Privacy Preserving Classification over Semantically Secure Encrypted Relational Data in Cloud Environment

Mr. Gaikwad Vijayendra Sanjay[1], Dr. Khan Rahat Afreen[2]

[1] PG student, Deogiri Institute of Engineering and Management Studies, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, Maharashtra state, India.
vij711@gmail.com

[2] Associate Professor, CSE Department, Deogiri Institute of Engineering and Management Studies, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, Maharashtra state, India. rahatkhan@dietms.org

---***---

Abstract— The Cloud environment, with its extensive resources has become a good choice for organizations to keep their data and access it on demand. When the organization's need is to just upload their data and use it as and when required from the cloud, the cloud service itself encrypts that data with its own credentials or in some cases, for maintaining the confidentiality, the data owners encrypt their data prior to outsourcing it. But there is no provision for processing some data within the cloud environment and at the same time maintain data confidentiality and privacy of user query. As a consequence, for classification, either the data needs to be decrypted by the cloud at some point of time and then processed to take proper classification decision or the data owner has no choice but to perform the same task at his/ her end partially or fully. Since the data used for classification is encrypted and placed onto a cloud, the conventional privacy preserving classification methods are not suitable. Some recent work has been done in this direction, but it is proven to be computationally costly and also not very practical. Our proposed system is an effort towards resolving this very problem of classifying encrypted user queries over encrypted data in a more effective and time efficient manner. This is achieved re- designing the existing privacy preserving protocol from a different perspective and by leveraging the properties of homomorphic cryptosystem. Our approach is computationally inexpensive and does not compromise the privacy of user query or the confidentiality of the database outsourced by the data owner.

Keywords— **encrypted database, homomorphic cryptosystem, k- nearest neighbors, security**

1. INTRODUCTION

The recent trends in cloud services have revolutionized the outlook of organizations towards leveraging the benefits of outsourcing their data. Cloud computing, with its platform as a service (PaaS) feature, has seriously grabbed the attention of organizations desiring to completely outsource their valuable data along with the some data management tasks. But, despite of various facilities that cloud avails, there are still some data confidentiality and privacy issues that keep the organizations from utilizing them. When data is straight away uploaded to the cloud, the cloud itself encrypts it, for securing it from any third party theft and then stores it. By doing so, the data is open for the cloud service providers at the first place which can be threat. If the data contains very sensitive information such as medical records of patients, then somewhere down the line the patients' privacy gets compromised. To avoid this, the first solution that organizations use is to encrypt their data, prior to uploading it to the cloud. But what when the use of this data is just not limited to its retrieval? To perform some processing over this encrypted data at the cloud without ever decrypting it is very difficult task.

The privacy issues involved in this kind of situations can be explained by the example. Consider that a hospital keeps their patients encrypted database on cloud along with the data mining task. Now, when a doctor wants assert about symptoms of a disease of the patient, which he/ she cannot affirmatively treat, the doctor can use relative classification process and find out the disease with which the patient is suffering. For getting a precise response, the doctor needs to trigger a query for the classification process on cloud, which would contain patient's highly personal information. So, it is very obvious that this query must be encrypted prior to sending it to the cloud, in order to protect the patient's privacy. Thus, it is important to consider the privacy of the users' query when it is involved in the data mining task. Also, any cloud malfunction activity can determine useful information about data access patterns although data are always encrypted. Therefore, we can say that, while performing a classification or any other data mining task on encrypted data in an outsourced environment as cloud, the data owner's confidentiality, user query's privacy and preventing the cloud from learning any access patterns must be the foremost objectives.

In this paper, we have proposed some methods which collaboratively solve the secure classification over encrypted data problem assuming that encrypted data and the classification process are outsourced. Although each of the classification techniques has their own pros and cons, our work concentrates only on the k-nearest neighbor technique; it being the most suitable for our work.

1.1 Problem Definition

Consider a data owner having a database with m attributes and n records where the first attribute (practically kept as 0^{th}) is the identifier, *I*, for the record and m^{th} attribute refers to the class label, *c*. The database is encrypted attribute- wise by the data owner, such that, $E_{pk}(t_{i,j})$ corresponds to an encrypted record value, for $1 \leq i \leq n$ and $0 \leq j \leq m$, where t is a tuple. The encryption function, E_{pk} belongs to a semantically secure encryption scheme [1]. The data owner then outsources his/her encrypted database, denoted by *EDB* along with the classification process and hereafter has no intervention in the classification process.

Any authorized user with query, denoted as $Q=(q_1 \ldots\ldots q_{m-1})$, will query this *EDB* in order to gain the class label, denoted as, c_q, for the query.

1.2 Our Contributions

In this paper, we present an improvised privacy preserving *k*- NN classifier [2] which works over semantically secure encrypted data. This improvised classifier is considerably inexpensive in terms of computational cost and proves to be a practically more feasible solution to the classification over encrypted data problem. As mentioned in [2], following are the privacy requirements for a privacy preserving *k*- NN protocol:

- User's query should not be disclosed to the cloud i.e. it should remain encrypted throughout the classification process.
- The actual database contents or the intermediate results of the process must not be revealed to the cloud.
- The records corresponding to the k- nearest neighbors of Q must neither be revealed to cloud nor to the user.
- The resulting class label, c_q, must be only revealed to the user.

