IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 September 2023, accepted 27 September 2023, date of publication 2 October 2023,
date of current version 10 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3321457

RESEARCH ARTICLE

Secure Cloud-Aided Approximate Nearest Neighbor Search on High-Dimensional Data

JIA LIU[1,2], WANG YINCHAI[2], FENGRUI WEI[1], QING HAN[4], YUNTING TAO[4], LIPING ZHAO[1], XINJIN LI[1], AND HONGBO SUN[1,3]

[1]Faculty of Big Data, Weifang Institute of Technology, Weifang 262500, China
[2]Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Sarawak 89007, Malaysia
[3]School of Computer and Control Engineering, Yantai University, Yantai 264005, China
[4]Network and Information Security Laboratory, Shandong University, Jinan 250100, China

Corresponding author: Jia Liu (lsegily@126.com)

This work was supported in part by the China Industry-University-Research Innovation Fund of The Science and Technology Development Center of the Ministry of Education–Collaborative Innovation Project "Design and Research of 'Intelligent' Education Platform for College Students Based on Big Data and Artificial Intelligence Technology" under Grant 2020QT18; in part by the Weifang Science and Technology Development Plan "Adaptive Navigation Technology of Inspection Robot Based on Machine Vision" under Grant 2022GX081; in part by the Research Foundation of Weifang Institute of Technology "Research on Nearest Neighbor Query Hash Algorithm for High Dimensional Multi-Party Data Based on Federated Learning" under Grant KJ-X202203; in part by the Shandong Higher Education Association "Higher Education research" Special Project "Research on Training Mode of Data Science and Big Data Technology Professionals from The Perspective of 'New Engineering;'" in part by the Weifang Institute of Technology College Students Ideological and Political Education Special Project "Research on Network Public Opinion Propagation path and Influencing Factors of University Public Events" under Grant SK-Z202301; in part by the Network Security Project of Shanghai Network Science, China Universities Innovation Fund "Research on Deep Learning Algorithm in Network Attacks and Its Application in Intrusion Detection;" and in part by the Shandong Provincial Teachers' Visiting Research and Training Funds.

ABSTRACT As one fundamental data-mining problem, ANN (approximate nearest neighbor search) is widely used in many industries including computer vision, information retrieval, and recommendation system. *LSH* (Local sensitive hashing) is one of the most popular hash-based approaches to solve ANN problems. However, the efficiency of operating *LSH* needs to be improved, as the operations of *LSH* often involve resource-consuming matrix operations and high-dimensional large-scale datasets. Meanwhile, for resource-constrained devices, this problem becomes more serious. One way to handle this problem is to outsource the heavy computing of high-dimensional large-scale data to cloud servers. However, when a cloud server responsible for computing tasks is untrustworthy, some security issues may arise. In this study, we proposed a cloud server-aided *LSH* scheme and the application model. This scheme can perform the *LSH* efficiently with the help of a cloud server and guarantee the privacy of the client's information. And, in order to identify the improper behavior of the cloud server, we also provide a verification method to check the results returned from the cloud server. Meanwhile, for the implementation of this scheme on resource-constrained devices, we proposed a model for the real application of this scheme. To verify the efficiency and correctness of the proposed scheme, theoretical analysis and experiments are conducted. The results of experiments and theoretical analysis indicate that the proposed scheme is correct, verifiable, secure and efficient.

INDEX TERMS Cloud computing, secure outsourcing, local sensitive hashing, approximate nearest neighbor search.

I. INTRODUCTION

As illustrated in [1], the computations of ANN (approximate nearest neighbor) search on high-dimensional dataset are basic in manly applications. The purpose of ANN is to find the approximate nearest neighbor vectors in a given dataset.

The associate editor coordinating the review of this manuscript and approving it for publication was Massimo Cafaro.

And, because of the better performance of ANN on high-dimensional large-scale data than other methods, ANN has been applied in various high-dimensional large-scale fields, including product recommendation [2], image retrieval [3], clustering [4], etc. However, existing algorithms for ANN can be efficient when the data dimensionality is small even if the data dimensionality is large [5], [6]. As the data dimension increases, these exact algorithms may even be less efficient than brute force linear scanning due to dimensionality disaster [7]. Nonetheless, many ANN problems often involve high-dimensional large-scale data, for example, many large multimedia retrieval applications use ANN searches [8]. These applications with high-dimensional large-scale data require efficient processing of ANN searches [8].

Moreover, with the continuous growth of the Internet of Things (IoT) and individual devices, the implementation of ANN on resource-constrained devices has become increasingly common. However, due to the limited computational power of these devices, they cannot perform the training and large-scale ANN search processes independently. Outsourcing the ANN search tasks to cloud servers can alleviate the resource constraints of edge devices, but this raises security and verification concerns. The sensitive data collected by resource-constrained devices must be securely outsourced to the cloud server, and a verification mechanism is required to identify any potential improper behavior from the cloud server. Additionally, to ensure the efficiency of the outsourcing and verification mechanism, they should be designed to avoid complex and time-consuming operations.

In light of these challenges, the motivation behind our research is to develop a secure and efficient outsourcing mechanism, the ''outsourced *LSH* algorithm,'' specifically tailored for ANN on high-dimensional large-scale data. This mechanism aims to enable resource-constrained devices to securely and efficiently perform the ANN task while preserving the privacy of the client's information and providing a lightweight verification mechanism with a high level of certainty. By addressing these issues, we aim to enhance the applicability and performance of ANN in various high-dimensional large-scale fields, promoting the advancement of data mining and related domains.

In the rest of this paper, we present our contributions and the proposed outsourced *LSH* algorithm, followed by a thorough analysis and evaluation of its correctness, security, and efficiency. We also demonstrate its potential application in real-world scenarios, concluding with future research directions.

The contributions can be summarized below:

- First, we propose a secure outsourcing mechanism (outsourced *LSH* algorithm) for ANN problems on high-dimensional large-scale data. This mechanism enables resource-constrained devices to complete the ANN task securely and efficiently.
- Second, to ensure the privacy of the resource-constrained client's information, we use a lightweight arithmetic method to obscure the client's information. Meanwhile, in order to verify the improper behavior of cloud servers, we built a lightweight verification mechanism with probability 1.
- Third, to analyze the proposed scheme in detail, we analyze the proposed method through detailed mathematical analysis. Meanwhile, to evaluate the efficiency and correctness of proposed scheme, we realize the proposed method with Python 3.7 and CIFAR-10 dataset. The results of the experiment show that our proposed scheme is correct and efficient.
- Fourth, to illustrate the application of the proposed scheme, we design an architecture of the application model and demonstrate the potential application fields and the method to apply the proposed scheme.

The remainder of the rest is organized as follows: Section II provides a literature review of *LSH*, secure outsourcing computations and cloud/edge computing. Section III gives a brief introduction to *LSH* and secure outsourcing computation. Section IV provides the detailed scheme and an application model for the proposed scheme. Section V performs a detailed theoretical correctness, security, and efficiency analysis of the proposed scheme. Section VI evaluates the efficiency and correctness of the proposed scheme through experiments and shows the results. Section VII takes an example to show the application of the proposed scheme. Finally, Section VIII concludes the study and gives future works.

