IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 22, 2018, accepted December 14, 2018, date of publication December 21, 2018,
date of current version January 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2888564

Efficient Authentication of Multi-Dimensional Top-*k* Queries

XIAOYU ZHU[1,2], JIE WU[2], (Fellow, IEEE), WEI CHANG[3], GUOJUN WANG[4], (Member, IEEE), AND QIN LIU[5]

[1]School of Information Science and Engineering, Central South University, Changsha 410083, China
[2]Center for Networked Computing, Temple University, Philadelphia, PA 19122, USA
[3]Department of Computer Science, Saint Joseph's University, Philadelphia, PA 19131, USA
[4]School of Computer Science and Technology, Guangzhou University, Guangzhou 510006, China
[5]College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Corresponding author: Guojun Wang (csgjwang@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61632009 and Grant 61472451, in part by the Guangdong Provincial Natural Science Foundation under Grant 2017A030308006, in part by the High-Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01, in part by NSF under Grant CNS 1629746, Grant CNS 1564128, Grant CNS 1449860, Grant CNS 1461932, Grant CNS 1460971, and Grant CNS 1439672, in part by the China Scholarship Council under Grant 201606370141, and in part by the Hunan Provincial Natural Science Foundation under Grant 2017JJ2333.

ABSTRACT With the increasing popularity of cloud computing, query outsourcing services are becoming widely available for many business applications. However, the third-party cloud server which provides query service is untrusted, and thus the correctness of query results needs to be authenticated by the users. Suppose there is a database where each record has multiple attributes, users submit multi-dimensional top-*k* queries to retrieve *k* records whose outputs with user-supplied scoring function are among the top *k*. Multi-dimensional top-*k* query is widely used in real applications, such as information retrieval, decision making, and disease prediction. Unfortunately, the traditional query authentication methods cannot be directly deployed on multi-dimensional top-*k* query, thus it is still a challenging problem to authenticate multi-dimensional top-*k* query results. We first propose an authentication solution to support multi-dimensional top-*k* query based on signature chain. By using signature chain for each record and its successors on each dimension, our solution allows users to efficiently verify the soundness and completeness of the multi-dimensional top-*k* query results. In addition, we propose an extended solution using larger grid size in order to decrease the overhead in the data owner side in sparse data distribution. The security analysis shows that our multi-dimensional top-*k* query authentication solutions are secure. Through theoretical analysis and simulation, we demonstrate the effectiveness and efficiency of our proposed solution.

INDEX TERMS Data outsourcing, multi-dimension, query authentication, signature chain, top-*k* query.

I. INTRODUCTION

Recently, due to the proliferation of cloud computing, many companies and institutions choose to outsource their database to cloud servers. It allows a data owner to delegate the maintenance and administration of his database to a powerful third-party server. Users access the database by contacting the server instead of the owner, such as ask SQL-like queries to Google's BigQuery or Oracle Cloud to analyze big datasets. This model is applicable to a wide range of computing platforms, including cloud computing [1], edge computing [2], and etc.

Motivating Example: Consider a hospital that wishes to outsource its diabetic patients' records to the cloud. Suppose there is a sample diabetic dataset with attributes age, weight and bs (blood sugar) as shown in Figure 1, the three-dimensional dataset is denoted as *diabetic (age, weight, bs)*. Consider two doctors Alice and Bob who want to find diabetics whose risks of serious complications are among top *k*; they can define different scoring functions according to different evaluation standards. For example, if Alice considers bs important in the evaluation standard, the scoring function can be $Score = age + 2weight + 8bs$, where *age*, *weight*, *bs* are the diabetic's corresponding attribute values. If Bob thinks weight is more important, the scoring function can be $Score = age + 5weight^2 + 3bs$.

However, database outsourcing poses the challenge that the server may be untrusted. The server may return incorrect results for a variety of reasons. For example, the server may

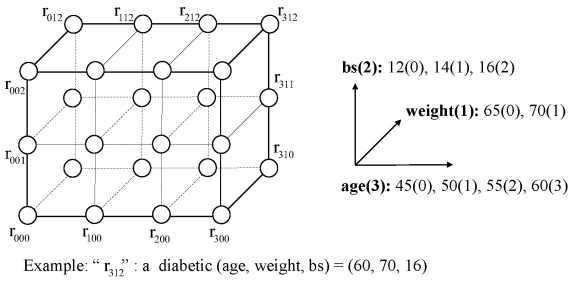


FIGURE 1. An example three-dimensional dataset.

manipulate data due to possible virus or even be controlled by an outside attacker. It may also return incomplete results in order to save computational resources. Hence, it is vital to provide users the service to authenticate the query results, regarding whether each result appears in the original dataset (soundness) or whether the query results include all records which satisfy the query condition in the original dataset (completeness).

The problem of authenticating the query results has been studied a lot in the past decades by a large number of studies [3]–[11]. Devanbu *et al.* [3] presented a solution to authenticate range query for the first time. The data records are sorted by the data owner, and a Merkle Hash tree (MH-tree) [12] is built based on these records. The signature chain method was proposed by Pang *et al.* [8], in which the data owner outsources the records and signatures to the server, and the query results are returned with a Verification Object (*VO*) to the user. A lot of query authentication solutions are inspired by the MH-tree and signature chain methods, such as top-*k* query [13], [14], range query [15], kNN query [16]–[18], skyline query [19], [20], spatial query [21], and aggregation query [22].

Recently, Yang *et al.* [23] proposed a function query authentication method, which is similar to the multi-dimensional top-*k* query. But their solution has several major limitations. Firstly, the scoring function definition is submitted by the data owner rather than the user, which limits users' query preference. Secondly, for three different kinds of function definitions, the data owner needs to construct different signature mesh, and the communication and computation cost of the signature construction process is very large, which will bring a lot of burden to the data owner who is generally equipped with limited resources. Thirdly, the dataset update process is not efficient, when the data owner wants to add or delete one data record, it needs to recompute all the signatures, which will lead to inefficiency in real world application. Finally is that their solution did not take the query process into consideration.

In this paper, we study multi-dimensional top-*k* query authentication problem. We assume that a dataset contains multiple dimensions/attributes. A multi-dimensional top-*k* query obtains data records whose scores rank among top *k*, where the ranking can be the sum of arbitrary composition of attributes, each attribute can be assigned a weight and have a degree. Multi-dimensional top-*k* query has many important applications, including information retrieval in cloud computing [24], resource allocation [25], disease prediction [26], [27], system monitoring [28], and etc.

As multi-dimensional top-*k* queries submitted by users are unpredictable, the possible query results are very large and cannot be all signed. To solve this problem, we propose a multiple signature chain method to authenticate multi-dimensional top-*k* queries. We further propose an extended solution to reduce the computation and communication costs for the owner side. To date, authenticating multi-dimensional top-*k* query efficiently remains a challenging problem; we want to enable users to submit queries according to their query preference and authenticate query results in an efficient way. In summary, the contributions of our paper are as follows:

- We propose to use grid partition and multiple signature chain to solve multi-dimensional top-*k* query authentication problem, and our design gives users great query flexibility; the query function definition can be submitted by the users instead of the data owner.
- We propose an efficient authentication solution and we develop a set of algorithms for multi-dimensional top-*k* query authentication. Our solution supports efficient signature construction process, and each record is chained with its successors on each dimension; the communication and computation costs are reduced due to the signature construction design.
- Extension of our authentication solution for multi-dimensional top-*k* queries in sparse data distribution to decrease the overhead on the data owner side. Our solution supports efficient data update by the data owner, which only needs to modify several signatures for the update operation.
- The security analysis show that our solution can achieve the security goals. We give the complexity analysis and conduct extensive experiments, which shows the effectiveness and efficiency of our solution.

The rest of the paper is organized as follows: We summarize the features of related work in Section II. In Section III, we introduce the system model, adversary model and security goal. Section IV gives the overview of our solution, and we describe the details of our multi-dimensional top-*k* query authentication solution in Section V. Following in Section VI, we extend our solution to a larger grid solution. Section VII extends two-dimensional solutions to multi-dimensional solutions. Section VIII presents the security analysis, the performance analysis and simulation results. Finally, Section IX concludes our paper.

II. RELATED WORK

There are two kinds of technique which have been widely used in query authentication, MH-tree-based solutions [3]–[7] and signature chain-based solutions [8]–[11]. In MH-tree-based solution [3], Figure 2 shows a MH-tree built

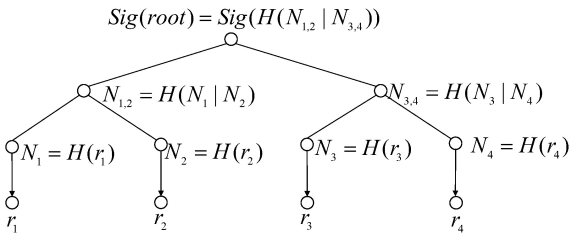


FIGURE 2. Merkle hash tree.

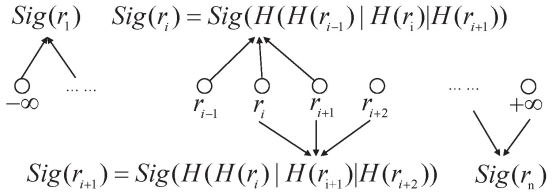


FIGURE 3. Signature chain.

from 4 sorted data records, $r_1 \leq r_2 \leq r_3 \leq r_4$, here $H(\cdot)$ is a hash function and "|" denotes node concatenation. The data owner signs the root node using its private key. The server returns the query results and a *VO* in response to a query. The user uses *VO* to reconstruct the digest and compare it with the root digest. The results are sound and complete if two digests are the same.

In signature chain-based solutions [8], data records are sorted and a signature is created for each data record; the signature of each data record is created based on its digest and the digests of its immediate predecessor and successor. Figure 3 shows a signature chain. Given a set of sorted query results $R = \{r_i < \cdots < r_j\}$, the *VO* contains the signatures of results, as well as the predecessor r_{i-1} and successor r_{j+1}. The signatures form a signature chain to prove that for any two consecutive records r_k and r_{k+1} in R, no record r_x exists in the original dataset such that $r_k < r_x < r_{k+1}$.