Our efforts in this paper are motivated by the work of Samanthula, Elmehdwi and Jiang in [2]. As mentioned in [2] about the scope for improvements in the efficiency of $\mathrm{SMIN_n}$ protocol, we concentrate on improving the time requirements of SMIN and some other sub- protocols. On a practical note, it has been observed that Paillier cryptosystem is not very effective in handling negative values as result of any Paillier addition. This problem might occur while performing attribute- wise subtraction in secure squared Euclidean distance (SSED) protocol [3]. To address this situation, we have proposed a new solution to securely compute the squared Euclidean distance. It is worth mentioning that, during our improvised protocols, all the above mentioned privacy requirements are satisfied as, the cloud remains unaware of which database records correspond to the derived nearest neighbors. Also, any intermediate values that are computed and are visible to the cloud are either encrypted random values or random numbers. Moreover, the final output is not known to cloud.

2. EXISTING RELATED WORK

Here, it is worth mentioning that any data mining task over encrypted data can be performed with comparatively less efforts using fully homomorphic cryptosystems [4], as this cryptosystem supports any number of arbitrary functions on encrypted data without having to decrypt it. But, these cryptosystems are very expensive in terms of computation and hence, their use may require extensive hardware support.

2.1 Privacy Preserving Protocols and their Limitations

In the recent past, there have been a few systems proposed for privacy preservation in data mining applications such as data perturbation by Agrawal and Srikant [5] and data distribution by Lindell and Pinkas [6]. The former one is the first decision tree based solution but, is not suitable for semantically secure encrypted data while, the later one is first decision tree solution that works in a two party setup, but it considers that data is distributed in plaintext form over many parties and not encrypted.

The SCONDB model proposed in [7] is a secure query processing model where, the nearest neighbors of the query are given to the user, who then decrypts uses the conventional k- NN technique to find the most relative class label. However, this model reveals the k- NN to the user out writes our objective.

In the most recent work [2], the k- nearest neighbors are not revealed to the cloud nor to the user. In [2], Samanthula, Elmehdwi and Jiang propose the PPkNN protocol and many new the security primitives along with their solutions and supporting security proofs, namely, secure minimum (SMIN), secure minimum from n numbers ($\mathrm{SMIN_n}$), secure frequency (SF). PPkNN protocol intially uses the secure squared Euclidean distances (SSED) protocol to determine the encrypted distance $E_{pk}(d)$ of each record in the database from the user query. The Secure Bit Decomposition protocol then converts these $E_{pk}(d_i)$ values to $[d_i]$, denoting the encryption of binary bits of d_i. The SMIN protocol then computes the encrypted bits of the minimum $[d_i]$ and corresponding class label, $E_{pk}(c_i)$. It is observed that computational cost of this SMIN protocol is very high and it incurs almost 67 % of the total computational cost.

Our primary aim is to reduce this computational cost without compromising on the security and privacy objectives mentioned in [2]. In this paper, we propose an improvised version of the PPkNN protocol by re-designing the SMIN protocol and removing the overhead incurred by SBD and SBOR protocols. We also present experimental results to support our predictions. Also, it is our observation that in the SSED protocol mentioned in [2], whenever a negative value result from a Paillier addition, these values cannot be deduced to their correct plaintext form. This is because Paillier addition does not support negative values as the result. Our work addresses this issue by seeing the SSED problem from a different perspective.

2.2 Paillier Cryptosystem

Paillier cryptosystem [1] is additively homomorphic in nature. It is a probabilistic public key encryption scheme that provides semantic security. If we assume E_{pk} is an encryption function with public key *pk* given by *(N, g)*, where N is product two large prime and *g* is a generator in $Z^*_{N^2}$. Also, D_{sk} is the decryption function with secret key *sk*, then for any two plaintexts x and y $\in Z_N$, the Paillier encryption scheme has the following properties:

a) Homomorphic Addition

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

b) Homomorphic Multiplication

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

3. IMPROVISED PRIMITIVES FOR PRIVACY PRESERVATION

In this section we discuss the working of some sub-protocols that act as the building blocks for computing the k- nearest neighbors in a more efficient way. We will hereafter consider a federation of two non- colluding, semi honest cloud service providers, $C_1$and C_2. C_2 is hosting the secret key *sk* and *pk* is public (known to both C_1 and C_2).

3.1 Improvised Secure Minimum (I-SMIN)

This protocol takes as input two vectors, *a*= ($E_{pk}(I_a)$, $E_{pk}(d_a)$, $E_{pk}(c_a)$) and *b*= ($E_{pk}(I_b)$, $E_{pk}(d_b)$, $E_{pk}(c_b)$) where, *I* denotes the unique identifier for each record, d_i is the distance of a record from the given query and c_i is the class label corresponding to that record. The aim of this protocol is to determine the minimum $E_{pk}(d_i)$ of the two, without revealing it to $C_1$and C_2.

