# Privacy-Preserving Approximate Nearest Neighbor Search on High-Dimensional Data

Yingfan Liu<sup>†</sup>, Yandi Zhang<sup>†</sup>, Jiadong Xie<sup>‡</sup>, Hui Li<sup>†,§,⋆</sup>, Jeffrey Xu Yu<sup>‡</sup>, Jiangtao Cui<sup>†</sup>

†School of Computer Science and Technology, Xidian University, Xi'an, China

†The Chinese University of Hong Kong, Hong Kong SAR, China

§Shanghai Yunxi Technology, China / \*Corresponding author

liuyingfan@xidian.edu.cn, ydzhang\_2@stu.xidian.edu.cn, jdxie@se.cuhk.edu.hk

hli@xidian.edu.cn, yu@se.cuhk.edu.hk, cuijt@xidian.edu.cn

Abstract—In the era of cloud computing and AI, data owners outsource ubiquitous vectors to the cloud, which furnish approximate k-nearest neighbors (k-ANNS) services to users. To protect data privacy against the untrusted server, privacy-preserving k-ANNS (PP-ANNS) on vectors has been a fundamental and urgent problem. However, existing PP-ANNS solutions fall short of meeting the requirements of data privacy, efficiency, accuracy, and minimal user involvement concurrently. To tackle this challenge, we introduce a novel solution that primarily executes PP-ANNS on a single cloud server to avoid the heavy communication overhead between the cloud and the user. To ensure data privacy, we introduce a novel encryption method named distance comparison encryption, facilitating secure, efficient, and exact distance comparisons. To optimize the trade-off between data privacy and search performance, we design a privacy-preserving index that combines the state-of-the-art k-ANNS method with an approximate distance computation method. Then, we devise a search method using a filter-and-refine strategy based on the index. Moreover, we provide the security analysis of our solution and conduct extensive experiments to demonstrate its superiority over existing solutions. Based on our experimental results, our method accelerates PP-ANNS by up to 3 orders of magnitude compared to state-of-the-art methods, while not compromising

Index Terms—data privacy, approximate k-nearest neighbor search, proximity graph

# I. INTRODUCTION

Approximate k-nearest neighbors (k-ANN) search (k-ANNS) on high-dimensional vectors has been a fundamental problem in various fields such as machine learning [14], information retrieval [1] and retrieval-augmented generation [17]. Let  $P \subset \mathbb{R}^d$  be a database with n vectors in d-dimensional space. Consider a k-ANNS query request denoted as (q, k), where  $q \in \mathbb{R}^d$  is a d-dimensional vector and k is an integer parameter. The k-ANNS retrieves k sufficiently close vectors in the database P for the query q.

In the era of cloud computing, data owners such as enterprises and organizations often opt to delegate the management of their vector databases and k-ANNS services to the cloud to cut down on data management costs and leverage scalable cloud resources. k-ANNS techniques applied to plaintext databases typically utilize index structures like locality-sensitive hashing [22], inverted files [13], and proximity graphs [23] to accelerate the search process. However, the

cloud is not entirely trusted and is typically considered *honest-but-curious* [25], [44]. As a result, conventional k-ANNS approaches are not directly applicable in cloud settings, as they risk exposing sensitive information, e.g., the database, query, and immediate search results, to the curious cloud. Consequently, there is an urgent need for privacy-preserving k-ANNS (PP-ANNS) solutions in such contexts.

As widely recognized [25], [27], [44], a solution for PP-ANNS should effectively uphold the following three properties simultaneously:

- (P1) Privacy-preserving: It must protect the privacy of database P and the query q against the untrusted server.
- **(P2) Efficient and Accurate**: Efficiency and accuracy are the key aspects of *k*-ANNS performance. Hence, the solution should efficiently return high-quality results.
- (P3) Minimizing User Involvement and Interaction:
  The user should not be involved in the search process except for encrypting queries and receiving results due to limited computing resources.

Existing Solutions. To preserve data privacy, existing solutions [10], [11], [27], [32], [44], [45] leverage encryption techniques to secure database vectors by storing them encrypted alongside an auxiliary index in the cloud. Those encryption techniques fall into two categories: distance incomparable encryption and distance comparable encryption. The former cannot directly compare the distances of vectors over their ciphertexts, while the latter can. The encryption methods such as Advanced Encryption Standard [28] and Data Encryption Standard [28] belong to distance incomparable encryption methods, where the user retrieves a sufficient number of encrypted candidate vectors from the cloud using the index and subsequently computes their distances to the guery after vector decryption. Nonetheless, such methods contend with substantial communication overhead between the cloud and the user, leading to inefficiencies and underutilization of cloud resources. On the other hand, distance comparable encryption methods, including asymmetric scalar-product-preserving encryption (ASPE) [32], [39], [46], distance-comparisonpreserving encryption (DCPE) [10], asymmetric matrix encryption (AME) [44] and homomorphic encryption (HE) [12],

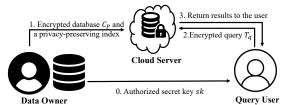


Fig. 1. The system model of our PP-ANNS scheme.

[42], [43] enable distance comparisons in the cloud, which further facilitates PP-ANNS service within the cloud upon receiving the encrypted query from the user. Unfortunately, ASPE and its variations fail to preserve data privacy as our analysis in Section III-A, while DCPE produces inaccurate results. Both AME and HE suffer from huge computational costs. Therefore, none of the existing solutions satisfies all three properties concurrently.

Our Solution. To fulfill P3, we introduce a new PP-ANNS scheme with the system model shown in Figure 1. The data owner outsources the encrypted database and a privacy-preserving index to the cloud. Throughout the search process, the user solely computes the encrypted query  $T_{m q}$  and forwards it to the cloud, which in turn processes the query and transmits the results back to the user.

To satisfy **P1**, we introduce a novel encryption technique named distance comparison encryption (DCE), which securely, exactly and efficiently answers distance comparisons over encrypted vectors. It comprises two phases: vector randomization and vector transformation. The first phase generates a random vector for the input vector through processes like random permutation, vector splitting, and the addition of random values, while the second phase converts the generated random vector into a DCE encrypted vector via operations such as element-wise vector manipulations and matrix encryption. We then provide theoretical proof of the correctness of distance comparison using the DCE method and analyze its efficiency and security.

To meet **P2**, integrating a privacy-preserving index into our DCE method is necessary to avoid scanning the entire database while returning accurate results. State-of-the-art k-ANNS techniques, proximity graphs such as hierarchical navigable small world (HNSW) [23], notably outperform alternatives like locality-sensitive hashing and inverted files [31]. A simple method is to deploy a proximity graph such as HNSW directly in the cloud and conduct k-ANNS on the graph with distance comparisons on DCE encrypted vectors. However, such a naive method causes two issues, i.e., (1) the risk of exposing sensitive information since the edges in the graph indicate the relationships between vectors, and (2) low efficiency since each DCE computation costs at least  $4\times$  that of a normal distance computation as analyzed in Section IV-B. To enhance data privacy, we design a privacy-preserving index that constructs an HNSW graph over the encrypted database using distancecomparison-preserving encryption (DCPE) [10], instead of the plaintext database. While DCPE is secure and efficient [10], it answers only approximate distances, and thus conducting k-ANNS over the DCPE encrypted vectors will decrease the accuracy even using HNSW as the index. To balance efficiency and accuracy, we develop a search approach based on the *filter* and-refine strategy. At a high level, in the *filter* phase, the k-ANN search with HNSW on DCPE encrypted database yields a selected group of top-quality candidates, where distances are computed over the DCPE encrypted vectors. Note that such a distance computation costs exactly the same as the normal distance computation. Next, in the *refine* phase, the server conducts exact distance comparisons based on our DCE method among these candidates to identify the best k vectors. In this way, we reduce the number of costly distance comparisons of DCE while not compromising the accuracy.

Contributions. Our main contributions are as follows.

- We introduce a novel encryption method, DCE, facilitating secure, exact, and efficient distance comparisons among vectors.
- We design a privacy-preserving index and introduce a filter-and-refine search strategy based on this index.
- Equipped with the above techniques, we present a novel PP-ANNS scheme that upholds all three properties.
- We conduct a theoretical security analysis of our proposed PP-ANNS scheme and perform comprehensive experiments on real-world datasets to showcase the superiority of our approach over state-of-the-art methods. Our experimental results demonstrate that our approach can be up to 1000× faster than state-of-the-art methods while maintaining comparable accuracy.

**Organization.** The remainder of this paper is organized as follows. We present the preliminaries in Section II and revisit the distance comparable encryption methods in III. In Section IV, we show our distance comparable encryption method DCE. Section V describes our PP-ANNS scheme. We present the security analysis in Section VI and performance evaluation in Section VII, respectively. Section VIII introduces the related works. Finally, we conclude this paper in Section IX.

#### II. PRELIMINARIES

In this section, we first present our system model and threat model, respectively. Afterward, we formally define the problem and present the challenges of our problem.

# A. System Model

In this work, we focus on the single-server and noninteractive scenario, where there are three types of participants in the system, i.e., data owner, user, and cloud server.

- Data Owner: Data owner owns a database  $P \subset \mathbb{R}^d$ , which includes n d-dimensional vectors. Due to the limited capability of computation and storage, data owner encrypts each vector  $\mathbf{p} \in P$  into ciphertexts  $C_{\mathbf{p}}$ , which is then outsourced to the cloud server. For simplicity, we define  $C_P = \{C_{\mathbf{p}} | \mathbf{p} \in P\}$ .
- User: User enjoys the query service by giving a query request with (q,k), where  $q \in \mathbb{R}^d$  is a query vector and  $k \in \mathbb{N}^+$  is the number of returned neighbors. To preserve the privacy of q, the user encrypts q into ciphertext  $T_q$  that is then sent to the cloud server to answer the query.

• Cloud Server: Cloud server stores the encrypted database  $C_P$  and responds to query requrests from user. When receiving  $(T_q, k)$ , it searches on  $C_P$  to retrieve k close neighbors and then returns to the user.

#### B. Threat Model

In this work, the data owner is trusted, and the user is assumed to be honest. Both of them will honestly use the k-ANNS service, while not leaking their keys to attackers or colluding with the server. On the other hand, the server is considered to be honest-but-curious. It means that the server will sincerely store the encrypted database for the data owner and offer the k-ANNS service to the user strictly following the predefined scheme. However, it is curious about sensitive information such as the database and queries. In this work, we aim to prevent the server from obtaining the database, the query vector, and immediate distance values during the search process. Besides, in line with the majority of secure query schemes with server-side index [25], [27], we compromise the information revealed by the index for the sake of the efficiency and accuracy of PP-ANNS. Although the index is built on pure ciphertexts, there still exists some information leaked by the index, e.g., the neighborhood relationship of vectors. Except that, nothing more should be revealed. As to other active attacks such as side-channel attacks, DOS attacks, and data pollution attacks, we leave them in our future works.

# C. Problem Definition and Challenges

Based on our system model and threat model, we formally define the privacy-preserving k-ANN search as follows.

Definition 1 (PP-ANNS): Under our system model, given an encrypted query  $T_q$  and an integer k, privacy-preserving k-ANN search (PP-ANNS) problem aims to return approximate k neighbors that are sufficiently close to the query vector  $\mathbf{q}$  by searching the encrypted database  $C_P$ , without leaking P or  $\mathbf{q}$  to cloud server under our threat model.

In this work, we use Euclidean distance to measure the distance between two vectors. Given a vector  $\mathbf{p} \in P$  and a query vector  $\mathbf{q}$ , the squared Euclidean distance between them is denoted as  $dist(\mathbf{p},\mathbf{q}) = ||\mathbf{p},\mathbf{q}||^2 = \sum_{i=1}^d (p_i - q_i)^2$ .

Based on the threat model, we advocate that a secure solution to PP-ANNS should satisfy the <u>ind</u>istinguishability under <u>known-plaintext attack (IND-KPA) [5]</u> in the real/ideal world security model. Let  $\mathcal L$  be the leaked information, the ideal world involves a probabilistic polynomial time adversary  $\mathcal A$  and a simulator with leakage  $\mathcal L$ . The real world involves a probabilistic polynomial time adversary  $\mathcal A$  and a challenger. We can formally define the IND-KPA security as follows.

