Secure KNN Computation On Cloud

Tikaram Sanyashi[1][0000−0002−7434−0468], Nirmal Kumar Boran[2][0000−0003−3942−7899], and Virendra Singh[3][0000−0002−7035−7844]

[1] Department of Information Security and Communication Technology
NTNU, Trondheim, Norway
{tikaram.sanyashi}@ntnu.no
[2] Department of Computer Science & Engineering
NIT Calicut, Kozhikode, India
{nirmalkboran}@nitc.ac.in
[3] Department of Computer Science & Engineering
IIT Bombay, Mumbai, India
{viren}@cse.iitb.ac.in

Abstract. Cloud computing has emerged as a trend of outsourcing database and query services to a powerful cloud to ease local storage and computing pressure. However, private data storage and computation in the cloud come with a risk of losing the privacy and confidentiality of the data. Thus, sensitive applications running on the cloud require to be encrypted before storing it in the cloud. Additionally, to run some data mining algorithms, viz., k-NN requires the data to be in the encrypted domain computation-friendly ciphertext form. Thus, data are encrypted using a searchable encryption scheme before outsourcing it into the cloud. Asymmetric scalar-product-preserving encryption (ASPE) scheme provides secure *k-nearest neighbors* computation that is designed to provide both *Data Privacy* and *Query Privacy*. However, this scheme assumed that the query users were trusted entities. Enhancements to this work further showed that trusted query user assumption is no longer necessary if we use the Paillier cryptosystem. In this work, we have shown that even if we do not use the Paillier cryptosystem, query privacy in the cryptosystem can be achieved using the ASPE technique alone. Using ASPE for query encryption reduces the query encryption time, further improving the practicality of the encryption scheme.

Keywords: Privacy preserving · k-NN· Cloud computing · Encrypted Data

1 Introduction

The rapid progress of cloud computing has sparked a burgeoning movement towards migrating databases to cloud-based environments. The cloud also provides users with a query service to ease the cloud storage and computing pressure. Typically, a data owner delegates their databases to a cloud service for management, leveraging the cloud's resources and flexibility to decrease database

maintenance expenses[1, 15, 16]. The cloud performs both database maintenance and outsourced computation. Nonetheless, this presents a challenge concerning data security and privacy. As a result, safeguarding data privacy in externalized databases has gained significant attention in recent times, driving intensive research efforts in this domain.

Considering a practical scenario involving location-based services, a local service provider (LSP) hosts an extensive geospatial database containing comprehensive information about numerous locations. In this setup, a user can transmit her current location to the LSP, prompting the LSP to furnish query results such as the top five closest hotels. A prevalent approach among LSPs is to entrust their geospatial database to a robust public cloud, thereby availing geospatial database storage and location-based query processing capabilities. Cloud computing, acknowledged as the next-generation computing paradigm, promises LSPs many advantages, including reduced operational costs and enhanced performance.

However, this adoption of cloud services introduces a conundrum regarding data security and privacy. The act of outsourcing geospatial data to a public cloud deprives LSPs of direct data control, consequently giving rise to security concerns that impede the widespread adoption of this novel computing paradigm. One pressing issue is the potential for the cloud to gather and track the location of the data user (i.e., the querier), thereby compromising user privacy. Moreover, the ominous possibility of a cloud breach exists, leading to the unauthorized exposure of stored data. Such a breach could empower malicious actors with commercial gains or unfair advantages.

To safeguard the data privacy of LSPs, it becomes imperative to encrypt geospatial data before transmitting it to the cloud. This, however, introduces a pertinent question: which encryption schemes should be employed for this purpose? Traditional encryption methods lack the ability to perform computations in the encrypted domain, and while homomorphic encryption schemes exist, their efficiency for real-world data computations might be questionable [14].

In such places, searchable encryption [2, 4, 7, 18, 19, 23] plays a significant role viz., a client encrypts a collection of sensitive data for privacy protection and delegates the encrypted database to a server capable of effectively responding to search queries without requiring decryption of the database. Established methodologies address intricate and comprehensive queries [7, 8] within the structural encryption (STE) conceptual framework [5]. More information can be obtained from the survey paper by Fuller et al. [9].

Computation of k-nearest neighbor (k-NN) of a given query vector finds wide application in many fields, such as location-based services, data mining, e-healthcare, etc. The (k-NN) query seeks to identify the closest k neighbors to a specified query vector. Nevertheless, the outcome of a k-NN query is intimately tied to a user's preferences and interests. Consequently, scholarly exploration has been into devising secure k-NN query processing algorithms that simultaneously uphold data privacy and query confidentiality. One such scheme is the asymmetric scalar-product-preserving encryption (ASPE) technique introduced by Wong et al. [23]. It finds many applications in secure cloud computation,

including k-NN, fuzzy keyword search, content-based image retrieval, keyword matching, etc.

In [23], Wong treated the cloud as an untrusted entity while the query users as a trusted entity; thus, the data encryption key is shared with them, which in real-world scenarios may not be an acceptable assumption to consider. Later, Zhu et al. [26] addressed this issue and presented a way to achieve query privacy without sharing the secret key with the query users. However, they used the help of Paillier cryptosystem, which requires comparatively higher encryption and decryption time for query encryption and decryption. This work replaces the Paillier encryption scheme using the ASPE technique and has shown that query privacy of the encryption can also be achieved using the ASPE technique. The use of ASPE reduces the query encryption time, making the encryption scheme more efficient and practical at the same time.

The rest of the paper is organized as follows: Section 2 introduces the background, and section 3 related work. Section 4 presents the system model, while section 5 presents the proposed encryption scheme. Section 6 presents the performance evaluation of the proposed encryption scheme, and section 7 presents the proposed work's empirical evaluation. Finally, section 8 concludes the paper.

Mathematical notations used in the paper can be found in Table 1.

2 Background

This section will briefly review the ASPE encryption scheme [26]. In the ASPE encryption scheme, given a database $\boldsymbol{\mathcal{D}} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_m)$ of d-dimension with i^{th} tuple as $(p_{i_1}, p_{i_2}, \cdots, p_{i_d})$. The database is encrypted first and stored in the cloud as $\boldsymbol{\mathcal{D}'} = (\boldsymbol{p}'_1, \boldsymbol{p}'_2, \cdots, \boldsymbol{p}'_m)$. ASPE encryption scheme increases the dimension of each data vector by $(1+c+\epsilon)$-dimension. Thus given a d-dimensional data vector $(p_{i_1}, p_{i_2}, \cdots, p_{i_d})$ the encrypted data vector results into $(d+1+c+\epsilon)$-dimensional vector as $(p'_{i_1}, p'_{i_2}, \cdots, p'_{i_{(d+1+c+\epsilon)}})$. Positive integers c and ϵ are the dimensions added to maintain the security of the encryption scheme.

ASPE encryption scheme can be modularized into four modules: key generation, data encryption, query encryption, and query processing. Below, we have covered each of these modules in brief.