Cheng *et al.* [29] proposed a multi-dimensional query authentication solution using signature chain, but their solution only supports range query. Tsou *et al.* [30] proposed to authenticate functional top-k queries in multi-dimensional space, and the users can only launch queries on the conjunction and sum of the attributes. Choi *et al.* [31] proposed to address the problem of authenticating top-k aggregation queries. Su *et al.* [13] put forward a solution to authenticate top-k spatial keyword query. Chen *et al.* [32] proposed to authenticate two-dimensional top-k queries based on R-tree in LBS Services. Wang *et al.* [33], Duan *et al.* [34] proposed solutions to authenticate multiple user-defined spatial queries. However, these methods can not be applied to multi-dimensional top-k query authentication. The closely related work is function query authentication scheme [23], which can be applied to multi-dimensional top-k query. However, the scoring definition is submitted by the data owner rather than the user, which limits the query flexibility, and the data update process's costs are very high. Our work focuses on the multi-dimensional top-k query and proposes efficient query authentication solutions.

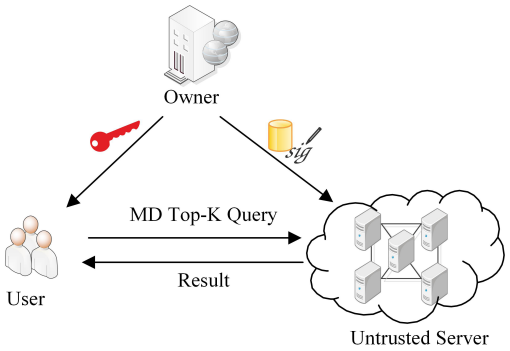


FIGURE 4. System model.

There are some other researchers focused on personalized query [35], user authentication [36]–[38], location-based query [39]–[41], and etc. And there are some works studied the authentication protocols used in business intelligent solutions, such as LDAP [42], NTLM [43], Essbase [44], Cognos [45], and etc. However, these methods are different from our settings.

III. PROBLEM FORMULATION

In this section, we first introduce the system model, then we introduce the adversary model and security goal.

A. SYSTEM MODEL

Figure 4 shows the system model, which involves three types of parties: data owner, server, and data user. The general setting works as follows: First, the data owner makes some pre-computation on the dataset to construct data structure (such as signature chain) that supports query authentication and computes the dataset's signatures S. Second, the data owner distributes the dataset and their signatures to the server. Third, a user sends a multi-dimensional top-k query Q to the server. The server computes the results R, a verification object *VO*, and both of them are sent to the user. Finally, the user verifies the soundness and completeness of the results R.

B. ADVERSARY MODEL AND SECURITY GOAL

The dataset D has d dimensions (each dimension is a numerical attribute). To query data records over D, a user provides a multi-dimensional top-k query, which includes a scoring function $Score(r)$ and a filter condition k. The query is defined below.

A **Multi-dimensional top-k query** $Q = \{Score(r), k\}$ retrieves data records whose ranking scores are among the k smallest. For each d-dimensional data record $r \in D$, the ranking score of this data record can be calculated using the scoring function. The scoring function $Score(r)$ is an aggregate function of the record's attribute values, and we assume each term in the scoring function is positive. Each attribute can have a degree, the degree of the dimension can be

TABLE 1. Summary of notations.

Notation	Description
pk, sk	The owner's public key, private key
d	Number of dimensions
T	A set of d-dimensional data records
D	A new dataset of records
B	Boundaries for dataset D
n	Number of data records in D
S	A set of signatures
k	Number of query results
Q	A multi-dimensional top-k query
R	Query results
VO	Verification object

up to m. For example, the scoring function can be $Score(r) = \text{age}+3\text{weight}+5\text{bs}^2$, where *age*, *weight* and *bs* are the record r's attribute values, the degree of the dimension m is 2.

Our security goal is to offer approaches for authenticating multi-dimensional top-k queries. In our setting, we assume that the server is untrusted and may present user a tampered result. Our proposed solutions can allow the user to verify the soundness and completeness of the query results.

Soundness: The user can verify that all qualifying data records returned are correct. The results are all from the original dataset.

Completeness: The user can verify that the results cover all the qualifying data records. The data records satisfying the query condition are all included in the results, and the number of the data records is at least k.

IV. OVERVIEW

In this section, we will introduce the general design, scheme outline and cryptographic primitives. For references, the notations are showed in Table 1.

A. GENERAL DESIGN

Potentially, there are several ways to authenticate multi-dimensional top-k query results, which lead to different computing and storage costs for the user, server, and owner. One naive solution, for example, would be for the server to simply return the entire dataset and their signatures, and let the user do his own multi-dimensional top-k query and authenticate results locally. If the dataset is very large, this solution entails huge data transmission costs and requires a lot of storage and processing by the user. Another potential solution is for the owner to pre-compute and sign a whole range of results to possible queries. However, this solution is not practical in general; there are simply too many possible queries the user might want to ask.

We propose a solution to solve the multi-dimensional top-k query authentication problem. Our solution is a compromise design in which the user does not need to receive and compute the entire dataset locally, nor does the owner need to pre-compute the signatures of arbitrary query results. We propose two solutions based on different grid sizes. In the basic solution, the dataset is split into small grids, the grid size is small so that each grid has no more than one data, then we add dummy data to the grid that does not have a data. The user receives exactly the top-k data records, but he needs to compute extra signatures for dummy data. Then, we propose an extended solution, which extends the grid size so that each grid has at least one data record. Next, we will handle multiple data records in the same grid. When the data is in sparse distribution, the data owner needs to generate many dummy data. Thus we propose an extended solution to decrease the overhead on the data owner side. In the extended solution, the user computes fewer signatures as there are no extra dummy data, but the user receives more than k results and needs to query locally.

B. SCHEME OUTLINE

Our scheme includes five algorithms as follows:

- $Init(T) \rightarrow D$: This algorithm takes a dataset T as input and outputs a new dataset D.
- $KeyGen(k) \rightarrow \{pk, sk\}$: This algorithm takes a security parameter k as input and outputs a public key pk and a private key sk.
- $SigGen(sk, D) \rightarrow S$: This algorithm takes private key sk and dataset D as inputs and outputs a set of signatures S.
- $GenProof(Q, D, S) \rightarrow \{R, VO\}$: This algorithm takes query Q, dataset D, and signatures S as inputs and outputs query results R and a verification object VO.
- $Verify(pk, R, VO) \rightarrow \{0, 1\}$: This algorithm takes public key pk, query results R, a verification object VO as inputs, and outputs 0 if verification fails; otherwise, the output equals 1.

From a systematic point of view, our scheme works as follows:

Initialization phase. The data owner runs the *Init* algorithm to generate D, and runs *KeyGen* algorithm to generate public key pk and private key sk. The public key pk is shared with the user, the private key sk is kept secret.

Signature phase. The data owner runs the *SigGen* algorithm to generate a set of signatures S for data records in D. Then, she uploads D and S to the server.

Query phase. The server first receives a multi-dimensional top-k query Q from the user, then the server outputs all data records matching Q into the query results R.

Verification object phase. The server runs the *GenProof* algorithm to generate a verification object VO. Finally, the server returns query results R and a verification object VO to the user.

Verification phase. The user runs *Verify* algorithm to check if the query results are correct or not.

C. CRYPTOGRAPHIC PRIMITIVES

One-way hash function. The one-way hash function is used to generate fixed-length digest for data records, and we utilize secure hash algorithm SHA-1 in this paper.

Digital signature. The digital signature is used to authenticate the integrity and origin of the data records, and we utilize RSA signature scheme to generate signatures.

Signature chain. Inspired by the signature chain, we propose a signature chain on multiple dimensions to enable authenticating the completeness of multi-dimensional top-k query results. In our scheme, we calculate the signature for each data record which has multiple dimensions. A data record's signature is composed of its own hash value and its successors' hash values over multiple dimensions.

V. BASIC SOLUTION

We start from the simplest case, two-dimensional top-k query authentication. In the following subsections, we propose a basic solution for two-dimensional top-k query authentication. We discuss how to create signature chains for two-dimensional dataset and how to verify query results using signature chains.

A. SIGNATURE CHAINS CONSTRUCTION

The signature chain construction contains three steps. First, initiate the dataset by partitioning grids and adding dummy data. Second, generate boundaries for the new dataset. Third, generate signatures for the new dataset.

Grid. The original dataset T can be sorted in an increasing order for x and y dimension. The data record is expressed as $r_{ij} = (x_i, y_j)$, where x_i and y_j are the attribute values on the x and y dimension, respectively. Some data records have two successors, one in the x dimension, another in the y dimension. For example, the successors of r_{11} in x and y dimension are r_{21} and r_{12}, respectively. This allows us to create two signature chains over the x and y dimensions for a two-dimensional data record. As there are some data records that do not have a successor, such as r_{22} in the y dimension, we need to add some dummy data and boundaries in order to guarantee that every record has two successors.

The dataset T is partitioned into grids according to the x and y dimension's attribute values; some grids have one record, and some do not have a record. The owner generates a dummy data for each empty grid, and the dummy data is assigned the corresponding grid's axes values. The new dataset D contains the original data T and generated dummy data. For example, in Figure 5(a), the dataset T is partitioned into 4×4 grids according to the x and y axis values; a dummy data $r_{24} = (x_2, y_4, dummy)$ is generated for the empty grid (x_2, y_4), where (x_2, y_4) are the axes values of this grid, and *dummy* is a flag to show that it's a dummy data created by the data owner. The new dataset is $D = \{r_{11}, \cdots, r_{44}\}$. Note that we use the example dataset in Figure 5(a) in our following illustrations.