II. RELATED WORK

A. MACHINE LEARNING AND LSH

In this information era, with the development of edge devices and networks, high-dimensional data are exploded on the Internet. With that comes the needs of fast ANN search on high-dimensional data in many fields. However, these high-dimensional data also bring a great challenge to fast ANN search because of the volume and high dimensionality of the data. ANN one of the most popular algorithms on nearest neighbor search because of its great benefit in the improvement of computational speed and storage reduction [9]. And, in multiple ANN search methods, *LSH* (Locality Sensitive Hashing) is one of the most popular methods for ANN search problems in high-dimensional data [10]. Actually, various *LSH* variants have been proposed to get approximate nearest neighbors on different fields [11], [12], [13], [14], [15], [16], [17]. For instance, Gan et al. [16] in 2012 presented a new *LSH* scheme -Collision Counting *LSH* (C2LSH), which can construct hash functions which are ''dynamic'' compound. With a base of m single *LSH* functions, this method improves the query quality of C-approximate NN search. In 2009, Kulis and Grauman [17] proposed a method to adapt arbitrary kernel functions using *LSH*, which allows for a sub-linear time approximate similarity search.

There is also a lot of research devoted to improving the accuracy and efficiency of *LSH* models [1], [11], [12], [13], [15]. For example, Datar et al. [18] proposed a novel

LSH scheme based on p-stable distributions to improve the performance of *LSH*. Liu et al. [12] presented an incremental search-based $\mathcal{C}$-ANN to improve the I/O performance. Tao et al. [11] developed an LSB-tree access method to improve the efficiency of ANN search problems.

B. SECURE OUTSOURCING COMPUTATIONS

Extensive research of security outsourcing schemes for scientific computing have been studied. In terms of the proposal of various outsourcing methods, Atallah et al. proposed the first generic architecture to outsource scientific computations in 2002 [19]. However, the architecture lacks the verification mechanism of computing results by the cloud servers. Then, in order to solve the large-scale systems of linear equations with the help of a fully malicious cloud server, Chen et al. [20] designed a secure outsourcing method in 2015. They utilized serval special sparse matrixes to obscure the input and output data. Meanwhile, they proved through experiments that the client can use their method to detect the improper behavior of cloud servers with probability 1. In 2018, Salinas et al. proposed a secure outsourcing approach for solving large-scale sparse linear systems of equations, which are the most common and fundamental problems in big data [21]. This method not only focuses on efficiency improvement but also focus on memory I/O complexities. They demonstrated that the memory issue of I/O operations is important in making the outsourcing method practical.

Much research has been done to free resource-constrained devices (UAVs, individual phones, etc.) from complex encryption operations [22], [23], [24], [25], [26]. For example, in order to free IoT devices from solving quadratic congruences, Zhang et al. proposed two practical and secure outsourcing algorithms [27]. The method outsourced the heavy computation to a single cloud server and guarantee the IoT device can identify the improper behavior with probability 1. They also take the Rabin algorithm as an example to show the application of the method in IoT applications. For addressing the problem of using untrusted cryptographic helpers, such as cloud servers, Hohenberger et al. provided a formal security definition for outsourcing computations to an untrusted helper [24]. They also provided a framework to quantify the efficiency and verifiability of an outsourcing method. They take the problem of outsourcing modular exponentiation as an example to show how to securely outsource computations. To alleviate the local burden of secret key updates, Yu et al. proposed a secure outsourced key updates method [28]. To verify the outsourced secrets key from an authorized party, they used a third-party auditor (TPA) to audit the storage and resist the key exposure.

C. CLOUD/EDGE COMPUTING

Cloud computing brings a new way for resource-constrained devices to operate complex computations [29], [30]. However, due to the requirement for data privacy in cloud computing, how to guarantee the data privacy when conduct cloud computing has become a critical issue. Many reaches have been done to guarantee privacy in cloud computing, including the security in storage [31], [32], [33] and the security in computation [20], [34], [35].

There is a great deal of research work on secure outsourcing architecture design. For example, in 2020, Lin et al. [36] focus on outsourcing the bilinear pairings computation, which is the most time-consuming operation in pairing-based cryptographic protocols, and designed a new outsourcing scheme for block chain devices. In order to solve the quadratic congruence problems which is a widely applied operation in cryptographic protocols on resource-constrained devices, Zhang et al. [27] presented two practical and secure outsourcing algorithms. The proposed algorithms enable the IoT devices to outsource the heavy computations of solving quadratic congruences to a single cloud server and do not leak the privacy of computation. Meanwhile, the proposed scheme can identify any improper behavior of the cloud server with probability 1. In 2016, Ye et al. [37] presented a secure outsourcing algorithm for modular exponentiation based on the new mathematical division under the setting of two non-colluding cloud servers. The privacy of outsourced data can be guaranteed, and efficiency is improved.

There has also been a lot of research on how to securely outsource scientific computations to cloud servers. For solving the modular exponentiation problems on IoT devices, Li et al. [38] first proposed a distributed outsourcing scheme for modular exponentiation in 2020. With this scheme, the user can not only protect the privacy of users in the outsourcing process but also identify the invalid results from edge nodes with a high probability. In order to solve the modular inversion problems on resource-constrained clients, Tian et al. [39] presented a new unimodular matrix transformation technique in 2022. This is the first secure outsource computation method that supports arbitrary and variable moduli. This method can verify the correctness of the cloud server results with 1 as the optimal probability in one round iteration.

III. PRELIMINARIES

In this section, we give a brief introduction to the *LSH* algorithm and secure outsourcing computation.

A. LOCAL SENSITIVE HASHING

LSH (Local Sensitive Hashing) is one of the ANN (approximate nearest neighbor) search algorithms. The advantage of this algorithm is that it allows us to only concentrate on similar objects of large-scale data. The original ANN search is to traverse each item of the given data and to find the approximate nearest neighbors. However, with the development of technology and the continuous improvement of edge devices' equipment performance, there are more and more high-dimensional large-scale datasets, and the volume of these datasets is getting larger and larger. If the original ANN search model is used to search these high-dimensional large-scale datasets, the time consumption is unacceptable.

In this situation, the *LSH* algorithm shows its powerful advantages.

The basic design idea of *LSH* is as follows. First, we need to build a number of different hash functions to compute the multiple hash items. One hash function needs to be carefully designed to represent the approximate similarity of the data in one aspect. This means that if items are similar in this aspect, they are likely to appear in the same hash bucket of this hash function. This way, if we want to search for similarity items, we can just check the candidates that fall into the same hash bucket. Meanwhile, if you want to search for similar items from various aspects, you can generate multiple hash functions to represent multiple aspects.

LSH was one algorithm that originally designed to solve the $(R,\ c)$-NN problem, that is, if given a query q, the algorithm will perform:

If there exists $o \in \mathcal{D}$ such that $dict(o, q) \leq R$, then return an object $o' \in \mathcal{D}$ satisfying $dict(o', q) \leq cR$.

If $dict(o, q) > R$ for all $o \in \mathcal{D}$, then return nothing.