Key Generation: In this step, the data owner (DO) generates the data encryption key as follows

- $\boldsymbol{s} = (s_1, s_2, \cdots, s_{d+1}) \in \mathbb{R}^{d+1}$ and $\boldsymbol{\tau} \in \mathbb{R}^c$, fixed long-term secrets.
- $\boldsymbol{v}_i \in \mathbb{R}^\epsilon$ is a per-tuple ephemeral secret while $\boldsymbol{r}^{(q)} \in \mathbb{R}^c$ and $\beta_q \in \mathbb{R}$ are per-query ephemeral secrets.
- $\boldsymbol{M}$ is an invertible matrix with $(d+1+c+\epsilon)$ rows/columns and with elements drawn uniformly at random from $\mathbb{R}$.
- $\boldsymbol{\pi}$ a secret permutation function of length $(d+1+c+\epsilon)$.
- The resultant secret key consist of $(\boldsymbol{M}, \boldsymbol{s}, \boldsymbol{\tau}, \boldsymbol{\pi})$.

Table 1: Table of Notations

Notation	Meaning
A - Z	matrices
a - z	vectors
a - z	constants
$\boldsymbol{\mathcal{D}}$	database of vectors
$\mathbf{p}_i$	i^{th} data vector
$\mathbf{q}$	query vector
d	dimension of data and query vector
$\mathbf{p}_i'$	encrypted data vector $\mathbf{p}_i$
$\boldsymbol{\mathcal{D}}'$	database $\boldsymbol{\mathcal{D}}$ in encrypted form
$\mathbf{p}_i''$	computed from $\mathbf{p}_i'$ for k-NN computation
$\boldsymbol{\mathcal{D}}''$	database created for k-NN computation
$\dot{\mathbf{q}}$	encrypted query vector by Query User
$\hat{\mathbf{q}}$	encrypted query vector by Data Owner
$\tilde{\mathbf{q}}_{vec}$	query encrypted for k-NN computation
$\mathbf{M}_{base}^{\eta\times\eta}$	invertible base secret matrix
$\mathbf{M}_{t}^{\eta\times\eta}$	invertible temporary secret matrix
$\mathbf{E}^{\eta\times\eta}$	error matrix used for encryption
$\mathbf{N}^{d\times d}$	diagonal matrix used for query encryption by QU
$\mathbf{N}'^{\eta\times\eta}$	diagonal matrix used for query decryption by QU
$ED(\boldsymbol{p}_i) = \|\boldsymbol{p}_i\|$	$\sqrt{\sum_{l=0}^{d} p_{il}^2}$
$ED(\boldsymbol{p}_i, \boldsymbol{q})$	$\sqrt{\sum_{l=0}^{d} (p_{il} - q_l)^2}$
c, ϵ	security parameters > 0
$\eta = (d + 1 + c + \epsilon)$	encrypted data/query vector length
$\mathbf{s}$	$(d+1)$-dimensional vector of reals
$\boldsymbol{w}$	c-dimensional fixed real vector
$\boldsymbol{z}$	ϵ-dimensional data encryption ephemeral vector
$\boldsymbol{x}$	c-dimensional query encryption ephemeral vector
β_1, β_2	random real numbers used for encryption

Data Encryption: In this step, the database of vectors is encrypted row-wise. For encryption, the following steps are performed.

– It starts by shifting each tuple of data vector $\boldsymbol{p}_i$ by the shifting vector $\boldsymbol{s}$ later augmented by two secrets, long-term secret $\boldsymbol{\tau}$ and per tuple ephemeral secret $\boldsymbol{v}_i$ creating the shifted data vector as

$$\hat{\boldsymbol{p}_i} = (s_1 - 2p_{i_1}, \cdots, s_d - 2p_{i_d}, s_{d+1} + \|\boldsymbol{p}_i\|^2, \boldsymbol{\tau}, \boldsymbol{v}_i)$$

– The shifted vector is later encrypted using the secret encryption matrix as

$$\boldsymbol{p}_i' = \hat{\boldsymbol{p}}_i \hat{\boldsymbol{M}}^{-1}$$

Here, $\hat{\boldsymbol{M}} = \pi(\boldsymbol{M})$, is the matrix obtained by permuting the columns of $\boldsymbol{M}$ using the permutation function π.

Query Encryption: For query encryption, interaction between the DO and query user (QU) is needed. The query user performs the first step of query encryption. In this step, QU encrypts the query vector using the Paillier cryptosystem and forwards it to the DO for its encryption. In the second step, DO re-encrypts the encrypted query vector and reverts it to the QU. QU performs the third and final step of query encryption. In this step, QU removes his encryption layer and forwards the encrypted query vector encrypted with only DO's secret key to the cloud service provider (CSP) for k-NN computation. The encryption scheme presented in this work removes the Paillier encryption scheme for query encryption and uses the ASPE techniques instead. Thus, understanding the query encryption of Zhu et al. [26] in detail is optional for the present work. In general, the encrypted query $\boldsymbol{q}'_{enc}$ have a form of

$$\boldsymbol{q}'_{enc} = \beta_q . \hat{\boldsymbol{M}} \hat{\boldsymbol{q}}_*$$

where $\hat{\boldsymbol{q}}_* = (\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_d, 1, \boldsymbol{r}^{(q)}, \boldsymbol{0}_{(\epsilon)})$

k-NN computation: In its simplest form, solving the k-nearest neighbors problem involves computing the distance, $D(\boldsymbol{p}_i, \boldsymbol{q})$ between the query vector, $\boldsymbol{q}$ and each database vector, $\boldsymbol{p}_i$. However, in the current context, we are dealing with an encrypted query vector and an encrypted database. Fortunately, a comparison between the distances, $D(\boldsymbol{p}_i, \boldsymbol{q})$ and $D(\boldsymbol{p}_j, \boldsymbol{q})$ reduces to a simple comparison in the encrypted domain as derived below:

$$\begin{aligned}
& D(\boldsymbol{p}_i, \boldsymbol{q}) > D(\boldsymbol{p}_j, \boldsymbol{q}) \\
\Leftrightarrow\ & \|\boldsymbol{p}_i - \boldsymbol{q}\|^2 > \|\boldsymbol{p}_j - \boldsymbol{q}\|^2 \\
\Leftrightarrow\ & \|\boldsymbol{p}_i\|^2 - 2\boldsymbol{p}_i\boldsymbol{q} + \|\boldsymbol{q}\|^2 > \|\boldsymbol{p}_j\|^2 - 2\boldsymbol{p}_j\boldsymbol{q} + \|\boldsymbol{q}\|^2 \\
\Leftrightarrow\ & -2\sum_{k=1}^{d} p_{ik}q_k + \|\boldsymbol{p}_i\|^2 > -2\sum_{k=1}^{d} p_{jk}q_k + \|\boldsymbol{p}_j\|^2 \\
\Leftrightarrow\ & \sum_{k=1}^{d}(s_k - 2p_{ik})q_k + s_{d+1} + \|\boldsymbol{p}_i\|^2 + \boldsymbol{\tau}.\boldsymbol{r}^{(q)} \\
& \quad > \sum_{k=1}^{d}(s_k - 2p_{jk})q_k + s_{d+1} + \|\boldsymbol{p}_j\|^2 + \boldsymbol{\tau}.\boldsymbol{r}^{(q)} \\
\Leftrightarrow\ & \boldsymbol{p}'_i\boldsymbol{q}' > \boldsymbol{p}'_j\boldsymbol{q}'
\end{aligned}$$

As presented above, we can safely compute the k-NN in the encrypted domain using the ASPE technique.

This completes the brief description of the ASPE encryption scheme and is sufficient to understand the work in this paper.

