Received January 11, 2018, accepted April 19, 2018, date of publication April 26, 2018, date of current version May 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2830355

Towards Secure Approximate *k*-Nearest Neighbor Query Over Encrypted High-Dimensional Data

YANGUO PENG[1], HUI LI[2,3], JIANGTAO CUI[1,3], JIANFENG MA[2,3], AND YINGFAN LIU[4], (Member, IEEE)
[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[2]School of Cyber Engineering, Xidian University, Xi'an 710071, China
[3]Shaanxi Key Laboratory of Network and System Security, Xidian University, Xi'an 710071, China
[4]Department of System Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong

Corresponding author: Jiangtao Cui (cuijt@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702403, Grant 61472298, Grant 61672408, Grant and 61702404, in part by the Fundamental Research Funds for the Central Universities under Grant JB170308 and Grant BDZ011402, in part by the China 111 Project under Grant B16037, in part by the Key Program of NSFC under Grant U1405255, and in part by the Shaanxi Science & Technology Coordination & Innovation Project under Grant 2016TZC-G-6-3.

ABSTRACT As many services have been subcontracted to the cloud, data in such services (i.e., management applications, signal processing, and so on) are stored on cloud in encrypted form for protecting user private information. In many emerging data and signal processing applications, *approximate k-nearest neighbor* (ANN) query is a prerequisite component for massive high-dimensional data (such as time series, biometric, signal, multimedia data, and so on). Unfortunately, existing *secure ANN* (SANN) methods encounter the limitation of dimensionality. To further resolve ANN query over high-dimensionally encrypted data, in this paper, we propose an effective SANN model in Euclidean space. In overview, a secure greedy partition method is carefully designed by applying locality sensitive hashing coding and an optimized linear order. Based on that, a novel *partition-based SANN* (SANN$_P$) and a multi-division version mSANN$_P$ resolve SANN query by sequentially scanning a candidate set, which is produced by matching a cloak query with a map index. Our proposed solutions guarantee security and accuracy simultaneously, and reduce communication cost significantly. Meanwhile, the greedy partition method is proved to be *indistinguishable secure under chosen-plaintext attack*, which is the foundation of security for the proposed solutions. Through extensive experimental studies on four data sets, the proposed mechanisms outperform the state-of-the-art approaches and provide effective and tradeoff between result accuracy and communication cost.

INDEX TERMS Secure approximate *k*-nearest neighbor, locality sensitive hashing, greedy partition method, IND-CPA.

I. INTRODUCTION

Cloud computing has provided efficient and on-demand data management for various organizations with minimal hardware investment and maintenance overhead. In fact, even though the outsourced sensitive data are strictly protected by the cloud's defence mechanisms, secure accidents to cloud are ongoing frequently, such as user information leakage in CSDN, iCloud photo leak, etc. A serious security problem is that the control of data is passed over from consumer to the cloud, due to which the risks of data security have been raised to an unprecedented level. Intuitively, encrypting outsourced data with consumer's security policy paves the way for taking back full control of data and protecting consumer's sensitive data.

Recently, *approximate k-nearest neighbor* (ANN) has been more preferred in high-dimensional data management [6]–[12], which mitigates ''Curse of Dimensionality'' [5] in *k*-nearest neighbor query schemes [1]–[3]. However, for high-dimensional database applications, such solutions are impractical due to the limitation of dimensionality, risk of security or expensive cryptographic computation.

To our best knowledge, there is little progress in *secure ANN* (SANN) for high-dimensional database applications, which is attracting more and more attentions [14], [15]. In this paper, we propose a novel SANN solution in Euclidean space, named *partition-based SANN* (SANN$_P$) model. The contributions of this work are as follows:

- A delicate partition method is the kernel in SANN model. We design a secure greedy partition method by carefully levering locality sensitive hashing function and an optimized linear order. In the method, neighboring objects are gathered with a great probability, which steadies the accuracy of ANN query result.
- We present a pair of novel SANN models, SANN$_P$ and *multi-division SANN$_P$* (mSANN$_P$), based on the new-defined greedy partition method. Both models are proved secure in the sense of IND-CPA with respect to a single query.
- Experimental studies are conducted over real and synthetic datasets. The results show that our work exhibits better performance comparing with three other state-of-the-art SANN approaches in aspects of response time and result accuracy.

A. APPLICATION SCENARIOS

Many emerging database applications (e.g., time series, biometric and scientific databases) in the cloud are performing *k-nearest neighbor* (*k*NN) over high-dimensional data. For instance, in medical institutions, an ordinary task may always be performed as follows. A new patient's *electrocardiogram* (ECG) may be submitted as a query, in order to find the most similar patients in existing medical records for diagnostics. Feature vectors extracted from ECGs are always represented as vectors in high-dimensional Euclidean space for recognition, search, etc. [4]. Similarly, for a given query, in biometric information retrieval systems which outsource existing biological information to the cloud, the *k*NN result should be returned. In such cases as well as content based retrieval systems in multimedia databases, the queries and data are often represented as high-dimensional data. In these applications, a query itself may be approximate (caused by damage, missing, etc.) or the *k*NN result should be approximate for practical consideration.

Formally, let $D = \{\boldsymbol{x_i} | \boldsymbol{x_i} \in \mathbb{R}^d \wedge 1 \leq i \leq n_D\}$ be the dataset, where d is the dimensionality and $n_D = \#(D)$ is the cardinality of D. We denote by $\boldsymbol{q}$ as the query, $SD_i \subseteq D$ as the subset and $\Phi \subseteq D$ as the candidate set which contains ANN result corresponding to $\boldsymbol{q}$. For ease of illustration, we use the notations in Table 1. The elements of the dataset are vectors, which are denoted with lowercase letters in bold.

Similar to [1], participants in SANN are also defined as three entities as shown in Fig. 1.

1) *Owner* owns the dataset, preprocesses the original data and outsources the preprocessed encrypted subsets to the cloud server.
2) *Client*, which may not necessarily be the *Owner*, is authorized to conduct a cloak query T by single-round interaction with the cloud server.
3) Cloud server (*Server* for short) holds dataset from *Owner* and looks up the subset with T.

TABLE 1. Primary notations.

Name	Meaning
D	A dataset.
$\widehat{D}$	An outsourced dataset.
SD_i	A subset of D.
$\widehat{SD_i}$	An outsourced subset associated with a tag.
Φ	A returned candidate set for answering query.
$\boldsymbol{p}, \boldsymbol{q}$	A vector and a query vector, respectively.
$C_{\boldsymbol{q}}$	A code of vector $\boldsymbol{q}$.
$E(X)$	The cipher of X. E is the encryption algorithm.
t_i	A tag for indexing the subset.
T	A clock query (which includes a set of tags).
h	A *Locality Sensitive hashing* (LSH) function.
G_m	A G function that contains m LSH functions.
m	The number of LSH functions in G_m.
$\#(X)$	The number of elements in X.
$\parallel \boldsymbol{q} - \boldsymbol{v} \parallel$	Euclidean distance between $\boldsymbol{q}$ and $\boldsymbol{v}$.
I	A map index for subsets.
$[X]_\sim$	An equal-code set. X is a code of vector.

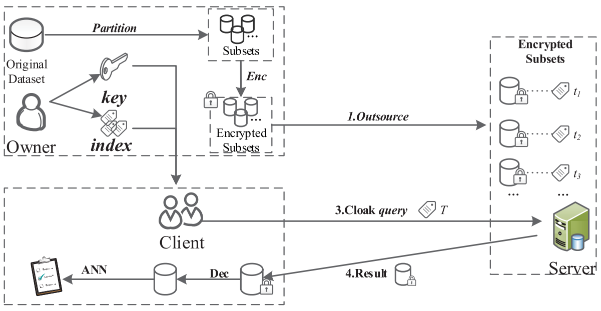


FIGURE 1. Framework of SANN$_P$.

Notably, within the framework in Fig. 1, both query $\boldsymbol{q}$ and dataset D contain confidential information. Before D is encrypted, it is divided into several subsets SD_i where $\cup SD_i = D$. A map index I for subsets is generated off-line. Then, SD_i is encrypted under *Owner*'s secret key *sk* with an *indistinguishability under chosen-plaintext attack* (IND-CPA) secure encryption algorithm, and is allocated a unique tag. All subsets with their own unique tags are outsourced to *Server*. Secret key *sk* and map index I are sent to *Client* in a secure channel (such as SSL).[1] *Client* generates cloak query T in order to find subset(s) in which the close vectors of $\boldsymbol{q}$ falls **with a sufficiently large probability**. T is sent to *Server* instead of original $\boldsymbol{q}$. Finally, *Server* returns the candidate set Φ whose unique tags match T. When *Client* receives a response, he decrypts the candidate set and find the ANN result from it. In fact, we find *k*NNs over the decrypted subsets, and the *k*NNs are the ANN result of the whole dataset D.

B. OVERALL COMPARISONS

In our SANN$_P$, *Server* is assumed to be semi-honest. We aim to prevent a semi-honest cloud from acquiring either the

[1] How to assure the security in transmission channel is beyond the scope of this paper, we assume the key and map index can be transmitted in a secure way.

TABLE 2. Overview of related researches.

Scheme	Information leakage	Dimensionality	Security assumption	Index Storage
$\mathrm{SANN_T}$ [16], [17]	Code book	High	NULL	Large
DCQR [18]	Ordering	2	NULL	Large
RS-SANN [14]	Ordering	Unlimited	IND-OCPA	Large
Zhu's [15]	Encrypted matrix	Unlimited	Inevitable matrix	Large
$\mathrm{SANN_P}$	NULL	Unlimited	IND-CPA	Tiny

data or the query. A summary of related works is elaborated in Table 2. The leakage of encrypted matrix in [15] leads to fragile security because the secret key can be derived by linear resolution. Hence, Zhu *et al.* [15] declared that the proposed scheme can only provide data privacy and resolve the key sharing problem which is outside this paper. Other schemes also leaks information of encrypted data. RS-SANN in [14] adopts a weak security assumption (i.e., IND-OCPA) to construct a concrete SANN scheme. However, the ordering of encrypted compound LSH values are leaked, which may lead to potential security risk. Oppositely, we avoid information leakage in our proposed schemes and prove the security based on IND-CPA, which is indeed essential for practical consideration.

The rest of the paper is organized as follows. In Section II, formulations and definitions of $\mathrm{SANN_P}$ are stated in detail. Greedy partition method is presented in Section III. Afterwards, we propose $\mathrm{SANN_P}$ model in Section IV, and analyze $\mathrm{SANN_P}$ in the viewpoints of security, accuracy and complexity. In Section V, we design an extended version $\mathrm{mSANN_P}$ which promotes the result accuracy. The experiments are conducted in Section VI. Section VII reviews the related work. Finally, we conclude this work in Section VIII.

II. PROBLEM STATEMENT

A formal definition of $\mathrm{SANN_P}$ is presented at the beginning. Besides, the challenges of designing $\mathrm{SANN_P}$ for high-dimensional data are described in detail. Afterwards, security requirements for $\mathrm{SANN_P}$ are formally defined.

