12-2017

Secure server-aided top-k monitoring

Yujue WANG
Guilin University of Electronic Technology

Hwee Hwa PANG
Singapore Management University, hhpang@smu.edu.sg

Yanjiang YANG
Huawei Singapore Research Center

Xuhua DING
Singapore Management University, xhding@smu.edu.sg

Published in Information Sciences, 2017 December, Volume 420, Pages 345-363
http://doi.org/10.1016/j.ins.2017.08.068

Secure server-aided top-k monitoring

Yujue Wang[a,b,*], HweeHwa Pang[b], Yanjiang Yang[c], Xuhua Ding[b]

[a] *Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China*
[b] *School of Information Systems, Singapore Management University, Singapore 188065, Singapore*
[c] *Huawei Singapore Research Center, Singapore*

ARTICLE INFO

Article history:
Received 29 November 2016
Revised 19 May 2017
Accepted 20 August 2017
Available online 23 August 2017

Keywords:
Privacy
Verifiability
Collusion-resistance
Correlation computation
Vector product

ABSTRACT

In a data streaming model, a data owner releases records or documents to a set of users with matching interests, in such a way that the match in interest can be calculated from the correlation between each pair of document and user query. For scalability and availability reasons, this calculation is delegated to third-party servers, which gives rise to the need to protect the *integrity* and *privacy* of the documents and user queries. In this paper, we propose a *server-aided data stream monitoring scheme* (DSM) to address the aforementioned integrity and privacy challenges, so that the users are able to verify the correlation scores obtained from the server. The scheme provides strong security protection, even in the event of collusion between the server and other users. We also offer techniques to bound the computation demand in decoding the correlation scores, and we demonstrate the practicality of the scheme through experiments with real data.

1. Introduction

With the rapid advances in communication technology and widespread adoption of mobile devices and RFID technology, there is an increasing number of data streaming applications, for example, Web access analysis, profile-driven marketing, environment sensing, stock trading and online bidding [5]. Such applications can generate high volumes of data, which are often streamed to an intermediary (i.e., a Server) for query processing and analysis to produce aggregate results for end-user consumption.

In this paper, we consider a data streaming system that comprises a data Owner, one or more Servers, and multiple Users. Users express their interests as query vectors $\mathbf{q}$ that are stored at the Server. Whenever the Owner wants to release a document, he generates a document vector $\mathbf{d}$ and provides it to the Server. With $\mathbf{d}$, the Server computes the match between $\mathbf{d}$ and each permitted $\mathbf{q}$, in the form of a protected correlation coefficient $v = \mathbf{q} \cdot \mathbf{d}$, and returns v to the issuing User. The User then extracts the correlation score. The benefits of having the Server rather than the User compute the correlation score are twofold. First, it significantly reduces Server-User communication cost, since the User only receives a score rather than the entire document vector. Second, it facilitates access enforcement over the user queries. For instance, low privilege users may not be allowed to search on certain features within the document vector. As the Server may not be trusted by the Owner or the Users, both documents and user queries need to be in *encoded* form to protect their privacy and integrity.

* Corresponding author.
E-mail addresses: wyujue2-c@my.cityu.edu.hk, yjwang@whu.edu.cn (Y. Wang).

In the data stream security literature, verifiability and privacy are generally addressed separately. The former includes [16] which allows users to verify the arrival, removal and update of data falling within a selection range on an attribute, and [10] for authenticating selection-aggregation queries over an attribute of interest. Privacy in data streaming has been achieved through randomization [1] and anonymization [2]. In cryptography, there is predicate encryption that supports the inner product of two vectors, e.g., [9]. However, predicate encryption schemes are not applicable to our data streaming model in which the document vector $\mathbf{d}$ and query vector $\mathbf{q}$ are generated by different parties and must be kept secret from each other.

The only existing scheme that simultaneously guarantees the integrity and privacy of documents and queries in a server-aided data streaming model is our earlier study in [5]. That work proposes a verifiable and private scheme for matching document vectors with query vectors. It is not secure against Server-User collusion though. Also, the scheme is built on composite bilinear group and incurs very high computation costs.

1.1. Our contributions

In this paper, we propose a privacy-preserving, verifiable and collusion-resistant *server-aided data stream monitoring* (DSM) system that matches documents $\mathbf{d}$ with queries $\mathbf{q}$ on their inner product scores $\mathbf{q}\cdot\mathbf{d}$ with the help of a server. In a DSM scheme, the documents $\mathbf{d}$ and standing queries $\mathbf{q}$ are encoded by the Owner and Users, respectively. The Owner does not need to retain any information about the documents after distributing them to the Server, while a User holds only some tag information of his queries. With this limited information, it is very challenging to realize verifiability at the User side, in such a way that the User could successfully recover $\mathbf{q}\cdot\mathbf{d}$ from the Server's encoded result without interacting with the Owner, while leaking no information to the Server. To filter out documents with low inner product scores, the User is able to set a threshold or bound on $\mathbf{q}\cdot\mathbf{d}$, so that only the k highest scores need to be recovered, thus achieving top-k monitoring.

To the best of our knowledge, our scheme is the first for the data streaming model that concurrently achieves strong integrity, privacy, verifiability and collusion-resistance protection. In particular, our DSM scheme achieves all the security features of [5] and more, including:

- **Integrity guarantee on documents and queries**: Although the Server holds all the encoded documents and standing queries, it cannot tamper with them and generate an encoded result for a given pair of document and query, in a way that still leads the User to extract a valid result value.
- **Collusion-resistance against the Server and Users**: Even if the Server colludes with some users, they can neither learn the content of the standing queries of other Users, nor temper with them to yield valid results. In addition, they cannot tamper with the encoded documents.
- **Clear monitoring target**: The Users can check whether an extracted result $\mathbf{q}\cdot\mathbf{d}$ is associated with a fresh document $\mathbf{d}$. In this sense, the Server cannot fool the Users by (re-)sending an old result.

Our scheme is general enough to accommodate the most common document-query matching measures: Both Pearson coefficient and Spearman coefficient can be computed as an inner product of the two vectors concerned if their coordinates are centered around the mean and normalized beforehand. Likewise, the cosine similarity measure is the product of two normalized vectors. Moreover, our scheme is constructed on bilinear group with prime order, which is much faster computationally than the composite bilinear group used in [5]. We confirm the performance differentiation through extensive experiments involving real datasets. The experiments also demonstrate the practicality of our DSM scheme for a broad spectrum of applications.

1.2. Applications

There are potentially many server-aided data stream monitoring applications. For example, in a surveillance scenario [5], the Owner operates various security checkpoints of a country or sensitive installation. A picture of the face is taken of each visitor passing through a checkpoint. From the picture, a feature vector of the relative position, size and shape of the eyes, nose, cheeks, jaw, etc. is automatically extracted. The feature vector forms a document $\mathbf{d}$ that is streamed to a shared Server, along with the document identifier and possibly other sanitized meta-data. One of the Users is an intelligence agency that is monitoring a list of subjects. The agency extracts a feature vector from the picture of each subject, and registers it with the Server as a standing query $\mathbf{q}$. The Server computes the correlation score between $\mathbf{d}$ and $\mathbf{q}$, and returns the score along with the document identifier to the agency. The documents that best match the queries are displayed on the agency's alert screen.

Another application of the data streaming model is social network monitoring. Here, $\mathbf{q}$ represents the profile of a User, which might capture the frequency that he uses certain services or visits certain sites, as well as his ratings for certain products. The User wishes to monitor his 'closest' friends who are online at the Server at any time, as determined by the correlation between $\mathbf{q}$ and the profile $\mathbf{d}$ of each friend who comes online. While the User and his friends are willing to facilitate computing the correlation between their profiles, for privacy reasons they will not release their detailed profiles directly to each other or the Server.

Table 1
Properties of existing security schemes for data streaming model.

Scheme	Query Privacy	Document Privacy	Result Verifiability	Collusion Resistance
Random noise injection [1]	No	Yes	No	No
Data condensation [2]	No	Yes	No	No
Classifier training on data stream [20]	No	Yes	No	No
Keyword search on file stream [4], [14]	Yes	No	No	No
Authenticate streamed data [16]	No	No	Yes	No
Authenticate selection-aggregation queries [10]	No	No	Yes	No
Authenticate aggregation functions [13]	No	No	Yes	No
Public key predicate encryption [9]	Yes	No	No	No
Symmetric key predicate encryption [18]	Yes	Yes	No	No
Authenticate linear algebra queries [15]	No	No	Yes	No
Cryptographic verifiable and private top-k monitoring [5]	Yes	Yes	Yes	No
This paper	Yes	Yes	Yes	Yes

The third application is document filtering. Consider an entrepreneur (User) who wishes to monitor potential competition to a product that he is building. He registers his interest, in the form of a standing search query $\mathbf{q}$, with a business intelligence publisher like Informa (Owner). To support its global clientele, the publisher streams its business news and market analysis reports $\mathbf{d}$ through third-party Servers situated in different geographical regions. Whenever a new report is published, one of the servers would evaluate the report's relevance to the standing query, and send the report summary along with a relevance score to the entrepreneur. The entrepreneur would likely set a score threshold on his client software to filter out irrelevant reports, and read only those summaries with high relevance scores to decide whether to purchase the full reports from the publisher. In this application, document privacy allows the documents to be queried while keeping their content private until purchased by the users. Concurrently, users are assured that their queries (which reflect their interest) remain private, and that they will not be misled into disregarding relevant documents due to the document scores being manipulated by competitors or compromised servers.

1.3. Related work

Data stream security has been studied extensively in various research communities. Table 1 summarizes the characteristics of the most relevant ones in terms of query/document privacy, result verifiability and collusion resistance.

In [11], Lindner and Meier discuss general security concerns in architecting a data stream management system. Other studies have sought to provide targeted security protection for data streams in specific contexts. To safeguard the privacy of data streams, [1] proposes to inject randomized noise. The condensation scheme in [2] to achieve anonymization supports incremental update, and is applicable to data streams. Xu et al. [20] considers how to train a classifier on input streams that are private. Bethencourt et al. [4] and Ostrovsky and Skeith [14] introduce protocols for a server to search a stream for files that contain given query keywords; the protocols protect the privacy of the query, but not the data stream.

Verifiability has been addressed in a different group of studies. They include [16] which allows users to verify the arrival, removal and update of data falling within a selection range, [10] for authenticating selection-aggregation queries over an attribute of interest, and [13] for verifying the output of aggregation functions like MAX and SUM. In network communication, studies such as [7,17] have addressed how to verify the authenticity of stream data in the presence of packet loss. None of the above schemes support the correlation computation that we need. Moreover, they provide verifiability but not privacy protection.

Among existing cryptographic protocols, the ones that support inner product, which is what the common correlation coefficients entail, are most relevant to our problem setting. The rest of this section focuses on those schemes and explains why they do not meet our requirements.

In the predicate encryption scheme proposed in [9], an entity possessing a secret key token associated with a vector $\mathbf{x}$ can decrypt a public key encrypted ciphertext associated with another vector $\mathbf{y}$ on the condition that $\mathbf{x} \cdot \mathbf{y} = \mathbf{0}$. The scheme is not suitable for our data streaming model because it does not protect the privacy of $\mathbf{x}$ against the token holder (the Server in our context); this is because the Server can produce the ciphertext for any chosen $\mathbf{y}$ with the public key in order to match against $\mathbf{x}$. Our model requires the privacy of both vectors against the Server.

The above privacy issue is addressed by the symmetric-key predicate encryption scheme in [18]. Here, generation of both the secret token for $\mathbf{x}$ and the ciphertext associated with $\mathbf{y}$ require secret inputs; this prevents the token holder from generating ciphertext for his chosen $\mathbf{y}$. However, this scheme requires a one-to-one mapping between the secrets used for encrypting $\mathbf{x}$ and $\mathbf{y}$, whereas our data streaming model requires a one-to-many mapping as the Owner's data stream serves multiple Users simultaneously.

Neither the symmetric-key nor public-key predicate encryption scheme supports verifiability. The only way for the User to verify a computation is to repeat it on his own, and compare with the inner product returned by the Server. Moreover, recovering the actual value of the inner product involves performing a discrete logarithm in a huge domain which is computationally infeasible.

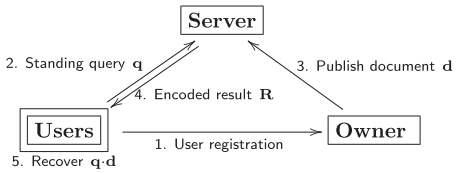

Fig. 1. DSM System Model.

In [15], Papadopoulos et al. reported an interesting scheme for authenticating linear algebra queries over data streams. The setting involves multiple sources that stream data to an outsourced server, and users who register queries over the data streams with the server; the sources and users are assumed to be trusted and share a common secret key. Periodically, the server reports the query results to the users, coupled with authentication proofs. Supported query operations include vector sum, dot product, and matrix product. The scheme does not hide the data from the server.

Ding et al. [5] proposes a scheme for an untrusted server to compute the inner product of an encrypted document vector and an encrypted query vector. The scheme achieves query privacy, document privacy as well as result verifiability, but it is vulnerable to collusion between the server and other users. Computationally, the scheme requires bilinear groups with an order that is the product of two large prime numbers, and hence is very expensive.

Our problem setting is also related to privacy-preserving scalar product schemes such as [6,21]. These schemes are built on an interactive protocol in which two parties, each holding a secret input, co-compute the scalar product without revealing their inputs to each other. They do not permit the computation to be carried out by an untrusted intermediary (the Server) though, a key requirement of our data streaming model. There is also no provision to check whether the scalar product is computed correctly.