Definition 2 (IND-KPA Security): A scheme is IND-KPA secure with leakage  $\mathcal{L}$  iff for any  $\mathcal{A}$  issuing a polynomial number of interactions, there exists a simulator such that the advantage that  $\mathcal{A}$  can distinguish the views of real and ideal experiments is negligible, i.e.,  $|\Pr[\text{View}_{\mathcal{A},\text{Real}} = 1] - \Pr[\text{View}_{\mathcal{A},\text{Ideal}} = 1]|$  is negligible.

To design a solution to PP-ANNS, as we previously discussed in the introduction, we must address two primary

challenges: (1) how to securely, exactly, and efficiently conduct distance comparisons, and (2) how to reduce the number of distance comparisons during *k*-ANNS. Notably, secure, exact, and efficient distance comparisons are fundamental operations in PP-ANNS. In tackling the first challenge, we first revisit existing methods and ascertain that none achieve the trifecta of security, exactness, and efficiency concurrently in Section III. Motivated by this, we introduce our DCE scheme in Section IV. In addressing the second challenge, we propose a privacy-preserving index that integrates the state-of-the-art *k*-ANNS method HNSW, and an approximate distance comparison encryption method DCPE, which strikes a balance between data privacy and search performance, in Section V.

# III. REVISITING SECURE DISTANCE COMPARISON

In this section, we revisit existing secure distance comparison (SDC) methods to determine that none of them achieves security, precision, and efficiency simultaneously.

SDC on the server is an essential operation in PP-ANNS. Existing SDC methods fall into two categories: those that compare distances through distance computation and those that perform direct distance comparisons. The former calculates distances over encrypted vectors to compare them, which contains methods based on asymmetric scalar-product-preserving encryption (ASPE) [11], [18], [24], [37] and distance-comparison-preserving encryption (DCPE) [10]. While the latter directly yields the result of the distance comparison without intermediate computations, which includes asymmetric matrix encryption (AME) [44] and homomorphic encryption (HE) [11], [34]. Note that, in the analysis in this section, we exclude HE-based methods due to their significant computational overhead [44].

#### A. ASPE Schemes Revisited

First, we briefly explain the ASPE scheme [32]. Given two vectors  $\boldsymbol{p},\boldsymbol{q}\in\mathbb{R}^d$  and an invertible matrix  $M\in\mathbb{R}^{d\times d}$ , we get the ciphertexts of vectors, i.e.,  $\operatorname{Enc}(\boldsymbol{p})=(\boldsymbol{p}^T\times M)^T$  and  $\operatorname{Enc}(\boldsymbol{q})=M^{-1}\times\boldsymbol{q}$ . The inner product  $\boldsymbol{p}^T\boldsymbol{q}$  can be recovered by  $\operatorname{Enc}(\boldsymbol{p})^T\times\operatorname{Enc}(\boldsymbol{q})$ , i.e., the inner product of their corresponding encrypted vectors. Even though ASPE and its variants were created for the inner product, we can convert the squared Euclidean distance  $||\boldsymbol{p},\boldsymbol{q}||^2$  to the inner product of two new vectors, i.e.,  $\boldsymbol{p}'=[\boldsymbol{p},1,||\boldsymbol{p}||^2]^T$  and  $\boldsymbol{q}'=[-2\boldsymbol{q},||\boldsymbol{q}||^2,1]^T$ .

Unfortunately, ASPE has been demonstrated to be KPA insecure for Euclidean distance and inner product [18], [20], [36]. Hence, several ASPE variants were proposed to strengthen its security [6], [29] via specific transformations of distance values. In the following, we revisit the security of these ASPE variants and prove that they are not KPA secure.

To be formal, we present the assumption under KPA. Let P be the database and Q be a set of queries. Given the encrypted database  $C_P = \{C_{\boldsymbol{p}} | \boldsymbol{p} \in P\}$  and the encrypted query set  $C_Q = \{T_{\boldsymbol{q}} | \boldsymbol{q} \in Q\}$ . We denote  $L(C_{\boldsymbol{p}}, T_{\boldsymbol{q}})$  as the information about  $dist(\boldsymbol{p}, \boldsymbol{q})$  leaked from the ciphertexts, where  $\boldsymbol{p} \in P, \boldsymbol{q} \in Q$ . We assume that the attacker owns

 $C_Q$ ,  $C_P$ , and  $P_{leak} \subset P$ , where  $|P_{leak}| = m$ .  $L(C_p, T_q)$  in ASPE and its enhanced schemes is a specific transformation of dist(p,q), such as linear [8], exponential [3], logarithmic [15], and square [2]. Then, we prove that those additional transformations cannot prevent the attacker from recovering the plaintexts of P and Q.

*Theorem 1:* The enhanced ASPE scheme that leaks **the linear transformation of distances** is not KPA secure.

Proof: Given  $C_Q$ ,  $C_P$  and  $P_{leak}$ , to recover a query vector  ${m q}$ , attacker constructs d+2 equations according to ASPE with linear transformation,  $[-2{m p}_i^T,||{m p}_i||^2,1] \times [r_1{m q}^T,r_1,r_2]^T = L(C_{{m p}_i},T_{{m q}})$ , where  ${m p}_i\in P_{leak},~1\leq i\leq d+2$  and  $r_1,r_2$  are random numbers. Those d+2 equations could be reformed as  $M_c{m x}={m b}$ , where the i-th  $(1\leq i\leq d+2)$  row of matrix  $M_c\in \mathbb{R}^{(d+2)\times(d+2)}$  is  ${m p}_i^T$ , the vector  ${m b}=[L(C_{{m p}_i},T_{{m q}}),\cdots,L(C_{{m p}_d},T_{{m q}_{d+2}})]^T$  and  ${m x}=[r_1{m q}^T,r_1,r_2]^T$ . When  $M_c$  is invertible,  ${m x}=M_c^{-1}{m b}$  and thus  ${m q}$  is recovered. Similarly, with d+2 recovered queries, to recover any database vector  ${m p}\in P\setminus P_{leak}$ , attacker constructs another d+2 equations in a similar way,  $[-2{m p}^T,||{m p}||^2,1]\times[r_{1j}{m q}_j^T,r_{1j},r_{2j}]^T=L(C_{{m p}},T_{{m q}_j})$   $(1\leq j\leq d+2)$ . Similarly,  ${m p}$  could be recovered. To sum up, the enhanced ASPE scheme that leaks the linear transformation of distances is not secure.

Corollary 1: The enhanced ASPE scheme that leaks the exponential transformation of distances is not KPA secure.

Proof: Given  $C_Q$ ,  $C_P$  and  $P_{leak}$ , to recover a query vector  ${\bf q}$ , attacker constructs d+2 equations according to ASPE with exponential transformation,  $[-2{\bf p}_i^T,\|{\bf p}_i\|^2,1] \times [r_1{\bf q}^T,r_1,r_2]^T=\ln L(C_{{\bf p}_i},T_{{\bf q}})$ , where  ${\bf p}_i\in P_{leak},\ 1\leq i\leq d+2$  and  $r_1,r_2\in \mathbb{R}$  are two random numbers. It is converted to Theorem 1 by replacing  $L(C_{{\bf p}_i},T_{{\bf q}})$  with its logarithmic value. Following the same idea of Theorem 1, each query vector could be recovered. Then, with d+2 recovered queries, each database vector can be recovered in the same way.

Corollary 2: The enhanced ASPE scheme that leaks the logarithmic transformation of distances is not KPA secure.

Proof: Given  $C_Q$ ,  $C_P$  and  $P_{leak}$ , to recover a query vector  $\boldsymbol{q}$ , attacker constructs d+2 equations according to ASPE with logarithmic transformation,  $[-2\boldsymbol{p}_i^T,\|\boldsymbol{p}_i\|^2,1] \times [r_1\boldsymbol{q}^T,r_1,r_2]^T=e^{L(C_{\boldsymbol{p}_i},T_{\boldsymbol{q}})}$ , where  $\boldsymbol{p}_i\in P_{leak}, 1\leq i\leq d+2$  and  $r_1,r_2\in\mathbb{R}$  are two random numbers. It is converted to Theorem 1 by replacing  $L(C_{\boldsymbol{p}_i},T_{\boldsymbol{q}})$  with its exponential value. Following the same idea of Theorem 1, the query vector  $\boldsymbol{q}$  could be recovered. In the same way, with d+2 recovered queries, each database vector can be recovered.

Theorem 2: The enhanced ASPE scheme that leaks the square of distances is not KPA secure.

*Proof:* Given  $C_Q$ ,  $C_P$  and  $P_{leak}$ , to recover a query vector  $\boldsymbol{q}$ , according to the ASPE scheme with the square of distances, we have  $L(C_{\boldsymbol{p}_i}, T_{\boldsymbol{q}}) = ([-2\boldsymbol{p}_i^T, \|\boldsymbol{p}_i\|^2, 1] \times [r_1\boldsymbol{q}^T, r_1, r_2]^T)^2$ . By extending this equation, we have

$$L(C_{p_i}, T_q) = r_1 \cdot (\|p_i\|^2 - 2 \cdot p_i^T q + r_2)^2 + r_3$$
  
=  $r_1 \cdot (\|p_i\|^4 - 4\|p_i\|^2 \cdot p_i^T q + 2\|p_i\|^2 \cdot r_2$   
+  $4 \cdot (p_i^T q)^2 - 4 \cdot p_i^T q \cdot r_2 + r_2^2) + r_3.$ 

For each vector  $\boldsymbol{p} = [p_1, p_2, \cdots, p_d]^T$ ,  $\boldsymbol{p}^2 = [p_1^2, p_2^2, \cdots, p_d^2]^T$  and  $\ddot{\boldsymbol{p}}^{-i} = [p_1, \cdots, p_{i-1}, p_{i+1}, \cdots, p_d]^T$ . Here,  $\ddot{\boldsymbol{p}}^{-i} \in \mathbb{R}^{d-1}$  is obtained by just removing the *i*-th coordinate of  $\boldsymbol{p}$ . Then,  $L(C_{\boldsymbol{p}_i}, T_{\boldsymbol{p}})$  could be represented as the inner product of two new vectors  $\boldsymbol{p}_i', \boldsymbol{q}' \in \mathbb{R}^{0.5d^2+2.5d+3}$ , where

$$\mathbf{p}'_{i} = [\|\mathbf{p}_{i}\|^{4}, \|\mathbf{p}_{i}\|^{2} \cdot \mathbf{p}_{i}, \|\mathbf{p}_{i}\|^{2}, 4 \cdot \mathbf{p}^{2}, 8p_{i1} \cdot \ddot{\mathbf{p}}_{i}^{-1}, 8p_{i2} \cdot \ddot{\mathbf{p}}_{i}^{-2}, \\ \cdots, 8p_{id} \cdot \ddot{\mathbf{p}}_{i}^{-d}, -4 \cdot \mathbf{p}_{i}, 1]^{T},$$

$$\mathbf{q}' = [r_{1}, -4r_{1} \cdot \mathbf{q}, 2r_{2}, r_{1} \cdot \mathbf{q}^{2}, r_{1}q_{1} \cdot \ddot{\mathbf{q}}^{-1}, r_{1}q_{2} \cdot \ddot{\mathbf{q}}^{-2}, \\ \cdots, r_{1}q_{d} \cdot \ddot{\mathbf{q}}^{-d}, r_{1}r_{2} \cdot \mathbf{q}, r_{1}r_{2}^{2} + r_{3}]^{T}.$$

Here,  $p_i = [p_{i1}, \cdots, p_{id}]^T$ . Given  $T_q$ ,  $C_P$  and  $P_{leak}$ , to recover a query vector  $\mathbf{q}$ , attacker constructs  $0.5d^2 + 2.5d + 3$  equations via the above two new vectors,  $\mathbf{p}_i'^T \mathbf{q}' = L(C_{\mathbf{p}_i}, T_{\mathbf{q}})$ , where  $1 \leq i \leq 0.5d^2 + 2.5d + 3$ . Following the same idea of Theorem 1,  $\mathbf{q}'$  could be recovered, and thus  $\mathbf{q}$  could be recovered. Again, with  $0.5d^2 + 2.5d + 3$  recovered queries, to recover each database vector  $\mathbf{p} \in P \setminus P_{leak}$ , the attacker constructs  $0.5d^2 + 2.5d + 3$  equations as the same procedure. Hence, the scheme that leaks the distances is not secure.

