i$ then invokes Local-OCkNN to compute the locally nearest neighbors over D_i'.

As for Local-OCkNN, we stress that C_i^A cannot derive any privacy regarding datasets and query through the re-encryption process. The SSD sub-protocol used to compute the encryptions of the squared Euclidean distances has been proven secure in Section IV-*B*. The current smallest distance value ($ed'_{\min}$) is selected using SMnN protocol based on set E containing encrypted distance, class label, and index. The output also includes the corresponding label ($c'_{\min}$) and index ($I'_{\min}$). Since $ed'_{\min}, c'_{\min}, I'_{\min}$ are updated obliviously in each iteration, C_i^A is not able to know which record corresponds to $ed'_{\min}$. So the access pattern in comparison operation is hidden. In ConvertMinToMax, nothing is revealed to C_i^A, because its inputs $I'_{\min}, d'_{\min}, \nu'$ are ciphertexts. C_i^B may decrypt μ', but the decryptions are either random values, or predefined offset, while the order of original difference vector μ is permutated by π. Thus, C_i^B does not know the access pattern, either.

The re-encryption technique is also used to unify ciphertexts of locally k encrypted tuples under pk_{C_p}. The final class label (c_q') is computed by SMCC which preserves not only the confidentiality of class labels and frequencies, but the access pattern for the given query. Since we assume there's no collusion among the cloud servers and clients, nothing about the query and database is revealed to other parties by the security and property of the underlying encryption scheme.

Based on the above discussions, we can build a simulator $\mathcal{F}$ in the ideal world that executes the protocol by using random numbers as inputs and returns outputs to adversary. The adversary $\mathcal{A}$ corrupting either cloud server cannot distinguish the view of real world and simulated world. Thus, the proposed OCkNN protocol achieves its security goals. ■

Remark: Using ElGamal cryptosystem to encrypt zero definitely results in part of ciphertext to be zero, because $ct_1 = m \cdot g^r$, where m is the plaintext message and ct_1 is the first part of ciphertext. Therefore, we should avoid encrypting zero directly to protect confidentiality. As for kNN classification, the original dataset which contains zero should be preprocessed by rotation or shifting transformation, the target of which is to avoid appearance of zero attributes and to preserve the Euclidean distance. Apart from this, our proposed scheme ensures server C_i^A cannot deduce privacy from encryption of zero. In SSD sub-protocol, C_i^A may obtain $\mathsf{Enc}_{pk_{C_i}}(0)$ if there are identical attributes or vectors, but nothing regarding the actual content of data is revealed. Likewise, in SM2N, C_i^A may get $\mathsf{Enc}_{pk_{C_i}}(0)$ during difference computation if the two values are the same whereas the data privacy is still protected. During the execution of SMCC and ConvertMinToMax in Local-OCkNN and Global-OCkNN, we add the non-zero offset ϕ on inputs. Though $\mathsf{Enc}_{pk_{C_i}}(0)$ possibly exists during computation, no private information like the equality of two class labels or the position of minimum value is disclosed to C_i^A or C_i^B, since ϕ is unknown to C_i^A and π is unknown to C_i^B. Thus, due to the semantic security of ElGamal cryptosystem and blindness of ϕ, C_i^A cannot deduce either the content of datasets or the access patterns from appearance of "zero" ciphertext.

TABLE 1. Complexity Comparison between OCkNN and PPkNN [6] when $n = 1$.

Algorithm	OCkNN	PPkNN
Computation complexity	$O(L)$	$O(pL\log L)$
Communication complexity	$O(L)$	$O(pL)$

