2013

Verifiable and private top-k monitoring

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

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

DOI: https://doi.org/10.1145/2484313.2484388

Verifiable and Private Top-k Monitoring

Xuhua Ding
School of Information Systems
Singapore Management University
xhding@smu.edu.sg

HweeHwa Pang
School of Information Systems
Singapore Management University
hhpang@smu.edu.sg

Junzuo Lai *
Department of Computer Science,
Jinan University, China
junzuolai@smu.edu.sg

ABSTRACT

In a data streaming model, records or documents are pushed from a data owner, via untrusted third-party servers, to a large number of users with matching interests. The match in interest is calculated from the correlation between each pair of document and user query. For scalability and availability reasons, this calculation is delegated to the servers, which gives rise to the need to protect the privacy of the documents and user queries. In addition, the users need to guard against the eventuality of a server distorting the correlation score of the documents to manipulate which documents are highlighted to certain users.

In this paper, we address the aforementioned privacy and verifiability challenges. We introduce the first cryptographic scheme which concurrently safeguards the privacy of the documents and user queries in such a data streaming model, while enabling users to verify the correlation scores obtained. We provide techniques to bound the computation demand in decrypting the correlation scores, and we demonstrate the overall practicality of the scheme through experiments with real data.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Query processing; H.2.7 [**Database Administration**]: Security, integrity, and protection

Keywords

Vector product, correlation computation, verifiability, privacy

1. INTRODUCTION

Data streaming applications like Web access analysis, profile-driven marketing, environment sensing, stock trading and online bidding are now commonplace, driven by the widespread adoption of mobile devices and RFID technology. Such applications can generate high volume of data, which are often streamed to an intermediary for query processing and analysis to produce aggregate results for end-user consumption. As the intermediary may not be trusted by the data owner/generator or the end-users, the data and queries need to be protected.

To motivate our data streaming setting, suppose that Figure 1 depicts the system model of a surveillance application. There are three parties in the model – a data Owner, one or more Servers, and many Users. The Owner operates various security checkpoints of a country or sensitive installation. Here, 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, jaw, etc. are automatically extracted. The original picture and feature vector form a document $\mathbf{d}$ that is streamed to a shared Server. One of the Users is an intelligence agency which is monitoring a list of subjects. The intelligence agency is likely to have pictures of each subject; the feature vector extracted from each picture is registered 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 k documents that best match the query are displayed on the agency's alert screen. Since the Server is administered by another agency or an outsourced service provider whose operators may not have appropriate security clearances, the following system and security requirements become necessary:

- *Query privacy*. The feature values in $\mathbf{q}$ must be known only to the particular User who issued it. The reason is while the original picture cannot be reconstructed directly from $\mathbf{q}$, an adversary could extract feature vectors from his own pictures and attempt to match them against $\mathbf{q}$ to discover what the intelligence agency is interested in. Furthermore, the actual correlation score between $\mathbf{d}$ and $\mathbf{q}$ should be available only to the issuing User. Neither the Owner who possesses $\mathbf{d}$, the Server that computes the score nor other users can be allowed to deduce $\mathbf{q}$.
- *Document privacy*. The feature values in $\mathbf{d}$ must not be revealed to the Server, so as to prevent an adversary from matching $\mathbf{d}$ against his own pictures. To safeguard the rights of the visitors who have their pictures taken, the Owner may selectively restrict the features in $\mathbf{d}$ that various Users are allowed to query over.
- *Verifiability*. Without direct access to $\mathbf{d}$, the User who issued $\mathbf{q}$ would want to verify the correlation score between $\mathbf{d}$ and $\mathbf{q}$. This is to guard against an adversary at the Server manipulating the score of $\mathbf{d}$ to elude detection by the User.
- *Result ranking*. In order to identify the k most promising document matches, the User needs to be able to retrieve the actual correlation score for each $\mathbf{d}$.

The system and security requirements of our data streaming model exceed the capability of existing schemes in the literature. In the

*The author carried out this work at the Singapore Management University.