Boundary. Then, the owner generates boundaries for the new dataset, each record should be bounded by two records. The new dataset is denoted as $D = \{r_{ij} | 1 \le i \le p, 1 \le j \le q\}$, where p is the number of attributes for the x dimension, q is the number of attributes for the y dimension. The owner first generates two values $(x_0 = -\infty, x_{p+1} = \infty)$ for the x axis, then generates two values $(y_0 = -\infty, y_{q+1} = \infty)$ for the y axis. The owner generates a set of lower boundaries B_l and upper boundaries B_u for the new dataset D. The boundaries are generated as follows:

- For $1 \le i \le p$, generate $r_{i0} = (x_i, y_0)$ and put it into B_l. For $1 \le j \le q$, generate $r_{0j} = (x_0, y_j)$ and put it into B_l.
- For $0 \le i \le p$, generate $r_{i,q+1} = (x_i, y_{q+1})$ and put it into B_u; For $0 \le j \le q$, generate $r_{p+1,j} = (x_{p+1}, y_j)$ and put it into B_u.
- The boundaries for D can be expressed as $B = \{B_l, B_u\}$.

Figure 5(b) shows an example of creating boundaries. First, generate (x_0, x_5), (y_0, y_5). Second, generate $B_l = \{r_{10}, r_{20}, r_{30}, r_{40}, r_{01}, r_{02}, r_{03}, r_{04}\}$ and $B_u = \{r_{05}, r_{15}, r_{25}, r_{35}, r_{45}, r_{50}, r_{51}, r_{52}, r_{53}, r_{54}\}$. Finally, put B_l and B_u into B.

Signatures. Given a set of n two-dimensional data records $D = \{r_{ij} | 1 \le i \le p, 1 \le j \le q\}$, each record can be sorted in two lists. Assuming that the order increases, we have $r_{i1} < r_{i2} < \cdots < r_{iq}$ for $1 \le i \le p$ in x dimension. Meanwhile, we have $r_{1j} < r_{2j} < \cdots < r_{pj}$ for $1 \le j \le q$ in y dimension.

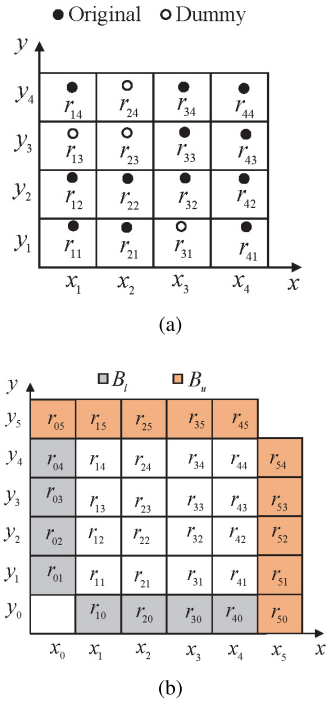


FIGURE 5. Dataset initiation. (a) Grid. (b) Boundary.

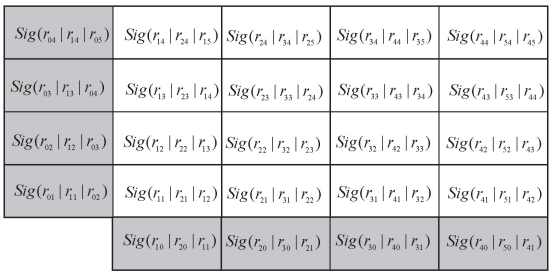


FIGURE 6. Signatures construction process.

Algorithm 1 Two-Dimensional Top-k Query

Input: Query $Q = \{Score(r), k\}$, dataset D.
Output: Results R.

```
Set cnt = 0, i = 1, j = 1 initially
while cnt < k do
    Find the record r_ij in D and add it into results R
    Add r_{i+1,j} and r_{i,j+1} into candidate results C
    Calculate Score(r) for each record r in C
    Find the next record r_{i'j'} having minimum score
    Remove r_{i'j'} from C
    Set i = i', j = j', cnt = cnt + 1
end while
return R
```

On these sorted lists, each record should be chained in two dimensions, the owner builds signature chains as follows: For each data record $r_{ij} \in D \cup B_l$, r_{ij} has a successor $r_{i+1,j}$ in x dimension and a successor $r_{i,j+1}$ in y dimension. The data owner creates the signature for r_{ij} as follows:

$$Sig(r_{ij}|r_{i+1,j}|r_{i,j+1}) = Sig(H(H(r_{ij})|H(r_{i+1,j})|H(r_{i,j+1}))) \tag{1}$$

Here, $H(\cdot)$ is a hash function (e.g., SHA1), and Sig is a signature generation algorithm (e.g., RSA). Through verifying this signature, it proves that there exists no record between r_{ij} and $r_{i+1,j}$ in the x dimension; and no record between r_{ij} and $r_{i,j+1}$ in y dimension. In other words, r_{ij} and $r_{i+1,j}$ are contiguous in the x dimension, r_{ij} and $r_{i,j+1}$ are contiguous in the y dimension. The whole set of signatures for dataset D is defined as S. Then, the owner sends the dataset D, the boundaries B, and signatures S to the server.

In the approach above, the owner creates $q + 1$ signature chains in the x dimension, $p + 1$ signature chains in the y dimension. The total number of signatures is equal to the number of records in $D \cup B_l$. Figure 6 shows the signatures created for dataset D; we take $Sig(r_{11}|r_{21}|r_{12}) = Sig(H(H(r_{11})|H(r_{21})|H(r_{12})))$ as an example, r_{11}'s successor in x dimension is r_{21}, its successor in y dimension is r_{12}, so r_{11}'s signature concatenates itself and its two successors.

B. QUERY RESULT VERIFICATION

In this subsection, we introduce the query result verification process. In the beginning, the server executes a two-dimensional top-k query. Then, the server generates a VO for query results R. Next, the user verifies the soundness and completeness of R. Finally, the owner can update the dataset efficiently.

Query. The user submits a two-dimensional top-k query Q to the server, then the server outputs the query results R. The server finds one record r_{ij}, which has the minimum score in each round for k times. r_{ij} is set as r_{11} initially, and the server puts r_{ij}'s successor $r_{i+1,j}$ in the x dimension and $r_{i,j+1}$ in the y dimension into candidate results C. Then, the server continues to find the next result $r_{i'j'}$, which has the minimum score in C until finding all the results. The details about the query process above is given in Algorithm 1.

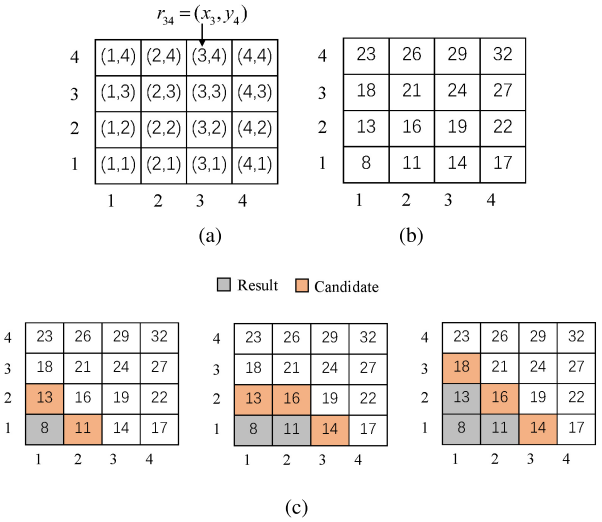


FIGURE 7. An example of two-dimensional top-3 query. (a) Data record. (b) $Score(r_{ij}) = 3x_i + 5y_j$. (c) Query process ($k = 3$).

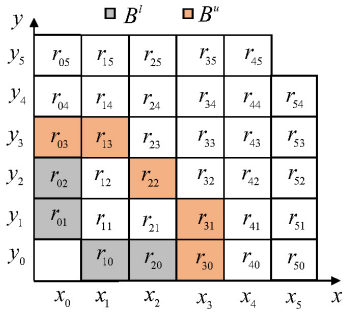


FIGURE 8. VO construction.

Figure 7 shows an example of a two-dimensional top-k query. The data record is showed in Figure 7(a); each record is represented as $r_{ij} = \{x_i, y_j\}$, Figure 7(b) shows each record's score using the scoring function $Score(r_{ij}) = 3x_i + 5y_j$. Figure 7(c) shows the query process to find the top 3 smallest records. The server first sets r_{11} as the first result, then continues to find the smaller one r_{21} as the next result from r_{21} and r_{12}, finally the server finds the query results $R = \{r_{11}, r_{21}, r_{12}\}$.

Verification object. Let R be the query results; the server generates a verification object VO for R as follows. First, find the maximum i and j in $R = \{r_{ij}\}$ and denote them as p and q respectively. Put $\{r_{i0}|1 \le i \le p\}$ and $\{r_{0j}|1 \le j \le q\}$ into lower boundaries B^l. Second, for each data record r_{ij} in $R \cup B^l$, if its successors $r_{i+1,j}$ or $r_{i,j+1}$ is not found in $R \cup B^l \cup B^u$, then add it into upper boundaries B^u.

Then for each data record r_{ij} in results R and lower boundaries B^l, the server finds its corresponding signature $Sig(r_{ij}|r_{i+1,j}|r_{i,j+1})$ and puts it into S_q. Finally, the server returns the verification object $VO = \{B^l, B^u, S_q\}$ to the

Algorithm 2 Construct Verification Object *VO*

Input: Results R, dataset D, signatures S.
Output: Verification object VO.