Algorithm 1: **I-SMIN (*a*, *b*)→ ($E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$)**

Requires: C_1 holds *a*= ($E_{pk}(I_a)$, $E_{pk}(d_a)$, $E_{pk}(c_a)$) and *b*= ($E_{pk}(I_b)$, $E_{pk}(d_b)$, $E_{pk}(c_b)$) and C_2 holds the secret key *sk*.

1) C_1:

- Generate a random number $r \in Z_N$
- Encrypt *r* with *pk* to get $E_{pk}(r)$
- Now, randomize the all elements of both vectors *a* and *b* with $E_{pk}(r)$, as follows:
$$I'_a = E_{pk}(I_a) * E_{pk}(r) \; mod \; N^2$$
$$d'_a = E_{pk}(d_a) * E_{pk}(r) \; mod \; N^2$$
$$c'_a = E_{pk}(c_a) * E_{pk}(r) \; mod \; N^2$$
- So we have, a'= (I'_a, d'_a, c'_a) and similarly b'= (I'_b, d'_b, c'_b)
- Send a' and b' to C_2

2) C_2:

- Receive a' and b' from C_1 and decrypt them using *sk* as,
    - a.) *u*= $D_{sk}(a')$ and *v*= $D_{sk}(b')$
- Now, compare d_u and d_v
    - a.) if $d_u \leq d_v$, then α= *u*
    - b.) else α= *v*
- Encrypt α as,
    - a.) α' = E_{pk}(α); and send α' to C_1

3) C_1:

- Receive α' from C_2 and remove randomization effect from all elements of α',

a.) $E_{pk}(I_{min}) = E_{pk}(I_{\alpha'}) * E_{pk}(r)^{N-1} \; mod \; N^2$

b.)$E_{pk}(d_{min}) = E_{pk}(d_{\alpha'}) * E_{pk}(r)^{N-1} \; mod \; N^2$

c.)$E_{pk}(c_{min}) = E_{pk}(c_{\alpha'}) * E_{pk}(r)^{N-1} \; mod \; N^2$

To start with, in this protocol, C_1 generates a random number $r \in Z_N$, encrypts it with the public key *pk* that C_1 already has and randomizes the all elements from both the vectors. The reason for both vectors being randomized with the same random value is to maintain their relevance. The resulting vectors a' and b' are then sent to C_2. At C_2, these randomized vectors are decrypted with the secret key *sk* to get *u* and *v*. Now, the distance parameters d_u and d_v from vectors *u* and *v*, respectively, are compared to find minimum of them. If $d_u$is found to be minimum then vector *u* is assigned to the new vector α, otherwise α is assigned with *v*. This vector α, is then encrypted with the public key *pk* to get α' and α' is sent to C_1. After receiving the minimum vector in the form of α', C_1 removes the randomness from it to get the required encrypted minimum vector of ($E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$). Similarly, we can also formulate an Improvised Secure Maximum (I-SMAX) to find out the maximum between vectors with varying parameters setting.

3.2 Protocol to securely compute the Squared Euclidean Distance

The aim of this protocol is securely calculate the square of the distance between an encrypted database record $E_{pk}(t_i)$ and the user's query $E_{pk}(q)$, for 1≤ i ≤ n, where n is total number of records. Here, all the encryptions are done with the public key pk, known to both clouds C_1 and C_2 and user. The secret key sk is only known to C_2. In order to classify a query over an encrypted database, the user is first required to select values for the predefined attributes. At the user's end these attribute values are summed together to get, $(\sum_{i=1}^{m-1} q_i)$ and squaring this summation we get, $((\sum_{i=1}^{m-1} q_i)^2)$, where m is the total number of attributes in the encrypted database. Finally, the user encrypts these two values with the data owner's public key pk, prior to sending them on C_1 as a classification request. So, this protocol is invoked with inputs, the encrypted query attribute summation, $E_{pk}(\sum_{i=1}^{m-1} q_i)$, encrypted square of query attribute summation, $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$ and encrypted record attribute summation, $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$ (independently computed on C_1). Following are the step involved in our new, Improvised SSED (I- SSED) protocol, where $C_1$and C_2 jointly compute $E_{pk}(d)$.

In this protocol, $C_1$and C_2 compute the encrypted square of $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$ using the secure multiplication (SM) protocol proposed in [2], for 1≤ i ≤ n, where n is number of records. Similarly, we compute the encrypted product of $(\sum_{i=1}^{m-1} q_i)$ and $(\sum_{j=1}^{m-1} t_{i,j})$. The resulting values from SM are only known to C_1. Now, we can apply the homomorphic properties with reference to the mathematical formula, $a^2 + b^2 - 2ab$, to get the encrypted squared Euclidean distance, $E_{pk}(d)$. There slight difference in the setting of information exchange between user and C_1as compared to the SSED protocol in [2]. In our protocol, we require the user to send an encrypted sum of all query attributes to C_1 instead of sending individual encryptions of attribute values. This does not incur the user any additional overhead as the user performs simple summation and then encrypts only the sum. In fact, compared to the requirements in [2], our protocol reduces the number of encryptions to be performed at the user's end from 6 to just 2. Also, the number of Paillier additions required for performing attribute- wise subtraction are reduced.