Moreover, other existing APSE variants [16], [24], [37] could be aligned with the aforementioned cases in this part. *In conclusion, ASPE schemes that reveal specific transformations of distances are not KPA secure.* 

#### B. DCPE Scheme Revisited

Unlike ASPE-based methods that reveal the distance values or their transformations, distance-comparison-preserving encryption (DCPE) only reveals approximate distance values by the  $\beta$ -approximate-distance-comparison-preserving ( $\beta$ -DCP) function [10].

Definition 3:  $(\beta\text{-DCP})$  For  $\beta \in \mathbb{R}^+$ , a  $\beta$ -DCP function  $f: \mathbb{R}^d \to \mathbb{R}^d$  satisfies that  $\forall o, p, q \in \mathbb{R}^d$ , if  $dist(o, q) < dist(p, q) - \beta$ , then dist(f(o), f(q)) < dist(f(p), f(q)).

Let  $f(P) = \{f(p) | p \in P\}$ . Let  $u^*$  be the exact nearest neighbor of f(q) on f(P). The authors in [10] claim that  $dist(q, u^*) \leq dist(q, p) + \beta$  holds for each  $p \in P$ . Besides, they propose such a  $\beta$ -DCP function which first scales each vector  $p \in P$  with the same scaling factor and then adds a random vector to p. As proven in [10], DCPE is IND-KPA secure. Notabley, the encrypted vector still has d dimensions, and dist(f(p), f(q)) costs the same as dist(f(o), f(q)). Hence, DCPE is efficient, but only returns approximate distance values. PP-ANNS with DCPE cannot ensure the accuracy.

# C. AME Scheme Revisited

Different to ASPE and DCPE, which reveal information about distance values, asymmetric matrix encryption (AME) [44] only reveals the results of distance comparisons. As in [44], AME is KPA secure, but pretty costly. To be specific, AME generates the secret key consisting of 32 matrices in  $\mathbb{R}^{(2d+6)\times(2d+6)}$ . With those matrices, each database vector is encrypted into 32 vectors in  $\mathbb{R}^{(2d+6)}$  and each query vector into 16 matrices in  $\mathbb{R}^{(2d+6)\times(2d+6)}$ . Thus, AME is space-inefficient. When calculating each SDC, AME conducts

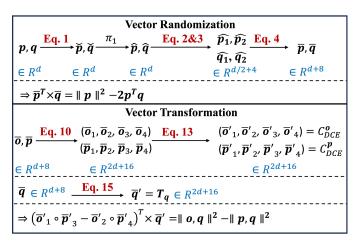


Fig. 2. The procedure of our DCE scheme, where  ${m o}, {m p} \in P$  and  ${m q}$  is the query vector.

16 vector-matrix multiplications and 16 inner products between vectors, where each vector is in  $\mathbb{R}^{(2d+6)}$  and each matrix in  $\mathbb{R}^{(2d+6)\times(2d+6)}$ . Hence, each SDC in AME requires  $64d^2+416d+676$  multiply-and-accumulate operations.

Hence, AME is neither cost-efficient nor space-efficient, especially for high-dimensional vectors. Our empirical study in Section VII-B also justifies this point.

# IV. DISTANCE COMPARISON ENCRYPTION

In this section, we present our SDC scheme, called Distance Comparison Encryption (DCE), which can process distance comparisons securely, exactly, and efficiently.

# A. Main Idea of Our Scheme

In this part, we present the core idea of DCE scheme, which consists of two phases: (1) **vector randomization** and (2) **vector transformation**, as illustrated in Figure 2. The former generates a random vector on top of the original one in order to add randomness and scramble attackers, while the latter further transforms the output of the first phase to produce the final ciphertexts.

**Vector Randomization**. For simplicity, let  $\mathbf{p} = [p_1, \dots, p_d] \in \mathbb{R}^d$  be a database vector and  $\mathbf{q} = [q_1, \dots, q_d] \in \mathbb{R}^d$  be a query vector. This phase consists of 4 steps.

Step 1: two new vectors  $\check{p}$  and  $\check{q}$  are generated according to the following equation, with  $\check{p}^T\check{q} = -2p^Tq$ :

$$\begin{cases} \tilde{\mathbf{p}} = [p_1 + p_2, p_1 - p_2, p_3 + p_4, \cdots, p_{d-1} + p_d, p_{d-1} - p_d]^T, \\ \tilde{\mathbf{q}} = -[q_1 + q_2, q_1 - q_2, q_3 + q_4, \dots, q_{d-1} + q_d, q_{d-1} - q_d]^T. \end{cases}$$
(1)

Step 2: random permutation. Let  $\pi_1 : \mathbb{R}^d \to \mathbb{R}^d$  be a random permutation on vector. Then, we have  $\hat{p} = \pi_1(\check{p})$  and  $\hat{q} = \pi_1(\check{q})$ . Moreover,  $\hat{p}^T \hat{q} = \check{p}^T \check{q} = -2p^T q$  holds.

Step 3: vector spliting and random numbers addition. First, four random nubmers  $r_1, r_2, r_3, r_4 \in \mathbb{R}$  are randomly selected for all database and query vectors. Then, we select five random numbers  $\alpha_{\boldsymbol{p},1}, \alpha_{\boldsymbol{p},2}, r'_{\boldsymbol{p},1}, r'_{\boldsymbol{p},2}, r'_{\boldsymbol{p},3} \in \mathbb{R}$  for  $\boldsymbol{p} \in P$ , and two random numbers  $\beta_{\boldsymbol{q},1}, \beta_{\boldsymbol{q},2} \in \mathbb{R}$  for each query vector  $\boldsymbol{q}$ . Next,

we define a variable  $\gamma_{\boldsymbol{p}}=(||\boldsymbol{p}||^2-r_1'r_1-r_2'r_2-r_3'r_3)/r_4$  accordingly.  $\widehat{\boldsymbol{p}}$  is divided into two vectors with random numbers via the following equation:

$$\begin{cases}
\widehat{\boldsymbol{p}}_{1} = \left[\widehat{p}_{1}, \widehat{p}_{2}, \dots, \widehat{p}_{d/2}, \alpha_{\boldsymbol{p},1}, -\alpha_{\boldsymbol{p},1}, r'_{\boldsymbol{p},1}, r'_{\boldsymbol{p},2}\right]^{T}, \\
\widehat{\boldsymbol{p}}_{2} = \left[\widehat{p}_{d/2+1}, \widehat{p}_{d/2+2}, \dots, \widehat{p}_{d}, \alpha_{\boldsymbol{p},2}, \alpha_{\boldsymbol{p},2}, r'_{\boldsymbol{p},3}, \gamma_{\boldsymbol{p}}\right]^{T}.
\end{cases}$$
(2)

Similarly,  $\hat{q}$  is divided into two vectors:

$$\begin{cases}
\widehat{q}_{1} = [\widehat{q}_{1}, \widehat{q}_{2}, \dots, \widehat{q}_{d/2}, \beta_{q,1}, \beta_{q,1}, r_{1}, r_{2}]^{T}, \\
\widehat{q}_{2} = [\widehat{q}_{d/2+1}, \widehat{q}_{d/2+2}, \dots, \widehat{q}_{d}, \beta_{q,2}, -\beta_{q,2}, r_{3}, r_{4}]^{T}.
\end{cases} (3)$$

Based on this, we have  $[\widehat{p}_1^T, \widehat{p}_2^T] \times [\widehat{q}_1^T, \widehat{q}_2^T]^T = ||p||^2 - 2p^Tq$ . Step 4: matrix encryption and random permutation. Let  $M_1, M_2 \in \mathbb{R}^{(d/2+4)\times(d/2+4)}$  be two random invertible matries, and  $\pi_2 : \mathbb{R}^{(d+8)} \to \mathbb{R}^{(d+8)}$  be a random permutation on vector. We define two variables  $\bar{p}, \bar{q}$  as follows:

$$\begin{cases} \bar{\boldsymbol{p}} = \pi_2([\hat{\boldsymbol{p}}_1^T M_1, \hat{\boldsymbol{p}}_2^T M_2]^T), \\ \bar{\boldsymbol{q}} = \pi_2([(M_1^{-1} \hat{\boldsymbol{q}}_1)^T, (M_2^{-1} \hat{\boldsymbol{q}}_2)^T]^T). \end{cases}$$
(4)

Further, we have the following equation:

$$\bar{\boldsymbol{p}}^T \bar{\boldsymbol{q}} = [\hat{\boldsymbol{p}}_1^T, \hat{\boldsymbol{p}}_2^T] \times [\hat{\boldsymbol{q}}_1^T, \hat{\boldsymbol{q}}_2^T]^T = ||\boldsymbol{p}||^2 - 2\boldsymbol{p}^T \boldsymbol{q}.$$
 (5)

According to the four steps of vector randomization, we generate  $\bar{p}, \bar{q} \in \mathbb{R}^{d+8}$  for  $p, q \in \mathbb{R}^d$  respectively.

**Vector Transformation**. Let  $o \in \mathbb{R}^d$  be another database vector and  $\bar{o} \in \mathbb{R}^{d+8}$  be its result of vector randomization. To answer whether dist(o,q) < dist(p,q), it equal to determine the sign of  $\bar{o}^T \bar{q} - \bar{p}^T \bar{q}$  according to Equation 5. Before further discussion, we define several element-wise operators between two vectors. Let  $\mathbf{a} = [a_1, \dots, a_d]$  and  $\mathbf{b} = [b_1, \dots, b_d]$ .

- Element-Wise Addition:  $\mathbf{a} + \mathbf{b} = [a_1 + b_1, \dots, a_d + b_d];$
- *Element-Wise Minus:*  $a b = [a_1 b_1, ..., a_d b_d];$
- Element-Wise Multiplication:  $\mathbf{a} \circ \mathbf{b} = [a_1 \cdot b_1, \dots, a_d \cdot b_d];$
- Element-Wise Division:  $\mathbf{a}/\mathbf{b} = [a_1/b_1, \dots, a_d/b_d].$

Let  $\mathbf{1}_d$  be the d-dimensional vector with all dimensions are 1, then we have

$$2a + 2b = (a + 1_d) \circ (b + 1_d) - (a - 1_d) \circ (b - 1_d).$$
 (6)

Let  $c, d \in \mathbb{R}^d$  be another two vectors, we have

$$(\boldsymbol{a} \circ \boldsymbol{b})/(\boldsymbol{c} \circ \boldsymbol{d}) = (\boldsymbol{a}/\boldsymbol{c}) \circ (\boldsymbol{b}/\boldsymbol{d}). \tag{7}$$

Now, let us focus on how to determine the sign of  $o^Tq-p^Tq$  via vector transformation. Here, let us introduce a random invertible matrix  $M_3 \in \mathbb{R}^{(2d+16)\times(2d+16)}$ , which is further divided into two sub-matrices, i.e.,  $M_{up}, M_{down} \in \mathbb{R}^{(d+8)\times(2d+16)}$ .  $M_{up}$  represents the first d+8 rows of  $M_3$  and  $M_{down}$  the other d+8 rows. Hence, we have

$$\bar{\boldsymbol{o}}^T \bar{\boldsymbol{q}} - \bar{\boldsymbol{p}}^T \bar{\boldsymbol{q}} = [\bar{\boldsymbol{o}}^T, \bar{\boldsymbol{p}}^T] \times [\bar{\boldsymbol{q}}^T, -\bar{\boldsymbol{q}}^T]^T \\
= \left( [\bar{\boldsymbol{o}}^T, \bar{\boldsymbol{p}}^T] \times \begin{bmatrix} M_{up} \\ M_{down} \end{bmatrix} \right) \times \left( M_3^{-1} \times [\bar{\boldsymbol{q}}^T, -\bar{\boldsymbol{q}}^T]^T \right).$$
(8)