The basic principle of designing *LSH* is that the hash objects that are similar (smaller than a given distance) can be hashed into the same bucket with a high probability. Meanwhile, to support efficient $(R,\ c)$-NN search, *LSH* typically implements a series of hash functions H, which are random and independent of each other. This *LSH* family will be used to build multiple hash tables. If we want to increase the similarity requirement of items, items are considered highly similar only if they are in the same bucket in multiple hash tables.

For a domain $\mathbb{R}^d$ of the items set, one hash function in the *LSH* family is a mapping from $\mathbb{R}^d$ to U. The sensitivity of an *LSH* family to the locality can be measured in the following way:

Definition 1: For any constant $c > 1$ and probabilities $p1 > p2$, an *LSH* family $\mathcal{H} - h : \mathbb{R}^d \rightarrow U$ is called (r, cr, p_1, p_2)-sensitive if any $u, v \in \mathbb{R}^d$ satisfy [1]

- $\Pr_{\mathcal{H}}[h(u) = h(v)] \geq p_1$ when $dict(u, v) \leq r$; and
- $\Pr_{\mathcal{H}}[h(u) = h(v)] \leq p_2$ when $dict(u, v) > cr$.

In order to make a locality-sensitive family to be valuable, when $\mathcal{D}$ is the similarity measure, the equation needs to satisfy the inequalities $p_1 > p_2$ and $r_1 > r_2$.

In terms of measuring the item distance in *LSH*, a lot of *LSH* families with different distance metrics have been proposed [18], [40], [41]. The most classic one is the distribution *LSH* family H proposed by Datar et al. [18] in 2004, which is for l_p metrics based on p-stable. The specific of this method's hash function is as follows:

$$h_{a,b}(o) = \left\lfloor \frac{a^T o + b}{w} \right\rfloor \tag{1}$$

where $a \in \mathbb{R}^{1\times d}$, $o \in \mathbb{R}^d$, $w \in z$, $b \in [0, w]$.

In eq. (1), a is a vector consisting of randomly selected elements from a p-stable distribution. o is the data of a given item. w is the specified bucket width. b is a random variable from the range $[0, w]$.

For example, when Euclidean distance $dist(u, v) = s$ is used to calculate the distance between items u and v, under the hash function $h_{a,b} \in \mathcal{H}$, the probability that u and v are in the same bucket can be computed as [18]:

$$p(s) = \Pr\left[h_{a,b}(u) = h_{a,b}(v)\right] = \int_0^w \frac{2}{s}\varphi(\frac{t}{s})(1 - \frac{t}{s})dt,$$

where $\varphi(\cdot)$ is the distribution function of the standard normal distribution $\mathcal{N}(0, 1)$. If the value of w is fixed, $p(s)$ monotonically decreases with s. Thus, according to the definition, the family of hash functions $h_{a,b}$ is $(r, cr, p(r), p(cr))$-sensitive.

To make $(R,\ c)$-NN search more efficient, *LSH* algorithm typically uses multiple hash functions to build multiple hash tables from different aspects. These hash functions are randomly selected from the $(R, cR, p(R), p(cR))$-sensitive hash family. When using the hash functions, the hash values of each item are calculated by these hash functions. And an item is considered similar to the query item q when the item is hashed into the same bucket in every hash function. By this, the selected similar candidates need to be similar in multiple aspects, and the rigor and credibility of similarity will increase.

B. SECURE OUTSOURCING COMPUTATION

One purpose of secure outsourced computing is to enable resource-constrained devices to compute complex computing problems with the help of cloud servers. However, the privacy of the client cannot be leaked to the cloud server in this process. Currently, there is a lot of related research in this area [20], [34], [35]. Here, we use the proposed properties of most security outsourcing needs to have by Gao et al. [42], these properties are:

- **Security:** This property requires that when computing tasks are outsourced to the cloud service, the cloud server cannot obtain any private information from the input/output data.
- **Verifiability:** To protect the correctness of the computing results of the cloud server, the client or a third-party authority needs to check the correctness of the outsourced computing results from the cloud server.
- **Efficiency:** Because the purpose of outsourced computing is to allow resource-constrained clients to perform complex computational tasks, the cost of outsourced tasks should not be too complex, at least less than the complexity of performing the computations themselves.

IV. PROPOSED SCHEME

In this section, we first propose an application model for the outsourced computation of *LSH*. Then the concept and basic principle of the scheme are expounded. Finally, the overall proposed scheme is explained in detail.

A. APPLICATION MODEL

The proposed application model of outsourced *LSH* is shown in Fig. 1. In the process of outsourcing the *LSH* algorithm

to the cloud server, the database server will firstly generate a Secret Key (SK), which is used to obscure the raw data. Then, using the SK to obscure the database server's data, from (x, w) to (x', w'), in which x is the raw data, w is the project matrix. Then, the obscured data will transfer to the cloud server. In turn, the cloud server will calculate multiple HashTable$'$ by multiple hash functions, and then return them to the database server. The database server will verify the correctness of the HashTable$'$ through the proposed verification mechanism. After passing the verification, the database server will recover the final *HashTable* through SK. Otherwise, the database server will reject the HashTable$'$. In the database server, the query_x is calculated using the $wquery\_x^T$ to generate the query_hash_value, and then a fast ANN query is performed with HashTable to retrieve the nearest neighbors. And, in the application of multi-party ANN search, each client sends the query data query_x to the database server, and the database server can perform an outsourced fast multi-party ANN search through this database and return the nearest neighbors to the client.

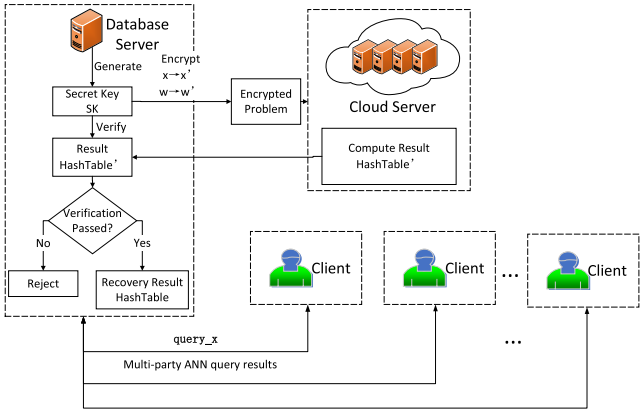


FIGURE 1. Application model.

B. CONCEPT AND BASIC PRINCIPLE

The basic principle of our scheme is to design a system that enables resource-constrained devices, such as individual phones, edge devices, and UAVs, to perform fast ANN search. To achieve this goal, we plan to utilize a secure outsourced computing mechanism to complete the computation task of *LSH*. In this study, we used the classic *LSH* algorithm, which is the formula shown in eq. (1), as the improved algorithm. For the purpose of outsourcing the *LSH* algorithm, we first analyzed the most computationally time-consuming part of this formula. Through the analysis of eq. (1), it is found that the most time-consuming part lies in the $a^T o$ calculation. Meanwhile, note that for an efficient ANN query, it is generally necessary to compute a hash family (multiple hash functions), that is, multiple $a^T o$ operations. Therefore, we considered outsourcing $a^T o$ computing to the cloud server. To make the complexity of outsourcing calculation not exceed that of local calculation, at least equal, the method of obscuring data with a series of elementary rotation matrices is chosen after considering a variety of obscuring algorithms and querying relevant literature. At the same time, to ensure the correctness of the cloud server computing results, we provide a verification mechanism to verify the correctness of results.

