An Efficient Privacy-Preserving Randomization-Based Approach for Classification Upon Encrypted Data in Outsourced Semi-Honest Environment

Vijayendra Sanjay Gaikwad[1], Kishor H. Walse[2], Mohammad Atique Mohammad Junaid[3]
P.G. Department of Computer Science & Engineering, Sant Gadge Baba Amravati University, Amravati, India[1, 3]
SCTR's PICT, Pune, India[1]
Sant Bhagwanbaba Kala Mahavidyalaya, Sindkhed Raja, Buldana, India[2]

***Abstract*—In cloud environment context, organizations often rely on the platform for data storage and on demand access. Data is typically encrypted either by the cloud service itself or by the data owners before outsourcing it to maintain confidentiality. However, when it comes to processing encrypted data for tasks like *k*NN classification; existing approaches either prove to be inefficient or delegate portion of the classification task to end users or do not satisfy all the privacy requirements. Also, the datasets used in many existing approaches to check the performance seem to have very less attributes and instances; but, it is observed that as dataset size increases, the efficiency and accuracy of many privacy-preserving approaches reduce significantly. In this work, we propose a set of privacy preserving protocols that collectively perform the *k*NN classification with encrypted data in outsourced semi-honest-cloud environment and also address the stated challenges. This is accomplished by building an efficient randomization-based approach called PP*k*C that leverages homomorphic cryptosystem properties. With protocol analysis we prove that the proposed approach satisfies all privacy requirements. Finally, with extensive experimentation using real-world and scaled dataset we show that the performance of proposed PP*k*C protocol is computationally efficient and also independent of the number of nearest neighbours considered.**

***Keywords*—Partial homomorphic encryption; classification using encrypted data; randomization; k- nearest neighbours**

I. INTRODUCTION

The progressive paradigm of information technology known as cloud computing provides the ability to deliver a variety of computing services including processing power, storage, and application platform, on demand. However, security has consistently posed a significant barrier to the general uptake of cloud computing technologies [1], [2]. The problem is further exacerbated by cloud computing service providers' inaccurate reporting of security flaws [3], [4], [5]. Cloud based services have raised the need to protect data privacy in outsourced databases has become a focal point of research. As discussed in study [6], [7], [8] and [9] since, a data owner (DO), contracts out the management of his or her databases to a cloud, the DO can lower database management costs by utilizing the cloud's resources as needed. Owing to the diverse range of users the dataset they offer encompasses multiple ranges of categories including personal health status details [10], back-office user database information [11], email information [12] as well as additional information about individual privacy or company trade secrets [13], [14], [15]. To safeguard the original data, access patterns as well as queries, research has been done on secure query processing over an encrypted database. Earlier approaches in [16], [17], [18], [19], [20] and [21] outsource plain texts to a cloud and alter them with their substituted data. Nonetheless, due to their vulnerability to different types of attacks, these earlier strategies are unable to fully protect both data as well as queries [22].

Consider a scenario where a hospital stores its encrypted patient database on the cloud for data mining tasks. When a doctor seeks to ascertain a patient's symptoms for diagnosis, they must submit a query containing highly personal information. To protect the patient's data privacy, cloud must be queried with only the encrypted query data. Moreover, any anomalies in cloud activity could reveal data access patterns, despite encryption. Therefore, maintaining privacy of involved data is paramount when performing classification tasks on encrypted data in an outsourced environment like the cloud.

Previously many approaches have been introduced to address this challenge, however, either the computation cost required to process the queries turned out to be inefficient or privacy requirements were not completely fulfilled. The previous methodologies predominantly relied on datasets with integer values, limiting their applicability to a narrow range of datasets; however, real-world datasets typically encompass a broader spectrum, often comprising floating-point values. Moreover, the pattern of accessing the data during k-nearest neighbours (k-NN) algorithm is also not safeguarded in [23], [24], [25] and [16]. The privacy-preserving algorithms in [26] and [27] conceal data access patterns while ensuring privacy of outsourced databases as well query. However, they have a high query processing cost as discussed in study [6].

We propose set of protocols that jointly address the privacy-preserving k-nearest neighbours outsourced classification issue with the assumption of the process of classification along with encrypted data is held in the cloud environment. The focus in this paper is only specifically on creating the privacy-preserving k-nearest neighbor technique, since k-NN is a popular and most suitable classifier for this work.

A. Problem Definition

In this paper, we assume a *'m'* dimensional database is possessed by a data holder containing total instances of size *'n'*. 0^{th} attribute serves as the record identifier (I), while the m^{th} attribute represents the class label (c). The data holder encrypts the database attribute by attribute to get $E_{pk}(t_{i,j})$ which denotes the encrypted value of a record, where *'t '*represents a tuple, *'i'* ranges from *1* to *'n'*, and *'j'* ranges from *0* to *'m'*.

E_{pk} is the encryption function of partial homomorphic encryption in [28]. After encryption, the encrypted database is sent to the cloud. After this the data holder doesn't get involved in any of the further privacy-preserving classification steps.

Authentic users can send encrypted queries $E_{pk}(Q)$= $(E_{pk}(q_1), \dots, E_{pk}(q_{m-1}))$ to the cloud to obtain resultant encrypted class label, denoted as $E_{pk}(c_q)$.

B. Our Contributions

We have proposed set of protocols that execute in a two-cloud setup to jointly address the issue of preserving privacy while classifying user's encrypted query using outsourced encrypted data. The protocols are for k-nearest neighbours classification algorithm. The proposed approach offers significantly reduced computational costs by utilizing parallel computing, so as to form more practical grounds for classification of encrypted data. Following are some requirements for privacy preserved outsourced classification:

- The user's query must stay encrypted throughout the entire classification task, ensuring it is not disclosed to the cloud.
- The original contents of the database and any intermediate computations must remain hidden from the cloud.
- The records that correspond to the k-nearest neighbours of the user's query should be kept secret from both the cloud and the user.
- Only the final class label should be disclosed to the user.

This work is inspired by the research of Samanthula, Elmehdwi, and Jiang [29], [30], focusing on enhancing the efficiency of the related sub-protocols. As indicated in [29] regarding potential enhancements to the efficiency of the $SMIN_n$ protocol, the attention in this work is directed towards enhancing execution time needed by it. Paillier cryptosystem [28] faces limitations when confronted with negative values resulting from Paillier addition. This challenge is particularly prominent if there are negative values while computing the encrypted Euclidean distance. It' s noteworthy to mention that in the proposed enhanced set of protocols accomplish all aforementioned requirements. The cloud remains oblivious to which database entries align with the nearest neighbours, and any intermediate data visible to the cloud consists solely of either encrypted or randomized values. Additionally, the resultant label remains undisclosed to both clouds.

The rest of this paper is organized as follows. We provide literature survey of state-of-the-art privacy preserving protocols in Section II. The primitives for building proposed approach are described in Section III. We describe the methodology and our proposed privacy preserving PP*k*C protocol and the sub-protocols as its building blocks along with algorithms in Section IV and in Section V, we explain the privacy analysis of these sub-protocols. In Section VI, we provide the experimental results of our proposed PP*k*C protocol using standard and scaled datasets and its comparative analysis with state-of-the-art privacy preserving protocols. We finally conclude this work in Section VII.

II. Literature Survey

In scenarios where queries for classification are executed on the cloud, the foremost and most critical requirement is to hide query details from cloud.

It is important to note that data mining operations can be conducted on encrypted data with relatively less effort using fully homomorphic encryption (FHE) introduced by Gentry et al. (2009) [31]. This cryptosystem allows arbitrary functions to be performed on encrypted data without decryption. However, FHE is computationally intensive, making it impractical for handling real-time classification requests.

Handling queries on encrypted data without the cloud decrypting it poses a significant challenge. The work by Samanthula et al. (2014) [30] outlines a collection of protocols designed to jointly solve the *k*-nearest neighbours (k-NN) query issue within an encrypted database, where both the data and the classification tasks are delegated to the cloud. As described in study [30], the objective of the secure *k*NN protocol is to find the top *k* records closest to the user's query while keeping all details hidden from the cloud. However, in this approach, the SkNN protocol in [30] results in the cloud being exposed to intermediate data, such as the calculated distance values and the subsequently determined *k* smallest distance values. In addition, the records associated with the *k* nearest neighbours of the user query are disclosed to the cloud and also exposed to the user. This again compromises the privacy of the database entries involved in the classification process.

According to Samanthula et al. (2015) in [29] ensuring privacy in *k* nearest neighbours classification is more challenging than running basic *k*NN queries on encrypted data. This complexity stems from the requirement that the k-nearest neighbours identified during the classification process must remain confidential from querying user and cloud.

Protocol presented in study [29] overlooks the issue of access patterns, which is a critical privacy concern for users. While the protocols in study [30] introduce a secure method for *k* nearest neighbours classification on encrypted data that safeguards data and user query privacy, they fail to conceal the data access patterns. Samanthula et al., in [29], expanded on their previous work from study [30] by introducing the PPkNN protocol, offering a new approach to the privacy preserved *k*-nearest neighbours classification issue.

