Received June 6, 2018, accepted July 16, 2018, date of publication July 25, 2018, date of current version August 20, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2859758

Efficient k-Nearest Neighbor Classification Over Semantically Secure Hybrid Encrypted Cloud Database

WEI WU, JIAN LIU, HONG RONG, HUIMEI WANG, AND MING XIAN
School of Electronic Science, National University of Defense Technology, Changsha 410073, China

Corresponding author: Wei Wu (goodwuwei18@163.com)

ABSTRACT Nowadays, individuals and companies increasingly tend to outsource their databases and further data operations to cloud service provides. However, utilizing the cost-saving advantages of cloud computing brings about the risk of violating database security and user's privacy. In this paper, we focus on the problem of privacy-preserving k-nearest neighbor (kNN) classification, in which a query user (QU) submits an encrypted query point to a cloud server (CS) and asks for the kNN classification labels based on the encrypted cloud database outsourced by a data owner (DO), without disclosing any privacy of DO or QU to CS. Previous secure kNN query schemes either cannot fully achieve required security properties or introduce heavy computation costs, making them not practical in real-world applications. To better solve this problem, we propose a novel efficient privacy-preserving kNN classification protocol over semantically secure hybrid encrypted cloud database using Paillier and ElGamal cryptosystems. The proposed protocol protects both database security and query privacy and also hides data access patterns from CS. We formally analyze the security of our protocol and evaluate the performance through extensive experiments. The experiment results show that the computation cost of our protocol is about two orders of magnitude lower than that of the state-of-the-art protocol while achieving the same security and privacy properties.

INDEX TERMS Privacy-preserving, kNN classification, cloud computing, encryption.

I. INTRODUCTION

In recent years, an increasingly number of data owners are motivated to utilize the powerful and flexible cloud computing paradigm [2] in order to reduce their cost for local data storages and operations. Generally, after outsourcing the database, a data owner may also outsource the further data analysis and computation tasks to cloud servers. Considering the security of its private database in the remote cloud, a data owner usually chooses to encrypt it before outsourcing. However, executing computations over encrypted databases is a very challenging problem if cloud servers cannot perform decryption.

As a fundamental database analysis operation, the k-nearest neighbor (kNN) query can be used as a standalone database query or a basic module of common data mining tasks [3]. Considering its significance in many applications, to support kNN query over encrypted cloud database, many works [1], [3]–[10] have been proposed in recent years. In these works, performing kNN query over encrypted database usually involves three different parties: the data owner (DO), the query user (QU) and the cloud server (CS). And the main goal of a secure kNN query scheme is to preserve (1) database security, (2) query privacy and also hide (3) data access patterns from the untrusted CS [1]. Unfortunately, these existing schemes either cannot simultaneously achieve the three security and privacy requirements [3]–[7], [9] or introduce heavy computation overheads [1], [10], making them not practical in real-world scenarios.

Specifically, Wong *et al.* [3] propose an asymmetric scalar-product-preserving encryption (ASPE) scheme, in which both database security and query privacy are efficiently protected. Other works such as [4]–[6] present several methods to securely compute an approximate kNN query over encrypted database. Although these works provide some good ideas to solve the problem of secure kNN query, they assume that all QUs can be trusted who have access to DO's secret key, which means an attacker can easily break the security of these schemes by corrupting a single QU. Considering the key confidentiality, Zhu *et al.* [7]–[9] propose some works in which DO stays on-line to protect its secret key during the process of query encryption. However, this DO-on-line requirement introduces extra computation and communication cost after DO already outsourced its database and query processes to CS. While none of the aforementioned schemes [3]–[9] consider hiding data access patterns during the query processes, Elmehdwi *et al.* [10] and Samanthula *et al.* [1] present two schemes using Paillier cryptosystem and two non-colluding clouds to achieve this privacy property. Nevertheless, their secure protocols are very complicated which results in a very high computation cost for hiding data access patterns. In our previous work [11], we efficiently hide data access patterns by using random permutation method and DO does not need to participate in the on-line encrypted computation processes. However, the scheme does not provide semantic security for the outsourced database which is achieved in the works [10] and [1].

In this paper, we propose a novel efficient privacy preserving kNN classification scheme over semantically secure hybrid encrypted database in outsourced cloud environments. In our protocol, after outsourcing the encrypted database, the data owner does not participate during the kNN classification process. Thus, no information is disclosed to the data owner. Besides, the proposed scheme meets all the aforementioned privacy requirements:

• **Database security** - The plain database should not be revealed to cloud servers.

• **Query privacy** - The plain query points and plain kNN classification labels should not be revealed to cloud servers.

• **Hiding data access patterns** - The data access patterns, such as the corresponding encrypted database tuples to the kNN classification labels, should not be revealed to cloud servers.

In our protocol, the intermediate computation values obtained by the cloud servers are either newly generated randomized (or re-encrypted) encryptions or random numbers. By combining a implementation of random permutation, we prevent cloud servers to derive out which database tuples correspond to the kNN classification labels. In addition, after submitting its encrypted query point to cloud server, QU does not participate in any computations during the kNN classification process. Thus, data access patterns are also hidden from QU (more details are given in Section IV). Our contributions in this paper can be summarized as follows.

(1) We propose a novel secure inner Product (SIP) protocol and analyze its security in Section III-A. In this protocol, no information of the private points are revealed to cloud servers. In addition, our SIP protocol has less computation complexity than the SSED (Secure Squared Euclidean Distance) protocol in [1] (more details are given in Section IV).

(2) We propose a novel collusion resistant Re-encryption key Generation (CRRKG) protocol and analyze its security in Section III-D. In this protocol, DO's secret ElGamal classification labels decryption key is protected from cloud servers even if they collude with QU.

(3) We propose a novel privacy preserving kNN classification (PPKC) protocol in Section IV and analyze its security in Section V. By using SIP and Proxy Re-encryption (PRE) as sub-protocols, as well as utilizing random permutation, our PPKC securely computes the encrypted kNN classification labels in an efficient way.

(4) We analyze the cost of our PPKC protocol and evaluate its performance with extensive experiments. The experiment results show that the computation cost of our PPKC protocol is about two orders of magnitude lower than the cost of the state-of-the-art PPkNN protocol in [1], while achieving the same security and privacy properties.

The rest of the paper is organized as follows. We introduce the notations, system architecture, threat model and the Paillier cryptosystem in Section II. The basic security primitives are given in Section III. Our new privacy preserving kNN classification (PPKC) protocol is proposed in Section IV. The security analysis of our PPKC protocol is given in Section V. We analyze the cost of our PPKC protocol and evaluate its performance with experiments in Section VI. The related work are discussed in Section VII. Lastly, we conclude the paper along with the future work in Section VIII.

II. BACKGROUND

In this section, we introduce the notations, system architecture, threat model and the Paillier cryptosystem.

A. NOTATIONS

We give the common notations used in this paper as follows.

n - The number of database tuples.

d - The dimension of plain database points and plain query points.

$\boldsymbol{D}$ - DO's private plain database, consisting of n data tuples $\boldsymbol{D} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n\}$.

$\boldsymbol{t}_i$ - A plain database tuple $\boldsymbol{t}_i = \{\boldsymbol{p}_i, c_i\}$.

$\boldsymbol{p}_i$ - A plain d-dimensional database point $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$.

c_i - A plain numerical classification label.

$\boldsymbol{D}'$ - The encrypted database of $\boldsymbol{D}$, outsourced to CS, $\boldsymbol{D}' = \{\boldsymbol{t}'_1, \boldsymbol{t}'_2, \ldots, \boldsymbol{t}'_n\}$.

$\boldsymbol{t}'_i$ - An encrypted database tuple $\boldsymbol{t}'_i = \{\boldsymbol{p}'_i, c'_i\}$.

$\boldsymbol{p}'_i$ - An encrypted database point of $\boldsymbol{p}_i$.

c'_i - An encrypted classification label of c_i.

$\boldsymbol{q}$ - A private plain d-dimensional query point of QU, $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$.

$\boldsymbol{q}'$ - An encrypted query point of $\boldsymbol{q}$.

$\{pk_p, sk_p\}$ - The public key and secret key of Paillier cryptosystem, generated by DO and used for the encrption of database points $\boldsymbol{p}_i$ and query point $\boldsymbol{q}$.

$\{pk_c, sk_c\}$ - The public key and secret key of ElGamal cryptosystem, generated by DO and used for the encrption of classification labels c_i.

sk_u - An unique ElGamal classification labels decryption key for a QU u.

rk_{u_1}, rk_{u_2} - Unique ElGamal classification labels Re-encryption keys for a QU u.

$\boldsymbol{C}'_q$ - The set of ElGamal re-encrypted kNN classification labels.

$\boldsymbol{C}_q$ - The set of plain kNN classification labels.

$\mathbb{Z}_N$ - The Paillier cryptosystem plaintext space

$\mathbb{Z}_{N^2}$ - The Paillier cryptosystem ciphertext space

$\mathbb{G}$ - The ElGamal cryptosystem plaintext space

$\mathbb{C}$ - The ElGamal cryptosystem ciphertext space

B. SYSTEM ARCHITECTURE

In this paper, there are three different parties in the outsourced cloud environment: the data owner (DO), the query user (QU) and two cloud servers C_1 and C_2, which are shown in Figure 1. We assume that the two cloud servers are non-colluding and this assumption has been used in recent works [1], [10], [12]. It is reasonable because cloud service providers are usually famous IT companies, considering their reputation and revenues, they are very unlikely to collude.

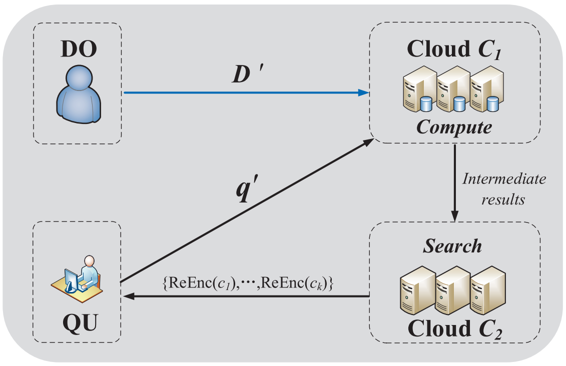

FIGURE 1. System architecture.