C. DETAILED SCHEME

We express our detailed algorithm in Algorithm 1. In the proposed algorithm, we first generate the projection matrix W on the local client, which is used to form the hash value of raw data. Then, in order to obscure the local client data, namely X and W, the client generates a series of elementary rotation matrices, l $(1 < l \ll \min(md))$ $m \times m$ Givens matrices $P_1, P_2, \cdots P_l$ and $ld \times d$ Givens matrices $Q_1, Q_2, \cdots Q_l$, including a random real number α as the secret key. Then, the secret key is used to obscure the data, as shown in eq. (4) and eq. (5). Then the obscured data is uploaded to the cloud server to calculate the hash tables of the obscured data.

The elementary rotation matrix (also known as Givens matrix) is chosen as the way of obscuring the data because the elementary rotation matrix is a sparse matrix with most elements being 0, so the computational burden is less than that of the original matrix multiplication, as shown in eq. (2). The specific construction of the elementary rotation matrix is as follows:

- Let $c = cos\theta$ and $s = sin\theta$.
- Set the i-th and j-th main diagonal elements as c.
- Set the other main diagonal elements as 1.
- Set the i-th row and j-th column elements as s.
- Set the j-th row and i-th column elements as $-s$.
- Set the other elements as 0.

$$T(i,j) = \begin{bmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & c & & & & s & & \\ & & & & 1 & & & & & \\ & & & \vdots & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & -s & & & & c & & \\ & & & & & & & & 1 & \\ & & & \vdots & & & & & & \ddots \\ & & & & & & & & & & 1 \end{bmatrix} \quad (2)$$

In eq. (1), elementary rotation matrix is orthogonal matrix. The elementary rotation matrix can perform the following three types of matrix operations:

• *Multiplication*: A multiplication operation multiplies the i-th row (resp. column) of a matrix by a non-zero scalar.

• *Permutation*: A permutation operation makes the two rows (resp. columns) of a matrix swap their location.

• *Addition*: An addition operation makes a matrix add j-th row (resp. column) multiplied by a non-zero scalar to the i-th row (resp. column).

Algorithm 1 The Proposed Algorithm

Input: $X \in \mathbb{R}^{m\times d}$, a large-scale matrix; $W \in \mathbb{R}^{t\times d}$, t projection vectors that conform to p-stable distributions.
Output: Verification result v; HashTable $\in \mathbb{R}^{t\times m}$, t = rows × banks.

Step 1. **KeyGen**(λ) → (SK):

- The client chooses a random real number α and $2l$ elementary rotation matrices: $P_1, P_2, \cdots P_l, Q_1, Q_2, \cdots Q_l$ as the secret key SK.

$$\mathrm{SK} = \{\alpha, P_1, P_2, \cdots P_l, Q_1, Q_2, \cdots Q_l\} \tag{3}$$

Step 2. **ComDis**$_{SK}(X, W) \longrightarrow (X', W')$:

- Client computes X' as:

$$X' = P_1P_2\cdots P_l(\alpha X)Q_1Q_2\cdots Q_l \tag{4}$$
$$W' = (\alpha W)Q_1Q_2\cdots Q_l \tag{5}$$

Step 3. **Compute**(X') $\longrightarrow$ (HashTable$'$):

- Cloud chooses t projection vectors that conform to p-stable distributions to form the projection matrix W.
- Computes HashTable$'$ as follows:

$$\mathrm{HashTable}' = W'X'^{T}, W \in \mathbb{R}^{t\times d} \tag{6}$$

Step 4. **Verify**(HashTable$'$) → v:

- The client randomly generates a vector $r \in \mathbb{R}^{1\times t}$ and calculates the local V_1 as:

$$V_1 = (rW')X'^{T} \tag{7}$$

- The client calculates the V_2 by r and HashTable$'$, as shown in eq.(8). Then, the client checks if V_1 equal to V_2. If they are equal, then accept HashTable$'$. Otherwise, rejects HashTable$'$.

$$V_2 = r\mathrm{HashTable}' \tag{8}$$

Step 5.**Recover**(SK, HashTable$'$) → (HashTable):

$$\mathrm{HashTable} = \left\lfloor \frac{\frac{1}{\alpha^2}(\mathrm{HashTable}'P_1P_2\cdots P_l) + b}{\mathrm{num\_hash}} \right\rfloor \tag{9}$$

In the actual operation of the proposed scheme, first of all, the client will use $SK = \{\alpha, P_1, P_2, \cdots P_l, Q_1, Q_2, \cdots Q_l\}$ as the secret key to obscure X and W to X' and W' through eq. (4) and eq. (5). Then the obscured data will upload to the cloud server for multiple hash functions calculation and multiple HashTable$'$ are obtained. Then, the result HashTable$'$ is passed back to the client. In this phase, the communication consumption is reduced because the dimension of the data becomes smaller. Then the client determines whether to accept HashTable$'$ by matching V_1 and V_2 according to the equation of eq. (7) and eq. (8). If V_1 is not equal to V_2, the cloud server will be considered to have improper behavior and the returned *HashTable*$'$ will be rejected. Otherwise, if they are the same, multiple HashTable$'$ will be recovered toHashTable by eq. (9). Finally, the multi-party clients passes the local HashTable to the database server for fast multi-party ANN queries later.

V. THEORETICAL PERFORMANCE ANALYSIS

A. CORRECTNESS ANALYSIS

The generated secret key in the client is SK $= \{\alpha, P_1, P_2, \cdots P_l, Q_1, Q_2, \cdots Q_l\}$, in which $P = P_1P_2\cdots P_l$ and $Q = Q_1Q_2\cdots Q_l$. The local client will use the SK to recover the HashTable from HashTable$'$ as HashTable $= \frac{1}{\alpha^2}$(HashTable$'P$), where

$$\begin{aligned}
\mathrm{HashTable} &= \left\lfloor \frac{\frac{1}{\alpha^2}(\mathrm{HashTable}'P) + \mathrm{b}}{\mathrm{num_hash}} \right\rfloor \\
&= \left\lfloor \frac{\frac{1}{\alpha^2}\left(W'X'^{T}P\right) + \mathrm{b}}{\mathrm{num_hash}} \right\rfloor \\
&= \left\lfloor \frac{\frac{1}{\alpha^2}\left((\alpha W)\,Q\,(P\,(\alpha X)\,Q)^{T}\,P\right) + b}{\mathrm{num_hash}} \right\rfloor \\
&= \left\lfloor \frac{\frac{1}{\alpha^2}\left((\alpha W)\,QQ^{T}\,(\alpha X)^{T}\,P^{T}P\right) + b}{\mathrm{num_hash}} \right\rfloor \\
&= \left\lfloor \frac{\frac{1}{\alpha^2}\left(\alpha^2 WX^{T}\right) + b}{\mathrm{num_hash}} \right\rfloor \\
&= \left\lfloor \frac{WX^{T} + b}{\mathrm{num_hash}} \right\rfloor
\end{aligned}$$

Thus, according to the above calculation, we can conclude that the recovered HashTable is the correct result, which is consistent with the calculation of the original HashTable, as shown in eq. (1). At the same time, to verify the correctness of the lightweight verification mechanism, we provide a correctness analysis of the verification scheme, where

$$\begin{aligned}
V_1 &= (\mathrm{r}W')X'^{T} \\
&= \mathrm{r}W'X'^{T} \\
&= \mathrm{r}\mathrm{HashTable}' \\
&= V_2
\end{aligned}$$

Thus, if $V_1 = \mathrm{r}\mathrm{HashTable}'$, the calculated results computed by the cloud server are correct.

