PLOS ONE

RESEARCH ARTICLE

Privacy-preserving parallel kNN classification algorithm using index-based filtering in cloud computing

Yong-Ki Kim[1], **Hyeong-Jin Kim**[2], **Hyunjo Lee**[2], **Jae-Woo Chang**[2]*

1 Dept. of IT Convergence System, Vision College of Jeonju, Jeonju, Jeollabuk-do, Republic of Korea, **2** Dept. of Computer Engineering, Chonbuk National University, Jeonju, Jeollabuk-do, Republic of Korea

* jwchang@jbnu.ac.kr

OPEN ACCESS

Citation: Kim Y-K, Kim H-J, Lee H, Chang J-W (2022) Privacy-preserving parallel kNN classification algorithm using index-based filtering in cloud computing. PLoS ONE 17(5): e0267908. https://doi.org/10.1371/journal.pone.0267908

Editor: Hua Wang, Victoria University, AUSTRALIA

Received: December 7, 2021

Accepted: April 18, 2022

Published: May 5, 2022

Data Availability Statement: The real data used are available in the following URL: (http://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King%29). It contains 28,056 instances with 6 attributes. The data belongs to a third party. The authors did not have any special access privileges to the data.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2019R1I1A3A01058375). The funders had no role in study design, data collection and analysis,

Abstract

With the development of cloud computing, interest in database outsourcing has recently increased. In cloud computing, it is necessary to protect the sensitive information of data owners and authorized users. For this, data mining techniques over encrypted data have been studied to protect the original database, user queries and data access patterns. The typical data mining technique is kNN classification which is widely used for data analysis and artificial intelligence. However, existing works do not provide a sufficient level of efficiency for a large amount of encrypted data. To solve this problem, in this paper, we propose a privacy-preserving parallel kNN classification algorithm. To reduce the computation cost for encryption, we propose an improved secure protocol by using an encrypted random value pool. To reduce the query processing time, we not only design a parallel algorithm, but also adopt a garbled circuit. In addition, the security analysis of the proposed algorithm is performed to prove its data protection, query protection, and access pattern protection. Through our performance evaluation, the proposed algorithm shows about 2 ~ 25 times better performance compared with existing algorithms.

1 Introduction

With the growing popularity of cloud computing, there has been growing interest in outsourcing databases. Cloud computing provides a service that allows internet-connected users to use virtual computing resources such as storage, computation, and network. Thus, a cloud service provider can maintain computing resources rapidly and flexibly. A data owner can reduce efforts to purchase, install, and expand computing systems, and mitigate the constraints of physical space. Cloud computing is attracting a lot of attention from individuals and companies because it can reduce the cost of system maintenance and data management, and can utilize computing resources needed without expertise. Meanwhile, we should consider three requirements in an outsourced database. First, it is necessary to protect the database because the database contains sensitive information of the data owner [1, 2]. Second, the query and the query result should not be exposed because personal information related to user preference

decision to publish, or preparation of the manuscript.

Competing interests: NO authors have competing interests.

may be uncovered. Third, data access patterns should be protected because the cloud provider is able to infer private information from the data access pattern.

Therefore, Data Mining over Encrypted Data (DMED) has been studied to protect the original database, user queries and data access patterns. Early studies modify plaintexts to substituted data and outsources them to a cloud [3–7]. However, these early studies have a disadvantage in that they cannot completely protect data and queries because they are vulnerable to various attacks such as chosen-plaintext attacks. To solve this problem, recent studies encrypt the database and outsource the encrypted database to the cloud [8–15]. Before a data owner outsources his/her database to a cloud service provider (cloud provider), he/she encrypts the database. The cloud provider processes the query received from an authorized user. The cloud provider can perform data management and system maintenance instead of the data owner. The authorized user can directly request the desired results from the cloud provider. The process of query processing over the outsourced database is shown in Fig 1.

Among DMED, the kNN classification algorithm is widely used for three reasons. First, the kNN classification algorithm has a relatively higher accuracy than other classification algorithms. Second, with the addition of more data, the kNN classification algorithm constantly evolves and is capable of quickly adapting to the changes in input dataset. Finally, the kNN classification algorithm gives a user a flexibility to choose a distance measure metric. Therefore, the kNN classification algorithm is used for various applications such as pattern analysis, image analysis, and user analysis [16].

Samanthula et al.'s work [16] and Kim et al.'s work [17] proposed kNN classification algorithms based on homomorphic encryption which can support various operations without decryption. Recent studies can also support data privacy, query privacy and hiding data access patterns. However, while processing the kNN classification algorithm, the recent works require high computation cost because they need to add random noise data to prevent exposure of the original data. Moreover, they require a large amount of processing time for kNN classification over the encrypted database. To the best of our knowledge, there is no existing parallel kNN classification algorithm which is suitable for processing a large amount of encrypted data.

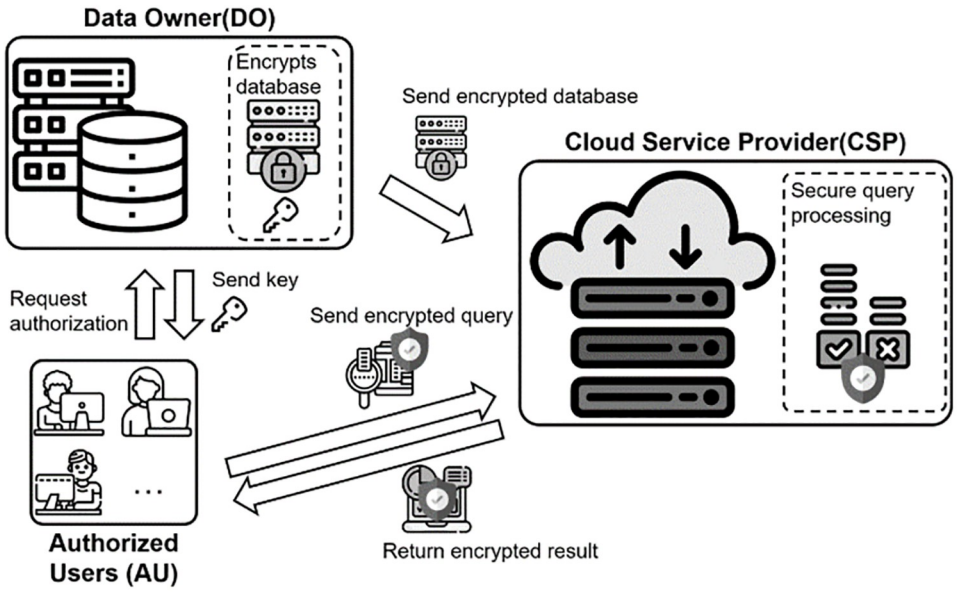


Fig 1. Database outsourcing model.

https://doi.org/10.1371/journal.pone.0267908.g001

The motivation of this paper is as follows. First, the existing algorithms suffer from high computational cost by using encrypted binary array to perform comparison operations. Therefore, we aim at reducing computational cost by proposing secure comparison protocol based on Yao's garbled circuit. Second, the existing algorithms require high data encryption cost. To deal with this problem, we propose an improved secure protocol by using an encrypted random value pool. Finally, to the best of our knowledge, there is no existing parallel kNN classification algorithm. We aim at designing a parallel kNN classification algorithm for processing a large amount of encrypted data.

The contributions of this paper are as follows.

- Supporting privacy preservation: By processing queries using homomorphic encryption without data decryption, we can protect the confidentiality of both data and user's queries while hiding data access patterns from an attacker.
- Reducing computation cost: By using the improved secure protocol based on an encrypted random value pool, we can reduce the high computation cost of the random value generation for the data encryption.
- Improving the performance of kNN classification: By proposing a new parallel kNN classification algorithm, we can reduce the amount of processing time for kNN classification.

The rest of this paper is as follows. In Section 2, we introduce the existing works on kNN classification algorithms over the encrypted database. In Section 3, we describe the overall system architecture and propose secure protocols for the proposed parallel kNN classification algorithm. In Section 4, we propose a parallel kNN classification algorithm that preserves both data and query privacy on the cloud. In Section 5, we provide the security proof of our kNN classification algorithm. In Section 6, we perform a performance analysis of the proposed algorithm. In Section 7, we describe the impact of the proposed parallel classification algorithm as a discussion. Finally, in Section 8, we conclude our paper with the future work.

2 Background and related work

2.1) Background

2.1.1 Paillier cryptosystem. The Paillier cryptosystem is a probabilistic asymmetric algorithm for public key cryptography [18]. In the Paillier cryptosystem, the encryption key pk is given as (N, g), where N is the multiplication value between two large prime numbers p and q in Z_{N^2}. Here, g is a random integer value at Z_{N^2} where Z_{N^2} denotes an integer domain ranged from 0 to Z_{N^2}. Meanwhile, the decryption key sk is given as (p, q). The Paillier cryptosystem has the following characteristics. First the Paillier cryptosystem can support homomorphic addition and multiplication. Assume that the encryption function of the Paillier cryptosystem is E(.) and its decryption function is D(.), For two encrypted data E(a) and E(b), the product E (a) × E(b) is equal to E(a+b), which is the encrypted value of the plaintext a+b, as shown in Eq (1).

$$E(a+b) = E(a) \times E(b) \ mod \ n^2 \tag{1}$$

For two plaintexts a and b, the b^th power of the encrypted data E(a), i.e, $E(a)^b$, is equal to E (a × b), which is the encrypted value of the plaintext a × b, as shown in Eq (2).

$$E(a \times b) = E(a)^b \ mod \ n^2 \tag{2}$$

Second, the Paillier cryptosystem supports semantic security where only negligible information about the plaintext can be feasibly extracted from the ciphertext. Specifically, any

probabilistic, polynomial-time algorithm (PPTA), which is given the ciphertext of a certain message m and its length, cannot determine any partial information on the message with a probability higher than all other PPTA's that only have access to the message length [19]. This concept is the computational complexity similar to Shannon's concept of perfect secrecy. Perfect secrecy means that the ciphertext reveals no information at all about the plaintext, whereas semantic security implies that any information revealed cannot be feasibly extracted.

2.1.2 Attack model. In the outsourcing database environment, two attack models can be considered: a semi-honest attack model and a malicious attack model [20]. In the semi-honest (or honest-but-curious) attack model, the cloud performs its own protocol honestly, but attempts to obtain sensitive data about the data owner and the authorized user during the protocol execution. To prevent a semi-honest attack, sensitive data must always be protected. A malicious attack model attempts to acquire sensitive data by deviating from a given secure protocol. Because a secure protocol can be contaminated by a malicious attack, it is difficult to recover the secure protocol. To protect sensitive data against the malicious attack model, a defender focuses on detecting attacks and recovering the damaged secure protocol. Since we aim at protecting sensitive data in cloud computing, we design our algorithm based on the semi-honest attack model. A secure protocol for the semi-honest attack model is defined as follows [17].

Definition 1. Assuming that a_i is the input data of cloud C_i, $\prod_i(\pi)$ is the execution image of C_i for the protocol π and b_i is the result data of C_i executing the π protocol. If the execution image $\prod_{Si}(\pi)$ simulating π is computationally indistinguishable from $\prod_i(\pi)$, the protocol π is said to be a secure protocol for the semi-honest attack model.

In Definition 1, the execution image generally includes the input data and output data of the protocol. The security of the protocol under the semi-honest attack model can be verified by showing that the protocol's execution image does not expose the cloud's data.

2.2) Related work

2.2.1 B. Yao et al.'s work. B. Yao et al. proposed a secure kNN classification algorithm [21] based on a partition-based secure Voronoi diagram (SVD) [22]. The SVD relies on any standard encryption scheme E such as public-key encryption RSA and symmetric-key encryption AES, rather than using any new encryption schemes. Because the SVD is as secure as E for any standard security model in which E is proven secure, the SVD is indistinguishable in either chosen plaintext or chosen ciphertext attacks. To process the secure kNN classification queries, the algorithm retrieves the relevant encrypted partition instead of finding the encrypted exact k-nearest neighbors. However, most of the computations are performed locally by the end-user while processing the kNN classification query. As a result, the algorithm conflicts the purpose of outsourcing the DBMS functionalities to the cloud. Furthermore, the algorithm leaks data access patterns to the cloud, such as the partition ID corresponding to a user query.

2.2.2 B. K. Samanthual et al.'s work. B. K. Samanthula et. al. proposed a secure k-NN classification algorithm, denoted by PPkNN, over encrypted data in the cloud [16]. PPkNN can protect the confidentiality of the data, user's input query, and data access patterns. PPkNN mainly consists of two stages: the secure retrieval of k-nearest neighbors and the secure computation of majority class. In the secure retrieval of k-nearest neighbors, a query user initially sends his query q (in encrypted form) to C_1. Then, C_1 and C_2 involve in a set of sub-protocols to securely retrieve the class labels corresponding to the k-nearest neighbors of the input query q. At the end of this step, the encrypted class labels of the k-nearest neighbors are known only to C_1. In the secure computation of the majority class, C_1 and C_2 jointly compute the class label with majority voting among the k-nearest neighbors of q. At the end of this step, only the

query user knows the class label corresponding to input query record q. However, PPkNN requires a very high computation cost for hiding data access patterns.