As shown in the figure, DO outsources its private database $\boldsymbol{D}$ to C_1 in an encrypted form $\boldsymbol{D}'$. The encrypted database $\boldsymbol{D}'$ consists of encrypted database points and encrypted classification labels which are encrypted using Paillier and ElGamal cryptosystems respectively. The Paillier public-secret key pair for encryption and decryption of database points is (pk_p, sk_p) and the ElGamal key pair for encryption and decryption of classification labels is (pk_c, sk_c). To enable the outsourced kNN classification, DO also outsources its Paillier secret key sk_p to C_2. After outsourcing, DO does not keep the original plain database $\boldsymbol{D}$ and all further operations are outsourced to the two cloud servers.

When a QU wants to determine the classification label of its private query point $\boldsymbol{q}$ based on the encrypted database $\boldsymbol{D}'$ according to the Euclidean distances, it uses the Paillier public key pk_p to encrypt $\boldsymbol{q}$ into $\boldsymbol{q}'$ and submits $\boldsymbol{q}'$ to C_1 for kNN classification. After receiving $\boldsymbol{q}'$, C_1 performs computations over $\boldsymbol{D}'$ and obtains the "encrypted distances". Here, C_1 cannot directly use these "distances" for comparison because they are in encrypted form. Next, C_1 re-encrypts the classification labels in $\boldsymbol{D}'$ and randomly permutes both the "encrypted distances" and re-encrypted labels, and sends these permuted intermediate results to C_2. Using the Paillier secret key sk_p, C_2 decrypts the received "encrypted distances" and gets the correct plain values (corresponding to Euclidean distances) for kNN search. Then, C_2 performs a second round re-encryptions and returns the new re-encrypted kNN classification labels $ReEnc(c_1),\ldots,ReEnc(c_k)$ to the corresponding QU. At last, the QU performs decryption using its unique ElGamal secret key sk_u and obtains the plain kNN classification labels.

C. THREAT MODEL

Similar as the previous work [1], we use the security definitions in the work of secure multi-party computation [13], [14]. We assume that the parties in our protocols are semi-honest. Briefly, we define the properties of a secure protocol under the semi-honest model in the following definition [15], [16].

Definition 1: Let a_i be the input of party P_i, $\Pi_i(\pi)$ be P_i's execution image of the protocol π and b_i be the output for party P_i computed from π. Then, π is secure if $\Pi_i(\pi)$ can be simulated from a_i and b_i such that distribution of the simulated image is computationally indistinguishable from $\Pi_i(\pi)$.

In the definition, an execution image usually contains the input, the output and the information exchanged during the execution of a protocol. To prove a protocol is secure under semi-honest model, we generally need to demonstrate there is no information leakage from the execution image of a protocol respecting the private inputs of participating parties [16].

In our proposed scheme, as discussed in work [1], we assume the two non-colluding clouds are the potential adversaries since they have accesses to the encrypted database and the secret key. To clearly describe the ability of adversaries, we give the following assumptions of collusion-attack:

- The two clouds cannot collude with each other.
- The two clouds cannot collude with DO.
- The two clouds can collude with one QU or some QUs.

The first assumption is used in work [1] and has been explained in Section II-B. The second assumption is reasonable because DO has the secret key to decrypt the encrypted query point. Once DO is corrupted, it will be easy to recover the plain query point. For the third one, we assume that some QUs may be corrupted by money or other benefits and share their ElGamal classification labels decryption keys to the clouds. However, we assume a QU will not disclose its private query point to the clouds even if it is corrupted. This is because the query point may contain much more private information than its kNN classification labels.

D. PAILLIER CRYPTOSYSTEM

The Paillier cryptosystem is an additive homomorphic and probabilistic asymmetric encryption scheme whose security is based on the decisional composite residuosity assumption [17]. Let $E_{pk}(\cdot)$ be the encryption function with public key pk given by (N, g), where N is a product of two large primes and g is a generator in $\mathbb{Z}^*_{N^2}$. Let $D_{sk}(\cdot)$ be the decryption fuction with secret key sk. For any given two plaintexts $m_1, m_2 \in \mathbb{Z}_N$, the Paillier encryption scheme has the following properties:

(1) Homomorphic addition.
$D_{sk}(E_{pk}(m_1 + m_2)) = D_{sk}(E_{pk}(m_1) * E_{pk}(m_2) \ mod \ N^2)$.

(2) Homomorphic multiplication.
$D_{sk}(E_{pk}(m_1 * m_2)) = D_{sk}(E_{pk}(m_1)^{m_2} \ mod \ N^2)$.

(3) Semantic security. The encryption scheme is semantically secure [16], [18]. Concisely, given a collection of ciphertexts, an adversary cannot figure out any additional information about the plaintext(s).

For conciseness, we drop the mod N^2 term during homomorphic operations in the rest of this paper.

III. BASIC SECURITY PRIMITIVES

In this section, we present a set of basic protocols that will be applied as sub-protocols in our proposed privacy preserving kNN classification (PPKC) protocol in Section IV. All of the following protocols are considered under the non-collusion assumption between the two cloud servers C_1 and C_2. As mentioned in Section II-B, only C_2 knows DO's Paillier secret key sk_p, while pk_p is the Paillier public key.

• Secure Inner Product (SIP) Protocol:

This protocol considers C_1 with two Paillier encrypted points $\boldsymbol{a}'$ and $\boldsymbol{b}'$ as inputs and outputs the encrypted inner product $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$ to C_1, where the plain points $\boldsymbol{a}$ and $\boldsymbol{b}$ are not known to C_1 and C_2. During this protocol, no information about $\boldsymbol{a}$ and $\boldsymbol{b}$ is disclosed to C_1 and C_2. The output $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$ is known only to C_1. Here the jth dimension of $\boldsymbol{a}$ (resp., $\boldsymbol{b}$) is encrypted into $E_{pk_p}(a_j)$ (resp., $E_{pk_p}(b_j)$).

• Proxy Re-encryption (PRE) Protocol:

This protocol considers a proxy (a third party) P_1 with ElGamal ciphertext $m'_i = Enc(pk_i, m)$ under the public key pk_i and Re-encryption key $rk_{i \to j}$ as inputs and outputs a new ElGamal ciphertext $m'_j = Enc(pk_j, m)$ under the public key pk_j, where the plaintext m is not known to P_1. During this protocol, no information about m is disclosed to P_1.

• Secure Re-encryption Key Generation (SRKG) Protocol:

This protocol considers DO with ElGamal secret key sk_c and QU with ElGamal secret key sk_u as inputs and outputs a Re-encryption key $rk_u = sk_c - sk_u$ to C_1, where sk_c is not known to C_1 and QU, and sk_u is not known to C_1 and DO. During this protocol, no information about sk_c and sk_u is disclosed to any other parties.

• Collusion Resistant Re-encryption Key Generation (CRRKG) Protocol:

This protocol considers DO with ElGamal secret key sk_c and QU with ElGamal secret key sk_u as inputs and outputs two Re-encryption keys: rk_{u_1} to C_1 and rk_{u_2} to C_2, where sk_c is not known to C_1 and C_2 even if they collude with QU (C_1 and C_2 cannot collude). During this protocol, no information about sk_c and sk_u is disclosed to any other parties.

Algorithm 1 SIP$(\boldsymbol{a}', \boldsymbol{b}') \rightarrow E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$

```
Require: $C_1$ has $\boldsymbol{a}'$ and $\boldsymbol{b}'$; $C_2$ has $sk_p$
1: $C_1$:
    (a). For $j \in [d]$, generate $2d$ random numbers $r_j, e_j \in \mathbb{Z}_N$
    (b). For $j \in [d]$ do:
        • $a''_j \leftarrow E_{pk_p}(a_j) * E_{pk_p}(r_j)$
        • $b''_j \leftarrow E_{pk_p}(b_j) * E_{pk_p}(e_j)$
    (c). Send $a''_j, b''_j$ to $C_2, j \in [d]$
2: $C_2$:
    (a). Receive $a''_j, b''_j$ from $C_1, j \in [d]$
    (b). For $j \in [d]$, $\tilde{a}_j \leftarrow D_{sk_p}(a''_j)$; $\tilde{b}_j \leftarrow D_{sk_p}(b''_j)$
    (c). $h \leftarrow \sum_{j=1}^{d} \tilde{a}_j * \tilde{b}_j \ mod \ N$
    (d). $h' \leftarrow E_{pk_p}(h)$; send $h'$ to $C_1$
3: $C_1$:
    (a). Receive $h'$ from $C_2$
    (b). $s_1 \leftarrow E_{pk_p}(a_1)^{e_1} * E_{pk_p}(a_2)^{e_2} * \cdots * E_{pk_p}(a_d)^{e_d}$
    (c). $s_2 \leftarrow E_{pk_p}(b_1)^{r_1} * E_{pk_p}(b_2)^{r_2} * \cdots * E_{pk_p}(b_d)^{r_d}$
    (d). $s_3 \leftarrow E_{pk_p}(\sum_{j=1}^{d} r_j e_j)$
    (e). $s \leftarrow s_1 * s_2 * s_3$
    (f). $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b}) \leftarrow h' * s^{N-1}$
```

We now discuss the detailed process for each of these protocols. We either propose a new approach or refer to a known efficient solution to each of them.