1.4. Paper organization

The remainder of this paper is organized as follows. We introduce the DSM system architecture and security requirements in Section 2. The framework and the corresponding security model are formalized in Section 3. Section 4 introduces our DSM scheme and proves its security. Section 5 explains how DSM may be enhanced to achieve query privacy against the data Owner. Section 6 then discusses optimizations that shorten the execution time incurred by the users. In Section 7, we evaluate the DSM scheme both analytically and empirically. Finally, Section 8 concludes the paper.

2. System model

2.1. System architecture

A DSM system, depicted in Fig. 1, comprises a data Owner, one or more Servers, and multiple Users. Each User should register with the Owner to seek permission to monitor the documents published by the Owner. After registration, the Users would have no more direct communication with the Owner. Each User encodes her standing query vectors to safeguard their privacy and integrity; the encoded query vectors are lodged with the Server. Whenever the Owner has a document to release, he generates for it an encoded document vector which is distributed to the Server. Upon receiving a new document vector $\mathbf{d}$, the Server computes an encoded result $\mathbf{R}$ of the correlation coefficient $\mathbf{q} \cdot \mathbf{d}$ for each query $\mathbf{q}$, and returns $\mathbf{R}$ to the issuing User. The User then decodes the output to obtain and verify the correlation score. The User is able to perform top-k monitoring to discard low scores without recovering them. The correlation score may reflect the similarity or association between $\mathbf{q}$ and $\mathbf{d}$, which depends on the application scenario. For example, in the surveillance scenario elaborated in Section 1.2, the correlation score measures the match between the pictures of a monitored subject and visitors. Based on this score, the k best matching visitor pictures at any time would be highlighted to the agency. In the system, the Server would always respond to all qualified Users when a fresh document is published to the Server.

We assume that temporary secure channels from the Owner to User and from the Owner to Server are available during the registration phase for transmitting the user key and server-side key. The Server implements access policies over the user queries, which may restrict each User to issuing fewer than some maximum m standing queries for matching against the same document, and also limit each User to accessing certain parts of the published documents (as explained in Section 1). Our DSM system does not require the `ServPro` procedure to perform any semantic/sanity checks on encoded standing queries and documents, whereas semantic/sanity checking is an independent problem. That is, our DSM scheme only requires that all the procedures work well when the standing queries and documents are correctly formulated, i.e., they are represented as vectors on finite field.

2.2. Adversary model and security goals

As the Server and the other users in the system may be administrated by outsourced service providers whose operators do not have appropriate security clearances, the system must be secure against any Server-User collusion. That is, in a

DSM system, the adversary could be a Server, acting in collusion with corrupted users, that executes the protocol honestly but is curious to know the document features and user queries, or cheats in computing the correlation scores where the corresponding standing queries and documents may have been tampered with. Accordingly, the security requirements against the Server-User collusion are enumerated below, whereas Server-Owner collusion will be discussed in Section 5.

- *Query privacy and integrity*. The values in **q** must be known only to the particular User who issued it, and guarded against any collusion between the Server and other users. For example, a surveillance application should guarantee that an adversary cannot match feature vectors extracted from his own pictures against **q** to discover who the intelligence agency is monitoring.
- *Document privacy and integrity*. The values in **d** must not be revealed to the Server, so as to prevent an adversary from matching **d** against his own queries.[1] Document integrity should be preserved against Server-User collusion attacks. For example, in surveillance applications, to safeguard the rights of the visitors who have their pictures taken, users of the system must be explicitly authorized by the Owner before they can query over **d**.
- *Result privacy*. The correlation score between **d** and **q** should be available only to the issuing User. With or without other colluding users, the Server that computes the score cannot be allowed to deduce the query result.
- *Result verifiability*. Without direct access to **d**, the User who issued **q** would want to verify the correlation score between **d** and **q**. This is to prevent the Server and other colluding users from manipulating **d, q** or the query processing to suppress certain documents from the User.

Remark 1. We elaborate further on a few key properties of the DSM system.

(1) The Users need to receive one message for every document. This cannot be avoided because query privacy prevents the Owner/Server from computing the correlation score and filtering low-scoring documents. The model reduces data traffic to the User as long as the document vector is larger than the encrypted correlation score.
(2) The Owner is trusted by the Users. This is reasonable because the Users are consuming documents generated by the Owner. Nevertheless, for those rare situations where the Owner may be curious about the User queries, we propose a variation of our solution that provides query privacy against even the Owner.
(3) Resistance against Server-User collusion. Even if the Server colludes with enough Users, the most that they can learn is the content of the document vectors. Query privacy, as well as integrity of the correlation scores, remain intact.

3. Definitions

3.1. Framework of DSM system

A *data stream monitoring* (DSM) system involves three types of entities – Owner, User and Server. Formally, a DSM scheme consists of the following six procedures.

- $\mathtt{Setup}(1^\ell, m) \to (pk, sk)$: On input 1^ℓ where ℓ is a security parameter, and dimensionality m of the query and document vectors, the set-up algorithm, which is carry out by the Owner, generates a pair of public/secret keys (pk, sk).
- $\mathtt{UserReg}(pk, sk) \to (uk, ssk_u)$: On input a pair of public/secret keys (pk, sk), the user registration algorithm, which is carried out by the Owner, outputs for the User a user key uk and a matching server-side key ssk_u that is given to the Server.
- $\mathtt{QueryGen}(pk, uk, \mathbf{q}) \to (\mathbf{Q_u}, \mathbf{T_q})$: On input a public key pk, a user key uk and a standing query **q** where the coordinates reflect the User's interests, the query generation algorithm, which is run by the User, outputs an encoded query $\mathbf{Q}_u$ and a secret parameter $T_\mathbf{q}$ for **q**.
- $\mathtt{DocGen}(pk, sk, \mathbf{d}) \to \tilde{\mathbf{D}}$: On input a pair of public/secret keys (pk, sk) and a document vector **d**, the document generation algorithm, which is carried out by the Owner, outputs a processed document $\tilde{\mathbf{D}}$ which contains an encoded document vector **D** and an unique identifier $id_\mathbf{d}$, and may also contain some other parameters related to **d**.
- $\mathtt{ServPro}(\tilde{\mathbf{D}}, ssk_u, \mathbf{Q_u}) \to \mathbf{R}$: On input a processed document $\tilde{\mathbf{D}}$, a server-side key ssk_u and an encoded query $\mathbf{Q}_u$, the server processing algorithm, which is carried out by the Server, outputs an encoded query result **R** on **q** and **d** for the issuing User.
- $\mathtt{UserDec}(pk, uk, \mathbf{R}, T_\mathbf{q}) \to v/\perp$: On input a public key pk, a user key uk, an encoded query result **R** and a secret parameter $T_\mathbf{q}$, the user decoding algorithm, which is run by the User, outputs a query result $v = \mathbf{q} \cdot \mathbf{d}$ or $\perp$.

Usage of DSM algorithms: Fig. 2 depicts how the DSM procedures are invoked. The Owner, who is the document publisher, first runs Setup to generate his own secret key and to initialize the system setting shared by all participants. To join the system, a User should register with the Owner. The Owner executes UserReg to produce a secret that the

[1] With query privacy protection, a user could formulate queries to elicit individual feature values in **d**. That is, when a user colludes with the Server or some other users, they can issue m queries $\mathbf{q}_i$ for $i \in [1, m]$ and get correlation scores v_i, where m denotes the dimensionality of the query and document vectors; then they would be able to deduce all the values in **d** by solving the system of equations $\mathbf{q_i} \cdot \mathbf{d} = v_i$. Hence document privacy is not achievable against colluding users.

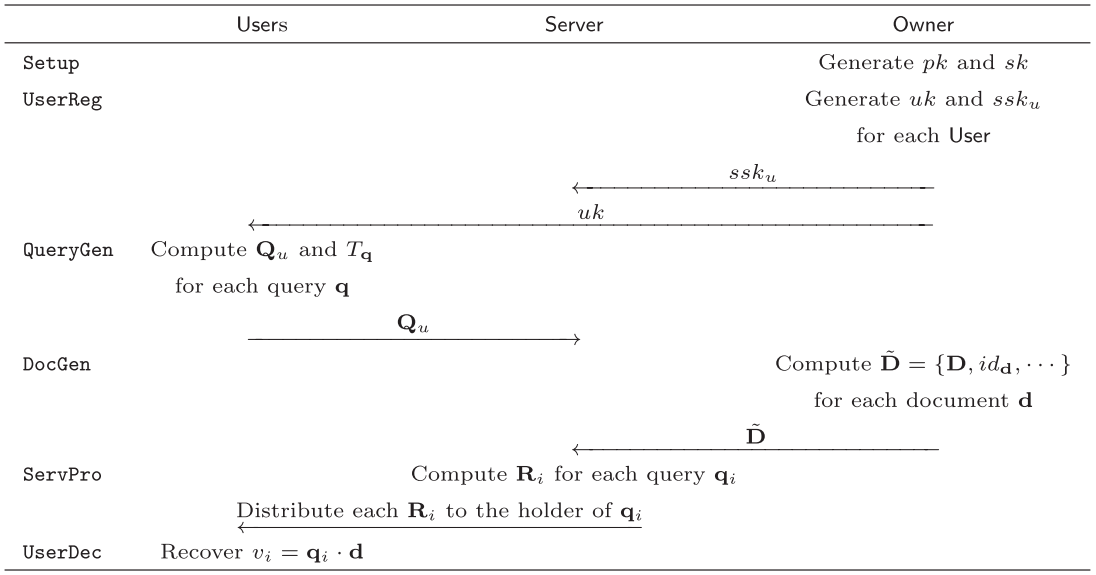

Fig. 2. A procedure of DSM scheme.

User utilizes to encode her queries, and a secret that enables the Server to process those queries. Based on her interest, the User runs QueryGen to generate the encoding $\mathbf{Q}_u$ of her query $\mathbf{q}$ and deposits $\mathbf{Q}_u$ with the Server, and locally retains secret parameter $T_{\mathbf{q}}$. No two users share the same secret. This ensures that encoded queries are user-specific.

At runtime, the Owner generates a stream of documents as well as their document vectors. For each new document vector $\mathbf{d}$, the Owner runs DocGen to produce a processed document $\tilde{\mathbf{D}}$ for the Server; $\tilde{\mathbf{D}}$ is common to *all* the Users in the system. The Server then executes ServPro on $\tilde{\mathbf{D}}$ for each registered query $\mathbf{Q}_u$ and returns the encoded correlation score $\mathbf{R}$ to the corresponding User. On receiving the response, the User runs UserDec with her secret to recover v within some target range. In top-k monitoring, the User only wants the k highest ranking documents, so the target range is bounded from below by the score of the kth ranked document. Thus, top-k monitoring actually improves the efficiency of UserDec. We will provide constraints that bound the search space for v, so that its decoding is computationally efficient.

When all the entities in the DSM system are honest, it should always output the correct query result.

Definition 3.1 (Correctness)**.** A DSM scheme is *correct* if, for any security parameter $\ell \in \mathbb{N}$, any dimensionality $m \in \mathbb{N}$, any key pair $(pk, sk) \leftarrow \mathtt{Setup}(1^\ell, m)$, any user registration $(uk, ssk_u) \leftarrow \mathtt{UserReg}(pk, sk)$, any encoded standing query $(\mathbf{Q_u}, \mathbf{T_q}) \leftarrow \mathtt{QueryGen}(\mathbf{pk}, \mathbf{uk}, \mathbf{q})$ for $\mathbf{q}$, and any processed document $\tilde{\mathbf{D}} \leftarrow \mathtt{DocGen}(pk, sk, \mathbf{d})$ for $\mathbf{d}$, we have $\mathtt{UserDec}(pk, uk, \mathbf{R}, T_{\mathbf{q}}) = \mathbf{q} \cdot \mathbf{d}$ for the server-generated encoded query result $\mathbf{R} \leftarrow \mathtt{ServPro}(\tilde{\mathbf{D}}, ssk_u, \mathbf{Q_u})$.

3.2. Security definitions

We proceed to define security models to achieve the security goals described in Section 2.2. Our DSM system involves a computation Server, which is similar to existing outsourcing/verifiable computation schemes. Thus, we follow the standard framework established for these schemes (e.g., [19,22]) to define the security model for integrity of user queries and documents.

In a secure DSM system, even when the Server colludes with a set U of users, it cannot tamper with an encoded standing query $\mathbf{Q}_u$ outsourced by a honest User without being caught. In other words, if some $\mathbf{Q}_u$ issued by a honest User has been tampered with, it can be detected by this User so that the UserDec algorithm would output $\perp$ to indicate a failure; if $\mathbf{Q}_u$ remains intact, then UserDec must output the expected value v. Formally, the *integrity* of encoded standing query $\mathbf{Q}_u$ is captured by the following security game $\mathrm{Game}_{\mathcal{A}}^{q\text{-}int}$ between a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Adversary $\mathcal{A}$ controls the Server and all Users in the corrupted set U, whereas challenger $\mathcal{C}$ simulates the Owner and all the honest Users.

Setup: With security parameter ℓ and dimensionality m, the challenger $\mathcal{C}$ runs $\mathtt{Setup}(1^\ell, m)$ to get a pair of public/secret keys (pk, sk). For each user in U, the challenger runs $\mathtt{UserReg}(pk, sk)$ to generate a user key uk_i and a server-side key ssk_i. It gives pk and $\{ssk_i\}_{\mathrm{U}}$ to $\mathcal{A}$.

Queries: In this phase, adversary $\mathcal{A}$ adaptively poses the following queries. Challenger $\mathcal{C}$ maintains query lists that record all the queries and responses.