3 Related Work

This section reviews the existing works on privacy-preserving secure k-NN computation. Privacy-preserving secure k-NN computation – retrieval of top-k database

records satisfying the smallest distances to a given query vector in the encrypted domain. It finds many applications in various fields, viz., data mining, pattern recognition, e-healthcare, location-based services, and signal processing, to name a few. Due to its ubiquitous application in various fields, it finds considerable attention from both industries and academia. Thus, it remains an active field of research in the outsourced computation environment, preserving data and query privacy.

In literature, different k-NN computing techniques are available, viz., matrix encryption, homomorphic encryption, private information retrieval, and other privacy preservation techniques. ASPE based on matrix multiplication initially presented by Wong et al. [23] in 2009 is used to realize k-NN computation. It focuses on the privacy of the database and query vector from the CSP while the QUs are treated as trusted users; thus, the data encryption key is shared with them for query encryption, which might not be a reasonable assumption to consider. Zhu et al. [26] improved the encryption scheme by treating query users as non-trusted entities and provided a privacy-preserving k-NN query computation using the Paillier encryption scheme. The scheme requires the DO's active participation in the query encryption process. Over time, many privacy-preserving schemes have been designed based on the ASPE scheme.

Additionally, many similarity range query schemes have also been proposed viz., [3, 20, 21, 11, 24, 26]. Later, it was shown that such a scheme could not resist known-plaintext attacks as mentioned in [13]. The key confidentiality claim of the encryption scheme [26] is later breached by the work of [17]. However, [17] also present a way to enhance the scheme and a technique to restrict known plaintext attack on the encryption scheme and shows a way to securely compute k-NN queries, keeping data privacy and query privacy intact. A variant of the k-NN problem is the reverse k-NN problem that produces k data vectors whose query vector is the k-NN. Li et al. [12] used the ASPE technique and other methods to realize the same.

Meanwhile, many schemes leverage homomorphic encryption techniques to protect data and query privacy viz., [10, 25]. In addition, some privacy-preserving k-NN query computing schemes [22, 6] were also designed by employing other privacy preservation techniques, e.g., private information retrieval. ASPE, however, still finds an active place in all these techniques due to its computation speed in the encrypted domain.

4 System Model

The security model of secure k-NN computation consists of three entities: a data owner (DO), a cloud service provider (CSP), and a group of query users (QU). Here, the DO encrypts its database intended to be stored in the cloud using a searchable encryption scheme and stores it in the cloud as shown in figure 1. A QU intending to compute k-NN of its query vector encrypts its query vector using the encryption scheme mentioned in section 5 and sends it to DO for query re-encryption by the DO's secret key. On receiving the query vector from

a QU, DO re-encrypts it using the procedure mentioned in section 5 and sends it back to the QU. On receiving an encrypted query vector from the DO, QU removes his encryption layer as mentioned in section 5 and sends the encrypted query vector for k-NN computation to the CSP. CSP checks the *user_id* and uses the corresponding data for k-NN calculation. The data prepared using the temporary secret key $\mathbf{M}_t$ uses the Euclidean distance as the distance metric to obtain k data vector. The computed k data labels are later forwarded to the QU as the obtained k-NN. The data labels may correspond to different disease names in the case of healthcare data and nearby restaurant names in the case of LSP and usually depend on the data type and query under consideration.

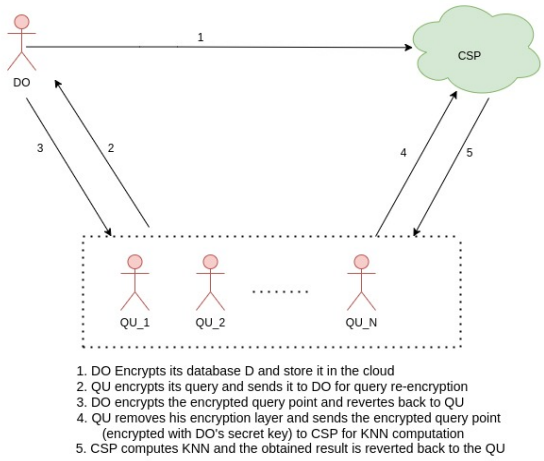


Fig. 1: Secure k-NN computation model in the cloud computation environment. In the figure, DO represents the data owner, QU represents the query users, and CSP represents the cloud service provider.

A semi-honest threat model has been considered in the proposed encryption scheme. Here, the cloud is the semi-honest adversary (a.k.a., honest-but-curious), viz., it does not deviate from the defined protocol. However, it uses the legitimately received data to infer other private or sensitive information. For instance, the cloud may want to infer the DO's geospatial data and the QU's query locations.

Consequently, to protect the DO's geospatial location data and QU's query location data, both the geospatial data and query location should be encrypted before sending it to the public cloud. The QU also behaves as a semi-honest entity; however, to gain information about the secret key used for data encryption, he tries to frame queries of his choice. Here, we assume that the CSP and QU are non-colluding entities.

5 Proposed Scheme

This section will discuss the proposed enhanced encryption scheme. The proposed encryption scheme can be modularized into the following four modules, similar to the previous work: key generation, data encryption, query encryption, and secure k-NN computation. Query encryption is performed in three steps: query encryption by QU, query encryption by DO, and finally, query encryption by QU. The encrypted query is later sent to the CSP for secure k-NN computation. The CSP computes the k-NN in the encrypted domain and reverts the obtained label to the query user. Below, each module mentioned above has been discussed in more detail.

Key Generation: In this step, the DO having a database $\mathbb{D}$ of n vectors with d-dimension generates the public parameters c and ϵ and computes η as $\eta = d + 1 + c + \epsilon$. Samples the base secret matrix $\mathbf{M}_{base}$ of $\eta \times \eta$-dimension, here we have termed the secret matrix as base secret matrix; the reason behind this will be discussed later. Furthermore, a long-term secret vector $\mathbf{s}$ of length $(d + 1)$ is also generated; it is used to hide the data vector by shifting each data vector by this secret vector. DO also generates a fixed vector $\mathbf{w}$ of length c uniformly at random for each data vector to be encrypted.

The secret key of the encryption scheme is $(\mathbf{s}, \mathbf{M}_{base}, \mathbf{w})$.

Data Encryption: In this step, DO samples a ephemeral secret vector $\mathbf{z}$ of length ϵ and pre-processes each data vector to obtain a pre-processed data vector as

$$\tilde{\mathbf{p}}_i = (s_1 - 2p_{i1}, \cdots, s_d - 2p_{id}, s_{d+1} + ||\mathbf{p}_i||^2, \mathbf{w}, \mathbf{z})$$

The pre-processed data vector is later encrypted by performing vector-matrix multiplication with base secret matrix $\mathbf{M}_{base}$ as shown in equation 1.

$$\mathbf{p}'_i = \tilde{\mathbf{p}}_i \mathbf{M}_{base}^{-1} \tag{1}$$

Encrypted data vectors are later stored in the cloud, where the rest of the query computations are performed. DO also computes the Euclidean norm (EN) of data vectors and stores the maximum norm as $Max\_norm = Max(EN(\boldsymbol{p}_1), \cdots, EN(\boldsymbol{p}_m))$, which will be used for query encryption later.

Performing the computation as mentioned above and storing the database in the cloud, DO can drop the database $\mathbf{D}$ from the local storage. In the future, if the database is needed, he can download it from the cloud and decrypt it to get the database in the plaintext format.

Query Encryption: Interaction between the QU and the DO is required to perform query encryption. Different steps performed by the DO and the QU are divided into three steps. Each of these three steps has been covered below.