A. SECURE INNER Product (SIP) PROTOCOL

Consider C_1 with two encrypted private points $\boldsymbol{a}' = (E_{pk_p}(a_1), \ldots, E_{pk_p}(a_d))$ and $\boldsymbol{b}' = (E_{pk_p}(b_1), \ldots, E_{pk_p}(b_d))$ as input which are encrypted using the Paillier public key pk_p, and C_2 with the Paillier secret key sk_p. The aim of secure inner Product (SIP) protocol is to return the Paillier encryption of $\boldsymbol{a} \cdot \boldsymbol{b}$, i.e., $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$ as output to C_1, where $\boldsymbol{a}$ (resp., $\boldsymbol{b}$) is the plaintext of $\boldsymbol{a}'$ (resp., $\boldsymbol{b}'$). During this protocol, no information regarding the private points $\boldsymbol{a}$ and $\boldsymbol{b}$ is disclosed to C_1 and C_2. The main idea of SIP is based on the following property that holds for any given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_N^d$:

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_d b_d = \sum_{j=1}^{d} (a_j + r_j)(b_j + e_j) - \sum_{j=1}^{d} (a_j e_j + b_j r_j + r_j e_j) \quad (1)$$

where all the arithmetic computations are performed under $\mathbb{Z}_N$ (the Paillier plaintext space). The overall steps of SIP are shown in Algorithm 1. Concisely, C_1 firstly randomizes a_j and b_j by computing $a''_j = E_{pk_p}(a_j) * E_{pk_p}(r_j)$ and $b''_j = E_{pk_p}(b_j) * E_{pk_p}(e_j)$, and sends them to C_2. Here r_j and e_j are random numbers in $\mathbb{Z}_N$ known only to C_1. After receiving, C_2 decrypts and computes $h = \sum_{j=1}^{d} (a_j + r_j)(b_j + e_j) \ mod \ N$. Then, C_2 encrypts h and sends it to C_1. After that, C_1 eliminates extra random factors from $h' = E_{pk_p}(\sum_{j=1}^{d} (a_j + r_j)(b_j + e_j))$ based on Eq. (1) to get $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$. To point out, for any given $m \in \mathbb{Z}_N$, "$N - m$" is equal to "$-m$" under $\mathbb{Z}_N$.

• *Proof of Security for SIP:* As mentioned in Section II-C, to formally prove the security of SIP under the semi-honest model, we need to show that the simulated image of SIP is computationally indistinguishable from the real execution image of SIP.

Generally, an execution image consists of the messages exchanged and the information derived out from these messages. Thus, according to Algorithm 1, let the execution image of C_2 be denoted by $\Pi_{C_2}(\text{SIP})$ as below

$$\{\left\langle a_j'', a_j + r_j \right\rangle, \left\langle b_j'', b_j + e_j \right\rangle \mid \quad for\ j \in [d]\}.$$

Note that $a_j + r_j$ and $b_j + e_j$ are the plaintexts of a_j'' and b_j'' respectively. Without loss of generality, assume the simulated image of C_2 be $\Pi^S_{C_2}(\text{SIP})$ as below

$$\{\langle \delta_{1,j}, \delta_{2,j}\rangle, \langle \delta_{3,j}, \delta_{4,j}\rangle \mid \quad for\ j \in [d]\}.$$

Here $\delta_{1,j}$ and $\delta_{3,j}$ are randomly generated from $\mathbb{Z}_{N^2}$ (the Paillier ciphertext space) while $\delta_{2,j}$ and $\delta_{4,j}$ are randomly generated from $\mathbb{Z}_N$ (the Paillier plaintext space). Since $E_{pk_p}(\cdot)$ is a semantically secure encryption scheme, a_j'' and b_j'' are computationally indistinguishable from $\delta_{1,j}$ and $\delta_{3,j}$, respectively. Also, as r_j and e_j are randomly generated from $\mathbb{Z}_N$, $a_j + r_j$ and $b_j + e_j$ are computationally indistinguishable from $\delta_{2,j}$ and $\delta_{4,j}$, respectively. Based on all these result, we can conclude that $\Pi_{C_2}(\text{SIP})$ is computationally indistinguishable from $\Pi^S_{C_2}(\text{SIP})$ according to Definition 1. This implies that during the execution of SIP, C_2 cannot figure out any information regarding $\boldsymbol{a}$ and $\boldsymbol{b}$. Actually, the information C_2 has during an execution of SIP is random, so this information does not leak anything of $\boldsymbol{a}$ and $\boldsymbol{b}$.

On the other hand, the execution image of C_1, denoted by $\Pi_{C_1}(\text{SIP}) = \{h', E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})\}$, where h' is the encrypted value received from C_2 at step 3(a) of Algorithm 1 and $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$ is the encrypted inner product computed by C_1 at step 3(f).

Let the simulated image of C_1 be $\Pi^S_{C_1}(\text{SIP}) = \{\delta_5, \delta_6\}$, where δ_5 and δ_6 are randomly generated from $\mathbb{Z}_{N^2}$. Since $E_{pk_p}(\cdot)$ is a semantically secure encryption scheme, it implies that h' and $E_{pk_p}(\boldsymbol{a} \cdot \boldsymbol{b})$ are computationally indistinguishable from δ_5 and δ_6, respectively. Therefore, $\Pi_{C_1}(\text{SIP})$ is computationally indistinguishable from $\Pi^S_{C_1}(\text{SIP})$ according to Definition 1, which means C_1 cannot learn any information regarding $\boldsymbol{a}$ and $\boldsymbol{b}$ during the execution of SIP. Combining everything together, we conclude the proposed SIP protocol is secure under the semi-honest model according to Definition 1.

B. PROXY Re-encryption (PRE) PROTOCOL

In a proxy Re-encryption (PRE) protocol [19], a proxy (a third party) can use a Re-encryption key $rk_{i\to j}$ to transform a ciphertext for one party under public key pk_i into a new ciphertext of the same plaintext under another party's public key pk_j. During the Re-encryption process, the proxy cannot deduce any information of the corresponding plaintext.

In this paper, we use a PRE protocol in [20] which is constructed on the ElGamal cryptosystem [21] with semantic security. The PRE protocol consists of the following five algorithms:

- $KeyGen(\mathbb{G}, p, g) \to \{pk_i, sk_i\}$: In the key generation algorithm, $\mathbb{G}$ is a multiplicative cyclic group of a prime order p with a generator g, such that discrete logarithm problem over the group $\mathbb{G}$ is hard. Then, it randomly chooses a secret key $sk_i \in Z_p^*$ and computes $h = g^{sk_i}$, then the public key $pk_i = (\mathbb{G}, p, g, h)$.
- $Enc(pk_i, m) \to \{m_i'\}$: The encryption algorithm takes the public key pk_i and a message $m \in \mathbb{G}$ as inputs, then it randomly chooses a number $r \in Z_p^*$ and outputs a ciphertext $m_i' = Enc(pk_i, m) = (g^r, m \cdot h^r)$.
- $ReEncKeyGen(sk_i, sk_j) \to \{rk_{i\to j}\}$: The algorithm of Re-encryption key generation takes two secret keys sk_i and sk_j as inputs, and outputs a Re-encryption key $rk_{i\to j} = sk_i - sk_j$.
- $ReEnc(rk_{i\to j}, m_i') \to \{m_j'\}$: The Re-encryption algorithm takes the Re-encryption key $rk_{i\to j}$ and the corresponding ciphertext m_i' as inputs, and outputs a new ciphertext $m_j' = (g^r, m \cdot h^r / g^{r\cdot rk_{i\to j}}) = (g^r, m \cdot g^{sk_i\cdot r}/g^{r\cdot(sk_i - sk_j)}) = (g^r, m \cdot g^{sk_j\cdot r}) = Enc(pk_j, m)$.
- $Dec(sk_j, m_j') \to \{m\}$: The decryption algorithm takes the secret key sk_j and a ciphertext $m_j' = Enc(pk_j, m) = (g^r, m \cdot g^{sk_j\cdot r})$ as inputs, then it outputs the plaintext $m = m \cdot g^{sk_j\cdot r}/(g^r)^{sk_j}$.

C. SECURE Re-encryption KEY GENERATION (SRKG) PROTOCOL

As shown in the $ReEncKeyGen(\cdot)$ algorithm of PRE protocol, the re-encryption key for the proxy is $rk_{i\to j} = sk_i - sk_j$. Now, we first consider C_1 to be the proxy with a Re-encryption key $rk_u = sk_c - sk_u$, where sk_c is DO's ElGamal secret key and sk_u is the ElGamal secret key of a QU u. To securely compute the Re-encryption key rk_u, we introduce the secure Re-encryption key generation (SRKG) protocol [20] in Algorithm 2, which keeps sk_c private to DO and sk_u private to QU.

```
Algorithm 2 SRKG($sk_c, sk_u$) → $rk_u$
  Require: DO has $sk_c$; QU has $sk_u$
  1: $C_1$: Choose a random number $\alpha \in Z_p^*$; send $\alpha$ to QU
  2: QU: $a \leftarrow sk_u + \alpha$; send $a$ to DO
  3: DO: $b \leftarrow sk_c - a$; send $b$ to $C_1$
  4: $C_1$: $rk_u \leftarrow b + \alpha$
```

In Algorithm 2, C_1 computes the Re-encryption key

$$rk_u = b + \alpha = sk_c - a + \alpha = sk_c - sk_u \tag{2}$$

where the ElGamal secret keys sk_c and sk_u are preserved from other parties by utilizing additive random perturbations. Specifically, DO cannot deduce sk_u from a without knowing the random number $\alpha \in Z_p^*$ (here p is the prime order of group $\mathbb{G}$ in ElGamal encryption). Similarly, C_1 cannot figure out sk_c and sk_u from $rk_u = sk_c - sk_u$ without knowing either of them. Obviously, QU cannot get sk_c because it only receives the random number α. According to the $ReEnc(\cdot)$ algorithm of PRE protocol, C_1 can use rk_u to re-encrypt the original encrypted classification labels c_i' (whose decryption key

Algorithm 3 CRRKG(sk_c, sk_u) $\to \{rk_{u_1}, rk_{u_2}\}$

```
Require: DO has $sk_c$; QU has $sk_u$
To generate $rk_{u_1}$:
  1: DO : Choose a random number $\alpha_1 \in Z_p^*, a \leftarrow sk_c + \alpha_1$; send $a$ to $C_1$
  2: $C_1$: Choose a random number $\alpha_2 \in Z_p^*, rk_{u_1} \leftarrow a - \alpha_2$
To generate $rk_{u_2}$:
  1: $C_2$: Choose a random number $\beta \in Z_p^*$; send $\beta$ to QU
  2: QU: $b_1 \leftarrow \beta - sk_u$; send $b_1$ to $C_1$
  3: $C_1$: $b_2 \leftarrow b_1 + \alpha_2$; send $b_2$ to DO
  4: DO: $b_3 \leftarrow b_2 - \alpha_1$; send $b_3$ to $C_2$
  5: $C_2$: $rk_{u_2} \leftarrow b_3 - \beta$
```

is sk_c) into new ciphertexts which can be decrypted using QU's secret key sk_u.