5) COMPLEXITY ANALYSIS OF OCkNN

Let $\texttt{Exp}, \texttt{Mult}$ denote operations of modular exponentiation, multiplication, respectively. Recall that L denotes the size of single user's database, n denotes the number of clouds, and m is the record dimension. For the basic arithmetic building blocks, homomorphic multiplication needs servers to take $2\texttt{Mult}$ operations while SA needs $24\texttt{Exp} + 24\texttt{Mult}$ operations. The overall computation complexity for cloud servers is bounded by $(48mnL + 51knL + 97kn\log L)\texttt{Exp}$ while communication complexity is bounded by $(52mnL + 104kn\log L + 52knL)|\mathbb{G}|$ when $f = 0$, where $|\mathbb{G}|$ is the encryption key size. Similar work [6] proposed outsourcing protocol called PPkNN constructed based on Paillier cryptosystem [25] on the assumption of single data owner. PPkNN achieves the same security level as ours and requires no participation of clients either. The computation complexity of PPkNN is bounded by $(3Lp + 3Lm + 3Lkp\log L)\texttt{Exp}$, where p is referred to as value size in bits while the communication complexity is bounded by $(6Lm + 2L + 10pL + 4Lk + 6pk\log L)|\mathbb{G}|$. Note that L is usually much larger than m, n, k, and Table 1 shows the complexity of our scheme and PPkNN when there's one cloud environment. It's apparent that overhead of our scheme incurs $O(L)$, less than that of PPkNN, mainly because their method invokes complex bit-decomposition protocol to compare encrypted distances and twice larger size of modulo of Paillier's scheme. Furthermore, PPkNN does not support multi-key scenario as we do.

6) PERFORMANCE OPTIMIZATION OF OCkNN

To boost the performance of our protocol, we propose two methods: (a) offline computation and (b) ciphertext randomization. Let's take SA scheme as an example in the following.

(a) Offline computation. C_i^A may adopt offline computation as preparation ahead of outsourced computation. First, C_i^A generates a large set of random numbers, denoted by $RN = \{rn_i | rn_i \in \mathbb{G}, i = 1, 2, \ldots, \rho\}$. RN then are encrypted as $RN' = \{\mathsf{Enc}_{pk_{C_i}}(rn_i) | rn_i \in RN, i = 1, 2, \ldots, \rho\}$. The set size ρ should be as large as possible to enhance randomness and security. The random numbers required in our methods can be selected from RN. And the inverse values such as r_1^{-1},

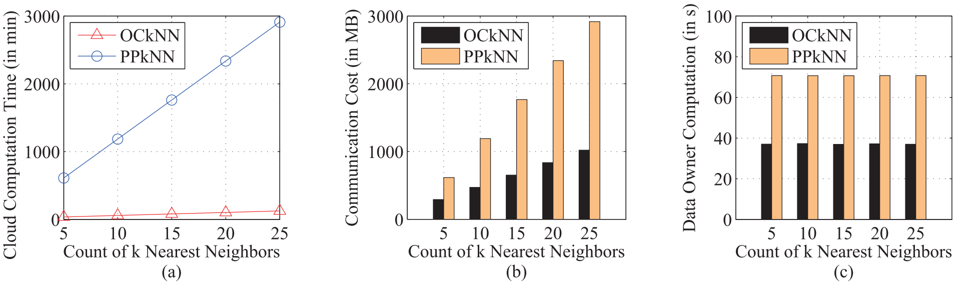

FIGURE 2. **Computation time and communication cost for different protocols with varying k when $n = 1$ and $f = 0$.**

r_2^{-1} in SA can be pre-computed. The encrypted combinations like $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha)$ *et.al.*, can also be calculated beforehand.

(b) Ciphertext randomization. Since encryptions of random numbers like $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha)$ and $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha)$ are pre-computed, they may be easily identified by server C_i^B based on inspections, thus lowering down the difficulty to guess the fraction of inputs. So $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha)$ and $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha)$ are further randomized during SA execution. These encrypted values are multiplied with $\mathsf{Enc}_{pk_{C_i}}(r') \in_R RN'$. Finally, C_i^A obtains $\mathsf{Enc}_{pk_{C_i}}(r_1\alpha r')$, $\mathsf{Enc}_{pk_{C_i}}(-r_2\alpha r')$, while the correctness of the optimized protocol still holds.

By the above boosting mechanisms, C_i^A saves 2 encryptions, 4 modular multiplications, 3 modular inverses as well as random number generation operations.

V. EXPERIMENTAL RESULTS

In this section, we evaluate and analyze the performance of our schemes for outsourced kNN classification under multi-keys in multi-cloud environments and compare our work with similar methods.

A. SETTINGS AND IMPLEMENTATION

The experiments of outsourcing protocols are performed on our local cluster, in which servers have identical configurations which are Intel Xeon E5-2620 @ 2.10 GHz with 12 GB RAM running CentOS 6.5. We implement a proof-of-concept version of the proposed protocol and PPkNN [6] in C++ using the Crypto++5.6.3 library.