The *k* nearest neighbours remain hidden from both the user and cloud in the PPkNN protocol [29]. Secure Minimum

protocol (SMIN) in study [30] is used to determine the nearest neighbours in a privacy preserved way. As proposed by Samanthula et al. in [29], the SMIN sub-protocol consumes nearlt 67% of the total processing time taken by the PPkNN protocol, which is significantly high and hence impractical in real-time scenario. So, the prevailing obstacle in any outsourced privacy preserving classification approach is to tackle this excess processing time which is caused due to the fact that protocols have to work upon encrypted values. A reduction in overall computational cost would bring even practical solution for the outsourced privacy preserving classification tasks.

As mentioned by Zhu et al. (2020) in [32], conventional privacy preserving approaches for kNN classification induce huge computational cost since operations are purely conducted on encrypted values. Park et al. (2020) in [33] have proposed an efficient version of the Samanthula et al.'s (2015) work in [29] by designing the PkNC protocol which executes its component protocols in parallel to find the class label.

Experimental results depict the gradual rise in execution time of the PkNC protocol. However, it decreases substantially regardless of *k* (the number of nearest neighbours). But, the primary drawback of this protocol emerges as the dataset increases. The execution time rises again in linear manner with increasing instances in dataset, largely because of the practical limitations on threading for parallel computation. Liu et al. (2020) [34] have investigated training of decision trees in outsourced environment. They asserted that encrypted dataset cannot be divided by the cloud based on best attribute and hence, they have proposed a new method which is splitting-free decision tree training.

However, the prominent defining factor is that problem domain of this work deals with k-NN classification, which does not require a separate training phase that is needed for decision tree classification. Also, the scheme proposed by the authors uses an additive secret-sharing method for privacy preservation. This induces more computational cost with the inclusion of share reconstruction. Moreover, experimentations show that the designed protocols' cost of communication and computation rises along with length of vector. Noteworthy observation is that protocols in study [34] were evaluated on relatively small datasets sizes, containing 24, 100, 120, and 958 instances, respectively. The protocols proposed in study [35] and [36] are more efficient than the pioneer protocol in [29], but they result in class labels corresponding to *k* nearest records instead of providing the final class of query. Thus, this is not the exact expected result for outsourced privacy-preserving *k*NN classification issue [37]. It also reveals the *k* nearest class labels to the querying user, which does not satisfy the privacy requirements as stated in study [29]. Moreover, it cannot protect all the intermediate information. The distribution of the inner products, which is used to describe the distance between two vectors like the Euclidean distance, is leaked to one of the cloud servers.

Examining the proposed scheme for a variety of datasets is an important experimental step toward deciding the range of classification tasks that potentially can be performed by any privacy-preserving approach. Until now, all of the existing schemes have experimented with only integer datasets (i.e. they can handle only integer values). This is the reason that in [38], Du et al. scheme's accuracy is not good enough for the real number datasets. In fact, the accuracy of this scheme severely drops with this Heart Disease dataset.

A privacy-preserving k-NN query scheme (QS) based on a secure multi-party computation mechanism was modeled by Xian Guo et al. in [3] to address security concerns when malicious attackers control the cloud and query users. This method showed that the scheme has a certain degree of feasibility and reliability. Furthermore, this method showed a better solution for privacy protection and security. However, a privacy-preserving k-NN query scheme was limited in real-world applications.

Hyeong-Jin Kim et al. in [6] implemented a privacy preserving k-NN query processing algorithm (QPA) via secure two-party computation based on encrypted data. In terms of query processing cost, the performance of the model was better than the existing methods. Nevertheless, the model did not solve other types of queries including Top-*k* and k-NN classification due to low-dimensional data. In addition, this model required increased computational cost. Developing high-dimensional data space requires a data dimensionality reduction technique which led to a challenging task.

Zhi Li et al. [39] presented a function secret sharing (FSS) based secure multi party kNN classification scheme (SecKNN). For secure computations, the presented scheme offered low computation and communication cost. The implementation of FSS reduced lot of computational overhead. Nonetheless, the deployment of the scheme on real time applications is limited.

A privacy-preserving kNN query scheme (QS) was employed by Yandong Zheng et al. [10] to return accurate query results and high query efficiency. The scheme achieved low computation cots and the max-heap accelerated the query efficiency. However, the kNN query scheme leaked the relative proximities of various data records.

Hyeong-Jin Kim et al. [22] employed a new Top-*k* query processing algorithm based on a homomorphic encryption system that is efficient and provides security also. Compared with the existing methods, the Top-k algorithm achieved more times better performance concerning query processing time. However, this model was only performed in specific privacy-preserving data mining algorithms.

III. PRIMITIVES

A. Synthetic Minority Over-sampling Technique (SMOTE)

SMOTE, introduced by Chawla et al [40] in 2002, addresses imbalanced class issues in machine learning. By synthesizing minority class samples through interpolation among existing instances, it counters bias favoring majority classes. Randomly selecting instances, it identifies *k* nearest neighbours and generates synthetic examples along the connecting line segments. This method improves the capacity of classifier for making accurate predictions by providing robust coverage of the minority class space. SMOTE generates synthetic data points, reducing model bias towards

the majority class. Particularly beneficial for imbalanced datasets where the minority class is underrepresented. Interpolation among existing minority class instances maintains diversity. Contrasts with simpler methods like random oversampling that may lead to overfitting. SMOTE introduces new instances near original minority class data points. Synthetic samples serve as plausible representations, reducing overfitting risk and aiding model generalization.

B. Paillier Cryptosystem

Paillier cryptosystem [28] is additively homomorphic and allows calculations upon encrypted values directly, eliminating the requirement to decrypt. It's extensively employed in safeguarding privacy, especially in situations requiring the secure processing of sensitive information while maintaining confidentiality. This scheme is capable of providing semantic security. Fundamentally, Paillier encryption operates on principles of modular operations and large prime numbers. Its security hinges on computing difficulty of factoring the product of two large prime numbers.

Consider *Epk* as encryption function associated with a public key *pk* represented by (N, g), where N is the product of two large prime numbers, and g is a generator in $Z^*_{N^2}$. Similarly, *Dsk* is the decryption function corresponding to the secret key *sk*. Following properties of Paillier encryption scheme [28] withstand for any two plaintext values a and b belonging to Z_N:

1) Homomorphic addition: It provides the addition operation on encrypted values, producing a sum which is also encrypted.

$$D_{sk}\left(E_{pk}(x+y)\right) = D_{sk}\left(E_{pk}(x) * E_{pk}(y)\ mod\ N^2\right) \quad (1)$$

2) Scalar multiplication homomorphism: It provides multiplication operation *x*y* and yields $E_{pk}(x*y)$ when an encrypted value $E_{pk}(x)$ is raised to the power with a scalar value *y*.

$$D_{sk}\left(E_{pk}(x * y)\right) = D_{sk}\left(E_{pk}(x)^y mod\ N^2\right) \quad (2)$$

IV. Methodology for Privacy Preserved Classification

This section elaborates on the operations of several sub-protocols that serve as the foundational components for enhancing the efficiency of computing the *k* nearest neighbours. By performing all operations on Cloud Server 1 (C_1) and utilizing Cloud Server 2 (C_2) for specific tasks with randomized and shuffled data, a robust privacy-preserving architecture can be established. As shown in Fig. 1, we operate within a genuine-but-curious scenario, where the two involved cloud platforms $C_1$and C_2 are non-colliding and adhere strictly to the protocol specifications. C_2 hosts *sk* (secret key) and does not share it with anyone whereas C_1 , C_2 and the querying user know the public key *pk*.

User data is encrypted and stored securely on C_1. *k*NN operations, including distance calculation and classification, are performed entirely on C_1, ensuring that sensitive information remains within a single secure environment.

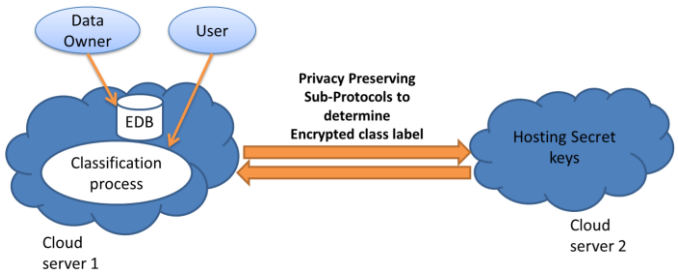

Fig. 1. Two cloud architecture setup.

For operations requiring additional computational resources, such as multiplication or comparison, C_1sends encrypted data in a randomized format to C_2. This randomized data prevents the exposure of sensitive information during the transmission and execution of these operations on C_2. Upon receiving the encrypted data, C_2 performs the necessary operations and returns the results to C_1. The results are de-randomized on C_1, ensuring that the true outcomes are obtained without compromising on data privacy and confidentiality.