Vector transformation is on top of Equation 8. We further extend the first term in the second line of Equation 8 and derive the following equation with Equation 6:

$$\mathcal{F}_{1}(\bar{\boldsymbol{o}}, \bar{\boldsymbol{p}}) = 2[\bar{\boldsymbol{o}}^{T}, \bar{\boldsymbol{p}}^{T}] \times \begin{bmatrix} M_{up} \\ M_{down} \end{bmatrix}$$

$$= 2\bar{\boldsymbol{o}}^{T} M_{up} + 2\bar{\boldsymbol{p}}^{T} M_{down}$$

$$= (\bar{\boldsymbol{o}}^{T} M_{up} + \mathbf{1}_{2d+16}) \circ (\bar{\boldsymbol{p}}^{T} M_{domn} + \mathbf{1}_{2d+16}) - (\bar{\boldsymbol{o}}^{T} M_{up} - \mathbf{1}_{2d+16}) \circ (\bar{\boldsymbol{p}}^{T} M_{domn} - \mathbf{1}_{2d+16}).$$

$$(9)$$

For simplicity, we denote four vectors for  $\bar{o}$  and  $\bar{p}$  as follows:

$$\begin{cases}
\bar{\boldsymbol{o}}_{1} = \bar{\boldsymbol{o}}^{T} M_{up} + \mathbf{1}_{2d+16}, & \bar{\boldsymbol{p}}_{1} = \bar{\boldsymbol{p}}^{T} M_{up} + \mathbf{1}_{2d+16} \\
\bar{\boldsymbol{o}}_{2} = \bar{\boldsymbol{o}}^{T} M_{up} - \mathbf{1}_{2d+16}, & \bar{\boldsymbol{p}}_{2} = \bar{\boldsymbol{p}}^{T} M_{up} - \mathbf{1}_{2d+16} \\
\bar{\boldsymbol{o}}_{3} = \bar{\boldsymbol{o}}^{T} M_{down} + \mathbf{1}_{2d+16}, & \bar{\boldsymbol{p}}_{3} = \bar{\boldsymbol{p}}^{T} M_{down} + \mathbf{1}_{2d+16} \\
\bar{\boldsymbol{o}}_{4} = \bar{\boldsymbol{o}}^{T} M_{down} - \mathbf{1}_{2d+16}, & \bar{\boldsymbol{p}}_{4} = \bar{\boldsymbol{p}}^{T} M_{down} - \mathbf{1}_{2d+16}
\end{cases}$$
(10)

With those vectors, we can use the following equation:

$$\mathcal{F}_1(\bar{\boldsymbol{o}}, \bar{\boldsymbol{p}}) = \bar{\boldsymbol{o}}_1 \circ \bar{\boldsymbol{p}}_3 - \bar{\boldsymbol{o}}_2 \circ \bar{\boldsymbol{p}}_4. \tag{11}$$

Let  $k_{v1}, k_{v2}, k_{v3}, k_{v4} \in \mathbb{R}^{2d+16}$  be four random vectors and satisfy  $k_{v1} \circ k_{v3} = k_{v2} \circ k_{v4}$ . We have the following equation by combining Equation 7 and Equation 11:

$$\mathcal{F}_{2}(\bar{\boldsymbol{o}}, \bar{\boldsymbol{p}}) = \mathcal{F}_{1}(\bar{\boldsymbol{o}}, \bar{\boldsymbol{p}})/(k_{v1} \circ k_{v3})$$

$$= (\bar{\boldsymbol{o}}_{1} \circ \bar{\boldsymbol{p}}_{3})/(k_{v1} \circ k_{v3}) - (\bar{\boldsymbol{o}}_{2} \circ \bar{\boldsymbol{p}}_{4})/(k_{v2} \circ k_{v4})$$

$$= (\bar{\boldsymbol{o}}_{1}/k_{v1}) \circ (\bar{\boldsymbol{p}}_{3}/k_{v3}) - (\bar{\boldsymbol{o}}_{2}/k_{v2}) \circ (\bar{\boldsymbol{p}}_{4}/k_{v4}).$$
(12)

For simplicity, we define another four vectors for  $\bar{o}$  and  $\bar{p}$  with two random numbers  $r_o, r_p \in \mathbb{R}^+$  as follows:

$$\begin{cases}
\bar{o}'_{1} = r_{o} \cdot \bar{o}_{1}/k_{v1}, & \bar{p}'_{1} = r_{p} \cdot \bar{p}_{1}/k_{v1} \\
\bar{o}'_{2} = r_{o} \cdot \bar{o}_{2}/k_{v2}, & \bar{p}'_{2} = r_{p} \cdot \bar{p}_{2}/k_{v2} \\
\bar{o}'_{3} = r_{o} \cdot \bar{o}_{3}/k_{v3}, & \bar{p}'_{3} = r_{p} \cdot \bar{p}_{3}/k_{v3} \\
\bar{o}'_{4} = r_{o} \cdot \bar{o}_{4}/k_{v4}, & \bar{p}'_{4} = r_{p} \cdot \bar{p}_{4}/k_{v4}.
\end{cases} (13)$$

Here, we introduce the two random numbers to add randomness for the sake of security. By combining Equation 12 and 13, we have the following equation:

$$\mathcal{F}_3(\bar{\boldsymbol{o}},\bar{\boldsymbol{p}}) = r_{\boldsymbol{o}} r_{\boldsymbol{p}} \mathcal{F}_2(\bar{\boldsymbol{o}},\bar{\boldsymbol{p}}) = \bar{\boldsymbol{o}}_1' \circ \bar{\boldsymbol{p}}_3' - \bar{\boldsymbol{o}}_2' \circ \bar{\boldsymbol{p}}_4'. \tag{14}$$

Now, let us consider the second term in the second line of Equation 8, i.e.,  $M_3^{-1} \times [\bar{q}^T, -\bar{q}^T]^T$ . With another random number  $r_q \in \mathbb{R}^+$  and the two vectors  $k_{v2}, k_{v4}$ , we define the following vector for  $\bar{q}$ :

$$\bar{q}' = r_{q} \cdot M_{3}^{-1} \times [\bar{q}^{T}, -\bar{q}^{T}]^{T} \circ (k_{v2} \circ k_{v4}).$$
 (15)

Then, we combine the equations above and have the following equation and prove it in Theorem 3:

$$\mathcal{F}_3(\bar{\boldsymbol{o}}, \bar{\boldsymbol{p}})^T \times \bar{\boldsymbol{q}}' = 2r_{\boldsymbol{o}}r_{\boldsymbol{p}}r_{\boldsymbol{q}}(||\boldsymbol{o}||^2 - 2\boldsymbol{o}^T\boldsymbol{q} - ||\boldsymbol{p}||^2 + 2\boldsymbol{p}^T\boldsymbol{q}).$$
(16)

According to Equation 16, we can determine whether or not  $dist(\mathbf{o}, \mathbf{q}) < dist(\mathbf{p}, \mathbf{q})$ , by precomputing four vectors of each database vector  $\mathbf{p} \in P$ , i.e.,  $\bar{p}_1, \bar{p}_4, \bar{p}_3, \bar{p}_4$  as in Equation 13, and  $\bar{\mathbf{q}}'$  as in Equation 15 for a given query  $\mathbf{q}$ .

#### B. Our DCE Scheme

four key functions: (1) KeyGen generates secret keys SK, (2) Enc employs SK to encrypt each database vector, (3) TrapGen produces the trapdoor for each query vector with SK, and (4) DistanceComp conducts secure distance comparison. As follows, we introduce them in detail, respectively. (1) KeyGen $(1^{\zeta}, d) \rightarrow SK$ . Given a security parameter  $r_3, r_4, k_{v1}, k_{v2}, k_{v3}, k_{v4}$ . As presented in Section IV-A, the random invertible matrices  $M_1, M_2 \in \mathbb{R}^{(d/2+4)\times(d/2+4)}$ , the random permutation functions  $\pi_1:\mathbb{R}^d \to \mathbb{R}^d, \pi_2:\mathbb{R}^{(d+8)} \to$  $\mathbb{R}^{(d+8)}$  and the four random numbers  $r_1, r_2, r_3, r_4 \in \mathbb{R}$  are used in the phase of vector randomization. On the other hand, the random invertible matrix  $M_3 \in \mathbb{R}^{(2d+16)\times(2d+16)}$  and four random vectors  $k_{v1}, k_{v2}, k_{v3}, k_{v4} \in \mathbb{R}^{(2d+16)}$  are used for vector transformation. Note that  $M_3$  is divided into two submatrices that contain the first half rows and the second half rows, respectively. Moreover,  $k_{v1} \circ k_{v3} = k_{v2} \circ k_{v4}$ .

In this section, we elaborate on our DCE scheme. It contains

- (2)  $\operatorname{Enc}(\boldsymbol{p},SK) \to C^{\boldsymbol{p}}_{DCE}$ . Given SK and  $\boldsymbol{p} \in P$ , this function outputs the encrypted vector  $C^{\boldsymbol{p}}_{DCE}$ . As discussed in Section IV-A, it contains two phases. In the phase of vector randomization, according to steps 1-4, a new vector  $\bar{\boldsymbol{p}} \in \mathbb{R}^{d+8}$  is generated for  $\boldsymbol{p}$ . In the second phase of vector transformation that takes  $\bar{\boldsymbol{p}}$  as input, four vectors  $\bar{\boldsymbol{p}}'_1, \bar{\boldsymbol{p}}'_2, \bar{\boldsymbol{p}}'_3, \bar{\boldsymbol{p}}'_4 \in \mathbb{R}^{2d+16}$  are produced via the following two steps.
  - i **matrix encryption:** generate four vectors  $\bar{p}_1, \bar{p}_2, \bar{p}_3, \bar{p}_4$  via Equation 10.
  - ii **adding randomness:** generate four vectors  $\bar{p}'_1, \bar{p}'_2, \bar{p}'_3, \bar{p}'_4$  via Equation 13.

Then, we get the ciphertexts of  $\bar{p}$ , i.e.,  $C_{DCE}^{p}=(\bar{p}_1',\bar{p}_2',\bar{p}_3',\bar{p}_4')$ .

- (3) TrapGen $(q, SK) \to T_q$ : Given SK and a query vector  $q \in \mathbb{R}^d$ , it outputs the trapdoor  $T_q$  via the following two steps, which are illustrated in Figure 2.
  - i **vector randomization:** generate a new vector  $\bar{q} \in \mathbb{R}^{d+8}$  according to the four steps in the first phase in Section IV-A.
  - ii **vector transformation:** taking  $\bar{q}$  as input,  $\bar{q}' \in \mathbb{R}^{2d+16}$  is generated via Equation 15. Hence, we get  $T_q = \bar{q}'$ .
- (4) DistanceComp $(C_{DCE}^{\mathbf{o}}, C_{DCE}^{\mathbf{p}}, T_{\mathbf{q}}) \rightarrow [\bar{o}_{1}' \circ \bar{p}_{3}' \bar{o}_{2}' \circ \bar{p}_{4}']^{T} \bar{q}'$ . According to Equation 16, we have  $Z_{o,p,q} =$  DistanceComp $(C_{DCE}^{o}, C_{DCE}^{p}, T_{\mathbf{q}}) = 2r_{o}r_{p}r_{q}(||o||^{2}-2o^{T}q-||p||^{2}+2p^{T}q)$ . Since  $r_{o}, r_{p}, r_{q} \in \mathbb{R}^{+}$  are all positive numbers, the sign of  $Z_{o,p,q}$  exactly answers the distance comparison. To be specific, if  $Z_{o,p,q} < 0$ , we have dist(o,q) < dist(p,q). Otherwise,  $dist(o,q) \geq dist(p,q)$ .

To be formal, we prove the correctness of the function  $DistanceComp(\cdot)$  in the following theorem.