Find maximum i in $R = \{r_{ij}\}$ and denote it as p
Find maximum j in $R = \{r_{ij}\}$ and denote it as q
for $1 \leq i \leq p$ **do**
  put r_{i0} into lower boundaries B^l
end for
for $1 \leq j \leq q$ **do**
  put r_{0j} into lower boundaries B^l
end for
for each $r_{ij} \in R \cup B^l$ **do**
  if $r_{i+1,j}$ or $r_{i,j+1}$ is not exist in $R \cup B^u$ **then**
    put $r_{i+1,j}$ or $r_{i,j+1}$ into upper boundaries B^u
  end if
end for
for each $r_{ij} \in R \cup B^l$ **do**
  Find $Sig(r_{ij}|r_{i+1,j}|r_{i,j+1})$ in S and put it into S_q
end for
$VO \leftarrow \{B^l, B^u, S_q\}$
Return VO

Algorithm 3 Query Result Verification

Input: Query $Q = \{Score(r), k\}$, results R, verification object $VO = \{B^l, B^u, S_q\}$.
Output: 0 or 1.

for each record $r_{ij} \in \{R \cup B^l\}$ **do**
  Find its corresponding signature in S_q
  Find $r_{i+1,j}$ and $r_{i,j+1}$ in $R \cup B^l \cup B^u$
  if Equation (2) is not satisfied **then**
    Return 0
  end if
end for
if the number of data in R is not equal to k **then**
  Return 0
end if
Set the maximum score of results in R as $Score_m$
for each r_{ij} in B^u **do**
  if $Score(r_{ij}) > Score_m$ **then**
    Return 0
  end if
end for
Return 1

user. The details of the above verification object construction process can be found in Algorithm 2.

For example, in Figure 8, first set $p = 2$, $q = 2$, next put $\{r_{10}, r_{20}, r_{01}, r_{02}\}$ into lower boundaries, put $\{r_{03}, r_{13}, r_{22}, r_{31}, r_{30}\}$ into upper boundaries, then find the signatures of $R = \{r_{11}, r_{12}, r_{21}\}$ and $B^l = \{r_{10}, r_{20}, r_{01}, r_{02}\}$ in S, and put them into S_q.

Verification. On receiving the query results R and verification object VO from the server, the user verifies the soundness and completeness of the query results as follows. For each data record $r_{ij} \in \{R \cup B^l\}$, the user finds its signature $Sig(r_{ij}|r_{i+1,j}|r_{i,j+1})$ in S_q, and finds its successors $r_{i+1,j}$ and $r_{i,j+1}$ in $R \cup B^l \cup B^u$. Then the user checks if the following equation holds:

$$Sig^{-1}(Sig(r_{ij}|r_{i+1,j}|r_{i,j+1}), pk) = H(H(r_{ij})|H(r_{i+1,j})|H(r_{i,j+1})) \quad (2)$$

where Sig^{-1} is the signature verification algorithm with the owner's public key pk.

If the check is passed, it means r_{ij}, $r_{i+1,j}$ and $r_{i,j+1}$ are in the original order, and there is no data record in between which satisfies the query condition. Then, the user checks whether the number of original data in result is equal to the filter condition k. Finally, the user proceeds to check the completeness of boundary records by verifying if the boundary's score is larger than the highest score of R. If any of them is larger than the highest score, the check is failed. If any of the check is failed, the user outputs 0. Otherwise, he outputs 1. A more formal description of the above verification process is given in Algorithm 3.

For example, the query results are $R = \{r_{11}, r_{21}, r_{12}\}$, the lower boundaries are $B^l = \{r_{10}, r_{20}, r_{01}, r_{02}\}$, the upper boundaries are $B^u = \{r_{03}, r_{13}, r_{22}, r_{31}, r_{30}\}$. First check the signatures of $\{r_{11}, r_{21}, r_{12}, r_{10}, r_{20}, r_{01}, r_{02}\} \in \{R \cup B^l\}$. Then check whether the number of query results is 3. Finally check whether the score of $\{r_{03}, r_{13}, r_{22}, r_{31}, r_{30}\} \in B^u$ is larger than the maximum score $Score(r_{12}) = 13$.

Data update. The data update process of our solution is simple. If the data owner modifies a data record, she first finds the data record's predecessors in each dimension, then generates a new signature for each predecessor. For example, in Figure 9, if the owner wants to modify r_{33}, she first finds r_{33}'s predecessors r_{23} and r_{32}, then she generates signatures for $\{r_{33}, r_{23}, r_{32}\}$. If the data owner deletes a data record, she replaces the original record with a dummy data. If the data owner inserts a new data record, she first generates dummy data for the new attributes, then she finds the new record and dummy data's predecessors in each dimension, finally she generates signatures for these dummy data and predecessors. For example, in Figure 9, if the owner wants to insert r_{ab}, she first generates dummy data $\{r_{a1}, r_{a2}, r_{a3}, r_{a4}, r_{1b}, r_{2b}, r_{3b}, r_{4b}\}$ for the new attributes x_a and y_b. The predecessors of the dummy data are $\{r_{11}, r_{12}, r_{13}, r_{14}, r_{21}, r_{31}, r_{41}\}$, then the owner generates new signatures for the dummy data and its predecessors.

VI. EXTENDED SOLUTION

In the above solution, the owner needs to add dummy data, which will enlarge the dataset greatly when the records are in sparse distribution. In order to solve this problem, we propose another solution that extends the grid size to a larger one, which will handle multiple data records in the same grid.

When the server finds grids which contain top k data records, the grids may totally contain more than k data records. Then after the server calculates the verification

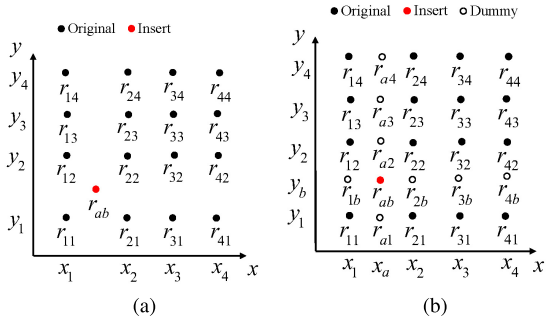


FIGURE 9. The data update process. (a) Data insert. (b) Dummy data.

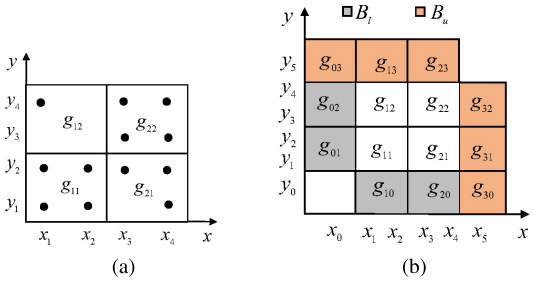


FIGURE 10. Dataset initiation into larger grid. (a) Larger grid. (b) Boundary.

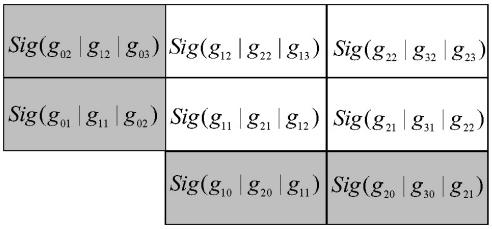


FIGURE 11. Signatures construction process for larger grid.

object for the query results, the verification object contain the boundary grids and the signatures of result grids. Then the server returns the results and verification object to the user. The user needs to re-query the top-k locally based on the query result, then he continues the verification process.

A. SIGNATURE CHAINS CONSTRUCTION

Grid. We now consider partitioning the data space into larger grids. The partition contains two requirements: 1) each grid should have more than one data record. 2) the grid size is partitioned into equal sizes for simplicity. The partition method can be changed for improvements. In this way, the owner does not need to add dummy data. Figure 10(a) shows an example of partitioning the data records into larger grids, and the records are partitioned into four grids $\{g_{11}, g_{12}, g_{21}, g_{22}\}$, each grid may contain multiple records. The new dataset is denoted as $D = \{g_{ij} | 1 \leq i \leq p, 1 \leq j \leq p\}$, where p is the number of attributes for the x and y dimension. Suppose grid g_{ij} consists of l records, defined as $g_{ij} = \{r_1, \cdots, r_l\}$.

Boundary. The owner generates boundaries for the new dataset D, and each record should be bounded by two records. The owner first generates two values $(x_0 = -\infty, x_{p+1} = \infty)$ for the x axis, then generates two values $(y_0 = -\infty, y_{p+1} = \infty)$ for the y axis. The owner generates a set of lower boundaries B_l and upper boundaries B_u for D the same as the basic solution. Next, the owner assigns values to these boundaries. Note that some boundary grids contain several attribute values in one dimension; the owner assigns the largest attribute value to these grids.

Figure 10(b) shows how to generate boundaries for dataset. First, generate (x_0, x_5), (y_0, y_5). Then, generate $B_l = \{g_{10}, g_{20}, g_{01}, g_{02}\}$ and $B_u = \{g_{03}, g_{13}, g_{23}, g_{30}, g_{31}, g_{32}\}$. Next, assign values to these boundaries, e.g., grid g_{10} has two x axis values x_1, x_2. Assign the largest one x_2 to the grid, so $g_{10} = (x_2, y_0)$. Finally, put B_l and B_u into B.

Signature. Suppose grid g_{ij} consists of l records, defined as $g_{ij} = \{r_1, \cdots, r_l\}$, we have $r_1 \leq \cdots \leq r_l$ for the x axis. The owner calculates the hash value $H(g_{ij})$ for each grid g_{ij} by concatenating its l records:

$$H(g_{ij}) = H(H(r_1)|H(r_2)|\cdots|H(r_l)) \tag{3}$$

The data owner creates the signature of g_{ij} by chaining it with its two successors:

$$Sig(g_{ij}|g_{i+1,j}|g_{i,j+1}) = Sig(H(H(g_{ij})|H(g_{i+1,j})|H(g_{i,j+1}))) \tag{4}$$

Figure 11 shows the created signatures; we take $Sig(g_{11}|g_{21}|g_{12}) = Sig(H(H(g_{11})|H(g_{21})|H(g_{12})))$ as an example, g_{11}'s successors in x and y dimension are g_{21} and g_{12}, respectively, so g_{11}'s signature concatenates g_{11}'s digest with g_{21}'s digest and g_{12}'s digest. Grid g_{11} contains four records $\{r_{11}, r_{12}, r_{21}, r_{22}\}$, so $H(g_{11}) = H(H(r_{11})|H(r_{12})|H(r_{21})|H(r_{22}))$.