A. PROBLEM FORMULATION

Definition 1 (Partition-Based SANN): An $\mathrm{SANN_P}$ includes the following four algorithms:

1) **Preparation**: Given the whole dataset D and a secret key sk, D is partitioned into several subsets $SD_1, SD_2, \ldots, SD_{nop}$, each of which is associated with a unique random tag $t_i \in \mathbb{Z}$. A map index I is generated to store the map between subsets and their associated tags. The encrypted outsourced subset is $E(\widehat{SD_i}) = \langle E(SD_i), t_i \rangle$ where $E(SD_i)$ is the encrypted form of SD_i under sk.
2) **QueryTag**: Given a query $\boldsymbol{q}$ and the map index I, generate a cloak query T.
3) **Search@Server**: Given a cloak query T, answer the query by returning the set $E(\Phi) = \{E(\widehat{SD_i}) | t_i \in T\}$.
4) **Search@Client**: Given $E(\Phi)$, $\boldsymbol{q}$ and a secret key sk, decrypt $E(\Phi)$ into Φ and perform kNN search over it.

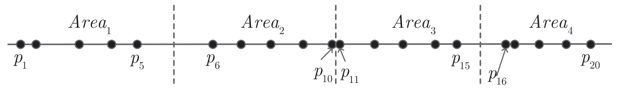


FIGURE 2. Basic partition method.

In $\mathrm{SANN_P}$, only encrypted subsets and corresponding tags are subcontracted to *Server*. Notice that $\boldsymbol{q}$ cannot be directly derived by T and nobody can decrypt the dataset without *Owner*'s secret key, so confidentiality is both guaranteed for $\boldsymbol{q}$ and D.

B. CHALLENGES

The first key problem of $\mathrm{SANN_P}$ is to partition original dataset into several subsets in **Preparation**.

For NN search in one-dimensional data, the partition method was proposed in [1] (as shown in Fig. 2). The partition line is the perpendicular bisector between two adjacent critical vectors. However, it is infeasible for high-dimensional data because of the absence of linear order relation between data. Another natural way is to partition original data by clustering method [19]. However, there are three main limitations for the high-dimensional data: 1) it is intractable to cluster massive data [20]; 2) simply clustering cannot guarantee that each cluster has the similar size, which may lead to data leakage; 3) there is no guarantee for search result accuracy. Hence, clustering cannot be applied in our problem.

For avoiding the leakage of query itself, original query cannot be transmitted from *Client* to *Server* in a public channel. Hence, a cloak query should be transmitted instead. For a semi-honest *Server*, it can only carry out **Search@Server** with the cloak query.

Based on above considerations, $\mathrm{SANN_P}$ must resolve the following challenges:

1) How to partition the whole dataset into subsets and establish a map index? (Section III-C)
2) How to generate cloak query according to the map index? (Section IV-A)

C. SECURITY REQUIREMENT

Aside from the aforementioned challenges, as an SANN scheme, $\mathrm{SANN_P}$ needs to be secure enough. In this part, we emphasize that the proposed partition method has to satisfy security in the sense of IND-CPA. Based on that, $\mathrm{SANN_P}$ is semantically secure enough to apply in practice. We assume that *Server* is semi-honest, and *Owner* and *Client* are fully trusted. If IND-CPA is satisfied, whenever answering a query $\boldsymbol{q}$, SANN$_P$ guarantees the followings: 1) *Server* cannot disclose the original dataset D, and 2) *Server* cannot learn anything about the query $\boldsymbol{q}$.

According to the requirements of IND-CPA, there exists an adversary who can obtain encrypted subset $E(SD_i)$ for any SD_i. In this strengthened assumption, the adversary aims to distinguish $E(SD_i)$ from $E(SD_j)$ through cryptographic deduction where $SD_i \neq SD_j$. In other words, when an adversary challenges on SD_i and SD_j for their encrypted form $E(SD_i)$ and $E(SD_j)$, challenger responds to it with SD_b where coin $b \in \{i, j\}$. The adversary aims to guess $b' = i$ or $b' = j$. The adversary's advantage $Adv_{\mathcal{A}}$ of winning the game is defined as $Pr[b = b'] - 1/2$.

Definition 2 (IND-CPA Secure): In polynomial time, if there is no adversary $\mathcal{A}$ breaking the partition method with a non-negligible function $Adv_{\mathcal{A}}$, the partition method is IND-CPA secure.

In the following sections, we propose an SANN$_P$ model with a greedy partition method that strictly follows the requirement of Definition 2. Moreover, we theoretically prove that our greedy partition method satisfies **IND-CPA** secure in Appendix I.

III. PARTITION

In this section, we address the first challenge we described in Section II-B. In offline preparation of SANN$_P$, a map index I and all encrypted subsets are produced. To achieve that, we propose an optimized linear order, based on *locality sensitive hashing* (LSH) which has theoretical guarantee of the ANN result accuracy. Based on that, a greedy partition method is proposed to partition the whole dataset.

A. VARIANT LOCALITY SENSITIVE HASHING

Before presenting our partition method, we briefly introduce an variant version of LSH. Indeed, the original LSH is a fundamental concept that is widely adopted in state-of-the-art linear order mapping and ANN schemes for high-dimensional data [9]. Formally, the variant version of LSH $h : \mathbb{R}^d \rightarrow \mathbb{Z}_{2^\lambda}$ maps a d-dimensional vector into a single integer as also introduced in [14].

$$h(\boldsymbol{v}) = \lfloor \frac{y(\boldsymbol{v}) + r}{\varpi} \rfloor. \quad (1)$$

In Equation 1, $\boldsymbol{v}$ is the original d-dimensional vector. The inner product is $y(\boldsymbol{v}) = \boldsymbol{\alpha} \cdot \boldsymbol{v}$, in which $\boldsymbol{\alpha}$ is a vector randomly drawn from a 2-stable Gaussian (normal) distribution defined by the density function $f(x) = (1/\sqrt{2\pi})e^{-x^2/2}$. Note that, we define $\varpi = (\max_{\boldsymbol{p}\in D}\{y(\boldsymbol{p})\} - \min_{\boldsymbol{p}\in D}\{y(\boldsymbol{p})\}) \cdot 2^{-\lambda}$ and r is drawn randomly from $[0, \varpi]$.

Definition 3 ((R_1,R_2,P_1,P_2)-Sensitive LSH [9]): Given $R_2 > R_1$, an LSH function $h : \mathbb{R}^d \rightarrow \mathbb{N}$ is (R_1, R_2, P_1, P_2)-sensitive if for any $\boldsymbol{q}, \boldsymbol{v} \in \mathbb{R}^d$:

- $Pr[h(\boldsymbol{v}) = h(\boldsymbol{q})] \geq P_1$ for $\| \boldsymbol{q} - \boldsymbol{v} \| \leq R_1$.
- $Pr[h(\boldsymbol{v}) = h(\boldsymbol{q})] \leq P_2$ for $\| \boldsymbol{q} - \boldsymbol{v} \| > R_2 = cR_1$ and $c > 1$.

Algorithm 1 LSH-Based Coding Algorithm

Require: A vector $\boldsymbol{p}$, a G function $G_m(\boldsymbol{v})$ and the code length λ

Ensure: The code C of $\boldsymbol{p}$

```
1:  function Code($\boldsymbol{p}$, $G_m$, $\lambda$)
2:      Compute the $G$ value $H_{\boldsymbol{p}} = (h_1, h_2, \ldots, h_m)$;
3:      $C = \emptyset$;
4:      for each $i = 1$ to $\lambda$ do
5:          for each $j = 1$ to $m$ do
6:              The $i$−th bit of $h_j$ is added into $C$;
7:          end for
8:      end for
9:      return $C$;
10: end function
```

In the above definition, the values of P_1 and P_2 are calculated as follows:

$$P_1 = \int_0^{\varpi} f(t)(1 - \frac{t}{\varpi})dt, \quad P_2 = \int_0^{\varpi} \frac{1}{c} f(\frac{t}{c})(1 - \frac{t}{\varpi})dt.$$

In practice, a group of LSH functions (as Definition 4) are always adopted together [21], [22].

Definition 4 (G Function): A G function G_m consists of m different (R_1,R_2,P_1,P_2)-sensitive LSH functions conforming to Equation 1 and Definition 3, denoted as $G_m(\boldsymbol{v}) = (h_1(\boldsymbol{v}), h_2(\boldsymbol{v}), \ldots, h_m(\boldsymbol{v}))$.

For convenience, the G value of vector $\boldsymbol{v}$ is represented as $H_{\boldsymbol{v}} = (h_{\boldsymbol{v}1}, h_{\boldsymbol{v}2}, \ldots, h_{\boldsymbol{v}m})$.

B. LINEAR ORDER

To construct a linear order, $H_{\boldsymbol{q}}$ is mapped into a single binary code $C_{\boldsymbol{q}}$ following z-order style [22]. The first bits of $h_{\boldsymbol{v}i}$ are placed at the beginning of $C_{\boldsymbol{v}}$ when $1 \leq i \leq m$, and then the second bits are placed at the next positions. The process repeats until the last bits are placed. For example, assuming $m = 2$, $\lambda = 3$ and $H_1 = (h_{11}, h_{12}) = (0\underline{0}1, 0\underline{1}1)$, the code of $\boldsymbol{v_1}$ is $C_{\boldsymbol{v}} = (00\underline{01}11)$. Such method is called LSH-based coding as shown formally in Algorithm 1.

Obviously, given $\mathbb{C} = \{C_{\boldsymbol{v}} | \boldsymbol{v} \in D\}$, $\langle \mathbb{C}, \leq \rangle$ is a linear order set where $\leq$ is bitwise comparison between two codes. In this way, we break the limitation of the linear order defined in [21]. Assume that $\boldsymbol{p_1} = (2.5, 2)$, $\boldsymbol{p_2} = (3, 2.5)$ and $\boldsymbol{p_3} = (2, 5)$ are three original data in D, and two LSH functions in G function with $\boldsymbol{\alpha_1} = (7, 1), r_1 = 0, \omega_1 = 11.5$ and $\boldsymbol{\alpha_2} = (2, 6), r_2 = 0, \omega_2 = 5.5$ are defined as:

$$h_1(\boldsymbol{v}) = \lfloor \frac{\boldsymbol{\alpha_1} \cdot \boldsymbol{v} + r_1}{\omega_1} \rfloor, \quad h_2(\boldsymbol{v}) = \lfloor \frac{\boldsymbol{\alpha_2} \cdot \boldsymbol{v} + r_2}{\omega_2} \rfloor.$$

It is obvious that $\boldsymbol{p_2}$ is closer to $\boldsymbol{p_1}$ than $\boldsymbol{p_3}$ in Euclidean space. According to Definition 4, $H_1 = (1, 3)$, $H_2 = (2, 3)$ and $H_3 = (1, 6)$ are G values of $\boldsymbol{p_1}, \boldsymbol{p_2}$ and $\boldsymbol{p_3}$ respectively. Obviously, according to $\leq_{\mathbb{H}}$ [21], H_3 is closer to H_1 rather than H_2. Therefore, pairwise distance between LSH keys is not well preserved under $\leq_{\mathbb{H}}$. However, in our linear order, H_1, H_2 and H_3 can be further mapped into $C_1 = (000111)$, $C_2 = (001101)$ and $C_3 = (010110)$. Notably, C_2 is closer to

C_1 in the linear order rather than C_3. It is consistent to our observations in original data.