Theorem 3: Denote  $Z_{o,p,q}$  =DistanceComp $(C_{DCE}^o,C_{DCE}^p,T_q)$ , then we have

$$\begin{cases}
Z_{\boldsymbol{o},\boldsymbol{p},\boldsymbol{q}} < 0 & \Leftrightarrow & dist(\boldsymbol{o},\boldsymbol{q}) < dist(\boldsymbol{p},\boldsymbol{q}) \\
Z_{\boldsymbol{o},\boldsymbol{p},\boldsymbol{q}} \ge 0 & \Leftrightarrow & dist(\boldsymbol{o},\boldsymbol{q}) \ge dist(\boldsymbol{p},\boldsymbol{q}).
\end{cases} (17)$$

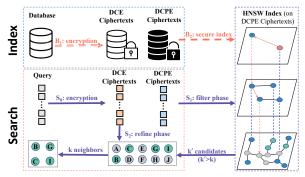


Fig. 3. The Overview of our PP-ANNS scheme.

*Proof:* DistanceComp $(C_{DCE}^{o}, C_{DCE}^{p}, T_{q})$ 

$$\begin{split} &= [\bar{\boldsymbol{o}}_{1}^{\prime} \circ \bar{\boldsymbol{p}}_{3}^{\prime} - \bar{\boldsymbol{o}}_{2}^{\prime} \circ \bar{\boldsymbol{p}}_{4}^{\prime}] \times \bar{\boldsymbol{q}}^{\prime} \qquad \text{(by definition)} \\ &= r_{o}r_{p}\mathcal{F}_{2}(\bar{\boldsymbol{o}},\bar{\boldsymbol{p}}) \times \bar{\boldsymbol{q}}^{\prime} \qquad \qquad (\because Eq. \ 14) \\ &= r_{o}r_{p}\mathcal{F}_{1}(\bar{\boldsymbol{o}},\bar{\boldsymbol{p}})/(k_{v1} \circ k_{v3}) \times \bar{\boldsymbol{q}}^{\prime} \qquad \qquad (\because Eq. \ 12) \\ &= 2r_{o}r_{p}r_{q}[\bar{\boldsymbol{o}}^{T},\bar{\boldsymbol{p}}^{T}] \times M_{3}/(k_{v1} \circ k_{v3}) \times \qquad \qquad (\because Eq. \ 12) \\ &= M_{3}^{-1} \times [\bar{\boldsymbol{q}}^{T}, -\bar{\boldsymbol{q}}^{T}]^{T} \circ (k_{v2} \circ k_{v4}) \qquad \qquad (\because Eq. \ 15) \\ &= 2r_{o}r_{p}r_{q}[\bar{\boldsymbol{o}}^{T},\bar{\boldsymbol{p}}^{T}] \times [\bar{\boldsymbol{q}}^{T}, -\bar{\boldsymbol{q}}^{T}]^{T} \ (\because k_{v1} \circ k_{v3} = k_{v2} \circ k_{v4}) \\ &= 2r_{o}r_{p}r_{q}(\bar{\boldsymbol{o}}^{T}\bar{\boldsymbol{q}} - \bar{\boldsymbol{p}}^{T}\bar{\boldsymbol{q}}) \\ &= 2r_{o}r_{p}r_{q}(||\boldsymbol{o}||^{2} - 2\boldsymbol{o}^{T}\boldsymbol{q} - ||\boldsymbol{p}||^{2} + 2\boldsymbol{p}^{T}\boldsymbol{q})) \\ &= 2r_{o}r_{p}r_{q}(dist(\boldsymbol{o},\boldsymbol{q}) - dist(\boldsymbol{p},\boldsymbol{q})). \end{split}$$

Since 
$$r_o, r_p, r_q \in \mathbb{R}^+$$
, we have  $Z_{o,p,q} < 0 \Leftrightarrow dist(o,q) < dist(p,q)$  and  $Z_{o,p,q} \geq 0 \Leftrightarrow dist(o,q) \geq dist(p,q)$ .

Theorem 3 proves that our DCE scheme precisely resolves the distance comparison. Besides, our DCE scheme efficiently handles the distance comparison, since each SDC computation in DCE only requires 4d+32 multiply-and-accumulate operations and a time cost of O(d). The dimension of each encrypted database vector is 8d+64, while that of each encrypted query vector is 2d+16. Furthermore, our DCE scheme is proved to be IND-KPA security in Section VI.

With our DCE scheme, we can securely conduct a k-NN search through a linear scan method utilizing a max heap H to retain the current top k results. Each database vector  $\mathbf{p} \in P$  is considered a candidate in the linear scan, assessing if it outperforms those in H at the complexity of  $O(d \times \log k)$ . Hence, employing the linear scan method with our DCE scheme incurs a cost of  $(n \times d \times \log k)$ , which can be prohibitive, particularly for large-scale datasets. As a result, in the next section, we consider designing an efficient index to diminish the count of SDC computations while compromising minimal accuracy.

# V. OUR PP-ANNS SCHEME

In this section, building upon the DCE scheme introduced earlier, we present our whole PP-ANNS scheme depicted in Figure 3. Our scheme comprises two components: **index** and **search**. We then delve into each of them in detail and analyze the computational and spatial costs of our PP-ANNS scheme.

```
Algorithm 1: Enc_{SAP}(s, \beta, \boldsymbol{p})
```

Input: secret keys  $(s, \beta)$ , a vector pOutput: the ciphertext of p  $C_{SAP}^{p}$ 1  $u \leftarrow \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ ;

2  $x' \leftarrow \mathcal{U}(0, 1)$ ;

3  $x \leftarrow \frac{s\beta}{4}(x')^{1/d}$ ;

4  $\lambda_p \leftarrow x \cdot \frac{u}{\|u\|}$ ;

5  $C_{SAP}^{p} \leftarrow s \cdot p + \lambda_p$ ;

6 return  $C_{SAP}^{p}$ ;

# A. Index Structure

To reduce the number of expensive SDC computations of DCE, we propose a privacy-preserving index based on two techniques, i.e., distance-comparison-preserving encryption (DCPE) [10] and HNSW [23]. The former is a privacy-preserving encryption technique for vectors, while the latter is the state-of-the-art method for *k*-ANNS on vectors.

**DCPE.** As mentioned in Section III-B, DCPE employs a  $\beta$ -DCP function to encrypt both database vectors and query vectors. Moreover, [10] proposes such an instance called Scaleand-Perturb (SAP). There are two secret keys: (1) the scaling factor  $s \in \mathbb{R}$  that is a random number, and (2) the permutation factor  $\beta \in [\sqrt{M}, 2M\sqrt{d}]$ , where  $M = \max_{p \in P} \max_{i=1}^{d} |p_i|$ . With those two secret keys, the SAP ciphertexts  $C_{SAP}^{p}$  for each  $p \in P$  is computed as in Algorithm 1. It first generates a random vector u drawn from the Guassian distribution  $\mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$  (Line 1), where  $\mathbf{0}_d$  is with all zeros and  $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ is the identity matrix, and a random number x' drawn from a uniform distribution  $\mathcal{U}(0,1)$  (Line 2). Then, we obtain xin Line 3, which will be the norm of the vector  $\lambda_p$  (Line 4). Finally, the ciphertext of p is the addition of two vectors, i.e., scaled p and a random vector  $\lambda_p$  that is randomly drawn in the ball  $B(\mathbf{0}_d, s\beta/4)$ . Here, we slightly modify the SAP function in Algorithm 1, which does not return the information for decryption. This is because we store  $\{C_{SAP}^{p}|p\in P\}$  on the server and we will not decrypt the ciphertext anywhere for security. Moreover, SAP uses  $dist(C_{SAP}^{p}, C_{SAP}^{q})$  as an approximation to  $dist(\mathbf{p}, \mathbf{q})$ .

Notably,  $\beta$  is key to the balance between the approximation of encrypted vectors and the data privacy, i.e., smaller  $\beta$  leads to a more accurate distance between encrypted vectors, but at the risk of revealing more distance information.

**HNSW.** Let  $C_{SAP}^P = \{C_{SAP}^p | p \in P\}$  denote the encrypted vectors of P. Note that  $C_{SAP}^p$  is still d-dimensional. The data owner then builds an HNSW [23] graph, one of the SOTA k-ANNS method [19], [31], to index  $C_{SAP}^P$ . As shown in Figure 3, HNSW contains multiple layers, where each layer has a graph structure that manages a subset of  $C_{SAP}^P$ . The higher layers contain fewer nodes than the lower layers and layer 0 contains the whole set  $C_{SAP}^P$ . HNSW is built from scratch and inserts each vector  $\boldsymbol{v} \in C_{SAP}^P$  into the graph one by one.

Notably, data owner does not build the HNSW graph on the plaintext database, since each edge in the graph reflects the

# Algorithm 2: Search $(T_q, C_{SAP}^q, C_{SAP}^P, C_{DCE}^P)$ **Input**: encrypted query vector: $T_q$ , $C_{SAP}^q$ , encrypted database: $C_{SAP}^{P}, C_{DCE}^{P}$ **Output:** the k-ANNS result // filter phase: 1 R' $\leftarrow$ the results of k'-ANNS on HNSW built on $C_{SAP}^{P}$ where the query is $C_{SAP}^{q}$ ; // refine phase: 2 Initialize a max heap $H = \emptyset$ ; 3 for each $p \in R'$ do if |H| < k then Insert p into H; 5 continue; $o \leftarrow H.top()$ ; 7 if $DistanceComp(C_{DCE}^{o}, C_{DCE}^{p}, T_{q}) > 0$ then Insert p into H; 10 **return** the elements in H;

neighboring relationship between the corresponding vectors. On the other hand, encrypted DCPE vectors are just the approximation of the original vectors and thus the edges of HNSW built on them do not reflect the exact neighborhood for each vector, which enhances the data privacy.

In summary, the index in our PP-ANNS scheme contains three parts, i.e.,  $C_{SAP}^P$ , the HNSW index built on  $C_{SAP}^P$ , and  $C_{DCE}^P = \{C_{DCE}^{\pmb{p}} | \pmb{p} \in P\}$ . Notably, our approach can leverage other proximity graph-based approaches for k-ANNS like the navigating spreading-out graph [9] and the  $\tau$ -monotonic neighborhood graph [26] to substitute HNSW for indexing the DCPE-encrypted vectors.

#### B. Search Method

With our index, we now propose a new search method for the k-ANNS as shown in Algorithm 2. Our search method follows the *filter-and-refine* strategy. In the *filter* phase (Line 1), we conduct k'-ANNS (k' > k) on HNSW built on  $C_{SAP}^P$ , which strictly follows the same procedure as the HNSW search method in [23] and thus returns a set R' of high-quality candidates. Note that we verify all the candidates by approximate distances of their SAP encrypted vectors to the encrypted vector  $C_{SAP}^q$  in this phase. In the *refine* phase (Lines 2-9), a max heap H is initialized as empty in Line 2, which is used to store the currently best k neighbors. Then, we try to insert each  $p \in R'$  into H.

Note that k' is key to the search performance of our method. A larger k' increases the probability of finding more accurate results in the *refine* phase, but at the expense of more SDC computations, each of which is  $O(d^2)$ . Hence, k' is key to the balance between the accuracy and efficiency of our method, which is investigated in the experiments of Section VII-A. In practice, we employ the grid search method to select the best value of k'.

Time Complexity. We use the DistanceComp function to compare two vectors and determine which one is closer to the query. Hence, each insertion in H may cause at most  $\log k$  distance comparisons, each of which costs O(d). As a result, the cost of the refine phase is  $O(k' \cdot d \cdot \log k)$ .

#### C. Cost Analysis

In this section, we analyze the cost complexity of our approach, including computational cost and space requirement. For the computational cost, we consider (1) the server-side computational cost, (2) the user-side computational cost, and (3) the communication overhead between the user and server. Server-Side Computational Cost. Search on server consists of two phases, i.e., (1) the *filter* phase that conducts k'-ANNS on HNSW and (2) the refine phase that finds the best k neighbors via SDC computations of DCE. According to HNSW [23], the complexity of the *filter* phase is  $O(d \cdot \log n)$ . In refine phase, we employ a max heap with at most kelements to obtain the best k neighbors from the k' candidates. When dealing with each candidate, it may update the heap with  $O(\log k)$  SDC computations of DCE, each of which costs O(d). Hence, the *refine* phase costs  $O(d \cdot k' \log k)$ . In total, the complexity of the search cost on the server is  $O(d \cdot (\log n + k' \log k))$  for each query.