B. QUERY RESULT VERIFICATION

Query. The user submits a multi-dimensional top-k query Q to the server. For each grid in the dataset D, the server calculates the score for each data in the grid, then the server finds several grids which contain at least k data records. The server will put all the grids matching into the query results R. For example, the user asks for top 3 smallest records, Figure 12(a) shows each record's score using the scoring function $Score(r_{ij}) = 3x_i + 5y_j$. Figure 12(b) shows the query results R which contains grid $g_{11} = \{r_{11}, r_{12}, r_{21}, r_{22}\}$.

Verification object. Let R be the query results; the server generates a verification object VO for R as follows. First, find the maximum i and j in $R = \{g_{ij}\}$ and denote them as p and q respectively. Put $\{g_{i0} | 1 \leq i \leq p\}$ and $\{g_{0j} | 1 \leq j \leq q\}$ into lower boundaries B^l. Second, for each data record g_{ij} in $R \cup B^l$, if its successors $g_{i+1,j}$ or $g_{i,j+1}$ is not found in $R \cup B^l \cup B^u$, then add it in upper boundaries B^u. Then for each data record g_{ij} in results R and lower boundaries B^l, the server finds its corresponding signature $Sig(g_{ij}|g_{i+1,j}|g_{i,j+1})$ and puts it into

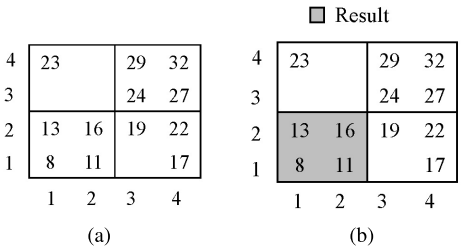


FIGURE 12. Two-dimensional top-k query for larger grid. (a) $Score(r_{ij}) = 3x_i + 5y_j$. (b) Query result ($k = 3$).

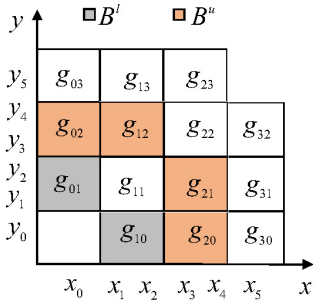


FIGURE 13. VO construction for larger grid.

S_q. Finally, the server returns the verification object $VO = \{B^l, B^u, S_q\}$ to the user. For example, in Figure 13, first set $p = 1$, $q = 1$, next put $\{g_{10}, g_{01}\}$ into lower boundaries, put $\{g_{02}, g_{12}, g_{21}, g_{20}\}$ into upper boundaries, then find the signatures of $R = \{g_{11}\}$ and $B^l = \{g_{10}, g_{01}\}$ in S, and put them into S_q.

Verification. On receiving the query results R and verification object VO from the server, the user verifies the soundness and completeness of the query results as follows. For each grid $g_{ij} \in \{R \cup B^l\}$, the user finds g_{ij}'s signature $Sig(g_{ij}|g_{i+1,j}|g_{i,j+1})$ in S_q, and find its successors $g_{i+1,j}$ and $g_{i,j+1}$ in $R \cup B^l \cup B^u$. The user calculates the hash value $H(g_{ij})$ for grid g_{ij}:

$$H(g_{ij}) = H(H(r_1)|H(r_2)|\cdots|H(r_l)) \tag{5}$$

Similarly, the user calculates the hash value $H(g_{i+1,j})$ and $H(g_{i,j+1})$ for grid $g_{i+1,j}$ and $g_{i,j+1}$. Next, the user checks if the following equation holds:

$$Sig^{-1}(Sig(g_{ij}|g_{i+1,j}|g_{i,j+1}), pk) = H(H(g_{ij})|H(g_{i+1,j})|H(g_{i,j+1})) \tag{6}$$

If the check is passed, it means g_{ij}, $g_{i+1,j}$ and $g_{i,j+1}$ are in the original order. Then, the user queries the results R and finds the top k query results. Finally, the user proceeds to check the completeness of boundary records by verifying if the boundary's score is larger than the largest score of top k query results. If any of them is larger than the largest score, the check is failed. If any of the check is failed, the user outputs 0. Otherwise, output 1.

For example, the query results are $R = \{g_{11}\}$, the lower boundaries are $B^l = \{g_{10}, g_{01}\}$, the upper boundaries are $B^u = \{g_{02}, g_{12}, g_{21}, g_{20}\}$. The user first checks the signatures of $\{g_{11}, g_{10}, g_{01} \in \{R \cup B^l\}$. Then the user queries R and finds that the final query results are r_{11}, r_{21}, r_{12}, then the user checks whether the number of query results is 3. Finally check whether the score of $\{g_{02}, g_{12}, g_{21}, g_{20}\} \in B^u$ is larger than the maximum score $Score(r_{12}) = 13$.

Data update. When the data owner updates a data record, the data owner first finds its corresponding grid, then the data owner finds the grid's predecessors in each dimension, finally generates new signatures for the corresponding grids and its predecessors. For example, if the owner wants to modify r_{22}, she first finds r_{22}'s corresponding grid g_{11}, then she finds g_{11}'s predecessors g_{01} and g_{10}, finally she generates signatures for $\{g_{11}, g_{01}, g_{10}\}$.

VII. MULTI-DIMENSIONAL SOLUTIONS

In this section, we extend two-dimensional top-k query authentication solutions to multi-dimensional.

A. BASIC SOLUTION FOR MULTI-DIMENSIONAL DATASET

We now considers a multi-dimensional dataset; the process of multi-dimensional top-k query authentication solution is similar to the two-dimensional solution.

Grid. Given a set of data records, each data record has d attributes. The owner first partitions the data space into small grids according to d dimensional attribute values. For each empty grid, the owner assigns it dummy data over d attributes, the dummy data's attribute in each dimension is equal to its corresponding axis value.

Boundary. The new dataset is denoted as $D = \{r_{i,\cdots,j}|r_{1,\cdots,1}, r_{2,\cdots,1}, \cdots, r_{p,\cdots,q}\}$, where p is the number of attributes for the first dimension, q is the number of attributes for the last dimension. First, generate two values ($x_0 = -\infty, x_{p+1} = \infty$) for the first dimension, and so on until generate two values ($y_0 = -\infty, y_{q+1} = \infty$) for the last dimension. The owner generates a set of lower boundaries B_l and upper boundaries B_u for the new dataset D. The boundaries are generated as follows:

- Generate lower boundaries $r_{i,\cdots,0} = (x_i, \cdots, y_0), \cdots, r_{0,\cdots,j} = (x_0, \cdots, y_j)$ for each dimension and put them into B_l.
- Generate upper boundaries $r_{i,\cdots,q+1} = (x_i, \cdots, y_{q+1}), \cdots, r_{p+1,\cdots,j} = (x_{p+1}, \cdots, y_j)$ for each dimension and put them into B_u.
- The boundaries for D can be expressed as $B = \{B_l, B_u\}$.

Signature. Each data record $r_{i,\cdots,j}$ has d successors, a successor $r_{i+1,\cdots,j}$ in the first dimension, a successor in the second dimension, and so on. $r_{i,\cdots,j+1}$ denotes the successor in the last dimension. The data owner creates the signature of $r_{i,\cdots,j}$ as follows:

$$Sig(r_{i,\cdots,j}) = Sig(H(H(r_{i,\cdots,j})|H(r_{i+1,\cdots,j})|\cdots|H(r_{i,\cdots,j+1}))) \tag{7}$$

Query. The user submits a multi-dimensional top-k query Q to the server. The server put the query results satisfying the query conditions in R.

Verification object. Then the server generates the verification object VO for results R. Compute the lower boundaries B^l, upper boundaries B^u, find the signatures S_q, and put them into VO. Finally, the server returns the query results R and a verification object VO to the user.

Verification. On receiving the query results R and verification object VO from the server, the user verifies the soundness and completeness of the query results as follows: For each data record $r_{i,\cdots,j} \in R \cup B^l$, find its signature $Sig(r_{i,\cdots,j})$ in S_q, find its d successors $r_{i+1,\cdots,j}, \cdots, r_{i,\cdots,j+1}$ in $R \cup B^l \cup B^u$. Then, the user checks if the following equation holds:

$$Sig^{-1}(Sig(r_{i,\cdots,j}), pk) = H(H(r_{i,\cdots,j})| = H(r_{i+1,\cdots,j})| \cdots |H(r_{i,\cdots,j+1})) \quad (8)$$

Then, the user proceeds to check the completeness for the boundary records.

B. EXTENDED SOLUTION FOR MULTI-DIMENSIONAL DATASET

Grid. The new dataset is denoted as $D = \{g_{i,\cdots,j} | g_{1,\cdots,1}, \cdots, g_{p,\cdots,p}\}$, where p is the partition size. First, generate two values ($x_0 = -\infty, x_{p+1} = \infty$) for the first dimension, and so on until generate two values ($y_0 = -\infty, y_{p+1} = \infty$) for the last dimension.

Boundary. The owner generates a set of lower boundaries B_l and upper boundaries B_u as the basic solution. Next, the owner assigns values to these boundaries.

Signature. Each grid may contain multiple records; suppose grid $g_{i,\cdots,j}$ consists of l records, defined as $g_{i,\cdots,j} = \{r_1, \cdots, r_l\}$, we have $r_1 < \cdots < r_l$ for the first dimension. The owner calculates the hash value $H(g_{i,\cdots,j})$ for each grid $g_{i,\cdots,j}$ by concatenating its l records:

$$H(g_{i,\cdots,j}) = H(H(r_1)|H(r_2)| \cdots |H(r_l)) \quad (9)$$

The data owner creates the signature of $g_{i,\cdots,j}$ by chaining it with its d successors:

$$Sig(g_{i,\cdots,j}) = Sig(H(H(g_{i,\cdots,j})|H(g_{i+1,\cdots,j})| \cdots |H(g_{i,\cdots,j+1}))) \quad (10)$$

Query. The user submits a multi-dimensional top-k query Q to the server. For each grid in the dataset D, the server calculates the score for each data in the grid, then the server finds several grids that altogether contain at least k data records. The server will put all the grids matching into the query results R.