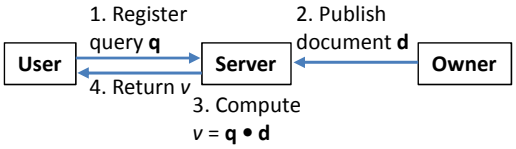


Figure 1: Data Streaming Model

data stream security literature, verifiability and privacy have been addressed separately. The former includes [12] which allows users to verify the arrival, removal and update of data falling within a selection range on an attribute, and [7] 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, the most relevant line of work is predicate encryption that supports the inner product of two vectors, e.g., [6]. However, predicate encryption schemes are not designed for our data streaming model in which the document vector $\mathbf{d}$ and the query vector $\mathbf{q}$ are generated by different parties and must be kept secret from each other.

CONTRIBUTIONS In this paper, we propose a Cryptographic Verifiable and Private Monitoring (CVPM) scheme to enable our data streaming model in Figure 1 which satisfies the aforementioned security requirements. In CVPM, each User encrypts her standing query vectors to safeguard their privacy; the encrypted query vectors are lodged with the Server. Whenever the Owner has a document to release, he generates for it an encrypted document vector which is distributed to the Server. Upon receiving a new encrypted document vector $\mathbf{d}$, the Server computes the protected correlation coefficient $v = \mathbf{q} \cdot \mathbf{d}$ for each query $\mathbf{q}$, and outputs v to the issuing User. The User then deciphers the output to obtain and verify the correlation score.

To the best of our knowledge, our scheme is the first for the data streaming model that concurrently achieves verifiability and privacy protection. Specifically, it ensures that the document vectors cannot be exploited by the Server or the User, other than for computing correlation scores. In addition, neither the Owner nor the Server gains an advantage in compromising the privacy of the user queries. Finally, the Server cannot tamper with the document correlation scores without being detected by the User. Through extensive experiments involving real datasets, we also demonstrate the practicality of CVPM for a broad spectrum of applications.

2. RELATED WORK

Data stream security has been studied extensively in various research communities. Table 1 summarizes the characteristics of the most relevant ones with respect to the system and security requirements identified in the Introduction.

In [8], Lindner and Meier discuss general security concerns in architecting a data stream management system. 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. [3] and [11] 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 by a different group of studies. These include [12] which allows users to verify the arrival, removal and update of data falling within a selection range on an attribute, [7] for authenticating selection-aggregation queries over an attribute of interest, and [10] for verifying the output of aggregation functions like MAX and SUM. None of the above schemes support the correlation computation that we need. Moreover, they provide either privacy or verifiability, but not both.

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 $\mathbf{q}$
n	$n = p_1 p_2$ where p_1, p_2 are distinct, large prime numbers
$\mathbb{G}, \mathbb{G}_T$	Multiplicative groups of order n
$\hat{e}$	Bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$
$\mathbb{G}_1$	Subgroup of $\mathbb{G}$, of order p_1
$\mathbb{G}_2$	Subgroup of $\mathbb{G}$, of order p_2
$H(.)$	A one-way hash function

Table 2: Notation

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 [6], an entity possessing a secret key token associated with a vector $\mathbf{x}$ can decrypt a public key encryption ciphertext associated with another vector $\mathbf{y}$ on the condition that $\mathbf{x} \cdot \mathbf{y} = 0$. 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), who can infer $\mathbf{x}$ by choosing any $\mathbf{y}$ and producing a ciphertext with the public key. Note that in our model, we require the privacy of both vectors against the Server.

The above privacy issue is addressed by the symmetric-key predicate encryption scheme in [13]. Here, the 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.

Our problem setting is also related to privacy-preserving scalar product schemes such as [5, 15]. 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.

3. CRYPTOGRAPHIC VERIFIABLE AND PRIVATE MONITORING SCHEME

In this section, we introduce our CVPM scheme for the data streaming model. Table 2 summarizes the frequently used notations, which will be explained as they are used.

