Achieving Efficient and Privacy-Preserving Top-k Query over Vertically Distributed Data Sources

Yandong Zheng, Rongxing Lu*, Xue Yang and Jun Shao

Contact Email: rlu1@unb.ca

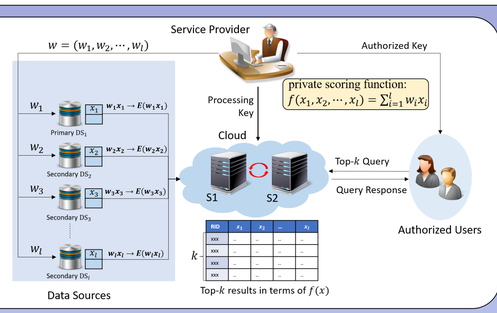
Canadian Institute for Cybersecurity (CIC), Faculty of Computer Science, University of New Brunswick (UNB)

ABSTRACT

Top-k query is an efficient way to find the most important objects from high volumes of data. A common way to process the top-k query over distributed data is to bring them to a centralized entity (e.g. cloud). However, there are privacy considerations during the top-k query when dealing with sensitive data (e.g. eHealthcare data) in such method. Apart from data privacy, efficiency also needs to be taken into consideration. In this work, we propose an efficient and privacy-preserving top-k query scheme over vertically distributed data. Specifically, we first design a data filtering technique to reduce the communication overhead and computational cost. Then, we propose a privacy-preserving top-k query scheme over encrypted data by deploying the homomorphic encryption technique, which can well preserve the private information and achieve the functionality at the same time. Besides, performance evaluation validates the efficiency of our proposed scheme.

System Model

- **Service Provider (SP)**: The SP is responsible for bootstrapping the whole scheme.
- **Data Sources** $DS = \{DS_1, \cdots, DS_l\}$: Data are vertically distributed in DS, where DS_1 is a primary data source with $w_1 > 0.5$ and other DS_i is secondary data source.
- **Cloud Server** $CS = \{S_1, S_2\}$: S_1 and S_2 cooperate to deal with data storing and query request.
- **Query-authorized Users** $U = \{U_1, U_2 \cdots\}$: Each U_i can request a top-k query to the cloud and receive the desirable result.



Design Goals

- Privacy preservation: The private scoring function, data stored in the cloud and the top-k query results should be privacy-preserving.
- Efficiency: The proposed scheme should be communication efficient and computation efficient.

A. Data Filtering Technique

(1) Top-k Query over Vertically Distributed Data Sources

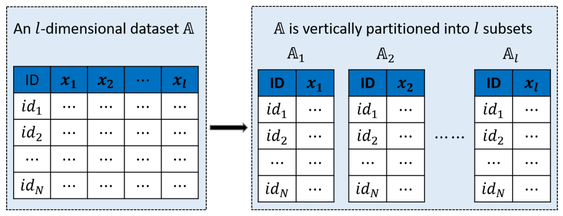


Fig. 1. Explanation for vertically distributed data sources

As shown in Fig. 1, an l-dimensional dataset $\mathbb{A}$ is vertically partitioned into l subsets $\{\mathbb{A}_1, \mathbb{A}_2, \cdots, \mathbb{A}_l\}$. Each data record x in $\mathbb{A}$ contains an identity id and l attributes $\{x_1, x_2, \cdots, x_l\}$. Each $\mathbb{A}_i$ contains x in the form of (id, x_i).

Suppose that the scoring function is $f(x_1, x_2, \cdots, x_l) = \sum_{i=1}^{l} w_i x_i$, where $w_i \in [0,1]$ and $\sum_{i=1}^{l} w_i = 1$. Then, the top-k query over vertically distributed data sources is to find k data records from a set of vertically distributed subsets $\{\mathbb{A}_1, \mathbb{A}_2, \cdots, \mathbb{A}_l\}$, where the selected k data records have the highest overall scores.

(2) Data Filtering Technique

Suppose that each DS_i has a dataset $\mathbb{A}_i$, the attribute values in $\mathbb{A}_1$ has been transformed to $[0,1]$ by the standardization and $\mathbb{A}_1$ is sorted in the descending order of the attribute values x_1. Before outsourcing data to the cloud for top-k query, the primary data source DS_1 can filter on the subset $\mathbb{A}_1$ and the filtered subset is $\mathbb{A}_1' = \{A_1[1], A_1[2], \cdots, A_1[N_1 - 1]\}$, where $A_1[N_1]$ is the first data record that satisfies $w_1 A_1[N_1].value + 1 - w_1 < w_1 A_1[k].value$. At the same time, each secondary DS_i refers $\mathbb{A}_1'$ to filter the subset $\mathbb{A}_i$ and the filtered subset $\mathbb{A}_i' = \{A_i[j] | A_i[j].id \in ID'\}$, where $ID' = \{A_1[j].id | 1 \le j \le N_1 - 1\}$ contains all identities in $\mathbb{A}_1'$.

B. The Proposed Top-k Query Scheme over Vertically Distributed Data

(1) System Initialization

The SP generates public key pk, private key $sk = (s_1, s_2)$, and a set of access keys $AK = \{ak_1, ak_2, \cdots, ak_l\}$. Meanwhile, the SP determines the weight values $\{w_1, w_2, \cdots, w_l\}$ of the scoring function $f(x_1, x_2, \cdots, x_l) = \sum_{i=1}^{l} w_i x_i$. Then, SP will send security keys and weight values to corresponding entities.

(2) Local Data Filtering and Outsourcing

Each DS_i has a dataset $\mathbb{A}_i$ and receives w_i from SP. Before data outsourcing, DS_i conducts data filtering and encryption as follows.