Algorithm 2: I-SSED $\boldsymbol{(E_{pk}(\sum_{i=1}^{m-1} q_i),\ E_{pk}((\sum_{i=1}^{m-1} q_i)^2),\ E_{pk}(\sum_{j=1}^{m-1} t_{i,j}))}$→ $\boldsymbol{E_{pk}(d)}$

Requires: C_1 holds $E_{pk}(\sum_{i=1}^{m-1} q_i)$, $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$, $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$ and C_2 holds the secret key sk.

1) $C_1$and C_2:

- Compute $E_{pk}((\sum_{j=1}^{m-1} t_{i,j})^2)$ using standard SM protocol
- Compute $E_{pk}((\sum_{i=1}^{m-1} q_i) * (\sum_{j=1}^{m-1} t_{i,j}))$ using again standard SM protocol

2) C_1:

- Compute the encrypted squared Euclidean distance, $E_{pk}(d)$, as,

$$E_{pk}(h) = E_{pk}((\sum_{j=1}^{m-1} t_{i,j})^2) * E_{pk}((\sum_{i=1}^{m-1} q_i)^2) \ mod\ N^2$$

$$E_{pk}(d) = E_{pk}(h) \ * E_{pk}((\sum_{i=1}^{m-1} q_i) * (\sum_{j=1}^{m-1} t_{i,j}))^{N-2} \ mod\ N^2$$

3.3 Secure Elimination protocol (S- ELIM)

Whenever a database record is found to be the nearest neighbor to the query, it is necessary that the corresponding distance is updated to exclude this record from participating in further classification process. But, since the Paillier homomorphic scheme is semantically secure, C_1 is unable to find which record corresponds to $E_{pk}(d_{min})$ and $E_{pk}(c_{min})$. The aim of this protocol is to update the distance $E_{pk}(d_{min})$ to a maximum value so that it is automatically left out in the next iterations. One input to this protocol is the collection of ($E_{pk}(I_i)$, $E_{pk}(d_i)$, $E_{pk}(c_i)$), for 1 ≤ i ≤ n, where n is total number of records. For convenience, we refer this collection as *I_d_c_Map*. Other inputs include $E_{pk}(I_{min})$, $E_{pk}(d_{min})$ and $E_{pk}(c_{min})$ which are credentials of the record that is found to be nearest to the query in an iteration. The output will a same collection with an updated encrypted distance for the appropriate record (refered here as *Updated_Map*) . C_1 initially computes δ, since it involves an exponentiation and is referred n times. Then, a vector named ID is used to store the Paillier subtraction of δ from each of the $E_{pk}(I_i)$, for 1 ≤ i ≤ n. Here it is worth noting that only one of the entries in ID will be computed as $E_{pk}(0)$. Now, ID is sent to C_2, where it is decrypted. Now, a new vector Θ is constructed by replacing all entries of non- zero values with $E_{pk}(0)$ and 0s with $E_{pk}(100)$ (maximum threshold considered as 100). $E_{pk}(100)$ will occur exactly once in Θ. This vector Θ is sent to C_1. At C_1, an entry in Θ is Paillier added to its corresponding $E_{pk}(d_i)$. As a result, C_1 obliviously updates only the encrypted distance corresponding to the nearest record and all other $E_{pk}(d_i)$

Algorithm 3: S- ELIM $\boldsymbol{(E_{pk}(I_{min}), E_{pk}(d_{min}), E_{pk}(c_{min}),}$ ***I_d_c_Map*)→ *Updated_Map***

Requires: C_1 holds Record_Cred_Map, newly computed nearest neighbour credentials ($E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$) and public key pk and C_2 holds the secret key sk.

1) C_1:
   - Compute, $\delta = E_{pk}(I_{min})^{N-1} \bmod N^2$
   - Compute a vector, ID, for i=1 to n
     a.) $ID_i = E_{pk}(I_i) * \delta \bmod N^2$
   - Send vector ID to C_2
2) C_2:
   - Receive ID from C_1 and decrypt it as,
     a.) $ID' = D_{sk}(ID)$
   - Now, compute vector Θ, for i=1 to n
     a.) if $ID'_i = 0$, then $\Theta_i = E_{pk}(100)$
     b.) else $\Theta_i = E_{pk}(0)$
   - Send vector Θ to C_1
3) C_1:
   - Receive Θ from C_2 and update all $E_{pk}(d_i)$ as, for i=1 to n
     a.) $E_{pk}(d_i) = E_{pk}(d_i) * \Theta_i \bmod N^2$
   - Updated_Map ← $((E_{pk}(I_1), E_{pk}(d_1), E_{pk}(c_1)), \ldots\ldots, (E_{pk}(I_n), E_{pk}(d_n), E_{pk}(c_n)))$

remain unchanged. As intended, C_1 and C_2 do not know which $E_{pk}(d_i)$ is updated.