- *Query generation queries*: Adversary $\mathcal{A}$ requests an encoded query vector for a standing query $\mathbf{q}_{l,i}$ on behalf of some user $U_l \notin \mathrm{U}$. If U_l does not exist in the query list, the challenger runs $\mathtt{UserReg}(pk, sk)$ to generate a user key uk_l and a

server-side key ssk_l; otherwise $\mathcal{C}$ retrieves uk_l and ssk_l generated previously for U_l. Following that, the challenger runs $\mathbf{Q_{l,i}} \leftarrow \texttt{QueryGen}(\mathbf{pk}, \mathbf{uk_l}, \mathbf{q_{l,i}})$, then sends $\mathbf{Q}_{l,\,i}$ and ssk_l to $\mathcal{A}$.

- *Document generation queries*: Adversary $\mathcal{A}$ requests a processed document for $\mathbf{d}_j$. The challenger runs $\tilde{\mathbf{D}}_j \leftarrow \texttt{DocGen}(pk, sk, \mathbf{d_j})$ and sends $\tilde{\mathbf{D}}_j$ to $\mathcal{A}$.
- *Decoding queries*: Adversary $\mathcal{A}$ generates an encoded query result $\mathbf{R}_{l,i,j}$ for user U_l's standing query $\mathbf{q}_{l,\,i}$ and document $\mathbf{d}_j$, and sends $\mathbf{R}_{l,i,j}$ to $\mathcal{C}$. The challenger responds with a result $v_{l,\,i,\,j}$ or $\perp$ by running $\texttt{UserDec}(pk, uk_l, \mathbf{R}_{l,i,j}, T_{\mathbf{q_{l,i}}})$.

Output: Eventually, the adversary outputs a pair of indices $(\hat{\imath}, \hat{l})$ that corresponds to query $\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}$ of user $U_{\hat{l}} \notin \mathtt{U}$ in the query list and wins the game if it can forge the correlation score between $\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}$ and a document in a round of data monitoring. That is, the challenger randomly picks a fresh document $\mathbf{d}$, generates $\tilde{\mathbf{D}} \leftarrow \texttt{DocGen}(pk, sk, \mathbf{d})$ and sends $\tilde{\mathbf{D}}$ to $\mathcal{A}$, and the adversary responds with $\mathbf{R}'$ such that:

$$\texttt{UserDec}(pk, uk_{\hat{l}}, \mathbf{R}', T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}}) \neq \texttt{UserDec}(pk, uk_{\hat{l}}, \mathbf{R}, T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}})$$

where $\mathbf{R} \leftarrow \texttt{ServPro}(\tilde{\mathbf{D}}, ssk_{\hat{l}}, \mathbf{Q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}})$ is generated with information that $\mathcal{C}$ maintains locally.

Let $\text{Prob}_{\mathcal{A}}^{q\text{-}int}$ be the probability that adversary $\mathcal{A}$ wins security game $\text{Game}_{\mathcal{A}}^{q\text{-}int}$, taken over the coin tosses made by both $\mathcal{A}$ and $\mathcal{C}$.

Definition 3.2 (Integrity of User Queries)**.** A DSM scheme guarantees the integrity of user queries against the Server and colluding users if, for any PPT adversary $\mathcal{A}$ who carries out security game $\text{Game}_{\mathcal{A}}^{q\text{-}int}$ with a challenger $\mathcal{C}$, there exists a negligible function $\epsilon(\cdot)$ such that $\text{Prob}_{\mathcal{A}}^{q\text{-}int} \leq \epsilon(1^\ell)$.

Also, when the Server colludes with a set U of users, to them the processed queries produced by other users should still be indistinguishable from each other.

Definition 3.3 (Privacy of User Queries)**.** A DSM scheme preserves the *privacy of user queries* against a colluding Server and a set U of users if, for any $\ell \in \mathbb{N}$, any dimensionality $m \in \mathbb{N}$ and any public/secret key pair $(pk, sk) \leftarrow \texttt{Setup}(1^\ell, m)$, any registration $(uk, ssk_u) \leftarrow \texttt{UserReg}(pk, sk)$ of a honest User, the following two distributions of any two standing queries $\mathbf{q}_1$ and $\mathbf{q}_2$ appear identical to the colluding Server and users:

$$\left\{ \mathbf{Q}_1 : \begin{array}{l} (uk_i, ssk_i) \leftarrow \texttt{UserReg}(pk, sk) \text{ for each } U_i \in \mathtt{U} \\ (\mathbf{Q}_1, T_{\mathbf{q}_1}) \leftarrow \texttt{QueryGen}(pk, uk, \mathbf{q}_1) \end{array} \right\} \approx \left\{ \mathbf{Q}_2 : \begin{array}{l} (uk_i, ssk_i) \leftarrow \texttt{UserReg}(pk, sk) \text{ for each } U_i \in \mathtt{U} \\ (\mathbf{Q}_2, T_{\mathbf{q}_2}) \leftarrow \texttt{QueryGen}(pk, uk, \mathbf{q}_2) \end{array} \right\}$$

We continue to formalize the security requirements for published documents. In a secure DSM system, the Server cannot forge or tamper with a processed document $\tilde{\mathbf{D}}$ of the Owner by colluding with a set U of users without being caught. In other words, if some $\tilde{\mathbf{D}}$ issued by the Owner has been tampered with, it can be detected by the registered Users so that the UserDec algorithm would output $\perp$ to indicate a failure; if $\tilde{\mathbf{D}}$ is used in ServPro without being changed, then UserDec must output the expected value v. Formally, the *integrity* of processed document $\tilde{\mathbf{D}}$ is captured by the following security game $\text{Game}_{\mathcal{A}}^{d\text{-}int}$ between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Similar to $\text{Game}_{\mathcal{A}}^{q\text{-}int}$, adversary $\mathcal{A}$ controls the Server and all Users in the corrupted set U, whereas challenger $\mathcal{C}$ simulates the Owner and all the honest Users.

Setup: The same as in $\text{Game}_{\mathcal{A}}^{q\text{-}int}$.

Queries: The same as in $\text{Game}_{\mathcal{A}}^{q\text{-}int}$.

Output: Eventually, the adversary outputs a tuple $(\mathbf{R}', id_{\mathbf{d}_{\hat{\mathbf{j}}}}, \hat{j}, \hat{l})$, which indicates the adversary forges a processed document for a queried $\mathbf{d}_{\hat{\mathbf{j}}}$, and $\mathbf{R}'$ is an encoded result for $\mathbf{d}_{\hat{\mathbf{j}}}$ and $\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{j}}}$ in the query list. The adversary wins the game if:

$$\texttt{UserDec}(pk, uk, \mathbf{R}', T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{j}}}}) \neq \texttt{UserDec}(pk, uk, \mathbf{R}, T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{j}}}})$$

where $\mathbf{R} \leftarrow \texttt{ServPro}(\tilde{\mathbf{D}}_{\hat{\imath}}, ssk_{\hat{l}}, \mathbf{Q}_{\hat{\mathbf{l}},\hat{\mathbf{j}}})$ is generated with information that $\mathcal{C}$ maintains locally.

Let $\text{Prob}_{\mathcal{A}}^{d\text{-}int}$ be the probability that adversary $\mathcal{A}$ wins security game $\text{Game}_{\mathcal{A}}^{d\text{-}int}$, taken over the coin tosses made by both $\mathcal{A}$ and $\mathcal{C}$.

Definition 3.4 (Integrity of Documents)**.** A DSM scheme guarantees the integrity of documents against the Server and colluding users if, for any PPT adversary $\mathcal{A}$ who carries out security game $\text{Game}_{\mathcal{A}}^{d\text{-}int}$ with a challenger $\mathcal{C}$, there exists a negligible function $\epsilon(\cdot)$ such that $\text{Prob}_{\mathcal{A}}^{d\text{-}int} \leq \epsilon(1^\ell)$.

Similarly, in a DSM scheme, the processed documents should be indistinguishable to the Server, even though it holds a server-side key.

Definition 3.5 (Privacy of Documents)**.** A DSM scheme preserves the *privacy of documents* against the Server if, for any $\ell \in \mathbb{N}$, any dimensionality $m \in \mathbb{N}$ and any public/secret key pair $(pk, sk) \leftarrow \texttt{Setup}(1^\ell, m)$, any user registration $(uk, ssk_u) \leftarrow \texttt{UserReg}(pk, sk)$, the following two distributions of any two documents $\mathbf{d}_1$ and $\mathbf{d}_2$ appear identical to the Server:

$$\left\{ \mathbf{D_1} : \tilde{\mathbf{D}}_1 \leftarrow \texttt{DocGen}(pk, sk, \mathbf{d_1}) \right\} \approx \left\{ \mathbf{D_2} : \tilde{\mathbf{D}}_2 \leftarrow \texttt{DocGen}(pk, sk, \mathbf{d_2}) \right\}$$

Clearly, if query privacy is protected against the colluding Server and users in a DSM system, then *result privacy* would also be achieved. Also, *result verifiability* is a basic functionality of a secure DSM system that requires no formalization.

Table 2
Notation.

Symbol	Meaning
$\mathbf{d}$	Document vector
k_d	Bit length of each coordinate in $\mathbf{d}$
$\mathbf{q}$	Query vector
k_q	Bit length of each coordinate in $\mathbf{q}$
m	Dimensionality of $\mathbf{d}$ and $\mathbf{q}$
m_q	Number of coordinates that the User specified in $\mathbf{q}$
v	$v = \mathbf{q} \cdot \mathbf{d}$ is the score of $\mathbf{d}$ given $\mathbf{q}$
$\mathbb{G}, \mathbb{G}_T$	Cyclic groups with bilinear mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
p	Large prime number that is the order of $\mathbb{G}$ and $\mathbb{G}_T$
$H(.)$	A one-way, collision-resistant hash function

4. Privacy-Preserving, verifiable and collusion-Resistant scheme

This section introduces our privacy-preserving, verifiable and collusion-resistant DSM scheme. Table 2 summarizes the frequently used notations, which will be explained as they are used.

Our DSM scheme is constructed on bilinear groups. Let $\mathbb{G}$ be a cyclic group of order p with generator g. A **bilinear map** is a mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is a cyclic group of order p, with the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Computability: $\forall u, v \in \mathbb{G}$, $\hat{e}(u, v)$ can be computed efficiently, in complexity that is polynomial in $\log p$.
- Non-degeneracy: $\hat{e}(g, g) \neq 1$.

4.1. Solution construction

In our system, document vectors and query vectors contain m coordinates each. In any document or query vector, a large number of coordinates are expected to be zero. Special precaution is needed to ensure an adversary cannot deduce that two coordinates in a vector have zero values. To this end, $4m + 4$ and $2m$ random values are respectively used in processing a standing query and a document with dimensionality m.

A round of document monitoring involves three steps, i.e., document generation (by the Owner) – Server processing – User decoding. Thus, the User has no direct communication with the Owner, and receives only the encoded query result $\mathbf{R}$ from the Server for the fresh document $\mathbf{d}$. With $\mathbf{R}$, the User must be able to decode $v = \mathbf{q} \cdot \mathbf{d}$ and verify its correctness. On one hand, we let the Owner sign the parameters of each published document $\mathbf{d}$. The signature $\chi_{\mathbf{d}}$ is sent to the Server and then forwarded to the User. The unforgeability and verifiability of $\chi_{\mathbf{d}}$ ensure that the Server cannot tamper with those parameters. On the other hand, each query $\mathbf{q}$ and document $\mathbf{d}$ are encoded into two independent parts (i.e., $(\{Q_{i,1}, \cdots, Q_{i,4}\}_{i=1}^{m}, Q_9)$ and $(\{Q_{i,5}, \cdots, Q_{i,8}\}_{i=1}^{m}, Q_{10})$, $(\{D_{i,1}, \cdots, D_{i,4}\}_{i=1}^{m}, D_9)$ and $(\{D_{i,5}, \cdots, D_{i,8}\}_{i=1}^{m}, D_{10})$, respectively), so that the Server's encoded query result $\mathbf{R}$ would also contain two independent elements W_1 and W_2. The result $v = \mathbf{q} \cdot \mathbf{d}$ is valid only if both W_1 and W_2 decode to v.

We now present our construction.

$\underline{\texttt{Setup}(1^\ell, m)}$: Choose a bilinear mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are cyclic groups of prime order p. Randomly select $\theta \in_R \mathbb{Z}_p^*$ and a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$. Randomly pick distinct $\sigma_{i,j} \in_R \mathbb{Z}_p^*$ for $1 \le i \le m$ and $1 \le j \le 6$, and $\alpha_j \in_R \mathbb{Z}_p^*$ for $1 \le j \le 4$. Pick a signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and invoke $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathcal{S}.\mathsf{KeyGen}$. Thus, the private key is $sk = (\theta, \{\sigma_{i,1}, \cdots, \sigma_{i,6}\}_{i=1}^{m}, \{\alpha_j\}_{j=1}^{4}, \mathsf{tsk})$ and the public key is $pk = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, H, \mathsf{tpk})$.

$\underline{\texttt{UserReg}(pk, sk)}$: Randomly select $a_u \in_R \mathbb{Z}_p^*$ and compute $b_u = \theta - a_u \bmod p$, $\Omega_u = g^{a_u}$ and $\Psi_u = g^{b_u}$. Compute $g^{1/\sigma_{i,j}}$ for $1 \le i \le m$ and $1 \le j \le 6$, and compute g^{1/α_j} for $1 \le j \le 4$. Thus, $ssk_u = \Psi_u$ and $uk = (\Omega_u, \{g^{1/\sigma_{i,1}}, \cdots, g^{1/\sigma_{i,6}}\}_{i=1}^{m}, \{g^{1/\alpha_j}\}_{j=1}^{4})$.