This is a simple secure protocol under the assumption that C_1 cannot collude with QU. However, in real-world situations, this assumption may not be reasonable because there are many QUs and it is very possible that one QU is corrupted for money or other benefits. If C_1 colludes with one QU and gets its secret key sk_u, it is very easy to compute sk_c by Eq. (2). Once the secret key sk_c is revealed, it severely threatens DO's database security. To better protect DO's secret key sk_c, we propose a new collusion resistant Re-encryption key Generation (CRRKG) protocol in next section.

D. COLLUSION RESISTANT Re-encryption KEY Generation (CRRKG) PROTOCOL

In this section, we propose a new collusion resistant Re-encryption key Generation (CRRKG) protocol that protects DO's ElGamal secret key sk_c from C_1 and C_2 even if they collude with QU. The new protocol is given in Algorithm 3 with two secret keys sk_c and sk_u as inputs and two Re-encryption keys rk_{u_1} and rk_{u_2} as outputs. We first discuss the detail of the protocol, then analyze its collusion resistant property.

As shown in Algorithm 3, two Re-encryption keys rk_{u_1} and rk_{u_2} are generated for C_1 and C_2 respectively. Under the non-collusion assumption between the two cloud servers, the trick of this protocol is that, by randomly generating two Re-encryption keys for C_1 and C_2, neither of them can get enough knowledge to deduce sk_c even if colluding with QU. Clearly, using rk_{u_1} and rk_{u_2}, there will be two Re-encryption processes performed by C_1 and C_2, and the details are given in the later Algorithm 4.

To generate rk_{u_1}, DO first randomizes its secret key sk_c by computing $a = sk_c + \alpha_1$ and sends it to C_1. Here α_1 is a random number in Z_p^* (p is the prime order of group $\mathbb{G}$ in ElGamal encryption) known only to DO. After reveiving a, C_1 also generates a random number α_2, and computes the first Re-encryption key

$$rk_{u_1} = a - \alpha_2 = sk_c - (\alpha_2 - \alpha_1) = sk_c - sk_1 \tag{3}$$

where the new ElGamal secret key

$$sk_1 = \alpha_2 - \alpha_1 \tag{4}$$

According to the *ReEnc*($\cdot$) algorithm of PRE protocol (Section III-B), now C_1 can use rk_{u_1} to re-encrypt the original encrypted labels c'_i into new ciphertexts c''_i which can be decrypted using the new secret key sk_1. However, C_1 cannot deduce α_1 from a, thus it cannot obtain sk_1 to decrypt the re-encrypted classification labels.

To generate rk_{u_2}, C_2 first picks up a random number β and sends it to QU. Then QU uses β to perturb its ElGamal secret key sk_u by computing $b_1 = \beta - sk_u$ and sends it to C_1. Next C_1 adds α_2 to b_1 and sends the result b_2 to DO. After receiving b_2, DO computes $b_3 = b_2 - \alpha_1$ and sends b_3 back to C_2, here α_1 is the random number to randomize DO's secret key sk_c. At last, C_2 computes the second Re-encryption key rk_{u_2} by removing the random number β from b_3 as below

$$rk_{u_2} = b_3 - \beta = (\alpha_2 - \alpha_1) - sk_u = sk_1 - sk_u \tag{5}$$

According to the *ReEnc*($\cdot$) algorithm of PRE protocol (Section III-B), now C_2 can use rk_{u_2} to re-encrypt the previous ciphertexts c''_i into new encrypted labels c'''_i which can be decrypted using QU's secret key sk_u. However, without knowing sk_1 and sk_u, C_2 cannot deduce either of them from rk_{u_2}, therefore it cannot decrypt c''_i and c'''_i to obtain the plain classification labels.

• *Proof of Security for CRRKG:* Now we analyze the security of our CRRKG protocol under the non-collusion assumption between C_1 and C_2.

Similar as in SRKG protocol, we first consider the simple circumstance that C_1 and C_2 cannot collude with QU. Without collusion, C_1 has the following knowledge:

$$(1)\ \alpha_2, \quad (2)\ a = sk_c + \alpha_1, \quad (3)\ b_1 = \beta - sk_u.$$

Apparently, C_1 cannot derive out sk_c from a without knowing α_1 and cannot derive out sk_u from b_1 without knowing β. Considering C_2, without collusion, it has the following knowledge:

$$(1)\ \beta, \quad (2)\ rk_{u_2} = (\alpha_2 - \alpha_1) - sk_u.$$

Obviously, C_2 cannot get sk_c and cannot deduce sk_u from rk_{u_2} without knowing α_1 and α_2. To summarize, our CRRKG protocol preserves both DO's secret key sk_c and QU's secret key sk_u from the two cloud servers.

Now we consider the situation that C_1 and C_2 may collude with QU to reveal DO's secret key sk_c. By colluding with QU, C_1 has the following knowledge:

$$(1)\alpha_2,\ (2)a = sk_c + \alpha_1,\ (3)\beta,\ (4)sk_u$$

Still, C_1 cannot figure out either DO' secret key sk_c or the random number α_1 from its new knowledge. Similarly, by collusion, the knowledge of C_2 is

$$(1)\beta,\ (2)rk_{u_2} = (\alpha_2 - \alpha_1) - sk_u,\ (3)sk_u$$

Apparently, C_2 can only derive out $\alpha_2 - \alpha_1$, but it cannot deduce sk_c based on its new knowledge. Therefore, even colluding with QU, neither C_1 nor C_2 can deduce DO's secret key sk_c under the assumption that they cannot collude.

IV. THE PROPOSED SCHEME

In this section, we propose a new privacy preserving kNN classification (PPKC) protocol over hybrid encrypted cloud database. Compared with the state-of-the-art PPkNN protocol in [1], our protocol is more efficient while achieving the same security and privacy properties: (1) database security, (2) query privacy and (3) hiding of data access patterns.

On one hand, similar as the previous work [1], we encrypt the database points using the Paillier cryptosystem, and utilize its homomorphic properties to perform computation over ciphertexts. On the other hand, to achieve better efficiency, our scheme is different from the work [1] in the following three aspects:

(1) We encrypt the classification labels using the ElGamal encryption instead of the Paillier encryption (used in [1]), so we can utilize the Re-encryption property of ElGamal cryptosystem to transform the encrypted labels.

(2) We compute encrypted inner products instead of encrypted squared Euclidean distances [1] under the Paillier encryption, and in this way less encryption and exponentiation operations are required.

(3) We use random permutation over intermediate computation results sent from C_1 to C_2 and efficiently hide data access patterns from both of the cloud servers.

Our proposed scheme consists of five different processes: (1) key generation; (2) database encryption; (3) query encryption; (4) encrypted kNN classification and (5) classification labels decryption. The details of each process are given as follows.

• *Key Generation*

In this process, DO randomly generates two public-secret key pairs: (1) a Paillier cryptosystem key pair (pk_p, sk_p) for database points $\boldsymbol{p}_i$; (2) an ElGamal cryptosystem key pair (pk_c, sk_c) for classification labels c_i.

• *Database Encryption*

Assume that DO's plain database $\boldsymbol{D}$ consists of n tuples $\boldsymbol{t}_i$, denoted by $\boldsymbol{D} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n\}$, and each $\boldsymbol{t}_i = \{\boldsymbol{p}_i, c_i\}$ contains a d-dimensional database point $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id})$ and a classification label c_i. Without loss of generality, we assume that $p_{ij} \in \mathbb{Z}_N$ (the Paillier plaintext space) and $c_i \in \mathbb{G}$ (the ElGamal plaintext space) for $i \in [n], j \in [d]$.

Since we introduce two different public key cryptosystems, to clearly distinguish them, we use $E_{pk_p}(\cdot)$ and $D_{sk_p}(\cdot)$ to denote the encryption and decryption operations for Paillier cryptosystem, and $Enc(pk_c, \cdot)$ and $Dec(sk_c, \cdot)$ to denote the encryption and decryption operations for ElGamal cryptosystem.

We first transform each $\boldsymbol{p}_i$ into a $(d + 1)$-dimensional point $\hat{\boldsymbol{p}}_i = (\boldsymbol{p}_i, -0.5 \left\|\boldsymbol{p}_i\right\|^2)$, which will be used to compute the inner product instead of the squared Euclidean distance. Then, for $i \in [n]$, DO computes the Paillier encrypted database points

$$\boldsymbol{p}'_i = (E_{pk_p}(\hat{\boldsymbol{p}}_{i,1}), E_{pk_p}(\hat{\boldsymbol{p}}_{i,2}), \ldots, E_{pk_p}(\hat{\boldsymbol{p}}_{i,d+1})) \tag{6}$$

and the ElGamal encrypted classification labels

$$c'_i = Enc(pk_c, c_i) \tag{7}$$

The hybrid encrypted database $\boldsymbol{D}' = \{\boldsymbol{t}'_1, \boldsymbol{t}'_2, \ldots, \boldsymbol{t}'_n\}$, in which the encrypted tuple $\boldsymbol{t}'_i = \{\boldsymbol{p}'_i, c'_i\}$. Then DO outsources $\boldsymbol{D}'$ to C_1, and outsources the Paillier secret key sk_p to C_2 for further interactive computations. After outsourcing, DO does not keep a copy of the plain database $\boldsymbol{D}$ and all further operations are performed by the two cloud servers.

• *Query Encryption*

For a private plain query point $\boldsymbol{q} = (q_1, q_2, \ldots, q_d)$, QU first generates a positive random number $\varepsilon \in \mathbb{Z}_N$ and transforms $\boldsymbol{q}$ into a $(d + 1)$-dimensional point $\hat{\boldsymbol{q}} = \varepsilon(\boldsymbol{q}, 1)$. Without loss of generality, we assume that each $q_j \in \mathbb{Z}_N$ for $j \in [d]$. Then, using DO's public key pk_p, QU computes the Paillier encrypted query point

$$\boldsymbol{q}' = (E_{pk_p}(\hat{\boldsymbol{q}}_1), E_{pk_p}(\hat{\boldsymbol{q}}_2), \ldots, E_{pk_p}(\hat{\boldsymbol{q}}_{d+1})) \tag{8}$$

and submits $\boldsymbol{q}'$ to C_1 for kNN classification.