4. SECURITY ANALYSIS OF IMPROVISED PROTOCOLS

Although, while improving the efficiency of protocols in [2], we have changed the functioning and the protocol requirement, the final output of all improvised protocols are always in encrypted format and are only known to C_1. In fact, the changes induced always contributed towards reducing the computations. Also, C_2 deals with only random values which have no relevance with original ones. Values computed on C_2 are always conveyed to C_1 in encrypted form. Following is the formal security analysis of all proposed improvisations.

4.1 Security Analysis for I- SMIN Protocol

To start with, the inputs given to this protocol are all encrypted and the encryption scheme being semantically secure, these inputs are never revealed. The main strength of I- SMIN protocol lies in the fact that, the scope of the random number r generated from Z_N is limited to C_1 and hence, after randomizing the encrypted distance values $E_{pk}(d_a)$ and $E_{pk}(d_b)$, C_2 cannot predict them. The decision taken by C_2 is based on randomized values due to which no extra information is leaked at C_2. Moreover, each time I- SMIN takes a new pair for comparison, a new random number is generated, so that C_2 should not acquire any knowledge on the relation between last and current pairs. On finding the minimum, C_2 encrypts its result prior to sending it to C_1. The ciphertext corresponding to the minimum distance received at C_1 (whether $E_{pk}(d_a)$ or $E_{pk}(d_b)$) is different from the one it was prior to sending it to C_2. Hence, C_1 cannot predict which of the two distances it sent to C_2 was found to be minimum. All together, it is evident that neither C_1 nor C_2 learns anything about d_a, d_b, c_a, c_b.

4.2 Security Analysis for I- SSED Protocol

In the problem setting of PPkNN [2], user was required to provide six encrypted attributes values which constituted the query for classification. The SSED proposed in [3] uses the Paillier additive property to perform an attribute- wise subtraction between the encrypted attributes of the record and query. However, the Paillier cryptosystem does not support a negative output of any subtraction performed using Paillier additive property. For example, if we perform a Paillier addition, $E_{pk}(5) * E_{pk}(10)^{N-1} \bmod N^2$, the expected output is an encryption of -5. However, decrypting this output does not give us the desired result. To tackle this problem, we introduce a new equation for securely computing the squared Euclidean distance. With this change, the user is now required to provide only two encrypted values, $E_{pk}(\sum_{i=1}^{m-1} q_i)$ and $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$ as described in algorithm- 2. During this protocol, $E_{pk}((\sum_{i=1}^{m-1} t_i)^2)$ is securely computed using $E_{pk}(\sum_{i=1}^{m-1} t_i)$ with the SM protocol [3]. Also, $E_{pk}((\sum_{i=1}^{m-1} q_i) * (\sum_{i=1}^{m-1} t_i))$ is computed using the same SM protocol. The security proof for SM protocol is already mentioned in [3]. The output of SM protocol is only known to C_1. Also, the results of the Paillier additions performed on C_1 are in encrypted form. Thus, the output of our I-SSED protocol is encrypted and only known to C_1.

4.3 Security Analysis for S- ELIM Protocol

This protocol ensures that the $E_{pk}(d_i)$ corresponding to nearest record at the end of an iteration, is restricted from participating in the next iteration. For this purpose, C_1 forms the vector ID which contains the encrypted differences. When C_2 decrypts this vector, it does not have any understanding about this vector and simply substitutes 0s with $E_{pk}(100)$and non- zero values with $E_{pk}(0)$. On receiving this vector Θ, C_1 adds this vector component- wise to the corresponding $E_{pk}(d_i)$ values, using Paillier additive property. Thus, even C_1 does not acquire any information about which $E_{pk}(d_i)$ is updated.

5. IMPROVISED PPkNN PROTOCOL (I- PPkNN)

Here we propose the improvised privacy preserving protocol for performing k- nearest neighbour classification for the user's encrypted query, n an encrypted database. In section 1, we discussed about the problem statement and he scenario. To start this section, we would like to put forth the other assumption made while constructing our protocol. Firstly, we assume that clouds C_1 and C_2 are configured to communicate with each other, only when required, during the execution of the sub- protocols. These clouds do not collude and are curious to learn any information about the data they exchange. The data owner outsources the encrypted database, *EDB*, to C_1 and the secret key *sk* to C_2. The public key *pk* is communicated to all the participating entities. The class label for user's query is computed with the help of I- PPkNN protocol that executes in two stages, which are explained below.