$\underline{\texttt{QueryGen}(pk, uk, \mathbf{q})}$: For a query vector $\mathbf{q} = \{\mathbf{q_i}\}_{\mathbf{i=1}}^{\mathbf{m}}$ where $q_i \in [0, 2^{k_q}) \subseteq \mathbb{Z}_p$, randomly select $\tau_1, \tau_2, \tau_3, \tau_4 \in_R \mathbb{Z}_p^*$, and $\mu_{i,j} \in_R \mathbb{Z}_p^*$ for $1 \le i \le m$ and $1 \le j \le 4$. For $1 \le i \le m$, compute

$$
\begin{aligned}
Q_{i,1} &= g^{(\tau_1 q_i + \tau_2 + \mu_{i,1})/\sigma_{i,1}}, & Q_{i,2} &= g^{\mu_{i,1}/\sigma_{i,2}} \\
Q_{i,3} &= g^{(\tau_1 q_i + \tau_2 + \mu_{i,2})/\sigma_{i,3}}, & Q_{i,4} &= g^{\mu_{i,2}/\alpha_1} \\
Q_{i,5} &= g^{(\tau_3 q_i + \tau_4 + \mu_{i,3})/\sigma_{i,4}}, & Q_{i,6} &= g^{\mu_{i,3}/\sigma_{i,5}} \\
Q_{i,7} &= g^{(\tau_3 q_i + \tau_4 + \mu_{i,4})/\sigma_{i,6}}, & Q_{i,8} &= g^{\mu_{i,4}/\alpha_2}
\end{aligned}
$$

as well as

$$Q_9 = g^{\tau_2/\alpha_3} \text{ and } Q_{10} = g^{\tau_4/\alpha_4}.$$

Thus, $\mathbf{Q_u} = (\{\mathbf{Q_{i,1}}, \cdots, \mathbf{Q_{i,8}}\}_{\mathbf{i=1}}^{\mathbf{m}}, \mathbf{Q_9}, \mathbf{Q_{10}})$ and $T_{\mathbf{q}} = (\tau_1, \cdots, \tau_4, \sum_{i=1}^{m} q_i, \sum_{i=1}^{m} \mu_{i,1}, \sum_{i=1}^{m} \mu_{i,3})$.

$\underline{\texttt{DocGen}(pk, sk, \mathbf{d})}$: For each document vector $\mathbf{d} = \{\mathbf{d_i}\}_{\mathbf{i=1}}^{\mathbf{m}}$ where $d_i \in [0, 2^{k_d}) \subseteq \mathbb{Z}_p$, randomly select $r \in_R \mathbb{Z}_p^*$ and compute $C = g^r$. Randomly pick $\beta_1, \beta_2 \in_R \mathbb{Z}_p^*$ and $h \in_R \mathbb{G}$, and compute

$$E_1 = \hat{e}(h, g),\ E_2 = \hat{e}(h, g)^{\beta_1} \text{ and } E_3 = \hat{e}(h, g)^{\beta_2}.$$

Randomly pick a unique identifier $id_{\mathbf{d}} \in_R \mathbb{Z}_p^*$ and, for $1 \le j \le 4$, compute $\phi_j = H(id_{\mathbf{d}} \| j \| \hat{e}(C, g^{\theta}))$, followed by

$$\chi_{\mathbf{d}} = \mathcal{S}.\mathsf{Sign}_{\mathsf{tsk}}(id_{\mathbf{d}} \| C \| \phi_1 \| \cdots \| \phi_4 \| E_1 \| E_2 \| E_3)$$

Randomly choose $\lambda_{i,1}, \lambda_{i,2} \in_R \mathbb{Z}_p^*$ for $1 \le i \le m$. For $1 \le i \le m$, compute

$$\begin{aligned} D_{i,1} &= h^{\sigma_{i,1}(d_i + \lambda_{i,1} + \phi_1)}, & D_{i,2} &= h^{\sigma_{i,2}(d_i + \lambda_{i,1} + \phi_2)} \\ D_{i,3} &= h^{\sigma_{i,3}\lambda_{i,1}}, & D_{i,4} &= h^{\alpha_1 \lambda_{i,1}} \\ D_{i,5} &= h^{\sigma_{i,4}(d_i + \lambda_{i,2} + \phi_3)}, & D_{i,6} &= h^{\sigma_{i,5}(d_i + \lambda_{i,2} + \phi_4)} \\ D_{i,7} &= h^{\sigma_{i,6}\lambda_{i,2}}, & D_{i,8} &= h^{\alpha_2 \lambda_{i,2}} \end{aligned}$$

as well as

$$D_9 = h^{\alpha_3(\beta_1 + \sum_{i=1}^{m} d_i)} \text{ and } D_{10} = h^{\alpha_4(\beta_2 + \sum_{i=1}^{m} d_i)}.$$

Thus, $\tilde{\mathbf{D}} = (id_{\mathbf{d}}, \chi_{\mathbf{d}}, \mathbf{D}, \mathbf{C}, \mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3})$ where $\mathbf{D} = (\{\mathbf{D_{i,1}}, \cdots, \mathbf{D_{i,8}}\}_{\mathbf{i=1}}^{\mathbf{m}}, \mathbf{D_9}, \mathbf{D_{10}})$. Here, $\mathbf{D}$ is the common encoded document vector for all the users in the system.

$\underline{\texttt{ServPro}(\tilde{\mathbf{D}}, ssk_u, \mathbf{Q_u})}$: Perform the following steps:

(1) Compute W_1:

$$W_1 = \prod_{i=1}^{m} \frac{\hat{e}(D_{i,1}, Q_{i,1})\hat{e}(D_{i,4}, Q_{i,4})}{\hat{e}(D_{i,2}, Q_{i,2})\hat{e}(D_{i,3}, Q_{i,3})} \Big/ \hat{e}(D_9, Q_9) \tag{1}$$

(2) Compute W_2:

$$W_2 = \prod_{i=1}^{m} \frac{\hat{e}(D_{i,5}, Q_{i,5})\hat{e}(D_{i,8}, Q_{i,8})}{\hat{e}(D_{i,6}, Q_{i,6})\hat{e}(D_{i,7}, Q_{i,7})} \Big/ \hat{e}(D_{10}, Q_{10}) \tag{2}$$

(3) Compute $C_1 = \hat{e}(C, \Psi_u)$.

Thus, $\mathbf{R} = (id_{\mathbf{d}}, \chi_{\mathbf{d}}, W_1, W_2, C, C_1, E_1, E_2, E_3)$.

Remark 2. As Theorem 4.1 will show, component W_1 in $\mathbf{R}$ embeds secret values τ_1, τ_2 and $\{\mu_{i,1}\}$ that the User keeps. Likewise, component W_2 embeds τ_3, τ_4 and $\{\mu_{i,3}\}$. Hence even if some d_i is disclosed through a collusion between the Server and other users, the privacy of every q_i value in the User's query $\mathbf{Q}_u$ remains intact.

Even with knowledge of $\{d_i\}$ and ϕ_1, ϕ_3 through a colluding user, an adversary cannot temper with $\{q_i\}$ consistently in W_1 and W_2 because of secret values τ_1 and τ_3. The adversary cannot temper with d_i consistently in W_1 and W_2 without secret values τ_1 to τ_4. Lastly, the adversary cannot temper with the result $\sum_{i=1}^{m} d_i q_i$ consistently in W_1 and W_2 without τ_1 and τ_3.

$\underline{\texttt{UserDec}(pk, uk, \mathbf{R}, T_{\mathbf{q}})}$: Carry out the following steps:

(1) Compute $C_2 = \hat{e}(C, \Omega_u)$.
(2) Compute $\phi_j = H(id_{\mathbf{d}} \| j \| C_1 \times C_2) = H(id_{\mathbf{d}} \| j \| \hat{e}(C, g^{\theta}))$ for $1 \le j \le 4$.
(3) Validate $\chi_{\mathbf{d}}$ on $id_{\mathbf{d}} \| C \| \phi_1 \| \cdots \| \phi_4 \| E_1 \| E_2 \| E_3$ using tpk. If it is invalid, output $\perp$ and abort.
(4) Compute $R_1 = \phi_1 \tau_1 \sum_{i=1}^{m} q_i + m\phi_1 \tau_2 + (\phi_1 - \phi_2) \sum_{i=1}^{m} \mu_{i,1} \mod p$.
(5) Compute $W = W_1 \times E_2^{\tau_2} / E_1^{R_1} = E_1^{\tau_1 \sum_{i=1}^{m} d_i q_i}$.
(6) Find $v \in [0, 2^{k_d + k_q} \cdot m_q)$ such that $(E_1^{\tau_1})^v = W$ and m_q is the number of coordinates used in $\mathbf{q}$.
(7) Compute $R_2 = \phi_3 \tau_3 \sum_{i=1}^{m} q_i + m\phi_3 \tau_4 + (\phi_3 - \phi_4) \sum_{i=1}^{m} \mu_{i,3} \mod p$.
(8) Verify that $E_1^{\tau_3 v + R_2} / E_3^{\tau_4} \stackrel{?}{=} W_2$. If it holds, output v; otherwise output $\perp$.

Remark 3. With each coordinate in $\mathbf{d}$ and $\mathbf{q}$ having bit length k_d and k_q respectively, v is in the range $[0, 2^{k_d + k_q} \cdot m_q)$ where $m_q \le m$ is the number of coordinates that the User specified in $\mathbf{q}$. This range is expected to be much smaller than the size of the ciphertext space, p, making the recovery of v computationally feasible. We will show in Section 6 how to speed up the computation of v. If v has a high value, indicating a close match between $\mathbf{d}$ and $\mathbf{q}$, the User may request for the actual document from the Owner; otherwise, the User ignores the document.

Remark 4. The verification in Algorithm UserDec hinges on the following facts:

- If there are colluding users, the Server may discover the values of d_i and ϕ_1 through ϕ_4. However, the Server still cannot deduce $h^{\sigma_{i,j}}$ (for $1 \le j \le 6$) and h^{α_j} (for $1 \le j \le 4$), which are needed in order to create encrypted vectors $\mathbf{D}$ for forged documents.

- The Server is unable to compose a fake document vector from multiple document vectors, because the one-way hash function $H(.)$ prevents the Server from finding a corresponding C for the fake document vector.
- The Server is unable to tamper with the query vector because it embeds the values $\tau_1, \tau_2, \tau_3, \tau_4, \sum_{i=1}^{m} q_i$, $\sum_{i=1}^{m} \mu_{i,1}, \sum_{i=1}^{m} \mu_{i,3}$ that are retained by the User.
- The Server has a negligible chance of tampering with the sum $\sum_{i=1}^{m} d_i q_i$ consistently in W_1 and W_2, without the User's secret values τ_1 and τ_3.
- The position-specific secret values $\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, \sigma_{i,4}, \sigma_{i,5}, \sigma_{i,6}$ in $\mathbf{D}$ and $\mathbf{Q}_u$ ensure that, for any $i \in [1, m]$, $D_{i,1}$ must be paired with $Q_{i,1}$, and $D_{i,5}$ with $Q_{i,5}$, to produce $d_i q_i$. Pairing $D_{i,1}$ and $D_{i,5}$ with $Q_{j,1}$ and $Q_{j,5}$, for any $j \neq i$, will produce random results that fail the user verification. The values $E_2 = \hat{e}(h,g)^{\beta_1}$ and $E_3 = \hat{e}(h,g)^{\beta_2}$ provided by the Owner, together with the sums $\sum_{i=1}^{m} q_i$, $\sum_{i=1}^{m} \mu_{i,1}$, $\sum_{i=1}^{m} \mu_{i,3}$ retained by the User, ensure that W_1 and W_2 are aggregated over all the coordinates $i \in [1, m]$.

Theorem 4.1. *The above DSM scheme is correct.*

Proof. The correctness on $\chi_{\mathbf{d}}$ is ensured by the signature scheme $\mathcal{S}$. In the following, we only concern ourselves with the verifiability of the User's decoded result.