So far, a linear order in $\mathbb{C}$ is built. Based on that, two vectors are close enough to each other if their codes are equal or close enough.

C. GREEDY PARTITION METHOD

A natural way is to partition the whole dataset according to codes of data like in Fig. 2. However, it is inadequate for partitioning by the codes since there are still several limitations:

1) The perpendicular bisector between two adjacent codes is incomputable and meaningless.
2) When partition dataset into subsets with equal-number codes, it leads to data explosion.

To address the limitations mentioned above, we use the exact codes of the critical vectors as the upper and lower bounds. Several principles during the greedy partition method are as follows:

P1 All data sharing the same code must be gathered into the identical subset and the size of subset should be minimized.

P2 Size of each subset should be uniform without leading to data explosion.

P3 Upper and lower bounds for each subset are independent and unique.

P1 ensures less communication cost and shorter average response time, because all vectors sharing the query's code are the query's ANNs with equal probability. To achieve **P2**, we gather equal-number vectors into a single subset instead of gathering remaining vectors with the same code into different subset. Meanwhile, according to **P3**, the map index I is derived from the vectors which is only included in current subset. Note that, **P1** is for accuracy guarantee since most probable nearest neighbors are in the same partition. **P2** provides security guarantee in the basis that encrypted partitions are of unique size, and hence are indistinguishable from each other. **P3** ensures the feasibility of map index by avoiding a query falling in multiple subsets.

The greedy partition method is generally illustrated in Fig. 3 and formally described by Algorithm 2. To facilitate the following discussion, we define $\boldsymbol{p_i} \sim \boldsymbol{p_j}$ iff $C_{\boldsymbol{p_i}} = C_{\boldsymbol{p_j}}$. Specially, we denote $[C_{\boldsymbol{p_i}}]_\sim = \{\boldsymbol{p_j} | \boldsymbol{p_i} \sim \boldsymbol{p_j}\}$.

The main partition algorithm is formally described in Algorithm 2. Dataset $D[]$ is the input. $D[]$ is sorted at the beginning (Line 2). I is initialized to be empty and partition width w is greedily computed (Line 3). The partition starts from the first element in dataset (Line 4). Then, a loop is carried out to continuously partition datasets (Lines 5-14). In the loop, w continuous elements are first added into a subset (Line 7), and then only the equal-code sets that are completely included in the subset are indexed (Lines 8-10). Following that, the index item I_i are generated by embedding another random tag t_i (Lines 6, 11-13). Finally, the array of subsets $SD[]$ and map index I are returned (Line 15).

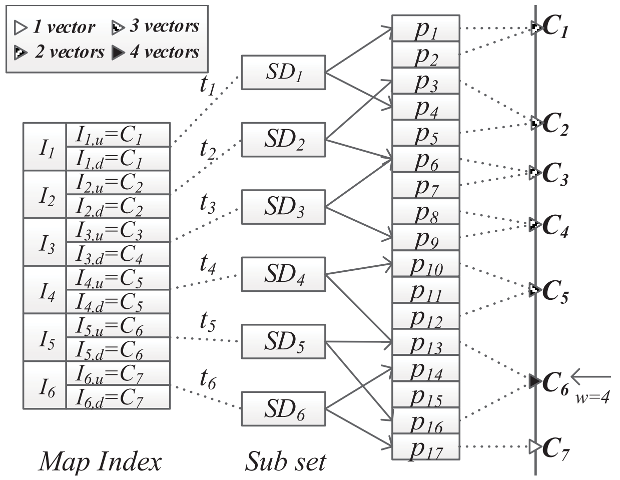

▷ 1 vector
▷ 2 vectors
▶ 3 vectors
▶ 4 vectors
I_1 $I_{1,u}=C_1$ $I_{1,d}=C_1$
I_2 $I_{2,u}=C_2$ $I_{2,d}=C_2$
I_3 $I_{3,u}=C_3$ $I_{3,d}=C_4$
I_4 $I_{4,u}=C_5$ $I_{4,d}=C_5$
I_5 $I_{5,u}=C_6$ $I_{5,d}=C_6$
I_6 $I_{6,u}=C_7$ $I_{6,d}=C_7$
t_1 t_2 t_3 t_4 t_5 t_6
SD_1 SD_2 SD_3 SD_4 SD_5 SD_6
p_1 … p_{17}
C_1 C_2 C_3 C_4 C_5 C_6 C_7
w=4
Map Index
Sub set

FIGURE 3. Greedy Partition Method.

Algorithm 2 Greedy Partition Algorithm

```
Require: Dataset $D[] = \{\boldsymbol{p_1}, \boldsymbol{p_2} \ldots, \boldsymbol{p_n}\}$.
Ensure: Union of subsets $SD[]$, Map Index $I[]$.
1:  function GreedyPartition($D[]$)
2:      Sort $D[]$ by their codes according to $< \mathbb{C}, \leq >$;
3:      Let $I \leftarrow \emptyset$, $w \leftarrow \max_{1 \leq k \leq n} \#([C_{\boldsymbol{p_k}}]_\sim)$;
4:      $i \leftarrow 1, j \leftarrow 1$; /* $i$ and $j$ are the subscripts of subsets and elements, respectively. */
5:      repeat
6:          $I_{i,u} \leftarrow C_j$;
7:          $SD_i \leftarrow \bigcup_{k=j}^{j+w-1} \{\boldsymbol{p_k}\}$; / *Add continuous $w$ elements into $SD_i$. */
8:          if $[C_{j+w-1}]_\sim \nsubseteq SD_i$ then /* The equal-code set of element $p_j$ is not completely included in $SD_i$. */
9:              $j \leftarrow \max_{j \leq k \leq j+w-1} \{[C_k]_\sim \subseteq SD_i\}$;
10:         end if
11:         $I_{i,d} \leftarrow C_j, j \leftarrow j + 1$;
12:         Generate $t_i$ by a PRG, and $I_i \leftarrow \langle I_{i,u}, I_{i,d}, t_i \rangle$;
13:         $I = I \bigcup I_i$;
14:     until $j > n$;
15:     return $SD[]$, $I[]$;
16: end function
```

An helpful example of such partition method is also described in detail in Appendix II.

Consequently, by applying **P3**, the upper and lower bounds for each subset are independent. Hence, the subsets may be overlapped slightly. However, it hardly exert any negative impact on the query efficiency. It's mainly because the number of overlapped vectors is small enough to be ignored, which will be justified by experiments in Section VI-D.

IV. SANN$_\text{P}$

In this section, we'll discuss how an arbitrary query is performed under SANN$_\text{P}$ framework, in order to address the second challenge proposed in the end of Section II-B.

Afterwards, the security, accuracy and complexity of SANN$_P$ are analyzed.

A. CONSTRUCTION OF SANN$_P$

1) PREPARATION

Owner carries out **Preparation** offline before subcontracting dataset to *Server*. First of all, security parameter SP should be set, which includes a random secret key sk, a G_m function comprising m modified LSH functions based on Equation 1, and an appropriate λ for the single code length. In this way, a secret parameter is formed as $SP = \langle sk, G_m, \lambda \rangle$.

At the beginning, all vectors in D are encoded using Algorithm 1, and sorted in ascending order. We then divide D into nop subsets according to Algorithm 2, and nop can be derived from $SD[]$. With the map index I of $SD[]$, each subset SD_i is associated with a random tag $t_i \in \mathbb{Z}$ by a *pseudo random generator* (PRG). Hence, $\widehat{D} = \cup_{i=1}^{nop} \langle SD_i, t_i \rangle$ and $I = \cup_{i=1}^{nop} \langle I_{i,u}, I_{i,d}, t_i \rangle$.

Since database cannot be stored securely as clear text, *Owner* chooses an IND-CPA secure encryption algorithm E. Afterwards, *Owner* encrypts SD_i using E with the private key sk, and obtains $E(\widehat{SD_i}) = \langle E(SD_i), t_i \rangle$. Finally, *Owner* subcontracts $E(\widehat{D}) = \cup_{i=1}^{nop} E(\widehat{SD_i})$ to *Server*.

Whenever *Client* issues a query $\boldsymbol{q}$, the following procedures are performed sequentially.

2) QueryTag

Client requests SP and I from *Owner*, and encodes $\boldsymbol{q}$ according the G_m function selected in **Preparation**. Then *Tag Query Algorithm* (Algorithm 3) is executed to obtain a cloak query T, which comprises of a set of tags t_i. Each tag corresponds to the subset in which the ANN result may fall.

In Algorithm 3, T and C_I are initialized to be empty (Line 1). C_I is constructed by sequentially connecting all upper and lower bounds (Line 2-4). Then, C_q is computed (Line 5). Afterwards, we locate subsets in which the ANN result of $\boldsymbol{q}$ falls and add the corresponding tags to T (Line 6-11). Finally, T is generated (Line 12). T is then sent to *Server* as the cloak query.

3) Search@Server

One of the greatest strengths in SANN$_P$ is that it avoids the complicated and expensive operations in *Server*, such as similarity (distance) computation in [17], retrieval of a large set of tags in [18], comparisons of encrypted codes [14], etc. *Server* holds encrypted datasets $E(\widehat{D}) = \cup_{i=1}^{nop} \langle E(SD_i), t_i \rangle$. When resolving a cloak query T, *Server* returns the result $E(\Phi) = \cup_{t_i \in T} E(SD_i)$.

In practice, *Server* stores $E(\widehat{SD_i})$ with file name t_i. Thus, when *Server* receives a cloak query T, he just sends the content of file t_i back to *Client* without any extra operation. Even if such mechanism is not adopted, in the worst case, $E(\widehat{D})$ can be managed using a low efficiency linear list. The cost to locate the returning subsets $E(\Phi)$ is $O(nop)$ where $nop \ll n_D$.

Algorithm 3 Tag Query Algorithm

```
Require: Index $I$, a query $\boldsymbol{q}$
Ensure: A cloak query $T$
1: $T = \emptyset, C_I = \emptyset$;
2: for each $i = 1$ to $nop$ do
3:     $C_I = C_I \cup I_{i,u} \cup I_{i,d}$;
4: end for
5: Compute $C_q$ of $\boldsymbol{q}$ according Algorithm 1;
6: Find the minimum $i$ such that $C_{i+1} \geq C_q \geq C_i$;
7: if $i$ is odd then /* fall in subset */
8:     $T = T \cup t_{(i+1)/2}$;
9: else[$i$ is even] /* fall in critical area */
10:    $T = T \cup t_{i/2} \cup t_{i/2+1}$;
11: end if
12: return $T$;
```

4) Search@Client

Once the result $E(\Phi)$ is returned to *Client*. *Client* decrypts each $E(SD_i) \subseteq E(\Phi)$ with secret key sk to get Φ. Then, sequential scanning over Φ is the kNN result.