5.1 Securely Determining the *k*- Nearest Class Labels (SD*k*-NCL)

The user selects the query attribute values $(q_1,\ldots,q_{m-1})$ sums them up to get $\sum_{i=1}^{m-1} q_i$, where *m* is total number of attributes in *EDB*. Also, he/ she computes $(\sum_{i=1}^{m-1} q_i)^2$. User then encrypts these two values with the public key (*pk*) of the data owner and sends a classification request to C_1 with these encrypted values. After receiving the query, C_1 computes the encrypted sum of all attribute values of each record in the *EDB*, $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$, using Pailler additive property, where $1 \le i \le n$. C_1 *and* C_2 jointly compute the encrypted squared Euclidean distance, $E_{pk}(d_i)$, between the query and each of *EDB* records, using our proposed I- SSED protocol. C_1 now build an identifier-distance-class label map (*I_d_c_Map*) where, $E_{pk}(I_i)$ and $E_{pk}(c_i)$ are the 0^{th} and m^{th} attribute values, respectively, of the i^{th} record. Now, the I-SMIN protocol, with two entries at a time from the *I_d_c_Map*, iteratively computes $E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$ corresponding to the first nearest neighbour. Then, the S- ELIM protocol securely updates the $E_{pk}(d_i)$ in *I_d_c_Map* corresponding to this $E_{pk}(I_{min})$, so that the record corresponding to $E_{pk}(I_{min})$ is automatically excluded from the next iterations. $E_{pk}(c_{min})$ is the first among *k*- nearest class labels. In the next *k*-1 iterations, we determine all of the *k*- nearest class labels. This set, $(E_{pk}(c_1),\ldots,E_{pk}(c_k))$ is then given to SDMCL protocol to find the most frequently occurring class label from these *k* class labels.

Algorithm 4: SD*k*-NCL $(\boldsymbol{EDB}, \boldsymbol{q}) \rightarrow (\boldsymbol{E_{pk}(c_1)},\ldots,\boldsymbol{E_{pk}(c_k)})$

Requires: C_1 holds *EDB* and public key *pk*, user has *pk* and C_2 holds the secret key *sk*.

1) User:
   - Select query attribute values and compute $\sum_{i=1}^{m-1} q_i$ and $(\sum_{i=1}^{m-1} q_i)^2$
   - Encrypt these values, as $E_{pk}(\sum_{i=1}^{m-1} q_i)$ and $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$ and send to C_1
2) C_1 *and* C_2:
   - Receive $E_{pk}(\sum_{i=1}^{m-1} q_i)$ and $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$from user
   - for *i*=1 to *n*
     a.) C_1 computes, $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$
     b.) $E_{pk}(d_i) \leftarrow$ I-SSED($E_{pk}(\sum_{i=1}^{m-1} q_i)$, $E_{pk}((\sum_{i=1}^{m-1} q_i)^2)$, $E_{pk}(\sum_{j=1}^{m-1} t_{i,j})$)
   - *I_d_c_Map*$\leftarrow$(($E_{pk}(I_1)$, $E_{pk}(d_1)$, $E_{pk}(c_1)$), ……, ($E_{pk}(I_n)$, $E_{pk}(d_n)$, $E_{pk}(c_n)$))
   - *Next_level_Map*$\leftarrow$ *I_d_c_Map*
   - for *s*=1 to *k*, do
     a.) *Next_level_Map*$\leftarrow$ *I_d_c_Map*
     b.) for *p*=1 to *size of Next_level_Map*, do
       - $E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$$\leftarrow$I-SMIN(*a*, *b*) where, *a*= $E_{pk}(I_p)$, $E_{pk}(d_p)$, $E_{pk}(c_p)$ and *b*= $E_{pk}(I_{p+1})$, $E_{pk}(d_{p+1})$, $E_{pk}(c_{p+1})$
       - Add ($E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$) to *Temp_Map*
       - *p=p+2*
     c.) if *size of Temp_Map* >1
       - *Next_level_Map*$\leftarrow$ *Temp_Map*
       - Repeat from a.)

       else
       - $E_{pk}(c_s)$$\leftarrow$$E_{pk}(c_{min})$
       - *I_d_c_Map* $\leftarrow$S- ELIM ($E_{pk}(I_{min})$, $E_{pk}(d_{min})$, $E_{pk}(c_{min})$, *I_d_c_Map*)
3) SDMCL ($E_{pk}(c_1)$,…. $E_{pk}(c_k)$)

