Authentication of Multi-dimensional Top-K Query on Untrusted Server

Xiaoyu Zhu[†‡], Jie Wu[‡], Wei Chang[§], Guojun Wang[¶*], and Qin Liu[‖]
[†]School of Information Science and Engineering, Central South University, Changsha, 410083, China
[‡]Center for Networked Computing, Temple University, PA, 19122, USA
[§]Department of Computer Science, Saint Joseph's University, PA, 19131, USA
[¶]School of Computer, Guangzhou University, Guangzhou, 510006, China
[‖]School of Computer Science and Electronic Engineering, Hunan University, Changsha, 410082, China
[†]zhuxiaoyu@csu.edu.cn, [‡]jiewu@temple.edu, [§]wchang@sju.edu, [‖]gracelq628@hnu.edu.cn
*Correspondence to: csgjwang@gmail.com

***Abstract*—Consider a database where each record has multiple attributes. An untrusted server is in charge of processing queries over this database, and we want to provide a mechanism for users to verify the correctness of their query results. Here each query, referred to as a multi-dimensional top-k query, retrieves k records whose output with user-supplied ranking function is among top k. Multi-dimensional top-k query is widely used in real applications. However, as the traditional query authentication methods cannot be directly deployed on multi-dimensional top-k query, it is still a challenging problem to authenticate the multi-dimensional top-k query results. In this paper, we 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 multi-dimensional top-k query results. Through theoretical analysis and simulation, we demonstrate the effectiveness of our proposed solution.**

***Index Terms*—Data outsourcing, multi-dimension, query authentication, top-k query**

I. Introduction

Database outsourcing has lately emerged as a common practice for companies and institutions. 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. This model is applicable to a wide range of computing platforms, including cloud computing, edge computing, database caching, etc. 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 manipulate data due to possible virus or even 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, regardless of whether each result appears in the original database (soundness) or whether all data records in the original database that satisfy the query condition are included in the query results (completeness).

The problem of authenticating the query results has been studied a lot in the past decades. Devanbu et al. [1] proposed a solution to solve the problem of authenticating range query for the first time. The proposed solution allows a data owner sort the data records to be outsourced and build a Merkle Hash tree (MH-tree) [2] on these data records. The data records and the corresponding MH-tree are then both outsourced to the third party. Without causing ambiguity, we will use the terms third party and server interchangeably. In response to a range query, the server is required to return the query result and some part of the MH-tree that can serve as the proof that the data records in the query result are original and contiguous in order. Pang et al. [3] proposed an alternative solution which builds a digital signature chain instead of an MH-tree. The two authentication data structures, MH-tree and signature chain have since inspired a large body of research (e.g.,[3],[4],[5],[6]) on authenticating various queries such as multi-dimensional range query, and KNN query and spatial queries. However, they support only simple query authentication.

In this paper, we consider the problem of authenticating multi-dimensional top-k query. The outsourced data contain multiple dimensions/attributes, the users can ask queries with ranking functions and retrieve k records whose output is among top k. The ranking function can be the sum of arbitrary composition of attributes, each attribute can be assigned a weight and have a degree. For example, Assume there is a three-dimensional dataset *Student(GPA, Award, Paper)*; a user may want to retrieve the students whose rank scores are among the top k. The user may submit such a ranking function *Score = 5GPA + 3Award + 2Paper*^{2}, where *GPA*, *Award*, *Paper* are the *Student*'s corresponding attribute values, (5, 3, 2) are the weights for (*GPA*, *Award*, *Paper*) respectively, and *Paper* has a degree up to 2. Multi-dimensional top-k query has many important applications, including information retrieval in cloud computing [7], [8], social network [9], location-based services [10], system monitoring [11], resource allocation [12], disease prediction [13], and etc. Users can use existing SQL tool to define and issue multi-dimensional top-k query.

Recently, Yang et al. [14] proposed a function query authentication method, which is similar to the multi-dimensional top-k query. But in their method, the ranking function definition is submitted by the data owner rather than the user, which limits users' query preference. In addition, their solution has several major limitations. The communication and computation cost of the signature construction process is very high, as each data record may have n signatures, the signature size can be $O(n^2)$ at most, where n is the total number of data records. However, the signature generation is a computationally expensive operation, which will bring a lot of burden to the data owner who is generally equipped with limited resources. Moreover, 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.