User-Side Computational Cost. The user only generates  $T_q$  for each given q, which contains two phases, i.e., vector randomization and vector transformation. The main cost of the first phase comes from Equation 4, whose complexity is  $O(d^2)$ , while the cost of the second phase is  $O(d^2)$  according to Equation 15. Besides, we also compute the DCPE encrypted query vector  $C_{SAP}^q$ , whose cost is only O(d). As a result, the user-side time complexity is  $O(d^2)$ . Hence, the user only performs very slight computation.

**Communication Overhead.** During the search procedure, only two messages are needed to transfer between user and server, i.e., (1) user sends the encrypted query vectors (i.e.,  $C_{SAP}^{q}$  and  $C_{DCE}^{q}$ ) and the parameter k to the server, which occupies  $(36 \cdot d + 260)$  bytes in total, and (2) server returns k IDs of found neighbors to the user, which requires only  $4 \cdot k$  bytes. As a result, there exists extremely limited communication overhead between the user and the server.

**Space Complexity.** We analyze the space requirement of our method in the server. It contains three parts: (1) the DCPE ciphertexts  $C_{SAP}^P$ , (2) the HNSW index, and (3) the ciphertexts  $C_{DCE}^P$ . The space requirement of  $C_{SAP}^P$  is the same as P, while that of  $C_{DCE}^P$  is (8+64/d) times that of P. As to HNSW, its space requirement is  $O(n\cdot m)$ , where m is the upper bound of the out-degree for each node in HNSW. Hence, the space requirement is  $O(n\cdot (d+m))$ .

# D. Discussion on Index Maintenance

In real-world applications, the vector database needs to support updates, e.g., the addition of new vectors or removal of existing ones. Enhancing the index maintenance significantly increases the practical usability of our method. For insertion, let  $\boldsymbol{u}$  be the vector to be inserted. The data provider first

generates two encrypted vectors, i.e.,  $C^u_{SAP}$  and  $C^u_{DCE}$  and then sends them to the server. Like inserting a new point in the original HNSW, the server conducts k-ANNS for  $C^u_{SAP}$  and selects diverse neighbors from the returned neighbors, followed by building edges between u and selected neighbors. Hence, the HNSW graph is updated for inserting u.

As to deleting an existing vector  $\boldsymbol{u}$ , only its in-neighbors will be affected, while its out-neighbors will not. For each inneighbor  $\boldsymbol{v}$  of  $\boldsymbol{u}$ , we can reinsert it into HNSW, by finding its k-ANN in the current graph and then building edges between  $\boldsymbol{v}$  and a selected subset of its k-ANN returned. Besides, the encrypted vectors of  $\boldsymbol{u}$ , i.e.,  $C_{SAP}^{\boldsymbol{u}}$  and  $C_{DCE}^{\boldsymbol{u}}$ , will be deleted. Unlike the insertion that needs the participation of the data provider, the deletion could be finished solely by the server.

#### VI. SECURITY ANALYSIS

In this section, we analyze the security of our DCE scheme and PP-ANNS scheme, respectively.

## A. Security of DCE Scheme

Our DCE scheme is an encryption that supports searchability. Like AME [44], we prove that our DCE scheme is IND-KPA secure in the real/ideal world security model. The real world is our proposal and the ideal world is an ideal function that only leaks some information specified by a leakage function, which is only related to some public information in our proposal. We prove that the real world is indistinguishable from the ideal world. To ensure fairness, we follow the proof steps in [44]. We first define the leakage function to introduce the real and ideal experiments.

Let  $o, p \in P$  be any two database vectors, and q be an arbitrary query vector. For simplicity, let  $C_o$ ,  $C_p$  and  $T_q$  be their ciphertexts in our DCE scheme, respectively. Then, the leakage of our DCE scheme is the comparison result between dist(o,q) and dist(p,q), i.e.,  $\mathcal{L}(o,p,q) = \texttt{DistanceComp}(C_o, C_p, T_q)$ . Based on this leakage function, we define the real and ideal experiments in the following. **Real experiment**. The real experiment involves two participants, i.e., a probabilistic polynomial time adversary  $\mathcal{A}$  and a challenger, which interact with each other as follows.

**Setup.** In the setup phase,  $\mathcal{A}$  first chooses a set  $P_1$  that contains  $n_1$  d-dimensional database vectors and sends them to the challenger. On receiving  $P_1$ , the challenger calls  $\operatorname{KeyGen}(1^\zeta,d)$  to generate a secret key  $SK = \{M_1,M_2,M_3,\pi_1,\pi_2,r_1,r_2,r_3,r_4,k_{v1},k_{v2},k_{v3},k_{v4}\}$ . Then the challenger calls  $\operatorname{Enc}(\boldsymbol{p},SK)$  to encrypt vectors  $\boldsymbol{p} \in P_1$ .

Query phase 1. In this phase,  $\mathcal{A}$  needs to choose a set  $Q_1$  with  $m_1$  d-dimensional query vectors, where  $m_1$  is a polynomial number. Then,  $\mathcal{A}$  sends  $Q_1$  to the challenger. On receiving them, the challenger calls  $\mathsf{TrapGen}(q,SK)$  to encrypt q into  $T_q$  for each query  $q \in Q_1$ . Then, the challenger returns  $\{T_q | q \in Q_1\}$  to  $\mathcal{A}$ .

**Challenge phase.** In the challenge phase, the challenger returns  $\{C_{p}|p\in P_{1}\}$  to  $\mathcal{A}$ .

**Query phase 2.** In this phase,  $\mathcal{A}$  needs to choose another set  $Q_2$  with  $m_2$  d-dimensional query vectors and gets the ciphertexts  $\{T_q | q \in Q_2\}$  from the challenger in the same way as Query phase 1.

**Ideal experiment**. The ideal experiment involves two participants, i.e., a probabilistic polynomial time adversary A and a simulator with the leakage  $\mathcal{L}$ , interacting as follows.

**Setup.** A chooses  $n_1$  d-dimensional database vector set  $P_1$  and sends them to the simulator. On receiving them, the simulator generates a  $(2d+16)\times(2d+16)$  matrix  $M_{sim}$  and a collection of random vectors  $C_{\boldsymbol{p},sim}=(\bar{\boldsymbol{p}}'_{1,sim},\bar{\boldsymbol{p}}'_{2,sim},\bar{\boldsymbol{p}}'_{3,sim},\bar{\boldsymbol{p}}'_{4,sim})$  as ciphertexts for each  $\boldsymbol{p}\in P_1$ , which satisfy  $\bar{\boldsymbol{p}}'_{1,sim}\circ\bar{\boldsymbol{p}}'_{3,sim}-\bar{\boldsymbol{p}}'_{2,sim}\circ\bar{\boldsymbol{p}}'_{4,sim}=M_{sim}^{-1}\times[\boldsymbol{v}_r^T,-\boldsymbol{v}_r^T]^T$ . And  $\boldsymbol{v}_r$  is a (d+8)-dimensional random vector.

Query phase 1.  $\mathcal{A}$  chooses a query set  $Q_1$  with  $m_1$  query vectors, where  $m_1$  is a polynomial number. Then,  $\mathcal{A}$  sends  $Q_1$  to the simulator. Upon receiving these query vectors, the simulator uses the leakage function  $\mathcal{L}$  and  $\{C_{\boldsymbol{p},sim}|\boldsymbol{p}\in P_1\}$  to generate the ciphertexts for  $Q_1$ . For each  $\boldsymbol{q}\in Q_1$ , the simulator generates a ciphertext  $T_{\boldsymbol{q},sim}$  as follows.

• First, the simulator generates two random  $n_1 \times n_1$  matrices  $R_q$  and  $R'_q$ , where  $r_{i,j} \in R_q$  and  $r'_{i,j} \in R'_q$  are random numbers and satisfy that for  $1 \le i, j \le n_1$ ,

$$\begin{cases} r_{i,j} > 0, r'_{i,j} > 0 & \mathcal{L}(\boldsymbol{o}, \boldsymbol{p}, \boldsymbol{q}) > 0 \\ r_{i,j} \leq 0, r'_{i,j} \leq 0 & \mathcal{L}(\boldsymbol{o}, \boldsymbol{p}, \boldsymbol{q}) \leq 0 \end{cases}$$

Here, o, p are the *i*-th and *j*-th point in  $P_1$ , respectively.

• Second, the simulator randomly chooses a vector  $T_{q,sim}$  such that  $T_{q,sim} = M_{sim}^T \times [v_{rq}, -v_{rq}],$ 

$$\begin{aligned} & \left( \bar{\boldsymbol{o}}_{1,sim}' \circ \bar{\boldsymbol{p}}_{3,sim}' - \bar{\boldsymbol{o}}_{2,sim}' \circ \bar{\boldsymbol{p}}_{4,sim}' \right)^T \times T_{\boldsymbol{q},sim} = r_{i,j}, \\ & \left( \bar{\boldsymbol{p}}_{1,sim}' \circ \bar{\boldsymbol{o}}_{3,sim}' - \bar{\boldsymbol{p}}_{2,sim}' \circ \bar{\boldsymbol{o}}_{4,sim}' \right)^T \times T_{\boldsymbol{q},sim} = r_{i,j}'. \end{aligned}$$

The  $v_{rq}$  is a random (d+8)-dimensional vector. The simulator returns  $\{T_{q,sim}|q\in Q_1\}$  to  $\mathcal{A}$ .

**Challenge phase.** In the challenge phase, the simulator returns  $\{C_{p,sim}|p \in P_1\}$  to  $\mathcal{A}$ .

**Query phase 2.** A chooses a query set  $Q_2$  with  $m_2$  d-dimensional query vectors and gets their ciphertexts in the same way as Query phase 1.

In the real experiment, the view of  $\mathcal{A}$  is  $\mathrm{View}_{\mathcal{A},\mathrm{Real}} = \{\{C_{\boldsymbol{p}}|\boldsymbol{p}\in P_1\}, \{T_{\boldsymbol{q}}|\boldsymbol{q}\in Q_1\cup Q_2\}\}$ . In the ideal experiment, the view of  $\mathcal{A}$  is  $\mathrm{View}_{\mathcal{A},\mathrm{Ideal}} = \{\{C_{\boldsymbol{p},sim}|\boldsymbol{p}\in P_1\}, \{T_{\boldsymbol{q},sim}|\boldsymbol{q}\in Q_1\cup Q_2\}$ . Then, based on the views of  $\mathcal{A}$  in the real and ideal experiments, we define the security of the DCE scheme.

In the following, we prove that the DCE scheme is IND-KPA secure with leakage  $\mathcal{L}$ .

Theorem 4: The DCE scheme is IND-KPA secure with  $\mathcal{L}$ . Proof: We prove the security of our DCE scheme by proving that  $\mathcal{A}$  cannot distinguish the views of real and ideal experiments. Since all ciphertexts  $\{C_{\boldsymbol{p},sim}|\boldsymbol{p}\in P_1\}$  and  $\{T_{\boldsymbol{q},sim}|\boldsymbol{q}\in Q_1\cup Q_2\}$  in the ideal experiment are randomly generated by the simulator, distinguishing the real view and the ideal view is equivalent to distinguish View $_{\mathcal{A},\mathrm{Real}}$ 

from random ciphertexts. To prove the indistinguishability, we consider three cases:

- Case 1: The ciphertexts  $\{C_{p}|p\in P_{1}\}$  are indistinguishable from random ciphertexts;
- Case 2: The ciphertexts  $\{T_q | q \in Q_1 \cup Q_2\}$  are indistinguishable from random ciphertexts;
- Case 3: Each intermediate result Z<sub>o,p,q</sub> (as defined in Theorm 3) is indistinguishable from random numbers, where o, p ∈ P<sub>1</sub>.

Due to the definition of View\_A,Real, we naturally need to consider the indistinguishability of Case 1 and Case 2. Besides, we consider the indistinguishability between the intermediate results and random ciphertexts. In our DCE scheme,  $\{Z_{o,p,q}|o,p\in P_1\}$  are the intermediate results with the least number of unknown variables. This is because  $Z_{o,p,q}=2r_or_pr_q(dist(o,q)-dist(p,q))$  as proved in Theorem 3. Obviously, it has eliminated the unknown matrices in the secret keys. Meanwhile, it is easy to verify that other intermediate results contain more unknown variables than  $\{Z_{o,p,q}|o,p\in P_1\}$ . Thus, if  $\{Z_{o,p,q}|o,p\in P_1\}$  are indistinguishable from random ciphertexts, all other intermediate results are indistinguishable from random ciphertexts. Thus, we consider Case 1, Case 2, and Case 3 in the indistinguishability proof, as described below.