The utilization of *k*NN within this two-cloud architecture facilitates privacy-preserving data mining operations in cloud environments. By encrypting data and performing all encrypted operations on C_1, sensitive information remains protected. Role of C_2 is specific to performing operations on randomized and shuffled data therefore, minimizing the risk of data breaches and unintended knowledge gain. The secure interaction between $C_1$and C_2 ensures that privacy is maintained throughout the outsourced classification process.

A. Privacy Preserving Euclidean Distance Protocol

Encrypted squared Euclidean distance is computed by this protocol. These encrypted distances are determined by computing difference between an encrypted user query, $E_{pk}(q)$ and each encrypted dataset instance, denoted as $E_{pk}(t_i)$. Here, i is between 1 and *n*, and *n* represents total instances in dataset. Both $E_{pk}(t_i)$ and $E_{pk}(q)$ have *m* number of attributes. C_2 holds *sk* (secret key) and does not share it with anyone where as C_1 , C_2 and the querying user know the public key *pk*.

The protocol employs parallelization via multiprocessing to expedite results. Additionally, it employs randomization for all intermediate operations necessitating interactions with C_2. In this process, the encrypted dataset values are randomized with an encrypted random number using the Paillier additive property [28], and the obtained result is later de-randomized. This approach guarantees that even if certain data points undergo decryption on C_2, they are presented in a randomized manner, thereby preventing complete exposure of any data point to C_2.

Algorithm-A: PPED(E_{pk}(X),E_{pk}(Y))→$\{E_{pk}(d_1),..,E_{pk}(d_n)\}$

Require: C_1 has E_{pk} (X) and E_{pk} (Y); C_2has *sk*

On C_1:

1. for *i = 1* to *n* do
2. for *j = 1* to *m* do
3. $E_{pk}\ (x_{ij} - y_j) \leftarrow E_{pk}\ (x_{ij}) * E_{pk}\ (y_j)^{N-1}$ mod N^2

(parallelization is used to compute attribute-wise differences concurrently)

4. end for
5. Generate random number $r \in Z_N$
6. $u \leftarrow \sum_{j=1}^{m} E_{pk}\ (x_{ij} - y_j)$
7. send $R \leftarrow u * E_{pk}\ (r) \bmod N^2$ to C_2

On C_2:

8. $u' \leftarrow D_{sk}(R)$
9. $v \leftarrow u' * u' \bmod N$
10. $v' \leftarrow E_{pk}\ (v)$
11. send v' to C_1

On C_1:

12. $r' = r * r$
13. $p = u^{2r} \bmod N^2$
14. $p' \leftarrow v' * E_{pk}(r')^{N-1} \bmod N^2$
15. $E_{pk}(d_i) \leftarrow E_{pk}\ ((x_i - y)^2) \leftarrow p' * p^{N-1} \bmod N^2$

(parallelization is used to concurrently compute the encrypted squared Euclidean distances for all instances)

16. end for
17. return $\{E_{pk}(d_1), \ldots, E_{pk}(d_n)\}$

A vector $E_{pk}(Y)$ having user's encrypted query attributes, and a data-frame E_{pk} (X) having attribute-wise encrypted instances from the dataset are used as inputs for the PPED protocol. Using PPED protocol, C_1 and C_2 jointly compute a vector having encrypted distances corresponding to each encrypted dataset instance.

This protocol implements parallelism in two phases. In the first phase the attribute-wise difference between each attributes of the user query and corresponding attribute of a given dataset instance is computed concurrently. In the second (outer) phase, the encrypted squared difference $E_{pk}(d_i)$ (i. e. $E_{pk}((x_i - y)^2)$) is concurrently computed for all instances of the dataset. During implementation this concurrency is achieved by using threading in Python. Thus, efficiency is improved significantly through these concurrent executions. Moreover, with additional computing resources such as multi-core processors, the PPED protocol can also be run in parallel to simultaneously compute the encrypted squared differences. We use parallelization for independently computing the attribute-wise differences for each i^{th} instance and also for overall execution of PPED algorithm across n instances as each squared distance can be independently computed for all n instances. At the end of PPED protocol, the resulting vector of encrypted distances is only available to C_1 however, these distances remain encrypted.

B. Privacy Preserving Shuffle Protocol (PPSP)

In order to get the k nearest neighbours for the encrypted user query, we need to find the k minimum encrypted distances from amongst the vector of encrypted squared differences generated by PPED protocol. Since, these distance values are encrypted, the k minimum distances cannot be found directly by C_1. The immediate solution is to send the vector of encrypted squared differences to C_2 so that C_2 can decrypt them with the secret key and then the squared differences could be compared in plaintext to get the required k minimum distances. However, this will reveal all the original distance values to C_2 and C_2 also gets to know which dataset records correspond to the selected k minimum distances. So, another solution which avoids this data leakage issue is to randomize the original distances at C_1 using additive homomorphic property and then send these randomized encrypted squared differences to C_2 for comparison. This solves the data leakage issue as the original distances are not revealed to C_2 but C_2 still gains knowledge about which k records from the dataset are selected as the nearest neighbours.

Algorithm-B: PPSP (D) → D'

Require: C_1 has $D \leftarrow \{E_{pk}\ (d_1 + r), \ldots, E_{pk}\ (d_n + r)\}$

On C_1:

1. $n \leftarrow$ *length_of (D)*
2. for $i = n$-1 to 1 do
3. Generate a random integer j , where $0 \leq j \leq i$
4. $temp = D[i]$
   $D[i] = D[j]$
   $D[j] = temp$
5. end for
6. return $D' \leftarrow$ *securely permuted vector of randomized encrypted squared differences*

Although, $C_1$and C_2 are non-colliding but revealing such data access patterns to C_2 can be potentially malicious. Hence, we propose a privacy preserving shuffle protocol (PPSP) that performs random permutation with the randomized encrypted squared differences before sending them to C_2 for comparison. Since, PPSP applies the random permutation directly on encrypted data, at the end of the protocol C_1 does not gain any information about the randomized encrypted squared differences. Moreover, C_2 remains unaware about the sequence of the randomized encrypted squared differences it receives from C_1 for comparison, as intermediate steps are not revealed. Thus, by utilizing proposed PPSP protocol before determining the k minimum encrypted distances, C_2 does not gain any knowledge about which k records from the dataset get selected as the target nearest neighbours and data access patterns are preserved.

C. Privacy Preserving k-Minimum Distances (PPkMD) Protocol

The objective of PPkMD protocol is to determine the encrypted k nearest neighbours of the encrypted user query such that the corresponding original dataset records are not revealed to either of the clouds (i.e.C_1 or C_2). Also, the intermediate results must be either encrypted or such that they must not lead C_1 or C_2 to gain any knowledge about the original values to avoid disclosure of data access patterns. The protocol takes a vector 'v' as input and each element of v is an object having three encrypted values namely, encrypted dataset record identifier $E_{pk}(id_i)$, encrypted distances (i.e. squared difference) $E_{pk}(d_i)$ and corresponding encrypted class label $E_{pk}(c_i)$. So, $v = \{(E_{pk}(id_1), E_{pk}(d_1), E_{pk}(c_1)), \ldots, (E_{pk}(id_n), E_{pk}(d_n), E_{pk}(c_n))\}$

The protocol begins with C_1 generating random integer r from Z_N, and randomizes vector v by homomorphically adding $E_{pk}(r)$ to $E_{pk}(id_i)$, $E_{pk}(d_i)$ and $E_{pk}(c_i)$ of each element of v This gives us an encrypted randomized vector, v'. This v' vector is shuffled using $PPSP_m$ protocol before it is transmitted to C_2. $PPSP_m$ is a variant of above proposed PPSP protocol used for shuffling the vector v', where $v'[i]$= $(E_{pk}(id'_i), E_{pk}(d'_i), E_{pk}(c'_i))$ and $1 \leq i \leq n$. Thus, $PPSP_m$ receives v' and shuffles it such that position of each element $v'[i]$ is changed. Then, the shuffled vector V is sent to C_2.

C_2 decrypts V to get a plaintext but randomized vectors V'. Now, C_2 constructs a min-heap with the randomized distances, $V'[i].d'$, where $1 \leq i \leq n$ and each node in the heap comprises of ($V'[i].id'$, $V'[i].d'$, $V'[i].c'$) i.e. an element of vectors V'. Since, the randomized distance value present at the root node of the min-heap is always the smallest value, we pop the root node from the heap to get the first amongst the k nearest neighbours. Similarly, we pop the heap to get all the remaining nearest neighbours and store them in vector K_{min}. As the min-heap is built with randomized distance values and each node in the heap structure has only randomized values, C_2 does not gain any information about original values. Also, since the randomized elements are already shuffled, therefore C_2 neither learns about their order nor it is able to determine which k records from the dataset were selected as the nearest records to the query. Moreover, the identifiers are also randomized and shuffled which avoids revealing any data access patterns. C_2 then encrypts the vector K_{min} to get encrypted vector K_{min}' and sends it to C_1.

Then, C_1 de-randomizes K_{min}' using the paillier additive property [28] to get the original encrypted k nearest elements in vector v_{min}. Since, every step at C_1 involves only encrypted data, no information is revealed to C_1.