As shown in the processes of database encryption and query encryption, we transform $\boldsymbol{p}_i$ and $\boldsymbol{q}$ into $\hat{\boldsymbol{p}}_i$ and $\hat{\boldsymbol{q}}$ before the Paillier encryption. This is because we will compute the encrypted inner products $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$ instead of the encrypted squared Euclidean distances $\left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|^2$ in the later classification process. Now we first prove the correctness of using $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$ to determine the kNN classification labels in Theorem 1 , then analyze the improvement of efficiency by this trick.

Theorem 1: The proposed scheme can correctly determine whether $\boldsymbol{p}_i$ is closer to $\boldsymbol{q}$ than $\boldsymbol{p}_j$ by evaluating $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} > \hat{\boldsymbol{p}}_j \cdot \hat{\boldsymbol{q}}$.

Proof: Note that

$$\begin{aligned}\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} - \hat{\boldsymbol{p}}_j \cdot \hat{\boldsymbol{q}} &= (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j) \cdot \hat{\boldsymbol{q}}\\ &= (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j)^T \hat{\boldsymbol{q}}\\ &= (\boldsymbol{p}_i - \boldsymbol{p}_j)^T (\varepsilon \boldsymbol{q}) + (-0.5 \left\|\boldsymbol{p}_i\right\|^2 + 0.5 \left\|\boldsymbol{p}_j\right\|^2)\varepsilon\\ &= 0.5\varepsilon(\left\|\boldsymbol{p}_j\right\|^2 - \left\|\boldsymbol{p}_i\right\|^2 + 2(\boldsymbol{p}_i - \boldsymbol{p}_j)^T \boldsymbol{q})\\ &= 0.5\varepsilon(\left\|\boldsymbol{p}_j - \boldsymbol{q}\right\|^2 - \left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|^2)\end{aligned} \tag{9}$$

In Eq. (9), $\left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|$ is the Euclidean distance between $\boldsymbol{p}_i$ and $\boldsymbol{q}$. Since $\varepsilon > 0$, the following equivalent condition is correct.

$$\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} > \hat{\boldsymbol{p}}_j \cdot \hat{\boldsymbol{q}} \Leftrightarrow \left\|\boldsymbol{p}_j - \boldsymbol{q}\right\|^2 > \left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|^2 \tag{10}$$

Therefore, we can use the inner products $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$ instead of the squared Euclidean distances $\left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|^2$ to determine the kNN classification labels. □

Now we analyze the reason why computing the encrypted inner products needs less computation cost than computing the encrypted squared Euclidean distances. Considering the computation complexity for plain values, each inner product $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$ takes $(d+1)$ multiplications and d additions, while each squared Euclidean distance $\left\|\boldsymbol{p}_i - \boldsymbol{q}\right\|^2$ takes d subtractions,

d multiplications and $(d-1)$ additions. If these operations are performed over plaintexts, the difference of the computation overheads is very small. However, using the homomorphic property, the computation of squared Euclidean distance over Paillier encrypted values needs more expensive encryptions and exponentiations.

Specifically, in our secure inner Product (SIP) protocol (Section III-A), computing $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$ needs $(2d+4)$ encryptions and $(2d+3)$ exponentiations. While using the SSED (Secure Squared Euclidean Distance) protocol in [1], the computation of encrypted $\|\boldsymbol{p}_i - \boldsymbol{q}\|^2$ needs d times of SM (Secure Multiplication) protocol [1] and d exponentiations. Since each SM protocol takes 3 encryptions and 3 exponentiations, the total computation cost of encrypted $\|\boldsymbol{p}_i - \boldsymbol{q}\|^2$ is $3d$ encryptions and $4d$ exponentiations. For the whole database, computing n encrypted squared Euclidean distances needs $7nd$ encryptions and exponentiations, while computing n encrypted inner products only needs $(4nd + 7n)$ encryptions and exponentiations. Apparently, when n and d are large, computing encrypted inner products using our SIP protocol is more efficient than computing encrypted squared Euclidean distances using the SSED protocol.

• *Encrypted kNN Classification*

This is the core process of our proposed scheme. In this process, we propose a new privacy preserving kNN classification (PPKC) protocol in Algorithm 4. By using our SIP protocol (Section III-A) and the PRE protocol (Section III-B) in our PPKC protocol, C_1 and C_2 collaborate to perform the kNN classification over the hybrid encrypted database and return the re-encrypted kNN classification labels to the corresponding QU. During this protocol, both database security and query privacy are well preserved, and data access patterns are also hidden from the two cloud servers.

As shown in Algorithm 4, the PPKC protocol takes the encrypted database $\boldsymbol{D}'$, the encrypted query point $\boldsymbol{q}'$, the Paillier secret key sk_p and two ElGamal Re-encryption keys rk_{u_1} and rk_{u_2} as input, and outputs a set $\boldsymbol{C}'_q$ consisting of the re-encrypted kNN classification labels. Here the two Re-encryption keys rk_{u_1} and rk_{u_2} are securely generated using our proposed CRRKG protocol (Section III-D). To point out, we only need to perform the CRRKG protocol once for each QU, while C_1 and C_2 will keep the corresponding Re-encryption keys for further kNN classification processes.

In the protocol, C_1 first performs several steps of computations, then sends the intermediate results to C_2, who performs the other steps and returns the re-encrypted kNN classification labels to the corresponding QU. Now we discuss our proposed PPKC in detail.

To start, C_1 randomly generates two one-time positive numbers $\lambda_1, \lambda_2 \in \mathbb{Z}_N$ (the Paillier plaintext space). Then, in step 1(b), C_1 first uses the SIP protocol to compute the Paillier encrypted inner product $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})$ between each encrypted database point $\boldsymbol{p}'_i$ and the encrypted query point $\boldsymbol{q}'$. Next, C_1 adds random noises to $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})$ and obtains $dst''_i = E_{pk_p}(\lambda_1 * \hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} + \lambda_2)$ according to the homomorphic properties of Paillier encryption. After that, C_1 uses the $ReEnc(\cdot)$ algorithm in the PRE protocol to compute the first round re-encrypted classification labels $c''_i = ReEnc(rk_{u_1}, c'_i)$. As mentioned in Section III-D, C_1 cannot decrypt the re-encrypted values because it does not know the new decryption key sk_1 (given in Eq. (4)) to c''_i. After that, C_1 generates a random permutation function π and uses it to permute the two sets $\boldsymbol{dst}''$ and $\boldsymbol{C}''$. Then, C_1 sends the intermediate computation results $\tilde{\boldsymbol{dst}} = \{\tilde{dst}_1, \tilde{dst}_2, \ldots, \tilde{dst}_n\}$ and $\tilde{\boldsymbol{C}} = \{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n\}$ to C_2. Here the random permutation prevents C_2 to figure out the correspondence between its received values and the encrypted database tuples, which hides data access patterns from C_2.

Algorithm 4 PPKC($\boldsymbol{D}', \boldsymbol{q}', sk_p, rk_{u_1}, rk_{u_2}$) $\rightarrow \boldsymbol{C}'_q$

```
Require: $C_1$ has $\boldsymbol{D}'$, $\boldsymbol{q}'$ and $rk_{u_1}$; $C_2$ has $sk_p$ and $rk_{u_2}$
1: $C_1$:
    (a). Generate two one-time positive random numbers $\lambda_1, \lambda_2 \in \mathbb{Z}_N$
    (b). For $i \in [n]$ do:
        • $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}) \leftarrow$ SIP($\boldsymbol{p}'_i, \boldsymbol{q}'$)
        • $dst''_i \leftarrow E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})^{\lambda_1} * E_{pk_p}(\lambda_2)$
        • $c''_i \leftarrow ReEnc(rk_{u_1}, c'_i)$
    (c). Generate a $n$-length random permutation function $\pi$
    (d). Perform the following permutations:
        • $\tilde{\boldsymbol{dst}} \leftarrow \pi(\boldsymbol{dst}'')$, where $\boldsymbol{dst}'' = \{dst''_1, dst''_2, \ldots, dst''_n\}$
        • $\tilde{\boldsymbol{C}} \leftarrow \pi(\boldsymbol{C}'')$, where $\boldsymbol{C}'' = \{c''_1, c''_2, \ldots, c''_n\}$
    (e). Send $\tilde{\boldsymbol{dst}}$ and $\tilde{\boldsymbol{C}}$ to $C_2$
2: $C_2$:
    (a). Receive $\tilde{\boldsymbol{dst}}$ and $\tilde{\boldsymbol{C}}$
    (b). For $i \in [n]$, $\hat{dst}_i \leftarrow D_{sk_p}(\tilde{dst}_i)$
    (c). Compute a fuction $Sort(\cdot)$ that places $\hat{dst}_i$ in a descending order
    (d). $\check{\boldsymbol{C}} \leftarrow Sort(\tilde{\boldsymbol{C}})$
    (e). For $j \in [k]$, $c'''_j \leftarrow ReEnc(rk_{u_2}, \check{c}_j)$
    (f). Send $\boldsymbol{C}'_q = \{c'''_1, c'''_2, \ldots, c'''_k\}$ to QU
```

After receiving $\tilde{\boldsymbol{dst}}$, C_2 uses the Paillier secret key sk_p to decrypt each $\tilde{dst}_i$ into the plain value $\hat{dst}_i$. Note that $\hat{dst}_i = \lambda_1 * \hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} + \lambda_2$, since both λ_1 and λ_2 are positive numbers, $\hat{dst}_i$ are in the same order with the corresponding inner products $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$. Therefore, according to Theorem 1, we can use $\hat{dst}_i$ to perform kNN search. Then, C_2 computes a fuction $Sort(\cdot)$ that places $\hat{dst}_i$ in a descending order since a larger inner product indicates a closer database point to the query point according to Theorem 1. After that, C_2 computes $\check{\boldsymbol{C}} = Sort(\tilde{\boldsymbol{C}})$ and picks the first k values $\check{c}_1, \check{c}_2, \ldots, \check{c}_k$ from $\check{\boldsymbol{C}}$. For each $\check{c}_j$, C_2 computes a second round classification label Re-encryption $c'''_j = ReEnc(rk_{u_2}, \check{c}_j)$ using the Re-encryption key $rk_{u_2} = sk_1 - sk_u$ (Eq. (5)). As discussed in Section III-D, now the new re-encrypted labels c'''_j can be decrypted using QU's ElGamal secret key sk_u. At last, C_2 sends the set of re-encrypted kNN classification labels $\boldsymbol{C}'_q$ to the corresponding QU.