2.2.3 H. Kim et al.'s work. H. Kim et. al. proposed a secure kNN classification algorithm which uses both the Paillier cryptosystem and an encrypted kd-tree index [17]. The Paillier cryptosystem is a homomorphic encryption scheme which is indistinguishable in either chosen-plaintext or chosen-ciphertext attacks, so that the cloud can process the kNN classification queries without decrypting any data or a user's query. Before outsourcing data to the cloud, a data owner builds a kd-tree index and encrypts both the original database and the leaf nodes of the kd-tree index. Therefore, the algorithm can protect the data, the query and the data access pattern. By using the encrypted kd-tree index, the algorithm can reduce the amount of query processing time. However, because the algorithm must generate encrypted random values for privacy-preserving, it requires a high computation cost.

2.2.4 W. Wu et al.'s work. W. Wu et al. proposed a privacy preserving kNN classification algorithm over encrypted database in outsourced cloud environments [23]. The algorithm newly generates unique classification label keys for each user through a secure three-party protocol. The keys are used to re-encrypt the labels into new ciphertexts that can only be decrypted by the corresponding user. The algorithm hides the data access patterns from a federated cloud server which performs the process of kNN classification by using two non-colluding clouds. However, the algorithm conflicts the purpose of outsourcing the DBMS functionalities to the cloud because both the data owner and authorized users must participate in the process of label re-encryption.

2.2.5 Y. Tan et al.'s work. Y. Tan et al. proposed a lightweight edge-based privacy-preserving kNN classification algorithm over a hybrid encrypted cloud database [24]. A data owner can upload his/her database to the cloud server, and an authorized user can send a query to the cloud server to execute kNN queries. The algorithm is performed against the semi-honest attack model. After the query is sent, the authorized user does not need to participate in the kNN classification. They also proposed a secure distance protocol in which the cloud servers cannot derive any private information from the authorized user. Compared with the SIP protocol in the state-of-the-art PPKC algorithm [16], the proposed secure distance protocol has less corrupted computation.

2.2.6 J. Du and F. Bian's work. J. Du and F. Bian proposed a non-interactive and efficient privacy-preserving kNN classification algorithm [25]. The algorithm is performed against the semi-honest attack model. To achieve privacy preservation, the algorithm encrypts all outsourced data and users' query records by using two encryption schemes: order preserving encryption [26] and the Paillier cryptosystem [16]. To hide the data access pattern, the information in the cloud server is always maintained in ciphertext format. In terms of classification accuracy, the algorithm is proven to be very close to one using both plaintext data and the non-interactive encrypted data query scheme.

Table 1 shows the comparison of the existing studies. We explain their comparison with respect to three major factors. First, B. K. Samanthula et al.'s work [16], H. Kim et al.'s work [17], W. Wu et al.'s work [23] and Y. Tan et al.'s work [24] support hiding access pattern, while B. Yao et al.'s work [21] and J. Du and F. Bian's work [25] do not support it. Second, W. Wu et al.'s work and Y. Tan et al.'s work require low computation overhead while B. K. Samanthula et al.'s work and H. Kim et al.'s work need high computation overhead. Finally, B. Yao et al.'s work, B. K. Samanthula et al.'s work, H. Kim et al.'s work and W. Wu et al.'s work have low risk in terms of security, while Y. Tan et al.'s work and J. Du and F. Bian's work have high risk in terms of security.

Table 1. Comparison of existing studies.

	Hiding access patterns	Index	Computation overhead	Encryption	User involvement	Exact match / Approximate match	Security risk
B. Yao et al.'s work [21]	Not support	Secure Voronoi diagram	Moderate	Any Standard Encryption	Involved	Approximate match	Low
B.K. Samanthula et al.'s work [16]	Support	None	Very high	Paillier	Not involved	Exact match	Low
H. Kim et al.'s work [17]	Support	Encrypted kd-tree	High	Paillier	Not involved	Exact match	Low
W. Wu et al.'s work [23]	Support	None	Low	self-production	Involved	Exact match	Low
Y. Tan et al.'s work [24]	Support	None	Low	Paillier (data), ElGamal (Classification label)	Not involved	Exact match	High
J. Du and F. Bian's work [25]	Not support	Encrypted kd-tree	Moderate	Paillier (label), Order preserving encryption(OPE)	Involved	Approximate match	High

https://doi.org/10.1371/journal.pone.0267908.t001

3 Overall system architecture and secure protocols

3.1) System architecture

In the outsourcing database environment, two attack models can be considered: a malicious attack model and a semi- honest attack model [20]. In a malicious attack model, the cloud can deviate from the protocol procedure. A protocol against malicious attack model is inefficient because it requires exceedingly high cost. In the semi-honest attack model, the cloud correctly follows the given protocol, but tries to acquire the sensitive information of both the data owner and the query issuer. However, a protocol against a semi-honest attack model is practical because the cloud has a higher level of authority than outsider attackers. Therefore, according to earlier work [16, 17], we also adopt the semi-honest attack model. Table 2 shows a list of

Table 2. Definitions of common notations.

Notations	Description
$E(\cdot)$, $D(\cdot)$	Encryption function and decryption function
t_i, $t_{i,j}$	i^{th} record and j^{th} attribute value of i^{th} record
t'_i, $t'_{i,j}$	i^{th} extracted record during the index search and its j^{th} attribute value
q, q_j	A query of a user and j^{th} attribute value of a query q
n, m	The total number of data and attributes in T
cnt	The number of data extracted during the index search step
h	Level of the kd-tree
$node$	z^{th} node of the kd-tree
$node_z.t_{s,j}$	j^{th} attribute of s^{th} record stored in z^{th} node of the kd-tree
$lb_{z,j}$, $ub_{z,j}$	j^{th} attribute value of lower/upper bound of z^{th} kd-tree node
F	Fan-out(maximum # of data in each leaf node)
α_z	Output of SCMP or SPE protocol for node
α	A set of values consisting of α
l	Domain size (in bits)
r	Random integers
L	Array for entire label
w	The number of labels
L'	k-array of labels for k-nearest neighbors

https://doi.org/10.1371/journal.pone.0267908.t002

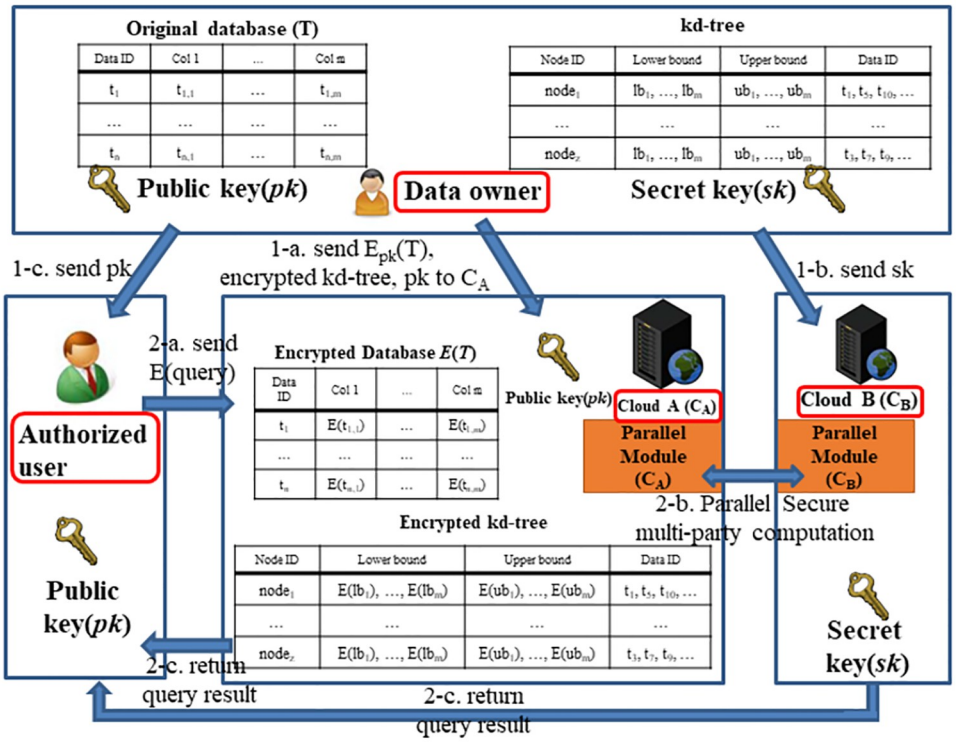


Fig 2. Overall system architecture.

https://doi.org/10.1371/journal.pone.0267908.g002

notations used in this paper. Our system architecture supports secure protocols between clouds by performing Secure Multiparty Computation (SMC). SMC is based on multi-party data processing in which several entities cooperate to perform calculations for deriving specific results. For this, the following factors must be satisfied to achieve the result of secure protocols while avoiding data leakage.

3.1.1 Input privacy. No information about private data held by multiple parties can be inferred from the messages sent during the protocol execution. The only information that can be inferred about private data is whatever could be inferred from seeing the output of the function alone.

3.1.2 Correctness. Any proper subset of adversarial colluding parties that is willing to share information or deviate from the instructions during the protocol execution should not be able to force honest parties to output an incorrect result. This correctness goal comes in two categories: either the honest parties are guaranteed to compute the correct output (a robust SMC protocol), or the honest parties abort if they find an error (an SMC protocol with abort).

Fig 2 shows the overall system architecture. The data owner holds the original database T consisting of n records t_i ($1 \leq i \leq n$). Each record t_i includes m attributes (or columns) and one label. Here, we call the j^{th} attribute of the i^{th} record as $t_{i,j}$($1 \leq i \leq n$, $1 \leq j \leq m + 1$). First, the data owner partitions the original data by using the kd-tree index. Assuming that the level of the constructed kd-tree is h, the total number of leaf nodes is 2^{h-1}. In the leaf node, an attribute stores its region information, i.e., a lower bound $lb_{z,j}$ and an upper bound $ub_{z,j}$, where $1 \leq z \leq 2^{h-1}$ and $1 \leq j \leq m$. Second, the data owner generates an encryption public key (pk) and a decryption secret key (sk) based on the Paillier cryptosystem [18]. Third, the data owner encrypts the database with the Paillier cryptosystem to protect the original data. Because the

unit of the encryption is the attribute of each record, $E(t_{i,j})$ $(1 \leq i \leq n, 1 \leq j \leq m+1)$ is generated. Finally, the leaf node of the constructed kd-tree is encrypted because the data owner needs to protect the data access pattern. Because the unit of the encryption is the attributes of each leaf node, $E(lb_{z,j})$ and $E(ub_{z,j})$ are generated$(1 \leq z \leq 2^{h-1}, 1 \leq j \leq m)$.

3.2) Secure protocols

3.2.1 Encrypted random value pool. To support data privacy in a cloud computing environment, the existing works [16, 17] prevent C_B from extracting meaningful information (Fig 2) while executing a secure protocol by using the Paillier cryptosystem. However, they require high computation cost because the secure protocol generates an encrypted random value for protecting the original data. Therefore, we propose an encrypted random value pool to reduce the computation cost for encryption. Before C_A processes a query (Fig 2), we generate the random plaintext from Z_N and store the encrypted random plaintext into an encrypted random value pool. While processing a query in C_A, a random ciphertext is selected from the encrypted random value pool whenever a secure protocol is called. Therefore, while processing a secure protocol, C_A not only prevents C_B from extracting meaningful information, but also reduces the cost of generating encrypted random values. Table 3 shows a comparison of the number of data encryptions for each secure protocol in our work and existing works [16, 17]. The Secure Multiplication protocol of the existing works requires three times as the number of encryptions as our work. The Secure Compare protocol used in B. K. Samanthula et al.'s work requires $\log_2 D$ times as the number of encryptions as our work while the one used in H. Kim et al.'s work requires three times as the number of encryptions as our work.

3.2.2 Secure multiplication protocol using an encrypted random value pool. We propose a Secure Multiplication protocol using an Encrypted random value pool (SME protocol) which multiplies two encrypted values $E(\alpha)$ and $E(\beta)$. Algorithm 1 shows the SME protocol. First, when two encrypted values $E(\alpha)$ and $E(\beta)$ are given as inputs, C_A selects two random values $E(r_a)$ and $E(r_b)$ from the encrypted random value pool (line 1). Second, C_A calculates $E(\alpha + r_a)$ and $E(\beta + r_b)$ by using Eq (1), then sends them to C_B (line $2 \sim 3$). Third, C_B decrypts $E(\alpha + r_a)$ and $E(\beta + r_b)$ by using the secret key and calculates the multiplication of the two plaintext $\alpha + r_a$ and $\beta + r_b$ (line 4). Fourth, C_B encrypts $(\alpha + r_a) \times (\beta + r_b)$ and send it to C_A (line 5). Finally, C_A obtains $E(\alpha \times \beta)$ by removing $\alpha \times r_b$, $\beta \times r_a$ and $r_a \times r_b$ from the received value, where 'N^{-x}' in the Z_N domain is the same as '-x' (line 6).