Verification object. For each query result $g_{i,\cdots,j} \in R$, compute its d successors and add them in VO. Then, add the results' lower boundaries B^l and upper boundaries B^u in VO. Finally, for each grid $g_{i,\cdots,j} \in R \cup B^l$, find the corresponding $Sig(g_{i,\cdots,j})$ in signatures S and add them into VO. Consequently, the server returns the query results R and verification object VO to the user.

Verification. The user verifies the soundness and completeness of the query results as follows: For each grid $g_{i,\cdots,j} \in R \cup B^l$, the user finds its signature $Sig(g_{i,\cdots,j})$ in VO. The user calculates the hash value $H(g_{i,\cdots,j})$:

$$H(g_{i,\cdots,j}) = H(H(r_1)|H(r_2)| \cdots |H(r_l)) \quad (11)$$

The user calculates the hash value $H(g_{i+1,\cdots,j})$, $H(g_{i,\cdots,j+1})$, and so on similarly, then the user checks if the following equation holds:

$$Sig^{-1}(Sig(g_{i,\cdots,j}), pk) = H(H(g_{i,\cdots,j})| H(g_{i+1,\cdots,j})| \cdots |H(g_{i,\cdots,j+1})) \quad (12)$$

Finally, the user proceeds to check the completeness by verifying the boundary's score.

VIII. PERFORMANCE ANALYSIS

In this section, we first give the security analysis, then we analyze the computation and communication costs of our proposed basic solution, finally we present the simulation results compared with other solution.

A. SECURITY ANALYSIS

We first prove that the basic solution can achieve the security goals as follows. Let R be the query results, B be the boundaries.

We first discuss the case in which R is not sound: Some record $r_{i,\cdots,j}$ in R is forged. The adversary creates a fake record $r_{i',\cdots,j'}$ to replace $r_{i,\cdots,j}$ in this case. A signature $Sig(r_{i,\cdots,j})$ should be returned to the user to authenticate the query result. But as the adversary doesn't have the owner's private key, it is impossible for the adversary to forge a correct signature using the forge record $r_{i',\cdots,j'}$.

Then we discuss three cases where R is not complete:

Case 1: At least one initial boundary record is forged. The adversary needs to replace the initial boundaries. In order to make the user accept the forged result, the adversary must forge fake signatures for initial boundaries. It is impossible as the adversary doesn't have the owner's private key.

Case 2: At least one end boundary record is forged. There are two ways to do so. The first one is to remove or add some end records; the user will detect the error and discover that the number of original records in result is not equal to k. The second one is to replace some end records with some other records in the original database. The error will be detected when the user checks the score of the boundary records. It will discover that some boundary record's score is smaller than the query result's maximum score.

Case 3: Two contiguous records in R are not contiguous in the original dataset. This happens when the adversary removes some record from R. Suppose record $r_{i,\cdots,j}$ is removed. In order to avoid being detected, the adversary must forge d signatures $\{Sig(r_{i-1,\cdots,j}), \cdots, Sig(r_{i,\cdots,j-1})\}$ to replace signature $Sig(r_{i,\cdots,j})$. It is impossible as the adversary doesn't have the owner's private key.

Similarly, we can prove that the extended solution can also achieve the security goals.

TABLE 2. Complexity comparison of query authentication solutions.

Comparison	Basic	Extended	FQA
Signature size	$O(n)$	$O(g)$	$O(f^2)$
VO size	$O(k)$	$O(l)$	$O(k)$
Verification cost	$O(k)$	$O(l)$	$O(k)$
Data update	$O(\sqrt[d]{n})$	$O(d)$	$O(f^2)$

B. COMPLEXITY ANALYSIS

In this subsection, we give the complexity analysis of our proposed basic and extended solution from four aspects: signature construction cost, *VO* construction cost, verification cost and data update cost, and compare our solutions with the benchmark method in [23] (denoted by FQA). The complexity comparison between our solutions and FQA is given in Table 2. In our paper, n, d and k denote the number of records, the number of dimensions and the number of results, respectively. For the extended solution, g and l represent the number of partitioned grids and the number of result grids, respectively. For FQA solution, f is the number of functions.

1) SIGNATURE CONSTRUCTION

The computation and communication costs on the data owner side incur in constructing the signatures. The data owner first preprocesses the dataset and generates a new dataset. In our basic solution, each record is chained with its successors on each dimension, only one signature needs to be generated for one record. Suppose there are n data in the new dataset with d dimensions, the data owner needs to create $O(n)$ signatures. The main impact is the the number of data records n, and it's not sensitive to the number of dimensions, as it only needs to execute one more hash operation for each record as the number of dimensions increases by one.

If the dataset is sparse and needs to add many dummy data, the signature construction costs of our basic solution will be large, we can choose the extended solution to solve this problem, where the signature construction costs depend on the grid partition. In our extended solution, suppose the dataset is partitioned into g grids, each grid has one signature, thus the data owner needs to create $O(g)$ signatures. The extended solution can decrease the signature construction costs significantly.

FQA needs to construct $O(f^2)$ signatures, where f is the number of functions. In FQA, each data record is mapped into one function, thus the number of functions is equal to the number of data records. FQA proposed three different techniques for univariate linear functions, multivariate linear function and multivariate high degree function. For three different kinds of function definitions, the data owner needs to construct different signature mesh, and the communication and computation costs of the signature construction process are very large. Meanwhile, the data owner needs extra computation costs in computing intersections and space partitioning, which will bring a lot of burden to the data owner who is generally equipped with limited resources. In total, the FQA's signature construction costs are higher than our solutions.

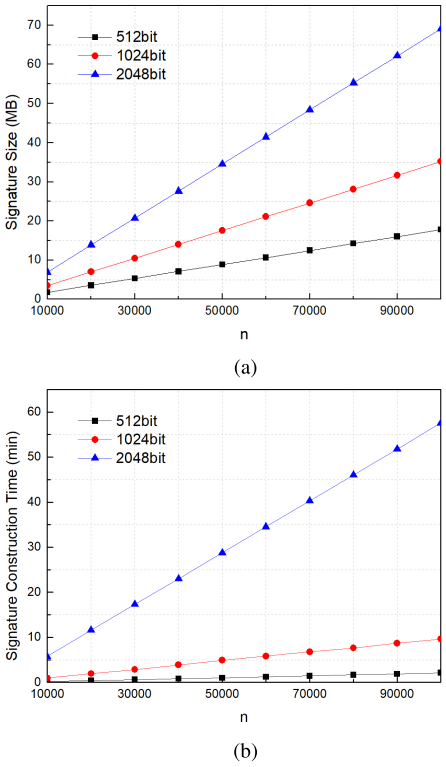


FIGURE 14. Communication and computation costs of the signature construction process. (a) Signature size. (b) Signature construction time.

2) *VO* CONSTRUCTION

The main cost of server comes from constructing the verification object and sending it to users. The verification object includes the signatures of the query results and the boundary records. The *VO* size of our basic solution can be written as $O(k)$, where k is the number of query results. Our extended solution's *VO* size is $O(l)$, where l is the number of result grids, l is smaller than k as one result grid includes more than one query result. The *VO* size of FQA is $O(k)$, actually the *VO* size is smaller than our basic solution, as FQA includes smaller boundaries. However, it's larger than our extended solution as l is smaller than k.

3) VERIFICATION COST

The user receives the query results and verification object from the server, then it verifies each query result's signature using data owner's public key. The verification cost of our basic solution can be written as $O(k)$. In our extended solution, the user receives l grids which contain more than k results, the extended solution's result accuracy is worse than our basic solution and FQA. The user needs to first query the grids and finds the k results, then the user continues to verify the signatures of the l grids, our extended solution's verification cost is $O(l)$. FQA's verification cost is $O(k)$, its verification cost is smaller than our basic solution and larger than our extended solution.

4) DATA UPDATE

In our basic solution, for d-dimensional dataset, if the owner modifies a data record, as each record is chained with d

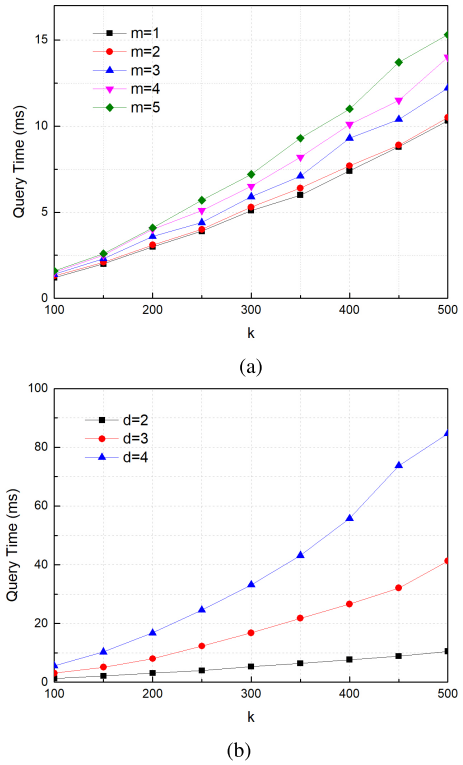


FIGURE 15. Computation cost of the query process. (a) The impact of degree *m*. (b) The impact of dimension *d*.