For each $1 \le i \le m$, we have

$$\hat{e}(D_{i,1}, Q_{i,1}) = \hat{e}(h,g)^{(d_i+\lambda_{i,1}+\phi_1)(\tau_1 q_i+\tau_2)} \hat{e}(h,g)^{\mu_{i,1}(d_i+\lambda_{i,1}+\phi_1)}$$
$$\hat{e}(D_{i,2}, Q_{i,2}) = \hat{e}(h,g)^{\mu_{i,1}(d_i+\lambda_{i,1}+\phi_2)}$$
$$\hat{e}(D_{i,3}, Q_{i,3}) = \hat{e}(h,g)^{\lambda_{i,1}(\tau_1 q_i+\tau_2+\mu_{i,2})}$$
$$\hat{e}(D_{i,4}, Q_{i,4}) = \hat{e}(h,g)^{\lambda_{i,1}\mu_{i,2}}$$
$$\hat{e}(D_{i,5}, Q_{i,5}) = \hat{e}(h,g)^{(d_i+\lambda_{i,2}+\phi_3)(\tau_3 q_i+\tau_4)} \hat{e}(h,g)^{\mu_{i,3}(d_i+\lambda_{i,2}+\phi_3)}$$
$$\hat{e}(D_{i,6}, Q_{i,6}) = \hat{e}(h,g)^{\mu_{i,3}(d_i+\lambda_{i,2}+\phi_4)}$$
$$\hat{e}(D_{i,7}, Q_{i,7}) = \hat{e}(h,g)^{\lambda_{i,2}(\tau_3 q_i+\tau_4+\mu_{i,4})}$$
$$\hat{e}(D_{i,8}, Q_{i,8}) = \hat{e}(h,g)^{\lambda_{i,2}\mu_{i,4}}$$
$$\hat{e}(D_9, Q_9) = \hat{e}(h,g)^{\tau_2\beta_1+\tau_2\sum_{i=1}^{m} d_i}$$
$$\hat{e}(D_{10}, Q_{10}) = \hat{e}(h,g)^{\tau_4\beta_2+\tau_4\sum_{i=4}^{m} d_i}$$

Thus, according to the equalities (1) and (2), we have

$$W_1 = \prod_{i=1}^{m} \hat{e}(h,g)^{(d_i+\phi_1)(\tau_1 q_i+\tau_2)+(\phi_1-\phi_2)\mu_{i,1}} \Big/ \hat{e}(h,g)^{\tau_2\beta_1+\tau_2\sum_{i=1}^{m} d_i}$$
$$= \hat{e}(h,g)^{\tau_1\sum_{i=1}^{m} d_i q_i} \hat{e}(h,g)^{\phi_1\tau_1\sum_{i=1}^{m} q_i + m\phi_1\tau_2 + (\phi_1-\phi_2)\sum_{i=1}^{m}\mu_{i,1}} \Big/ \hat{e}(h,g)^{\tau_2\beta_1}$$

and

$$W_2 = \prod_{i=1}^{m} \hat{e}(h,g)^{(d_i+\phi_3)(\tau_3 q_i+\tau_4)+(\phi_3-\phi_4)\mu_{i,3}} \Big/ \hat{e}(h,g)^{\tau_4\beta_2+\tau_4\sum_{i=4}^{m} d_i}$$
$$= \hat{e}(h,g)^{\tau_3\sum_{i=1}^{m} d_i q_i} \hat{e}(h,g)^{\phi_3\tau_3\sum_{i=1}^{m} q_i + m\phi_3\tau_4 + (\phi_3-\phi_4)\sum_{i=1}^{m}\mu_{i,3}} \Big/ \hat{e}(h,g)^{\tau_4\beta_2}$$

Defining

$$R_1 = \phi_1\tau_1\sum_{i=1}^{m} q_i + m\phi_1\tau_2 + (\phi_1-\phi_2)\sum_{i=1}^{m}\mu_{i,1} \mod p,$$

we have

$$W = W_1 \times E_2^{\tau_2} / E_1^{R_1}$$
$$= \frac{\hat{e}(h,g)^{\tau_1\sum_{i=1}^{m} d_i q_i} \times \hat{e}(h,g)^{R_1} \times \hat{e}(h,g)^{\beta_1\tau_2}}{\hat{e}(h,g)^{\tau_2\beta_1} \times \hat{e}(h,g)^{R_1}}$$
$$= \hat{e}(h,g)^{\tau_1\sum_{i=1}^{m} d_i q_i}$$
$$= E_1^{\tau_1\sum_{i=1}^{m} d_i q_i}$$

There must exist a unique $v \in [0, 2^{k_d+k_q} \cdot m_q)$ such that $(E_1^{\tau_1})^v = W$, which means $v = \sum_{i=1}^{m} d_i q_i \mod p$. Furthermore, for a valid v and letting

$$R_2 = \phi_3\tau_3\sum_{i=1}^{m} q_i + m\phi_3\tau_4 + (\phi_3-\phi_4)\sum_{i=1}^{m}\mu_{i,3} \mod p,$$

we must have

$$E_1^{\tau_3 v+R_2}/E_3^{\tau_4} = \hat{e}(h,g)^{\tau_3 v+R_2}/\hat{e}(h,g)^{\beta_2\tau_4}$$
$$= \hat{e}(h,g)^{\tau_3\sum_{i=1}^{m} d_i q_i}\hat{e}(h,g)^{\phi_3\tau_3\sum_{i=1}^{m} q_i + m\phi_3\tau_4 + (\phi_3-\phi_4)\sum_{i=1}^{m}\mu_{i,3}} \Big/ \hat{e}(h,g)^{\tau_4\beta_2}$$
$$= W_2$$

The correctness of the DSM scheme follows. □

4.2. Security analysis

We analyze the security of our construction in Section 4.1, showing in turn how it satisfies the various security requirements defined in Section 3.2.

Theorem 4.2. *The proposed DSM scheme guarantees the integrity of user queries against the Server and colluding users; that is, even when the Server colludes with a set* U *of users, it cannot modify the standing queries of honest users and still generate valid encoded results for documents.*

Proof. We show that if signature scheme $\mathcal{S}$ is existentially unforgeable and the discrete logarithm problem is hard, then there exists no PPT adversary controlling the Server and user set U, who can break the integrity of user queries in the DSM scheme with non-negligible probability.

Setup: Following the proposed scheme, challenger $\mathcal{C}$ runs $(pk, sk) \leftarrow \mathtt{Setup}(1^\ell, m)$, and $(uk_i, ssk_i) \leftarrow \mathtt{UserReg}(pk, sk)$ for each user in U. $\mathcal{C}$ then gives pk and $\{ssk_i\}_{\mathtt{U}}$ to adversary $\mathcal{A}$.

Queries: As defined in Definition 3.2, adversary $\mathcal{A}$ adaptively issues the following three types of queries. Challenger $\mathcal{C}$ maintains the corresponding query lists that record all the queries and responses.

- *Query generation queries*: For each query $\mathbf{q_{l,i}} = \{\mathbf{q_{l,i,j}}\}_{\mathbf{j=1}}^{\mathbf{m}}$ of user $U_l \notin \mathtt{U}$ where $q_{l,i,j} \in [0, 2^{k_q}) \subset \mathbb{Z}_p$, the challenger follows algorithm $\mathtt{QueryGen}(pk, uk_l, \mathbf{q_{l,i}})$ to produce $\mathbf{Q}_{l,i}$ and returns $\mathbf{Q}_{l,i}$ to $\mathcal{A}$. Here, if U_l has not posed this type of queries before, the challenger runs $\mathtt{UserReg}(pk, sk)$ to generate a user key uk_l and a server-side key ssk_l, then gives ssk_l to the adversary along with $\mathbf{Q}_{l,i}$.
- *Document generation queries*: For each query $\mathbf{d_j} = \{\mathbf{d_{j,t}}\}_{\mathbf{t=1}}^{\mathbf{m}}$ where $d_{j,t} \in [0, 2^{k_d}) \subset \mathbb{Z}_p$, the challenger follows algorithm $\mathtt{DocGen}(pk, sk, \mathbf{d_j})$ to produce $\tilde{\mathbf{D}}_j$ and returns $\tilde{\mathbf{D}}_j$ to $\mathcal{A}$.
- *Decoding queries*: For each encoded query result $\mathbf{R}_{l,i,j}$ for a queried standing query $\mathbf{q}_{l,i}$ of user $U_l \notin \mathtt{U}$ and a queried document $\mathbf{d}_j$, the challenger runs $\mathtt{UserDec}(pk, uk_l, \mathbf{R}_{l,i,j}, T_{\mathbf{q_{l,i}}})$ and sends back the corresponding result v or $\perp$.

Output: Eventually, the adversary outputs an index pair $(\hat{\imath}, \hat{l})$ that corresponds to some $\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}$ of user $U_{\hat{l}} \notin \mathtt{U}$ in the query list. $\mathcal{C}$ then challenges the adversary with $\tilde{\mathbf{D}} = (id_{\mathbf{d}}, \chi_{\mathbf{d}}, \mathbf{D}, \mathbf{C}, \mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}) \leftarrow \mathtt{DocGen}(\mathbf{pk}, \mathbf{sk}, \mathbf{d})$ where $\mathbf{d}$ is a randomly chosen fresh document. Suppose the adversary responds with $\mathbf{R}' = (id_{\mathbf{d}}, \chi_{\mathbf{d}}, W_1', W_2', C, C_1', E_1, E_2, E_3)$ and wins the game. There are two cases to consider:

Case 1: $C_1' \neq C_1$ where C_1 is generated with information that $\mathcal{C}$ maintains locally.

In this case, the signature $\chi_{\mathbf{d}_{\hat{\mathbf{i}}}}$ must be valid for a string of parameters $id_{\mathbf{d}} \| C \| \phi_1' \| \cdots \| \phi_4' \| E_1 \| E_2 \| E_3$, where $\phi_j' = H(id_{\mathbf{d}} \| j \| C_1' \times \hat{e}(C, \Omega_u))$ for $1 \le j \le 4$. It implies the adversary has generated a forged signature for a new message, and in doing so breaks the signature scheme $\mathcal{S}$.

Case 2: $C_1' = C_1$.

In this case, algorithm $\mathtt{UserDec}(pk, uk_{\hat{l}}, \mathbf{R}', T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}})$ outputs v' such that the following equalities hold:

$$E_1^{\tau_1 v'+R_1} = W_1' \times E_2^{\tau_2}, \qquad E_1^{\tau_3 v'+R_2} = W_2' \times E_3^{\tau_4}$$

Furthermore, according to the information maintained by the challenger, the following equalities hold for $v \leftarrow \mathtt{UserDec}(pk, uk_{\hat{l}}, \mathbf{R}, T_{\mathbf{q}_{\hat{\mathbf{l}},\hat{\mathbf{i}}}})$:

$$E_1^{\tau_1 v+R_1} = W_1 \times E_2^{\tau_2}, \qquad E_1^{\tau_3 v+R_2} = W_2 \times E_3^{\tau_4}$$

Clearly, $v' \neq v$ otherwise both $W_1' = W_1$ and $W_2' = W_2$ would hold, which in turn implies $\mathbf{R}' = \mathbf{R}$. Denoting $\Delta v = v - v' \bmod p$,

$$E_1^{\tau_1 \Delta v} = W_1/W_1', \qquad E_1^{\tau_3 \Delta v} = W_2/W_2'$$

which yields

$$\left(\frac{W_1}{W_1'}\right)^{\tau_3} = \left(\frac{W_2}{W_2'}\right)^{\tau_1} \tag{3}$$

Note that W_1 and W_2 are independent, since the underlying $D_{i,j}$ and $Q_{i,j}$ rely on uniformly distributed elements $\lambda_{i,j}$, $\mu_{i,j}$ and τ_j, etc. Thus, the adversary needs to know the value of τ_1 and τ_3 in order to output a "valid" pair of W_1' and W_2'

with non-negligible probability that satisfy Equality (3). However, this contradicts the assumption that discrete logarithm is hard. □

Theorem 4.3. *The proposed DSM scheme guarantees the privacy of user queries against the Server and a set* U *of colluding users; that is, they can neither distinguish between the standing queries outsourced by a honest User, nor obtain the content of the queries.*

Proof. In the composition of $\mathbf{Q}_u$, $(\{Q_{i,1}, D_{i,2}, D_{i,3}, Q_{i,4}\}_{i=1}^{m}, Q_9)$ and $(\{Q_{i,5}, D_{i,6}, D_{i,7}, Q_{i,8}\}_{i=1}^{m}, Q_{10})$ are independent as they involve different sets of τ and μ values. In fact, the elements in $\mathbf{Q}_u$ look random to the Server if the elements in both $\mathbb{Z}_p^*$ and $\mathbb{G}$ are uniformly distributed. Specifically, in encoding $q_i \in \mathbf{q}$, $Q_{i,2}$ and $Q_{i,4}$ are determined by random elements $\mu_{i,1}$ and $\mu_{i,2}$ respectively, while $Q_{i,1}$ and $Q_{i,3}$ depend on τ_1, τ_2, $\mu_{i,1}$, $\mu_{i,2}$. Similarly, $Q_{i,6}$ and $Q_{i,8}$ are determined by random elements $\mu_{i,3}$ and $\mu_{i,4}$ respectively, while $Q_{i,5}$ and $Q_{i,7}$ depend on τ_3, τ_4, $\mu_{i,3}$, $\mu_{i,4}$. In addition, Q_9 and Q_{10} rely on τ_2 and τ_4 respectively. The bases of $Q_{i,j}$, Q_9 and Q_{10} are in the User's secret key, i.e., $g^{1/\sigma_{i,j}}$ and g^{1/α_j}, which are shared by all users who have registered with the Owner. In this collusion scenario, these bases are thus known to the adversary. That, however, does not compromise the privacy of the encoded queries since all the randomness are realized in the exponents. Since all the values of τ_j and $\mu_{i,j}$ are randomly chosen from $\mathbb{Z}_p^*$, the elements in $\mathbf{Q}_u$ are random elements of group $\mathbb{G}$ from the Server's perspective.

Furthermore, suppose two queries $\mathbf{q}_1$ and $\mathbf{q}_2$ contain the same value at some coordinate, for example, $q_{1,j} = q_{2,j} = \mathtt{q}$. In our construction, the encoding of $\mathtt{q}$ in different queries uses different sets of random values τ and μ. Thus, the colluding Server and user set U cannot distinguish between encoded queries that contain the same coordinate values. Since all elements q_i in a query $\mathbf{q}$ are raised to elements of $\mathbb{G}$, the colluding Server and user set U cannot recover q_i as long as the discrete logarithm assumption holds. □

Theorem 4.4. *The proposed DSM scheme guarantees the integrity of documents against the Server and colluding users; that is, even when the Server colludes with a set* U *of users, it cannot modify the processed documents and still generate valid encoded results for standing queries.*

Proof. We show that if the signature scheme $\mathcal{S}$ is existentially unforgeable and the discrete logarithm problem is hard, then there exists no PPT adversary controlling the Server and user set U, who can break the integrity of processed documents in the DSM scheme with non-negligible probability.

Setup: The same as in the proof of Theorem 4.2.