– Step 1: QU performs the first step of query encryption. A QU having a query vector $\mathbf{q}$ of d-dimension samples a diagonal matrix $\mathbf{N}^{d\times d}$ of real numbers and computes the encrypted query $\dot{\mathbf{q}}$ as

$$\dot{\mathbf{q}} = \beta_1 \cdot \mathbf{q} \cdot \mathbf{N}$$

Here, β_1 is an encryption constant and is a real number. The encrypted query $\dot{\mathbf{q}}$ obtained above is later sent to the DO along with query user id as $(user\_id, \dot{\mathbf{q}})$ to obtain a doubly encrypted query vector $\hat{\mathbf{q}}$. This completes step 1 of query encryption.

– Step 2: This step of query encryption is performed by the DO. Here the DO at first computes the largest number present in the encrypted query vector $\dot{\mathbf{q}}$ as $q_{max} = max(\dot{\mathbf{q}})$. Next, it samples a temporary secret matrix $\mathbf{M}_t^{\eta\times\eta}$, with all elements except the diagonal elements of matrix $\mathbf{M}_t$ uniformly at random larger than q_{max}. In contrast, the diagonal elements are sampled uniformly at random larger than Max_norm. Later, matrix $\mathbf{M}_t$ is multiplied with $\mathbf{M}_{base}$ to obtain a temporary secret matrix for the query encryption as $\mathbf{M}_{sec} = \mathbf{M}_t\mathbf{M}_{base}$.
The obtained query vector $\dot{\mathbf{q}}$ is appended to obtain a new query vector $\mathbf{q}'$ of η-dimension as

$$\mathbf{q}' = (\dot{\mathbf{q}}, 1, \mathbf{x}, \mathbf{0}_\epsilon)$$

where $\mathbf{x}$ is an ephemeral integer vector of length c and $\mathbf{0}_\epsilon$ is a zero vector of length ϵ. Next it uses the operator O_e to convert vector $\mathbf{q}'$ into a $\eta \times \eta$ dimensional diagonal matrix $\mathbf{q}_{\eta\eta}$ as follows

$$O_e : \mathbf{q}' \rightarrow \mathbf{q}_{\eta\eta} \; with \quad q_{ii} = q'_i$$

Next, it performs computation as presented by equation 2 below

$$\hat{\mathbf{q}} = \beta_2(\mathbf{M}_{sec}\mathbf{q}_{\eta\eta} + \mathbf{E}) \tag{2}$$

Here the error matrix $\mathbf{E}^{\eta\times\eta}$ is used to hide the secret matrix $\mathbf{M}_{sec}$ from QU and the real number β_2 is a DO encryption constant. Elements of the error matrix $\mathbf{E}$ are sampled uniformly at random larger than q_{max}. The reason is that the QU should not learn any content of the secret matrix by looking into the obtained doubly encrypted query vector. It is to be noted that the elements of the matrix $\mathbf{E}$ are sampled in such a way that the correctness of the k-NN computation remains intact; in the later section, we will discuss it further.
DO returns the doubly encrypted query matrix $\hat{\mathbf{q}}$ to the QU and send the temporary secret matrix and $user\_id$ to the CSP as $(user\_id, \mathbf{M}_t)$. The CSP prepares the temporary database $\mathbf{D}''$ for the query user $user\_id$ for computing k-NN computation as

$$\begin{aligned} \mathbf{p}''_i &= \mathbf{p}'_i\mathbf{M}_t^{-1} \\ \mathbf{p}''_i &= \tilde{\mathbf{p}}_i\mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \end{aligned} \tag{3}$$

This completes the step 2 of query encryption.

– Step 3: This step of query encryption is performed by the QU. In this step, QU removes his encryption layer to obtain $\tilde{\mathbf{q}}_{enc}$. QU constructs a diagonal matrix $\mathbf{N}'^{\eta\times\eta}$ with first d diagonal elements same as that of the matrix $\mathbf{N}^{d\times d}$ and rest all 1. Computes its inverse and uses it to remove his encryption layer as follows

$$\begin{aligned}\tilde{\mathbf{q}}_{enc} &= \hat{\mathbf{q}}\mathbf{N}'^{-1} \\ &= \beta_2(\mathbf{M}_{sec}\mathbf{q}''_{\eta\eta}\mathbf{N}' + \mathbf{E}) \cdot \mathbf{N}'^{-1} \\ &= \beta_2(\mathbf{M}_{sec}\mathbf{q}''_{\eta\eta} + \mathbf{E} \cdot \mathbf{N}'^{-1}) \end{aligned} \tag{4}$$

where $\mathbf{q}''_{\eta\eta}$ is a diagonal matrix with diagonal element $(\beta_1\mathbf{q}, 1, \mathbf{x}, \mathbf{0}_\epsilon)$.
The encrypted query matrix $\tilde{\mathbf{q}}_{enc}$ is converted to a vector $\tilde{\mathbf{q}}_{vec}$ using an operator O_r as presented below.

$$O_r : \tilde{\mathbf{q}}_{enc} \rightarrow \tilde{\mathbf{q}}_{vec}$$

Here $\tilde{\mathbf{q}}_{vec_j} = \sum_{i=1}^{\eta} \tilde{\mathbf{q}}_{enc_{ji}}$, for $j \in \{1, \eta\}$ viz., the j^{th} element of vector $\tilde{\mathbf{q}}_{vec}$ is obtained by adding all columns of row j of matrix $\tilde{\mathbf{q}}_{enc}$. In short

$$\tilde{\mathbf{q}}_{vec} = \beta_2(\mathbf{M}_{sec}\mathbf{q}''_{\eta} + err) \tag{5}$$

where $\mathbf{q}''_{\eta}$ is a vector of η elements as $(\beta_1\mathbf{q}, 1, \mathbf{x}, \mathbf{0}_\epsilon)$ and err is a vector obtained by computing $Row\_Sum(\mathbf{E} \cdot \mathbf{N}'^{-1})$. The encrypted vector $\tilde{\mathbf{q}}_{vec}$ is later sent to the CSP for k-NN computation. This completes stage 3 of query encryption and, as a whole, query encryption as well.

Decryption: Decryption of encrypted data vectors is straightforward and is performed as

$$\tilde{\mathbf{p}}_i = \mathbf{p}'_i\mathbf{M}_{base}$$

From $\tilde{\mathbf{p}}_i$ data tuples are recovered by following

$$p_{ij} = s_j - \tilde{\mathbf{p}}_{ij}$$

k-NN computation: The cloud performs the k-NN computation by performing multiplication of encrypted data vectors $\mathbf{p}''_i$ from the temporary database $\mathbf{D}''$ and an encrypted query vector $\tilde{\mathbf{q}}_{enc}$ as follows

$$\begin{aligned}\mathbf{p}''_i\tilde{\mathbf{q}}_{vec} &= \beta_2[\tilde{\mathbf{p}}_i\mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1}(\mathbf{M}_{sec}\mathbf{q}''_{\eta} + err)] \\ &= \beta_2(\tilde{\mathbf{p}}_i\mathbf{q}''_{\eta} + \tilde{\mathbf{p}}_i \cdot \mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \cdot err) \\ &= \beta_2(\tilde{\mathbf{p}}_i\mathbf{q}''_{\eta} + err'_i) \qquad\qquad [err'_i = \tilde{\mathbf{p}}_i \cdot \mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \cdot err]\end{aligned}$$

The proposed encryption scheme performs correct k-NN computation provided it satisfies

$$(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_j)\mathbf{q}''_{\eta} > (err'_i - err'_j)$$

This is ensured by selecting parameters as discussed in step 2 of query encryption.