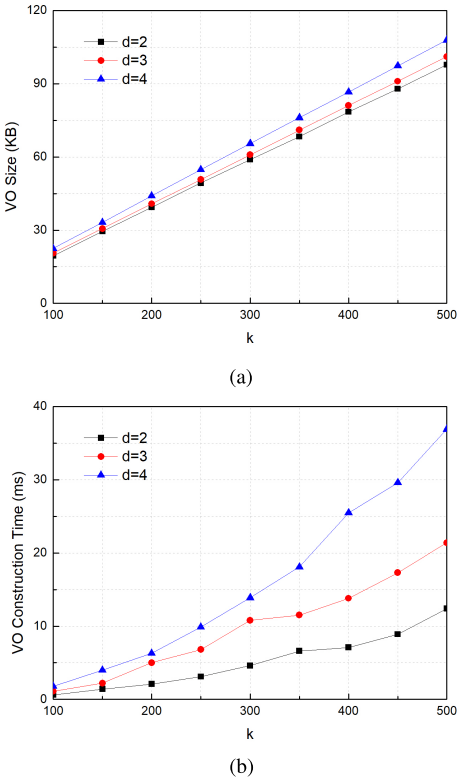


FIGURE 16. Communication and computation costs of the *VO* construction process. (a) *VO* size. (b) *VO* construction time.

records in our method, only d signatures need to be modified in update process. If she inserts a data record, the data owner needs to update $O(\sqrt[d]{n})$ signatures, so the update cost is bounded by $O(\sqrt[d]{n})$. In our extended solution, if a new record is inserted, the corresponding grid and its d predecessors' signatures need to be changed, thus the update cost is $O(d)$. However, in FQA, the number of signatures is bounded by $O(f^2)$. As d is much smaller than f generally, our solution's update cost is much smaller than FQA.

C. SIMULATION RESULTS

In this subsection, we evaluate the performance of our basic solution for multi-dimensional top-k queries over synthetic data set. We use a data generator to generate a number of synthetic data. The input to this generator includes the number of data records n and the number of dimensions d. We use SHA-1 for digest function and RSA for digital signature. Our simulation platform is a Windows server with Intel 64-bit i7 CPU running on 2.00GHz and 8 GB RAM. The parameters and default values are given in Table 3. We are interested in four performance metrics, including the signature construction cost, the query cost, the *VO* construction cost and the verification cost. Finally, we give the simulation result compared with FQA [23] in two-dimensional linear query setting. As our extended solution returns more than k results, thus it's not fair to compare it with other solutions in the same experimental setting, thus we only discuss the performance of our basic solution.

TABLE 3. Simulation settings.

Parameter	Description	Range	Default
n	Number of data records	10,000 - 100,000	50,000
k	Size of query result	100 - 500	250
d	Number of dimensions	2 - 4	2
m	The degree of dimension	1 - 5	2
sk	The key length of RSA	512, 1024, 2048	512

We first study the effects of n and sk on the communication and computation cost of constructing signatures. We adjusted the number of records from 10,000 to 100,000 with RSA key length in 512, 1024 and 2048 bits, respectively. The other parameters used the default value. The signature size is showed in Figure 14(a), and the execution time of constructing signatures is showed in Figure 14(b). Our method's signature construction time and signature size increase with the number of data records and the key length.

When the server receives a multi-dimensional top-k query, the server queries the dataset and finds the top-k results. We adjusted the number of query results returned to the user from 100 to 500 and reported the query cost. The computation cost of the query process with the effect of m and d are showed in Figure 15(a) and Figure 15(b), respectively. In Figure 15(a), the default dimension is 2, we adjusted the highest degree from 1 to 5, the scoring function is $Score(r_{ij}) = 3x_i + 5y_j$, $Score(r_{ij}) = 3x_i^2 + 5y_j^2$, $\cdots$, and $Score(r_{ij}) = 3x_i^5 + 5y_j^5$ when $m = 1$, $m = 2$, $\cdots$, and $m = 5$, respectively. In Figure 15(b), the default degree

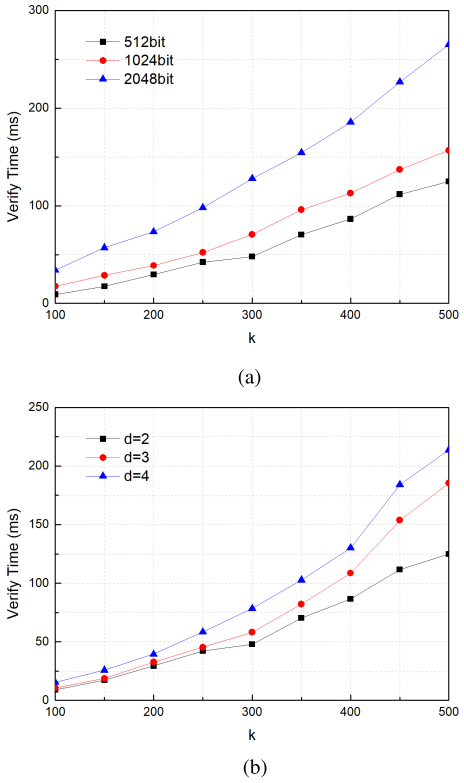


FIGURE 17. Computation cost of the verify process. (a) The impact of key length *sk*. (b) The impact of dimension *d*.

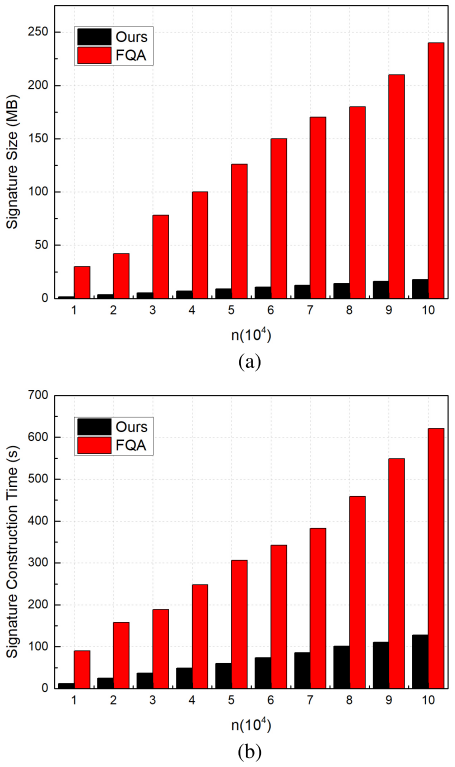


FIGURE 18. Performance comparison. (a) Signature size. (b) Signature construction time.

is 2, we adjusted the number of dimensions from 2 to 4, the scoring function is $Score(r_{ij}) = 3x_i^2 + 5y_j^2$, $Score(r_{ijp}) = 3x_i^2+4y_j^2+5z_p^2$, and $Score(r_{ijpq}) = 3x_i^2+4y_j^2+5z_p^2+6t_q^2$ when $d = 2$, $d = 3$ and $d = 4$, respectively, where x_i, y_j, z_p and t_q are the attribute values of each dimension. The result shows that the query time increases as the number of query results, the degree of dimension and the number of dimensions.

The communication and computation cost of constructing *VO* with varying number of query results and dimension are showed in Figure 16(a) and Figure 16(b), respectively. The results show that the *VO* size grows linearly with the number of query results. The computation cost of generating *VO* is determined by the size of query result and the number of dimensions.

In the verification phase, the computation cost of verifying the query results is showed in Figure 17. Figure 17(a) illustrates that the verify time is impacted by the size of the query results and key length. Figure 17(b) shows the time cost of verification process with varying number of dimensions.

Figure 18 shows the comparison of our scheme with FQA, in terms of the computation and communication cost of constructing signatures. As we observe from Figure 18, the signature size and signature construction time are much less than FQA.

In conclusion, as observed from the complexity analysis and simulation results, our solution can largely reduce the communication and computation cost in constructing signatures. In our solution, the signatures generated by data owner are much smaller than FQA, and the data owner has much smaller computation power than the server. Meanwhile, the data owner may insert, modify, or delete the data record frequently. Our method only needs to modify several signatures for one data record, but FQA needs to compute all the signatures, thus our solution is more applicable to the dynamic dataset.

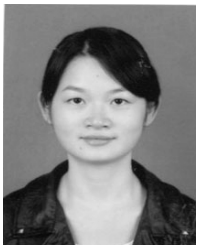
IX. CONCLUSION

In this paper, we consider that the data owner outsource a database to an untrusted server, the user submits a multi-dimensional top-*k* query to the server, and the server returns results which satisfy the query. We propose a multi-dimensional top-*k* query authentication solution to verify the query results. The user can submit queries according to their preference, and the data owner does not need to pre-compute all the possible results. Based on signature chain, each record is chained with its successors in each dimension. Through this design, any attempt to modify or drop the query result will be detected. Meanwhile, we propose an extend solution using larger grid size for sparse data partition, which can decrease the computation and communication costs in the data owner side. We prove that our multi-dimensional top-*k* query authentication solutions are secure. The complexity analysis and simulation results show that the proposed solution is practial and efficient.

REFERENCES

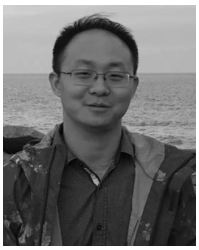
[1] Q. Liu, G. Wang, X. Liu, T. Peng, and J. Wu, "Achieving reliable and secure services in cloud computing environments," *Comput. Elect. Eng.*, vol. 59, pp. 153–164, Apr. 2017.