B. SECURITY AND VERIFICATION ANALYSIS

We analyze the security and verification performance of the proposed outsourced *LSH* scheme through detailed theoretical analysis.

1) DATA PRIVACY

The proposed outsourced *LSH* scheme can protect the privacy of the input/output.

Proof: To protect the input data of the local client, that is, X and W, we adopt a lightweight data obscuring scheme to obscure X and W. That is, the X and W will be calculated with the private key SK through eq. (4) and eq. (5) to hide the information of the original data. In the process, X will multiply by $2l$ elementary rotation matrices and a real number a, and W will multiply by l elementary rotation matrices and a real number a. That is to say, if malicious parties want to obtain the original X and W, the malicious cloud server or other malicious parties need to crack a and $2l$ elementary rotation matrices. However, as a is a real number, the range of a is large, and $2l$ elementary rotation matrices are also randomly generated, so the range of brute force cracking is large. So, the original data is well obscured. In terms of brute force cracking the original data, if the probability of guessing an element of the input matrix is assumed to be $\frac{1}{\delta}$ (since each element of the matrix is a real number, the range is wide), then the overall probability of guessing X is $\text{negli}(md) = \frac{1}{\delta^{md}}$. The overall probability of guessing W is $\text{negli}(td) = \frac{1}{\delta^{td}}$. The overall probability of guessing the HashTable is $\text{negli}(tm) = \frac{1}{\delta^{tm}}$. They are all nonpolynomial-time functions. At the same time, since the HashTable is already the hash value of the original data, the original client data cannot be recovered even if the HashTable is guessed. Therefore, for the input and output of the proposed scheme, the data privacy of the input/output can be better guaranteed.

2) VERIFICATION PERFORMANCE

The correctness of results from the cloud server can be verified with 100% probability.

Proof: Based on the verifiability requirement of outsourcing computation, as shown in section III (b), an outsourcing scheme needs to verify the computing results from the cloud server. Therefore, in the proposed scheme, assuming that the cloud server has improper behavior, the client needs verification mechanism to check the occurrence of improper behavior and reject the incorrect calculation results of the cloud server. In the proposed scheme, we use the verification mechanism of checking whether V_1 is equal to V_2 to verify the correctness of the results. If the cloud service wants to forge the result HashTable$'$, at least every column of the HashTable$'$ that needs to forge suitable values. However, every element of the HashTable$'$ is a real number. It is difficult to forge a suitable *HashTable*$'$. Thus, the correctness of results from the cloud server can be verified with 100% probability.

C. EFFICIENCY ANALYSIS

We analyze the efficiency of the proposed scheme through detailed theoretical analysis in terms of computation efficiency, and communication consumption.

1) COMPUTATION EFFICIENCY

The proposed algorithm is $((2md + 2td + 2tm + 4ml + 4dl)/(\text{num\_hash\_table} * tdm))$ efficient.

Proof: We assume the scalar multiplication operation is SM and the fewer computational computations are neglected in the analysis. When one hash table is generated, in the step ProbGen, the client performs $md + td + 2ml + 4dl$SM. In the Compute phase, the cloud server performs tdmSM. In the Verify phase, the client executes $td + dm + tm$SM. In the Recover phase, the client executes $tm + 2ml$SM. Therefore, to summarize, the client executes a total $2md + 2td + 2tm + 4ml + 4dl$SM. And the cloud server performs a total of tdmSM. Without outsourcing, the client executes a total tdmSM, as shown in eq. (1), the same as the complexity of the phase Compute. Thus, the proposed scheme is $((2md + 2td + 2tm + 4ml + 4dl)/(tdm))$ efficient when generating one hash table. However, in the *LSH* algorithm, multiple hash tables are often generated. Therefore, if num_hash_table is the number of generated hash tables, the proposed scheme is $((2md + 2td + 2tm + 4ml + 4dl)/(\text{num\_hash\_table} * tdm))$ efficient. The time cost of different phases is shown in Table 1.

TABLE 1. Time cost of phases.

	ProbGen	Compute	Verify	Recover
SM	$md + td + 2ml + 4dl$	tdm	$td + dm + tm$	$tm + 2ml$

2) COMMUNICATION CONSUMPTION

We show the communication consumption in the proposed scheme, as shown in Table 2. Among 5 phases, only ComDis phase and Compute phase have communication consumption. Because the KeyGen, Verify, and Recover phases are all carried out on the client side, these three phases have no communication consumption. Therefore, the total communication consumption includes two parts. The first one is communication consumption in the ComDis phase, the client needs to send the obscured matrix (X', W') to the cloud server. The second is the communication consumption in the Computephase. After the cloud server completes the computation of the HashTable$'$, the cloud server needs to transfer the HashTable$'$ to the client.

TABLE 2. Communication cost of phases.

Phase	Communication Cost
ComDis	$C \xrightarrow{(X'_{m\times d}, W'_{t\times d})} S$
Compute	$S \xrightarrow{\text{HashTable}'_{t\times m}} C$
Verify	0
Recover	0

VI. EXPERIMENTAL PERFORMANCE EVALUATION

In this section, we verify the correctness and efficiency of the proposed scheme through experiments. The experimental results show our proposed scheme is efficient and correct.

A. EVALUATION METHODOLOGY

In our experiments, we used the computer with Windows server 2012, Intel Xeon Bronze 3106 CPU at 1.7 GHz, and 512GB memory as the testbed. Actually, we initially deployed the original *LSH* algorithm to a normal computer, but when we tried to incorporate more image data into the calculation, the memory burst. But if we used a hard disk to store the calculation results, the I/O efficiency would not be acceptable. So, we used a computer with 512GB of memory as the local client. In terms of the cloud server, we choose Alibaba Cloud. For the programming language, we use Python 3.7 to simulate our proposed algorithm.

In terms of the efficiency metric, we tested the time consumption of different phases, time consumption on the cloud server side and the client side, and the time consumption of outsourced *LSH* and non-outsourced *LSH*, and compared them. In terms of the correctness verification of the proposed algorithm, we verify the consistency of V_1 and V_2 through experiments.

We use the CIFAR-10 benchmark dataset as the experimental dataset, as shown in Table 3. The characteristics of the dataset are summarized as follows:

- *CIFAR* − 10 is an image dataset that contains 60,000 color images of 10 classes. Each class contains 6,000 images (the size of each image is 32×32).