5.1 Correctness Analysis

The correctness of the encryption scheme needs two-fold proof. First, we need to show that the decryption of the encrypted data produces consistent plaintext data. This has already been shown in the decryption part above. Second, we must show that the encryption scheme produces correct k-NN computation in the encrypted domain. For this, it is sufficient to prove the lemma 1 presented below.

Lemma 1. *The encrypted data point $\mathbf{p}_i''$ and encrypted query point $\tilde{\mathbf{q}}_{vec}$ multiplication result comparison evaluates the k-NN.*

Proof. The encrypted data point $\mathbf{p}_i''$ when multiplied with encrypted query point $\tilde{\mathbf{q}}_{vec}$ it gives rise to

$$\mathbf{p}_i''\tilde{\mathbf{q}}_{vec} = \beta_2(\tilde{\mathbf{p}}_i\mathbf{q}_\eta'' + err_i')$$

Here,

$$\begin{aligned}\mathbf{p}_i''\tilde{\mathbf{q}}_{vec} &= \beta_1\beta_2[(s_1q_1 + \cdots + s_dq_d + s_{d+1}) - 2(p_{i1}q_1 + \cdots + p_{id}q_d) + ||\mathbf{p}_i||^2] \\ &\quad + \beta_2 \cdot \mathbf{wx} + \beta_2 \cdot err_i'\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbf{p}_j''\tilde{\mathbf{q}}_{vec} &= \beta_1\beta_2[(s_1q_1 + \cdots + s_dq_d + s_{d+1}) - 2(p_{j1}q_1 + \cdots + p_{jd}q_d) + ||\mathbf{p}_j||^2] \\ &\quad + \beta_2 \cdot \mathbf{wx} + \beta_2 \cdot err_j'\end{aligned}$$

Thus,

$$\begin{aligned}(\mathbf{p}_i'' - \mathbf{p}_j'')\tilde{\mathbf{q}}_{vec} &= \beta_1\beta_2[(||\mathbf{p}_i||^2 - 2p_{i1}q_1 - \cdots - 2p_{id}q_d + ||\mathbf{q}||^2) \\ &\quad - (||\mathbf{p}_j||^2 - 2p_{j1}q_1 - \cdots - 2p_{jd}q_d + ||\mathbf{q}||^2)] + \beta_2(err_i' - err_j') \\ &= \beta_1\beta_2[(p_{i1} - q_1)^2 - (p_{j1} - q_1)^2 + \cdots + (p_{id} - q_d)^2 - (p_{jd} - q_d)^2] \\ &\quad + \beta_2(err_i' - err_j')\end{aligned}$$

The above term evaluates to correct comparison provided it satisfies the inequality given below

$$\begin{aligned}(err_i' - err_j') &< \beta_1[(p_{i1} - q_1)^2 - (p_{j1} - q_1)^2 + \cdots + (p_{id} - q_d)^2 - (p_{jd} - q_d)^2] \\ \beta_1(\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_j)\mathbf{q}_\eta'' &> (\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_j)\mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \cdot err \\ \beta_1\mathbf{q}_\eta'' &> \mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \cdot Row\_Sum(\mathbf{E} \cdot \mathbf{N}'^{-1}) \qquad (6)\end{aligned}$$

In equation 6 above, by choosing elements of $\mathbf{E}$ and $\mathbf{M}_t$ as mentioned in step 2 of query encryption makes $\mathbf{M}_{base}^{-1}\mathbf{M}_t^{-1} \cdot Row\_Sum(\mathbf{E} \cdot \mathbf{N}^{-1})$ negligible. This is because elements of matrix $\mathbf{M}_t$ are large, making elements of matrix $\mathbf{M}_t^{-1}$ quite small. Furthermore, it further gets multiplied with elements of matrix $\mathbf{M}_{base}^{-1}$ making overall multiplication negligible, which when multiplied with vector $Row\_Sum(\mathbf{E} \cdot \mathbf{N}'^{-1})$, the resultant vector have negligibly large values. Thus the overall result becomes quite small, which enforces the evaluation of correct k-NN.

5.2 Security Analysis

The proposed encryption scheme is claimed to hold following properties: *Key Confidentiality*, *Data Privacy*, *Query Privacy* and *Known Plaintext Attack*. We will discuss each of these properties one by one below.

Data Privacy: DO outsource its database in the encrypted form to the CSP. Thus, CSP can access the DO's private data other than the DO. Privacy of the DO's private data remains intact if encrypted data does not leak any information about the plaintext data. So, it is sufficient to prove lemma 2 presented below.

Lemma 2. *Without the knowledge of secret matrix $\boldsymbol{M}_{base}$, encrypted database $\mathbf{D}'$ behaves as a random number.*

Proof. In the presented encryption scheme an encrypted data vector $\mathbf{p}'_i$ is obtained by performing computation as

$$\mathbf{p}'_i = \tilde{\mathbf{p}}_i \mathbf{M}^{-1}_{base}$$

where the initial data vector $\mathbf{p}_i$ is pre-processed to obtain $\tilde{\mathbf{p}}_i$ as

$$\tilde{\mathbf{p}}_i = (s_1 - 2p_{i1}, \cdots, s_d - 2p_{id}, s_{d+1} + ||\mathbf{p}_i||^2, \mathbf{w}, \mathbf{z})$$

In the pre-processed data vector $\tilde{\mathbf{p}}_i$, an ephemeral random vector $\mathbf{z}$ of length ϵ is used. The use of ephemeral random vector $\mathbf{z}$ for encrypting $\mathbf{p}_i$ makes it random even if the same data vector is encrypted for the second time, resulting in a semantically secure encryption scheme. Thus, no information about the plaintext data can be derived from the ciphertext data unless the secret key of the encryption scheme is known. This completes the proof of the lemma.

Query Privacy: Query Privacy of the encryption scheme needs to be established from the DO and the CSP. We will discuss each of these two cases separately.

– Case1: Query privacy against DO – It is established by encrypting the data vector by the QU. The encrypted query vector $\dot{\mathbf{q}}$ received by the DO have a form of

$$\dot{\mathbf{q}} = \beta_1 \cdot \mathbf{q} \cdot \mathbf{N}$$

Here query vector $\mathbf{q}$ is multiplied with a diagonal random matrix $\mathbf{N}$ and a real number β_1 to obtain the encrypted query vector $\dot{\mathbf{q}}$. Each element of query vector $\dot{\mathbf{q}}$ looks something like

$$\dot{q}_i = \beta_1 \cdot q_i \cdot n_{ii} \tag{7}$$

In the above equation, the i-th diagonal element n_{ii} and β_1 are unknown real numbers. Without their knowledge, retrieval of q_i is infeasible.