[2] H. H. Pang and K.-L. Tan, "Authenticating query results in edge computing," in *Proc. 20th Int. Conf. Data Eng.*, Apr. 2004, pp. 560–571.
[3] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, "Authentic data publication over the Internet," *J. Comput. Secur.*, vol. 11, no. 3, pp. 291–314, 2003.
[4] W.-S. Ku, L. Hu, C. Shahabi, and H. Wang, "Query integrity assurance of location-based services accessing outsourced spatial databases," in *Proc. Int. Symp. Spatial Temporal Databases*, 2009, pp. 80–97.
[5] M. L. Yiu, Y. Lin, and K. Mouratidis, "Efficient verification of shortest path search via authenticated hints," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, Mar. 2010, pp. 237–248.
[6] M. L. Yiu, E. Lo, and D. Yung, "Authentication of moving kNN queries," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 565–576.
[7] C. Xu, Q. Chen, H. Hu, J. Xu, and X. Hei, "Authenticating aggregate queries over set-valued data with confidentiality," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 4, pp. 630–644, Apr. 2018.
[8] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan, "Verifying completeness of relational query results in data publishing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 407–418.
[9] M. Narasimha and G. Tsudik, "Authentication of outsourced databases using signature aggregation and chaining," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2006, pp. 420–436.
[10] H. Pang, J. Zhang, and K. Mouratidis, "Scalable verification for outsourced dynamic databases," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 802–813, 2009.
[11] X. Zhu, J. Wu, W. Chang, G. Wang, and Q. Liu, "Authentication of multi-dimensional top-K query on untrusted server," in *Proc. IEEE/ACM Int. Symp. Quality Service*, 2018, pp. 1–6.
[12] R. C. Merkle, "A certified digital signature," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 218–238.
[13] S. Su, H. Yan, X. Cheng, P. Tang, P. Xu, and J. Xu, "Authentication of top-k spatial keyword queries in outsourced databases," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2015, pp. 567–588.
[14] D. Wu, B. Choi, J. Xu, and C. S. Jensen, "Authentication of moving top-k spatial keyword queries," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 4, pp. 922–935, Apr. 2015.
[15] Y. Tang, L. Liu, T. Wang, X. Hu, R. Sailer, and P. Pietzuch, "Outsourcing multi-version key-value stores with verifiable data freshness," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar./Apr. 2014, pp. 1214–1217.
[16] D. Yung, Y. Li, E. Lo, and M. L. Yiu, "Efficient authentication of continuously moving $k$nn queries," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1806–1819, Sep. 2015.
[17] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with Voronoi neighbors," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 863–876, Apr. 2013.
[18] Y. Jing, L. Hu, W.-S. Ku, and C. Shahabi, "Authentication of k nearest neighbor query on road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 6, pp. 1494–1506, Jun. 2014.
[19] W. Chen, M. Liu, R. Zhang, Y. Zhang, and S. Liu, "Secure outsourced skyline query processing via untrusted cloud service providers," in *Proc. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[20] X. Lin, J. Xu, H. Hu, and W.-C. Lee, "Authenticating location-based skyline queries in arbitrary subspaces," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 6, pp. 1479–1493, Jun. 2014.
[21] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, "Authenticated indexing for outsourced spatial databases," *VLDB J.*, vol. 18, no. 3, pp. 631–648, 2009.
[22] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, "Authenticated index structures for aggregation queries," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 4, 2010, Art. no. 32.
[23] G. Yang, Y. Cai, and Z. Hu, "Authentication of function queries," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, May 2016, pp. 337–348.
[24] Q. Liu, S. Wu, S. Pei, J. Wu, T. Peng, and G. Wang, "Secure and efficient multi-attribute range queries based on comparable inner product encoding," in *Proc. IEEE Conf. Commun. Netw. Secur.*, May/Jun. 2018, pp. 1–9.
[25] W. Chang and J. Wu, "Progressive or conservative: Rationally allocate cooperative work in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 2020–2035, Jul. 2015.
[26] C. Reitz, M.-X. Tang, N. Schupf, J. J. Manly, R. Mayeux, and J. A. Luchsinger, "A summary risk score for the prediction of alzheimer disease in elderly persons," *Arch. Neurology*, vol. 67, no. 7, pp. 835–841, 2010.
[27] E. Luo, M. Z. A. Bhuiyan, G. Wang, M. A. Rahman, J. Wu, and M. Atiquzzaman, "Privacyprotector: Privacy-protected patient data collection in IoT-based healthcare systems," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 163–168, Feb. 2018.
[28] H. Zheng, W. Chang, and J. Wu, "Coverage and distinguishability requirements for traffic flow monitoring systems," in *Proc. IEEE/ACM 24th Int. Symp. Quality Service*, Jun. 2016, pp. 1–10.
[29] W. Cheng, H. Pang, and K.-L. Tan, "Authenticating multi-dimensional query results in data publishing," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2006, pp. 60–73.
[30] Y.-T. Tsou, Y.-L. Hu, Y. Huang, and S.-Y. Kuo, "SFTopk: Secure functional top-k query via untrusted data storage," *IEEE Access*, vol. 3, pp. 2875–2890, 2015.
[31] S. Choi, H.-S. Lim, and E. Bertino, "Authenticated top-K aggregation in distributed and outsourced databases," in *Proc. Int. Conf. Social Comput.*, Sep. 2012, pp. 779–788.
[32] Q. Chen, H. Hu, and J. Xu, "Authenticating top-k queries in location-based services with confidentiality," *Proc. VLDB Endowment*, vol. 7, no. 1, pp. 49–60, 2013.
[33] Y. Wang, S. Gao, J. Zhang, X. Nie, X. Duan, and J. Chen, "Authenticating multiple user-defined spatial queries," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jun. 2016, pp. 471–480.
[34] X. Duan, Y. Wang, J. Chen, and J. Zhang, "Authenticating preference-oriented multiple users spatial queries," in *Proc. IEEE 41st Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2017, pp. 602–607.
[35] Q. Zhang, Q. Liu, and G. Wang, "PRMS: A personalized mobile search over encrypted outsourced data," *IEEE Access*, vol. 6, pp. 31541–31552, 2018, doi: 10.1109/ACCESS.2018.2845468.
[36] S. Kumari, M. K. Khan, and X. Li, "An improved remote user authentication scheme with key agreement," *Comput. Elect. Eng.*, vol. 40, no. 6, pp. 1997–2012, 2014.
[37] S. Kumari, X. Li, F. Wu, A. K. Das, H. Arshad, and M. K. Khan, "A user friendly mutual authentication and key agreement scheme for wireless sensor networks using chaotic maps," *Future Gener. Comput. Syst.*, vol. 63, pp. 56–75, Oct. 2016.
[38] S. Kumari, X. Li, F. Wu, A. K. Das, K.-K. R. Choo, and J. Shen, "Design of a provably secure biometrics-based multi-cloud-server authentication scheme," *Future Gener. Comput. Syst.*, vol. 68, pp. 320–330, Mar. 2017.
[39] S. Zhang, X. Li, Z. Tan, T. Peng, and G. Wang, "A caching and spatial K-anonymity driven privacy enhancement scheme in continuous location-based services," *Future Gener. Comput. Syst.*, vol. 94, pp. 40–50, May 2019.
[40] S. Zhang, K.-K. R. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Gener. Comput. Syst.*, vol. 86, pp. 881–892, Sep. 2018.
[41] S. Zhang, G. Wang, M. Z. A. Bhuiyan, and Q. Liu, "A dual privacy preserving scheme in continuous location-based services," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4191–4200, Oct. 2018, doi: 10.1109/JIOT.2018.2842470.
[42] C. N. Gutierrez, M. H. Almeshekah, E. H. Spafford, M. J. Atallah, and J. Avery, "Inhibiting and detecting offline password cracking using ersatz-passwords," *ACM Trans. Privacy Secur.*, vol. 19, no. 3, 2016, Art. no. 9.
[43] E. Glass. *The NTLM Authentication Protocol and Security Support Provider*. Accessed: 2018. [Online]. Available: http://davenport.sourceforge.net/ntlm.html
[44] M. Schrader *et al.*, *Oracle Essbase & Oracle OLAP: The Guide to Oracle's Multidimensional Solution*. New York, NY, USA: McGraw-Hill, 2009.
[45] D. Volitich and G. Ruppert, *IBM Cognos Business Intelligence 10: The Official Guide*. New York, NY, USA: McGraw-Hill, 2012.



XIAOYU ZHU received the B.Sc. degree in electronic information engineering and the M.Sc. degree in computer science from Central South University, Changsha, China, in 2010 and 2013, respectively, where she is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Information Science and Engineering. She has been a Visiting Student with Temple University, Philadelphia, PA, USA. Her research interests include cloud computing, searchable encryption, and query authentication.

JIE WU is currently the Director of the Center for Networked Computing and also a Laura H. Carnell Professor with Temple University. He also serves as the Director of international affairs with the College of Science and Technology. He served as a Chair of the Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and as an Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. He was the Program Director with the National Science Foundation and was a Distinguished Professor with Florida Atlantic University. He regularly publishes in scholarly journals, conference proceedings, and books. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He was a recipient of the 2011 CCF Overseas Outstanding Achievement Award. He was a Chair of the IEEE Technical Committee on Distributed Processing. He was a General Co-Chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, and a Program Co-Chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He serves on several editorial boards, including the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON SERVICE COMPUTING, the *Journal of Parallel and Distributed Computing*, and the *Journal of Computer Science and Technology*. He was an IEEE Computer Society Distinguished Visitor and an ACM Distinguished Speaker. He is a China Computer Federation (CCF) Distinguished Speaker.



WEI CHANG received the B.S. degree from the Beijing University of Posts and Telecommunications, in 2009, and the Ph.D. degree in computer science from Temple University, in 2016. He is currently an Assistant Professor with the Computer Science Department, Saint Joseph's University, Philadelphia, PA, USA. He serves as the Coordinator of Cybersecurity Program with the Computer Science Department, Saint Joseph's University. His research interests include social information-assisted system design, and its related security and privacy issues.

GUOJUN WANG received the B.Sc. degree in geophysics, the M.Sc. degree in computer science, and the Ph.D. degree in computer science from Central South University, Changsha, China, in 1992, 1996, and 2002, respectively. He had been a Professor with Central South University, China, an Adjunct Professor with Temple University, USA, a Visiting Scholar with Florida Atlantic University, USA, a Visiting Researcher with the University of Aizu, Japan, and a Research Fellow with The Hong Kong Polytechnic University. He is currently a Professor with the School of Computer Science and Educational Software, Guangzhou University. His research interests include network and information security, the Internet of Things, and cloud computing. He is a Distinguished Member of CCF and a member of ACM and IEICE.

QIN LIU received the B.Sc. degree in computer science from Hunan Normal University, Changsha, China, in 2004, and the M.Sc. and Ph.D. degrees in computer science from Central South University, Changsha, China, in 2007 and 2012, respectively. She has been a Visiting Student with Temple University, Philadelphia, PA, USA. She is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan. Her research interests include security and privacy issues in cloud computing.