Algorithm-C: PPkMD (v) → v_{min}

Requires: C_1 holds v= $\{(E_{pk}(id_1),E_{pk}(d_1),E_{pk}(c_1)),\ldots, (E_{pk}(id_n), E_{pk}(d_n), E_{pk}(c_n))\}$ and C_2 holds the secret key sk.

On C_1:

1. Generate random integer $r \in Z_N$ and encrypt it, $E_{pk}(r)$
2. Build randomized vector v' by adding $E_{pk}(r)$ to each element of v :
3. for i = 1 to n do
4. $E_{pk}(id'_i) = E_{pk}(id_i) * E_{pk}(r) \bmod N^2$
5. $E_{pk}(d'_i) = E_{pk}(d_i) * E_{pk}(r) \bmod N^2$
6. $E_{pk}(c'_i) = E_{pk}(c_i) * E_{pk}(r) \bmod N^2$
7. end for
8. So we have, encrypted randomized vector as, v'= $\{(E_{pk}(id'_1),E_{pk}(d'_1),E_{pk}(c'_1)),\ldots,(E_{pk}(id'_n), E_{pk}(d'_n), E_{pk}(c'_n))\}$
9. $V \leftarrow PPSP_m(v')$ to shuffle all elements in v'
10. Send shuffled vector V to C_2

On C_2:

11. Decrypt all elements in V using sk such as,
12. for i = 1 to n do
13. $V'[i] \leftarrow (D_{sk}(V[i].id'), D_{sk}(V[i].d'), D_{sk}(V[i].c'))$
14. end for
15. Construct a *min-heap* based on randomized distances $V'[i].d'$, where $1 \leq i \leq n$ and each node in the heap comprises of ($V'[i].id'$, $V'[i].d'$, $V'[i].c'$)
16. for i = 1 to k do
17. $K_{min}[i] \leftarrow$ pop the root node from the *min-heap*
18. end for
19. Vector K_{min} is encrypted using pk such as,
20. for i = 1 to k do
21. $K_{min}'[i] \leftarrow (E_{pk}(K_{min}[i].id'), E_{pk}(K_{min}[i].d'), E_{pk}(K_{min}[i].c'))$
22. end for
23. Send encrypted vector K_{min}' to C_1

On C_1:

24. Receive K_{min}' from C_2 and get the de-randomized vector v_{min} :
25. for i = 1 to k do
26. $E_{pk}(id_{\min\_i}) = K_{min}'[i].id' * E_{pk}(r)^{N-1} \bmod N^2$
27. $E_{pk}(d_{\min\_i}) = K_{min}'[i].d' * E_{pk}(r)^{N-1} \bmod N^2$
28. $E_{pk}(c_{\min\_i}) = K_{min}'[i].c' * E_{pk}(r)^{N-1} \bmod N^2$
29. end for
30. So, we have the encrypted k nearest neighbours in v_{min} as,
31. return v_{min} = $\{(E_{pk}(id_{\min\_1}),E_{pk}(d_{\min\_1}),E_{pk}(c_{\min\_1})),\ldots, (E_{pk}(id_{\min\_k}), E_{pk}(d_{\min\_k}), E_{pk}(c_{\min\_k}))\}$

D. Privacy Preserving Frequency Counting (PPFC) Protocol

The list of encrypted class labels of the dataset is also outsourced to C_1 along with the *EDB*. V=($E_{pk}(c_1)$,…., $E_{pk}(c_w)$) denotes the encrypted class labels list held by C_1.Hence, we have definite class labels (i.e. w). Also, at the end of PPkMD protocol, C_1 holds the encrypted class labels corresponding to the k- nearest records. Let these class labels be denoted as, U= ($E_{pk}(c_1)$,…., $E_{pk}(c_k)$). The goal of PPFC protocol is to compute the frequency of occurrence for each class label in *EDB* in privacy preserved manner i.e. $E_{pk}(f(c_j))$, where $1 \leq j \leq w$.

$PPSP_f$ is another variant of the proposed PPSP protocol used for shuffling vector G_i, where $1 \leq i \leq k$, which comprises of encrypted randomized class label difference values. Then, the shuffled matrix G' is sent to C_2. This ensures that C_2 does not learn anything about which randomized difference value in G' corresponds to which class label. The inverted matrix I received from C_2 is then de-shuffled using the reverse permutation protocol $PPSP'_f$. Finally, the column-wise homomorphic addition of matrix I' gives the encrypted occurrence frequency of class label c_j , where $1 \leq j \leq w$.

Algorithm-D: PPFC (U,V) →F=$\{E_{pk}(f(c_1)),\ldots E_{pk}(f(c_w))\}$

Requires: C_1 holds U= $\{E_{pk}(c_{\min\_1}),..,E_{pk}(c_{\min\_k})\}$ and V= $\{E_{pk}(c_1),..,E_{pk}(c_w)\}$

On C_1:

1. for $1 \leq i \leq k$
2. for $1 \leq j \leq w$
3. $G_{i,j} = E_{pk}(c_j) * E_{pk}(c_{\min\_i})^{N-1}$
4. Generate a random integer $t_{i,j} \in Z_N$
5. $G'_{i,j} = G_{i,j}{}^{t_{i,j}}$
6. end for
7. $G'_i \leftarrow PPSP_f(G'_i)$ to shuffle all elements of G'_i
8. end for
9. Send shuffled matrix G' to C_2

On C_2:
10. for $1 \le i \le k$
11. for $1 \le j \le w$
if $D_{sk}(G'_{i,j}) = 0$
$I_{i,j} = E_{pk}(1)$
otherwise, $I_{i,j} = E_{pk}(0)$
end if
12. end for
13. end for
14. Send matrix I to C_1

On C_1:
15. for $1 \le i \le k$
16. $I'_i \leftarrow$PPSP$'_f(I_i)$ to de-shuffle all elements of I_i
17. end for
18. for $1 \le j \le w$
19. $F[j] \leftarrow E_{pk}(f(c_j)) = \prod_{i=1}^{k} I'_{i,j}$
20. end for

E. Privacy Preserving Max-Frequency (PPMF) Protocol

The PPFC protocol yields a vector *F*, that has encrypted occurrence (counts) frequencies of all class labels of given dataset. The objective of PPMF protocol is to determine which these encrypted frequency values is the largest. The class label corresponding to the largest encrypted randomized frequency will be the final class label for the user's query. So, PPMF protocol can be similar to the proposed PP*k*MD protocol where C_1 prepares a randomized version of the vector F, shuffles it using another suitable variant of PPSP protocol and sends it to C_2; each element is a pair of randomized encrypted class label and corresponding randomized encrypted frequency $<E_{pk}(c_j + r), E_{pk}(f(c_j) + r)>$, where $1 \le j \le w$. C_1 also sends the randomizing factor, *r*, to the querying user at this stage. C_2 decrypts the received vector and now builds a max-heap based on the decrypted but randomized and shuffled frequency values. Each node in the max-heap comprises of the randomized class label and its corresponding randomized frequency value. Since, the randomized frequency value present at the root node of the max-heap is always the largest value; we pop the root node from the heap to get the maximum frequency and its corresponding class label, both randomized. This is the final but randomized class label, $(c_{final} + r)$ for the user query. C_2 sends this randomized final class label to the querying user.

F. Privacy Preserving k-NN Classifcation (PPkC) Protocol

This protocol serves as the base protocol for performing outsourced k-NN classification. It utilizes the above proposed privacy preserving protocols as building blocks for classifying user queries. The querying user is expected to encrypt all query attributes $(E_{pk}(y_1), \ldots, E_{pk}(y_m))$ and send it to C_1. Let us suppose that the encrypted database (EDB) at C_1 is denoted by E_{pk}(X) and the encrypted user query is E_{pk}(Y). With these inputs, the PP*k*C protocol starts its execution.

Algorithm-E: PP*k*C $(E_{pk}(X), E_{pk}(Y)) \rightarrow (E_{pk}(c_1), .., E_{pk}(c_k))$

Requires: C_1 has E_{pk}(X) and receives E_{pk}(Y)
1. $\{E_{pk}(d_1), .., E_{pk}(d_n)\} \leftarrow$ PPED$(E_{pk}(X), E_{pk}(Y))$
2. $v = \{(E_{pk}(id_1), E_{pk}(d_1), E_{pk}(c_1)), \ldots, (E_{pk}(id_n), E_{pk}(d_n), E_{pk}(c_n))\}$
3. $v_{min} \leftarrow$PP*k*MD (v)
4. $U = \{E_{pk}(c_{\min\_1}), .., E_{pk}(c_{\min\_k})\}$ and
5. $V = \{E_{pk}(c_1), .., E_{pk}(c_w)\}$
6. $F \leftarrow$ PPFC (U,V)
7. for $1 \le i \le w$
8. $F'[i] = < E_{pk}(c_i), E_{pk}(f(c_i)) >$
9. User receives $(c_{final} + r) \leftarrow$ PPMF (F')
10. With *r* received from C_1, the user computes the final class label as, $(c_{final} + r) - r$

At the end of PPMF protocol, the querying user receives the final randomized class label, $(c_{final} + r)$ from C_2. With r received from C_1, the user de-randomizes $(c_{final} + r)$ to determine the final class label, c_{final}.