Queries: The same as in the proof of Theorem 4.2.

Output: Eventually, the adversary outputs a tuple $(\mathbf{R}', id_{\mathbf{d}_{\hat{\mathbf{i}}}}, \hat{f}, \hat{l})$ and wins the game, where $\mathbf{R}' = (id_{\mathbf{d}_{\hat{\mathbf{i}}}}, \chi'_{\mathbf{d}_{\hat{\mathbf{i}}}}, W'_1, W'_2, C', C'_1, E'_1, E'_2, E'_3)$. With the locally maintained information, the challenger gets

$$\tilde{\mathbf{D}}_{\hat{l}} = (id_{\mathbf{d}_{\hat{\mathbf{i}}}}, \chi_{\mathbf{d}_{\hat{\mathbf{i}}}}, \mathbf{D}_{\hat{\mathbf{i}}}, \mathbf{C}, \mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}) \leftarrow \mathtt{DocGen}(\mathbf{pk}, \mathbf{sk}, \mathbf{d}_{\hat{\mathbf{i}}}),$$

and

$$\mathbf{R} = (id_{\mathbf{d}_{\hat{\mathbf{i}}}}, \chi_{\mathbf{d}_{\hat{\mathbf{i}}}}, W_1, W_2, C, C_1, E_1, E_2, E_3) \leftarrow \mathtt{ServPro}(\tilde{\mathbf{D}}_{\hat{l}}, ssk_{\hat{l}}, \mathbf{Q}_{\hat{\mathbf{i}}\hat{\mathbf{j}}}).$$

There are two cases to consider:

Case 1: $(\chi'_{\mathbf{d}_{\hat{\mathbf{i}}}}, C', E'_1, E'_2, E'_3) \neq (\chi_{\mathbf{d}_{\hat{\mathbf{i}}}}, C, E_1, E_2, E_3)$.

In this case, the signature $\chi'_{\mathbf{d}_{\hat{\mathbf{i}}}}$ must be valid for a string of parameters $id'_{\mathbf{d}_{\hat{\mathbf{i}}}} \| C' \| \phi'_1 \| \cdots \| \phi'_4 \| E'_1 \| E'_2 \| E'_3$, where $\phi'_j = H(id'_{\mathbf{d}_{\hat{\mathbf{i}}}} \| j \| C'_1 \times \hat{e}(C', \Omega_u))$ for $1 \leq j \leq 4$. It implies the adversary has generated a forged signature for a new message and, in doing so, breaks the signature scheme $\mathcal{S}$.

Case 2: $(\chi'_{\mathbf{d}_{\hat{\mathbf{i}}}}, C', E'_1, E'_2, E'_3) = (\chi_{\mathbf{d}_{\hat{\mathbf{i}}}}, C, E_1, E_2, E_3)$.

Here, the discussion is similar to Case 2 in the proof of Theorem 4.2. □

Theorem 4.5. *The proposed DSM scheme guarantees the privacy of documents against a malicious Server; that is, the Server can neither distinguish between the processed documents, nor obtain their contents.*

Proof. In the composition of $\tilde{\mathbf{D}}$, $(\{D_{i,1}, D_{i,2}, D_{i,3}, D_{i,4}\}_{i=1}^{m}, D_9)$ and $(\{D_{i,5}, D_{i,6}, D_{i,7}, D_{i,8}\}_{i=1}^{m}, D_{10})$ are independent as they involve different λ and β values. In fact, the elements in $\tilde{\mathbf{D}}$ look random to the Server if the elements in both $\mathbb{Z}_p^*$, $\mathbb{G}$ and $\mathbb{G}_T$ are uniformly distributed. Specifically, in processing $d_i \in \mathbf{d}$, the elements C, E_1, E_2, E_3 in $\tilde{\mathbf{D}}$ essentially rely on the random values r, β_1, β_2, h. As H is a collision-resistant hash function, ϕ_i are pseudo-random values over $\mathbb{Z}_p^*$. Hence, $D_{i,1}, \cdots, D_{i,4}$ are determined by the random elements h, $\lambda_{i,1}$, ϕ_1, ϕ_2. Similarly, $D_{i,5}, \cdots, D_{i,8}$ are determined by the random elements h, $\lambda_{i,2}$, ϕ_3, ϕ_4. In addition, D_9 and D_{10} rely on β_1 and β_2 respectively. Notice that the exponents of $D_{i,j}$, D_9 and D_{10} are further randomized by the Owner's secret key, i.e., $\sigma_{i,j}$ and α_j. Since all the β_j and $\lambda_{i,j}$ values are randomly chosen from $\mathbb{Z}_p^*$, the elements in $\mathbf{Q}_u$ are random elements of group $\mathbb{G}$ and group $\mathbb{G}_T$ from the Server's view.

Furthermore, suppose two documents $\mathbf{d}_1$ and $\mathbf{d}_2$ contain the same value at some coordinate, for example, $d_{1,j} = d_{2,j} = \mathtt{d}$. In our construction, the encoding of $\mathtt{d}$ in different documents uses different sets of random values λ, ϕ, β and h. Thus, the Server cannot distinguish between processed documents that contain the same coordinate values. Since all elements

d_i in a document $\mathbf{d}$ are raised to elements of $\mathbb{G}$, the Server cannot recover their values as long as the discrete logarithm assumption holds. □

Remark 5. Our DSM scheme is proposed under the assumption that the Server would always respond to all qualified Users when a fresh document is published (see Section 2.1). In some situation, the Server may refuse to provide service, that is, the Server will not match some queries with some documents. This issue can be easily addressed by adding an *auditing* phase to the DSM system. That is, at run time, the Owner will record all the identifiers of the published documents and each User will also record all the document identifiers included in the received monitored results $\mathbf{R}$. Then, the User can periodically report to the Owner a list of document identifiers that were matched with her queries. This way, the Owner can compare the received identifiers with the locally maintained ones for auditing.

5. Query privacy against the owner

Our scheme in Section 4 achieves query privacy against the Server and colluding users, but not the Owner. This level of protection is usually adequate, as a User has to trust the Owner implicitly to utilize the documents served out by the latter. Nevertheless, there may still be situations where users desire query privacy against even the Owner. We show how that can be achieved in this section.

We begin by explaining why our original scheme does not protect query privacy against the Owner. With secret key sk, the Owner could compute $Q_{i,1}^{\sigma_{i,1}}/(Q_{i,2}^{\sigma_{i,2}} \times Q_9^{\alpha_3}) = g^{\tau_1 q_i}$, for example. This allows the Owner to deduce whether $q_i = 0$, and whether two coordinates q_i and q_j in a query vector have the same value.

To achieve query privacy against the Owner, the changes that we need to make to the scheme in Section 4 are summarized below.

$\texttt{Setup}(1^\ell, m)$: In addition to the existing steps, select $\alpha_5, \alpha_6 \in_R \mathbb{Z}_p$ and include them in secret key sk.

$\texttt{UserReg}(pk, sk)$: In the output to the User, change $\{g^{1/\alpha_j}\}_{j=1}^4$ to $\{g^{1/\alpha_j}\}_{j=1}^6$.

$\texttt{QueryGen}(pk, uk, \mathbf{q})$: Changes to Algorithm QueryGen are:

(1) Select $\tau_1, \tau_{i,2}, \tau_3, \tau_{i,4}, \tau_5, \tau_6 \in_R \mathbb{Z}_p$ for $1 \le i \le m$.
(2) Compute $\mathbf{Q_u} = \{\mathbf{Q_{i,1}}, \mathbf{Q_{i,2}}, \cdots, \mathbf{Q_{i,10}}\}_{\mathbf{i=1}}^{\mathbf{m}}$ where:

$$\begin{aligned} Q_{i,1} &= g^{(\tau_1 q_i + \tau_{i,2} + \mu_{i,1})/\sigma_{i,1}}, & Q_{i,2} &= g^{\mu_{i,1}/\sigma_{i,2}} \\ Q_{i,3} &= g^{(\tau_1 q_i + \tau_{i,2} + \mu_{i,2})/\sigma_{i,3}}, & Q_{i,4} &= g^{\mu_{i,2}/\alpha_1} \\ Q_{i,5} &= g^{(\tau_3 q_i + \tau_{i,4} + \mu_{i,3})/\sigma_{i,4}}, & Q_{i,6} &= g^{\mu_{i,3}/\sigma_{i,5}} \\ Q_{i,7} &= g^{(\tau_3 q_i + \tau_{i,4} + \mu_{i,4})/\sigma_{i,6}}, & Q_{i,8} &= g^{\mu_{i,4}/\alpha_2} \\ Q_{i,9} &= g^{\tau_{i,2}\tau_5/\alpha_3}, & Q_{i,10} &= g^{\tau_{i,4}\tau_6/\alpha_4} \end{aligned}$$

(3) Send $\mathbf{Q}_u$ to the Server, retain $T_\mathbf{q}$ which includes τ_1, τ_3, $\{\tau_{i,2}, \tau_{i,4}\}_{i=1}^m$, τ_5, τ_6, $\sum_{i=1}^m q_i$, $\sum_{i=1}^m \mu_{i,1}$ and $\sum_{i=1}^m \mu_{i,3}$.

$\texttt{DocGen}(pk, sk, \mathbf{d})$: Changes to Algorithm DocGen are:

(1) Compute $E_1 = \hat{e}(h, g)$.
(2) Compute $\chi_\mathbf{d} = \mathcal{S}.\mathrm{Sign}_{\mathrm{tsk}}(id_\mathbf{d} \| C \| \phi_1 \| \cdots \| \phi_4 \| E_1)$.
(3) Select $\lambda_{i,j} \in_R \mathbb{Z}_p$ for $1 \le i \le m$ and $1 \le j \le 4$.
(4) Compute $\mathbf{D} = \{\mathbf{D_{i,1}}, \mathbf{D_{i,2}}, \cdots, \mathbf{D_{i,12}}\}_{\mathbf{i=1}}^{\mathbf{m}}$ where:

$$\begin{aligned} D_{i,9} &= h^{\alpha_3(d_i + \lambda_{i,3})}, & D_{i,10} &= h^{\alpha_4(d_i + \lambda_{i,4})} \\ D_{i,11} &= h^{\alpha_5 \lambda_{i,3}}, & D_{i,12} &= h^{\alpha_6 \lambda_{i,4}} \end{aligned}$$

(5) Send $\tilde{\mathbf{D}} = (id_\mathbf{d}, \chi_\mathbf{d}, \mathbf{D}, \mathbf{C}, \mathbf{E_1})$ to the Server.

$\texttt{ServPro}(\tilde{\mathbf{D}}, ssk_u, \mathbf{Q_u})$: Replace the steps in Algorithm ServPro with the following:

(1) Compute $w_{i,1}$ $\forall 1 \le i \le m$:

$$\begin{aligned} \hat{e}(D_{i,1}, Q_{i,1}) &= \hat{e}(h, g)^{(d_i + \lambda_{i,1} + \phi_1)(\tau_1 q_i + \tau_{i,2})} \hat{e}(h, g)^{\mu_{i,1}(d_i + \lambda_{i,1} + \phi_1)} \\ \hat{e}(D_{i,2}, Q_{i,2}) &= \hat{e}(h, g)^{\mu_{i,1}(d_i + \lambda_{i,1} + \phi_2)} \\ \hat{e}(D_{i,3}, Q_{i,3}) &= \hat{e}(h, g)^{\lambda_{i,1}(\tau_1 q_i + \tau_{i,2} + \mu_{i,2})} \\ \hat{e}(D_{i,4}, Q_{i,4}) &= \hat{e}(h, g)^{\lambda_{i,1}\mu_{i,2}} \\ \hat{e}(D_{i,9}, Q_{i,9}) &= \hat{e}(h, g)^{\tau_{i,2}\tau_5(d_i + \lambda_{i,3})} \end{aligned}$$

$$w_{i,1} = \frac{\hat{e}(D_{i,1}, Q_{i,1})\hat{e}(D_{i,4}, Q_{i,4})}{\hat{e}(D_{i,2}, Q_{i,2})\hat{e}(D_{i,3}, Q_{i,3})} = \hat{e}(h, g)^{(d_i + \phi_1)(\tau_1 q_i + \tau_{i,2}) + (\phi_1 - \phi_2)\mu_{i,1}}$$

(2) Compute

$$W_1 = \prod_{i=1}^{m} w_{i,1} = \hat{e}(h,g)^{\sum_{i=1}^{m}(d_i+\phi_1)(\tau_1 q_i+\tau_{i,2})+(\phi_1-\phi_2)\sum_{i=1}^{m}\mu_{i,1}}$$

$$\Psi_1 = \prod_{i=1}^{m} \hat{e}(D_{i,9}, Q_{i,9}) = \hat{e}(h,g)^{\tau_5 \sum_{i=1}^{m} \tau_{i,2}(d_i+\lambda_{i,3})}$$

(3) Compute $w_{i,2}$ $\forall 1 \le i \le m$:

$$\hat{e}(D_{i,5}, Q_{i,5}) = \hat{e}(h,g)^{(d_i+\lambda_{i,2}+\phi_3)(\tau_3 q_i+\tau_{i,4})}\hat{e}(h,g)^{\mu_{i,3}(d_i+\lambda_{i,2}+\phi_3)}$$

$$\hat{e}(D_{i,6}, Q_{i,6}) = \hat{e}(h,g)^{\mu_{i,3}(d_i+\lambda_{i,2}+\phi_4)}$$

$$\hat{e}(D_{i,7}, Q_{i,7}) = \hat{e}(h,g)^{\lambda_{i,2}(\tau_3 q_i+\tau_{i,4}+\mu_{i,4})}$$

$$\hat{e}(D_{i,8}, Q_{i,8}) = \hat{e}(h,g)^{\lambda_{i,2}\mu_{i,4}}$$

$$\hat{e}(D_{i,10}, Q_{i,10}) = \hat{e}(h,g)^{\tau_{i,4}\tau_6(d_i+\lambda_{i,4})}$$

$$w_{i,2} = \frac{\hat{e}(D_{i,5}, Q_{i,5})\hat{e}(D_{i,8}, Q_{i,8})}{\hat{e}(D_{i,6}, Q_{i,6})\hat{e}(D_{i,7}, Q_{i,7})} = \hat{e}(h,g)^{(d_i+\phi_3)(\tau_3 q_i+\tau_{i,4})+(\phi_3-\phi_4)\mu_{i,3}}$$

(4) Compute

$$W_2 = \prod_{i=1}^{m} w_{i,2} = \hat{e}(h,g)^{\sum_{i=1}^{m}(d_i+\phi_3)(\tau_3 q_i+\tau_{i,4})+(\phi_3-\phi_4)\sum_{i=1}^{m}\mu_{i,3}}$$

$$\Psi_2 = \prod_{i=1}^{m} \hat{e}(D_{i,10}, Q_{i,10}) = \hat{e}(h,g)^{\tau_6 \sum_{i=1}^{m} \tau_{i,4}(d_i+\lambda_{i,4})}$$

(5) Compute $C_1 = \hat{e}(C, \Psi_u)$.
(6) Send $\mathbf{R} = (id_{\mathbf{d}}, \chi_{\mathbf{d}}, W_1, W_2, C, C_1, E_1, \Psi_1, \Psi_2, \{D_{i,11}, D_{i,12}\}_{i=1}^{m})$ to the User.