3.1 Overview

As shown in Figure 1, our data streaming model comprises a set of Users who issue standing queries $\mathbf{q}$, an Owner who generates a stream of documents $\mathbf{d}$, and one or more Servers that match $\mathbf{q}$ with $\mathbf{d}$. The Server may also enforce access policies on

Scheme	Query Privacy	Document Privacy	Verifiability	Result Ranking
Random noise injection [1]	No	Yes	No	No
Data condensation [2]	No	Yes	No	No
Classifier training on data stream [14]	No	Yes	No	No
Keyword search on file stream [3], [11]	Yes	No	No	No
Authenticate streamed data [12]	No	No	Yes	No
Authenticate selection-aggregation queries [7]	No	No	Yes	No
Authenticate aggregation functions [10]	No	No	Yes	No

Table 1: Properties of Existing Security Schemes for Data Streaming Model

which features in $\mathbf{d}$ can be queried by particular users. Our CVPM scheme is made up of six algorithms that are executed by the parties in the model: `Setup`, `UserReg`, `QueryGen`, `DataGen`, `ServPro` and `UserDec`.

Owner first runs `Setup` to initialize the cryptographic setting shared by all participants and generate its own secret key. To join the system, a User first registers with the Owner. If the User satisfies the admission policy, the Owner executes `UserReg` to (i) produce a secret for the User to encrypt her queries, and (ii) produce a secret that enables the Server to process the User's queries subsequently. Based on her interest, the User runs `QueryGen` to generate the encryption $\mathbf{Q}_u$ of her query $\mathbf{q}$ and deposits $\mathbf{Q}_u$ with the Server. No two users share the same secret. This ensures that the encrypted queries are user-specific; in other words, identical queries from two users would lead to different encrypted queries.

At runtime, the Owner generates a stream of documents as well as the corresponding document vectors. For each newly published document vector $\mathbf{d}$, the Owner runs `DataGen` to produce an encrypted document vector $\mathbf{D}$ for the Server; $\mathbf{D}$ is common to *all* the Users in the system. The Server then executes `ServPro` for each registered query $\mathbf{Q}_u$ and returns the encrypted correlation score v to the corresponding User.

On receiving the response, the User runs `UserDec` with her secret to recover v. We provide constraints that bound the search space for v, so that its decryption is computationally feasible. A User cannot use her secret to decrypt correctly the encrypted scores intended for other users.

Adversarial model & security objectives The adversary could be a Server that executes the protocol honestly but is curious to know the document features and user queries, or cheats in computing the document scores. The adversary could also be an unauthorized party who managed to intercept and issue messages in the system. We assume the adversary is rational in that it does not run denial-of-service attacks. The security objectives include query privacy, document privacy and verifiability, as defined in the Introduction.

3.2 Solution Construction

Setup The Owner carries out the following initialization steps. It first generates two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$, both of order $n = p_1 p_2$ where p_1, p_2 are distinct prime numbers, and g is a generator of $\mathbb{G}$. Furthermore, there exists a non-degenerate and efficient bilinear mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Hence, $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T = \langle \hat{e}(g, g) \rangle$. The Owner computes $g_1 = g^{p_2}$, $g_2 = g^{p_1}$, and $h_1 = \hat{e}(g_1, g_1)$, $h_2 = \hat{e}(g_2, g_2)$. Let $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ denote two subgroups in $\mathbb{G}$ of order p_1 and p_2, respectively. The Owner then selects x randomly from $\mathbb{Z}_n$, denoted as $x \in_R \mathbb{Z}_n$. The secret key of the Owner is $sk = (g_1, h_1, x)$. It publishes $(\mathbb{G}, g, n, g_2, h_2)$ to enable the Server and Users to perform group operations in $\mathbb{G}$.

User Registration The Owner executes Algorithm `UserReg` described in Figure 2 to initialize the User's state with $\{A_u, h_u, K_u\}$, and to update the Server with B_u.

Algorithm: `UserReg`
Input: sk; Output: $\{B_u\}$ to Server, $\{A_u, h_u, K_u\}$ to User

1. Select $a_u, b_u \in_R \mathbb{Z}_n$ such that $a_u + b_u = x \bmod n$.
2. Compute $A_u = g^{a_u}$ and $B_u = g^{b_u}$.
3. Select $k_u \in_R \mathbb{Z}_n$; compute $h_u = h_1^{k_u}$ and $K_u = g_1^{k_u}$.
4. Securely send B_u to the Server; securely send $\{A_u, h_u, K_u\}$ to the User.