V. PROTOCOL ANALYSIS

Although we focus on building an efficient approach for outsourced *k*NN classification, in this section we also highlight the effectiveness, security and total privacy preservation provided by the proposed protocols. We have guaranteed that the final result of all the proposed privacy preserving protocols remains encrypted. Additionally, C_2 works with only random and shuffled values that bear no connection to the original data. Furthermore, all computations performed on C_2 are consistently sent back to C_1 in encrypted format. Hence, at no stage the original data is revealed to C_1 or C_2. The complexities of the proposed protocols employed in the proposed PP*kC* protocol is presented in Table I.

TABLE I. COMPLEXITIES OF PROPOSED PROTOCOLS

Protocols	PPED	PPSP	PP*k*MD	PPFC
Complexity	O(*m logN*)	O(*n*)	O(*k logn*)	O(*kwn*)

A. PPED Analysis

PPED protocol is carried out with randomization to prevent the sum of attribute-wise differences (i.e. *u*) from getting revealed at C_2. The sum of attribute-wise differences is randomized using the Paillier additive property [28] at C_1 and the randomized value is sent to C_2 for squaring. C_2 decrypts the randomized value, computes square of the randomized values and then encrypts the square again before sending it to C_1. C_1 receives the encrypted square of randomized value and de-randomizes the square through mathematical formulae and homomorphic properties to get the encrypted square of the original *u* value. Hence, the resulting encrypted square of *u* is only available to C_1. Also, the randomizing factor *r* is only known to C_1 so, C_2 does not gain any information about the original value of *u*.

Parallelization is employed in two phases; firstly for computing the attribute-wise difference of each i^{th} instance and also for independently computing the *n* squared distances. With parallelization, assuming ideal conditions, these

computations can be done in O(m) time for the attribute-wise difference and O(n) time for the actual squared distance. The de-randomizing step takes O($nlogN$) time across all n instances, N being the . So with parallelization, the overall complexity could be reduced to O($mlogN$) if the operations are fully parallelized.

B. PPkMD Analysis

The encrypted vector v is randomized with an encrypted random factor $E_{pk}(r)$ and then shuffled using $PPSP_m$ protocol to get vector V at C_1. C_2 then builds the min-heap with the decrypted distances but they are randomized and shuffled distance values so, C_2 does not gain any information about the original distances and access patterns since C_2 takes decisions based on randomized and shuffled values and hence, no extra information is leaked at C_2. Once C_2 determines the randomized k nearest neighbours, C_2 encrypts them and sends to C_1. Now, since paillier cryptosystem is semantically secure, the cipher texts received by C_1 are not the same as the ones which were sent to C_2. Hence, C_1 cannot determine which k distances amongst the sent n distances are received as nearest distances. Ultimately, no knowledge is acquired by C_1 and C_2 about the original data during PPkMD protocol. Since, min-heap is built with O($logn$) and k minimum neighbours are to be extracted, the overall complexity of the PPkMD is O($klogn$).

VI. Experimental Results and Performance Analysis

The various experiments were performed in Google Colab environment with an Intel® 2 GHz system having 4 cores, RAM of 8 GB and 3 MB Cache Size. We utilized the python homomorphic encryption (i.e. phe) library for implementing the Paillier cryptosystem in [28] which is required in the proposed protocols. Existing privacy preserving solutions also use the same cryptosystem so, the performance of the proposed PPkN protocol can be easily compared with them.

We compare the performance of proposed PPkC protocol with the recent and state-of-the-art work in [33], [34], [35], [41] and [43]. The state-of-the-art privacy preserving solutions in [33], [34] and [35] have used datasets the same UCI KDD archive's Car Evaluation dataset [42] in their experimentation. To clearly showcase the improved performance of the proposed PPkC protocol, we firstly conducted experimentation using the same Car Evaluation dataset [42] having 1728 records and six attributes.

For extensive performance evaluation of proposed PPkC protocol with huge dataset and comparison with the most recent work in studies [41] and [43] we used a suitable scaled version of the Car Evaluation dataset in experimentation having 10000 records and six attributes. All the above mentioned recent solutions have used different hardware specifications while performing the performance evaluation of their privacy preserving protocols.

We encrypted both datasets firstly keeping the key size as 512 bits and then as 1024 bits (i.e. K=512/1024), and also varied the values for the nearest neighbours i.e. k and the number of records i.e. n to evaluate the performance of proposed PPkN protocol.

A. Experimental Analysis

1) Performance of proposed protocols with varying key size (K): The below figures illustrate the execution time (in seconds) required by each of the proposed component protocols namely PPED, PPkMD, PPFC, PPMF along with the total execution time taken by PPkC while using datasets encrypted with key size (K) as 512 and 1024 and k= 5. The execution time of PPSP is inclusive in time taken by above mentioned protocols. The protocols are listed on the x-axis, while the y-axis represents the execution time in seconds.

Fig. 2 illustrates the execution time required by all protocols when the Car Evaluation dataset (dataset-1) is encrypted with a key size of 512 and 1024 bits. It is clearly observed that the PPED requires more time as compared to other component protocols with relatively insignificant time requirements.

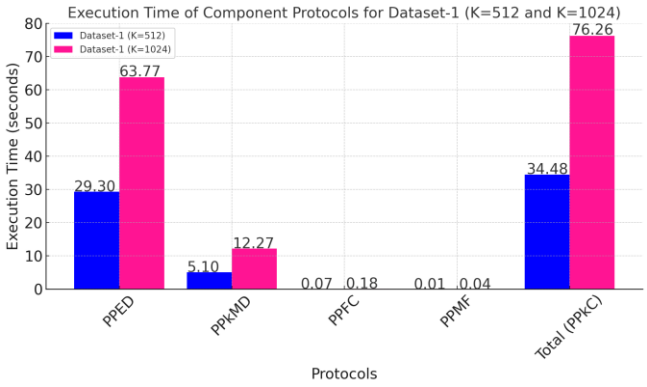

Fig. 2. Execution time of proposed protocols with encrypted Car Evaluation dataset (K= 512, 1024, n=1724).

Fig. 3 illustrates the execution time required by all protocols when the scaled Car Evaluation dataset (dataset-2) is encrypted with a key size of 512 and 1024 bits. As the number of data records are more in this scaled dataset, the execution time taken by all the component protocols is relatively more. However, the growth in time required by PPkMD, PPFC, PPMF protocols is insignificant.

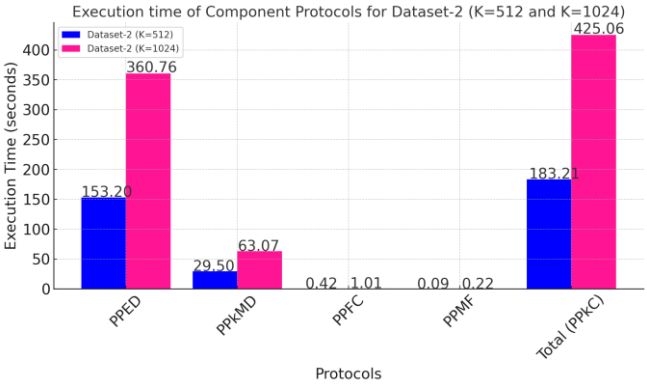

Fig. 3. Execution time of proposed protocols with encrypted Scaled Car Evaluation dataset (K= 512, 1024, n=10000).

TABLE II. SUMMARY OF EXECUTION TIME (SEC) OF COMPONENT PROTOCOLS WITH BOTH DATASETS

Protocols	PPED	PP*k*MD	PPFC,	PPMF	Total (PP*k*C)
Dataset-2 (K= 512)	153.2	29.5	0.42	0.09	183.21
Dataset-2 (K= 1024)	360.76	63.07	1.01	0.22	425.06
Dataset-1 (K= 512)	29.3	5.1	0.07	0.01	34.48
Dataset-1 (K= 1024)	63.77	12.27	0.18	0.04	76.26

Table II shows the summary of execution time incurred by all proposed component protocols during the user query classification with both Car Evaluation dataset and its scaled version having 1724 and 10000 records, respectively, encrypted under key sizes (K) of 512 and 1024 bits.

TABLE III. DATA TRANSFERRED DURING QUERY CLASSIFICATION USING PP*k*C

K (bits)	Data Transferred (MB)				
	k= 5	*k*= 10	*k*= 15	*k*= 20	*k*= 25
512	18.366	25.827	33.355	40.822	48.0
1024	35.587	50.217	64.834	79.474	94.224

Table III shows the data transferred (in Megabytes) during user query classification using the proposed PP*k*C protocol in the two cloud setup with the Car Evaluation dataset encrypted under key sizes (K) 512 and 1024 bits. The table presents the data transferred for various values of *k* i.e. number of neighbours considered.