All experiments are simulated over the real dataset–Wine Quality dataset from the UCI Machine Learning Repository [26], consisting of 4898 instances, 12 attributes, and 11 class labels. The dataset is horizontally partitioned and distributed to data owners. The key size of the encryption scheme is chosen to be 1024 bits, which is a commonly acceptable size.

B. EMPIRICAL ANALYSIS

We evaluate the performance of our protocols based on the parameters: the count of nearest neighbors (k), the count of clouds (n), and the count of noise (f). k varies from 5 to 25 and f changes from 0 to 4. Every data owner's encrypted database is outsourced to its corresponding cloud server for collaborative kNN classification. The query is selected from the distributed databases randomly for each test. Then, we consider to evaluate the overhead caused by database encryption and outsourced computation. The results presented in the following are averaged over ten test samples.

First, we compare the computation and communication overhead with similar protocol with varying number of nearest neighbors, and assume $n = 1$ in the system since PPkNN can only work in centralized model. From Fig.2(a), it can be clearly observed that the computation time of PPkNN grows sharply with the increase of k while our approach's (OCkNN) grows much more slightly. For example, when $k = 5$, it takes PPkNN 610.2min to process encrypted data while it only requires 35.59min for OCkNN. The time gap scales up with k, so our speedup rate is at least 17 times. This is mainly caused by expensive bit-decomposition and encryptions of comparison processes in PPkNN, which degrades performance whereas we apply blinding technique in SM2N to boost and achieve the same security.

Fig.2(b) shows the communication cost increases with the growth of k. The reason is that the larger k is, the more comparisons over the ciphertexts the outsourced protocols require, which is an indeed expensive operation. Obviously, the traffic caused by PPkNN mounts up much more rapidly. In the worst case, when $k = 25$, it produces around 2.45 times flow as much as ours. As for computation time at data owner's side, Fig.2(c) suggests that OCkNN produces almost half workloads of clients compared with PPkNN because of the complexity of the underlying encryption schemes. Besides, we notice that the burden for data owner is greatly relieved due to kNN outsourcing, in that the time for owners accounts for small proportion of total computation (less than 1% for OCkNN).

Secondly, we evaluate the overhead caused by OCkNN for cloud servers under different distributed model, where the real dataset is uniformly distributed among all owners. From statistics given in Fig.3(a), we can see that the time cloud servers spend on computation grows with the rise of k regardless of the number of clouds. This is consistent with our previous observation. By contrast, the cost decreases with the growth of n, which mainly results from the fact that the more clouds are involved, the more jobs can be processed in parallel. For example, when $n = 2$, it takes servers 17.86min

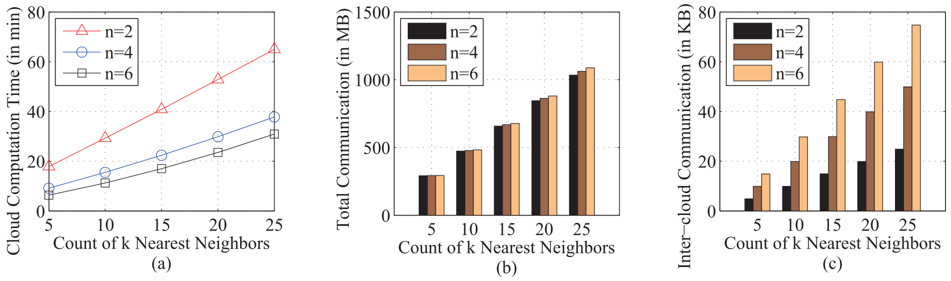

FIGURE 3. Cloud computation time and communication cost with varying k and n when $f = 0$.

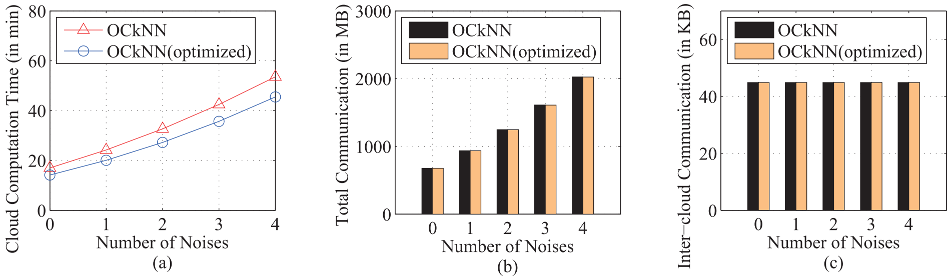

FIGURE 4. Cloud computation time and communication cost with varying f when $k = 15$ and $n = 5$.