– Case2: Query privacy against CSP – It comes from the query encryption performed by the DO. The query vector sent for k-NN computation to the CSP has a form of

$$\tilde{\mathbf{q}}_{vec} = \beta_2(\mathbf{M}_{sec}\mathbf{q}''_{\eta} + err) \tag{8}$$

where $\mathbf{q}''_{\eta} = (\beta_1\mathbf{q}, 1, \mathbf{x}, \mathbf{0}_{\epsilon})$ and err is a vector obtained by computing $Row\_Sum(\mathbf{E} \cdot \mathbf{N}'^{-1})$. The i^{th} element of $\tilde{\mathbf{q}}_{vec}$ is computed as

$$\tilde{\mathbf{q}}_{vec_i} = \beta_2(\mathbf{M}_{sec_i}\mathbf{q}''_{\eta} + err_i) \tag{9}$$

From equation 8 and 9, it is clear that retrieval of $\mathbf{q}''_{\eta}$ from $\tilde{\mathbf{q}}_{vec}$ requires the knowledge of β_2, $\mathbf{M}_{sec}$ and err_i. Thus, without the knowledge of β_2, $\mathbf{M}_{sec}$ and err_i retrieval of the plain query is infeasible. In short, encryption performed by the DO can successfully withstand the query privacy against the CSP.

Key Confidentiality : Key confidentiality of the encryption scheme means the data encryption key should not be revealed to anyone and should remain confidential. Key confidentiality of the encryption scheme from CSP is straightforward as the DO only stores its encrypted database in the CSP, which looks completely random. Furthermore, for each query user interested in performing k-NN computation, a temporary key $\mathbf{M}_t$ is sent to the CSP to prepare data for the query user. This new temporary key $\mathbf{M}_t$ keeps on changing for each new k-NN computation request and is entirely independent of $\mathbf{M}_{base}$. Thus, by looking into the random-looking encrypted database and $\mathbf{M}_t$, no information about the base key $\mathbf{M}_{base}$ can be derived.

Additionally, as discussed in the *query privacy* part above, CSP can not figure out the data encryption key used for database encryption by looking into the encrypted database $\mathbf{D}'$ this is due to the use of the ephemeral secret vector $\mathbf{z}$, which makes the encrypted data vector pseudorandom. Thus by using database $\mathbf{D}'$ and query $\tilde{\mathbf{q}}_{enc}$ only k-NN can be computed, and no further information about the secret key matrix $\mathbf{M}_{base}$ can be obtained unless the error term err and constant β_1 and β_2 are known.

Key confidentiality against the QU comes from the fact that the encrypted query vector sent to the QU is of the form of equation 2. When QU multiplies matrix $\mathbf{N}'^{-1}$ to equation 2 the equation evaluates to

$$\hat{\mathbf{q}}\mathbf{N}'^{-1} = \beta_2(\mathbf{M}_{sec}\mathbf{q}''_{\eta\eta} + \mathbf{E} \cdot \mathbf{N}'^{-1}) \tag{10}$$

Here $\mathbf{q}''_{\eta\eta}$ is a diagonal matrix with diagonal elements $(\beta_1\mathbf{q}, 1, \mathbf{x}, \mathbf{0}_{\epsilon})$.

In equation 10, query user knows matrix $\mathbf{q}''_{\eta\eta}$ and $\mathbf{N}'^{-1}$. As the QU can ask queries of any form, he can construct a query of his choice to extract maximum information about the secret key matrix used for the data encryption. Below we tried to cover each possible queries a QU can frame and tried to analyze key confidentiality against the query user.

– Case 1: Query user constructs a query with all query elements set to 1 and send the query vector as shown below for encryption

$$\dot{\mathbf{q}} = \mathbf{1N}$$

The received query after the removal of $\mathbf{N}$ will be of the following form

$$\tilde{\mathbf{q}} = \beta_2(\mathbf{M}_{sec}\mathbf{1} + \mathbf{E}\mathbf{N}'^{-1}) \tag{11}$$

Here the value of $\mathbf{E}$ and β_2 are unknown to the QU. So he can not get meaningful information about the secret key matrix $\mathbf{M}_{sec}$.

– Case 2: The QU constructs a query by setting the first element to non-zero, rest all zeros, and sends it for encryption. On the received encrypted query, when the QU removes his encryption layer by multiplying $\mathbf{N}'^{-1}$, the obtained query has a form as shown below.

$$\begin{aligned}\tilde{\mathbf{q}} &= \beta_2 \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1\eta} \\ m_{21} & m_{22} & \cdots & m_{2\eta} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\eta 1} & m_{\eta 2} & \cdots & m_{\eta\eta} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \beta_2 \begin{pmatrix} E_{11}N_{11}^{-1} & E_{12}N_{22}^{-1} & \cdots & E_{1\eta}N_{\eta\eta}^{-1} \\ E_{21}N_{11}^{-1} & E_{22}N_{22}^{-1} & \cdots & E_{2\eta}N_{\eta\eta}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ E_{\eta 1}N_{11}^{-1} & E_{\eta 2}N_{22}^{-1} & \cdots & E_{\eta\eta}N_{\eta\eta}^{-1} \end{pmatrix} \\ &= \beta_2 \begin{pmatrix} m_{11} + E_{11}N_{11}^{-1} & E_{12}N_{22}^{-1} & \cdots & E_{1\eta}N_{\eta\eta}^{-1} \\ m_{21} + E_{21}N_{11}^{-1} & E_{22}N_{22}^{-1} & \cdots & E_{2\eta}N_{\eta\eta}^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\eta 1} + E_{\eta 1}N_{11}^{-1} & E_{\eta 2}N_{22}^{-1} & \cdots & E_{\eta\eta}N_{\eta\eta}^{-1} \end{pmatrix} \end{aligned} \tag{12}$$

From equation 12 without the knowledge of error matrix $\mathbf{E}$ and constant β_2 no information about the secret matrix $\mathbf{M}_{base}$ can be retrieved. Furthermore, for each new query secret matrix $\mathbf{M}_{sec}$ keeps on changing making the secret key retrieval impossible.

Known Plaintext Attack: Assuming a CSP with an encrypted database has access to a polynomial number of plaintext data vectors, he can perform a known plaintext attack as shown in [13]. However, the same attack is not possible in the proposed encryption scheme; this is because in [13], ASPE computes the exact value of $\mathbf{p}_i \cdot \mathbf{q}$ to compare k-NN, which is later exploited using known plaintext data vectors to break the query confidentiality and later data privacy of the encryption scheme. However, extending this to the proposed encryption scheme is impossible because we do not compute the exact distance to compute k-NN in the proposed encryption scheme. In the proposed encryption scheme, only the relative distance is compared to find the k-NN, as shown below.

$$\begin{aligned}(\mathbf{p}_i'' - \mathbf{p}_j'')\tilde{\mathbf{q}}_{vec} &= \beta_1\beta_2[(||\mathbf{p}_i||^2 - 2p_{i1}q_1 - \cdots - 2p_{id}q_d + ||\mathbf{q}||^2) - (||\mathbf{p}_j||^2 - 2p_{j1}q_1 \\ &\quad - \cdots - 2p_{jd}q_d + ||\mathbf{q}||^2)] + \beta_2(err_i' - err_j') \\ &= \beta_1\beta_2[||\mathbf{p}_i||^2 - 2p_{i1}q_1 - \cdots - 2p_{id}q_d + 2p_{j1}q_1 + \cdots + 2p_{jd}q_d \\ &\quad - ||\mathbf{p}_j||^2] + \beta_2(err_i' - err_j')\end{aligned}$$

From the above equations, without the knowledge of β_1, β_2 and $(err'_i - err'_j)$, the CSP can not frame linear equations to retrieve the query vector $\mathbf{q}$ uniquely. Thus, we can not retrieve $\mathbf{q}$ from the above equations.