2) Performance of proposed protocols with varying number of nearest neighbours (k): We examined the execution time required by each of the proposed component protocols and also the total execution time taken by PP*k*C algorithm while varying values of number of neighbours (i.e. *k*) with encrypted Car Evaluation dataset using key size of 512. Across all component protocols a consistent pattern of constant execution times is maintained as the value of *k* is increased from 5 to 25. Hence, the total execution time of proposed PP*k*C protocol remains almost constant when *k* is changed from 5 to 25. Fig. 4 shows this merit of consistent pattern of constant execution times for all proposed protocols while varying *k*.

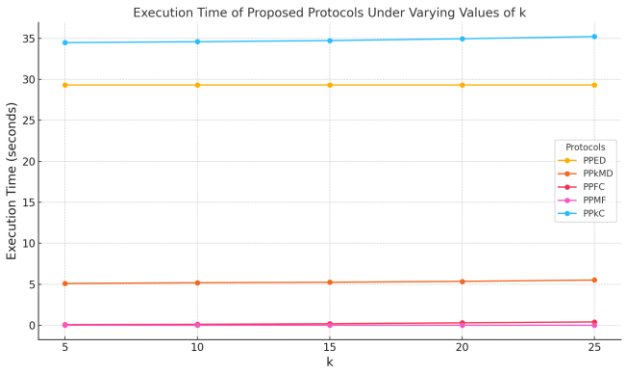

Fig. 4. Constant execution time of proposed protocols under varying values of *k*.

TABLE IV. SUMMARY OF EXECUTION TIME (SEC) OF PROPOSED COMPONENT PROTOCOLS UNDER VARYING VALUES OF *K*

	PPED	PP*k*MD	PPFC	PPMF	PP*k*C
k= 5	29.3	5.1	0.07	0.01	34.48
k= 10	29.3	5.18	0.11	0.01	34.59
k= 15	29.3	5.24	0.18	0.01	34.73
k= 20	29.3	5.35	0.29	0.01	34.95
k= 25	29.3	5.51	0.4	0.01	35.22

Table IV clearly indicates that increase in the number of neighbours considered for query classification (i.e. *k*) does not affect the execution time of proposed PP*k*C protocol which is a significant achievement. On other hand, when compared with recent privacy preserving solutions, the execution time of protocols in [34] and [41] grows linearly along with increasing value of *k*. Although the execution time of protocol in [33] remains almost constant while varying *k* from 5 to 25, still the time required is significantly more. This is discussed in detail in the comparative analysis section.

B. Comparative Analysis

The existing recent privacy preserving solutions in [33], [34], [35], [41] and [43] are compared with proposed PP*k*C protocol. The performance of proposed PP*k*C protocol is examined in terms of its execution time by varying the number of records (*n*) and the number of nearest neighbours considered (*k*) during the outsourced classification. For fair analysis, the execution time of PP*k*C protocol is compared with execution time of protocols in [33], [34] and [35] under above stated varying parameters and using the UCI KDD archive's Car Evaluation dataset [42] having 1728 records, 6 attributes and 4 unique classes. Size of encryption the key (K) used in [33], [34] and [35] is 1024 bits and hence, we maintain the same in our experiment.

Additionally, for extensive performance evaluation with much larger dataset, the comparative analysis on the execution time of proposed PP*k*C protocol and that of the most recent protocols in [41] and [43] is made under same varying parameters while using the SMOTE [40] based scaled version of the Car Evaluation dataset having 10000 records and 6 attributes. Size of the encryption key (K) used in [41] and [43] is 512 bits and hence, to maintain fairness we use the same key size in experiment with the scaled dataset.

1) Analysis with the car evaluation dataset: The state-of-the-art prior work in [33], [34] and [35] used the Car Evaluation dataset [42] encrypted with key size (K) of 1024 bits in their experimentations. Hence, for fairness of comparison we have also used the same encrypted dataset in experiments with the proposed PP*k*C approach. The results on the execution time and performance of proposed PP*k*C protocol under varying parameter of *n* and *k* are compared with state-of-the-art prior work. Table V shows the execution time required by proposed PP*k*C protocol as compared to other state-of-the-art protocols when varying the nearest neighbours i.e. *k*, from 5 to 25.

TABLE V. EXECUTION TIME (SEC) OF PP*K*C AND OTHER STATE-OF-THE-ART PROTOCOLS WITH VARYING VALUES OF *K* (WITH *N*=1724, *K*=1024)

	k= 5	*k*= 10	*k*= 15	*k*= 20	*k*= 25
proposed PP*k*C	76.26	76.61	76.9	77.37	77.82
Park et al. (2020) [33]	249.6	249.6	249.6	249.6	249.6
Liu et al. (2019) [34]	181.14	375.18	560.52	728.46	923.28
Wu et al. (2018) [35]	53.34	56.58	60.3	63.18	65.82

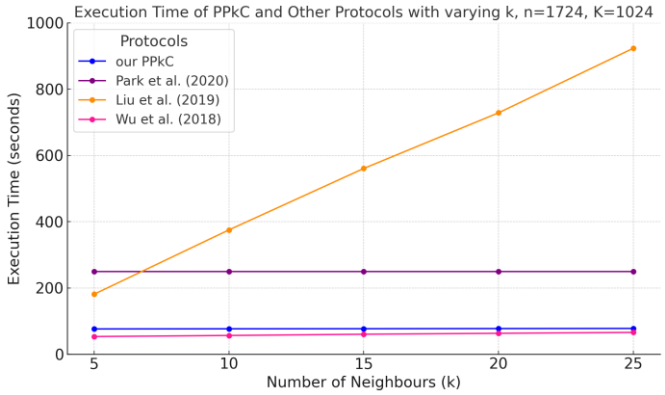

Fig. 5. Comparison on execution time of PP*k*C and existing state-of-the-art protocols with varying *k* using Car evaluation dataset.

Fig. 5 shows the analysis of execution time on the encrypted Car evaluation dataset with key size 1024 bits under varying values *k*. The running time of proposed PP*k*C varies from 76.26 to 77.82 seconds when the number of neighbours are changed from 5 to 25, respectively. Since, execution time of proposed approach remains almost constant so, we can significantly establish that the performance of proposed PP*k*C protocol is not much affected by changes in *k*. When *k*=25, the time taken for execution by PP*k*C protocol is 77.82 seconds which shows that it performs 11.86 times (i.e. 91.57 %) better than the SKC protocol in [34], 3.21 times (i.e. 68.82 %) better than the P*k*NC protocol in [33] and just 1.18 times less better than the PPKC protocol in [35]. As indicated by Table III, the memory usage of the PP*k*C protocol increases gradually with an increase in value of *k* but, so is the case with the other compared protocols. In fact, except protocol in [35] all the other compared protocols require more memory than proposed PP*k*C protocol when using key size of 1024 bits and with varying values of *k*.

Table VI shows the execution time required by proposed PP*k*C protocol as compared to other state-of-the-art protocols when varying the number of records i.e. *n*, from 500 to 1724.

TABLE VI. EXECUTION TIME (SEC) OF PP*K*C AND OTHER STATE-OF-THE-ART PROTOCOLS WHILE VARYING *N* (WITH *K*=25, *K*=1024)

	n= 500	*n*= 1000	*n*= 1500	*n*= 1724
our PP*k*C	22.58	45.12	67.66	77.82
Park et al. (2020) [33]	72.2	144.4	216.6	249.6
Liu et al. (2019) [34]	267.77	535.54	803.31	923.28
Wu et al. (2018) [35]	19.08	38.16	57.24	65.82

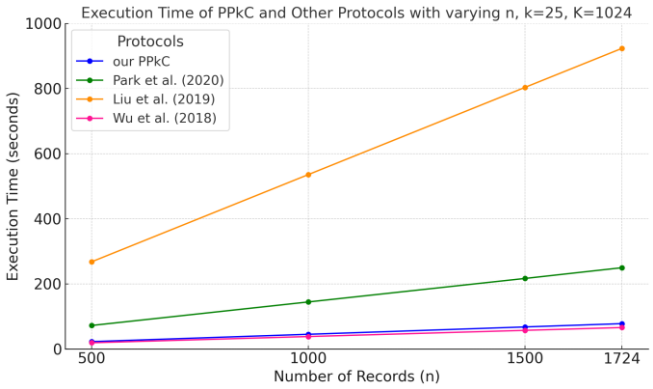

Fig. 6. Comparison on execution time of PP*k*C and existing state-of-the-art protocols with varying *n* using Car evaluation dataset.

Fig. 6 shows the analysis of execution time on the encrypted Car evaluation dataset with key size 1024 bits under varied number of records i.e. *n*, from 500 to 1724. The running time of proposed PP*k*C varies from 22.58 to 77.82 seconds when *n* is changed from 500 to 1724, respectively. Since, execution time drops significantly with proposed PP*k*C approach even while varying the number of records so, we can clearly establish that the performance of proposed protocol is much better than that of P*k*NC protocol and SKC protocol in [33] and [34], respectively. When *n*=1724, the time taken for execution by PP*k*C protocol is 77.82 seconds which shows that it again performs 11.86 times (i.e. 91.57 %) better than the SKC protocol in [34], 3.21 times (i.e. 68.82 %) better than the P*k*NC protocol in study [33] and just 1.18 times less better than the PPKC protocol in study [35]. However, protocol in [35] only aims at determining the class labels corresponding to *k* nearest records instead of providing the final class for user's query. It also reveals the *k* nearest class labels to the querying user, which does not satisfy the privacy requirements as stated in study [29].