to compute kNN for $k = 5$ while the same task for $n = 4$, $n = 6$ require 9.16min and 6.34min, respectively. Thus, OCkNN can accelerate the outsourced kNN in the multi-cloud environments. Fig.3(b) and (c) reveal the costs of total communication and inter-cloud communication, respectively, which not only grow with the rise of k, but also with the number of public clouds. However, the traffic between clouds merely occupies a tiny part within overall communication. For instance, the highest point of Fig.3(c) is 74.7KB with $k = 25$ and $n = 6$ while that of total cost is 1090MB, several orders higher than inter-cloud cost. Our scheme minimizes external traffic considering that the network channels among different CSPs are not as stable and fast as in-house bandwidth. Furthermore, out-band connections are usually charged by CSP [24].

According to SA protocol proposed in Section IV-*B*, C_i^A inserts encrypted random numbers (i.e., $\{\mathsf{Enc}_{pk_{C_i}}(\omega_j)|0 \leq j \leq f\}$) in Γ in order to enhance security. Next, we evaluate the performance of cloud servers with varying number of random noises inserted in Γ when $k = 15$. Fig.4 shows the computation time and communication costs of OCkNN and its optimized version increase with number of f, respectively. The primary reason for this is that the more noises inserted in Γ, the more intermediate results need to be processed by C_i^B during addition operation. The optimized protocol saves computation power by at least 20% due to offline preparation, while the inter-cloud and intra-cloud communication costs for both schemes are roughly the same. Nevertheless, when $f = 4$, the outsourcing time is beyond 45min for both protocols. So it's essential to make a tradeoff between security and efficiency. For instance, the probability of guessing correct input ratio can be as low as 3.33% when $f = 1$, whereas the optimized scheme takes 20min to compute kNN with 0.914GB traffic. This kind of cost is acceptable considering the guess difficulty. Additionally, Fig.4(c) suggests the inter-cloud traffic is not related to f, because only the locally top k results are transmitted across clouds.

VI. RELATED WORK

In this section, we review the existing outsourced privacy-preserving kNN approaches under different models.

A. DATA DISTRIBUTION MODEL

Similarly, this model concerns collaborative mining over partitioned data, requiring all parties to compute kNN classification using SMC techniques [27]–[30]. However, we claim that the outsourced kNN problem cannot be addressed by SMC, in that in our case the data are encrypted and stored in cloud instead of being held by data owners meanwhile the mining results are still needed to be preserved. Besides, the majority computation workload should be completed by CSP.

B. SINGLE-KEY OUTSOURCED MODEL

This model assumes the data are encrypted and outsourced in the cloud while all the participants share the same key. Most recent methods have been proposed based on this assumption. Distance-Recoverable Encryption (DRE) is the straightforward solution, but it's not secure against level-2 or level-3 attacks [7]. Wong *et al.* [7] proposed an Asymmetric Scalar-product-Preserving Encryption (ASPE) scheme to defeat signature linking attack while preserving the order of distance. They suppose that the query users are fully trusted and thus the symmetric key is shared among all query users and data owner, posing great threat to database security if one query user is compromised by the adversary. Paper [15] solved this by proposing a new scheme which only reveals partial information of data owner's key to query users. Nevertheless, we argue that their work is different from ours mainly in the following aspects: (1) their system model consists of only one data owner and many query users whereas we assume there are multiple data owners and query users; (2) encrypting a query in [15] requires data owner's participation, while each party can use its own public key to encrypt data independently in our proposal; (3) both [7] and [15] do not conceal access patterns of the given query, since it is necessary for the cloud server to know the order of distances to return the k closest tuples; (4) they do not consider parallelization during kNN outsourcing.