TABLE 3. Dataset in the experiments.

Dataset	Dimensionality d	Size
CIFAR-10	32×32	60,000

In the experiment, we generate randomly 1000 images (10 categories, 100 images per category) as the query dataset and 1000 to 9500 images (10 categories, 100 to 950 images per category) as the *HashTable* generation dataset. The query is considered valid only if the category label of the queried image is consistent with that of the retrieved image.

B. EVALUATION DEMONSTRATION

In Fig. 2, we briefly describe the experimental process of the proposed outsourced *LSH* algorithm. First, we select a certain amount of data from CIFAR-10 as the *HashTable* generation dataset and a certain amount of data as the query dataset. Then, we execute our proposed algorithm to generate the *HashTable* of *HashTable* generation dataset. Then, the data in the query dataset are successively queried through the *HashTable* for ANN search, and top k is selected as the query result, and the accuracy is calculated.

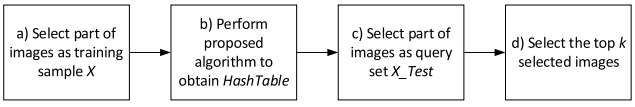
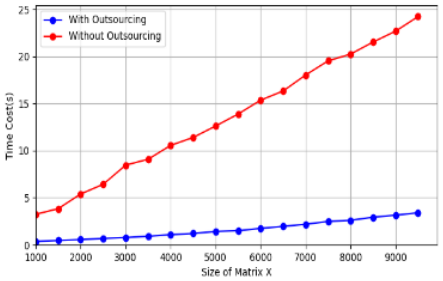
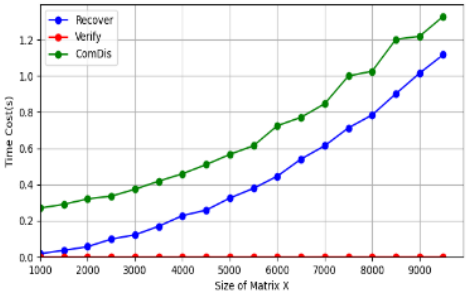


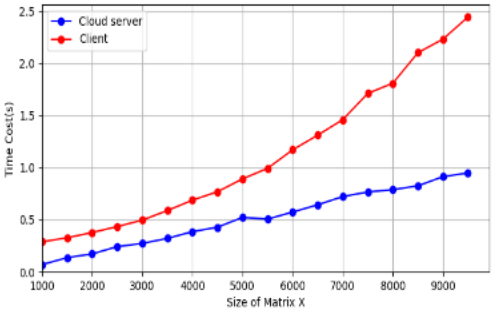
FIGURE 2. The experimental procedure.



(a) Time consumption on the client



(b)Time consumption among 3 phases



(c)Time consumption on the client and cloud server

FIGURE 3. Time consumption comparison for the proposed algorithm.

C. EVALUATION RESULTS

We analyze the efficiency of the proposed scheme by comparing the time consumption in Fig. 3. The amount of *HashTable* generation dataset (the size of X) is from 1000 to 9500. In Fig. 3(a), the time consumption of *LSH* outsourcing and non-outsourcing is compared. It can be seen that the outsourced *LSH* algorithm does have a great improvement in time consumption metric. Fig. 3(b) shows the client time consumption in the Recover, Verify and ComDis phases, showing that the ComDis phase has the highest time consumption. This is because we need to obscure the input data X and W by SK. X has to do matrix computations with $2l$ elementary rotation matrix operations and the real number a. W needs to do matrix computations with l elementary rotation matrix operations and the real number a. In these operations, although the elementary rotation matrix is sparse, there is still a certain amount of time consumption. The second time consumption phase is the *Recover* phase. This is because the dimension of the returned data from the cloud server has decreased com-

pared with the original data. Although it also needs to operate with l elementary rotation matrices and the real number $1/\alpha^2$, the overall operation time consumption will decrease. Fig. 3(c) shows the time consumption on the client side versus the cloud side in the outsourced *LSH* algorithm. We can see that the cloud server consumes less time. It is believed that if more cloud resources are allocated to assist computing, the cloud time consumption will be smaller. Therefore, it is further proved that because the proposed algorithm is for a resource-constrained client, we need to reduce the computational complexity and storage requirements of the client to improve the efficiency of the algorithm.

To verify the correctness of the proposed scheme, we compare the V_1 and V_2 in the experiments, as shown in Fig. 4. In the Fig. 4, the v is the local V_1 calculated by eq.7. The $v2$ is the V_2 calculated by HashTable$'$, as shown in eq.8. It can be seen than they are the same, which verifies the correctness of the proposed scheme.

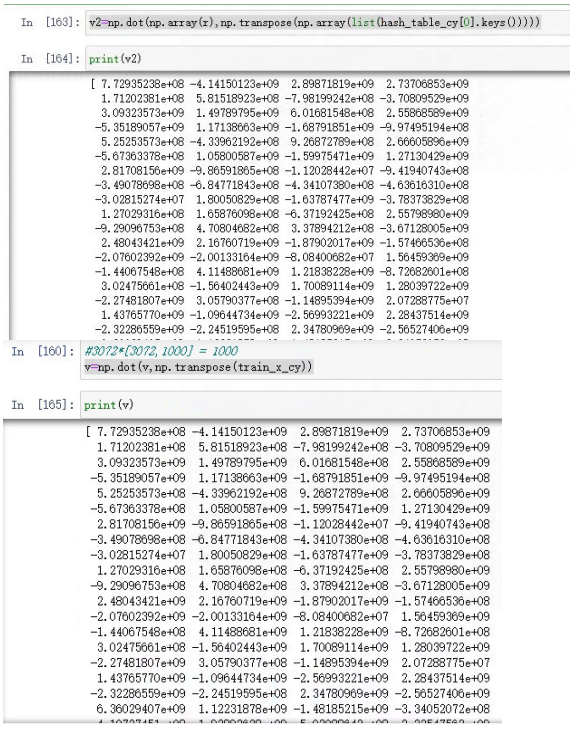


FIGURE 4. Verification of v.

VII. APPLICATIONS

In this section, we present some potential practical applications of the proposed scheme. Since *LSH* is considered one of the most promising indexing methods for approximate nearest neighbor search (ANN search) [43], it has been widely used in many fields such as entity resolution [44], fingerprint matching [45], and user context privacy protection [46]. And for the ANN search on high-dimensional large-scale data, the computing power and memory requirement of many edge/individual devices is not enough. Meanwhile, in these applications, if the client's data contains private information, the use of outsourced computing will also face the privacy of the client's data. Therefore, to protect the privacy of clients' data while utilizing cloud servers for outsourcing computing, our proposed *LSH* security outsourcing scheme will be a solution.