In Algorithm 4, to ensure the correctness of comparison, C_1 needs to carefully choose λ_1 and λ_2 to make sure that the randomized inner products $\hat{dst}_i = \lambda_1 * \hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}} + \lambda_2$ are all smaller than N. As shown in the equation, the values also depend on the original inner products $\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}$. Since we cannot obtain every plain query point in advance, we can compute the inner products between database points and use their value range as an approximate estimation. This is reasonable because query points should locate in the same space of the database points. This process is a one-time work and can be performed by DO before outsourcing the database. Note that we also use a positive random number ε to multiplicatively perturb a query point before query encryption, which is only known by the corresponding QU. Therefore, C_1 should choose a relatively small λ_1 depending on the value range of inner products (provided by DO) and the value range of the random number ε (provided by QU). Since λ_2 is an additive perturbation, its value range can be relatively large while guaranteeing the values of the randomized inner products are all smaller than N.

- *Classification Labels Decryption*

After receiving $\boldsymbol{C}'_q$ from C_2, QU uses its ElGamal secret key sk_u to perform classification labels decryption $c_j = Dec(sk_u, c'''_j)$ and obtains the plain kNN classification labels $\boldsymbol{C}_q = \{c_1, c_2, \ldots, c_k\}$.

V. SECURITY ANALYSIS

Firstly, we emphasize that the submitted query point $\boldsymbol{q}'$ is encrypted using the semantically secure Paillier cryptosystem, therefore, QU's plain query point $\boldsymbol{q}$ is protected from DO, C_1 and C_2 in our PPKC protocol. Apart from preserving the plain query point, the goal of PPKC is to achieve database security, hide data access patterns and protect the plain kNN classification labels.

In this section, to prove the security of our PPKC protocol under the semi-honest model, we use the well-known security definition from the work of SMC (secure multi-party computation) as given in Section II-C. Based on Definition 1, we provide the formal security proof for PPKC under the semi-honest model as follows.

According to Algorithm 4, the execution image of C_2 is $\Pi_{C_2}(\text{PPKC})$ as below

$$\{\left\langle \tilde{dst}_i, \hat{dst}_i \right\rangle, \tilde{c}_i, c'''_j \,|\, for\ i, j\ \in [n], [k]\}$$

where $\tilde{dst}_i$ is the randomly permuted perturbed inner product under Paillier encryption and $\hat{dst}_i$ is the corresponding decrypted value, and $\tilde{c}_i$ is the permuted first-round ElGamal re-encrypted classification label while c'''_j is the second-round ElGamal re-encrypted label. Without loss of generality, let the simulated image of C_2 be $\Pi^S_{C_2}(\text{PPKC})$ as below

$$\{\left\langle \delta_{7,i}, \delta_{8,i} \right\rangle, \delta_{9,i}, \delta_{10,j} \,|\, for\ i, j\ \in [n], [k]\}$$

Here $\delta_{7,i}$ is randomly generated from $\mathbb{Z}_{N^2}$ (the Paillier ciphertext space) and $\delta_{8,i}$ is randomly generated from $\mathbb{Z}_N$ (the Paillier plaintext space), while $\delta_{9,i}$ and $\delta_{10,j}$ are randomly generated from $\mathbb{C}$ (the ElGamal ciphertext space). Since both $E_{pk_p}(\cdot)$ (the Paillier encryption) and $Enc(pk_c, \cdot)$ (the ElGamal encryption) are semantically secure encryption schemes, and the Re-encryption algorithm $ReEnc(rk_{u_2}, \check{c}_j)$ (Section III-B) does not disclose any information of the corresponding plaintext c_i, we claim that $\tilde{dst}_i$, $\tilde{c}_i$ and c'''_j are computationally indistinguishable from $\delta_{7,i}$, $\delta_{9,i}$ and $\delta_{10,j}$. Besides, since the random permutation function π is only known to C_1, $\hat{\boldsymbol{dst}} = \{\hat{dst}_1, \hat{dst}_2, \ldots, \hat{dst}_n\}$ is a random vector in $\mathbb{Z}_N$. Therefore, $\hat{\boldsymbol{dst}}$ is computationally indistinguishable from $\boldsymbol{\delta}_8 = \{\delta_{8,1}, \delta_{8,2}, \ldots, \delta_{8,n}\}$. This implies that C_2 cannot trace back to the corresponding database tuples, which hides data access patterns from C_2. Based on all these results, we can conclude that $\Pi_{C_2}(\text{PPKC})$ is computationally indistinguishable from $\Pi^S_{C_2}(\text{PPKC})$. This guarantees that C_2 learns nothing during the execution of PPKC protocol.

On the other hand, the execution image $\Pi_{C_1}(\text{PPKC})$ of C_1 is given by

$$\{E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}}), c''_i \,|\, for\ i\ \in [n]\}$$

where $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})$ is the Paillier encrypted inner product and c''_i is the first round ElGamal re-encrypted classification label. Let the simulated image $\Pi^S_{C_1}(\text{PPKC})$ of C_1 be given by

$$\{\delta_{11,i}, \delta_{12,i} \,|\, for\ i\ \in [n]\}$$

where $\delta_{11,i}$ is randomly generated from $\mathbb{Z}_{N^2}$ and $\delta_{12,i}$ is randomly generated from $\mathbb{C}$. Since we proved that our SIP protocol is secure in Section III-A and the Re-encryption algorithm $ReEnc(rk_{u_1}, c'_i)$ does not disclose any information of the corresponding plaintext c_i, we claim that $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})$ and c''_i are computationally indistinguishable from $\delta_{11,i}$ and $\delta_{12,i}$. This implies that $\Pi_{C_1}(\text{PPKC})$ is computationally indistinguishable from $\Pi^S_{C_1}(\text{PPKC})$. Therefore, C_1 cannot deduce anything during the execution of PPKC protocol. Apparently, since $E_{pk_p}(\hat{\boldsymbol{p}}_i \cdot \hat{\boldsymbol{q}})$ are encrypted values, C_1 cannot use them to determine the kNN results, which hides data access patterns from C_1. Based on all these results, we claim that our proposed PPKC protocol is secure under the semi-honest model.

VI. PERFORMANCE EVALUATION

We analyze the cost of our proposed PPKC protocol and evaluate its performance with experiments in this section.

A. COST ANALYSIS

As given in Section II-A, n is the number of database tuples, d is the dimension of data points and k is the number of nearest neighbors. In our proposed PPKC protocol, C_1 and C_2 together need to perform n times SIP protocol, $n + k$ classification label re-encryptions, n Paillier decryptions and other random permutation and sort operations. As discussed in Section IV, for the whole database, computing n times SIP protocol takes $(4nd + 7n)$ encryptions and exponentiations, while computing n times SSED (Secure Squared Euclidean Distance) protocol [1] needs $7nd$ encryptions and exponentiations. Therefore, when n and d are large, using SIP protocol needs less computation cost than using the SSED protocol. In addition, the PPkNN protocol in [1] needs to execute n times SBD (secure bit-decomposition) sub-protocol and k times SMIN_n (secure minimum out of n numbers) sub-protocol. Since the SMIN_n sub-protocol is computationally expensive, our PPKC protocol is much more efficient than the PPkNN protocol while achieving the same security properties. Besides, as shown in the experiment results in [1], the computation costs increase linearly with the number of nearest neighbors k, while the computation cost of our PPKC protocol is not mainly subject to k. In next section, we show the performance of our PPKC protocol in experiments by comparing it with the state-of-the-art PPkNN protocol in [1].

B. EXPERIMENT RESULTS

In this section, we present some experiments to demonstrate the performance of our PPKC protocol under different parameter settings using Python language. All experiments were conducted on Windows with Intel Core i7 2.8 GHz CPU and 8 GB RAM. We compare our PPKC protocol with the state-of-the-art PPkNN protocol in [1]. The results show that our PPKC protocol is much more efficient than the PPkNN protocol while achieving the same security properties.

Our experiments are performed on synthetic databases. A synthetic database consists of n database tuples in which each database tuple contains an uniformly distributed random d-dimensional database point and a numerical classification label. We encrypted the database points using the Paillier encryption whose key size K_1 is 1024 bits, and encrypted the classification labels using the ElGamal encryption whose key size K_2 is 1024 bits. We ran random queries over the encrypted database and returned the re-encrypted kNN classification labels.

We do not discuss the computation cost for DO since it is a one-time cost. Considering the on-line computation cost for a QU, encrypting a query point needs about 0.1 s when $d = 5$, while decrypting an encrypted label only needs about 4 ms, i.e., it also needs about 0.1 s to decrypt all the received kNN labels when $k = 25$. Compared with the computationally expensive on-line encrypted kNN classification, the computation burden for QU is almost negligible and acceptable in real-world situation.

To clearly show the performance and comparison affected by different values of n, d and k, we present three different experiments by changing one parameter and keep the other two parameters fixed. Here, n is the number of database tuples, d is the dimension of data points and k is the number of nearest neighbors. The detailed experiment results and analysis are given as follows.

1) VARYING THE NUMBER OF DATABASE TUPLES n

In this experiment, we randomly generate databases by varying n from 1000 to 5000 with fixed $d = 5$. For each synthetic database, we run 100 random queries with fixed $k = 5$ and record the average execution time. The performance of our PPKC protocol and the PPkNN protocol are shown in Table 1 and Figure 2.