Algorithm 1 SME Protocol

```
Input: E(α), E(β)
Output: E(α × β)
C_A:
1: Pick random value E(r_a) and E(r_b) in the encrypted random value pool
2: E(α′)←E(α) × E(r_a); E(β′)←E(β) × E(r_b)
3: Send E(α′), E(β′) to C_B
C_B:
4: h ← D(E(α′) × D(E(β′) mod N // h = α × β + α × r_b + β × r_a + r_a × r_b
5: Send E(h) to C_A
```

Table 3. A comparison of amount in secure protocols.

algorithms	Secure Multiplication Protocol	Secure Compare Protocol
B. K. Samanthula et al.'s work	$3 \times E$	$log_2 D \times E$
H. Kim et al's work	$3 \times E$	$3 \times E$
Proposed algorithm	$1 \times E$	$1 \times E$

https://doi.org/10.1371/journal.pone.0267908.t003

C_A:

6: $E(\alpha \times \beta) \leftarrow E(h) \times E(\alpha)^{N-r_b} \times E(\beta)^{N-r_a} \times E(r_a \times r_b)^{N-1}$

3.2.3 Garbled secure compare protocol using encrypted random value pool. We propose the Garbled Secure Compare protocol using an Encrypted random value pool (GSCE protocol) which is performed by using a garbled circuit consisting of two ADD gates and one CMP gate [27]. Assume that $E(u)$ and $E(v)$ are ciphertext for two plaintext u and v. When $E(u)$ and $E(v)$ are given to C_A, the GSCE protocol returns $E(1)$ if $u \leq v$ is satisfied, otherwise it returns $E(0)$. Algorithm 2 shows the GSCE protocol. First, C_A selects two random value $E(r_u)$ and $E(r_v)$ from the encrypted random value pool (line 1). Second, C_A calculates $E(m_1) = E(u)^2 \times E(r_u)$ and $E(m_2) = E(v)^2 \times E(1) \times E(r_v)$ (line 1 ~ 2). Third, C_A randomly selects one of two random functions, i.e., F_0 and F_1. The selected random function is not disclosed to C_B. If C_A selects F_0, C_A sends an encrypted ordered pair $<E(m_2), E(m_1)>$ to C_B. If C_A selects F_1, C_A sends an encrypted ordered pair $<E(m_1), E(m_2)>$ to C_B (line 3 ~ 7). Fourth, C_B decrypts the data received from C_A (line 8 ~ 11). When C_A selects F_0, C_B acquires an ordered pair $<m_2, m_1>$, otherwise C_B acquires an ordered pair $<m_1, m_2>$. Fifth, C_A creates a garbled circuit consisting of two ADD gates and one CMP gate. If F_0 is selected, $-r_v$ and $-r_u$ are transferred to the first ADD gate and the second ADD gate, respectively. Otherwise, $-r_u$ and $-r_v$ are transferred to the first and the second ADD gates, respectively (lines 12 ~ 16). Sixth, C_B transfers the first data to the first ADD gate, and the second data to the second ADD gate. Therefore, when F_0 is selected, C_B transfers m_2 and m_1 to the first and the second ADD gates, respectively. Otherwise, m_1 and m_2 are transferred to the first and the second ADD gates, respectively (line 17 ~ 20). Seventh, the first ADD gate adds two input values: $-r_v$ and m_2 for F_0 and $-r_u$ and m1 for F_1. The result of the first ADD gate ($result_1$) is transferred to the CMP gate (line 21 ~ 24). Eighth, the second ADD gate adds two input values: $-r_u$ and m_1 for F_0 and $-r_v$ and m_2 for F_1. The result of the second ADD gate ($result_2$) is transferred to the CMP gate (line 25 ~ 28). Due to the characteristics of the garbled circuit, the exposure of any information does not occur in the ADD gate. Ninth, the CMP gate returns $\alpha = 1$ if $result_1 \leq result_2$, and $\alpha = 0$ otherwise (line 29 30). Finally, the result α can be checked on C_B side, and C_B transmits $E(\alpha)$ to C_A (line 31). Because C_B does not know whether F_0 or F_1 is selected by C_A, C_B cannot determine the result of comparison of $E(u)$ and $E(v)$. When F_0 is selected, C_A changes $E(\alpha)$ through the SBN protocol [11] and returns $E(\alpha)$ (line 32 ~ 34). Here, C_A cannot obtain the actual value of α due to the characteristics of the Pallier cryptosystem.

Algorithm 2 GSCE Protocol

Input: $E(u)$, $E(v)$
Output: $E(1)$ when $u \leq v$, $E(0)$ otherwise
C_A:
01: Pick random value $E(r_u)$ and $E(r_v)$ in the encrypted random value pool
02: $E(m_1) \leftarrow E(u)^2 \times E(r_u)$
03: $E(m_2) \leftarrow E(v)^2 \times E(1) \times E(r_v)$
04: $h \leftarrow D(E(\alpha')) \times D(E(\beta')) \bmod N$ // $h = \alpha \times \beta + \alpha \times r_b + \beta \times r_a + r_a \times r_b$
05: Randomly choose F_0 or F_1
06: If F_0 u>v is chosen, then
07: Send $<E(m_2), E(m_1)>$ to C_B
08: else
09: Send $<E(m_1), E(m_2)>$ to C_B
C_B:
10: If F_0 u>v is chosen, then
11: Obtain $<m_2, m_1>$ by decrypting $<E(m_2), E(m_1)>$
12: else
13: Obtain $<m_1, m_2>$ by decrypting $<E(m_1), E(m_2)>$
C_A:

14: Generate garbled circuit
15: If F_0 u>v is chosen, then
16: Put $-r_v$ and $-r_u$ into 1^{st} and 2^{nd} ADD gates
17: else
18: Put $-r_u$ and $-r_v$ into 1^{st} and 2^{nd} ADD gates
C_B:
19: If F_0 u>v is chosen, then
20: Put m_2 and m_1 into 1^{st} and 2^{nd} ADD gates
21: else
22: Put m_1 and m_2 into 1^{st} and 2^{nd} ADD gates
1^{st} ADD Gate:
24: If F_0 u>v is chosen, then
25: $result_1 = calculate - r_v + (v + r_v)$
26: else
27: $result_1 = calculate - r_u + (u + r_u)$
2^{nd} ADD Gate:
29: If F_0 u>v is chosen, then
30: $result_2 = calculate - r_u + (u + r_u)$
31: else
32: $result_2 = calculate - r_v + (v + r_v)$
CMP Gate
34: If $result_1 > result_2$ is chosen, then
35: output $\alpha = 1$ to C_B
36: else
37: output $\alpha = 0$ to C_B
C_B:
38: $E(\alpha) \leftarrow$ encrypt α
C_A:
39: If F_0 u>v is chosen, then
40: $E(\alpha) \leftarrow SBN(E(\alpha))$
41: Return $E(\alpha)$

4 Privacy-preserving parallel kNN classification algorithm using index filtering

The proposed parallel kNN classification algorithm can support the protection of data, query, and data access pattern in a cloud computing environment. For this, the proposed privacy-preserving parallel kNN classification algorithm is composed of four phases: secure index search, k-nearest neighbors search, kNN verification, and kNN classification, as shown in Fig 3.

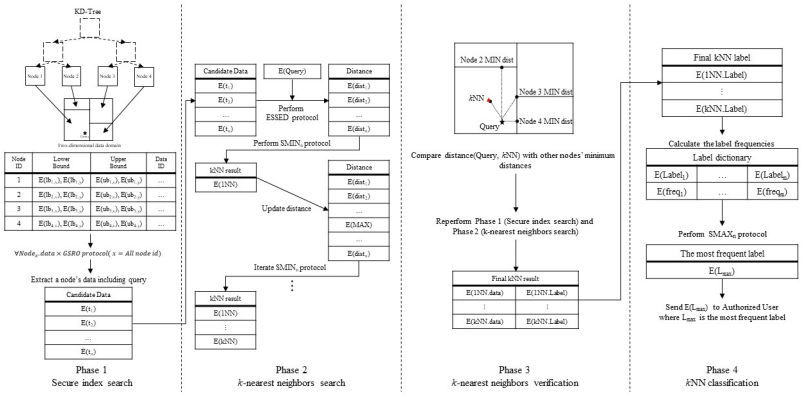


Fig 3. Proposed parallel kNN classification algorithm.

https://doi.org/10.1371/journal.pone.0267908.g003

4.1) Secure index search phase

In the secure index search phase, the proposed algorithm determines the leaf node which includes the given query in the encrypted kd-tree. The procedure of the secure index search is shown in Algorithm 3. First, C_A makes t number of partitions and allocates them to the given threads (line 1). Here, t is calculated by dividing the number of leaf nodes by the number of threads. Second, by using the GSRO protocol, the algorithm finds which leaf node includes the query in each thread. If a node includes the query, the GSRO protocol returns $E(1)$, otherwise the protocol returns $E(0)$. The result of the GSRO protocol is stored in an array $E(\alpha)$. The algorithm randomly reorders the members of the array $E(\alpha)$ and transfers the reordered array $E(\alpha')$ to C_B(line 2 ∼ 7). Third, C_B decrypts the array $E(\alpha')$ and makes groups by allocating the decrypted members uniformly based on the number of 1s. If a node has the decrypted value of 1, it becomes a seed of a group. C_B sends groups to C_A (line 8–15). Finally, C_A extracts all the encrypted data in the node corresponding to $E(1)$. If a node has $E(1)$, the algorithm can safely extract the data of the node because the node includes the query. Otherwise, the algorithm can remove the data of the node because it does not include the query (line 16 ∼ 30).

Algorithm 3 Secure Index Search

Input: $kd-tree = E(node_1), ..., E(node_{Num_{Node}}), query = E(q)$

Output: $E(\alpha_1), ..., E(\alpha_{Num_{Node}}) | \alpha \in (0, 1)$

```
C_A:
01: t = NumNode/NumThread
02: Run thread
03: for 1 ≤ i ≤ NumThread
04:   for t × (i − 1)≤j ≤ t × i
05:     E(α_j) = GSRO(E(q), E(q), E(range_j.lb), E(range_j.ub))
06: Terminate thread
07: E(α′) = ∏(E(α)); Send E(α′) to C_B
C_B:
08: α′ = D(E(α′))
09: c = the number of'1' in α′
10: Create c number of node groups
11: for each node group
12:   assign a node with α′ = 1
13:   assign (num_node/c) − 1 nodes with α′ = 0
14:   shuffle the sequence of nodes
15: Send node group to C_A
C_A:
16: cnt = 0
17: for each node group
18:   permute node IDs using Π^{-1}
19: t = F/NumThread
20: Run thread
21: for 1 ≤ i ≤ NumThread
22:   for each node group
23:     for t × (i − 1)≤s ≤ t × i
24:       for 1 ≤ z ≤ num //num is # of nodes in the selected group
25:         E(t′_{z,j}) ← SM(node_z.t_{s,j}, E(α_z)) for 1 ≤ j ≤ m + 1
26:       for 1 ≤ j ≤ m + 1
27:         E(cand_{cnt,j}) ← ∏_{z=1}^{num} E(t′_{z,j})
28:       cnt ← cnt + 1
29: Terminate thread
30: return E(cand)
```

4.2) k-Nearest neighbors search phase

In the k-nearest neighbors search phase, our algorithm finds k-nearest points among the encrypted candidates which are extracted from the index search phase. The procedure of the k-nearest neighbors search is shown in Algorithm 4. First, C_A calculates the squared Euclidean distance set $E(d_i)$ $(1 \leq i \leq cnt$, where cnt is the number of candidates) between the query and the encrypted candidates through the ESSED protocol [17] in a parallel way (line 1 ∼ 6). Second, C_A finds the minimum value $E(d_{min})$ among $E(d_i)(1 \leq i \leq cnt)$ through the SMS_n protocol [16] (line 8–10). Additionally, C_A calculates the difference between $E(d_{min})$ and $E(d_i)$ $(1 \leq i \leq cnt)$ by using $E(d_{min}) \times E(d_i)^{N-1}$, and stores the results into an array $E(\tau_i)$ $(1 \leq i \leq cnt)$. C_A makes $E(\tau'_i)$ $(1 \leq i \leq cnt)$ by raising $E(\tau_i)$ to the power of a random integer. C_A makes $E(\beta^i)$ $(1 \leq i \leq cnt)$ by applying a shuffling function π to $E(\tau'_i)$ $(1 \leq i \leq cnt)$ and sends it to C_B (line 11 ∼ 18). Therefore, the original distance and data access patterns are protected from C_B. Third, if the i^{th} decrypted value of $E(\beta_i)$ $(1 \leq i \leq cnt)$ is 0, C_B sets to $E(1)$ the i^{th} value of a temporary array $E(U_i)$ $(1 \leq i \leq cnt)$. Otherwise, C_B sets to $E(0)$ the i^{th} value of a temporary array $E(U_i)(1 \leq i \leq cnt)$. C_B sends $E(U_i)(1 \leq i \leq cnt)$ to C_A (line 19 ∼ 22). Fourth, C_A makes $E(V_i)(1 \leq i \leq cnt)$ by applying a deshuffling function $\pi - 1$ to $E(U_i)(1 \leq i \leq cnt)$. C_A performs the SM protocol between $E(V_i)(1 \leq i \leq cnt)$ and $E(cand_{i,j})(1 \leq i \leq cnt$ and $1 \leq j \leq m + 1$, where m is the data dimension). C_A stores the result of the SM protocol in a temporary array $E(V'_{i,j})(1 \leq i \leq cnt$ and $1 \leq j \leq m + 1)$. Next, C_A calculates Eq 3 by using Eq 1 (line 23 ∼ 31). Fifth, if the algorithm does not find k-nearest neighbors, C_A updates $E(d_i)$ $(1 \leq i \leq cnt)$ by calculating Eq 4 in a parallel way, where $E(max)$ is the maximum value of the data domain (line 32 ∼ 38). If $E(V_i)$ equals to $E(1)$, $E(d_i)$ corresponding to $E(V_i)$ is updated to $E(max)$ through Eq 4. Otherwise, $E(d_i)$ corresponding to $E(V_i)$ is maintained. Finally, C_A terminates the k-nearest neighbors search phase if k-nearest neighbors are found (line 39).