2) Analysis with the scaled Car Evaluation dataset: In the literature survey, we observed that the performance of many existing privacy preserving classification protocols has depleted when they were tested with huge datasets. Specifically, the execution time of even the most recent protocols in [41] and [43] grows linearly while varying the number of records (*n*) due to the linear growth in computational cost. So, for extensive performance evaluation of proposed PP*k*C protocol we conducted experiments with the SMOTE [40] based scaled version of the Car Evaluation dataset which is a much larger dataset having 10000 records. Table VII shows the execution time required by PP*k*C protocol as compared to the most recent protocols in [41] and [43] when varying the nearest neighbours i.e. *k*, from 5 to 20.

Fig. 7 shows the comparative analysis on execution time of proposed PP*k*C with the most recent protocols in [41] and [43] using the scaled Car evaluation dataset encrypted with key size of 512 bits under varying values *k*. When *n*=10000 and the number of neighbours are changed from 5 to 20, the running time of proposed PP*k*C varies from only 183.21 to 184.68 seconds, respectively whereas the running time of [41] ranges from 60.32 to 202.12 seconds. Since, execution time of

proposed approach remains almost constant with scaled dataset also, we can significantly establish that even with huge datasets the performance of proposed PP*k*C protocol is not much affected when *k* is increased. Same is the case with the protocol in [43], its run time is also almost independent of *k*. However, the runtime of PP*k*C protocol is nearly 5 times better than that of protocol in [43]. When *k*=20, the time taken for execution by proposed PP*k*C protocol is 184.68 seconds which shows that it performs 9.96 % better than the protocol in study [41]. Also, it is worth observing that the execution time of protocol in [41] grows linearly with increasing value of *k* whereas it remains almost constant for proposed PP*k*C protocol across all values of *k*.

TABLE VII. EXECUTION TIME (SEC) OF PPKC AND RECENT EFFICIENT PROTOCOLS WITH VARYING VALUES OF *K* (*K*= 512)

	with *n*= 10000		with *n*= 6000	
	proposed PP*k*C	Kim et al. (2022) [41]	proposed PP*k*C	Wang et al. (2024) [43]
k= 5	183.21	60.32	103.24	600.26
k= 10	183.93	98.22	104.53	600.33
k= 15	184.32	135.87	104.77	601.67
k= 20	184.68	202.12	104.94	602.88

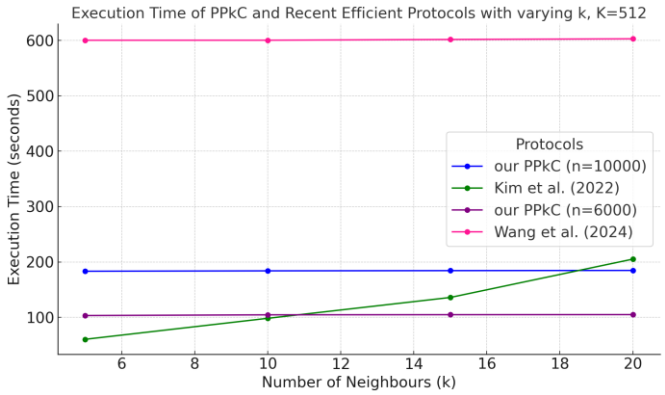

Fig. 7. Comparative analysis on execution time with varying *k* using the scaled dataset.

Table VIII shows the execution time required by proposed PP*k*C protocol as compared to the most recent protocols in [41] and [43] when varying the number of records i.e. *n*, from 2000 to 10000.

TABLE VIII. EXECUTION TIME (SEC) OF PPKC AND RECENT EFFICIENT PROTOCOLS WITH VARYING VALUES OF *N* (WITH *K*=10, *K*= 512)

	Proposed PP*k*C	Kim et al. (2022) [41]	Wang et al. (2024) [43]
n= 2000	39.12	22.44	240
n= 5000	87.1	56.11	480
n= 10000	183.93	98.22	1200

Fig. 8 shows the analysis of execution time of proposed PP*k*C protocol with the most recent protocols in [41] and [43] while using the scaled Car evaluation dataset encrypted with key size of 512 bits under varied number of records i.e. *n*, from 2000 to 10000. The running time of proposed PP*k*C varies from 39.12 to 183.93 seconds when *n* is changed from 2000 to 10000, respectively. Since, execution time drops significantly with proposed PP*k*C approach while using the scaled dataset also so, we can clearly establish that the performance of proposed protocol is much better than that of the protocol in [43]. When *n*=10000, the time taken for execution by proposed PP*k*C protocol is 183.93 seconds which shows that it again performs 6.52 times (i.e. 84.67 %) better than the protocol in [43] and just 1.87 times less better than the protocol in [41] with increasing value of *n*.

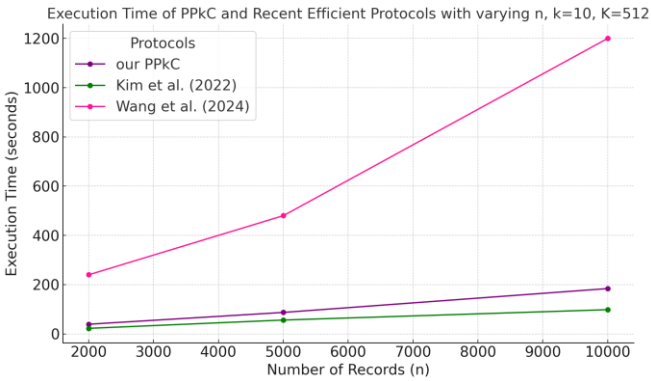

Fig. 8. Comparative analysis on execution time with varying *n* using the scaled dataset.

VII. CONCLUSION AND FUTURE SCOPE

In this paper, we have proposed efficient privacy-preserving *k*NN classification approach, named as PP*k*C protocol and its component protocols that leveraging partial homomorphic encryption (PHE). Our endeavor focused on demonstrating the feasibility and efficacy of PHE in safeguarding sensitive data while allowing for *k*NN classification in outsourced environments.

First and foremost, it is noteworthy that proposed protocols are preserving privacy of user's query, dataset values and the final class label. The protocol analysis shows that no information is ever disclosed and no knowledge can be potentially gained by both the clouds (C_1 and C_2) during execution of any of the component protocols for performing the outsourced *k*-NN classification on encrypted data. In the protocol analysis and implementation, we underscore that the proposed enhanced privacy preserving component protocols fully adhere to the privacy requirements outlined earlier in this paper. Notably, both cloud servers, $C_1 and\ C_2$, are kept oblivious to the identities of the database records associated with the computed nearest neighbours of the user query. Moreover, the intermediate data available to either of the clouds consist of encrypted random values or random numbers only. Also, the privacy preserving shuffling protocol eliminates the risk of C_2 understanding the data accessing patterns.

Furthermore, we explored the possibility of using the heap structure firstly to determine the *k* minimum distances and their corresponding class labels and then for finding the class label with maximum occurrence frequency. With this we were able to significantly reduce the computational overhead involved in classifying the encrypted user's query on

encrypted data and hence enhanced the efficiency of the overall PP*k*C protocol. This approach proved instrumental in optimizing performance with real-world datasets and also particularly in scenarios involving scaled datasets.

Through the experimental investigations, we have drawn several noteworthy conclusions. In the comparative analysis on execution time using the Car evaluation dataset and its scaled version encrypted with key size 1024 bits and 512 bits, respectively it is observed that the running time of proposed PP*k*C remains almost constant while varying values of *k* from 5 to 25. Even while varying *n* the execution time drops significantly with proposed PP*k*C approach with both the dataset. Hence, we established that the performance of proposed PP*k*C protocol is independent of variation in *k* and is much better than that of other recent protocols while varying *k* and *n*.

In our future work, we plan to utilize fully homomorphic encryption (FHE) schemes for working in an encrypted environment since it gives access to a wide range of operations thereby minimizing communication costs incurred in a two-cloud setup. However, its computational overhead must be considered as well and efforts must be taken to improve the same.

REFERENCES

[1] D. Zissis and D. Lekkas, “Addressing Cloud Computing Security Issues,” Future Generation Computer Systems, Vol. 28, No. 3, pp. 583-592, 2012.

[2] M. Zhang, Y. Zhang, Y. Jiang and J. Shen, "Obfuscating EVES Algorithm and Its Application in Fair Electronic Transactions in Public Clouds," in *IEEE Systems Journal*, vol. 13, no. 2, pp. 1478-1486, June 2019.