Elmehdwi *et al.* [9] leveraged Paillier Cryptosystem's additively homomorphic property to construct privacy-preserving primitives, based on which the outsourcing protocol handles kNN query over encrypted database, and returns the k closest records to the user. Their approach protects data confidentiality, query privacy, and access patterns. The work is further extended in [6], in which they proposed new solutions for secure minimum (SMIN), secure frequency (SF), etc for PPkNN, together with formal security analysis. The computation of majority class is also outsourced in the extended version. Their work is similar with ours in terms of privacy and access pattern protection, whereas their method hardly satisfies some critical requirements of our system model due to the following reasons. Firstly, the secure primitives in [9] and [6] do not support that data owners and querists use their own keys for data encryption. Instead, all data are encrypted under the server's public key. Secondly, the computational and communication cost of PPkNN is too high for real-world application. To compute the minimum value during SMIN, secure bit-decomposition protocol is invoked and most operations are performed on the encrypted bits, incurring significant overhead on cloud servers. Since SMIN is essential step to compute kNN, it explains why PPkNN is not so efficient. On the contrary, our strategy to compare ciphertexts seems simple but practical as discussed in Algorithm 4. Thirdly, PPkNN is not parallelized under multi-cloud setting as OCkNN.

Instead of computing distance straightway, Xu *et al.* [8] proposed a method combining order preserving encryption, dimensionality expansion, random noise injection and random projection and developed efficient range query services that achieve sublinear time complexity of processing queries. But their scheme needs frequent interactions with client, which deviates from the intention of data mining outsourcing. Based on practical observations, the single-key model faces potential security risks: the key leakage problem and snooping attacks by the corrupted cloud server.

C. MULTI-KEY OUTSOURCED MODEL

This model supposes different parties hold their respective keys, hence mitigating the single-key risks. A fully homomorphic encryption (FHE) scheme under multiple keys was proposed in [31], but its efficiency is still impractical as other FHE schemes [32]. Besides, it's not resoluble to make comparison over ciphertexts for FHE. Paper [11] utilized the double decryption mechanisms of BCP cryptosystem [18] to convert ciphertexts under different keys into ones under a common key under two non-colluding servers model. Following this, Wang *et al.* [12], [13] made further improvements in efficiency via PRE, but their additively homomorphic scheme merely supports short plaintext due to solving discrete logarithm problem, while the multiplicatively homomorphic scheme discloses ratio of inputs directly in two-server model. Another recent work [14] studied the collaborative mining under multi-owner setting and proposed several security enhanced schemes. Though each party owns a exclusive symmetric key, their schemes leak data owners' keys to cloud service users to re-encrypt the database. Moreover, the underlying encryption is deterministic and not secure against advanced attacks. To the best of our knowledge, none of existing works address privacy-preserving outsourced kNN using public key encryption under multi-cloud case.

VII. CONCLUSION

In this paper, we focused on the multi-cloud scenario where data owners upload the encrypted databases to their corresponding public clouds for joint kNN classification. To protect privacy under this system model, we proposed a series of efficient secure building blocks and an outsourcing protocol, named OCkNN for short. Our scheme allows clients to encrypt data with their own keys while require no user-server interactions during the outsourcing stage. Theoretical analysis shows that the proposed scheme ensures the confidentiality of data, kNN query, query result and access patterns with small computational and communication costs. We also highlight the practicability of our protocols by performing experiments on real datasets under different parameter settings in comparison with similar works. Since OCkNN offers a high probabilistic guarantee on little privacy leakage and is not rapid enough for large-scale dataset, we plan to investigate more secure and efficient solutions as our future work.

REFERENCES

[1] D. Song, E. Shi, I. Fisher, and U. Shankar, "Cloud data protection for the masses," *IEEE Comput.*, vol. 45, no. 1, pp. 39–45, Jan. 2012.

[2] K. Ren, C. Wang, and Q. Wang, "Secure challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan. 2012.

[3] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. ACM CCS*, 2009, pp. 199–212.

[4] P. Williams, R. Sion, and B. Carbunar, "Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage," in *Proc. ACM CCS*, 2008, pp. 139–148.

[5] W. Du and M. J. Atallah, "Privacy-Preserving Cooperative Statistical Analysis," in *Proc. 17th Annu. Comput. Secur. Appl. Conf.*, Dec. 2012, pp. 9–19.

[6] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "K-nearest neighbor classification over semantically secure encrypted relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1–14, May 2015.

[7] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted database," in *Proc. ACM SIGMOD*, 2009, pp. 139–152.

[8] H. Xu, S. Guo, and K. Chen, "Building confidential and efficient query services in the cloud with RASP data perturbation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 322–335, Feb. 2014.

[9] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. ICDE*, Mar./Apr. 2014, pp. 664–675.