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 this paper, we propose an idea to solve the multi-dimensional top-k authentication problem.

The contributions of this paper are summarized as follows:

- We propose an authentication solution to verify multi-dimensional top-k query results efficiently. Our method supports efficient signature construction process, and each record is chained with its successors on each dimension; the communication and computation cost is reduced due to the signature construction design.
- Our method supports efficient data updates by the data owner, which only needs to modify several signatures for the update operation, and our method gives users great query flexibility; the query function definition can be submitted by the users instead of the data owner.
- We give the performance analysis and conduct extensive experiments, which shows the effectiveness and efficiency of our method.

The remainder of the paper is organized as follows: Section II gives the overview of our solution. In section III, we describe the details of our proposed solution. Following in section IV, we extend our solution. Section V presents the security analysis and performance analysis. Finally, section VI concludes our paper.

II. Overview

A. Query Definition

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 ranking 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 r whose ranking score $Score(r)$ is among the k smallest. For each data record $r \in D$, the ranking function of this data record is defined as

$$\begin{aligned} Score(r) =& w_0 + \\ & w_{11}t_1 + w_{12}t_1^2 + \cdots + w_{1m}t_1^m + \\ & w_{21}t_2 + w_{22}t_2^2 + \cdots + w_{2m}t_2^m + \\ & \cdots \\ & w_{d1}t_d + w_{d2}t_2^2 + \cdots + w_{dm}t_d^m \end{aligned} \quad (1)$$

Here, d is the number of dimensions, m is the highest degree, $(t_1, t_2, \cdots, t_d)$ is the data record r's attribute values on dimension $(1, 2, \cdots, d)$ respectively, and w_{ij} is a positive weight for term t_d^m, where $1 \leq i \leq d$ and $1 \leq j \leq m$.

B. Security Goal

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 to the 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. They have not been tampered with, nor have spurious data records been introduced.

Completeness: The user can verify that the results covers 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.

C. Scheme Outline

The system involves three types of parties: data owner, server, and data user. Our scheme includes five algorithms as follows:

- $Init(T) \rightarrow D$: The data owner takes a dataset T as input and outputs a new dataset D.
- $KeyGen(k) \rightarrow \{pk, sk\}$: The data owner takes a security parameter k as input and outputs a public key pk and a private key sk.
- $SigGen(sk, D) \rightarrow S$: The data owner takes private key sk and dataset D as inputs and outputs a set of signatures S.
- $GenProof(Q, D) \rightarrow \{R, VO\}$: The server takes query Q and dataset D as inputs and outputs query results R and a verification object VO.
- $Verify(pk, R, VO) \rightarrow \{0, 1\}$: The user 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. 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.

III. Proposed Solution

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.

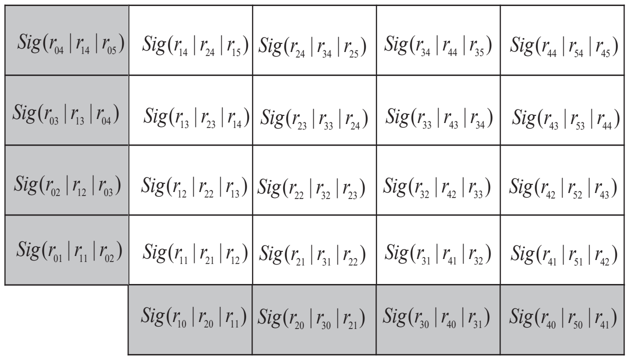
Grid. The original dataset T can be sorted in an increasing order for x and y dimension. The data record is expressed as $r_{ij} = (x_i, y_j)$, where x_i and y_j are the attribute values on the x and y dimension, respectively. Some data records have two successors, one in the x dimension, another in the y dimension. 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, 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.

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 $B = \{B_l, B_u\}$.