In 2011, Raffaele Cappelli et al. once proposed a new *LSH*-based query scheme for fingerprint identification in large databases [45]. Six datasets are used in their experiment, as shown in Table 4. However, in terms of privacy protection, a person's fingerprint characteristics should belong to private information, especially when the purpose of fingerprint identification is for commercial profit, the user's fingerprint information should not be disclosed, otherwise, the disclosure of data may cause quite serious losses. For example, Twitter has been fined $150 million and subjected to new oversight for disclosing users' privacy in 2022. Therefore, in order to ensure data privacy and accelerate the algorithm, institutions can use our proposed security outsourcing *LSH* algorithm to complete fingerprint ANN search. By outsourcing part of the calculation of the *LSH* algorithm and transferring the results to the database server, the client can efficiently calculate the fingerprint hash tables with the help of the remote cloud server and ensure the privacy of the information and the verifiability of the results. Then, the server or a third party can perform fast ANN searches through the hash tables database server.

TABLE 4. Dataset in the method of raffaele cappelli [45].

Dataset	Size
NIST DB4	2,000 fingers
NIST DB4 (Natural)	1,204 fingers
NIST DB14	2700 fingers
FVC2000 DB2	100 fingers
FVC2000 DB3	100 fingers
FVC2002 DB1	100 fingers

VIII. CONCLUSION AND FUTURE WORK

In this study, we proposed a secure and verifiable scheme to outsource the classical *LSH* algorithm for the ANN query problems on resource-constrained devices. The scheme avoids the leakage of input data by obscuring the client data. At the same time, the correctness of the computing results of the cloud server can be verified with 100% possibility through the proposed verification mechanism. We detailed the design principles and the proposed scheme in Section IV. To verify the efficiency and correctness of our scheme, we conduct experiments based on the CIFAR-10 dataset and described the experimental results in Section V. The experiments show that the proposed scheme is efficient and correct.

With the rise of machine learning, more and more devices, such as individual phones and edge devices, need to do complex machine learning operations. Although with the development of technology, the performance of personal devices and edge devices is getting higher and higher, the volume and dimension of processed data are also increasing. Therefore,

cloud-aided machine learning is an idea to solve this problem. We can outsource part of the complex computations of machine learning algorithms to enable resource-constrained devices to complete complex machine learning tasks on the basis of ensuring the privacy of client's data. This not only requires cloud servers to be able to run part of machine learning algorithms' computations but also needs to run on encrypted data. Now, there are a lot of studies on this problem, such as homomorphic encryption techniques. However, a serious problem is that since the computing is done on resource-constrained devices, the encryption algorithm should not be too complex, and the results returned by the cloud service should be able to enable the client to recover the data correctly. Therefore, based on these considerations, based on the research of others, we designed an outsourced *LSH* scheme that can verify the results from the cloud server with probability 100% to help resource-constrained devices quickly complete the *LSH* algorithm for fast ANN searches, especially, for high-dimensional and large-scale dataset.

In future work, we plan to add federated learning and multi-party computation mechanism to the proposed scheme, bring more devices into the scope of fast ANN query, and explore a more secure, efficient, outsourced, practical and multi-party ANN scheme.

REFERENCES

[1] H. Wang, C. Yang, X. Zhang, and X. Gao, "Efficient locality-sensitive hashing over high-dimensional streaming data," *Neural Comput. Appl.*, vol. 35, no. 5, pp. 3753–3766, Feb. 2023, doi: 10.1007/s00521-020-05336-1.

[2] J. Suchal and P. Návrat, "Full text search engine as scalable k-nearest neighbor recommendation system," in *Artificial Intelligence in Theory and Practice III* (IFIP Advances in Information and Communication Technology), vol. 331. AICT, 2010, pp. 165–173, doi: 10.1007/978-3-642-15286-3_16.

[3] G. Giacinto, "A nearest-neighbor approach to relevance feedback in content based image retrieval," in *Proc. 6th ACM Int. Conf. Image Video Retr.* New York, NY, USA: ACM Press, Jul. 2007, pp. 456–463, doi: 10.1145/1282280.1282347.

[4] T. Liu, C. Rosenberg, and H. Rowley, "Clustering billions of images with large scale nearest neighbor search," in *Proc. IEEE Workshop Appl. Comput. Vis. (WACV)*, Feb. 2007, p. 28, doi: 10.1109/WACV.2007.18.

[5] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, doi: 10.1145/361002.361007.

[6] K. L. Cheung and A. W.-C. Fu, "Enhanced nearest neighbour search on the R-tree," *ACM SIGMOD Rec.*, vol. 27, no. 3, pp. 16–21, Sep. 1998, doi: 10.1145/290593.290596.

[7] R. Weber, H. J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proc. 24rd Int. Conf. Very Large Data Bases*, 1998, pp. 194–205.

[8] O. Jafari and P. Nagarkar, "Experimental analysis of locality sensitive hashing techniques for high-dimensional approximate nearest neighbor searches," in *Proc. Australas. Database Conf.*, 2021, pp. 62–73, doi: 10.1007/978-3-030-69377-0_6.

[9] K. Zhao, H. Lu, and J. Mei, "Locality preserving hashing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 28, no. 1, Jun. 2014, doi: 10.1609/aaai.v28i1.9133.

[10] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529, Accessed: Oct. 8, 2022. [Online]. Available: http://dl.acm.org/citation.cfm?id=645925.671516

[11] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Efficient and accurate nearest neighbor and closest pair search in high-dimensional space," *ACM Trans. Database Syst.*, vol. 35, no. 3, pp. 1–46, Jul. 2010, doi: 10.1145/1806907.1806912.

[12] W. Liu, H. Wang, Y. Zhang, W. Wang, and L. Qin, "I-LSH: I/O efficient c-approximate nearest neighbor search in high-dimensional space," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1670–1673, doi: 10.1109/ICDE.2019.00169.

[13] Y. Liu, J. Cui, Z. Huang, H. Li, and H. T. Shen, "SK-LSH: An efficient index structure for approximate nearest neighbor search," *Proc. VLDB Endowment*, vol. 7, no. 9, pp. 745–756, May 2014, doi: 10.14778/2732939.2732947.

[14] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," *Proc. VLDB Endowment*, vol. 9, no. 1, pp. 1–12, Sep. 2015, doi: 10.14778/2850469.2850470.

[15] T. Christiani, "Fast locality-sensitive hashing frameworks for approximate near neighbor search," in *Proc. Int. Conf. Similarity Search Appl.*, 2019, pp. 3–17, doi: 10.1007/978-3-030-32047-8_1.

[16] J. Gan, J. Feng, Q. Fang, and W. Ng, "Locality-sensitive hashing scheme based on dynamic collision counting," in *Proc. Int. Conf. Manag. Data (SIGMOD)*. New York, NY, USA: ACM Press, 2012, p. 541, doi: 10.1145/2213836.2213898.

[17] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 2130–2137, doi: 10.1109/ICCV.2009.5459466.

[18] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on *p*-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, Jun. 2004, p. 253, doi: 10.1145/997817.997857.

[19] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. E. Spafford, "Secure outsourcing of scientific computations," *Adv. Comput.*, vol. 54, pp. 215–272, Jan. 2002, doi: 10.1016/S0065-2458(01)80019-X.

[20] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 69–78, Jan. 2015, doi: 10.1109/TIFS.2014.2363765.