$$E(t'_{s,j}) = \prod_{i=1}^{cnt} E(V_{i,j})(1 \leq j \leq m+1) \tag{3}$$

$$E(d_i) = SM(E(V_i),\ E(max) \times SM(SBN(E(V_i)),\ E(d_i)) \tag{4}$$

Algorithm 4 k-nearest neighbor search phase

```
Input: E(q), E(cand), k
Output: t'//candidatekNNresult
C_A:
01: Run thread
02:   t = NumNode/NumThread
03:   for 1 ≤ i ≤ NumThread
04:     for t × (i − 1)≤j ≤ t × i
05:       E(d_j) = ESSED(E(q), E(cand_j))
06: Terminate thread
07: for 1 ≤ s ≤ k
08:   Run thread
09:     E(d_min) = SMS_n(E(d_1), ..., E(d_cnt))
10:   Terminate thread
11:   Run thread
12:     t = cnt/NumThread
13:     for 1 ≤ i ≤ NumThread
14:       for t × (i − 1)≤j ≤ t × i
15:         E(τ_j) = E(d_min) × E(d_j)^(N−1)
16:         E(τ'_j) = E(τ_j)^(r_i)
17:   Terminate thread
```

```
18:   E(β)←∏(τ_j); Send E(β) to C_B
C_B:
19:   for 1 ≤ i ≤ cnt
20:     If D(E(β_j)) = 0, then E(U_i)←E(1)
21:     Else E(U_i)←E(0)
22:   Send E(U) to C_A
C_A:
23:   E(V)←∏^{-1}(U)
24:   Run thread
25:     t = cnt/NumThread
26:     for 1 ≤ u ≤ NumThread
27:       for t × (u − 1)≤i ≤ t × u
28:         for 1 ≤ j ≤ m + 1
29:           E(V'_{i,j}) ← SM(E(V_i, E(cand_{i,j})))
30:           E(t'_{s,j}) ← ∏_{i=1}^{cnt} E(V'_{i,j})
31:   Terminate thread
32:   Run thread
33:     t = cnt/NumThread
34:     for 1 ≤ i ≤ NumThread
35:       for t × (i − 1)≤j ≤ t × i
36:         If s < k then,
37:           E(d_j) = SM(E(V_j), E(max)) × SM(SBN(E(V_j)), E(d_j))
38:   Terminate thread
39: return E(t')
```

4.3) k-Nearest neighbors verification phase

In the k-nearest neighbors verification phase, the algorithm verifies whether the distance between the a node and the query($E(q) = <E(q_1), E(q_2), \ldots, E(q_m)>$, where m is the data dimension) is shorter than the distance, $E(dist_k)$, between the query and k^{th} nearest neighbor ($E(t') =< E(t'_{k,1}), E(t'_{k,2}), \ldots, E(t'_{k,m}) >$). The procedure of the k-nearest neighbors verification phase is shown in Algorithm 5. First, C_A calculates $E(dist_k)$ between $E(q)$ and $E(t'_k)$ using the ESSED protocol (line 1). Second, the algorithm performs the GSCE protocol between $E(q_j)$ and the lower bound of $node_z(E(node_z.lb_j)$ $(1 \leq z \leq num_{node})$ for each dimension $j(1 \leq j \leq m)$, and stores the result of the GSCE protocol into $E(\psi_{1,j})$. If $E(q_j)$ $(1 \leq j \leq m)$ is less than or equal to $E(node_z.lb_j)$, $E(\psi_{1,j})$ is $E(1)$. Then, the algorithm performs the GSCE protocol between $E(q_j)$ $(1 \leq j \leq m)$ and the upper bound of $node_z$ $(E(node_z.ub_j)(1 \leq z \leq num_{node})$ for each dimension j, and stores the result of the GSCE protocol into $E(\psi_{2,j})$ (line 2 ∼ 5). If $E(q_j)$ is less than or equal to $E(node_z.ub_j)$, $E(\psi_{2,j})$ is $E(1)$. Third, the algorithm performs the SBXOR protocol [16] between $E(\psi_{1,j})$ and $E(\psi_{2,j})$, and stores the result of the SBXOR protocol into $E(\psi_{3,j})$ (line 6). Fourth, the algorithm calculates the shortest point of $node_z$ $(1 \leq z \leq num_{node})$, $E(sp_z) = < E(sp_{z,1}), E(sp_{z,2}), \ldots, E(sp_{z,m}) >$ where m is the data dimension, by using Eqs 5 and 6 (line 7 ∼ 10).

$$f(E(lb_j), E(ub_j)) = \ SM(E(\psi_{1,j}), E(lb_j)) \times SM(SBN(\psi_{1,j}), E(ub_j)) \tag{5}$$

$$E(sp_{z,j}) = SM(E(\psi_{3,j}), E(q_j)) \times SM(SBN(E(\psi_{3,j})),\ f(E(lb_j),\ E(ub_j))) \tag{6}$$

Fifth, C_A calculates the squared Euclidean distance between $E(q)$ and $E(sp_z)(1 \leq z \leq num_{node})$ through the ESSED protocol and stores the result into the shortest distance of the $node_z$, $E(spdist_z)(1 \leq z \leq num_{node})$ (line 11). In addition, C_A updates $E(spdist_z)(1 \leq z \leq num_{node})$ by using Eq 7 (line 12 13). $E(\alpha_z)$ in Eq 7 is the result of the GSRO protocol in algorithm 1. This update avoids an unnecessary index search phase by updating the shortest distance of the node

already searched in the previous phase.

$$E(spdist_z) \leftarrow SM(E(\alpha_z), E(max)) \times SM(SBN(E(\alpha_z)), E(spdist_z)) \quad (7)$$

Sixth, C_A performs the GSCE protocol between $E(spdist_z)$ and $E(dist_k)$, and stores the result into $E(\alpha_z)$ (line 14). If $E(spdist_z)$ is less than $E(dist_k)$, the $node_z$ needs additional searching. Finally, by performing lines 9 ∼ 33 of the secure index search phase, C_A extracts the encrypted data belonging to the $node_z$ and adds them to $E(t')$. In addition, C_A obtains the kNN result array, $E(result_i)(1 \leq i \leq k)$, by performing the k-nearest neighbors search phase (line 15 ∼ 17). C_A stores the label of the k-nearest neighbors into $E(L'_i)(1 \leq i \leq k)$ (line 18 ∼ 19).

Algorithm 5 k-nearest neighbors verification phase

```
Input: E(q), E(node), E(t'), k
Output: result
C_A:
01: E(dist_k) = ESSED(E(q), E(t'_k))
02: for 1 ≤ z ≤ num_node
03:   for 1 ≤ j ≤ m
04:     E(ψ_1) = GSCMP(E(q_j), E(node_z.lb_j))
05:     E(ψ_2) = GSCMP(E(q_j), E(node_z.ub_j))
06:     E(ψ_1) = SBXOR(E(ψ_1), E(ψ_2))
07:     E(temp) = SM(E(ψ_1), E(node_z.lb_j))
08:     E(temp) ←E(temp) × SM(SBN(E(ψ_1)), E(node_z.ub_j))
09:     E(temp) = SM(E(temp), SBN(E(ψ_3)))
10:     E(sp_z,j) = E(temp) × SM(E(ψ_1), E(q_j))
11:   E(spdist_z) = ESSED(E(q), E(sp_z))
12:   E(temp) = SM(E(α_z), E(max))
13:   E(spdist_z) = E(temp) × SM(SBN(E(α_z)), E(spdist_z))
14:   E(α_z)←GSCMP(E(spdist_z), E(dist_k))
15: E(t'')← perform 7 ∼ 36 lines of Algorithm 1
16: E(t')← append E(t'') to E(t')
17: result ← performAlgorithm2
18: for 1 ≤ i ≤ k
19:   E(L'_i) = E(result_i,m+1)
```

4.4) k-Nearest neighbors classification phase

In the kNN classification phase, the algorithm extracts the most frequent label from the label of the k-nearest neighbors, $E(L'_i)(1 \leq i \leq k)$. The procedure of the kNN classification phase is shown in Algorithm 6. C_A and C_B calculate the frequency of $E(L'_i)(\ 1 \leq i \leq k)$ by using the secure frequency protocol [17] (line 1). The label with the highest frequency is selected (line 2). C_A adds a random integer r_q to the selected label and stores the result into a temporary variable $E(r_q)$ (line 3). C_A sends $E(r_q)$ to C_B and r_q to AU (line 4). C_B decrypts $E(r_q)$ and sends it to AU (line 5–6). AU obtains the final result by combining the results of C_A and C_B (line 7 ∼ 8).

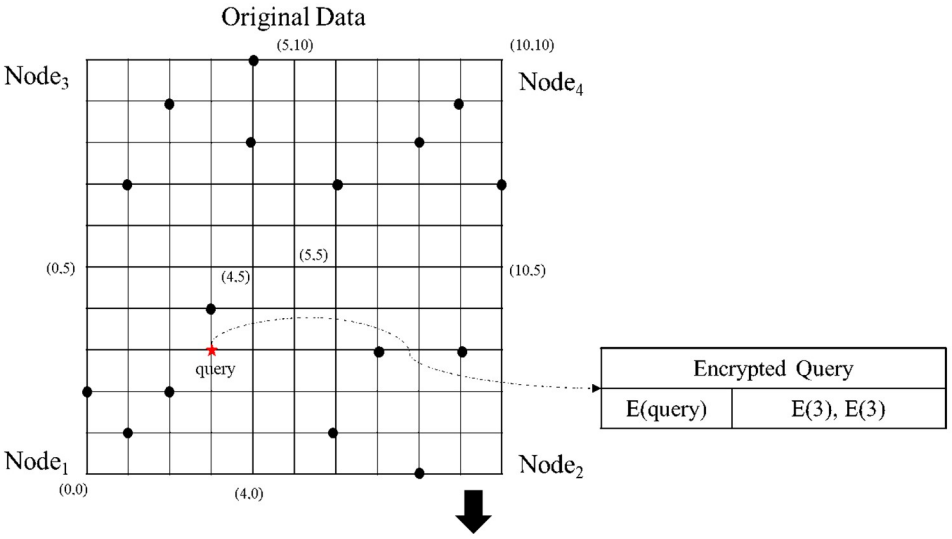
Algorithm 6 Knn classification

```
Input: < E(L_1), ..., E(L_w) >, < E(L'_1), ..., E(L'_w) >
Output: E(L_q)
C_A and C_B:
01: <E(f(L_1)), ..., E(f(L_w)))> = SF(Δ, Δ'), where
Δ =< E(L_1), ..., E(L_w) >, Δ' =< E(L'_1), ..., E(L'_w) >
02: (f(max), E(L_q)) = SXS_w(<E(f(L_1)), ..., E(f(L_w)) >, < E(L_1), ..., E(L_w)
>)
C_A:
03: E(λ_q) = E(c_q) × E(r_q), where r_q ∈ Z_N
04: Send E(λ_q) to C_B and r_q to AU
C_B:
```

05: Receive $E(\lambda_q)$ from C_A
06: $\lambda'_q = D(\lambda_q)$; Send λ'_q to AU
AU:
07: Receive r_q from C_A and λ_q from C_B
08: $c_q = \lambda'_q - r_q$

4.5) Example of kNN classification

Here, an example of the proposed secure kNN classification algorithm is described. Assume that the original data is indexed and encrypted by using the kd-tree, as shown in Fig 4. The encrypted kd-tree contains 4-fold attributes for each leaf node, i.e., a node identifier (ID), an encrypted lower bound of the node, an encrypted upper bound of the node, and the encrypted data. Fig 5 shows how to extract data in a selected node through the secure index search phase. First, C_A sends a node identifier (ID), an encrypted lower bound, an encrypted upper bound, an encrypted query to all the threads. In each thread, the algorithm performs the GSRO protocol to determine whether a node includes the query or not. If a node includes the query, the GSRO protocol returns $E(1)$. Otherwise it returns $E(0)$. Second, the algorithm performs the RSM protocol by multiplying the encrypted data in each node($E(node_z.data)$) and the results of the GSRO protocol. As a result, $E(node_z.data)$ is returned only if the result of the GSRO is $E(1)$. Finally, the algorithm can safely obtain the encrypted data by merging the results of the RSM protocol. Fig 6 shows how to obtain kNN candidates through the k-nearest neighbors search phase. First, the algorithm selects the encrypted data which has the minimum distance