Notice that *Server* does not know D, SD_i, or Φ which are all encrypted. Thus, *Owner*'s data are confidential as long as the secret key is secure. $\boldsymbol{q}$, C_q and I are never leaked to *Server*. In this manner, *Server* will learn nothing about the query or the candidate set. More theoretical studies over this scheme are given in the following.

B. SECURITY ANALYSIS

The security of SANN$_P$ relies on IND-CPA secure encryption algorithm.

Theorem 1: Given that the encryption algorithm adopted is IND-CPA secure, the greedy partition method is IND-CPA secure with respect to a single query. (The proof is in Appendix I-A)

The security guarantee presented above only considers an adversary for a single query with power to obtain a large number of plain-cipher pairs. This assumption is equivalent to that in [23], in which access pattern and search pattern are divulged inherently. For multiple queries, the distribution of *Client*'s queries (i.e., search pattern and access pattern) can be obtained by adversary, and adversary can rebuild the relation between the tags and ranges of each encrypted subset. In order to reduce such security risk, the partition width should be large enough so that the distribution is meaningless. Indeed, there is no outstanding work to expound the meaning of distribution over high-dimensional data. Even so, through the proof of Theorem 1, the partition width should be large enough to guarantee the security. When the size is equal to 1 or other small values, the proposed SANN$_P$ becomes weak and thus inefficient and impractical. Hence, such mechanism should avoid a small size of subset.

Indeed, access pattern is meaningless for a single query but very meaningful for multiple queries. Along with multiple queries, it leads to inferring information by exploiting the correlation between different queries. SANN$_P$ inherently does not achieve protection of access pattern as be proved. Generally, if access pattern is leaked, it will lead to huge private information leakage. For example, in GIS system, the user's location commonly contains private information and is regarded as the access pattern to submit. If the history of location is leaked, we can facilely infer that the user is in company, since company is the place user visited most frequently. Another applications (e.g., biometric information, multimedia feature, etc.) face the analogical problems. A general method for providing security of multiple queries is to apply *private information retrieval* (PIR) [24] protocol, or *oblivious transfer* (OT) [25] protocol. How to achieve protection of access pattern is beyond this paper. Hence, we only explain the approach of adopting PIR in our proposed SANN$_P$ concisely.

In general, PIR protocols allow *Client* fetching an encrypted subset while preventing *Server* from knowing it under limited computing resource. *Client* issues a specific set of fake queries and the true query. Then, *Server* returns all encrypted subsets corresponding to the specific set. *Client* only picks up the encrypted partition corresponding to the true query. Throughout the process, *Server* will not know which subset *Client* picks up. However, due to PIR protocol, the security guarantee would come at cost, and the communication efficiency will be too bad in practice. How to design an SANN or SNN that both protects access pattern and provides high query efficiency is still an open problem. Thus, the tradeoff between security and efficiency should be seriously considered according to the specific applications.

C. ACCURACY ANALYSIS

Similar to LSH-based ANN, the proposed SANN$_P$ also exhibits satisfactory accuracy guarantee which is derived by Definition 3. In this part, we present theoretical analysis over the accuracy of SANN$_P$.

Theorem 2: Given $\boldsymbol{p}, \boldsymbol{v} \in \mathbb{R}^d$, and $\boldsymbol{p}$ is the vector close to $\boldsymbol{v}$. Let $R_\beta = cR_\alpha > R_\alpha = \parallel \boldsymbol{p} - \boldsymbol{v} \parallel$.

For each vector $\boldsymbol{q} \in \mathbb{R}^d$ such that $\parallel \boldsymbol{q} - \boldsymbol{v} \parallel \leq R_\alpha$:

$$Pr[C_v = C_q] \geq P_\alpha \geq (1 - \frac{2}{\sqrt{2\pi}\varpi_{min}})^m, \tag{2}$$

where $\varpi_{min} = \min_{1 \leq i \leq m} \{\varpi_i\}$.

For a proper constant $B \geq 0$ and each vector $\boldsymbol{q} \in \mathbb{R}^d$ such that $\parallel \boldsymbol{q} - \boldsymbol{v} \parallel > R_\beta$:

$$Pr[C_v = C_q] \leq P_\beta < (1 - Bc/\varpi_{max})^m, \tag{3}$$

where $\varpi_{max} = \max_{1 \leq i \leq m} \{\varpi_i\}$.

The proof of theorem is referred to [14]. According to Equation 3, given two different vectors $\boldsymbol{q}, \boldsymbol{v} \in \mathbb{R}^d$, if $\parallel \boldsymbol{q} - \boldsymbol{v} \parallel > R_\beta$ then $C_v = C_q$ never happens if m is large enough. That means, if $C_v = C_q$ occurs, the probability of $\parallel \boldsymbol{q} - \boldsymbol{v} \parallel \leq R_\beta$ occurring is high.

In SANN$_P$, the ultimate goal for *Server* is to return the set Φ with the same code of $\boldsymbol{q}$. Theorem 2 assures that the close vectors of $\boldsymbol{q}$ are in Φ with a great probability.

D. COMPLEXITY ANALYSIS

In this part, complexity analysis is conducted for *Owner*, *Server* and *Client* respectively. Before D is outsourced to *Server*, it is preprocessed by *Owner*. Firstly, D is partitioned into *nop* subsets. Let $n_D = \#(D)$. In the partition phase, quick sorting algorithm needs $O(n_D \log n_D)$ swaps of codes. Hence, D needs $O(d \cdot w \cdot nop)$ space for storage. Each t_i is in fact a single integer and the space needed is $O(1)$. Assume a single encryption of vector needs $O(dt_E)$ time and $O(ds_E)$ space, the time to encrypt $\widehat{D}$ is $O(d \cdot w \cdot nop \cdot dt_E)$. Thus, *Server* needs $O(s_E(d+1) \cdot d \cdot w \cdot nop)$ space for storing $E(\widehat{D})$ and the whole time cost of preprocessing is $O(d \cdot w \cdot nop \cdot (\log(d \cdot w \cdot nop) + dt_E))$.

Then, given a query $\boldsymbol{q}$, *Client* performs **QueryTag** to get T. Let $n_T = \#(T)$. The time cost is $O(nop \log nop)$ and the communication cost is $O(n_T)$ where $n_T \leq 2$ according to Algorithm 3. When receiving a cloak query T, the search time for *Server* is $O(n_T)$ at worst. The communication cost of returning result is $O(n_T \cdot d \cdot w \cdot nop)$. Hence, the total time cost is $O(n_T)$ and communication cost is $O(n_T)$.

Finally, *Client* receives the encrypted candidate set $E(\Phi)$, then performs *Dec* step, and get the candidate set Φ. Let $n_\Phi = \#(\Phi)$ which is $2w$ at most. Then, kNN is run to find the close vectors to query vector $\boldsymbol{q}$ in Φ. The complexity of searching reduces to $O(n_\Phi)$ from $O(n_D)$. n_D is always 1-3 magnitudes larger than n_Φ under appropriate settings of m and λ.

V. MULTI-DIVISION SANN$_P$

For a single G_m function, according to Theorem 2, there is a limitation: the smaller the subset is, the lower the result accuracy is.

In fact, for two vectors who are very close in D, there is a non-negligible probability that they do not share the same code under a single G_m function. In order to overcome the limitation and improve the accuracy, we introduce a multi-division strategy into SANN$_P$ and propose mSANN$_P$.

A. CONSTRUCTION OF MULTI-DIVISION SANN$_P$

1) MULTI-PREPARATION

Owner carries out **Preparation** offline ℓ times before subcontracting dataset to *Server*, where ℓ is the number of divisions adopted. The secret parameter is $mSP = \cup_{j=1}^{\ell} SP^j$.

For SP^j, the encrypted outsourced dataset is $E(\widehat{D^j}) = \cup_{i=1}^{nop} E(\widehat{SD_i^j})$ where $E(\widehat{SD_i^j}) = \langle E(SD_i^j), t_i^j \rangle$, and the map index is $I^j = \cup_{i=1}^{nop^j} \langle I_{i0}^j, I_{i1}^j, t_i^j \rangle$. Hence, in mSANN$_P$, the encrypted outsourced dataset is $mE(\widehat{D}) = \cup_{j=1}^{\ell} E(\widehat{D^j})$ and the map index is denoted as $mI = \cup_{j=1}^{\ell} I^j$. That is, $mE(\widehat{D})$ is outsourced to *Server*, and mI and mSP are sent securely to *Client*.

2) MULTI-QueryTag

When *Client* issues a query $\boldsymbol{q}$, he requests mSP and mI from *Owner*. Then, *Client* encodes $\boldsymbol{q}$ according to the

G_m^j function selected in **Multi-Preparation** where $j \in \{1, 2, \ldots, \ell\}$. *Client* executes Algorithm 3 to obtain a single cloak query T^j for $\widehat{D^j}$. Hence, in mSANN$_\mathrm{P}$, the final cloak query is $mT = \cup_{j=1}^{\ell} T^j$.

3) MULTI-Search@Server

Server holds encrypted dataset $E(\widehat{D^j}) = \cup_{i=1}^{nop} E(\widehat{SD_i^j})$. Whenever receiving a query request, *Server* performs **Search@Server** ℓ times for each $E(\widehat{D^j})$ with T^j, and returns $E(m\Phi) = \cup_{j=1}^{\ell} E(\Phi^j)$ to *Client* as the response, where $E(\Phi^j) = \cup_{t_i^j \in T^j} E(SD_i^j)$.

4) MULTI-Search@Client

Before carrying out kNN over $m\Phi$, an authorized *Client* gains access to secret key sk^j from *Owner*. *Client* decrypts all the vectors in $E(m\Phi)$ and removes all repeat ones. Then, sequential scan is performed to find kNNs in the remaining vectors.

B. ANALYSIS OF MULTI-DIVISION SANN$_\mathrm{P}$

It is obvious that there is no additional secure concerns in mSANN$_\mathrm{P}$ other than SANN$_\mathrm{P}$. Thus, mSANN$_\mathrm{P}$ is as secure as SANN$_\mathrm{P}$ with respect to a single query. Analysis of accuracy and complexity follows on.

Theorem 3: With respect to a single query, the greedy partition method is still IND-CPA secure as long as the encryption algorithm adopted is IND-CPA secure. (The proof is in Appendix I-B)

Server carries out **Preparation** offline ℓ times independently. As a result, the probability that two close vectors in $\mathbb{R}^d$ collide in $\mathbb{C}$ is at least

$$P_\alpha = 1 - (1 - P_{\alpha_1})(1 - P_{\alpha_1}).$$

where P_{α_1} and P_{α_2} are the probabilities that they collide in G_m^1 and G_m^2, respectively. In fact, the result accuracy increases with the increase of ℓ. Experimental results justify that in next section.

Similar to accuracy analysis, time and space complexity of mSANN$_\mathrm{P}$ is ℓ times that of SANN$_\mathrm{P}$. In compensation for the limited sacrifice in time and space costs, accuracy reaches a much higher level compared to other solutions.

VI. EXPERIMENTAL RESULTS

We conduct the experiments in C++ on a Workstation with Intel Xeon CPU E5-2680 v2 @ 2.80 GHz and 256 GB memory running Linux. We use MIRACL[2] library to implement all the standard encryption schemes.