6 Performance Evaluation

This section will analyze the cost of our proposed encryption scheme, viz., the computation and communication costs.

6.1 Computation Cost

The computational complexity of different steps of the proposed encryption scheme is presented in table 2. DO generates different secrets for the key generation step, including the secret matrix $\mathbf{M}_{base}$. Thus, the computational complexity of the KeyGen step of the proposed encryption evaluates to $O(\eta^2)$. For the database encryption, DO multiplies m data vectors– each enhanced to η-dimensional vector for security, with the secret matrix $\mathbf{M}_{base}^{-1}$. Thus, the total computational complexity of database encryption evaluates to $O(m\eta^2)$.

Query encryption is performed in three steps, with QU performing the first step. In this step, the QU samples the diagonal matrix $\mathbf{N}$ and multiplies it with the query vector. This requires multiplication of d diagonal elements with d-dimensional query vector. So, the total computational cost evaluates to $O(d)$. The DO performs the second step of the query encryption by sampling a secret matrix $\mathbf{M}_t$ and multiplying it with the base secret key matrix $\mathbf{M}_{base}$, which is later multiplied with the query vector – enhanced to η dimension. Thus, the total computational complexity of this step evaluates to $O(\eta^3)$. The third and final step of query encryption is performed by the QU, which multiplies $\mathbf{N'}^{-1}$ to the received query vector, evaluating the total computational cost of this step to $O(\eta^3)$. Thus, the overall computational complexity of the query encryption evaluates to $O(\eta^3)$.

The computational cost of k-NN evaluation using the proposed encryption scheme evaluates to $O(m\eta \log k)$. This is because it requires the multiplication of m data vectors with η-dimension with a query vector of the same dimension and comparing the distances to figure out the k-NN.

Process	DO	QU	CSP	Total
Key Generation	$O(\eta^2)$	O(1)	-	$O(\eta^2)$
Database Encryption	$O(m\eta^2)$	-	-	$O(m\eta^2)$
Query Encryption	$O(\eta^3)$	$O(\eta^3)$	-	$O(\eta^3)$
k-NN Computation	-	-	$O(m\eta \log k)$	$O(m\eta \log k)$

Table 2: Computation complexity of the proposed encryption scheme

6.2 Communication Cost

Considering the proposed encryption scheme's communication cost, the initial key generation and database encryption involve only DO. Thus, zero communication cost is required in this step. Considering each tuple of encrypted query vectors as $\log n$ bits, QU sends d-dimensional encrypted query to the DO. DO encrypts it and sends the encrypted matrix of η^2 elements. Thus, the total communication cost between DO and QU equals $O(\eta^2 \log n)$ bits. The encrypted query vector is then sent to CSP for k-NN computation with a communication cost of $O(\eta \log n)$ bits. Finally, CSP returns computed k-NN result with communication cost equivalent to $O(kc)$ bits, where $c = \lceil \log(\mathbb{L}) \rceil$, here L represents the total number of labels in the database.

7 Empirical Evaluation

This section will cover the empirical performance of the proposed encryption scheme presented in section 5, it also covers performance analysis in detail with respect to the earlier encryption schemes, viz. Zhu et al. [26] for varying dimensions and the number of samples considering synthetic database generated using some computer programs.

7.1 Experiment Setup

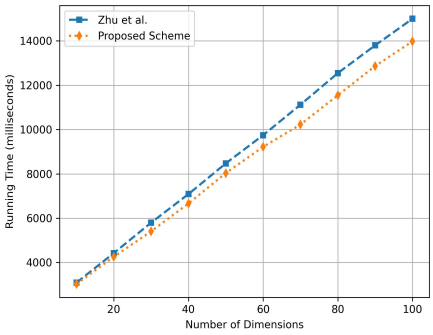
We have implemented the encryption scheme proposed in section 5 and Zhu et al. [26] in Python. The security parameters of both the encryption schemes have been fixed to $c = 5$ and $\epsilon = 5$. Paillier encryption scheme with a key size of 1024 bits is used to maintain query privacy between QU and DO in the case of Zhu et al. All experiments are performed on an Intel Core $i7 - 8700$ CPU @$3.20GHz \times 12$ with 32GB RAM running Ubuntu 22.04.

7.2 Experiments Performed

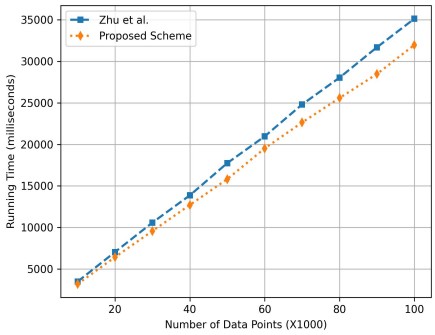
To better analyze the encryption scheme, we have performed five experiments. Each of these experiments has been covered in detail below.

Encryption time-varying dimension: To compare the encryption time of the proposed encryption schemes, we have created a database for varying dimensions starting from 10 up to 100 in a step of 10, keeping the number of samples fixed to $100,000$. The behavior of the encryption schemes has been plotted and is shown in figure 2a. The figure shows that the run time of the proposed encryption scheme increases linearly with dimension and behaves almost similarly to that of the Zhu encryption scheme. The minute difference in the figure is caused by removing the permutation function used in the Zhu et al. [26].

Encryption time varying samples: To compare the data encryption time of the proposed encryption scheme, we have kept the dimension of the encryption scheme fixed as 10 and varied the number of samples starting from $100,000$ to



(a) Varying Dimension



(b) Varying Samples

Fig. 2: Comparison of the data encryption time of the proposed encryption scheme with that of the Zhu encryption scheme for varying dimensions with data samples fixed to $n = 100,000$ (2a) and varying number of data samples with data dimension fixed to $d = 10$ (2b).

$1,000,000$ in the step of $100,000$. The behavior of the encryption scheme has been plotted and is shown in figure 2b. From figure 2b, it is clear that with an increase in the number of samples keeping the dimension fixed, both the encryption schemes behave similarly, and the slight difference in the encryption time between the two is caused by the permutation function used in the Zhu encryption scheme.

Query encryption time-varying dimension: The main contribution of this work lies in query encryption; thus, it is most important to show the outcome of the new way of encrypting the query vector. Figure 3 shows the same for varying dimensions in a step of 10. From figure 3, it is clear that the new way of encrypting the query vector significantly reduces the query encryption time compared to the earlier encryption scheme. This is because it involves simple matrix multiplication operations, compared to the operation involved in the Zhu et al. [26], which requires query encryption by the Paillier encryption scheme.

k-NN computation time for varying number of samples and dimensions: Figure 4a and 4b shows the secure k-NN computation time for varying number of samples and dimensions and is also compared with that of the plaintext k-NN computation time. The figures show that k-NN computation in the encrypted environment remains almost similar for both encryption schemes and is slightly higher than that of the plaintext computation time. From figure 4a and 4b, it is clear that the computational overhead of computing k-NN in the encrypted domain is negligible in comparison to the plaintext domain. Thus, the proposed encryption scheme perfectly suits for the practical applications.