The ciphertexts  $\{C_{\boldsymbol{p}}|\boldsymbol{p}\in P_1\}$  are indistinguishable from random ciphertexts. In our scheme,  $C_{\boldsymbol{p}}=(\bar{\boldsymbol{p}}_1',\bar{\boldsymbol{p}}_2',\bar{\boldsymbol{p}}_3',\bar{\boldsymbol{p}}_4')$ . According to Equation 13, 10 and 4, we have

$$\begin{cases}
\bar{\mathbf{p}}'_{1} = r_{\mathbf{p}}(\pi_{2}([\hat{p}_{1}^{T}, \hat{p}_{2}^{T}]^{T}) \times M_{up} + \mathbf{1}_{2d+16})/k_{v1} \\
\bar{\mathbf{p}}'_{2} = r_{\mathbf{p}}(\pi_{2}([\hat{p}_{1}^{T}, \hat{p}_{2}^{T}]^{T}) \times M_{down} + \mathbf{1}_{2d+16})/k_{v2} \\
\bar{\mathbf{p}}'_{3} = r_{\mathbf{p}}(\pi_{2}([\hat{p}_{1}^{T}, \hat{p}_{2}^{T}]^{T}) \times M_{up} - \mathbf{1}_{2d+16})/k_{v3} \\
\bar{\mathbf{p}}'_{4} = r_{\mathbf{p}}(\pi_{2}([\hat{p}_{1}^{T}, \hat{p}_{2}^{T}]^{T}) \times M_{down} - \mathbf{1}_{2d+16})/k_{v4}.
\end{cases} (18)$$

Here,  $\hat{p}_1$  and  $\hat{p}_1$  could be derived via Equations 2 and 1.

To generate  $C_p$ , we add many random numbers such as  $r_p, \alpha_{p,1}, \alpha_{p,2}, r'_{p,1}, r'_{p,2}$  and  $r'_{p,3}$ . Then, we add multiply those random numbers by the vector, which makes the results indistinguishable. Besides, we use two random permutations (i.e.,  $\pi_1, \pi_2$ ) and four random vectors (i.e.,  $k_{v1}, k_{v2}, k_{v3}, k_{v4}$ ) to perturb the specific values in  $C_p$ . Thus, these random numbers and the random permutations can guarantee that  $\{C_p | p \in P_1\}$  are indistinguishable from random ciphertexts.

The ciphertexts  $\{T_{\boldsymbol{q}}|\boldsymbol{q}\in Q_1\cup Q_2\}\}$  are indistinguishable from random ciphertexts. In our scheme,  $T_{\boldsymbol{q}}=r_{q}M_3^{-1}\times [\bar{\boldsymbol{q}}^T,-\bar{\boldsymbol{q}}^T]^T\circ (k_{v2}\circ k_{v4})$  according to Equation 15, where  $\bar{\boldsymbol{q}}$  could be derived via Equation 4, 3 and 1.

To compute  $T_q$ , we introduce many random numbers such as  $r_q$ ,  $\beta_{q,1}$  and  $\beta_{q,2}$ . Besides, we use two random permutations (i.e.,  $\pi_1$  and  $\pi_2$ ) and two random vectors (i.e.,  $k_{v2}$  and  $k_{v4}$ ) to perturb the specific values in  $T_q$ , which make each  $T_q$  for  $q \in Q_1 \cup Q_2$  indistinguishable from random ciphertexts.

The intermediate results  $\{Z_{o,p,q}|o,p\in P_1\}$  are indistinguishable from random numbers. As defined in Theorm 3,  $Z_{o,p,q}=2r_or_pr_q(dist(o,q)-dist(p,q))$ . We can see that there still exist three random numbers, i.e.,  $r_o,r_p,r_q$ , which makes  $Z_{o,p,q}$  indistinguishable from random ciphertexts.

TABLE I STATISTICS OF DATASETS

DIMINITIES OF DIMINISTRE					
Datasets	Sift1M	Gist	Glove	Deep1M	
#dimensions	128	960	100	96	
#vectors	1,000,000	1,000,000	1,183,514	1,000,000	
#queries	10,000	1,000	10,000	10,000	

In conclusion, we can deduce that  $View_{\mathcal{A},Real}$  is indistinguishable from random ciphertexts. That is,  $View_{\mathcal{A},Real}$  is indistinguishable from  $View_{\mathcal{A},Ideal}$ , and  $\mathcal{A}$  cannot distinguish the views of real and ideal experiments. Therefore, our DCE scheme is IND-KPA secure with the leakage  $\mathcal{L}$ .

#### B. Security of Our PP-ANNS Scheme

In this part, we analyze the security of our PP-ANNS scheme. Since we focus on privacy-preserving properties, we show that both database and query requests are privacy-preserving as follows.

**Database is privacy-preserving.** In our scheme, the cloud server is semi-honest, so it is curious about the plaintexts of the database vectors. However, all database vectors in the cloud server have been encrypted by DCE and DCPE. The security of DCPE scheme has been proven in [10], while we prove the security of DCE scheme in the last section. Moreover, there is no relation between DCE and DCPE schemes, and the cloud server cannot recover plaintexts of database vectors even after obtaining both of them. As a result, nothing is revealed except for the HNSW index. Notably, it contains only approximate neighboring relationships between encrypted vectors.

The query vector is privacy-preserving. The query vector should be kept secret from the cloud server. Each query vector q has been encrypted by both DCE and DCPE schemes before sending to the cloud, and the security of DCE and DCPE can guarantee that the cloud server cannot recover the plaintext of q from  $T_q$ . Thus, each query vector is privacy-preserving.

#### VII. EXPERIMENTS

In this section, we evaluate the performance of our PP-ANNS method and compare it with other baseline methods to demonstrate its superiority. First of all, we introduce the experimental settings.

**Datasets.** We conduct experiments on four widely-used datasets, i.e.,  $SiftlM^1$ ,  $Gist^1$ ,  $Glove^2$  and  $DeeplM^3$ . The statistics of those datasets have been presented in Table I. In addition, we conducted experiments on random samples of the large-scale data  $SiftlB^1$  with 1 billion SIFT vectors and  $DeeplB^4$ , in order to verify the scalability of our method.

**Performance Metrics.** Our solution is mainly performed on the server, so we focus on the server-side search performance in both efficiency and accuracy. Efficiency is measured by queries per second (QPS), while accuracy is estimated by recall denoted as Recall@k. Given a query q, let  $N^*(q)$  be the exact k-nearest neighbors of q, while N(q) be the

<sup>1</sup>http://corpus-texmex.irisa.fr/

<sup>&</sup>lt;sup>2</sup>https://nlp.stanford.edu/projects/glove/

<sup>&</sup>lt;sup>3</sup>http://sites.skoltech.ru/compvision/noimi/

<sup>4</sup>https://disk.yandex.ru/d/11eDCm7Dsn9GA/

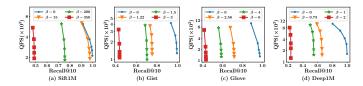


Fig. 4. The effects of  $\beta$  on the search performance in the *filter* phase. Note that  $\beta = 0$  indicates that there exists no noise in the DCPE encrypted vectors.

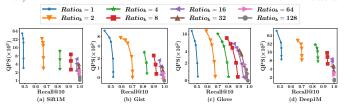


Fig. 5. The effects of  $Ratio_k$  on the search performance of our solution, where  $k'=Ratio_k\cdot k$ .

set of k approximate neighbors. We have  $Recall@k(q) = |N^*(q) \cap N(q)|/k$ , and use the average recalls over the queries to report Recall@k. We set the number k to 10 by default unless specified setting. All results are averaged over 5 runs. **Compared Methods.** We use the following methods as baselines for comparisons in the experiments.

- **RS-SANN** [25]. It encrypts the database via Advanced Encryption Standard (AES) and employs locality-sensitive hashing (LSH) as the index. It requires the user to decrypt selected encrypted candidates from the server and then select the best *k* neighbors among them.
- PACM-ANN [45]. It uses proximity graphs as the index and employs private information retrieval to obtain graph edges and encrypted vectors from the server in a multiround manner. The search process is controlled by user.
- PRI-ANN [27]. It uses LSH as the index and employs
  private information retrieval to obtain the candidates from
  the server. The search process consists of a single round
  of communication between the client and the two servers.
  And no communication is required between the servers.

Besides, we consider a variant of our method denoted **HNSW-AME** as a baseline. Like our method, HNSW-AME builds an HNSW graph on the encrypted database  $C^P_{DCPE}$ , but stores AME [44] instead of DCE. During the search process, HNSW-AME shares the same *filter* phase but conducts the SDC computations via AME in the *refine* phase.

**Computing Environment.** All experiments are conducted on a server equipped with two Intel(R) Xeon(R) Silver 4210R CPUs, each of which has 10 cores, and 256 GB RAM as its main memory. The OS version is CentOS 7.9.2009. All codes are implemented in C++, and the search performance is conducted with a single thread.

# A. Effects of Key Parameters

In this part, we study the effects of the key parameters of our method on the search performance. First, there are key parameters of building HNSW, i.e., m and efConstruction. We set efConstruction = 600 and m = 40 to build HNSW for each dataset, which is selected by a grid search method.

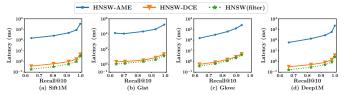


Fig. 6. Comparing HNSW-AME with our method denoted as HNSW-DCE. HNSW(filter) indicates our method with only the *filter* phase.

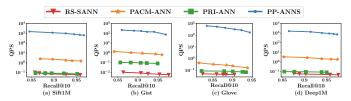
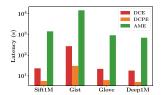


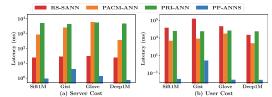
Fig. 7. Comparing our method with the baseline methods.

Besides, we vary the search parameter ef Search of HNSW in order to show the balance between efficiency and accuracy. Effects of DCPE Parameters. According to [10], two key parameters in DCPE are s and  $\beta$ . We set s = 1024 as recommended in [4].  $\beta$  should be in the range of  $\sqrt{M}$  and  $2M\sqrt{d}$ , where M indicates the maximum absolute coordinates. Hence, we need to tune  $\beta$ . A larger  $\beta$  indicates more noise added and the encrypted vector is further from the plaintext one, which enhances the privacy but degrades the quality of candidates returned in the *filter* phase. Hence,  $\beta$  should be carefully tuned. We show the influence of  $\beta$  on the search performance of our scheme with only filter phase in Figure 4, where k' = k. As  $\beta$  increases, the upper bound of recall decreases due to more noise added by DCPE. In order to balance data privacy and accuracy, we choose the best  $\beta$  for each dataset so that the upper bound of recall in *filter* phase is around 0.5. In this case, the attacker's probability of guessing the true neighbor correctly is only 50%. Specifically, the best  $\beta$  is set as 450, 2.5, 5, and 1.1 for Sift1M, Gist, Glove, and Deep1M, respectively. **Effects of** k'. The number k', indicating the number of returned neighbors in filter phase, is a key parameter in our scheme. For simplicity, we use the ratio  $Ratio_k = k'/k$  to imply k'. As shown in Figure 5, we can see that as  $Ratio_k$ rises, the upper bound of Recall@k grows but the efficiency decreases. This is because a larger  $Ratio_k$  indicates more neighbors verified in refine phase, which increases the chance of finding better neighbors but at the expense of more distance comparisons. In the rest of the experiments, we vary  $Ratio_k$ for best performance under different Recall@k.