`UserDec`$(pk, uk, \mathbf{R}, T_{\mathbf{q}})$: Replace the steps in Algorithm `UserDec` with the following:

(1) Validate $\chi_{\mathbf{d}}$ on $id_{\mathbf{d}}\|C\|\phi_1\|\cdots\|\phi_4\|E_1$ using tpk. If it is invalid, output $\perp$ and abort.
(2) Compute $R_1 = \phi_1\tau_1\sum_{i=1}^{m} q_i + \phi_1\sum_{i=1}^{m}\tau_{i,2} + (\phi_1-\phi_2)\sum_{i=1}^{m}\mu_{i,1}$.
(3) Compute $R_2 = \Psi_1^{1/\tau_5} / \prod_{i=1}^{m} \hat{e}(D_{i,11}, g^{\tau_{i,2}/\alpha_5}) = E_1^{\sum_{i=1}^{m} d_i\tau_{i,2}}$.
(4) Compute $W = W_1/(E_1^{R_1} \times R_2) = E_1^{\tau_1\sum_{i=1}^{m} d_i q_i}$.
(5) Find $v \in [0, 2^{k_d+k_q} \cdot m_q)$ such that $\left[E_1^{\tau_1}\right]^v = W$ and m_q is the number of coordinates used in $\mathbf{q}$.
(6) Compute $R_3 = \phi_3\tau_3\sum_{i=1}^{m} q_i + \phi_3\sum_{i=1}^{m}\tau_{i,4} + (\phi_3-\phi_4)\sum_{i=1}^{m}\mu_{i,3}$.
(7) Compute $R_4 = \Psi_2^{1/\tau_6} / \prod_{i=1}^{m} \hat{e}(D_{i,12}, g^{\tau_{i,4}/\alpha_6}) = E_1^{\sum_{i=1}^{m} d_i\tau_{i,4}}$.
(8) Verify that $E_1^{\tau_3 v+R_3} \times R_4 \stackrel{?}{=} W_2$. If the condition holds, then output v, otherwise output $\perp$.

With the modified scheme in this section, the communication cost between the Server and users, as well as the user decoding cost, become linear in $|\mathbf{d}|$ (the length of a document vector). These are on the same order as the corresponding costs incurred by the näive method of having the Owner distribute his document vectors directly to the users, which also affords query privacy, result privacy and verifiability. The advantage of our modified scheme is it adds document privacy to query privacy, result privacy and verifiability. With the scheme, a User can only defeat document privacy with the help of a colluding Server; for example, a Server that allows $|\mathbf{d}|$ probing queries from the User to discover all the coordinates in a document vector.

6. Optimizations for query result decoding

In Algorithm `UserDec` in Section 4.1, the User needs to extract the inner product $v = \mathbf{q}\cdot\mathbf{d} \in [\mathbf{0}, \mathbf{2^{k_d+k_q}\cdot m_q})$ such that $(E_1^{\tau_1})^v = W$. We now discuss techniques to optimize the extraction of v. We will validate the practicality of the optimized algorithm through experiments in Section 7.2.

6.1. Constraining the range of the query result

To constrain the search for v, the User may predefine a threshold t to filter out documents with low scores, signifying that they are too dissimilar to the User's interest and may be discarded without examination. This limits the search for v to a narrower range $[t, 2^{k_d+k_q}\cdot m_q)$. Here t is chosen by and known only to the User.

Moreover, when the document streaming rate is high, the User may wish to monitor only the top-k documents [8] within a sliding window (of some number of recent documents, or of documents that arrived within the last several

Table 3
Theoretical performance comparison.

		Scheme in [5]	Section 4.1	Section 5
Element size	Encoded query	$m\xi_{G'}$	$(8m+2)\xi_G$	$10m\xi_G$
	Processed document	$(m+1)\xi_{G'}$	$\xi_s+\xi_Z+(8m+3)\xi_G+3\xi_T$	$\xi_s+\xi_Z+(12m+1)\xi_G+1\xi_T$
Computation cost	`UserReg`	$2\delta_{G'}+\delta_{T'}$	$(6m+6)\delta_G$	$(6m+8)\delta_G$
	`QueryGen`	$2m\delta_{G'}$	$(8m+2)\delta_G$	$10m\delta_G$
	`DocGen`	$2m\delta_{G'}+\delta_{\hat{e}'}$	$(8m+4)\delta_G+2\delta_T+2\delta_{\hat{e}}+\delta_S$	$12m\delta_G+\delta_{\hat{e}}+\delta_S$
	`ServPro`	$(m+1)\delta_{\hat{e}'}$	$(8m+3)\delta_{\hat{e}}$	$(10m+1)\delta_{\hat{e}}$
	`UserDec`	$\delta_{\hat{e}'}+\delta_{T'}+\delta_\ell$	$\delta_{\hat{e}}+4\delta_T+\delta_V+\delta_\ell$	$(2m+2)\delta_G+2\delta_T+2m\delta_{\hat{e}}+\delta_V+\delta_\ell$

seconds). The score v_k of the kth ranked document provides another lower bound to limit the search for v. Therefore, the User only needs to search for the score v of the new document within a bounded range:

$$v \in [\max\{t, v_k\}, 2^{k_d+k_q} \cdot m_q) \tag{4}$$

The constraints in the formula enable the User to avoid a full discrete logarithm computation in $\mathbb{Z}_p$ for v, by exploiting secret application-specific values t, v_k and m_q which only the User knows.

In our target applications, typically the number of features that Users specify in queries satisfies $m_q \ll m$ and is within a hundred or two. Moreover, k_d and k_q are also small. Thus, if the constrained range for v is narrow, the User can pre-generate a look-up table for v. If not, the User has to perform a "bounded" discrete logarithm computation for v as explained next.

6.2. Bounded baby-step giant-step method

There exist several algorithms for solving the discrete logarithm problem, such as the baby-step giant-step algorithm, the Pohlig–Hellman algorithm and the index-calculus algorithm [12]. Although the latter two are superior in asymptotic time complexity, they are difficult to optimize to take advantage of the constraints expressed in Formula 4. Instead, we pick the baby-step giant-step algorithm for the User's discrete logarithm computation. The optimization works as follows to find $v \in [\max(t, v_k), 2^{k_d+k_q} \cdot m_q)$ satisfying $(E_1^{\tau_1})^v = W$ in the `UserDec` algorithm.

Let $b = E_1^{\tau_1}$, $\gamma = \left\lceil \sqrt{2^{k_d+k_q} \cdot m_q} \right\rceil$ and $v = c_1 \cdot \gamma + c_0$ for some $0 \le c_0,\ c_1 < \gamma$. Thus, $b^{-c_1\gamma} W = b^{c_0}$. The User creates beforehand a lookup table for $\langle b^{c_0}, c_0 \rangle$ with b^{c_0} as search key, for $0 \le c_0 < \gamma$. For each v, the User iteratively checks whether $b^{-c_1\gamma} W$ exists in the lookup table, decrementing c_1 from $\gamma - 1$ down towards $\lfloor \max(t, v_k)/\gamma \rfloor$. The procedure may be suspended after any iteration; as long as c_1 is kept, the procedure can be resumed subsequently.

Suppose that the User has a set of top-k documents in place when he obtains the answer W for a new document $\mathbf{d}$. He needs to evaluate immediately whether its score v falls within $[\max(t, v_k), 2^{k_d+k_q} \cdot m_q)$. If so, $\mathbf{d}$ replaces one of the earlier documents in the top-k result. If not, the User has established an upper bound $\overline{v} = v_k$ for v. The User also knows the earliest that $\mathbf{d}$ may be considered for the top-k result again is when one of the current top-k documents expires and v_k changes.

Now suppose that in between document arrivals, the User has several documents $\mathbf{d}_i$ with upper bound score $\overline{v}_i = v_k$. Instead of recovering the actual score v_i for each of those documents in turn, the User lowers their upper bounds $\overline{v}_i$'s uniformly (by decrementing c_1 in the computation of their document scores in unison). This allows the User to fully decode the higher result scores first, and discover the more relevant candidates that are likely to replace the next expiring document in the top-k result.

7. Performance analysis

7.1. Theoretical analysis

We analyze the efficiency of our DSM construction in Section 4.1 and its extension in Section 5, and compare them in Table 3 with the existing scheme in [5] in terms of the sizes of encoded queries and documents, and computation costs of each procedure. The analyses focus on resource-intensive computations including exponentiation and bilinear mapping, while lightweight computations such as addition, multiplication and hash evaluation are omitted. The costs of encoding one query $\mathbf{q}$ and one document $\mathbf{d}$ with dimensionality m, and of processing (at the server side) and decoding one score $\mathbf{q} \cdot \mathbf{d}$ are considered for the procedures `QueryGen`, `DocGen`, `ServPro` and `UserDec`, respectively.

In Table 3, for a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}$ and $\mathbb{G}_T$ have prime order p, we use ξ_G, ξ_T and ξ_Z to represent the element size in $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{Z}_p$, and δ_G and δ_T to denote the costs of evaluating an exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. We also use $\delta_{\hat{e}}$ to denote the evaluation cost of such a bilinear mapping $\hat{e}(\cdot,\cdot)$. Similarly, for a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}$ and $\mathbb{G}_T$ have composite order $n = p_1 p_2$ and p_1, p_2 are distinct prime numbers, we use $\xi_{G'}$ and $\xi_{T'}$ to represent the element size in $\mathbb{G}$ and $\mathbb{G}_T$, and $\delta_{G'}$ and $\delta_{T'}$ to denote the costs of evaluating an exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. We also use $\delta_{\hat{e}'}$ to denote the evaluation cost of such a bilinear mapping $\hat{e}(\cdot,\cdot)$ for composite order bilinear group. For the

Table 4
Experiment parameters.

Symbol	Meaning	Setting
λ	Document arrival rate per minute	1–**10**–30
w	Sliding window of documents for query results (minute)	**60**
k	Top-k documents for query results	**10**–60

signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, δ_S and δ_V respectively represent the computation costs of running $\mathcal{S}.\mathsf{Sign}$ and $\mathcal{S}.\mathsf{Verify}$, whereas ξ_s denotes its signature size. We let δ_ℓ denote the cost of finding v in the `UserDec` procedure.

From Table 3, we see that the scheme in [5] relies on bilinear groups with composite order which is less efficient than bilinear groups with prime order that our schemes are based on. In all the schemes, the sizes of encoded queries and processed documents are linear with the dimensionality m. The computation costs of the user registration algorithm `UserReg` and user decoding algorithm `UserDec` in [5] are independent of the dimensionality m, whereas only `UserDec` in our basic DSM construction in Section 4.1 enjoys this property. All the other algorithms need to perform computations that are linear with m. Although our extended scheme in Section 5 is not as efficient as the basic one in Section 4.1, it provides more stringent privacy guarantee for user queries.

7.2. Experiments

In this section, we evaluate the overall practicality of our DSM scheme, and the effectiveness of the optimization techniques introduced in Section 6 in particular. While we include the existing CVPM scheme from [5] as baseline in our evaluation, we emphasize that the DSM scheme in this paper provides strictly stronger security protection.

7.2.1. Experiment set-up

We begin by describing the set-up of the experiments. In Table 4 which summarizes the experiment parameters, the default setting of each parameter is highlighted in bold font.

Datasets: To ensure that our observations are generalizable, we have run DSM on several datasets. Here we report on two real datasets from the UCI KDD Archive.[2] These datasets are picked because they vary from each other in key properties that stress the various algorithms in DSM. The first, Corel Image, contains feature vectors extracted from 68040 photo images. Each feature vector is a point in 32-dimensional HSV color space, so $m = 32$. We discretize every dimension into 2^8 integer values. The feature vectors contain many zero coordinates, and the correlation between feature vectors are generally low. The dataset relates closely to the surveillance application in Section 1 that we used to motivate our problem formulation.