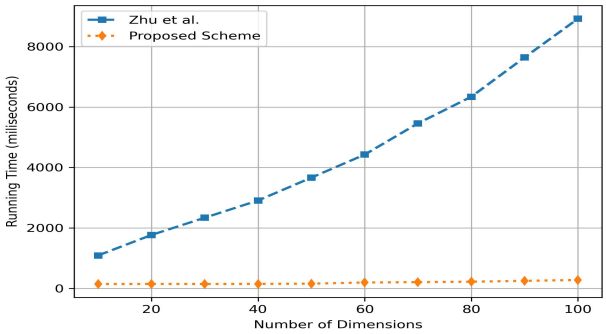
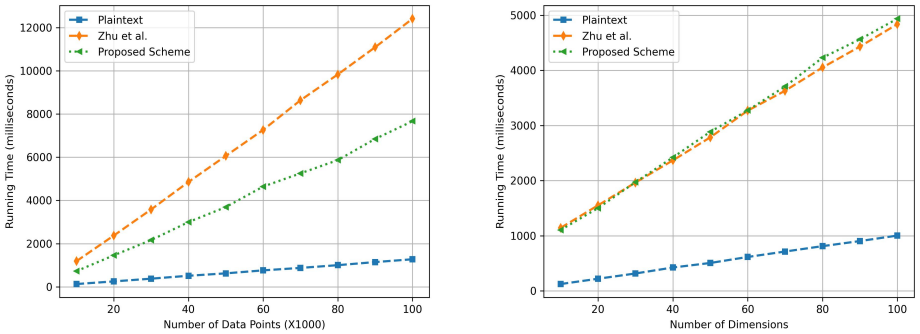


Fig. 3: Query encryption time comparison of the proposed encryption scheme with the Zhu encryption scheme for varying dimensions in a step size 10.



(a) Varying Samples (b) Varying Dimension

Fig. 4: k-nearest neighbor computation time for a varying number of data samples keeping dimension fixed to $d = 10$ (4a) and varying number of data dimensions keeping data samples fixed to $n = 100,000$) (4b) for $k = 20$.

8 Conclusion

The work presented in this paper suggests performing query encryption in an untrusted query user setting of the ASPE encryption scheme without compromising the query privacy of the QU. The new way of performing query encryption replaces the Paillier encryption technique used in the earlier work, which required comparatively higher computation time; as a result, it may not be suggestable for real-world applications. This work tries to fix this issue by removing the Pailler cryptosystem used for query encryption and using the ASPE technique instead. Using the ASPE technique for query encryption increases the overhead of CSP's computation in the cloud, which requires preparing a temporary database for k-NN computation. However, this step can be avoided by directly multiplying

the query obtained from the QU with the temporary matrix and using the obtained vector for k-NN computation. The work of this paper shows that query privacy using ASPE alone is achievable, and the experimental result indicates a considerable reduction in query encryption time. Using the ASPE technique for query encryption instead of the Paillier Cryptosystem performs better and is achievable without compromising the security requirements.

9 Acknowledgments

This work was supported by AI powered adaptive cyber defense framework and solution for national critical information infrastructure, National Cyber Security Council, Government of India.

References

1. Ahmad, A., Ahmad, M., Habib, M.A., Sarwar, S., Chaudhry, J., Latif, M.A., Dar, S.H., Shahid, M.: Parallel query execution over encrypted data in database-as-a-service (daas). The Journal of Supercomputing **75**, 2269–2288 (2019)
2. Bost, R.: $\sum$ oφoς: Forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1143–1154 (2016)
3. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on parallel and distributed systems **25**(1), 222–233 (2013)
4. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.C., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. Cryptology ePrint Archive (2014)
5. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 577–594. Springer (2010)
6. Choi, S., Ghinita, G., Lim, H.S., Bertino, E.: Secure knn query processing in untrusted cloud environments. IEEE Transactions on Knowledge and Data Engineering **26**(11), 2818–2831 (2014)
7. Demertzis, I., Papadopoulos, S., Papapetrou, O., Deligiannakis, A., Garofalakis, M.: Practical private range search revisited. In: Proceedings of the 2016 International Conference on Management of Data. pp. 185–198 (2016)
8. Faber, S., Jarecki, S., Krawczyk, H., Nguyen, Q., Rosu, M., Steiner, M.: Rich queries on encrypted data: Beyond exact matches. In: Computer Security–ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part II 20. pp. 123–145. Springer (2015)
9. Fuller, B., Varia, M., Yerukhimovich, A., Shen, E., Hamlin, A., Gadepally, V., Shay, R., Mitchell, J.D., Cunningham, R.K.: Sok: Cryptographically protected database search. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 172–191. IEEE (2017)
10. Guan, Y., Lu, R., Zheng, Y., Shao, J., Wei, G.: Toward oblivious location-based k-nearest neighbor query in smart cities. IEEE Internet of Things Journal **8**(18), 14219–14231 (2021)

11. Li, H., Yang, Y., Luan, T.H., Liang, X., Zhou, L., Shen, X.S.: Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. IEEE Transactions on Dependable and Secure Computing **13**(3), 312–325 (2015)
12. Li, X., Xiang, T., Guo, S., Li, H., Mu, Y.: Privacy-preserving reverse nearest neighbor query over encrypted spatial data. IEEE Transactions on Services Computing **15**(5), 2954–2968 (2021)
13. Lin, W., Wang, K., Zhang, Z., Chen, H.: Revisiting security risks of asymmetric scalar product preserving encryption and its variants. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). pp. 1116–1125. IEEE (2017)
14. Liu, J., Wang, C., Tu, Z., Wang, X.A., Lin, C., Li, Z.: Secure knn classification scheme based on homomorphic encryption for cyberspace. Security and Communication Networks **2021**, 1–12 (2021)
15. Oh, D., Kim, I., Kim, K., Lee, S.M., Ro, W.W.: Highly secure mobile devices assisted with trusted cloud computing environments. ETRI Journal **37**(2), 348–358 (2015)
16. Raja, J., Ramakrishnan, M.: Confidentiality-preserving based on attribute encryption using auditable access during encrypted records in cloud location. The Journal of Supercomputing **76**, 6026–6039 (2020)
17. Sanyashi, T., Menezes, B.: Secure computation over encrypted databases. CoRR **abs/2308.02878** (2023). https://doi.org/10.48550/arXiv.2308.02878, https://doi.org/10.48550/arXiv.2308.02878
18. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE symposium on security and privacy. S&P 2000. pp. 44–55. IEEE (2000)
19. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. Cryptology ePrint Archive (2013)
20. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security. pp. 71–82 (2013)
21. Wang, B., Yu, S., Lou, W., Hou, Y.T.: Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In: IEEE INFOCOM 2014-IEEE conference on computer communications. pp. 2112–2120. IEEE (2014)
22. Wang, B., Hou, Y., Li, M.: Practical and secure nearest neighbor search on encrypted large-scale data. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. pp. 1–9. IEEE (2016)
23. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. pp. 139–152 (2009)
24. Yu, J., Lu, P., Zhu, Y., Xue, G., Li, M.: Toward secure multikeyword top-k retrieval over encrypted cloud data. IEEE transactions on dependable and secure computing **10**(4), 239–250 (2013)
25. Zheng, Y., Lu, R., Shao, J.: Achieving efficient and privacy-preserving k-nn query for outsourced ehealthcare data. Journal of medical systems **43**, 1–13 (2019)
26. Zhu, Y., Huang, Z., Takagi, T.: Secure and controllable k-nn query over encrypted cloud data with key confidentiality. Journal of Parallel and Distributed Computing **89**, 1–12 (2016)