TABLE 1. Average execution time vs. n ($d = 5, k = 5$).

n	1000	2000	3000	4000	5000
PPkNN (in min)	36.62	69.53	104.36	139.03	173.92
PPKC (in s)	23.72	38.06	57.47	75.99	95.30

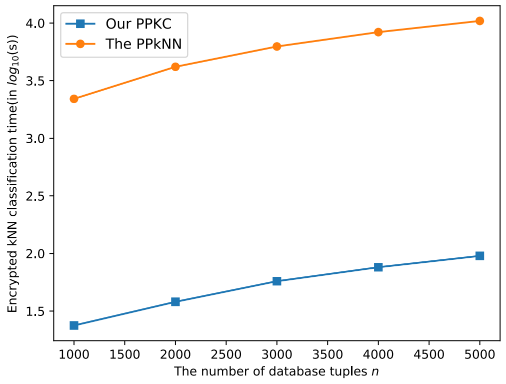

FIGURE 2. Average execution time (in $log_{10}(s)$) vs. n ($d = 5, k = 5$).

As shown in Table 1, the computation costs for both of the two protocols increase along with the number of database tuples n. For our PPKC protocol, it takes 23.72 s to execute one encrypted kNN classification process over a small database with $n = 1000$ and takes 95.30 s for a relatively large database with $n = 5000$. While for the PPkNN protocol, it needs 36.62 minutes for $n = 1000$ and 173.92 minutes (almost 3 hours) for $n = 5000$. Apparently, our PPKC protocol is much more efficient than the PPkNN protocol. The main reason is that in our PPKC protocol we only utilize proxy Re-encryption (PRE) and random permutation to hide data access patterns, whose costs are relatively cheap. While in PPkNN protocol, it needs to execute n times SBD (secure bit-decomposition) sub-protocol and k times computationally expensive SMIN_n (secure minimum out of n numbers) sub-protocol [1]. Another reason is that we use our SIP sub-protocol instead of SSED (Secure Squared Euclidean Distance) sub-protocol (used in PPkNN), so we only need to compute $(4nd + 7n)$ encryptions and exponentiations instead of $7nd$ encryptions and exponentiations. Therefore, when n is large, using SIP sub-protocol needs less computation cost than using the SSED sub-protocol.

To better show the efficiency of our PPKC protocol, we also draw the logarithmic computation cost in Figure 2. It is clear that our PPKC protocol needs about two orders of magnitude less execution time than the PPkNN protocol. Therefore, with the same security and privacy properties, our PPKC protocol is more practical than the previous PPkNN protocol.

2) VARYING THE DIMENSION OF DATA POINTS d

In this experiment, we randomly generate databases by varying d from 5 to 25 with fixed $n = 1000$. For each synthetic database, we run 100 random queries with fixed $k = 5$ and record the average execution time. The performance of our PPKC protocol and the PPkNN protocol are shown in Table 2 and Figure 3.

TABLE 2. Average execution time vs. d ($n = 1000, k = 5$).

d	5	10	15	20	25
PPkNN (in min)	36.77	37.17	37.59	37.97	38.35
PPKC (in s)	22.83	27.70	35.92	40.47	47.78

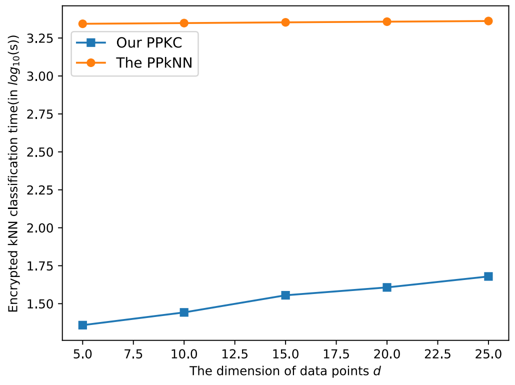

FIGURE 3. Average execution time (in $log_{10}(s)$) vs. d ($n = 1000, k = 5$).

As shown in Table 2, the computation costs for our PPKC protocol increase along with the dimension of data points d, while the costs for the PPkNN protocol almost stay the same when d changes (only increase a little with d). For our PPKC protocol, it takes 22.83 s to execute one encrypted kNN classification process with a small data dimension $d = 5$ and takes 47.78 s for a relatively large data dimension $d = 25$. While for the PPkNN protocol, it needs 36.77 minutes for $d = 5$ and 38.35 minutes for $d = 25$. Obviously, our PPKC protocol still needs much less computation cost than the PPkNN protocol when data dimension d grows. The reason why the execution time of the PPkNN protocol remain almost unchanged is that only the SSED sub-protocol in PPkNN is subject to the data dimension d while the main computation cost comes from the execution of k times SMIN$_n$ sub-protocol. We will see that the PPkNN protocol needs apparently more computation cost when the number of nearest neighbors k increases in the next experiment.

To clearly show performance improvement of our PPKC protocol compared with the PPkNN protocol, we also draw the logarithmic computation cost in Figure 3. As shown in the figure, the execution time of the PPkNN protocol stay unchanged, while our PPKC protocol takes more time when d grows. However, when $d = 25$, the computation cost of our PPKC protocol is still about 1.7 orders of magnitude lower than that of the PPkNN protocol. Hence, our PPKC protocol also performs much more efficient than the PPkNN protocol when data dimension d increases.

3) VARYING THE NUMBER OF NEAREST NEIGHBORS k

In this experiment, we randomly generate one database with fixed $n = 1000$ and $d = 5$. Over this synthetic database, we vary the number of nearest neighbors k from 5 to 25 and run 100 random queries for each value of k. We record the average execution time in Table 3 and draw the logarithmic computation cost in Figure 4.

TABLE 3. Average execution time vs. k ($n = 1000, d = 5$).

k	5	10	15	20	25
PPkNN (in min)	36.89	67.63	117.35	149.72	163.75
PPKC (in s)	23.32	23.85	24.13	24.42	24.73

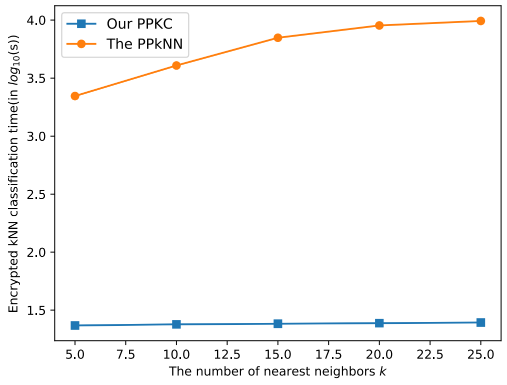

FIGURE 4. Average execution time (in $log_{10}(s)$) vs. k ($n = 1000, d = 5$).

As we can see in Table 3, the computation costs for the PPkNN protocol grow along with the number of nearest neighbors k, while the costs for our PPKC protocol remain almost the same when k varies (only grow a little with k). For our PPKC protocol, it takes 23.32 s to execute one encrypted kNN classification process with a small $k = 5$ and takes 24.73 s for a relatively large $k = 25$. While for the PPkNN protocol, it needs 36.89 minutes when $k = 5$ and 163.75 minutes (almost 3 hours) when $k = 25$. Obviously, our PPKC protocol is much more efficient than the PPkNN protocol especially when more nearest neighbors are required (a larger k). The reason why the execution time of our PPKC protocol remain almost unchanged is that only the computations of the second round re-encryptions (Algorithm 4 Step 2(e)) are subject to the number of nearest neighbors k, which are not the main computation cost in PPKC. On the contrary, as discussed before, the main computation cost of PPkNN comes from the execution of k times SMIN$_n$ sub-protocol, making the protocol computationally expensive when k is large.

To better show the efficiency of our PPKC protocol, we also draw the logarithmic computation cost in Figure 4. As we can see in the figure, the execution time of our PPKC protocol stay the same, while the PPkNN protocol needs more time when k increases. Specifically, when $k = 5$, our PPKC protocol is about 2 orders of magnitude more efficient than the PPkNN protocol and when $k = 25$, our PPKC protocol achieves 2.5 orders of magnitude faster than the PPkNN protocol. Apparently, the PPkNN protocol is very slow when more nearest neighbors are required (a larger k), while our PPKC protocol keeps efficient when the number of nearest neighbors k increases.

VII. RELATED WORK

In recent years, a great number of works have been proposed to deal with data security and user privacy, such as access control techniques [22], [23], searchable encryption schemes [24], [25] and and privacy preserving data mining methods [26]–[28]. Recently, to efficiently perform comparison over encrypted data, Li *et al.* [29] propose a protocol based on a hybrid approach of Paillier cryptosystem and garbled circuits. Nevertheless, none of these existing works can be directly applied to solve the problem of secure kNN query over encrypted cloud database.

To protect data security in outsourced classification scenario, Rahulamathavan *et al.* [30] propose a privacy preserving SVM classification scheme by using the Paillier cryptosystem. Pointed out by Li *et al.* [31], the scheme in [30] has some soundness and security problems and they present a new scheme fixing the problems with better efficiency. Although both of the schemes focus on privacy preserving classification, they only provide a client-server model in their protocols. Moreover, the normalization and SVM parameters in the server side are all plain values, while the outsourced databases for kNN classification are generally in encrypted form. Thus, we also cannot use the schemes in [30] and [31] to deal with the problem of privacy preserving kNN classification.

One method to execute encrypted kNN query is the distance-preserving transformation (DPT) [32], which is proved to be insecure in the work [33]. Later, Wong *et al.* [3] provide a proof that distance-recoverable encryptions (DRE), such as DPT, are vulnerable against a low level attack in which the attacker have access to the encrypted database and some plain database tuples (the level-2 attack in [3]). To better protect the private database, Wong *et al.* [3] propose an asymmetric scalar-product-preserving encryption (ASPE) scheme, in which the encryption (decryption) key for both database points and query points is an invertible random matrix. Since the basic ASPE scheme can only resist the level-2 attack, the authors also propose an enhanced ASPE scheme that introduces random asymmetric splitting with additional artificial dimensions, making it secure against the level-3 attack [3]. Considering its good security property and efficiency, the ASPE scheme has been utilized as a black-box in many other works, such as [34]–[36].