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. 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:



$Sig(r_{04} \| r_{14} \| r_{05})$	$Sig(r_{14} \| r_{24} \| r_{15})$	$Sig(r_{24} \| r_{34} \| r_{25})$	$Sig(r_{34} \| r_{44} \| r_{35})$	$Sig(r_{44} \| r_{54} \| r_{45})$
$Sig(r_{03} \| r_{13} \| r_{04})$	$Sig(r_{13} \| r_{23} \| r_{14})$	$Sig(r_{23} \| r_{33} \| r_{24})$	$Sig(r_{33} \| r_{43} \| r_{34})$	$Sig(r_{43} \| r_{53} \| r_{44})$
$Sig(r_{02} \| r_{12} \| r_{03})$	$Sig(r_{12} \| r_{22} \| r_{13})$	$Sig(r_{22} \| r_{32} \| r_{23})$	$Sig(r_{32} \| r_{42} \| r_{33})$	$Sig(r_{42} \| r_{52} \| r_{43})$
$Sig(r_{01} \| r_{11} \| r_{02})$	$Sig(r_{11} \| r_{21} \| r_{12})$	$Sig(r_{21} \| r_{31} \| r_{22})$	$Sig(r_{31} \| r_{41} \| r_{32})$	$Sig(r_{41} \| r_{51} \| r_{42})$
	$Sig(r_{10} \| r_{20} \| r_{11})$	$Sig(r_{20} \| r_{30} \| r_{21})$	$Sig(r_{30} \| r_{40} \| r_{31})$	$Sig(r_{40} \| r_{50} \| r_{41})$

Fig. 1. Signatures construction process.

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

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 are 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 $D \cup B_l$, which is bounded by $O(n)$. Figure 1 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 its signature concatenates itself and its two successors.

B. Query Result Verification

In this subsection, we introduce three parts. First, 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.

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 in upper boundaries B^u. Then for each data record r_{ij} in results R and lower boundaries B^l, the

Algorithm 1 Query result verification

```
Input: Query $Q = \{Score(r), k\}$, results $R$, verification object $VO = \{B^l, B^u, S_q\}$.
Output: 0 or 1.
1:  for For each record $r_{ij} \in \{R \cup B^l\}$ do
2:      Find its corresponding signature in $S_q$
3:      Find its successors $r_{i+1,j}$ and $r_{i,j+1}$ in $R \cup B^l \cup B^u$
4:      if Eqn. 3 is not satisfied then
5:          Return 0
6:      end if
7:  end for
8:  if the number of data in $R$ is not equal to $k$ then
9:      Return 0
10: end if
11: Set the maximum score of results in $R$ as $Score_m$
12: for each $r_{ij}$ in $B^u$ do
13:     if $Score(r_{ij}) > Score_m$ then
14:         Return 0
15:     end if
16: end for
17: Return 1
```

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 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 $r_{ij} \in \{R \cup B^l\}$, the user finds its signature $Sig(r_{ij}|r_{i+1,j}|r_{i,j+1})$ in S_q, and find 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:

$$\begin{aligned} &Sig^{-1}(Sig(r_{ij}|r_{i+1,j}|r_{i,j+1}), pk) \\ &\qquad = H(H(r_{ij})|H(r_{i+1,j})|H(r_{i,j+1})) \end{aligned} \tag{3}$$

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 1.

Data update. The data update process of our solution is very simple. When the data owner updates a data record, the data owner first finds the data record's predecessors in each dimension, then generates a new signature for these predecessors.

IV. Extensions

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

A. Multi-dimensional Top-k Query Authentication

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 a dummy data over d attributes, the dummy data's attribute in each dimension is equal to its corresponding axis value. The original data and dummy data constructs the new dataset D.

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 you 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 the successor in the last dimension. The data owner creates the signature of $r_{i,\cdots,j}$ as follows:

$$\begin{aligned} Sig(r_{i,\cdots,j}) = &Sig(H(H(r_{i,\cdots,j})|H(r_{i+1,\cdots,j})| \\ &\cdots|H(r_{i,\cdots,j+1}))) \end{aligned} \tag{4}$$

Verification object. 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. 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:

$$\begin{aligned} &Sig^{-1}(Sig(r_{i,\cdots,j}), pk) = H(H(r_{i,\cdots,j})| \\ &H(r_{i+1,\cdots,j})|\cdots|H(r_{i,\cdots,j+1})) \end{aligned} \tag{5}$$

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

V. Performance Study

In this section, we study the performance of the proposed solutions through security, overhead analysis, and simulation.