Encrypted kd-tree			
Node ID	Lower Bound	Upper Bound	Data ID
$Node_1$	E(0), E(0)	E(4), E(5)	$E(d_1), E(d_2)\ E(d_3), E(d_4)$
$Node_2$	E(4), E(0)	E(10), E(5)	$E(d_5), E(d_6)\ E(d_7), E(d_8)$
$Node_3$	E(0), E(5)	E(5), E(10)	$E(d_9), E(d_{10})\ E(d_{11}), E(d_{12})$
$Node_4$	E(5), E(5)	E(10), E(10)	$E(d_{13}), E(d_{14})\ E(d_{15}), E(d_{16})$

Fig 4. Data and query used in example.

https://doi.org/10.1371/journal.pone.0267908.g004

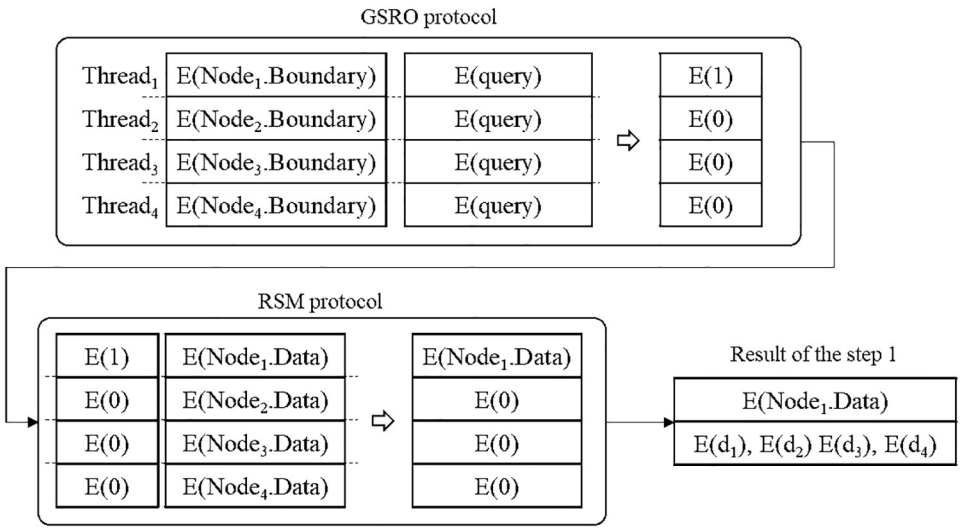


Fig 5. Example of secure index search phase.

https://doi.org/10.1371/journal.pone.0267908.g005

from the query by using the GSMIN$_n$ protocol. In Fig 6, $E(d_3)$ is selected as 1NN because the distance of d_3 is the minimum. Second, the algorithm sets the distance of the selected data to the maximum value for excluding the selected data. Therefore, the distance of $E(d_3)$ is set to E (MAX). Finally, the algorithm is repeated until the k^{th} nearest data is selected. In the same way, $E(d_4)$ and $E(d_2)$ are selected as 2NN and 3NN, respectively. As a result, the algorithm can safely select the k number of nearest neighbors. Figs 7 and 8 show the examples of index search and k-nearest neighbor search in the kNN verification phase, respectively. In each thread, the algorithm calculates the shortest distance $E(spdist_z)$ between the query and a leaf node($node_z$), and compares $E(spdist_z)$ with $E(dist_k)$. If $E(spdist_z)$ is smaller than $E(dist_k)$, the data in the $node_z$ is extracted. In Fig 7, because $E(spdist_2)$, i.e., $(E(1))$, is smaller than $E(dist_k)$, i.e., $(E(5))$, $node_2$ is selected. Fig 8 shows how to obtain the final kNN. The algorithm merges the kNN candidates and obtains the final k-nearest neighbors. In the kNN classification phase, the algorithm calculates the frequency of labels in $E(L')$. Because the frequency of $E(L_1)$ is the highest in kNN, $E(L_1)$ is selected as the final result, as shown in Fig 9.

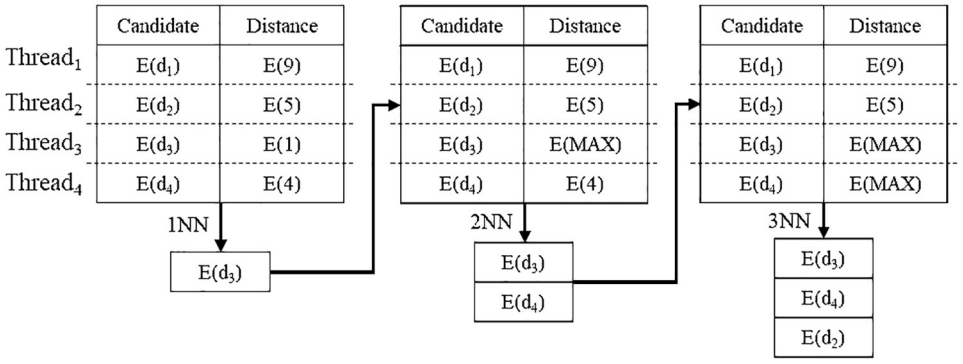


Fig 6. Example of kNN search phase.

https://doi.org/10.1371/journal.pone.0267908.g006

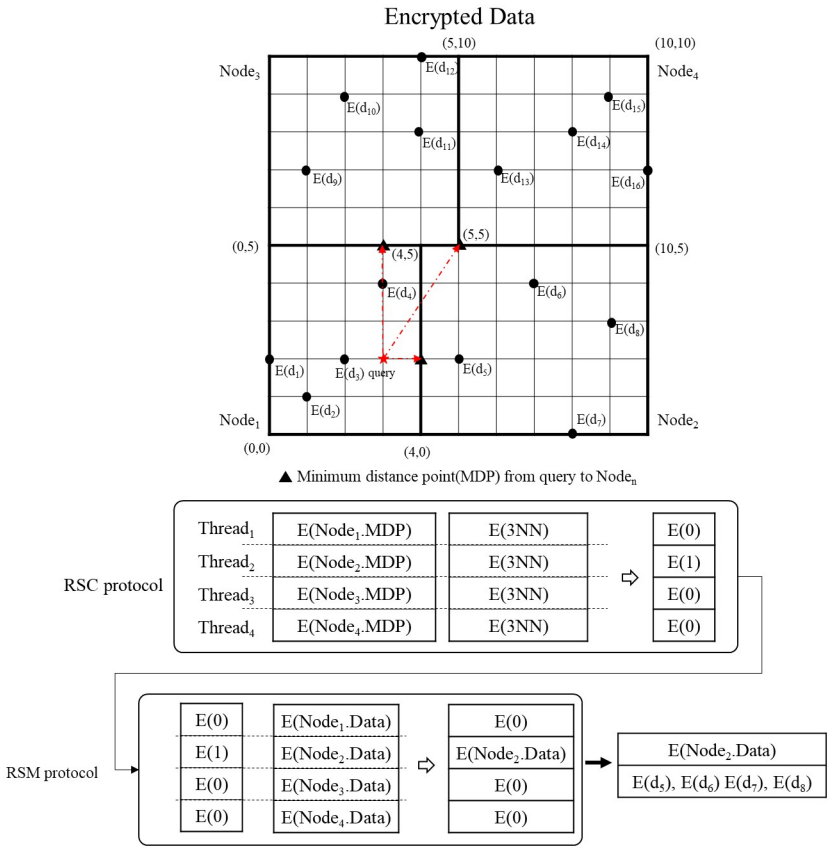


Fig 7. Example of index search in kNN verification phase.

https://doi.org/10.1371/journal.pone.0267908.g007

5 Random value pool's security proof

5.1) Security proof of the secure protocols

In this section, we describe the security proof of the SME and the GSCE protocols proposed in Section 3. To prove that the proposed protocols are secure under the semi-honest model, we show that the simulated images of the proposed protocols are computationally indistinguishable from their actual execution images. Security proof of the SME protocol: We describe the security proof of the SME protocol by analyzing the security of the execution images of C_A and C_B. First, the execution image on C_B side, i.e., $\prod_{C_B}(SME)$, is shown in Eq 8. Here, $E(\nu'_1)$ and $E(\nu'_2)$ are the encrypted data received from C_A (line $1 \sim 2$ of Algorithm 1), ν'_1 and ν'_2 and are obtained through the decryption of $E(\nu'_1)$ and $E(\nu'_2)$, respectively. Also, α is a result which is calculated by the SME protocol using ν'_1 and ν'_2 on C_B side.

$$\prod_{C_B}(SME) = \{< E(\nu'_1), E(\nu'_2), \nu'_1, \nu'_2 >, \alpha\} \tag{8}$$

For example, assume that $\prod C_{B^s}(SME) = \{< E(s'_1), E(s'_2), s'_1, s'_2 >, s_3\}$ is the simulated execution image using the SME protocol on C_B side. Here, $E(s'_1)$ and $E(s'_2)$ are the non-deterministic numbers selected in Z_{N^2}, and s'_1 and s'_2 are the indistinguishable numbers which are added by each value in the random value pool. s'_3 is the result of the SME protocol using s'_1 and s'_2 on C_B side. Because the SME protocol is implemented based on the Paillier cryptosystem, it can

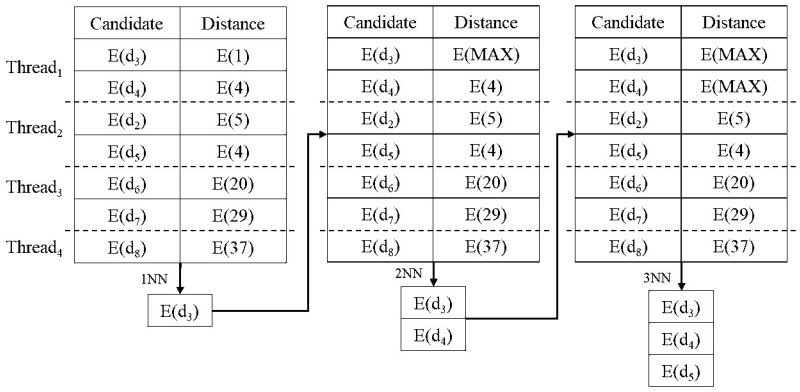


Fig 8. Example of k-nearest neighbor search in kNN verification phase.

https://doi.org/10.1371/journal.pone.0267908.g008

support semantic security. Therefore, $E(s'_1)$ and $E(s'_2)$ are computationally indistinguishable from s'_1 and s'_2, respectively. s'_3 is indistinguishable from s'_1 and s'_2 because s'_3 is calculated by multiplying two indistinguishable numbers in C_A, s'_1 and s'_2. Therefore, it can be said that $\prod_{C_B}(SME)$ is computationally indistinguishable from $\prod_{C_{BS}}(SME)$. Because C_B can check only the result (e.g., α) of the multiplication between the non-deterministic numbers (e.g., v'_1 and v'_2), C_B cannot obtain the original data while performing the SME protocol. Meanwhile, the execution image of C_A is $\prod_{C_A}(SME) = \{E(\alpha)\}$ such that *$E(\alpha)$ from* C_B *can be regarded as the result of the SME protocol. Suppose that the simulated image of* C_A *is* $\prod_{C_{AS}}(SME) = \{E(s_4)\}$, *where* $E(s_4)$ *is randomly generated from* Z_{N^2}. Therefore, $E(\alpha)$ is computationally indistinguishable from $E(s_4)$. According to the above analyses, there is no information leakage both at C_A and C_B side. Therefore, we can conclude that the proposed SME protocol is secure under the semi-honest adversarial model. Security proof of the GSCE protocol: We describe the security proof of the GSCE protocol by analyzing the security of the execution images of C_A side and C_B side. First, the execution image on C_B side, i.e., $\prod_{C_B}(GSCE)$, is shown in Eq 9. Here, $E(\sigma'_1)$ and $E(\sigma'_2)$ refer to the encrypted data received from C_A (line $1 \sim 2$ of Algorithm 2), and both σ'_1 and σ'_2 are obtained through decryption of σ'_1 and σ'_2, respectively. Also, *β is the result which is calculated by the GSCE protocol using* σ'_1 *and* σ'_2 *on* C_B *side.*

$$\prod_{C_B}(GSCE) = \{< E(\sigma'_1), E(\sigma'_2), \sigma'_1, \sigma'_2 >, \beta\} \tag{9}$$