Figure 2: User Registration by the Owner

Query submission With $\{A_u, h_u, K_u\}$ from the Owner, the User initializes her standing query request $\mathbf{q} = \{q_i\}_{i=1}^m$ where $q_i \in [0, 2^{k_q}) \subseteq \mathbb{Z}_{p_1}$, according to her interests; k_q is the bit length of each coordinate in $\mathbf{q}$. She deposits the encryption $\mathbf{Q}_u$ of $\mathbf{q}$ with the Server as a standing query over the data stream. The algorithm for the User's query generation is described in Figure 3.

Algorithm: `QueryGen`
Input: K_u, $\mathbf{q} = \{q_i\}_{i=1}^m$; Output: $\mathbf{Q}_u$ to Server, $\mathbf{r}$ to User

1. Choose a random vector $\mathbf{r} = \{r_i\}_{i=1}^m$ with $r_i \in \mathbb{Z}_n$.
2. Compute vector $\mathbf{Q}_u = \{Q_i\}_{i=1}^m$ where $Q_i = K_u^{q_i} g_2^{r_i}$ for every $i \in [1, m]$.
3. Securely send $\mathbf{Q}_u$ to the Server.

Figure 3: Query Generation by Users

Data encryption For each document vector $\mathbf{d} = \{d_i\}_{i=1}^m$ where $d_i \in [0, 2^{k_d}) \subseteq \mathbb{Z}_{p_1}$ and k_d is the bit length of the coordinates in $\mathbf{d}$, the Owner prepares an encrypted document vector $\mathbf{D}$ by running Algorithm `DataGen` described in Figure 4. $\mathbf{D}$ is the common document ciphertext for all the users in the system. In the algorithm, the Owner encrypts each d_i with a random number t_i derived from a fresh seed r, and $H(.)$ is a secure one-way hash function.

Algorithm: `DataGen`
Input: sk, $\mathbf{d} = \{d_i\}_{i=1}^m$; Output: $\{\mathbf{D}, C\}$ to Server

1. Choose $r \in_R \mathbb{Z}_n$ and compute $C = g^r$.
2. Set $t_i = H(i, \hat{e}(g^r, g^x))$ for every $i \in [1, m]$.
3. Compute $\mathbf{D} = \{D_i\}_{i=1}^m$ where $D_i = g_1^{d_i} g_2^{t_i}$.
4. Send $\{\mathbf{D}, C\}$ to the Server.

Figure 4: Data Generation by the Owner

Server processing On receiving encrypted document vector $\mathbf{D}$, the Server executes Algorithm `ServPro` shown in Figure 5 for all the users in its service domain. Given registered standing query $\mathbf{Q}_u$ from a User, the algorithm first computes a ciphertext of the correlation score for the User, i.e., an encrypted form of the inner product of $\mathbf{d}$ and $\mathbf{q}$. Then, it computes a partial result to aid the User in deciphering the correlation score.

User decryption The User executes Algorithm `UserDec` in Figure 6 to get the final correlation score v. With each coordinate in

Algorithm: `ServPro`
Input: $\{\mathbf{D}, C\}$, $\{\mathbf{Q_u}, B_u\}$; Output: $\{W, C, C_1\}$ to User

For every standing query $\mathbf{Q_u}$, perform the following steps:

1. Compute $w_i = \hat{e}(D_i, Q_i)$ $\forall 1 \leq i \leq m$; then compute $W = \prod_{i=1}^{m} w_i$.
2. Compute $C_1 = \hat{e}(C, B_u)$.
3. Send $\{W, C, C_1\}$ to the User.