[3] X. Guo, Y. Li, Y. Jiang, J. Wang, J. Fang, “Privacy-Preserving k-Nearest Neighbor Classification over Malicious Participants in Outsourced Cloud Environments,” in *Cryptography*, vol. 7, no. 4, p. 59, 2023.

[4] R. Barona and E. A. M. Anita, "A survey on data breach challenges in cloud computing security: Issues and threats," *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*, Kollam, India, 2017, pp. 1-8, doi: 10.1109/ICCPCT.2017.8074287.

[5] M. Zhang, Y. Chen and J. Lin, "A Privacy-Preserving Optimization of Neighborhood-Based Recommendation for Medical-Aided Diagnosis and Treatment," in *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10830-10842, July, 2021.

[6] H.J. Kim, H. Lee, Y. K. Kim and J.W. Chang, “Privacy-preserving k NN query processing algorithms via secure two-party computation over encrypted database in cloud computing,” in *The Journal of Supercomputing*, vol. 78, no. 7, pp.9245-9284, 2022.

[7] D. Oh, I. Kim, K. Kim, S. M. Lee and W.W. Ro, “Highly secure mobile devices assisted with trusted cloud computing environments,” in *ETRI Journal*, vol. 37, no. 2, pp.348-358, 2015.

[8] J. Raja and M. Ramakrishnan, “Confidentiality-preserving based on attribute encryption using auditable access during encrypted records in cloud location,” in *The Journal of Supercomputing*, vol. 76, no. 8, pp.6026-6039, 2020.

[9] A. Ahmad, M. Ahmad, M. A. Habib, S. Sarwar, J. Chaudhry, M. A. Latif, S. H. Dar, and M. Shahid, “Parallel query execution over encrypted data in database-as-a-service (DaaS),” in *The Journal of Supercomputing*, vol. 75, pp.2269-2288, 2019.

[10] Y. Zheng, R. Lu, S. Zhang, J. Shao and H. Zhu, "Achieving Practical and Privacy-Preserving kNN Query over Encrypted Data," in *IEEE Transactions on Dependable and Secure Computing*, (Early Access), pp. 1-13, March, 2024.

[11] A. Alabdulkarim, M. Al-Rodhaan, T, Ma, Y. Tian, “PPSDT: A Novel Privacy-Preserving Single Decision Tree Algorithm for Clinical Decision-Support Systems Using IoT Devices” in *Sensors*, vol. 19, no.1, 2019.

[12] P. Centonze, “Security and Privacy Frameworks for Access Control Big Data Systems,” in *Computers, Materials & Continua*, vol. 59, no. 2, pp. 361-374, 2019.

[13] D. Patel, K. Srinivasan, C. Y. Chang, T. Gupta and A. Kataria, “Network anomaly detection inside consumer networks—a hybrid approach,” in *Electronics*, vol. 9, no. 6, pp.923, 2020.

[14] M. M. Salim,I. Kim, U. Doniyor, C. Lee and J. H. Park, “Homomorphic encryption based privacy-preservation for IoMT,” in *Applied Sciences*, vol. 11, no. 18, p.8757, 2021.

[15] N. B. A. Ghani, M. Ahmad, Z. Mahmoud and R. M. Mehmood, “A Pursuit of Sustainable Privacy Protection in Big Data Environment by an Optimized Clustered-Purpose Based Algorithm,” in *Intelligent Automation & Soft Computing*, vol. 26, no. 6, 2020.

[16] M. L. Yiu, G. Ghinita, C. S. Jensen and P. Kalnis, “Enabling search services on outsourced private spatial data,” in *The VLDB Journal*, vol. 19, pp.363-384, 2010.

[17] A. Boldyreva, N. Chenette, Y. Lee and A. O’neill, “Order-preserving symmetric encryption,” in Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, Proceedings 28, pp. 224-241, Springer Berlin Heidelberg, 2009.

[18] A. Boldyreva, N. Chenette, and A. O’Neill, “Order-preserving encryption revisited: Improved security analysis and alternative solutions,” in *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference*, CA, USA, Proceedings 31, pp. 578-595, Springer Berlin Heidelberg, 2011.

[19] Y. Qi and M. J. Atallah, "Efficient Privacy-Preserving k-Nearest Neighbor Search," *2008 The 28th International Conference on Distributed Computing Systems*, Beijing, China, pp. 311-319, 2008.

[20] M. Shaneck, Y. Kim and V. Kumar, “Privacy preserving nearest neighbor search,” in *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, Springer US, pp. 247-276, 2009.

[21] J. Vaidya and C. Clifton, “Privacy-preserving top-k queries,” in *21st International Conference on Data Engineering (ICDE'05)*, IEEE, pp. 545-546, April, 2005.

[22] H.J. Kim, Y.K Kim, H.J. Lee and J.W. Chang, “Privacy-Preserving Top-k Query Processing Algorithms Using Efficient Secure Protocols over Encrypted Database in Cloud Computing Environment,” in *Electronics*, vol. 11 no. 18, p.2870, 2022.

[23] W. K. Wong, D. W. L. Cheung, B. Kao and N. Mamoulis, “Secure kNN computation on encrypted databases,” in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 139-152, June, 2009.

[24] H. Hu, J. Xu, C. Ren and B. Choi, “Processing private queries over untrusted data cloud through privacy homomorphism,” *2011 IEEE 27th International Conference on Data Engineering*, Hannover, Germany, pp. 601-612, 2011.

[25] Y. Zhu, R. Xu and T. Takagi, “Secure k-NN computation on encrypted cloud data without sharing key with query users,” in *Proceedings of the 2013 international workshop on Security in cloud computing*, pp. 55-60, May, 2013.

[26] Y. Elmehdwi, B. K. Samanthula and W. Jiang, “Secure k-nearest neighbor query over encrypted data in outsourced environments,” in *2014 IEEE 30th International Conference on Data Engineering*, pp. 664-675, March, 2014.

[27] H. I. Kim, H. J. Kim and J. W. Chang, “A secure kNN query processing algorithm using homomorphic encryption on outsourced database,” in *Data & Knowledge Engineering*, vol. 123, pp.101602, 2019.

[28] P. Paillier, “Public key cryptosystems based on composite degree residuosity classes,” in *Eurocrypt*, pp. 223–238, 1999.

[29] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, “k- Nearest Neighbor Classification over Semantically Secure Encrypted Relational Data,” in *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, 2015.

[30] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "Secure k- Nearest Neighbor Query over Encrypted Data in Outsourced Environment," in *IEEE ICDE*, pp. 664- 675, 2014.

[31] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *ACM STOC*, pp. 169–178, 2009.

[32] Y. Zhu, X. Li, J. Wang, and J. Li, "Cloud-assisted secure biometric identification with sub-linear search efficiency," in *Soft Computing*, vol. 24, p. 5885–5896, 2019.

[33] J. Park and D. H. Lee, "Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments," in *IEEE Access*, vol. 8, p. 64617–64633, 2020.

[34] L. Liu, J. Su, X. Liu, R. Chen, K. Huang, R. H. Deng, and X. Wang, "Toward highly secure yet efficient knn classification scheme on outsourced cloud data," in *IEEE Internet of Things Journal*, vol. 6, pp. 9841–9852, 2019.

[35] W. Wu, J. Liu, H. Rong, H. Wang, and M. Xian, "Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database," in *IEEE Access*, vol. 6, pp. 41771–41784, 2018.

[36] W. Wu, U. Parampalli, J. Liu, and M. Xian, "Privacy preserving knearest neighbor classification over encrypted database in outsourced cloud environments," in *World Wide Web*, vol. 22, p. 101–123, 2018.

[37] Gaikwad VS, Walse KH, Thakare VM, "Review of the state-of-the-art methods for privacy preserved classification in outsourced environment," in *Proc. Int. Conf. Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, Feb. 2020, pp 1–6.

[38] J. Du and F. Bian, "A privacy-preserving and efficient k-nearest neighbor query and classification scheme based on k-dimensional tree foroutsourced data," in *IEEE Access*, vol. 8, pp. 69333–69345, 2020.

[39] Z. Li, H. Wang, S. Zhang, W. Zhang and R. Lu, "SecKNN: FSS-Based Secure Multi-Party KNN Classification Under General Distance Functions," in *IEEE Transactions on Information Forensics and Security*, vol. 19, pp.1326-1341, 2024.

[40] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," in *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, 2002.

[41] Y. K. Kim, H. J. Kim, H. Lee and J.W. Chang, "Privacy-preserving parallel kNN classification algorithm using index-based filtering in cloud computing," in *PLoS One*, vol. 17, no. 9, 2022.

[42] B. Z. M. Bohanec. (1997). *Car Evaluation Data Set*, UCI Machine Learning Repository. Accessed: May. 24, 2023. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

[43] H. Wang, Y. Zhao, Z. Cai and H. Zhao, "Privacy-Preserving kNN Classification Query Scheme for Encrypted Data in Outsourced Environments for Smart Grid," *4th International Conference on Computer Communication and Artificial Intelligence (CCAI)*, Xi'an, China, 2024, pp. 162-169.