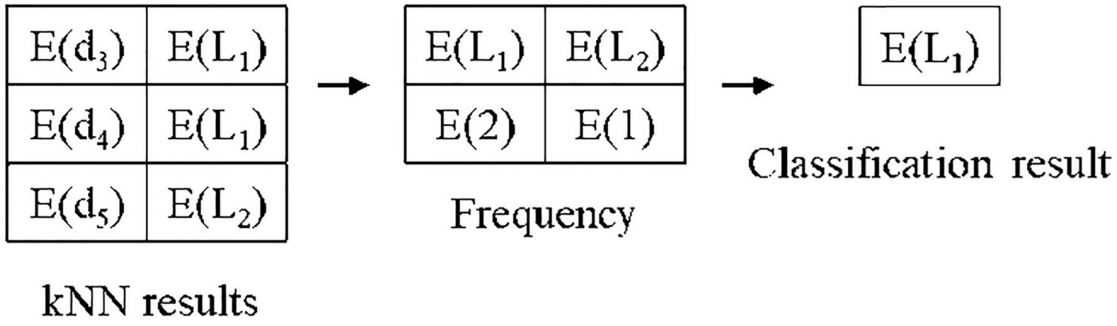


Fig 9. Example of kNN classification phase.

https://doi.org/10.1371/journal.pone.0267908.g009

For example, assume that $C_{B^S}(GSCE) = \{< E(s'_1), E(s'_2), s'_1, s'_2 >, s_3\}$ for the simulated execution image using the GSCE protocol on C_B side. Here, $E(s'_1)$ and $E(s'_2)$ are the non-deterministic numbers selected in Z_{N^2}, and both s'_1 and s'_2 are the indistinguishable numbers selected in the random value pool. s'_3 is the result of the GSCE protocol using s'_1 and s'_2 on C_B side. Because the GSCE protocol is implemented based on the Paillier cryptosystem, it can support semantic security. Therefore, $E(s'_1)$ and $E(s'_2)$ are computationally indistinguishable from s'_1 and s'_2, respectively. s'_3 is indistinguishable from s'_1 and s'_2 because s'_3 is calculated by comparing two indistinguishable numbers in C_A, s'_1 and s'_2. Therefore, it can be said that $\prod_{C_B}(GSCE)$ is computationally indistinguishable from $\prod_{C_{B^S}}(GSCE)$. Because C_B can check only the result (e.g., β) of the comparison between the non-deterministic numbers (e.g., σ'_1 and σ'_2), C_B cannot obtain the original data while performing the GSCE protocol. Meanwhile, the execution image of C_A is $\prod_{C_A}(GSCE) = E(\beta)$ such that *$E(\beta)$ from* C_B *can be regarded as the result of the GSCE protocol. Suppose that the simulated image of* C_A *is* $\prod_{C_A^S}(GSCE) = E(s_4)$*, where* $E(s_4)$ *is randomly generated from* Z_{N^2}. Therefore, $E(\beta)$ is computationally indistinguishable from $E(s_4)$. According to the above analyses, there is no information leakage both at C_A and C_B side. Therefore, we can conclude that the proposed GSCE protocol is secure under the semi-honest adversarial model

5.2) Security proof of the proposed kNN classification algorithm

We prove that the proposed kNN classification algorithm on the encrypted database is safe under the semi-honest attack model. The proposed kNN classification algorithm in the cryptographic database consists of a secure index search phase (Algorithm 3), a kNN search phase (Algorithm 4), a kNN verification phase (Algorithm 5), and a kNN classification phase (Algorithm 6). To show that the proposed secure kNN classification algorithm is safe under the semi-honest attack model, security analysis is performed at each execution phase. First, because the secure index search phase is composed of the GSRO protocol [17] which has been proven to be safe, the Algorithm 3 is safe under the semi-honest attack model by composition theory [17]. Second, the kNN search phase is safe in C_A side, because C_A performs the ESSED, $SMIN_n$ and SM protocols which have been proven to be safe in the previous studies [16, 17]. Even though the kNN search phase decrypts the received data from C_A, C_B cannot extract the original data. This is because the data received from C_A is modified by raising the original data to the power of a random integer and applying a shuffling function. Therefore, according to the composition theory, Algorithm 4 is safe under the semi-honest attack model. Third, the images which are generated by the kNN verification phase are the same as those generated by Algorithms 3 and 4. Therefore the kNN verification phase (Algorithm 5) is safe under the semi-honest attack model. Lastly, the kNN classification phase (Algorithm 6) is safe under the semi-honest attack model because Algorithm 6 has been proven safe in the previous work [16, 17]. As a result, all the phases of the proposed secure kNN classification algorithm is safe under the semi-honest attack model.

6 Performance analysis

Because there is no privacy-preserving parallel kNN Classification algorithm, we compare our privacy-preserving parallel kNN classification algorithm with the extension of existing works. That is, we make parallel SkNNC-M by extending B. K. Samanthula et. al.'s work [16] in a naive way so that it may operate in a multi-core environment. We make parallel SkNNC-G by extending H. J. KIM et. al.'s work [17] in the same way. For performance evaluation, three algorithms were implemented by using C++ under an Intel(R) Xeon(R) CPU E5–2630 v4 @

Table 4. Parameters used in performance evaluation for synthetic data.

Parameter	Values	default
the number of data(n)	5k,10k,20k,30k	10k
k	5, 10, 15, 20	10
level of kd-tree(h)	5, 6, 7, 8, 9	7
the number of threads	1, 2, 5, 10	10
the number of data dimension(m)	3, 6, 9, 12	6
Size of encryption key(K)	512	-
bit size for data domain	12	-

https://doi.org/10.1371/journal.pone.0267908.t004

2.20GHz and 64GB (16GB × 4AE) DDR3 UDIMM 1600MHz in a Linux Ubuntu 18.04.2 environment. We compare three parallel algorithms in terms of the query processing time by varying the number of data, the number of k, the level of the kd-tree, the number of the data dimension, and the number of threads. We use both a synthetic dataset and real dataset [28] for our experiments.

6.1) Performance analysis of kNN classification algorithm for synthetic dataset

Table 4 shows the parameters used in the performance evaluation for the synthetic dataset. For the synthetic dataset, we randomly generate 30,000 integer data with 12 dimensions. The domain of data is ranged from 0 to 212. We do an experiment to find the optimal value of the level of kd-tree(h). It is shown that the performances of both SkNNC-G and the proposed algorithm are best when h is 7. So, we set h to 7 in our experiment.

The performance of the kNN classification algorithms is evaluated for synthetic data. Fig 10 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G

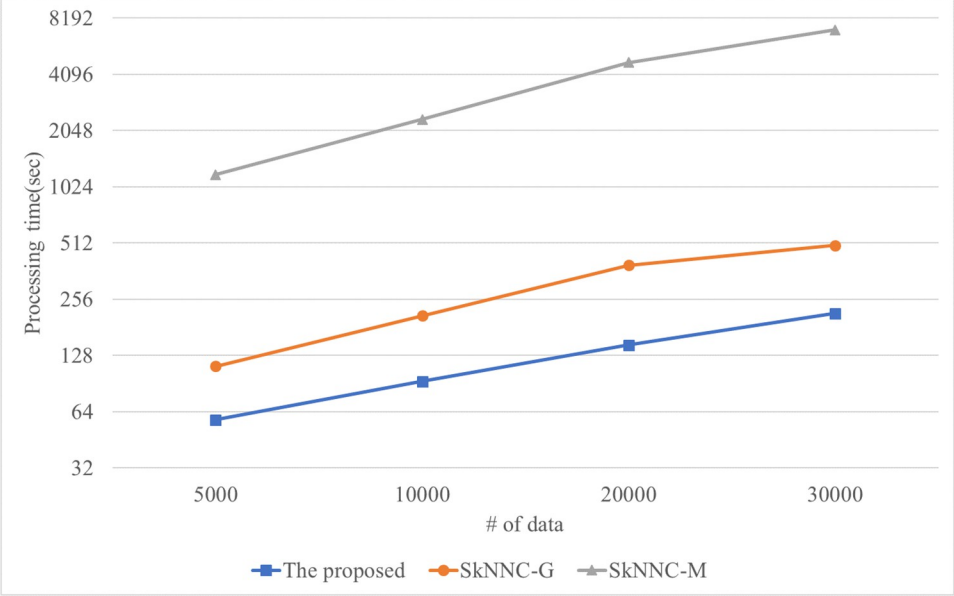


Fig 10. Processing time with varying the number of data.

https://doi.org/10.1371/journal.pone.0267908.g010

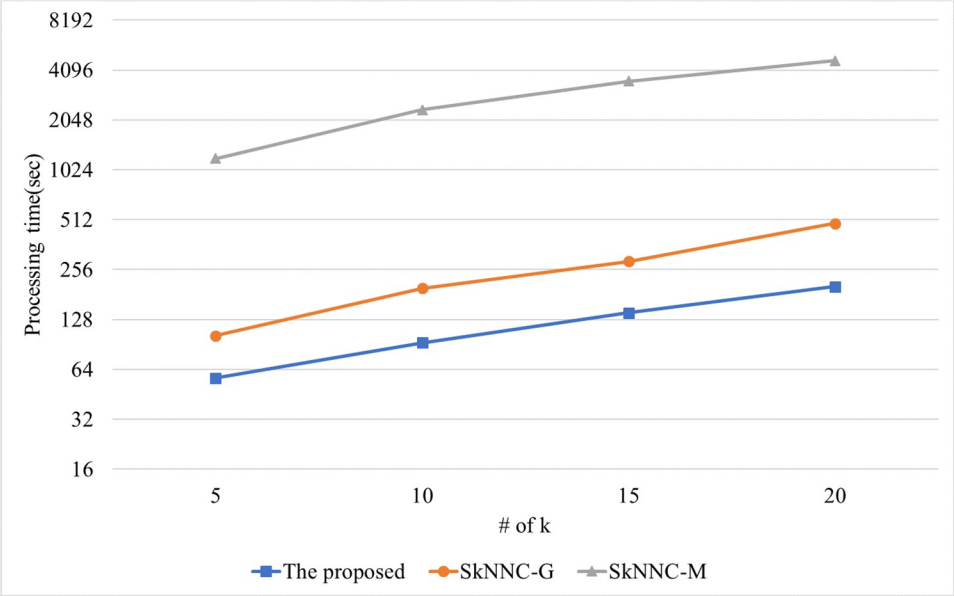


Fig 11. Processing time with varying k.

https://doi.org/10.1371/journal.pone.0267908.g011

according to the number of data. When n = 30k, the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M require 215, 497, and 7,089 seconds, respectively. That is, the proposed algorithm shows 2.3 times better performance than parallel SkNNC-G and 32 times better performance than parallel SkNNC-M. This is because our secure protocols (SME and GSCE protocols) can reduce the number of data encryptions by selecting an encrypted value from the random value pool instead of generating it, as mentioned in Table 3. Fig 11 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G according to k. When k = 20, the proposed parallel algorithm, parallel SkNNC-G, and parallel SkNNC-M require 202, 487, and 4,658 seconds, respectively. That is, the proposed algorithm shows 2.4 times better performance than parallel SkNNC-G and 23 times better performance than parallel SkNNC-M. The reason is the same as mentioned in Fig 10.

Fig 12 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G according to the number of data dimension(m). When m = 6, the proposed parallel algorithm, parallel SkNNC-G, and parallel SkNNC-M require 57, 112, and 2,353 seconds, respectively. That is, the proposed algorithm shows 2 times better performance than parallel SkNNC-G and 15 times better performance than parallel SkNNC-M. The reason is the same as mentioned in Fig 10. Fig 13 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G according to the number of threads. When the number of threads = 1(single-core), the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M require 443, 894, and 15,572 seconds, respectively. That is, the proposed algorithm shows 2 times better performance than parallel SkNNC-G and 35 times better performance than parallel SkNNC-M. This is because our secure protocols (SME and GSCE protocols) can reduce the number of data encryptions by selecting an encrypted value from the random value pool instead of generating it. When the number of threads = 10, the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M require 93, 203, and 2350 seconds, respectively. That is, the proposed algorithm shows 2.1 times better performance than parallel SkNNC-G and 25 times

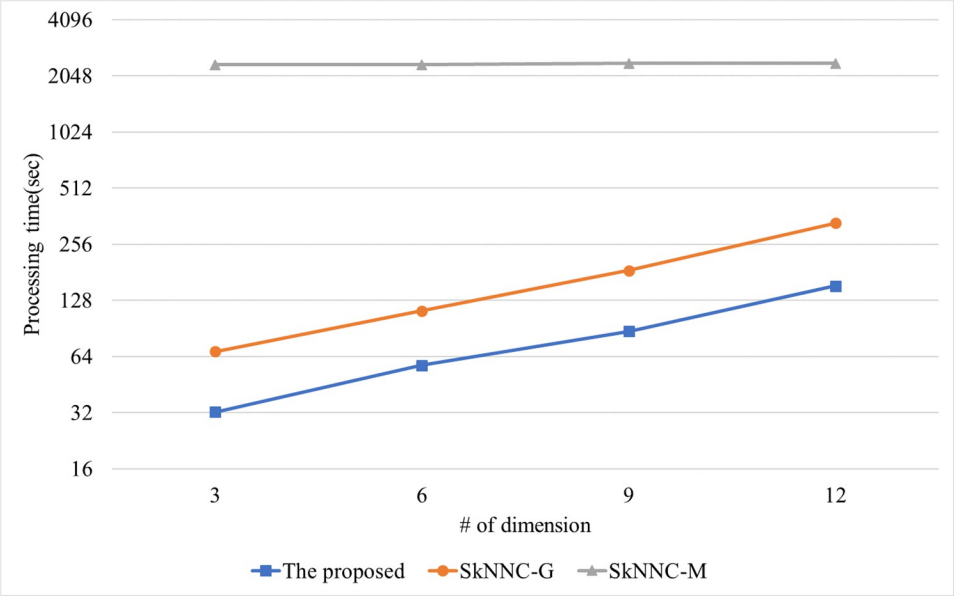


Fig 12. Processing time with varying the number of dimensions.

https://doi.org/10.1371/journal.pone.0267908.g012

better performance than parallel SkNNC-M. Because a thread performs secure protocols concurrently without interfering with each other, query processing time linearly decreases as the number of threads increases. As a result, our parallel algorithm shows better performance than the existing algorithms in a multi-core environment.