A. Security Analysis

We prove that the proposed multi-dimensional top-k query authentication scheme 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. This happens when the adversary creates a fake record $r_{i',\cdots,j'}$ to replace $r_{i,\cdots,j}$. In order to make the user accept the manipulated query result, the adversary must forge a signature $Sig(r_{i,\cdots,j})$ using the digest of fake $r_{i',\cdots,j'}$. This is computationally infeasible without knowing the private key of the data owner.

Then we discuss three cases that 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 let user accept the forged result, the adversary must forge fake signatures for initial boundaries. This is computationally infeasible without knowing the private key of the data owner.

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 checking 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 $Sig(r_{i,\cdots,j})$. This is computationally infeasible without knowing the private key of the data owner.

B. Overhead

We now analyze the overhead introduced by the proposed technique on the data owner, the server, and the user side, respectively. We give the performance analysis compared with the benchmark method in [14] (denoted by FQA).

1) Data Owner Overhead: The data owner's computation cost incurred in constructing the signatures. The key factor is the size of signatures, and it's not sensitive to the number of dimension, as it only needs to execute one more hash operation for each record as the number of dimension increases one number. Next, we analyze the total number of signatures and signature chains that need to be created, and we compare our solution with FQA.

The data owner first preprocesses the dataset and generates a new dataset. Suppose there are n data in the new dataset with d dimensions. The data owner needs to create n signatures, FQA needs to construct $2 * C_2^{n+2}$ signatures, which is much larger than our method. The signature generation is a computationally expensive operation. Moreover, in FQA, the data owner needs extra computation costs in computing intersections and space partitioning, which is really a large computation cost. For two-dimensional dataset, the total number of signature chains is $2\sqrt{n}$. The number of chains is $2\sqrt[3]{n^2} + \sqrt[3]{n}$ for three-dimensional dataset, which is bounded by $O(n)$. The number of chains is bounded by $O(dn)$ for d-dimensional dataset. In FQA, the number of signature chains is bounded by $O(n^2)$. As d is much smaller than n generally, our method's signature chains are much smaller than FQA. As for the update cost, each record is chained with d records in our method, so only d signatures need to be modified in update process. However, FQA needs to construct all the signatures, which is really a large communication and computation cost for the data owner.

2) Server Overhead: For the server, there is a one-time cost in receiving the dataset and corresponding signatures from the data owner. After that, the main cost is constructing the verification object and sending it to users. The average cost of constructing VO can be written as $O(k)$, where k is the number of query results.

3) User Overhead: Each user needs to retrieve the public key from the data owner. This one-time cost is minimal. There are two main overheads, the communication cost of receiving R and VO from the server, and the computation cost in the verification process. The average cost of verifying signatures can be written as $O(k)$.

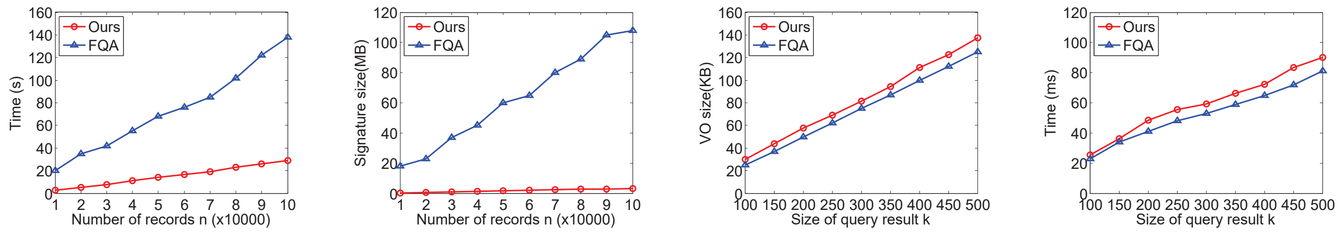
C. Simulation

We have implemented a detailed simulator that allows us to evaluate the performance of the proposed technique from various aspects. Our simulator incorporate a data generator that can be configured to generate various kinds of data. We use SHA-1 for digest function and RSA (256 bits) for digital signature. Our simulation platform is a Windows server with Intel 64-bit i7 CPU running on 2.00GHz and 8 GB RAM. We give the simulation result compared with the benchmark method in [14] (denoted by FQA) in two-dimensional linear query setting. The number of dimension d and the highest degree m are default as 2 and 1 respectively.