The second dataset, US Census, is a discretized version of part of the data collected in the 1990 U.S. census. The dataset includes 2,458,285 records (vectors), each with $m = 68$ attribute values. The attribute domains vary in size, with the widest being [0, 20]. The correlation between vectors are high, relative to that in the Corel Image data. The data in this collection are essentially user profiles, hence they simulate the user profile matching application described in the Introduction.

Methodology: For both datasets, we extract 100 vectors randomly to be user queries. The rest of the vectors are shuffled and fed into the document stream from the Owner. The arrival rate is λ documents/minute. The User maintains a sliding window of the documents that arrived in the last w minutes, and the k documents with highest correlation scores in this window constitute the result of a query. For each experiment setting, the performance measures are averaged over the 100 queries.

Competing schemes: We compare the DSM scheme proposed in this paper with the CVPM scheme from [5]. Both schemes are implemented in C language, on the PBC cryptography library from Stanford University.[3] For DSM, we set the group order p to a 160-bit prime number, which provides 80 bits of security and is equivalent in strength to 1024-bit RSA keys [3]. In the case of CVPM, the group order is set to the product of two 512-bit prime numbers to give the same level of security. The experiments are run on a MacBook Pro with 2.8 GHz Intel Core i7 CPU and 8GB of main memory.

Metrics: Our performance metrics include: (a) average time taken by the Owner to register a User with the `UserReg` algorithm; (b) average time taken by the User to generate a query with the `QueryGen` algorithm; (c) average time taken by the Owner in executing the `DocGen` algorithm on a new document; (d) average time taken by the Server to execute the `ServPro` algorithm for each document-query pair; and (e) average time taken by the User to run the `UserDec` algorithm and update her top-k result upon a document arrival, with and without the optimization described in Section 6. Among the metrics, the execution time of the `DocGen`, `ServPro` and `UserDec` algorithms affect runtime responsiveness and are our primary concerns.

[2] http://kdd.ics.uci.edu/databases/.
[3] http://crypto.stanford.edu/pbc/.

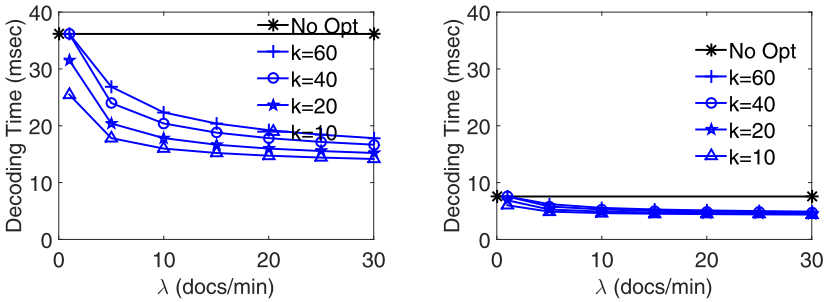

Fig. 3. User decoding time for Corel Image data.

Table 5
Processing times (ms) for Corel Image data.

	UserReg	QueryGen	DocGen	ServPro
CVPM	12.87	206.24	220.66	275.56
DSM	0.10	46.08	17.25	36.99

Table 6
Processing times (ms) for US Census data.

	UserReg	QueryGen	DocGen	ServPro
CVPM	12.87	438.26	452.28	576.19
DSM	0.10	97.93	36.32	78.28

7.2.2. Corel image experiment

For the Corel Image data, we set the sliding window w to 60 min and vary λ from 1 to 30 documents/minute. Referring to Formula 4, the threshold t is set to 0, so pruning of the search space for the correlation score v of a new document relies solely on v_k, the correlation score of the k-th result document.

Fig. 3 plots the average execution time of the UserDec algorithm, for $k = 10, 20, 40$ and 60. The figure also gives the execution time of 'No Opt', which disables the optimization techniques in Section 6. Here, the time taken by DSM is consistently just over 20% that of CVPM. The main cause of this performance difference is that DSM operates on a bilinear group with prime order, whereas CVPM requires a much larger bilinear group with composite order.

Another observation is that with a larger k, v_k becomes lower. If the correlation score v of an arriving document is below v_k, the User will have to execute more iterations in the baby-step giant-step algorithm before she can conclude that the new document scores below v_k and hence is not (yet) eligible for the top-k result. This explains why a larger k lengthens the execution time.

For a fixed k setting, the execution time drops as λ increases. The reason is that the sliding window accumulates more documents, in the process pushing up the kth highest correlation score and narrowing the search space for v.

In the worst case, the User executes the baby-step giant-step algorithm until v is discovered, so the execution time is bounded from above by the 'No Opt' method. The results demonstrate that the optimization techniques in Section 6 have the potential to cut user decoding time significantly.

Table 5 summarizes the other performance metrics. Again, we observe that DSM executes much faster than CVPM due to the difference between their group orders. For DocGen and ServPro, the two algorithms beside UserDec that affect runtime responsiveness, DSM's execution time is merely 7.8% and 13.4% of CVPM's, respectively.

7.2.3. US Census experiment

Turning to the US Census data, we again set the sliding window w at 60 min. The document arrival rate λ varies from 1 to 10 documents/minute.

The average user decoding time is summarized in Fig. 4. Here the performance effects of k and λ, and hence of the optimization techniques in Section 6, are not noticeable. This is because the narrower feature domains (i.e., smaller k_d and k_q values in Formula 4) constrain the search space for the correlation score v of arriving documents. Even so, DSM remains consistently three times faster than CVPM.

The execution time of the other algorithms are reported in Table 6. With the exception of set-up time, the execution times here are higher than in the previous experiment, on account of the larger number of features in this dataset – $m = 68$ – compared to $m = 32$ in the Corel Image data. Between the two schemes, DSM continues to achieve performance gains over CVPM that are similar to those seen in the previous experiment.

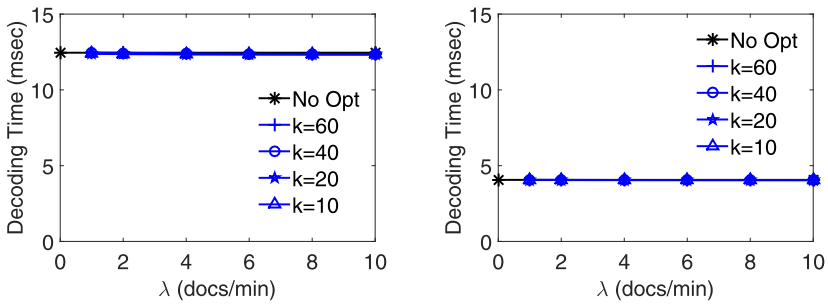

Fig. 4. User decoding time for US Census data.

7.2.4. Summary of experiment results

Besides Corel Image and US Census data, we experimented with several other datasets like the Insurance Company Benchmark, also available from the UCI KDD Archive. The common observations across experiments are as follows:

- The User decoding cost is sensitive to the number of features and their domain size. For applications that utilize many features with large domain sizes, it may be necessary to bucketize the feature values; effectively, this induces a coarser resolution on the features.
- The optimization techniques described in Section 6 are particularly effective with high document rate λ, large sliding window w or low k settings, as they tend to raise the correlation score v_k of the kth result document.
- Where the feature vectors have high dimensionality, it may be necessary to parallelize the `DocGen` and `ServPro` algorithms executed at the Owner and Server. This is straightforward to implement, as the computation on different coordinates in a document vector can be carried out independently.
- Our DSM scheme delivers acceptable performance for many practical application settings, with sub-second execution time for every party in the data streaming model.

8. Conclusion

In this paper, we formalize the framework of server-aided data stream monitoring (DSM) system and formulate the corresponding security requirements. In DSM, an untrusted server functions as intermediary for computing the correlation scores between documents streamed from the data owner, and standing queries issued by users. The correlation computation translates to an inner product of the document and query vectors concerned. We present a DSM construction that concurrently safeguards the integrity and privacy of the documents and queries, while enabling users to verify the correctness of the correlation scores received. We prove that the scheme offers strong security protection, and exceeds the previous method proposed in [5] in resisting any collusion between the server and other users. Through extensive experiments, we demonstrate that the proposed scheme also significantly outperforms the method of [5], achieving practical execution time for a wide spectrum of application settings.

Acknowledgement

This material is based on research work supported by the Singapore National Research Foundation under NCR Award Number NRF2014NCR-NCR001-012.

References

[1] C.C. Aggarwal, On randomization, public information and the curse of dimensionality, in: 2007 IEEE 23rd International Conference on Data Engineering, 2007, pp. 136–145, doi:10.1109/ICDE.2007.367859.
[2] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 439–450, doi:10.1145/342009.335438.
[3] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid, Recommendation for key management part 1: general (revision 3), NIST Special Publication 800-57, July 2012, National Institute of Standards and Technology, 2012. Available at http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57_Part1_rev3_general.pdf.
[4] J. Bethencourt, D.X. Song, B. Waters, New constructions and practical applications for private stream searching, in: 2006 IEEE Symposium on Security and Privacy (SP'06), 2006, pp. 132–139, doi:10.1109/SP.2006.27.
[5] X. Ding, H. Pang, J. Lai, Verifiable and private top-$k$monitoring, in: 8th ACM Symposium on Information, Computer and Communications Security (AsiaCCS), ACM, New York, NY, USA, 2013, pp. 553–558, doi:10.1145/2484313.2484388.
[6] B. Goethals, S. Laur, H. Lipmaa, T. Mielikäinen, On private scalar product computation for privacy-preserving data mining, in: Proceedings of the 7th International Conference on Information Security and Cryptology, in: ICISC'04, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 104–120, doi:10.1007/11496618_9.
[7] P. Golle, N. Modadugu, Authenticating streamed data in the presence of random packet loss (extended abstract), in: ISOC Network and Distributed System Security Symposium, 2001, pp. 13–22.

[8] I.F. Ilyas, G. Beskales, M.A. Soliman, A survey of top-K query processing techniques in relational database systems, ACM Comput. Surv. 40 (4) (2008) 11:1–11:58, doi:10.1145/1391729.1391730.
[9] J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, J. Cryptology 26 (2) (2013) 191–224, doi:10.1007/s00145-012-9119-4.
[10] F. Li, K. Yi, M. Hadjieleftheriou, G. Kollios, Proof-infused streams: enabling authentication of sliding window queries on streams, in: Proceedings of the 33rd International Conference on Very Large Data Bases, in: VLDB '07, VLDB Endowment, 2007, pp. 147–158. http://dl.acm.org/citation.cfm?id=1325851.1325871.
[11] W. Lindner, J. Meier, Towards a secure data stream management system, in: Proceedings of the 31st VLDB Conference on Trends in Enterprise Application Architecture, in: TEAA'05, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 114–128, doi:10.1007/11681885_10.
[12] A.J. Menezes, S.A. Vanstone, P.C.V. Oorschot, Handbook of Applied Cryptography, 1st, CRC Press, Inc., Boca Raton, FL, USA, 1996.
[13] S. Nath, H. Yu, H. Chan, Secure outsourced aggregation via one-way chains, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '09, ACM, New York, NY, USA, 2009, pp. 31–44, doi:10.1145/1559845.1559851.
[14] R. Ostrovsky, W.E. Skeith, Private Searching on Streaming Data, in: V. Shoup (Ed.), Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14–18, 2005. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 223–240, doi:10.1007/11535218_14.
[15] S. Papadopoulos, G. Cormode, A. Deligiannakis, M. Garofalakis, Lightweight authentication of linear algebraic queries on data streams, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '13, ACM, New York, NY, USA, 2013, pp. 881–892, doi:10.1145/2463676.2465281.
[16] S. Papadopoulos, Y. Yang, D. Papadias, CADS: continuous authentication on data streams, in: Proceedings of the 33rd International Conference on Very Large Data Bases, in: VLDB '07, VLDB Endowment, 2007, pp. 135–146. http://dl.acm.org/citation.cfm?id=1325851.1325870.
[17] A. Perrig, R. Canetti, J.D. Tygar, D.X. Song, Efficient authentication and signing of multicast streams over Lossy channels, in: Proceeding 2000 IEEE Symposium on Security and Privacy. SP 2000, 2000, pp. 56–73, doi:10.1109/SECPRI.2000.848446.
[18] E. Shen, E. Shi, B. Waters, Predicate privacy in encryption systems, in: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography, in: TCC '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 457–473, doi:10.1007/978-3-642-00457-5_27.
[19] Y. Sun, Y. Yu, X. Li, K. Zhang, H. Qian, Y. Zhou, Batch verifiable computation with public verifiability for outsourcing polynomials and matrix computations, in: Proceedings, Part I, of the 21st Australasian Conference on Information Security and Privacy - Volume 9722, Springer-Verlag New York, Inc., New York, NY, USA, 2016, pp. 293–309, doi:10.1007/978-3-319-40253-6_18.
[20] Y. Xu, K. Wang, A.W.-C. Fu, R. She, J. Pei, Privacy-preserving data stream classification, in: C.C. Aggarwal, P.S. Yu (Eds.), Privacy-Preserving Data Mining: Models and Algorithms, Springer US, Boston, MA, 2008, pp. 487–510, doi:10.1007/978-0-387-70992-5_20.
[21] Z. Yang, R. Wright, H. Subramaniam, Experimental analysis of a privacy-preserving scalar protocol, Int. J. Comput. Syst. Sci.Eng. 21 (1) (2006) 47–52.
[22] L.F. Zhang, R. Safavi-Naini, Batch verifiable computation of polynomials on outsourced data, in: G. Pernul, P. Y A Ryan, E. Weippl (Eds.), Computer Security – ESORICS 2015: 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21–25, 2015, Proceedings, Part II, Springer International Publishing, Cham, 2015, pp. 167–185, doi:10.1007/978-3-319-24177-7_9.