Figure 5: Correlation Computation by the Server

$\mathbf{d}$ and $\mathbf{q}$ having bit length k_d and k_q respectively, v is in the range of $[0, 2^{k_d+k_q} \cdot m_q)$ where $m_q \leq m$ is the number of coordinates that the User specified in $\mathbf{q}$. However, the User may predefine a threshold t to filter out documents with low correlation scores, which signify that they are too dissimilar to the User's interest and can be discarded without examination. This limits the search for v to a narrower range $[t, 2^{k_d+k_q} \cdot m_q)$ in Step 5. Note that t is chosen by and known only to the User.

Furthermore, when the document streaming rate is high, the User may wish to monitor only the top-k documents within a sliding window (of the most recent documents, or of documents that arrived within the last several seconds). The correlation score v_k of the k-th highest ranking document provides another threshold 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:

$$v \in [\max\{t, v_k\}, 2^{k_d+k_q} \cdot m_q) \tag{1}$$

The constraints in the formula enable the User to avoid a full discrete logarithm computation for v, by exploiting secret application-specific values t, v_k and m_q which only the User knows. If v is found, the User may request for the actual document from the Owner; otherwise, the User ignores the document.

Algorithm: `UserDec`
Input: $\{W, C, C_1\}$, $\{\mathbf{r}, h_u, A_u\}$; Output: v

1. Compute $C_2 = \hat{e}(C, A_u)$ and $s = C_1 C_2$.
2. For $1 \leq i \leq m$, compute $t_i = H(i, s)$.
3. Compute $R = \sum_{i=1}^{m} r_i t_i$.
4. Compute $W' = W / h_2^R$.
5. Check whether there exists $v \in [\max\{t, v_k\}, 2^{k_d+k_q} \cdot m_q)$ such that $h_u^v = W'$, t is a relevance threshold, v_k is the score of the current k-th result document, and m_q is the number of coordinates used in $\mathbf{q}$.

Figure 6: User Decryption

Where the Owner has specified access policies, W in Algorithm `ServPro` and R in Algorithm `UserDec` would aggregate over only features that the User is authorized to query.

Finally, if any input parameter to `UserDec` has been tampered with, with overwhelming probability R would not match, so W' would have order n and there exists no v that satisfies $h_u^v = W'$.

3.3 Decryption of Document Score

The `UserDec` algorithm requires a discrete logarithm in Step 5 to get the document score v. Although general discrete logarithm is considered to be intractable for a large cyclic group, it is feasible for the User to find v within the restricted range in Formula 1 with t, v_k and m_q which are known only to her. If the range for v is narrow, the User can pre-generate a look-up table for v. If not, the User executes the baby-step giant-step procedure, with space and time complexities $O(\sqrt{2^{k_d+k_q} \cdot m_q})$, to find v. 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, we expect the time and space complexity of the baby-step giant-step procedure to be tolerable. Additionally, the optimization opportunities described below often allow the discrete logarithm computation to terminate early. We will validate the practicality of the optimized `UserDec` algorithm through experiments in Section 4.

3.3.1 Optimizing Document Score Decryption

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 [9]. Although the latter two are superior in asymptotic time complexity, they are difficult to optimize to take advantage of the constraints expressed in Formula 1. 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 $h_u^v = W'$ in the `UserDec` algorithm.

Let $\tau = \left\lceil \sqrt{2^{k_d+k_q} \cdot m_q} \right\rceil$ and $v = c_1 \cdot \tau + c_0$ for some $0 \leq c_1 < \tau$ and $0 \leq c_0 < \tau$. Thus, $h_u^{-c_1\tau} W' = h_u^{c_0}$. The User creates beforehand a lookup table for $\langle h_u^{c_0}, c_0 \rangle$ with $h_u^{c_0}$ as search key, for $0 \leq c_0 < \tau$. For each v, the User iteratively checks whether $h_u^{-c_1\tau} W'$ exists in the lookup table, decrementing c_1 from $\tau - 1$ down towards $\lfloor \max(t, v_k)/\tau \rfloor$. The procedure can 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 $\mathbf{d}_i$ for which the 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 will lower their $\overline{v}_i$'s uniformly (by decrementing c_1 in the computation of their document scores in unison). This allows the User to fully decrypt the higher document scores first, and discover the more relevant candidates that are likely to replace the next expiring document in the top-k result.