The experiment results show the comparison of our methods and FQA in four overheads, which are the owner's computation overhead, owner-CSP communication overhead, CSP-user communication overhead, and user's computation overhead, respectively.

Fig. 2(a) shows the comparison of ours with FQA in owner's computation overhead with the number of records n from 10,000 to 100,000. We can see that FQA needs much higher computation overhead than ours, accurately FQA incurs 6 times construction overhead of our method in average.

Fig. 2(b) shows the comparison of ours with FQA in Owner-CSP communication overhead with the number of records n from 10,000 to 100,000. The size of constructed signatures



(a) Owner's computation overhead (b) Owner-CSP comm. overhead (c) CSP-user comm. overhead (d) User's computation overhead

Fig. 2. Computation and communication cost

grow linearly with the number of data records, FQA incurs about 33 larger signature size than our solution in average.

Fig. 2(c) shows the comparison of ours with FQA in CSP-user communication overhead with the size of query results k from 100 to 500. The result shows that the size of VO increases linearly to the size of query result, and our method's VO size is larger than FQA's, as our method needs to generate more boundaries for the query results.

Fig. 2(d) shows the comparison of ours with FQA in user's computation overhead with the size of query results k from 100 to 500. We can see that our method is a bit larger than the FQA, as our method needs to verify more boundaries than FQA.

In conclusion, our method incurs a larger cost in the CSP-user communication overhead and user computation overhead. However, we can largely reduce the communication and computation cost in constructing signatures. In our solution, the signatures generated by data owner is 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.

VI. Conclusion

In this paper, we consider the problem of outsourcing 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 users' query. We propose a solution that allows users to verify if the query results are sound and complete. The user can submit queries according to their preference, and the data owner does not need to pre-compute all the possible results. We develop a solution for efficient verification of multi-dimensional top-k query based on creating a signature chain for data records in each dimension. We prove that our solution is secure, and the experiment results show the proposed solutions are practical and can be used in real-world applications.

Acknowledgment

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

References

[1] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine, "Authentic data publication over the internet," *Journal of Computer Security*, vol. 11, no. 3, pp. 291–314, 2003.
[2] R. C. Merkle, "A certified digital signature," in *Conference on the Theory and Application of Cryptology*, 1989, pp. 218–238.
[3] H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan, "Verifying completeness of relational query results in data publishing," in *ACM SIGMOD*, 2005, pp. 407–418.
[4] Y. Tang, T. Wang, X. Hu, R. Sailer, L. Liu, and P. Pietzuch, "Outsourcing multi-version key-value stores with verifiable data freshness," in *ICDE*, 2014, pp. 1214–1217.
[5] Y. Yang, S. Papadopoulos, and D. Papadias, "Authenticated indexing for outsourced spatial databases," *The VLDB Journal*, vol. 18, no. 3, pp. 631–648, 2009.
[6] S. Su, H. Yan, X. Cheng, P. Tang, P. Xu, and J. Xu, "Authentication of top-k spatial keyword queries in outsourced databases." in *DASFAA*, 2015, pp. 567–588.
[7] 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 *IEEE CNS*, to be appeared, 2018.
[8] Q. Liu, G. Wang, X. Liu, T. Peng, and J. Wu, "Achieving reliable and secure services in cloud computing environments," *Computers & Electrical Engineering*, 2016.
[9] Q. Liu, G. Wang, F. Li, S. Yang, and J. Wu, "Preserving privacy with probabilistic indistinguishability in weighted social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 5, pp. 1417–1429, 2017.
[10] S. Zhang, K.-K. R. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Generation Computer Systems*, 2017.
[11] H. Zheng, W. Chang, and J. Wu, "Coverage and distinguishability requirements for traffic flow monitoring systems," in *IWQoS*, 2016, pp. 1–10.
[12] W. Chang and J. Wu, "Progressive or conservative: Rationally allocate cooperative work in mobile social networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 7, pp. 2020–2035, 2015.
[13] 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," *Archives of neurology*, vol. 67, no. 7, pp. 835–841, 2010.
[14] G. Yang, Y. Cai, and Z. Hu, "Authentication of function queries," in *ICDE*, 2016, pp. 337–348.