5.2 Securely Determining the Majority Class Label (SDMCL)

Firstly, we assume that the data owner also outsources the list of all encrypted class labels ($E_{pk}(c_1)$,…., $E_{pk}(c_w)$) to C_1, where *w* is the total number of unique class labels in *EDB.* In the first step, C_1 *and* C_2 compute the encrypted frequency of each of the *w* class labels with $E_{pk}(c_1)$,…., $E_{pk}(c_k)$) and ($E_{pk}(c_1)$,…., $E_{pk}(c_w)$) as input to the SF protocol [2]. Let, $E_{pk}(f(c_i))$ be the encrypted frequency for class label c_i, for $1 \le i \le w$. Now, C_1 *and* C_2 collaboratively execute the I- SMAX protocol (as suggested in the description of I- SMIN protocol), with input ($E_{pk}(f(c_i))$, $E_{pk}(c_i)$), for $1 \le i \le w$. The I- SMAX protocol, iteratively determines the encrypted class label, $E_{pk}(c_q)$ with maximum frequency, $E_{pk}(f(c_{max}))$. Now, only C_1 knows the output of I- SMAX protocol and the only task left is to securely communicate the class label to the user. For this purpose, C_1 randomizes the encrypted class label $E_{pk}(c_q)$ with $r_q \in Z_N$ by computing $\beta_q = E_{pk}(c_q) * E_{pk}(r_q)$. C_1, then sends β_q to C_2 and r_q to the user. C_2 decrypts β_q to obtain the randomized majority class, $\beta_q{}' = D_{sk}(\beta_q)$ and send $\beta_q{}'$ to the user. User now has r_q from C_1 and $\beta_q{}'$ from

C_2. So, the user computes the final output class label for her/ his query as, $c_q = \beta_q{}' - r_q$.

6. EXPERIMENTAL RESULTS AND PERFORMANCE

In this section, we declare the system configuration and other setting for successful implementation of our proposed protocols. We have used evaluated the performance of our protocols on Intel® CORE i5® CPU with processing speed of 1.7 GHz and 4 GB of RAM running on Windows 8 operating system. The additive homomorphic scheme used throughout our algorithms is Paillier cryptosystem [1], and hence, our evaluated results are easily comparable with those in [2].

6.1 Dataset

For our implementation, we have also used the Car Evaluation dataset from the UCI KDD archive [8], as used in the experimental setup of [2]. This dataset consists of 1728 records i.e. n= 1728, and 6 attributes. One more attribute is there to represent the class label. Moreover, as per our requirement, we have added one more identifier attribute to all the records in the dataset (i.e. now, m=8). This dataset is classified into 4 unique classes, w=4. We have encrypted our dataset using Paillier encryption with key size of 512 bit (K=512) and varied the value for k.

6.2 Performance evaluation of I- PP*k*NN Protocol

As mentioned in section 5, the I- PP*k*NN protocol has two stages, namely, SD*k*NCL and SDMCL. We have evaluated the computation costs for both the stages by varying, the number of nearest neighbors (k) from 5 to 15.We have also assessed the performance by varying the key size as, K=512 and K=1024. We now compare our results with those in [2] under exactly same parameter settings. For K=512, SD*k*NCL takes 14.7 minutes to 57.5 minutes when k is change from 5 to 20, respectively. Thus, the computation cost increases linearly with k. Fig. 1 shows this linear change. It is also observed that as k is doubled the cost of SD*k*NCL also gets doubled. So, when k=10, the computation time of SD*k*NCL was 28.6 minutes. With K=512 and k=5, the Stage- 1 of PPkNN in [2] incurred 9.98 minutes. However, they evaluated the computation costs on a machine with Intel® XEON® Six- Core CPU with processing speed of 3.07 GHz and 12 GB of RAM. So, considering our machine configuration (which is 3 times smaller), we can assert that our proposed improvisations to the protocols in [2] can reduced the overall computation cost considerably.

On the other hand, for K=1024, SD*k*NCL takes 93.6 minutes to 372.94 minutes when k is change from 5 to 20, respectively. With K=1024 and k=5, the Stage- 1 of PPkNN in [2] incurred 66.97 minutes. Again, considering the differences in machine configuration, our results are very good.

Algorithm 5: SDMCL $(E_{pk}(c_1),...., E_{pk}(c_k)) \rightarrow c_q$

Requires: C_1 holds $(E_{pk}(c_1),...., E_{pk}(c_k))$, $(E_{pk}(c_1),...., E_{pk}(c_w))$ and pk and C_2 holds the secret key sk.

1) C_1 *and* C_2:
   - $(E_{pk}(f(c_1)),...., E_{pk}(f(c_w))) \leftarrow$ SF(A, B)
     where, A= $(E_{pk}(c_1),...., E_{pk}(c_w))$ and
     B= $(E_{pk}(c_1),...., E_{pk}(c_k))$
   - $(E_{pk}(f(c_{max})), E_{pk}(c_q)) \leftarrow$I-SMAX $((E_{pk}(f(c_1)), E_{pk}(c_1)), ...,$ $(E_{pk}(f(c_w)), E_{pk}(c_w)))$
2) C_1:
   - $\beta_q = E_{pk}(c_q) * E_{pk}(r_q) \bmod N^2$, where $r_q \in Z_N$
   - Send β_q to C_2 and r_q to user
3) C_2:
   - Receive β_q from C_1
   - Compute, $\beta_q{}' = D_{sk}(\beta_q)$; send $\beta_q{}'$ to user
4) User:
   - Receive $\beta_q{}'$ from C_2 and r_q from C_1
   - $c_q = \beta_q{}' - r_q \bmod N$