3.3.2 Computing Document Score in Memory Constrained User Devices

The optimization in Section 3.3.1 limits the search for the document score as expressed in Formula 1, and is particularly helpful when $2^{k_d+k_q} \cdot m_q$ is large. However, the size of the lookup table remains at $\lceil \sqrt{2^{k_d+k_q} \cdot m_q} \rceil$ entries, which may exceed the available memory in user devices that are severely resource-constrained. We now discuss how our extended baby-step giant-step procedure trades computation for memory space in finding $v \in [\max(t, v_k), 2^{k_d+k_q} \cdot m_q)$ such that $h_u^v \equiv W'$ for given h_u and W'.

The User begins by allocating a lookup table within the available memory space. Let τ denote the number of entries in the table. We represent v as a polynomial $v = c_\alpha \tau^\alpha + c_{\alpha-1}\tau^{\alpha-1} + \ldots + c_0$, where $\alpha = \lfloor \log_\tau (2^{k_d+k_q} \cdot m_q) \rfloor$ and $c_i \in [0, \tau)$ $\forall i \in [0, \alpha]$. Thus, $h_u^{-(c_\alpha \tau^{\alpha-1} + \ldots + c_1)\tau} W' = h_u^{c_0}$. As before, the User populates the lookup table for $\langle h_u^{c_0}, c_0 \rangle$ with $h_u^{c_0}$ as search key, for $0 \leq c_0 < \tau$. Following that, the User iteratively checks whether $h_u^{-i\tau} W'$ exists in the lookup table, decrementing i from $\tau^\alpha - 1$ towards $\lfloor \max(t, v_k)/\tau \rfloor$.

3.4 Compressing Document & Query Vectors

In certain applications, the document vector $\mathbf{d}$ and query vector $\mathbf{q}$ are expected to contain mostly zero coordinates. One such example is the vector space model for text retrieval [16], where each feature corresponds to a dictionary word. Since most of the documents and queries contain only a very small subset of the dictionary words, only a small number of coordinates in the document and query vectors are non-zero. In this situation, the Owner may leave out some of the zero coordinates in $\mathbf{d}$, so that the Server can exclude them from query processing. This reduces the server processing overhead, at the expense of exposing the dropped coordinates in $\mathbf{d}$. Likewise, the User may drop some of the zero coordinates in $\mathbf{q}$. The concrete scheme is as follows.

For each $\mathbf{d}$, the Owner creates a certified Bloom filter [4] on the index position of the non-zero coordinates in $\mathbf{d}$. After that, those zero coordinates whose index gets a false positive on the Bloom filter must also stay in $\mathbf{d}$; the remaining zero coordinates whose index gets a negative on the Bloom filter may be omitted.

As proposed in [4], a Bloom filter is designed to support membership checks on a set $\mathcal{F}$ of f key values, $\mathcal{F} = \{F_1, F_2, \ldots, F_f\}$. To construct a Bloom filter with ϕ bits, we choose ρ independent hash functions $H_1, H_2, \ldots, H_\rho$, each with a range of $[1, \phi]$. For each value $F_i \in \mathcal{F}$, the filter bits at positions $H_1(F_i)$, $H_2(F_i)$, ..., $H_\rho(F_i)$ are set to 1. To check whether a given F is in $\mathcal{F}$, we examine the bits at $H_1(F), H_2(F), \ldots, H_\rho(F)$. If any of the bits is 0, F cannot possibly be in $\mathcal{F}$; otherwise there is a high probability that F is in $\mathcal{F}$. In other words, the Bloom filter admits controlled false positive rates but no false negatives. The false positive rate is

$$FP = \left(1 - \left(1 - \frac{1}{\phi}\right)^{\rho f}\right)^{\rho} \approx \left(1 - e^{-\rho f/\phi}\right)^{\rho} \qquad (2)$$

Mathematically, FP is minimized for $\rho = \phi \ln 2/f$. Since ρ must be an integer, we will use $\rho = \lfloor \phi \ln 2/f \rfloor$. Given the value of f and the target FP rate, we can thus set ρ and ϕ accordingly.