Step-1: The primary DS_1 sorts the data records in $\mathbb{A}_1$ in the descending order of the attribute values and searches the first N_1 such that $w_1 A_1[N_1].value + 1 - w_1 < w_1 A_1[k].value$. Let $\mathbb{A}_1' = \{A_1[1], A_1[2], \cdots, A_1[N_1 - 1]\}$ and $ID' = \{A_1[j].id | 1 \le j \le N_1 - 1\}$. Then, DS_1 broadcasts ID' to other data sources.

Step-2: Based on ID', each secondary DS_i generates the filtered dataset $\mathbb{A}_i' = \{A_i[j] | A_i[j].id \in ID'\}$.

Step-3: Each DS_i encrypts each $A_i'[j] \in \mathbb{A}_i'$ as $E_{pk}(w_i A_i'[j].value)$ and $AES_{ak_i}(A_i'[j].value)$, where $1 \le j \le N_1 - 1$.

Step-4: Each DS_i outsources encrypted data to S_1 and S_1 organizes the encrypted dataset as $\mathbb{A}' = \{id = A_1'[j].id,$

$$cipher_1 = (E_{pk}(w_1 A_1'[j].value), \cdots, E_{pk}(w_l A_l'[j].value)),$$

$$cipher_2 = (AES_{ak_1}(A_1'[j].value), \cdots, AES_{ak_l}(A_l'[j].value))$$

$$| 1 \le j \le N_1 - 1\}.$$

(3) Top-k Query over Encrypted Data

User U_i can enjoy the top-k query service as the following steps.

Step-1: U_i chooses a random session key ak and sends query request with $E_{pk}(k||ak)$ to S_1.

Step-2: On receiving query request, S_1 and S_2 cooperate to recover k and ak from $E_{pk}(k||ak)$. Then, S_1 first builds a minimum heap of size k using the first k data records of $\mathbb{A}'$ and the overall score of each heap node is used as the key value of the heap. For $A'[j] \in \mathbb{A}'$, the heap node has attributes $A'[j].id$, $A'[j].cipher_1$, $A'[j].cipher_2$ and $E(f(A'[j]))$, where $E(f(A'[j]))$ is the overall score of $A'[j]$ in ciphertext and it can be computed as $E(f(A'[j])) = \prod_{i=1}^{l} E_{pk}(w_i A_i'[j].value)$.

Step-3: Compare each $A'[j] \in \mathbb{A}'$ with the root node of the heap, If $f(A'[j])$ is larger than that of the root node, delete the root node and insert $A'[j]$ into the heap. Otherwise, continue to consider the next data record in $\mathbb{A}'$. The top-k query is finished until all data records in $\mathbb{A}'$ have been checked.

Step-4: Suppose that $T \subseteq \mathbb{A}'$ is the k data records in the heap, S_1 encrypts each $T[j] \in T$ as $AES_{ak}(T[j].cipher_2)$ and sends them to U_i.

Step-5: On receiving the query results, the query user U_i first uses the session key ak to recover $\{T[j].cipher_2 | 1 \le j \le k\}$. Then, he/she can recover the data records in plaintext by using the access key AK.

C. Performance Evaluation

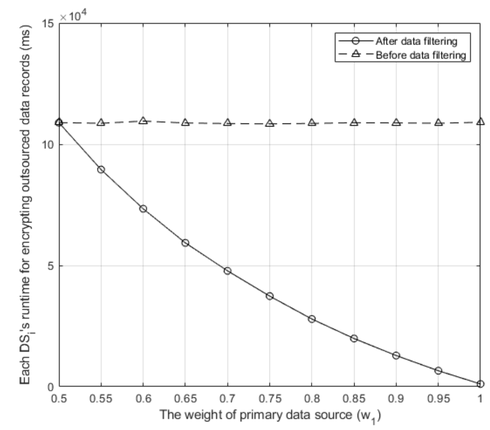


Fig. 2. Each DS_i's runtime for encrypting outsourced data records varying with w_1

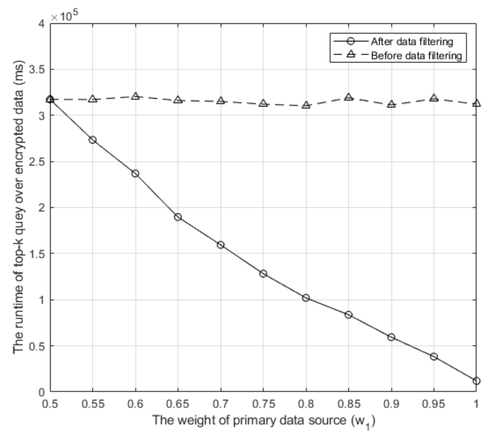


Fig. 3. The runtime of top-k query over encrypted data varying with w_1

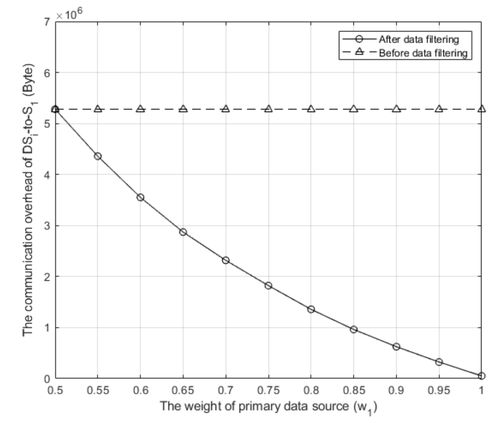


Fig. 4. The communication overhead of DS_i-to- S_1 varying with w_1

*This work has been accepted by IEEE International Conference on Communications 2019.