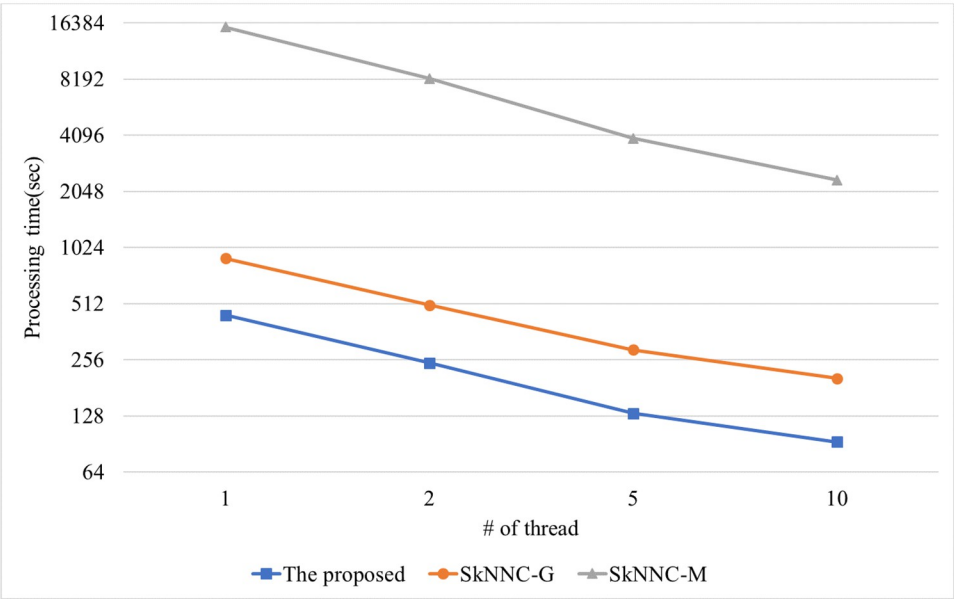


Fig 13. Processing time with varying the number of threads.

https://doi.org/10.1371/journal.pone.0267908.g013

Table 5. Parameters used in performance evaluation for real data.

Parameter	Values	default
the number of data(n)	28056	-
k	5, 10, 15, 20	10
level of kd-tree(h)	7	-
the number of threads	1, 2, 5, 10	10
the number of data dimension(m)	6	-
Size of encryption key(K)	512	-
bit size for data domain	12	-

https://doi.org/10.1371/journal.pone.0267908.t005

6.2) Performance analysis of kNN classification algorithm for real dataset

Table 5 shows the parameters used in the performance evaluation for real data. For this, we used a chess dataset [28] generated by a chess endgame database for white king and rook against black king. The chess dataset aims to classify the optimal depth of win for white. With the real dataset, we do an experiment to find the optimal value of the level of kd-tree(h). It is shown that the performances of both SkNNC-G and the proposed algorithm are best when h is 7. So, we set h to 7 in our experiment.

Fig 14 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G according to k. When k = 20, the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M require 425, 894, and 13,175 seconds, respectively. That is, the proposed algorithm shows 2 times better performance than parallel SkNNC-G and 27 times better performance than parallel SkNNC-M. This is because our algorithm uses both SME and GSCE protocols which can reduce the number of data encryptions by selecting an encrypted value from the random value pool. Fig 15 shows the performance of the proposed algorithm, parallel SkNNC-M, and parallel SkNNC-G according to the number of threads. When the number of threads = 1 (single-core), the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M

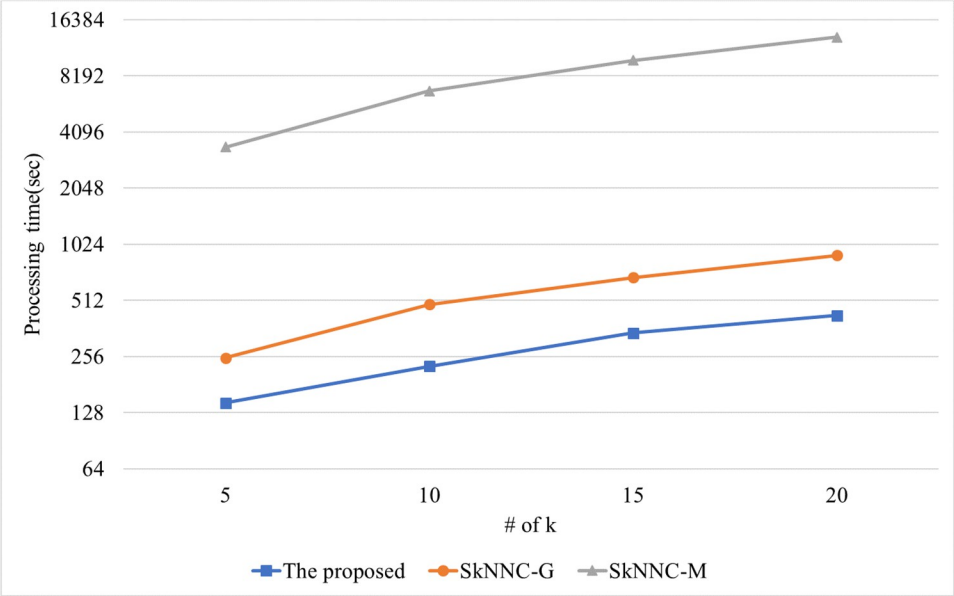


Fig 14. Processing time with varying k.

https://doi.org/10.1371/journal.pone.0267908.g014

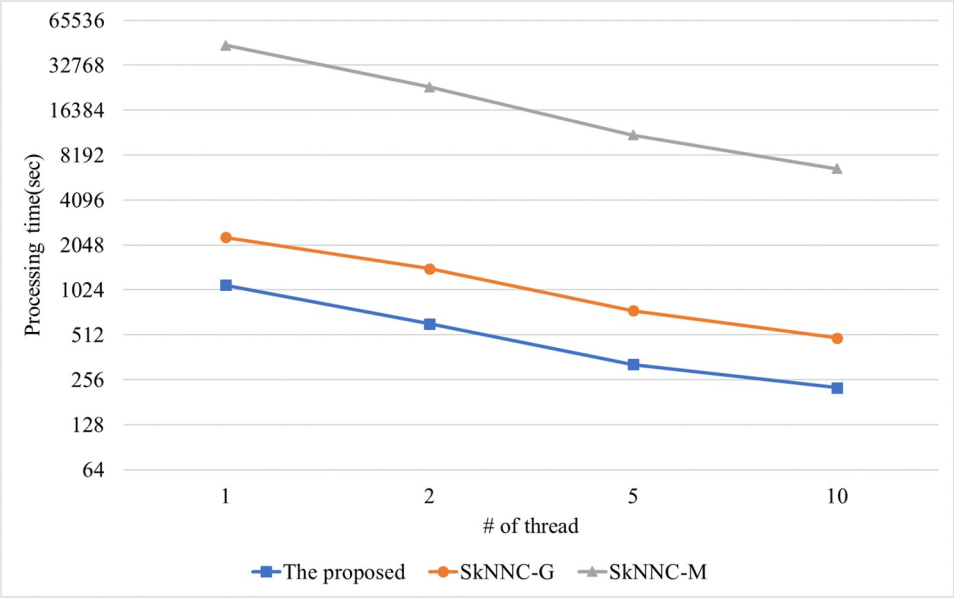


Fig 15. Processing time with varying the number of threads.

https://doi.org/10.1371/journal.pone.0267908.g015

require 1106, 2306, and 44,570 seconds, respectively. That is, the proposed algorithm shows 2 times better performance than parallel SkNNC-G and 40 times better performance than parallel SkNNC-M. The reason is the same as mentioned in Fig 14. When the number of threads = 10, the proposed algorithm, parallel SkNNC-G, and parallel SkNNC-M require 227, 487, and 6,639 seconds, respectively. That is, the proposed algorithm shows 2 times better performance than parallel SkNNC-G and 29 times better performance than parallel SkNNC-M. Because a thread performs secure protocols concurrently without any interference of each other, it can be seen that query processing time linearly decreases as the number of threads increases.

6.3) Theoretical analysis of the proposed algorithm in terms of privacy

Assuming that an attacker does not have any information of original data items, an adversary needs tremendous time to obtain the original plaintext from paillier cryptosystem while using a brute force attack. It means that it is impossible to do an experiment to prove data protection, query protection and access pattern protection. Therefore, instead of experimental analysis, we conduct the theoretical analysis of data privacy, query privacy and access pattern privacy to support the security analysis of the proposed algorithm. For this, we estimate the time complexity it takes for the original data to be exposed and calculate the probability of access pattern leakage.

6.3.1 Theoretical analysis of data privacy. In C_A, an attacker only obtains the ciphertext of data. Because the data is protected by the paillier cryptosystem, the security performance is measured through the time complexity of the brute force attack to break down the paillier cryptosystem. Our paillier cryptosystem uses 512-bit encryption key size. Assuming that CPU cycle is 4GHz, the time required to decrypt the ciphertext by changing the key is as shown in Eq (10).

$$BFAtime(sec) = \frac{2^{512}}{4GHz} \approx \frac{1.3 \times 10^{154}}{4GHz} \tag{10}$$

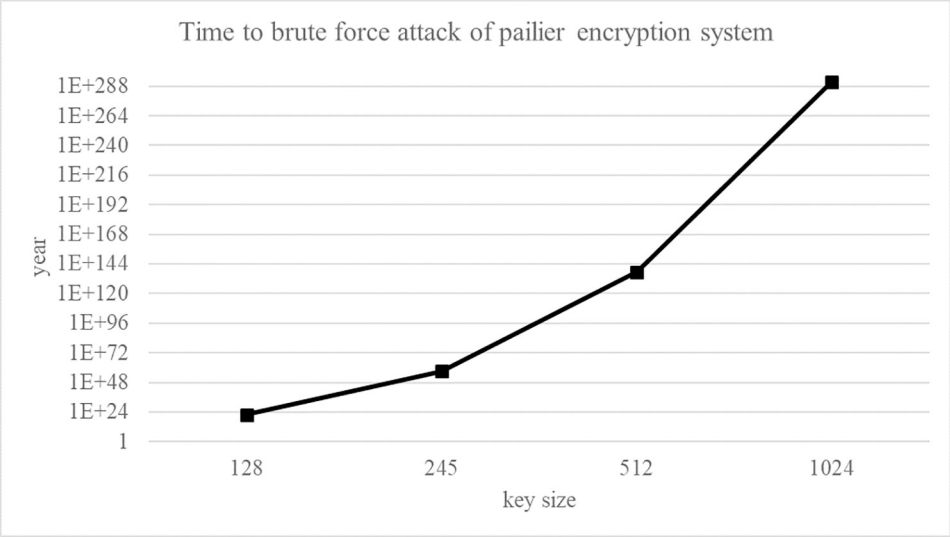


Fig 16. Brute force attack time with varying the key size.

https://doi.org/10.1371/journal.pone.0267908.g016

It is impossible to break down a paillier cryptosystem because it takes about 4.2×10^{146} years with 512-bit key size. It means that the proposed privacy preserving kNN classification algorithm is secure in terms of data privacy even if the ciphertext is exposed. Fig 16 shows the time taken for a brute force attack in C_A as the key size is changed. In C_B, an attacker only obtains a plaintext data which adds a random number to the original data. In the paillier cryptosystem, because the range of the plaintext data is $0 \leq m \leq 2^{512}$, brute force attack time in C_B has the same as that in C_A.

6.3.2 Theoretical analysis of query privacy. In C_A, an attacker only obtains the ciphertext of query. Because the query is protected by the paillier cryptosystem, the security performance is measured through the time complexity of the brute force attack to break down the paillier cryptosystem. Since our paillier cryptosystem uses 512-bit encryption key size, the time required to decrypt the ciphertext by changing the key is as shown in 10, where CPU cycle is 4GHz. It is impossible to break down a paillier cryptosystem because it takes about 4.2×10^{146} years with 512-bit key size. It means that the proposed privacy preserving kNN classification algorithm is secure in terms of query privacy even if the ciphertext is exposed. The times taken for a brute force attack in C_A is the same as that of data privacy in C_A (Fig 16). In C_B, query privacy is preserved because C_B does not receive the query.