Other works [4]–[6] present some schemes to approximately perform the secure kNN query. Yiu *et al.* [4] propose several transformation methods which provide some trade-offs among data security, query computation cost and result accuracy. Yao *et al.* [5] present a secure kNN query scheme based on the partition-based secure Voronoi diagram (SVD). Xu *et al.* [6] propose a random space perturbation (RASP) technique to securely perform the kNN query. In addition, Choi *et al.* [37] present two secure kNN query methods (VD-kNN and TkNN) based on the mutable order-preserving encoding (mOPE) [38]. However, all these works [3]–[6], [37] assume that the query users are trusted who have access to DO's secret key. Under this assumption, once one QU is corrupted, the attacker can easily break the encrypted database.

To perform query encryption without sharing DO's secret key, Zhu *et al.* [7], [8] propose a scheme using an interactive protocol between QU and DO. In this protocol, a QU can only figure out partial information of DO's secret key. Later, Zhu *et al.* [9] propose another scheme introducing the Paillier cryptosystem in query encryption. However, the use of the Paillier cryptosystem brings about heavy computation cost for both QU and DO, making the process inefficient. Although these works [7]–[9] protect DO's secret key against the query users, they require DO staying on-line to participate during the process of query encryption. Nevertheless, this on-line requirement may not be practical in real-world scenarios since it introduces extra computation and communication overheads for DO [39]. To address this problem, Zhou *et al.* [39] present an efficient secure kNN query scheme over encrypted database revealing only limited information of DO's secret key to query users without DO's on-line participation. Although these works [7]–[9], [39] protect DO's secret key, they can neither resist a high level attack (the level-3 attack [3]) nor hide data access patterns from cloud servers.

To hide data access patterns from cloud servers, Elmehdwi *et al.* [10] introduce two non-colluding semi-honest clouds to securely perform kNN query over Paillier encrypted database. In this work, based on the homomorphic properties of Paillier cryptosystem, several secure sub-protocols, such as SSED (secure squared Euclidean distance) protocol, SBD (secure bit-decomposition) protocol, SMIN (secure minimum) protocol and SMIN$_n$ (secure minimum out of n numbers) protocol are used to build the SkNN$_m$ (fully secure kNN) protocol. Although this is the first work that hides data access patterns from cloud servers, it introduces very high computation cost which may prevent its application over large databases. Besides, the computation complexity is subject to the number of nearest neighbors k, since k times computationally expensive SMIN$_n$ sub-protocol are required in one kNN query process. Based on [10], by introducing other sub-protocols, such as SF (secure frequency) protocol and SCMC$_k$ (secure computation of majority class) protocol, a PPkNN (privacy preserving kNN) classification protocol is proposed in [1]. Similarly, this work also hides data access patterns with expensive computation costs. Recently, Rong *et al.* [40] propose a collaborative kNN protocol base on a set of secure building blocks using the ElGamal encryption in multiple cloud environments. The protocol also hides data access patterns and is designed without bit-decomposition (used in [1]), and therefore needs less computation cost than the protocol in [1]. Nevertheless, as shown in the experiment results [40], performing their protocol with $k = 5$ still needs 35.59 minutes over an encrypted database with $n = 4898$ and $d = 12$. Compared with our PPKC protocol, which only needs 95.30 s (about 1.6 minutes) when $k = 5$, $n = 5000$ and $d = 5$ (in Table 1), the protocol in [40] also requires much more computation cost to achieve the same security properties.

In our previous work [11], we design an efficient privacy preserving kNN classification scheme, which protects database security, DO's key confidentiality, query privacy and hides data access patterns. Both this work and the previous one use random permutation method to hide data access patterns. To guarantee that only the corresponding QU can correctly decrypt the kNN classification labels, we present different Re-encryption and Re-encryption key generation approaches in the two works, and the new proposed CRRKG protocol can also resist collusion between cloud server and QU. We emphasize that this new work utilizes Paillier and ElGamal cryptosystems which provide semantic security for the outsourced database, while our previous work does not achieve this security property. On the other hand, as shown in the experiments in [11], the previous scheme is very efficient even for large database ($n = 1000K$) and big data dimension ($d = 500$). Therefore, if the database is highly sensitive, DO can choose this new scheme for better privacy protection, while if the required privacy level is low, DO can choose our previous scheme for a better computation performance.

VIII. CONCLUSION AND FUTURE WORK

This paper proposed a novel efficient protocol PPKC addressing the problem of privacy preserving kNN classification over semantically secure encrypted cloud database. The proposed protocol protects database security, query privacy and hides data access patterns from cloud servers. We formally proved the security of our PPKC protocol and evaluated its efficiency in experiments compared with the state-of-the-art PPkNN protocol. The experiment results showed that our PPKC protocol is much more efficient than the PPkNN protocol while achieving the same security properties. For future work, we plan to investigate and extend our work with more practical scenarios, such as performing privacy preserving kNN classification in multiple clouds environments.

REFERENCES

[1] 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. 1261–1273, May 2015.

[2] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Tech. Rep., 2011.

[3] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[4] M. L. Yiu, I. Assent, C. S. Jensen, and P. Kalnis, "Outsourced similarity search on metric data assets," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 338–352, Feb. 2012.

[5] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. 29th Int. Conf. Data Eng. (ICDE)*, Apr. 2013, pp. 733–744.

[6] 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.

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

[8] Y. Zhu, R. Xu, and T. Takagi, "Secure k-NN query on encrypted cloud database without key-sharing," *Int. J. Electron. Secur. Digit. Forensics*, vol. 5, nos. 3–4, pp. 201–217, 2013.

[9] Y. Zhu, Z. Huang, and T. Takagi, "Secure and controllable k-NN query over encrypted cloud data with key confidentiality," *J. Parallel Distrib. Comput.*, vol. 89, pp. 1–12, Mar. 2016.

[10] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng. (ICDE)*, Mar./Apr. 2014, pp. 664–675.

[11] W. Wu, U. Parampalli, J. Liu, and M. Xian, "Privacy preserving K-nearest neighbor classification over encrypted database in outsourced cloud environments," in *Proc. World Wide Web*, 2018, pp. 1–23.

[12] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, and T. Schneider, "Twin clouds: An architecture for secure cloud computing," in *Proc. Workshop Cryptogr. Secur. Clouds (WCSC)*, 2011, pp. 1–11.

[13] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, Nov. 1982, pp. 160–164.

[14] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 9th Annu. ACM Symp. Theory Comput.*, 1987, pp. 218–229.

[15] O. Goldreich, "General cryptographic protocols," in *The Foundations of Cryptography*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[16] O. Goldreich, "Encryption schemes," in *The Foundations of Cryptography*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004, pp. 373–470. [Online]. Available: http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/enc.ps

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT*, vol. 99. Prague, Czech Republic: Springer, 1999, pp. 223–238.

[18] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[19] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1998, pp. 127–144.

[20] J. Liu, H. Wang, M. Xian, H. Rong, and K. Huang, "Reliable and confidential cloud storage with efficient data forwarding functionality," *IET Commun.*, vol. 10, no. 6, pp. 661–668, Apr. 2016.

[21] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.

[22] H. Wang, Y. Zhang, and J. Cao, "Effective collaboration with information sharing in virtual universities," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 6, pp. 840–853, Jun. 2009.

[23] M. Li, X. Sun, H. Wang, Y. Zhang, and J. Zhang, "Privacy-aware access control with trust management in Web service," in *Proc. World Wide Web*, vol. 14, no. 4, 2011, pp. 407–430.

[24] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[25] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.

[26] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," *Inf. Syst.*, vol. 29, no. 4, pp. 343–364, 2004.

[27] C. C. Aggarwal and S. Y. Philip, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-Preserving Data Mining*. Boston, MA, USA: Springer, 2008, pp. 11–52.

[28] M. E. Kabir, H. Wang, and E. Bertino, "Efficient systematic clustering method for k-anonymization," *Acta Inf.*, vol. 48, no. 1, pp. 51–66, 2011.

[29] X.-X. Li, Y.-W. Zhu, and J. Wang, "Efficient encrypted data comparison through a hybrid method," *J. Inf. Sci. Eng.*, vol. 33, no. 4, pp. 953–964, 2017.

[30] Y. Rahulamathavan, R. C.-W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 5, pp. 467–479, Sep./Oct. 2014.

[31] X. Li, Y. Zhu, J. Wang, Z. Liu, Y. Liu, and M. Zhang, "On the soundness and security of privacy-preserving SVM for outsourcing data classification," *IEEE Trans. Dependable Secure Comput.*, to be published.
[32] S. R. M. Oliveira and O. R. Zaiane, "Privacy preserving clustering by data transformation," in *Proc. Brazilian Symp. Databases*, 2003, pp. 304–318.
[33] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Knowledge Discovery in Databases: PKDD*. Berlin, Germany: Springer-Verlag, 2006, pp. 297–308.
[34] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. INFOCOM*, Apr. 2011, pp. 829–837.
[35] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *Proc. 31st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2011, pp. 393–402.
[36] W. Sun *et al.*, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, 2013, pp. 71–82.
[37] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure kNN query processing in untrusted cloud environments," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2818–2831, Nov. 2014.
[38] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2013, pp. 463–477.
[39] L. Zhou, Y. Zhu, and A. Castiglione, "Efficient k-NN query over encrypted data in cloud with limited key-disclosure and offline data owner," *Comput. Secur.*, vol. 69, pp. 84–96, Aug. 2016.
[40] H. Rong, H.-M. Wang, J. Liu, and M. Xian, "Privacy-preserving k-nearest neighbor computation in multiple cloud environments," *IEEE Access*, vol. 4, pp. 9589–9603, 2016.

WEI WU is currently pursuing the Ph.D. degree with the School of Electronic Science, National University of Defense Technology. His research interests include privacy preserving data mining and cloud computing security.

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 School of Electronic Science, National University of Defense Technology. His research interests include secure data outsourcing in cloud computing, public auditing, and access control mechanisms.

HONG RONG received the M.S. and Ph.D. degrees from the National University of Defense Technology in 2013 and 2018, respectively. His research interests include privacy-preserving data mining, cloud computing security, and big data security.

HUIMEI 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 School of Electronic Science, National University of Defense Technology. Her research interests include evaluation techniques in network security, cloud computing, and distributed systems.

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 School of Electronic Science, 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.