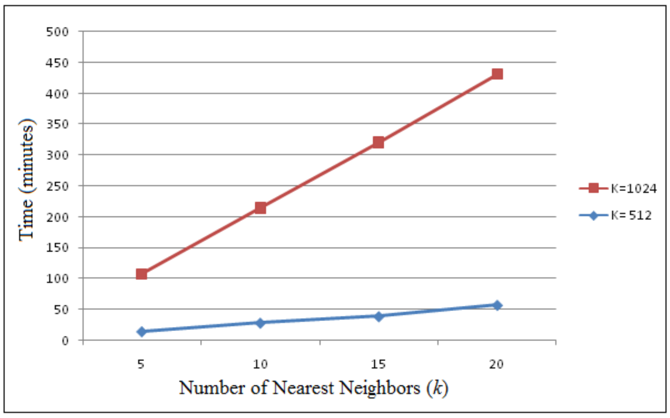

Fig.1. Total cost of SDkNCL

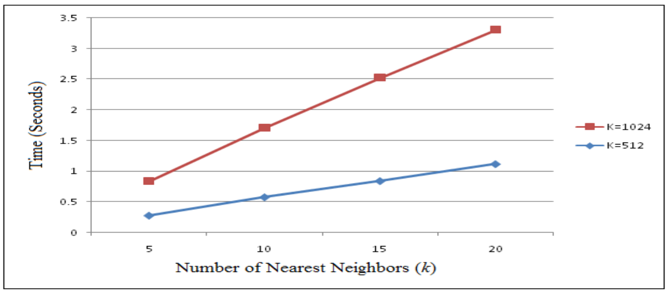

Fig.2. Total cost of SDMCL

With K=512, our SDMCL takes 0.281 to 1.118 seconds when k is scaled from 5 to 20, respectively. On the other hand, with K=1024, our SDMCL takes 0.556 to 2.184 seconds when k is scaled from 5 to 20, respectively. Fig. 2 shows the linear growth in the cost of SDMCL as k is increased. We can observe that, SDMCL incurs very low cost as compared to SD*k*NCL, since the number of computations in I-SMAX are significantly less than those in I-SMIN.

7. CONCLUSION AND FUTURE WORK

Our work in this paper proposed some improvisations to the existing PP*k*NN [2] protocol in order to improve its efficiency and to overcome some of the limitations of its sub- protocols. To address the issue of, negative values as result of any Paillier addition, while performing attribute-wise subtraction in secure squared Euclidean distance (SSED) protocol [3], has been resolved by our I-SSED protocol. In this protocol, we introduce a new approach to compute the squared Euclidean distance securely effectively and more efficiently. With the security analysis and practical implementation, we emphasize that, during our improvised protocols, all the privacy requirements mentioned in section 1 are met; the cloud C_1 and C_2 remains unaware of which database records correspond to the derived nearest neighbors. Also, any intermediate values that are computed and are visible to either of the clouds, are encrypted random values or random numbers. The work by Samanthula, Elmehdwi and Jiang to propose PP*k*NN [2] is the first of its type and after evaluating the performance, we can optimistically conclude that our improvised protocols are computationally inexpensive than those proposed in [2] under the same experimental settings.

Executing the I-SMIN protocol in parallel will be a huge boost to compute the k- nearest neighbors. This is can be possible as records are independent of each other and a cluster of servers, to execute I-SMIN in parallel, can be easily configured in a single cloud environment. Similarly, the I-SSED protocol can also be parallelized and we plan to do it in our future work.

REFERENCES

[1] P. Paillier, "Public key cryptosystems based on composite degree residuosity classes," in Eurocrypt, pp. 223–238, 1999.

[2] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "k-Nearest Neighbor Classification over Semantically Secure Encrypted Relational Data," IEEE Transactions on Knowledge and Data Engineering, Volume: 27, 2015.

[3] 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.

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

[5] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in ACM Sigmod Record, volume: 29, pp. 439–450, ACM, 2000.

[6] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in Advances in Cryptology (CRYPTO), pp. 36–54, Springer, 2000.

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

[8] Bohanec and B. Zupan. The UCI KDD Archive, 1997. http://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

BIOGRAPHIES

Mr. Gaikwad Vijayendra Sanjay received the Bachelors degree in Information technology from Pune Vidhyarthi Griha's College of Engineering and Technology in 2014. He is currently pursuing Masters degree in Computer Science and Engineering from Deogiri Institute of Engineering and Management Studies. Areas of interest are data mining and cloud backup techniques and secured cloud data processing.

Dr. Khan Rahat Afreen received the B.E. degree in Computer Science & Engineering from Marathwada Institute of Technology in 2001 and M.E. degree in Computer Science & Engineering from Government Engineering College, Aurangabad. She has also received her Ph.D in Computer Science & Engineering with the topic 'Elliptic Curve Cryptography' and is currently working as Associate Professor in Deogiri Institute of Engineering and Management Studies. Her areas of specializations include cryptography, elliptic curve cryptography, cryptography related mathematics and field based arithmetic.