Applying the Bloom filter to our setting, f corresponds to the number of non-zero coordinates in $\mathbf{d}$, and the number of false positives is $FP \times (m - f)$. Thus, the Server sees $f + FP \times (m - f)$ remaining coordinates in $\mathbf{d}$. Together with the parameters ϕ and ν, the Server can deduce the value of f, i.e., the actual number of non-zero coordinates among the remaining coordinates. If we want to prevent that, then the index positions of some of the zero coordinates should go into the Bloom filter too, so that f yields only a loose upper bound on the number of non-zero coordinates.

Likewise for $\mathbf{q}$, the User may randomly drop some of the zero coordinates. The remaining coordinates, along with their index positions, are submitted to the Server.

In processing a document $\mathbf{d}$ against a query $\mathbf{q}$, the Server evaluates the inner product $\mathbf{q} \cdot \mathbf{d}$ only over the common coordinates in $\mathbf{q}$ and $\mathbf{d}$ that are accessible to each User. Along with the answer W', the Server returns the index positions $\mathcal{I}$ that are in $\mathbf{q}$ but have been omitted from $\mathbf{d}$, as well as the certified Bloom filter. The user can verify $\mathcal{I}$ against the Bloom filter, which by construction will not produce false positives on the indices in $\mathcal{I}$.

4. EMPIRICAL VALIDATION

In this section, we evaluate the overall practicality of our CVPM scheme, and the effectiveness of the optimization techniques introduced in Section 3.3.1 in particular.

4.1 Experiment Set-Up

Datasets: To ensure that our observations are generalizable, we have run CVPM on several datasets. Here we report on two real datasets from the UCI KDD Archive[1]. The datasets are picked because they vary from each other in key properties that stress different algorithms in CVPM. The first, Corel Image, contains feature vectors extracted from 68,040 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 with the main motivating application in Section 1.

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. All the attributes are in the range [0, 20]. The correlation between vectors are high, relative to that in the Corel Image data. The data in this collection essentially are 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 hour, and the k documents with the highest correlation scores in this window constitute the query result. For each experiment setting, the performance measures are averaged across the 100 queries.

We implemented the CVPM scheme in C language, on the PBC cryptography library from Stanford University[2]. The experiments are run on a MacBook Pro with 2.8 GHz Intel Core i7 CPU and 8 GB of main memory.

Metrics: Our primary performance metrics include: (a) the average time taken by the Owner in executing the `DataGen` algorithm on a new document; (b) the average time taken by the Server to execute the `ServPro` algorithm for each document-query pair; and (c) the 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 3.3.1.

4.2 Corel Image Experiment

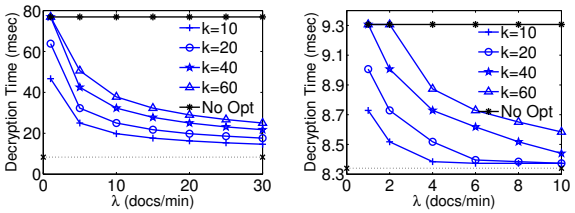
For the Corel Image data, we set the sliding window w to one hour and vary λ from 1 to 30 documents/minute. Referring to Formula 1, the threshold t is set to 0, so pruning of the search space for the correlation score v of a new document relies solely on v_k, the correlation score of the k-th result document. Figure 7(a) plots the average user execution time per query for $k = 10, 20, 40$ and 60. The figure also gives the execution time of 'No Opt', which disables the optimization techniques in Section 3.3.1.

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 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 k-th highest correlation score and narrowing the search space for v. The execution time is bounded from below by the cost of the bilinear pairing operation $\hat{e}(.,.)$ (indicated by the dotted horizontal line). 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 execution time incurred by the Owner and Server are summarized in Table 3. The cost of the `DataGen` algorithm, incurred

[1] http://kdd.ics.uci.edu/databases/

[2] http://crypto.stanford.edu/pbc/



(a) Corel Image Data (b) US Census Data

Dataset	Owner	Server
Corel Image	217.80	275.56
US Census	449.82	576.19

Table 3: Owner and Server Processing Times (msec)

once for every new document, is just over 200 msec; this cost is dominated by the exponentiation operations in Step 3 of the algorithm. The cost of the `ServPro` algorithm is incurred once for every document-query pair and stands at 275 msec; this cost is dominated by the bilinear pairing operations in Step 1 of the algorithm. These performance levels can be maintained for higher data rates by employing multiple CPUs.