[10] F. Li, R. Shin, and V. Paxson, "Exploring privacy preservation in outsourced k-nearest neighbors with multiple data owners," in *Proc. ACM CCSW*, 2015, pp. 53–64.

[11] A. Peter, E. Tews, and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.

[12] B. Wang, M. Li, S. S. M. Chow, and H. Li, "Computing encrypted cloud data efficiently under multiple keys," in *Proc. 4th Int. Workshop Secur. Privacy Cloud Comput.*, 2013, pp. 504–513.

[13] B. Wang, M. Li, S. S. M. Chow, and H. Li, "A tale of two clouds: Computing on data encrypted under multiple keys," in *Proc. IEEE CNS*, Oct. 2014, pp. 337–345.

[14] Y. Huang, Q. Lu, and Y. Xiong, "Collaborative outsourced data mining for secure cloud computing," *J. Netw.*, vol. 9, no. 9, pp. 2655–2664, 2014.

[15] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN computation on encrypted cloud data without sharing key with query users," in *Proc. ACM Cloud Comput.*, 2013, pp. 55–60.

[16] M. D. Van and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in *Proc. HotSec*, 2010, pp. 1–8.

[17] S. S. M. Chow, J. H. Lee, and M. Strauss, "Two-party computation model for privacy-preserving queries over distributed databases," in *Proc. NDSS*, 2009.

[18] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *Proc. ASIACRYPT*, 2003, pp. 37–54.

[19] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. EUROCRYPT*, 1998, pp. 127–144.

[20] J. Weng, R. H. Deng, S. Liu, and K. Chen, "Chosen-ciphertext secure bidirectional proxy re-encryption schemes without pairings," *Inf. Sci.*, vol. 180, no. 24, pp. 5077–5089, 2010.

[21] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encrytpion," in *Proc. ACM CCS*, 2007, pp. 185–194.

[22] O. Goldreich, *The Foundations of Cryptography*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[23] *Google Compute Engine*, accessed on 2016. [Online]. Available: https://cloud.google.com/console

[24] *Amazon Elastic Compute Cloud (EC2)*, accessed on 2016. [Online]. Available: https://aws.amazon.com/cn/ec2/

[25] P. Paiilier, "Public key cryptosystems based on composite degree residuosity classes," in *Proc. Eurocrypt*, 1999, pp. 139–148.

[26] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Syst., Elsevier*, vol. 47, no. 4, pp. 547–553, 2009.

[27] H. Hu, J. Xu, C. Ren, and B. Choi, "Processing private queries over untrusted data cloud through privacy homomorphism," in *Proc. IEEE ICDE*, 2011, pp. 601–612.

[28] Y. Qi and M. J. Atallah, "Efficient privacy-preserving k-nearest neighbor search," in *Proc. IEEE ICDCS*, 2008, pp. 311–319.

[29] M. Kantarcioglu and C. Clifton, "Privately computing a distribted k-nn classifer," in *Proc. PKDD*, 2008, pp. 279–290.

[30] L. Xiong, S. Chitti, and L. Liu, "K nearest neighbor classification across multiple private databases," in *Proc. ACM CIKM*, 2006, pp. 840–841.

[31] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. STOC*, 2012, pp. 1219–1234.

[32] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. ITCS*, 2012, pp. 309–325.

HONG RONG received the M.S. degree from the School of Electronic Science and Technology, National University of Defense Technology, in 2013, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of CEMEE. His research interests include privacy-preserving data mining, cloud computing security, and big data security.

HUI-MEI WANG received the B.S. degree from Southwest Jiaotong University in 2004 and the M.S. and Ph.D. degrees from the National University of Defense Technology in 2007 and 2012, respectively. She is currently a Lecturer with the State Key laboratory of CEMEE, National University of Defense Technology. Her research interests include evaluation techniques in network security, cloud computing, and distributed systems.

JIAN LIU received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology in 2009, 2011, and 2016, respectively. He is currently a Lecturer with the CEMEE State Key Laboratory, National University of Defense Technology. His research interests include secure data outsourcing in cloud computing, public auditing, and access control mechanisms.

MING XIAN received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology in 1991, 1995, and 1998, respectively. He is currently a Professor with the State Key Laboratory of CEMEE, National University of Defense Technology. His research interests include on cryptography and information security, cloud computing, wireless sensor network and system modeling, and simulation and evaluation. His research was supported by the National High Technology Research and Development Program of China.