We perform our experiments on two real datasets (*Hong-Kong*[3] and *SIFT1M*)[4] and two synthetic datasets (*Gaussian64*, *Gaussian256*) with different dimensionalities. *Hong-Kong* consists of 1,384,420 2-dimensional POIs (Point of Interest) in Hong Kong, China. *SIFT1M* consists of 1,000,000 128-dimensional sift descriptors. We generate 1,000,000 64-/256-dimensional vectors following Gaussian distribution for *Gaussian64* and *Gaussian256*, respectively. Moreover, 10,000 vectors in each dataset are generated for testing.

All four datasets are normalized before being processed. Let a vector in dataset be $\boldsymbol{p_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$. The normalized parameter is as follows:

$$\mathcal{Z} = \max_{i\in[1,d]}\{\max_{j\in[1,\#(D)]}\{x_{ji}\} - \min_{j\in[1,\#(D)]}\{x_{ji}\}\}$$

Then, let $\boldsymbol{p_i} = (\boldsymbol{p_i}/\mathcal{Z})$.

Through all experiments, AES symmetric encryption is adopted as the secure encryption algorithm. It will guarantee the confidentiality of data against *Server*. The encryption process is independent of our partition method. Hence, a series of other symmetric and asymmetric encryption methods can also be adopted, such as DES, IDEA, RC4, RSA, ECC, IBE, etc. Additionally, *Client* does not release the cache of ever decrypted elements to promote searching efficiency.

To justify the empirical results, we compare accuracy and efficiency of our work (i.e., SANN$_\mathrm{P}$ and mSANN$_\mathrm{P}$) with three baseline state-of-the-art SANN approaches. Among these approaches, DCQR [18] was proposed for 2-dimensional data, and SANN$_\mathrm{T}$ [17] was proposed for multi-dimensional data. The security they considered is weak for aforementioned application scenarios in Section I, which is enhanced in our proposals. The only scheme, which reaches the same secure level as SANN$_\mathrm{p}$ and mSANN$_\mathrm{p}$, is SNN proposed in [1]. However, as we will mention in related work, SNN cannot resolve SANN in 2-dimensional data, and thus it is not considered as a baseline approach. Here, RS-SANN_MI in [14] is introduced because of the excellent performance on *ratio*. All the parameters in the above three schemes are optimized according to their instructions.

In the following, we introduce the measures for evaluating the accuracy and efficiency of experiments. Afterwards, we discuss how the algorithm parameters are optimally selected. Following that, we show the comparisons between our schemes and the three state-of-the-art approaches, DCQR, SANN$_\mathrm{T}$ and RS-SANN_MI, to demonstrate the superiority of our schemes. Finally, we show the additional properties and advantages of our work that none of other approaches provide.

A. MEASURES

The following measures are employed to evaluate the performance for all schemes:

- *ratio* is used to evaluate the accuracy of returned results. For n_q test queries $\boldsymbol{q_i}$ with their k close vectors $\boldsymbol{o_{ij}}$, where $1 \leq i \leq n_q$ and $1 \leq j \leq k$, k vectors $\boldsymbol{o_{ij}^*}$ for $\boldsymbol{q_i}$ are returned as search results. *ratio* is defined as follows:

$$ratio = \frac{1}{n_q}\sum_{i=1}^{n_q}\left(\frac{1}{k}\sum_{j=1}^{k}\frac{\| \boldsymbol{o_{ij}^*} - \boldsymbol{q_i} \|}{\| \boldsymbol{o_{ij}} - \boldsymbol{q_i} \|}\right).$$

[2]http://www.certivox.com/miracl/
[3]http://metro.teczno.com/#hong-kong
[4]ftp://ftp.irisa.fr/local/texmex/corpus/sift.tar.gz

- *ART@Server* is used to evaluate *Average Response Time* (ART) of search scheme for *Server*. For n_q test queries $\boldsymbol{q_i}$ and $1 \leq i \leq n_q$, *Server* needs st_i to find Φ. *ART@Server* is defined as follows:

$$ART@Server = \frac{1}{n_q}\sum_{i=1}^{n_q} st_i.$$

- *ART@Clent* is used to evaluate ART of search scheme for *Client*. For n_q test queries $\boldsymbol{q_i}$ and $1 \leq i \leq n_q$, *Client* needs ct_i to do **Search@Client** step in Φ. *ART@Clent* is defined as follows:

$$ART@Client = \frac{1}{n_q}\sum_{i=1}^{n_q} ct_i.$$

- *ART* is used to evaluate the integral *ART* for both *Server* and *Client*. *ART* is defined as follows:

$$ART = ART@Client + ART@Server.$$

B. PARAMETER SELECTION

In our schemes, several key parameters need to be considered: the number m of LSH functions, the length λ of a single code and the times ℓ to repeat query processing which is only valid in the mSANNP model. All these parameters should be selected appropriately. In the following, we perform experiments to test the effects of these parameters, and find optimal values.

Note that λm is the total length of code. Let m be fixed. Larger λ excludes more true negative vectors in a single subset. Hence, with the increase of λ, partition size decreases and it leads to the decrease of accuracy. We vary λ for the four datasets and pick up those which give best tradeoffs between true negative and *ratio*. We fix $\lambda = 8$ for *Hong-Kong*, and $\lambda = 2$ for *Gaussian64*, *SIFT1M* and *Gaussian256*.

In all figures shown in this section, the tick marks in X-axis appear as the form "a/b/c/d", where a,b,c,d correspond to the value in *Hong-Kong*, *Gaussian64*, *SIFT1M* and *Gaussian256* dataset, respectively. For instance, in Fig. 4, the second x-tick "4/7/15/9" means that $m = 4$ for *Hong-Kong*, $m = 7$ for *Gaussian64*, $m = 15$ for *SIFT1M* and $m = 9$ for *Gaussian256*.

1) EFFECT OF m

We vary m to test how the performance is affected in Fig. 4. As expected, when m increases, SANNP results in a smaller partition size because of a smaller number of vectors sharing the same code. Therefore, *ratio* decreases while *ART* increases. The tradeoff between accuracy and *ART* must be weighed carefully for different applications.

In SANNP model, there is no effective strategy to improve ANN results' accuracy in only a single candidate set while there is plenty time to process more returned vectors. In order to provide the availability of ANN result, *ratio* is strictly restricted to a lower level and *ART* is relaxed to an acceptable

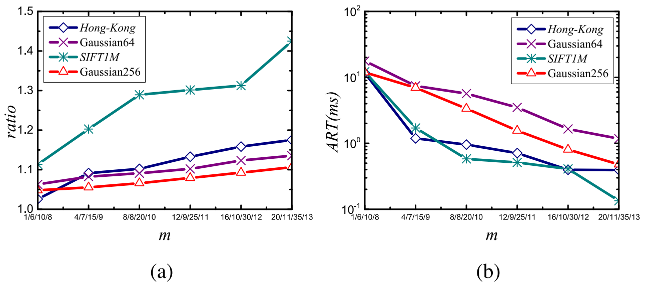


FIGURE 4. Effect of *m*. (a) *ratio*. (b) *ART*.

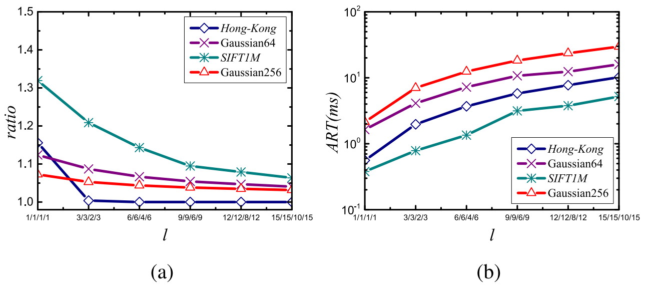


FIGURE 5. Effect of ℓ. (a) *ratio*. (b) *ART*.

value ($\approx$ 10 *ms*). It is worthwhile to improve accuracy by sacrificing time. We fix $m = 8$ for *Hong-Kong*, $m = 6$ for *Gaussian64*, $m = 15$ for *SIFT1M* and $m = 8$ for *Gaussian256* to achieve better *ratio*. Consequently, it will take a longer time to find the ANN result.

The multi-division query strategy in Section V significantly improves the accuracy. Hence, a lower accuracy for a single partition does not lead to a lower accuracy in mSANNP. Instead, *ART* increases linearly along with the increase of ℓ and it leads to an unacceptable response level. Here, we make a strict restriction ($\approx$ 1 *ms*) to *ART* and experimental results show that it is very worthwhile to promote efficiency by sacrificing accuracy (notably, the accuracy guarantee gets back through the multi-division strategy). We make a sound tradeoff by fixing $m = 12$ for *Hong-Kong*, $m = 11$ for *Gaussian64*, $m = 25$ for *SIFT1M* and $m = 11$ for *Gaussian256*. In consequence, though seemingly reduced, the accuracy will be eventually improved by multi-division strategy.

2) EFFECT OF ℓ

Note that ℓ is only valid in mSANNP model. Here is a tradeoff between accuracy and *ART* which is selected according to the customized requirement. By fixing m, it is easy to see that $\ell = 3$ for *Hong-Kong* dataset achieves higher accuracy and satisfactory *ART* in Fig. 5. We set the parameter in other datasets in the same way, $\ell = 11$ for *Gaussian*, $\ell = 9$ for *SIFT1M* and $\ell = 6$ for *Gaussian256*.

Up to now, we have selected the appropriate values of parameters. All the selected values are listed in Table 3, based on which the following experimental studies are conducted.

TABLE 3. Values of parameters.