[21] S. Salinas, C. Luo, X. Chen, W. Liao, and P. Li, "Efficient secure outsourcing of large-scale sparse linear systems of equations," *IEEE Trans. Big Data*, vol. 4, no. 1, pp. 26–39, Mar. 2018, doi: 10.1109/TBDATA.2017.2679760.

[22] B. Chevallier-Mames, J.-S. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.*, 2010, pp. 24–35, doi: 10.1007/978-3-642-12510-2_3.

[23] H. Tian, F. Zhang, and K. Ren, "Secure bilinear pairing outsourcing made more efficient and flexible," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur.*, Apr. 2015, pp. 417–426, doi: 10.1145/2714576.2714615.

[24] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. Theory Cryptogr. Conf.*, 2005, pp. 264–282, doi: 10.1007/978-3-540-30576-7_15.

[25] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology (CRYPTO)*. Berlin, Germany: Springer, pp. 89–105, doi: 10.1007/3-540-48071-4_7.

[26] X. Chen, W. Susilo, J. Li, D. S. Wong, J. Ma, S. Tang, and Q. Tang, "Efficient algorithms for secure outsourcing of bilinear pairings," *Theor. Comput. Sci.*, vol. 562, pp. 112–121, Jan. 2015, doi: 10.1016/j.tcs.2014.09.038.

[27] H. Zhang, J. Yu, C. Tian, G. Xu, P. Gao, and J. Lin, "Practical and secure outsourcing algorithms for solving quadratic congruences in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2968–2981, Apr. 2020, doi: 10.1109/JIOT.2020.2964015.

[28] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016, doi: 10.1109/TIFS.2016.2528500.

[29] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018, doi: 10.1109/ACCESS.2017.2778504.

[30] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017, doi: 10.1109/JIOT.2017.2683200.

[31] X. Lu, Z. Pan, and H. Xian, "An integrity verification scheme of cloud storage for Internet-of-Things mobile terminal devices," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101686, doi: 10.1016/j.cose.2019.101686.

[32] J. Yu and R. Hao, "Comments on SEPDP: Secure and efficient privacy preserving provable data possession in cloud Storage," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 2090–2092, Nov. 2021, doi: 10.1109/TSC.2019.2912379.

[33] X. Gao, J. Yu, W.-T. Shen, Y. Chang, S.-B. Zhang, M. Yang, and B. Wu, "Achieving low-entropy secure cloud data auditing with file and authenticator deduplication," *Inf. Sci.*, vol. 546, pp. 177–191, Feb. 2021, doi: 10.1016/j.ins.2020.08.021.
[34] J. Zhang, Y. Yang, and Z. Wang, "Outsourcing large-scale systems of linear matrix equations in cloud computing," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2016, pp. 438–447, doi: 10.1109/ICPADS.2016.0066.
[35] X. Lei, X. Liao, T. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Inf. Sci.*, vol. 280, pp. 205–217, Oct. 2014, doi: 10.1016/j.ins.2014.05.014.
[36] C. Lin, D. He, X. Huang, X. Xie, and K.-K.-R. Choo, "Blockchain-based system for secure outsourcing of bilinear pairings," *Inf. Sci.*, vol. 527, pp. 590–601, Jul. 2020, doi: 10.1016/j.ins.2018.12.043.
[37] J. Ye, Z. Xu, and Y. Ding, "Secure outsourcing of modular exponentiations in cloud and cluster computing," *Cluster Comput.*, vol. 19, no. 2, pp. 811–820, Jun. 2016, doi: 10.1007/s10586-016-0571-z.
[38] H. Li, J. Yu, H. Zhang, M. Yang, and H. Wang, "Privacy-preserving and distributed algorithms for modular exponentiation in IoT with edge computing assistance," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8769–8779, Sep. 2020, doi: 10.1109/JIOT.2020.2995677.
[39] C. Tian, J. Yu, H. Zhang, H. Xue, C. Wang, and K. Ren, "Novel secure outsourcing of modular inversion for arbitrary and variable modulus," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 241–253, Jan. 2022, doi: 10.1109/TSC.2019.2937486.
[40] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal LSH for angular distance," 2015, *arXiv:1509.02897*.
[41] K. Eshghi and S. Rajaram, "Locality sensitive hash functions based on concomitant rank order statistics," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, p. 221, doi: 10.1145/1401890.1401921.
[42] P. Gao, H. Zhang, J. Yu, J. Lin, X. Wang, M. Yang, and F. Kong, "Secure cloud-aided object recognition on hyperspectral remote sensing images," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3287–3299, Mar. 2021, doi: 10.1109/JIOT.2020.3030813.
[43] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, 1998, pp. 604–613.
[44] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *Proc. VLDB Endowment*, vol. 11, no. 11, pp. 1454–1467, Jul. 2018, doi: 10.14778/3236187.3236198.
[45] R. Cappelli, M. Ferrara, and D. Maltoni, "Fingerprint indexing based on minutia cylinder-code," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 1051–1057, May 2011, doi: 10.1109/TPAMI.2010.228.
[46] L. Qi, C. Hu, X. Zhang, M. R. Khosravi, S. Sharma, S. Pang, and T. Wang, "Privacy-aware data fusion and prediction with spatial–temporal context for smart city industrial environment," *IEEE Trans. Ind. Informat.*, vol. 17, no. 6, pp. 4159–4167, Jun. 2021, doi: 10.1109/TII.2020.3012157.

JIA LIU is currently pursuing the Ph.D. degree in artificial intelligence. She is an Associate Professor with the Weifang Institute of Technology. Her current research interests include information security, machine learning, and network security.

WANG YINCHAI received the Ph.D. degree from Universiti Sains Malaysia. He is currently a Professor with Universiti Malaysia Sarawak (UNIMAS), where he is also the Director of the Tourism Innovation Center. His current research interests include AI, image processing, and network security.

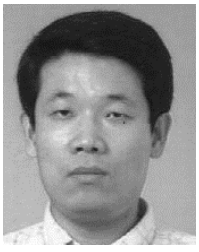
FENGRUI WEI received the master's degree from Wuhan Textile University. She is currently an Associate Professor with the Weifang Institute of Technology, where she is also the Vice Dean of the Faculty of Big Data. Her current research interests include AI and big data analysis.

QING HAN is currently pursuing the master's degree in network security from Shandong University. His current research interests include network security and secure multiparty computation.

YUNTING TAO is currently pursuing the master's degree in software engineering from Shandong University. His main research interests include cryptography, homomorphic encryption, and wireless network security.

LIPING ZHAO is currently an Associate Professor with the Weifang Institute of Technology, where he is also the Director of the Big Data Teaching and Research Section. His current research interests include big data analysis and mining, and programming methods.

XINJIN LI received the master's degree from Zhejiang Normal University. He is currently an Associate Professor with the Weifang Institute of Technology. His current research interests include big data analysis and mining, and programming methods.



HONGBO SUN received the Ph.D. degree from Tsinghua University. He is currently an Associate Professor with the Weifang Institute of Technology and Yantai University, where he is also the Dean of the Faculty of Big Data. His current research interests include mass intelligence, inspection robots, and machine vision.