# B. Comparisons with Baseline Methods

In this part, we compare four baseline methods: HNSW-AME, RS-SANN [25], PACM-ANN [45], and PRI-ANN [27]. First, we compare HNSW-AME with our method. HNSW-AME has the same parameter settings as ours. Besides, we compare with HNSW(filter) which only has the *filter* phase of ours. We show the results in Figure 6, where our method is denoted as HNSW-DCE. We can see that HNSW-DCE significantly outperforms HNSW-AME in efficiency, and achieves





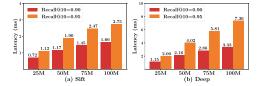


Fig. 8. Comparisons in vector encryption cost.

Fig. 9. Comparisons in server-side and user-side costs when Recall@k=0.9 (user cost is simulated in the server).

Fig. 10. Scalability study of our PP-ANNS scheme.

at least  $100\times$  speedup over HNSW-AME. Note that both of them share the same *filter* phase but distinct *refine* phase with different SDC methods. Notably, each SDC of DCE costs O(d), while that of AME costs  $O(d^2)$ . On the other hand, the latency of HNSW-DCE is pretty close to that of HNSW(filter) for the same accuracy, which indicates that the *refine* phase of our method is very efficient.

Notably, the data provider needs to encrypt the plaintext vectors by DCPE, DCE or AME, and thus consumes significant cost. Here, we compare the encryption cost of the vectors by different encryption methods in Figure 8. We can see that AME consumes considerably more cost than DCE, while DCPE costs the least due to its simple encryption method.

Moreover, we compare our method with the other three baseline methods, as shown in Figure 7 and Figure 9. We can see that our method significantly outperforms its competitors in search performance, and has considerably less cost on both the server side and user side. RS-SANN follows filter-andrefine strategy and conducts filter phase in the server, but performs the refine phase on the user side, which leads to numerous user-side costs. Besides, it uses LSH as the index and has to retrieve many more candidates to reach the same accuracy as ours. Like ours, PACM-ANN employs a proximity graph as the index but conducts the search process on the user side, which interactively fetches index data and database vectors from the server. Hence, it suffers from heavy computational costs on the user side and communication overhead. Like RS-SANN, PRI-ANN employs LSH as the index, which leads to numerous candidates for high accuracy. It incurs heavy computational consumption for servers and users. In addition, we conduct experiments for k-ANNS with HNSW on plaintext vectors. Compared with it, our PP-ANNS scheme consumes  $5\times$ ,  $7\times$ ,  $3\times$ , and  $4\times$  costs when Recall@k = 0.9.

#### C. Scalability of Our Method

In this part, we study the scalability of our PP-ANNS scheme. We conduct experiments on four random samples of Sift1B and Deep1B with various sizes, including 25 million, 50 million, 75 million, and 100 million vectors respectively. The experimental results are shown in Figure 10. We can see that the search performance of our method scales well as the data size increases. To be specific, when achieving the same accuracy, the latency of each query sublinearly grows as the data size increases.

#### VIII. RELATED WORKS

There are two problems related to our work: k-ANNS and PP-ANNS on plaintext and encrypted databases.

k-ANNS Methods in Plaintext Database. There is a bulk of k-ANNS methods in the literature, which could be divided into index-based methods and embedding-based methods. The former employs an index structure, such as locality-sensitive hashing [21], [22], inverted files [13], [41], and proximity graphs [9], [23], [33], [35], [38], [40] to accelerate the search process. On the other hand, the latter embeds vectors into other embeddings such as production quantization codes [13] and hashing codes [30], and replaces the expensive distance by a fast and approximate distance.

**PP-ANNS Schemes.** A number of PP-ANNS methods have been proposed in the last decades. Those methods could be divided into distance incomparable encryption and distance comparable encryption according to the encryption methods used. The former [7], [25], [27], [45] has to receive encrypted candidate vectors retrieved from the server and then decrypt them for distance comparisons on the user side. Such methods cause heavy user-side involvement and considerable communication overhead. On the other hand, the latter employ secure distance comparison methods such as asymmetric scalar-product-preserving encryption [6], [24], [29], [32] distance-comparison-preserving encryption [10], asymmetric matrix encryption [44] and homomorphic encryption [11], [34].

#### IX. CONCLUSION

In this paper, we study the privacy-preserving *k*-approximate nearest neighbor search (PP-ANNS) problem and design a new scheme that achieves data privacy, high efficiency, high accuracy, and minimal user-side involvement simultaneously. Our scheme conducts the search process in the server upon receiving an encrypted query vector, in order to minimize the user-side involvement. To protect data privacy, we propose a novel encryption method called distance comparison encryption for secure, exact, and efficient distance comparisons. Moreover, we propose a privacy-preserving index by combining HNSW and DCPE, so as to balance data privacy and search performance. Moreover, we theoretically analyze the security of our method and conduct extensive experiments to demonstrate its superiority over existing methods.

#### ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (No. 62002274, 62272369, 62372352) and Research Grants Council of Hong Kong (No.14205520).

#### REFERENCES

- Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. Modern Information Retrieval. ACM Press / Addison-Wesley, 1999.
- [2] Donald G. Bailey. An efficient euclidean distance transform. In *IWCIA*, volume 3322 of *Lecture Notes in Computer Science*, pages 394–408. Springer, 2004.
- [3] O Barndorff-Nielsen, Preben Blaesild, J Ledet Jensen, and B Jørgensen. Exponential transformation models. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 379(1776):41–65, 1982.
- [4] Dmytro Bogatov. Secure and efficient query processing in outsourced databases. PhD thesis, Boston University, 2022.
- [5] Valerio Cambareri, Mauro Mangia, Fabio Pareschi, Riccardo Rovatti, and Gianluca Setti. On known-plaintext attacks to a compressed sensingbased encryption: A quantitative analysis. *IEEE TIFS*, 10(10):2182– 2195, 2015.
- [6] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE TPDS, 25(1):222–233, 2014.
- [7] Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya P. Razenshteyn, and M. Sadegh Riazi. SANNS: scaling up secure approximate k-nearest neighbors search. In *USENIX Security*, pages 2111–2128, 2020.
- [8] Charles G Cullen. Matrices and linear transformations. Courier Corporation, 2012.
- [9] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. PVLDB, 12(5):461–474, 2019.
- [10] Georg Fuchsbauer, Riddhi Ghosal, Nathan Hauke, and Adam O'Neill. Approximate distance-comparison-preserving symmetric encryption. In SCN 2022, volume 13409 of Lecture Notes in Computer Science, pages 117–144. Springer, 2022.
- [11] Teddy Furon, Hervé Jégou, Laurent Amsaleg, and Benjamin Mathon. Fast and secure similarity search in high dimensional space. In WIFS 2013, pages 73–78. IEEE, 2013.
- [12] Yunguo Guan, Rongxing Lu, Yandong Zheng, Jun Shao, and Guiyi Wei. Toward oblivious location-based k-nearest neighbor query in smart cities. *IEEE Internet Things J.*, 8(18):14219–14231, 2021.
- [13] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1):117–128, 2011.
- [14] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.
- [15] Oliver N Keene. The log transformation is special. Statistics in Medicine, 14(8):811–819, 1995.
- [16] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In SCN 2018, volume 11035 of Lecture Notes in Computer Science, pages 544–562. Springer, 2018.
- [17] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledgeintensive NLP tasks. *NeurIPS*, 33:9459–9474, 2020.
- [18] Rui Li, Alex X. Liu, Ying Liu, Huanle Xu, and Huaqiang Yuan. Insecurity and hardness of nearest neighbor queries over encrypted data. In *ICDE* 2019, pages 1614–1617. IEEE, 2019.
- [19] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data experiments, analyses, and improvement. *IEEE TKDE*, 32(8):1475–1488, 2019.
- [20] Weipeng Lin, Ke Wang, Zhilin Zhang, and Hong Chen. Revisiting security risks of asymmetric scalar product preserving encryption and its variants. In *ICDCS*, pages 1116–1125. IEEE, 2017.
- [21] Yingfan Liu, Jiangtao Cui, Zi Huang, Hui Li, and Heng Tao Shen. Sklsh: an efficient index structure for approximate nearest neighbor search. PVLDB, 7(9):745–756, 2014.
- [22] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In VLDB, pages 950–961, 2007.
- [23] Yury A. Malkov and Dmitry A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE TPAMI*, 42(4):824–836, 2020.

- [24] Yinbin Miao, Yutao Yang, Xinghua Li, Linfeng Wei, Zhiquan Liu, and Robert H Deng. Efficient privacy-preserving spatial data query in cloud computing. *IEEE TKDE*, 2023.
- [25] Yanguo Peng, Jiangtao Cui, Hui Li, and Jianfeng Ma. A reusable and single-interactive model for secure approximate k-nearest neighbor query in cloud. *Information Sciences*, 387:146–164, 2017.
- [26] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. Efficient approximate nearest neighbor search in multi-dimensional databases. *Proc. ACM Manag. Data*, 1(1):54:1–54:27, 2023.
- [27] Sacha Servan-Schreiber, Simon Langowski, and Srinivas Devadas. Private approximate nearest neighbor search with sublinear communication. In S&P 2022, pages 911–929. IEEE, 2022.
- [28] Gurpreet Singh. A study of encryption algorithms (rsa, des, 3des and aes) for information security. *International Journal of Computer Applications*, 67(19), 2013.
- [29] Bing Wang, Shucheng Yu, Wenjing Lou, and Y. Thomas Hou. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *INFOCOM* 2014, pages 2112–2120. IEEE, 2014.
- [30] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE TPAMI*, 40(4):769–790, 2017.
- [31] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. PVLDB, 14(11):1964–1978, 2021.
- [32] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. Secure knn computation on encrypted databases. In SIGMOD, pages 139–152, 2009.
- [33] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. Fast approximate similarity join in vector databases. In SIGMOD. ACM, 2025.
- [34] Wenzhuo Xue, Hui Li, Yanguo Peng, Jiangtao Cui, and Yu Shi. Secure k nearest neighbors query for high-dimensional vectors in outsourced environments. *IEEE Transactions on Big Data*, 4(4):586–599, 2018.
- [35] Shuo Yang, Jiadong Xie, Yingfan Liu, Jeffrey Xu Yu, Xiyue Gao, Qianru Wang, Yanguo Peng, and Jiangtao Cui. Revisiting the index construction of proximity graph-based approximate nearest neighbor search. PVLDB, 2025
- [36] Bin Yao, Feifei Li, and Xiaokui Xiao. Secure nearest neighbor revisited. In ICDE, pages 733–744. IEEE, 2013.
- [37] Jiawei Yuan and Yifan Tian. Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset. *IEEE Transactions* on Cloud Computing, 7(2):568–579, 2019.
- [38] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. VBASE: unifying online vector similarity search and relational queries via relaxed monotonicity. In OSDI, pages 377–395. USENIX Association, 2023.
- [39] Zhilin Zhang, Ke Wang, Chen Lin, and Weipeng Lin. Secure top-k inner product retrieval. In CIKM 2018, pages 77–86. ACM, 2018.
- [40] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. Towards efficient index construction and approximate nearest neighbor search in high-dimensional spaces. PVLDB, 16(8):1979–1991, 2023.
- [41] Bolong Zheng, Ziyang Yue, Qi Hu, Xiaomeng Yi, Xiaofan Luan, Charles Xie, Xiaofang Zhou, and Christian S Jensen. Learned probing cardinality estimation for high-dimensional approximate nn search. In *ICDE*, pages 3209–3221. IEEE, 2023.
- [42] Yandong Zheng and Rongxing Lu. An efficient and privacy-preserving k-nn query scheme for ehealthcare data. In *Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 358–365. IEEE, 2018.
- [43] Yandong Zheng, Rongxing Lu, and Jun Shao. Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data. *Journal* of Medical Systems, 43:1–13, 2019.
- [44] Yandong Zheng, Rongxing Lu, Songnian Zhang, Jun Shao, and Hui Zhu. Achieving practical and privacy-preserving knn query over encrypted data. *IEEE TDSC*, 2024.
- [45] Mingxun Zhou, Elaine Shi, and Giulia Fanti. Pacmann: Efficient private approximate nearest neighbor search. IACR Cryptol. ePrint Arch., page 1600, 2024.
- [46] Youwen Zhu, Rui Xu, and Tsuyoshi Takagi. Secure k-nn computation on encrypted cloud data without sharing key with query users. In SCC@ASIACCS, pages 55–60. ACM, 2013.