Dataset		$SANN_P$	$mSANN_P$	Dataset		$SANN_P$	$mSANN_P$
	λ	8	8		λ	2	2
Hong-Kong	m	8	12	*SIFT1M*	m	15	25
	ℓ	-	3		ℓ	-	9
	λ	2	2		λ	2	2
Gaussian64	m	6	11	*Gaussian256*	m	8	11
	ℓ	-	11		ℓ	-	6

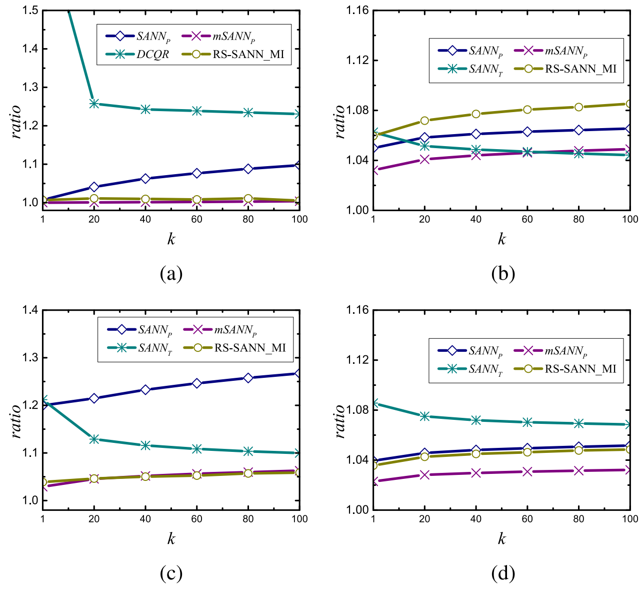


FIGURE 6. Comparisons on *ratio*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

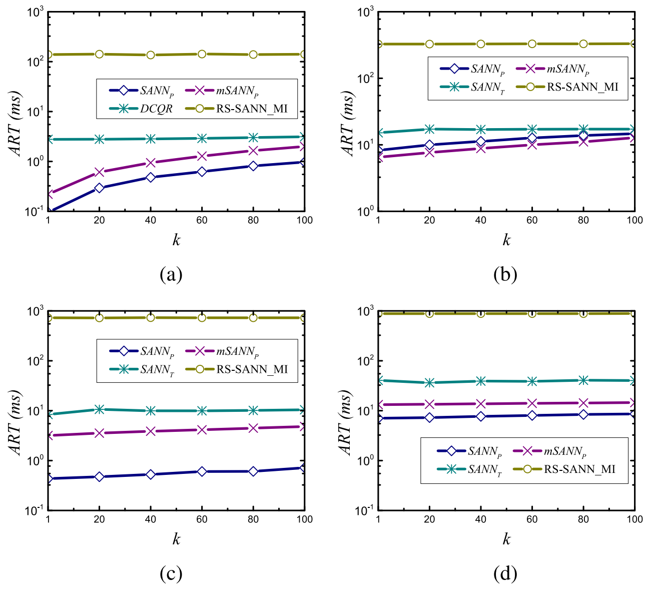


FIGURE 7. Comparisons on *ART*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

C. RESULTS COMPARISON

In this part, we compare DCQR, $SANN_P$, $mSANN_P$ and RS-SANN_MI with $SANN_T$ on the four datasets. In SkNN outsourcing applications, there are 5 widely-investigated measures, i.e., *ratio*, *ART*, *ART@Server*, *ART@Client* and storage space [1], [16], [22]. We adopt these measures in concentrated SANN outsourcing applications, and all processes in this section are done without encryption for fairness. Within the comparison, we vary $k \in [1, 100]$, the number of returned neighbors. The vectors in datasets are normalized into $[0, 1]$ according to the normalization function in the beginning of Section VI.

ratio. Fig. 6 shows the performance of *ratio* when we vary the values of k. Note that, $SANN_P$ and $mSANN_P$ outperform other three schemes. The *ratios* can be guaranteed because that $SANN_P$ and $mSANN_P$ scan several partitions which include more vectors than those of other three approaches. The *ratio* of $SANN_P$ and $mSANN_P$ decreases as k increases. Their differences between DCQR and $SANN_T$ decreases as k increases too. That indicates our models return the best results when k is small. Moreover, it is important in the motivated applications, where the number of returned nearest neighbors is limited.

There are two remarkable phenomenons in Fig. 6. Firstly, when adopting multi-division strategy in $mSANN_P$, *ART* further decreases with the same level of *ratio* as shown in Fig. 7. Secondly, $mSANN_P$ outperforms $SANN_P$ and DCQR a lot for *Hong-Kong*. That means LSH outperforms other methods in 2-dimensional ANN.

ART. We present the comparison on *ART* between four schemes in Fig. 7. The *ARTs* of both $SANN_P$ and $mSANN_P$ are much less than those of DCQR and $SANN_T$. In general, $SANN_P$ outperforms other methods with the lowest *ratio* shown in Fig. 6, because $SANN_P$ needs the shortest time to find kNN over the candidate set. It is contrary for *Gaussian64* which will be explained in the following.

Although, $mSANN_P$ adopting multi-division strategy may intuitively leads to an increase in *ART*. However, as illustrated in Fig. 11, $mSANN_P$ needs to access fewer vectors than $SANN_P$. The reason is that the sizes of subsets in $SANN_P$ and $m$$SANN_P$ are completely different, and $SANN_P$ needs to enlarge partition size w (roughly equals to $2\% \sim 5\%$ of $\#(D)$) to ensure the accuracy of the ANN result, while the partition size in $mSANN_P$ for a single table structure is always 1 magnitude smaller than that in $SANN_P$. The total number of vectors which $mSANN_P$ accessed is smaller than

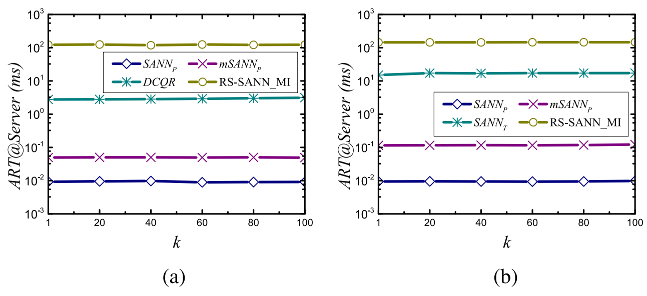
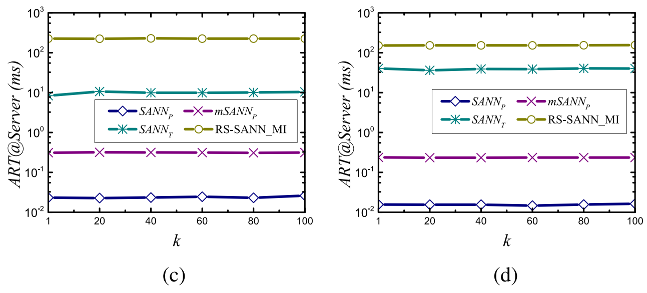



FIGURE 8. Comparisons on *ART@Server*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

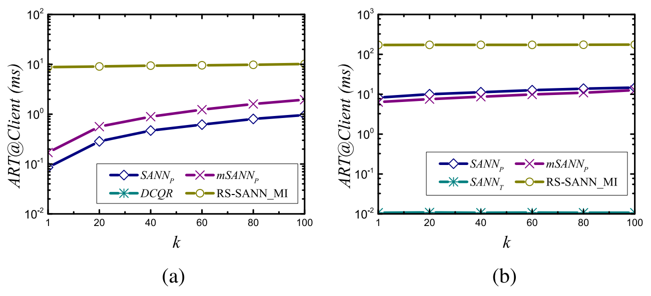
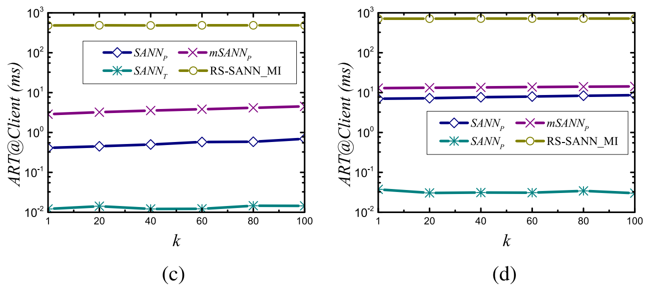



FIGURE 9. Comparisons on *ART@Client*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

that of SANN$_P$. Hence, mSANN$_P$ has a better performance in *ART* as shown in Fig. 7(b).

It is worth noting that the highest cost for RS-SANN_MI comes from two aspects. The first is that RS-SANN_MI accesses encrypted data as many as mSANN$_P$ does as shown in Fig. 11. The second is that we do not adopt caching decrypted data in RS-SANN_MI and it results in repeated decryption of the same element in dataset. However, we emphasize that the absent of caching in RS-SANN_MI does not put essential effect on the efficiency because of the equivalent of data access volume as shown in Fig. 11. Indeed, the comparisons of encrypted codes in RS-SANN_MI take most of the computation cost.

In a word, mSANN$_P$ achieves an optimal tradeoff between *ratio* and *ART*. Note that *ART* consists of *ART@Server* and *ART@Client* which are illustrated in the following.

ART@Server. The comparisons on *ART@Server* are shown in detail in Fig. 8. In general, our approaches outperform the others. At server side, SANN$_P$ exhibits the best *ART* by returning only one candidate set. mSANN$_P$ returns multiple candidate sets for promoting *ratio* and leads to an increase in *ART*. Even so, as both SANN$_T$ and DCQR need similarity computation at server side, SANN$_P$ and mSANN$_P$ provides much better *ART* than SANN$_T$, DCQR and RS-SANN_MI. The highest cost for RS-SANN_MI is due to the computation cost of the adopted comparable encryption, which is expensive than other computation at server side.

ART@Client. The comparisons on *ART@Client* of all schemes are shown in Fig. 9. Note that there is no time cost at client side for DCQR, so it is absent in Fig. 9(a). In general, our approaches perform worse than others, because we shift computational burden from *Server* to *Client*. At client side, SANN$_T$ and DCQR have better *ARTs* by returning only *k*NN results. SANN$_P$/mSANN$_P$ returns 1/multiple candidate set(s). SANN$_T$ and DCQR provide better *ARTs* than ours at client side, but they cannot recover original data from the returned data, which is not applicable to scenario described in Section I.

1) SPACE COMPLEXITY

The comparisons on space complexity of outsourced datasets and map index I are shown in Table 4. For index, the unit is the size of integers. For data, the unit is the size of floats. Our models occupy the most and the least storage space for data and index respectively. Since query speed at server side is only determined by the scale of index, our schemes have the best *ART@Server* as shown in Fig. 8.

For data space, without considering data compression (it goes beyond this research), SANN$_P$ takes a small space to store encrypted data. Both DCQR and SANN$_T$ take even less spaces because they cannot recover original data from subcontracted data. mSANN$_P$ takes the most space but provides the highest accuracy as shown in Fig. 6. Moreover, to improve the efficiency in mSANN$_P$, adopting ℓ clouds to perform **Search@Client** ℓ times is a reasonable approach(it goes beyond this research too).

Experimental results show that our approaches achieves better results on *ratio* and *ART* while preserving confidential of data compared to other solutions. Thus, our schemes are practical and exhibit many advantages when applied to high confidential scenarios.

D. OTHER PERFORMANCE

Both SANN$_P$ and mSANN$_P$ are proved secure with respect to a single query, which is not satisfied in DCQR or SANN$_T$. In this section, we evaluate other performance measures which do not exist in the other solutions.

TABLE 4. Comparisons on space complexity.