4.3 US Census Experiment

Turning to the US Census data, we again set the sliding window w at one hour. Here λ varies from 1 to 10 documents/minute. The average user execution time per query is summarized in Figure 7(b). The qualitative behavior of CVPM observed here is the same as in the previous experiment. Quantitatively, the execution times are lower now because the narrower feature domains (i.e., smaller k_d and k_q values in Formula 1) constrain the search space for the correlation score v of arriving documents.

Finally, the corresponding execution time of the Owner and Server are reported in Table 3. Both 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.

4.4 Summary of Experiment Results

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

- The optimization arising from Formula 1 is 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 k-th result document.
- The User cost is sensitive to the number of features and their domain sizes. If there are many features with large domains, it may be necessary to bucketize the feature values; effectively, this induces a coarser resolution on the features.
- Where the feature vectors have high dimensionality, it may be necessary to parallelize the computation at the Owner and Server. This is straightforward to implement, as the computation on different coordinates in the document vector can be carried out independently.
- CVPM delivers acceptable performance for many practical application settings, with sub-second execution time for every party in the data streaming model.

5. CONCLUSION

In this paper, we formulate the security requirements in a data streaming model. The model employs an untrusted server to compute the correlation scores between documents streamed from the data owner, and the standing queries issued by users. The correlation computation translates to an inner product of the respective document and query vectors. We present the first cryptographic scheme that concurrently safeguards the privacy of the documents and queries, while enabling users to verify the correctness of the correlation scores received. Through extensive experiments, we demonstrate that the proposed scheme achieves practical execution time for a wide spectrum of application settings.

Acknowledgements

HweeHwa Pang is supported by Research Grant 12-C220-SMU-004 from the Singapore Management University.

6. REFERENCES

[1] C. C. Aggarwal. On randomization, public information and the curse of dimensionality. In *Proc. of ICDE*, 2007.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of ACM SIGMOD*, 2000.

[3] J. Bethencourt, D. X. Song, and B. Waters. New constructions and practical applications for private stream searching. In *Proc. of IEEE S& P*, 2006.

[4] B. Bloom. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7):422–426, July 1970.

[5] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On private scalar product computation for privacy-preserving data mining. In *Proc. of ICISC*, 2004.

[6] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, April 2008.

[7] F. Li, K. Yi, M. Hadjieleftheriou, and G. Kollios. Proof-infused streams: Enabling authentication of sliding window queries on streams. In *Proc. of VLDB*, pages 147–158, September 2007.

[8] W. Lindner and J. Meier. Towards a secure data stream management system. In *Trends in Enterprise Application Architecture*, pages 114–128, August 2005.

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[10] S. Nath, H. Yu, and H. Chan. Secure outsourced aggregation via one-way chains. In *Proc. of ACM SIGMOD*, 2009.

[11] R. Ostrovsky and W. E. S. III. Private searching on streaming data. In *Advances in Cryptology (CRYPTO)*, 2005.

[12] S. Papadopoulos, Y. Yang, and D. Papadias. CADS: Continuous authentication on data streams. In *Proc. of VLDB*, pages 135–146, September 2007.

[13] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Proc. of TCC*, 2009.

[14] Y. Xu, K. Wang, A. W.-C. Fu, R. She, and J. Pei. Privacy-preserving data stream classification. In *Privacy-Preserving Data Mining*. Springer, 2008.

[15] Z. Yang, R. Wright, and H. Subramaniam. Experimental analysis of a privacy-preserving scalar protocol. *International Journal of Computer Systems Science and Engineering*, 21(1):47–52, January 2006.

[16] J. Zobel and A. Moffat. Inverted Files for Text Search Engine. *ACM Computing Surveys*, 38(2):6, July 2006.