6.3.3 Theoretical analysis of access pattern privacy. The access pattern means the sequence of accessing a data item. In the proposed algorithm, the sequence of accessing a data item consists of the leaf node access of kd-tree and data access in the leaf node. In C_A, an attacker only obtains the ciphertext of leaf node. Because all the leaf nodes have the same number of data items, an attacker cannot distinguish the leaf node by using density of data items. If the kd-tree level is h, the number of leaf node is 2^{h-1}. The probability that an attacker can distinguish a node($node_i$) from the others, i.e., $P(node_i)$, is $\frac{1}{2^{h-1}}$. Because $node_i$ includes the same number of data items as fanout, the probability that an attacker can distinguish a data item from the others in $node_i$, i.e., $P(node_i.data_j)$, is $\frac{1}{fanout} = \frac{1}{\frac{the\ \ number\ \ of\ \ data}{2^{h-1}}} = \frac{2^{h-1}}{n}$. Therefore, the

probability of data access pattern leakage (P_{APL}) is shown in Eq (11).

$$P_{APL} = P(node_i) \times P\left(node_i.data_j\right) = \frac{1}{2^{h-1}} \times \frac{2^{h-1}}{n} = \frac{1}{n} \tag{11}$$

P_{APL} is equal to the probability that an attacker distinguishes a specific data item from the others in the entire data items. Therefore, the proposed algorithm can preserve the access pattern privacy in C_A. In C_B, access pattern privacy is preserved because C_B does not have any data item.

7 Discussion

7.1 Impact of hiding data access patterns

The data access pattern is one of the most important factors for privacy preservation. If an attacker possesses the order or the frequency of data, he/she can infer the original data by using data access patterns. Therefore, hiding data access patterns is as important as encrypting data. First, B. Yao et al.'s work [21] proposed a secure kNN classification algorithm using the Voronoi diagram [22]. However, the order of accessing the Voronoi diagram is distinguishable and an attacker can partially infer the original data from the query. Second, J. Du and F. Bian's work [25] proposed a kNN classification algorithm using an order-preserving index. However, the index access patterns are exposed because the order of accessing the index can be easily obtained from the query. This allows an attacker to easily infer the original data if he/she has an index access pattern. Meanwhile, our algorithm uses the Paillier cryptosystem which supports semantic security for data protection. As a result, all of the ciphertext is indistinguishable and secure from frequency-based attacks. In addition, the kd-tree filtering technique used in our algorithm is secure from the exposure of data access patterns because our algorithm accesses only the encrypted leaf nodes of the kd-tree without accessing the index by using a top-down approach. Therefore, our algorithm can hide the data access patterns.

7.2 Impact of parallel algorithm with garbled circuit

First, a garbled circuit is used for efficient processing of secure protocols. B. K. Samanthual et al.'s work [16] has high overhead by using a secure protocol based on the comparison of binary array. To overcome this problem, our secure protocols use a garbled circuit that performs a fast and secure comparison operation in the state of the ciphertext. Second, the existing algorithms do not use parallelism for the privacy-preserving classification algorithm [16, 17, 25]. On the contrary, our algorithm proposes a parallel classification algorithm adopting the garbled circuit. Our algorithm performs three phases in parallel: index searching, kNN searching and kNN verification. As shown in our performance evaluation, our parallel classification algorithm shows performance improvement in proportion to the number of threads.

7.3 Impact of encrypted random value pool

In our secure system, we use two-party computation for the parallel kNN classification algorithm. Thus, we need to prevent C_B from extracting meaningful information while executing secure protocols. For this, C_A generates a random value r from Z_N and encrypts r by using the Paillier cryptosystem. Then, C_A adds the encrypted random value $E(r)$ to the encrypted plaintext $E(m)$ by computing $E(m + r) = E(m) \times E(r)$. Because $m \pm r$ is independent from m, C_B cannot obtain meaningful information with decryption. However, adding a random value to the ciphertext in the Paillier cryptosystem leads to performance degradation because both encryption and decryption operations require higher computation cost than other encrypted

operations. In the Secure Multiplication protocol, both B. K. Samanthula et al.'s work and H. Kim et al.'s work require three times of the encryption: 2 encryptions for random values at C_A and 1 encryption for the result of multiplication at C_B. Meanwhile, our algorithm requires only one encryption for the result of multiplication at C_B because it selects the encrypted random values from the random value pool without encrypting the random values at C_A. In the Secure Compare protocol, B. K. Samanthula et al.'s work requires $log_2 D$ times of encryption where D is a data domain. H. Kim et al.'s work requires three times of the encryption: 2 encryptions for random values at C_A and 1 encryption for the result of the comparison between two values at C_B. Meanwhile, our algorithm requires only one encryption for the result of comparison at C_B by using the random value pool. Therefore, our algorithm can reduce the amount of computation cost for encryption by using the encrypted random value pool.

7.4 Practical example of proposed kNN classification

The proposed secure kNN classification algorithm can be used in various fields. For example, first, it can be used to diagnose a disease by classifying the patterns of the patient's symptoms [29]. Because the existing disease diagnosis system depends on only the doctor's knowledge and experience, it may cause damage to patients due to misdiagnosis. Therefore, kNN classification algorithms can help doctors classify the pattern of the patient's symptoms so as to diagnose what kind of disease it is. However, because patients' information contains sensitive data, such as past medical history, family history and allergies, the proposed privacy-preserving kNN classification algorithm can be used to protect the sensitive data of patients. Second, the proposed privacy-preserving kNN classification algorithm can be used to solve the problem of insurance coverage recommendation where insurance companies provide the most suitable coverage for customers [30]. The insurance coverage recommendation classifies customers' grades based on various customers' information, such as movement patterns and lifestyles. To perform the classification of customers' grades, the proposed privacy-preserving kNN classification algorithm can be used to protect the personal information of customers.

8 Conclusion

In this paper, we proposed a parallel kNN classification algorithm over encrypted data to preserve data privacy, query privacy, and access pattern privacy in cloud computing. To reduce the computation cost for encryption, we proposed two secure protocols, SME and GSCE, which support secure multi-party computation by using an encrypted random value pool. To reduce the query processing time, we not only designed a parallel algorithm, but also adopted a garbled circuit. In addition, we proved that our algorithm over the encrypted database is safe under the semi-honest attack model. Through our performance evaluation, our algorithm showed about $2 \sim 25$ times better performance compared with the existing algorithms. For future work, we plan to apply our parallel query processing algorithm to secure k-Means clustering.

Supporting information

S1 File.
(BST)

Author Contributions

Conceptualization: Yong-Ki Kim, Hyeong-Jin Kim, Hyunjo Lee, Jae-Woo Chang.

Project administration: Jae-Woo Chang.

Resources: Yong-Ki Kim.

Software: Hyeong-Jin Kim.

Supervision: Jae-Woo Chang.

Writing – original draft: Yong-Ki Kim, Hyeong-Jin Kim, Hyunjo Lee.

Writing – review & editing: Yong-Ki Kim, Hyeong-Jin Kim, Hyunjo Lee, Jae-Woo Chang.

References

1. Ge YF, Yu WJ, Cao J, Wang H, Zhan ZH, Zhang Y, et al. Distributed memetic algorithm for outsourced database fragmentation. IEEE Transactions on Cybernetics. 2020; 51(10):4808–4821. https://doi.org/10.1109/TCYB.2020.3027962
2. Ge YF, Orlowska M, Cao J, Wang H, Zhang Y MDDE: multitasking distributed differential evolution for privacy-preserving database fragmentation. The VLDB Journal. 2022;1–19.
3. Brian H, Brunschwiler T, Dill H, Christ H, Falsafi B, Fischer M, et al. Cloud computing. Communications of the ACM. 2008 July; 51(7):9–11. https://doi.org/10.1145/1364782.1364786
4. Josep AD, Katz R, Konwinski A, Gunho LEE, Patterson D, Rabkin A. A view of cloud computing. Communications of the ACM. 2010; 53(4):50–58. https://doi.org/10.1145/1721654.1721672
5. Xiong L, Chitti S, Liu L. Topk queries across multiple private databases. In 25th IEEE International Conference on Distributed Computing Systems. 2005 June;145–154.
6. Gutscher A. Coordinate transformation-a solution for the privacy problem of location based services?. In Proceedings 20th IEEE International Parallel Distributed Processing Symposium. 2006 April;7.
7. Hassanat AB. Two-point-based binary search trees for accelerating big data classification using KNN. PloS one. 2018; 13(11):e0207772. https://doi.org/10.1371/journal.pone.0207772 PMID: 30475862
8. Wong WK, Cheung DWL, Kao B, Mamoulis N. Secure knn computation on encrypted databases. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. 2009;139–152.
9. Hu H, Xu J, Ren C, Choi B. Processing private queries over untrusted data cloud through privacy homomorphism. In 2011 IEEE 27th International Conference on Data Engineering. 2011;601–612.
10. Wang B, Hou Y, Li M, Wang H, Li H. Maple: scalable multi-dimensional range search over encrypted cloud data with tree-based index. In Proceedings of the 9th ACM symposium on Information, computer and communications security. 2014 June;111–122.
11. Elmehdwi Y, Samanthula BK, Jiang W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In 2014 IEEE 30th International Conference on Data Engineering. 2014 March;664–675.
12. Jiang ZL, Guo N, Jin Y, Lv J, Wu Y, Liu Z, et al. Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. Information Sciences. 2020; 518:168–180. https://doi.org/10.1016/j.ins.2019.12.051
13. Alabdulatif A, Khalil I, Yi X. Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption. Journal of Parallel and Distributed Computing, 2020; 137:192–204. https://doi.org/10.1016/j.jpdc.2019.10.008
14. Pang H, Wang B. Privacy-preserving association rule mining using homomorphic encryption in a multikey environment. IEEE Systems Journal. 2020; 15(2):3131–3141. https://doi.org/10.1109/JSYST.2020.3001316
15. Wu W, Liu J, Wang H, Hao J, Xian M. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique. IEEE Transactions on Knowledge and Data Engineering. 2020; 33(10):3424–3437. https://doi.org/10.1109/TKDE.2020.2969633
16. Samanthula BK, Elmehdwi Y, Jiang W. K-nearest neighbor classification over semantically secure encrypted relational data. IEEE transactions on Knowledge and data engineering. 2014; 27(5):1261–1273. https://doi.org/10.1109/TKDE.2014.2364027
17. Kim HJ, Kim HI, Chang JW. A Privacy-Preserving kNN Classification Algorithm Using Yao's Garbled Circuit on Cloud Computing. In 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). 2017 June;766–769.
18. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In International conference on the theory and applications of cryptographic techniques. 1999 May; 223–238.

19. Goldwasser S, Micali S. Probabilistic encryption & how to play mental poker keeping secret all partial information. In Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali. 2019; 173–201.

20. Brickell J, Shmatikov V. Privacy-preserving graph algorithms in the semi-honest model. In International Conference on the Theory and Application of Cryptology and Information Security. 2005 December;236–252.

21. Yao B, Li F, Xiao X. Secure nearest neighbor revisited. In 2013 IEEE 29th international conference on data engineering (ICDE). 2013 April; 733–744.

22. Erwig M. The graph Voronoi diagram with applications. Networks: An International Journal. 2000; 36 (3):156–163. https://doi.org/10.1002/1097-0037(200010)36:3%3C156::AID-NET2%3E3.0.CO;2-L

23. Wu W, Liu J, Rong H, Wang H, Xian M. Efficient k-nearest neighbor classification over semantically secure hybrid encrypted cloud database. IEEE Access. 2018; 6:41771–41784. https://doi.org/10.1109/ACCESS.2018.2859758

24. Tan Y, Wu W, Liu J, Wang H, Xian M. Lightweight edge based kNN privacy preserving classification scheme in cloud computing circumstance. Concurrency and Computation: Practice and Experience. 2020; 32(19). https://doi.org/10.1002/cpe.5804

25. Du J, Bian F. A Privacy-Preserving and Efficient k-nearest neighbor query and classification scheme based on k-dimensional tree for outsourced data. IEEE Access. 2020; 8:69333–69345. https://doi.org/10.1109/ACCESS.2020.2986245

26. Boldyreva A, Chenette N, Lee Y, O'neill A. Order-preserving symmetric encryption. In Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2009 April; 224–241.

27. Yao ACC. How to generate and exchange secrets. In 27th Annual Symposium on Foundations of Computer Science. 1986 October;162–167.

28. Michael B. Chess (King-Rook vs. King) Data Set. The UCI KDD Archive. 1994 June; http://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King%29.

29. Hashi EK, Zaman MSU, Hasan MR. An expert clinical decision support system to predict disease using classification techniques. In 2017 International conference on electrical, computer and communication engineering (ECCE). 2017 February;396–400.

30. Khalili-Damghani K, Abdi F, Abolmakarem S. Solving customer insurance coverage recommendation problem using a two-stage clustering-classification model. International Journal of Management Science and Engineering Management. 2019; 14(1):9–19. https://doi.org/10.1080/17509653.2018.1467801