Methods		DCQR [18]/SANN$_T$ [16]	RS-SANN_MI [14]	SANN$_P$	mSANN$_P$
Hong-Kong	Data	$O((2+1)n)$	$O(3\times 2n)$	$O(2n)$	$O(3\times 2n)$
	Index	$O(2n)$	$O(3n)$	$O(2nop)$	$O(3\times 2nop)$
Gaussian64	Data	$O(16n+16^2)$	$O(4\times 64n)$	$O(64n)$	$O(11\times 64n)$
	Index	$O(n)$	$O(4n)$	$O(2nop)$	$O(11\times 2nop)$
SIFT1M	Data	$O(16n+16^2)$	$O(6\times 128n)$	$O(128n)$	$O(9\times 128n)$
	Index	$O(n)$	$O(6n)$	$O(2nop)$	$O(9\times 2nop)$
Gaussian256	Data	$O(32n+32^2)$	$O(4\times 256n)$	$O(256n)$	$O(6\times 256n)$
	Index	$O(n)$	$O(4n)$	$O(2nop)$	$O(6\times 2nop)$

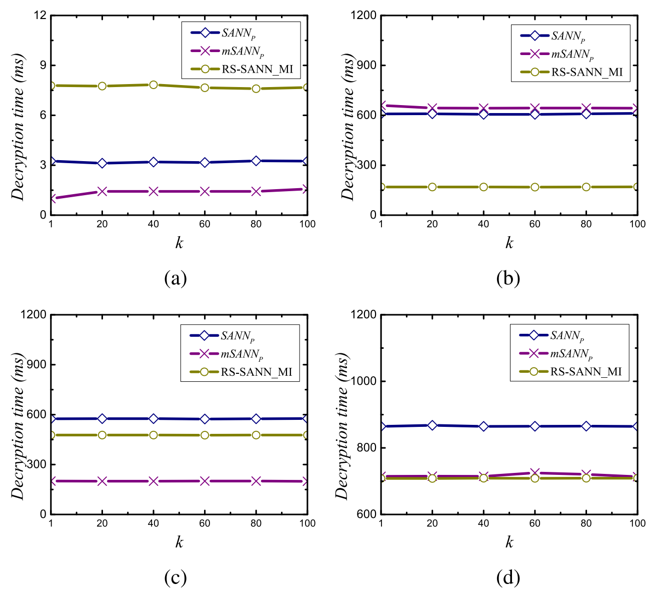


FIGURE 10. Decryption time vs. *k*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

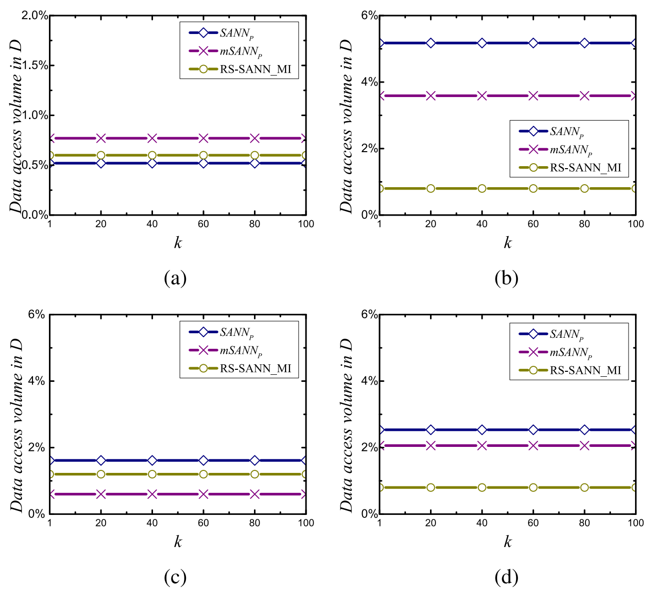


FIGURE 11. Data access volume in *D* vs. *k*. (a) *Hong-Kong*. (b) *Gaussian64*. (c) *SIFT1M*. (d) *Gaussian256*.

1) DECRYPTION TIME

In both SANN$_P$ and mSANN$_P$, after receiving encrypted $E(\Phi)$ and $E(m\Phi)$, *Client* first decrypts them and conducts kNN over Φ and $m\Phi$. The decryption process affects the efficiency of the whole solutions. Decryption time for different datasets is shown in Fig. 10. Notably, there is a significant increase in decryption time from 2-dimensional data to 64-dimensional data. However, it tends to be almost constant when dimensionality is over 64. The light increasing of decryption time from Gaussian64 to Gaussian256 comes from the significant reduction of ratio shown in Figure 6. Indeed, we can lower decryption time by increasing ratio. Hence, our approaches exhibit good scalability.

2) DATA ACCESS VOLUME IN *D*

This measure reflects the real distance computation scales of Φ and $m\Phi$ for ANN results at client side. It also reflects the communication cost because all the involved data should be transmitted to *Client* from *Server*. By varying k, data access volume is invariable because we restrict the volume of each subset to be much larger than k. In Fig. 11, it is obvious that the overlapping of subsets does not effect the efficiency of the proposed SANN$_P$ and mSANN$_P$, and data access volumes for all datasets reach a sound level.

VII. RELATED WORK

Fig. 1 exemplifies the functionality of SANN$_P$. There are two types of sensitive data, *Client*'s query and *Owner*'s dataset. Both data face different secure challenges, which are extensively discussed in latest works. We divide all these works into two categories, unidirectional transformation-based solutions and bi-directional transformation-based ones. We investigate both groups of solutions in the following.

A. UNIDIRECTIONAL TRANSFORMATION

This group of solutions focus only on providing confidentiality of query and dataset by unidirectional transformation, which means original data cannot be recovered from outsourced data. *Transformation-based SANN* (SANN$_T$) [16], [17] was proposed based on product-quantization codes (PQ-codes). In these works, original metric is transformed into a fast-compute approximate distance based on PQ-codes.

Whenever *Client* submits a query $\boldsymbol{q}$, PQ-code of $\boldsymbol{q}$ is sent to *Server*, and *Server* conducts kNN on PQ-codes of dataset. In [17], by confiscating the codebook, *Server* cannot transform PQ-codes of $\boldsymbol{q}$ back into $\boldsymbol{q}$ and therefore $\boldsymbol{q}$ is protected against disclosure. In [16], combining both symmetric and asymmetric metrics for PQ-codes, the confidentiality of query and dataset is provided.

DCQR, as a *kNN with privacy protection* method, was proposed in [18] for 2-dimensional data. Both query and data are transformed into H-values from Hilbert curves. Whenever *Client* submits a query $\boldsymbol{q}$, H-value of $\boldsymbol{q}$ is sent to *Server*, and *Server* conducts kNN on dataset's H-values and sends back the result's H-values to *Client*.

Although the aforementioned approaches are effective to protect the privacies of query and dataset, they suffer from two limitations. Firstly, these approaches cannot be theoretically proved to be IND-CPA secure. Secondly, in all these approaches, *Client* cannot recover original data from the outsourced datasets. Hence, it is not applicable to the real-world applications as shown in Section I.

B. BI-DIRECTIONAL TRANSFORMATION

This group of solutions additionally provide inverse transformation from outsourced data to recover original data. Wong *et al.* [3] developed an asymmetric scalar-product-preserving encryption and constructed two secure schemes that support *secure kNN* (SkNN). Similarly, Hu *et al.* [26] proposed a secure traversal framework and an encryption scheme based on privacy homomorphism. These works are fundamental in secure multimedia data management, but they are proved to be insecure in [1]. Yao *et al.* [1] focused on *secure nearest neighbor* (SNN). They proposed a secure search framework by Voronoi diagram based partition. When the dimensionality of data is greater than 2, Voronoi diagram becomes too complicated to perform partition on. Additionally, Choi *et al.* [27] proposed the first SkNN solution with IND-OCPA security for only 2-dimensional elements. Therefore, these approaches are not applicable in high-dimensional data management such as the real-world scenarios in Section I.

Elmehdwi *et al.* [2] focused on solving SkNN problem over relational database subcontracted to the cloud. The framework includes two non-colluding semi-honest clouds, while only single cloud is invoked in Fig. 1. Although a *secure squared Euclidean distance* (SSED) is proposed based on an additive homomorphic and probabilistic asymmetric encryption scheme (i.e., Paillier cryptosystem), SSED is too expensive for large-scale high-dimensional data to achieve. Kuzu *et al.* [28] proposed a secure similar search scheme, which focuses on several real applications in non-Euclidean space, such as fault-tolerant typographical, similarity search in the text information, etc. There are two main drawbacks which leads to be inapplicable for our scenarios: 1) The returned candidates cannot be guaranteed for either amount or quality of nearest neighbors; 2) The proposal is a mechanism with multi-rounds of communication, which raises security concern and reduces efficiency. Peng *et al.* [14] proposed a concrete SANN based on comparable encryption and achieved high efficiency. It owns fragile security because of the inherent drawback of comparable encryption so far. Zhu *et al.* [15]'s scheme focused only on data privacy and resolved the key sharing problem which is outside this paper. However, it does not provide rigorous security proof of data security or query security. In a word, these approaches are not applicable in scenarios showed in Section I.

VIII. CONCLUSION

In this paper, we proposed SANN$_P$ and mSANN$_P$ based on greedy partition method. First of all, a linear order of vectors in dataset is defined based on sortable characteristic of LSH-based codes. Two adjacent vectors in linear order are close to each other **with a great probability**. Afterwards, a greedy partition method is adopted to divide dataset into several encrypted subsets with different random tags. Through a carefully designed map index, *Client* first judges which subset the query falls in and then gets the corresponding cloak query. Whenever *Client* conducts a query, only a few subsets will be returned. Hence, the communication cost is reduced to $O(n_\Phi)$ which is **much smaller** than $O(n_D)$. Some theoretical results are also elaborated to illustrate effectiveness and security of the proposed schemes.

Meanwhile, extensive experiments are carried out with MIRACL library. Comparison results show that *ratio* of our models are much better than other three state-of-the-art solutions, DCQR, SANN$_T$ and RS-SANN, while *ART* in our schemes are less than that in others. In a word, SANN$_P$ and mSANN$_P$ provide a balance between accuracy and average response time, satisfy provable secure which is not satisfied in others, and are scalable for high-dimensional data. As part of future work, we will extend the framework to secure range query, promote efficiency for location-based services, subcontract the index to *Server*, etc.

APPENDIX I
PROOFS OF THEOREMS

In this appendix, we present all the skipped proofs of theorem through the main body in detail.

A. PROOF OF THEOREM 1

Proof: There are 2 parts to be proved: 1) *Server* cannot disclose the original data D; 2) *Server* cannot learn anything about $\boldsymbol{q}$. We prove the two parts respectively.

Proof of the First Part: In this part, we must provides security for the encrypted subset in the sense that no matter how much vectors are included in each subset. That means, *Server* cannot learn any piece $\boldsymbol{x_i}$ of data in the subset. Existing works assume that such premise is correct [1]. We here discuss such assumption in detail by inductive proof.

The adopted encryption scheme is IND-CPA secure, which is generally proved by reduction in previous works. That means if there exists an attacker $\mathcal{A}$ breaking the encryption algorithm, $\mathcal{A}$'s advantage is $Adv_{\mathcal{A}} \leq \epsilon$ where ϵ is a negligible function. Obviously, while each subset contains only a single piece of data, the security relies on the security of adopted encryption algorithm. Hence, the advantage of adversary in breaking SANN is $Adv_1 \leq \epsilon$ while $\#(SD_0) = \#(SD_1) = 1$ and $SD_0 \neq SD_1$. Adv_1 is a negligible function.

Then, supposing that $Adv_i \leq \tau(\epsilon)$ when $\#(SD_0) = \#(SD_1) = i$ and $SD_0 \neq SD_1$, $Adv_i \leq \tau'(\epsilon)$ is proved when $\#(SD_0') = \#(SD_1') = i+1$ and $SD_0' \neq SD_1'$. Here $\tau(\epsilon)$ and $\tau'(\epsilon)$ are both negligible function of ϵ. Hence, the theorem can be proved by mathematical induction method. Recall that, SD_0 and SD_1 share the same size as stated in **P2** in Section III-B. Therefore, in each step of inductive proof, the sizes of SD_0 and SD_1 are the same to each other.

Let $\#(SD_0) = \#(SD_1) = i$ and $Adv_i \leq \tau(\epsilon)$. For $\#(SD_0') = \#(SD_1') = i+1$, $SD_0' = SD_0 \cup \boldsymbol{q_0}$ and $SD_1' = SD_1 \cup \boldsymbol{q_1}$. If $\boldsymbol{q_0} = \boldsymbol{q_1}$, then it is obvious that $Adv_{i+1} \leq \tau(\epsilon)$ which is negligible. If $\boldsymbol{q_0} \neq \boldsymbol{q_1}$, the challenge is divided into two sub challenges. The first sub challenge is on SD_0 and SD_1, and the second one is on $\boldsymbol{q_0}$ and $\boldsymbol{q_1}$. For the first sub challenge, $Adv_f \leq \tau(\epsilon)$. For the second one, $Adv_s \leq \epsilon$. Once the adversary wins the first sub challenge or the second one, the adversary wins the game. Therefore, $Adv_{i+1} = \tau'(\epsilon) = \max\{Adv_f, Adv_s\} \leq \max\{\tau(\epsilon), \epsilon\}$, which is still negligible. Finally, the adversary's advantage of distinguishing $E(SD_0')$ and $E(SD_1')$ is negligible.

By mathematical induction, the advantage of $\mathcal{A}$ is a negligible function. So, the encrypted subset is IND-CPA secure by adopting an IND-CPA secure encryption algorithm regardless of how much vectors are included in each subset. Furthermore, for any two subsets $E(SD_a)$ and $E(SD_b)$ with the same size, an adversary can query for the two decrypted subsets SD_a and SD_b simultaneously. The adversary still cannot determine plain-cipher pair $\langle E(SD_a), SD_a\rangle$ or $\langle E(SD_b), SD_b\rangle$ with any non-negligible probability. Hence, *Server* cannot learn anything about the original data D.

Proof of the Second Part: Whenever a query $\boldsymbol{q}$ is submitted, a cloak query T is transmitted to *Server* where T consists of several random tags generated by a PRG. There is no association that can be inferred between the tags, the code boundaries and the plaintext of corresponding subsets. When an adversary captures a tag t_i, he cannot recovery the query $\boldsymbol{q}$ or even determine any reasonable range regarding to $\boldsymbol{q}$'s codes. Thus, the adversary can learn something about $\boldsymbol{q}$ if and only if he breaks the PRG. Hence, *Server* cannot learn anything about $\boldsymbol{q}$, since the PRG is non-fragile.

In a word, with respect to a single query, the greedy partition method is IND-CPA secure as long as the encryption algorithm adopted is IND-CPA secure. □

B. PROOF OF THEOREM 3

Proof: In **Multi-Preparation** phase, *Server* carries out **Preparation** offline ℓ times independently. Thus, selections of secret keys and G_m functions for each **Preparation** are completely independent. Hence, an adversary cannot infer any effective correlation between different subsets.

The greedy partition method is proved to be IND-CPA secure in Theorem 1. In addition, an adversary can only infer correlation inside a single partition but nothing else. Hence, mSANN$_{\text{P}}$ exhibits the same property as that in SANN$_{\text{P}}$. □

APPENDIX II
EXAMPLE OF GREEDY PARTITION METHOD

We exemplify the greedy partition method as shown in Fig. 3 where the size of dataset is $\#(D) = 17$. Different style triangles indicate that only a fixed number of vectors share the same corresponding code as shown in the legend. Here, $C_{\boldsymbol{p_1}} = C_{\boldsymbol{p_2}} = C_1$, $C_{\boldsymbol{p_3}} = C_{\boldsymbol{p_4}} = C_{\boldsymbol{p_5}} = C_2$, etc.

Before partition, the vectors in D are sorted ascendingly based on the predefined linear order $\langle \mathbb{C}, \leq\rangle$. Let the partition width be $w = \max_{p_i \in D} \#([C_{p_i}]_\sim)$ (e.g., $\#([C_1]_\sim) = 2, \#([C_2]_\sim) = 3, \ldots, \#([C_6]_\sim) = 4, \ldots, w = 4$).

The greedy partition method starts from greedily setting the partition width $w = \max_{\boldsymbol{p_i} \in D} \#([C_{\boldsymbol{p_i}}]_\sim)$. Formally, we first allocate $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$ and $\boldsymbol{p_4}$ to SD_1. Because the equal-code set $[C_2]_\sim$ is not completely included in SD_1, C_1 is the set as $I_{1,u}$ and $I_{1,d}$, and $\boldsymbol{p_3}$ is the next starting element to be allocated. Afterwards, we group every w vectors into a single subset downwards. Meanwhile, the map index I is created to connect subsets and random tags.

For instance, $\boldsymbol{p_{10}}, \boldsymbol{p_{11}}, \boldsymbol{p_{12}}$ and $\boldsymbol{p_{13}}$ are allocated to SD_4. However, the code value C_6 is not included in I_4 because that $[C_6]_\sim \nsubseteq SD_4$. Then, $\boldsymbol{p_{13}}, \boldsymbol{p_{14}}, \boldsymbol{p_{15}}$ and $\boldsymbol{p_{16}}$ are allocated to SD_5 since $\boldsymbol{p_{13}}$ is the next starting element to be allocated. Finally, $\boldsymbol{p_{17}}$ is the only remainder element to be allocated. Hence, three elements ($\boldsymbol{p_{14}}, \boldsymbol{p_{15}}$ and $\boldsymbol{p_{16}}$) are allocated to SD_6 while all the codes of these three elements are not included in I_6.

REFERENCES

[1] B. Yao, F. Li, and X. Xiao, "Secure nearest neighbor revisited," in *Proc. ICDE*, Apr. 2013, pp. 733–744.

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

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

[4] A. K. Mishra and S. Raghav, "Local fractal dimension based ECG arrhythmia classification," *Biomed. Signal Process. Control*, vol. 5, no. 2, pp. 114–123, 2010.

[5] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD*, 1984, pp. 47–57.

[6] S. Arya and D. M. Mount, "Approximate nearest neighbor queries in fixed dimensions," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 1993, pp. 271–280.

[7] M. Bern, "Approximate closest-point queries in high dimensions," *Inf. Process. Lett.*, vol. 45, no. 2, pp. 95–99, 1993.

[8] Y. Cao *et al.*, "Binary hashing for approximate nearest neighbor search on big data: A survey," *IEEE Access*, vol. 6, no. 99, pp. 2039–2054, 2018.

[9] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annu. ACM Symp. Comput. Geometry*, 2004, pp. 253–262.

[10] J. Gan, J. Feng, Q. Fang, and W. Ng, "Locality-sensitive hashing scheme based on dynamic collision counting," in *Proc. SIGMOD*, 2012, pp. 541–552.

[11] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. STOC*, 1998, pp. 604–613.

[12] H. Ma, J. Gou, X. Wang, J. Ke, and S. Zeng, "Sparse coefficient-based k-nearest neighbor classification," *IEEE Access*, vol. 5, pp. 16618–16634, 2017.

[13] J. R. Troncoso-Pastoriza, D. Gonzalez-Jimenez, and F. Perez-Gonzalez, "Fully private noninteractive face verification," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 7, pp. 1101–1114, Jul. 2013.

[14] Y. Peng, J. Cui, H. Li, and J. Ma, "A reusable and single-interactive model for secure approximate *k*-nearest neighbor query in cloud," *Inf. Sci.*, vol. 387, pp. 146–164, May 2017.

[15] 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.

[16] T. Furon, H. Jegou, L. Amsaleg, and B. Mathon, "Fast and secure similarity search in high dimensional space," in *Proc. WIFS*, Nov. 2013, pp. 73–78.

[17] B. Mathon, T. Furon, L. Amsaleg, and J. Bringer, "Secure and efficient approximate nearest neighbors search," in *Proc. IH&MMSec*, 2013, pp. 175–180.

[18] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proc. SSTD*, 2007, pp. 239–257.

[19] H.-P. Kriegel and P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 1, pp. 1–58, 2009.

[20] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[21] Y. Liu, J. Cui, Z. Huang, H. Li, and H. T. Shen, "SK-LSH: An efficient index structure for approximate nearest neighbor search," *Proc. VLDB Endowment*, vol. 7, no. 9, pp. 745–756, 2014.

[22] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and efficiency in high dimensional nearest neighbor search," in *Proc. SIGMOD*, Providence, RI, USA, 2009, pp. 563–576.

[23] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Jan. 2011.

[24] A. Beimel, Y. Ishai, E. Kushilevitz, and I. Orlov, "Share conversion and private information retrieval," in *Proc. CCC*, Jun. 2012, pp. 258–268.

[25] M. O. Rabin, "How to exchange secrets with oblivious transfer," Aiken Comput. Lab, Harvard Univ., Cambridge, MA, USA, Tech. Rep. TR-81, 1981.

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

[27] S. Choi, G. Ghinita, H.-S. Lim, and E. Bertino, "Secure *k*NN query processing in untrusted cloud environments," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 11, pp. 2818–2831, Nov. 2014.

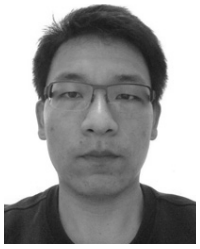
[28] M. Kuzu, M. S. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. ICDE*, 2012, pp. 1156–1167.

YANGUO PENG received the M.S. degree from Guizhou University and the Ph.D. degree from Xidian University, Xi'an, China. He is currently a Lecturer in computer architecture with Xidian University. His research interests include public key encryption with keyword search, high-dimensional secure search, and data privacy protection.

HUI LI received the B.Eng. degree from the Harbin Institute of Technology in 2005 and the Ph.D. degree from Nanyang Technological University, Singapore, in 2012. He is currently an Associate Professor with the School of Cyber Engineering, Xidian University, China. His research interests include data mining, knowledge management and discovery, privacy-preserving query, and analysis in big data.

JIANGTAO CUI received the M.S. and Ph.D. degrees in computer science from Xidian University, Xi'an, China, in 2001 and 2005, respectively. From 2007 to 2008, he was with the Data and Knowledge Engineering Group, The University of Queensland, Australia, involved in high-dimensional indexing for large scale image retrieval. He is currently a Professor with the School of Computer Science and Technology, Xidian University. His current research interests include data and knowledge engineering and high-dimensional indexing.

JIANFENG MA received the M.E. and Ph.D. degrees in computer software and communications engineering from Xidian University, Xi'an, China, in 1988 and 1995, respectively. He is currently the Committee Secretary of the School of Cyber Engineering, Xidian University. His research interests include information security, coding theory, and cryptography. He is a member of the Executive Council of the Chinese Cryptology Society and a Yangtze River Scholar Especially Hires Professor of the Ministry of Education, China.



YINGFAN LIU received the bachelor's and master's degrees in computer science from Xidian University in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of System Engineering and Engineering Management, The Chinese University of Hong Kong. His main research interest is similarity search for large-scale data.

• • •