Hindawi
Security and Communication Networks
Volume 2017, Article ID 6841216, 12 pages
https://doi.org/10.1155/2017/6841216

Research Article

Efficient Two-Step Protocol and Its Discriminative Feature Selections in Secure Similar Document Detection

Sang-Pil Kim,[1] Myeong-Sun Gil,[1] Hajin Kim,[1] Mi-Jung Choi,[1] Yang-Sae Moon,[1] and Hee-Sun Won[2]

[1]*Department of Computer Science, Kangwon National University, 1 Kangwondaehak-gil, Chuncheon-si, Gangwon 24341, Republic of Korea*
[2]*Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Republic of Korea*

Correspondence should be addressed to Yang-Sae Moon; ysmoon@kangwon.ac.kr

Received 27 July 2016; Revised 31 January 2017; Accepted 6 February 2017; Published 28 March 2017

Academic Editor: Kai Rannenberg

Recently, the risk of information disclosure is increasing significantly. Accordingly, privacy-preserving data mining (PPDM) is being actively studied to obtain accurate mining results while preserving the data privacy. We here focus on secure similar document detection (SSDD), which identifies similar documents of two parties when each party does not disclose its own sensitive documents to the another party. In this paper, we propose an efficient two-step protocol that exploits a feature selection as a lower-dimensional transformation, and we present discriminative feature selections to maximize the performance of the protocol. The proposed protocol consists of two steps: the *filtering* step and the *postprocessing* step. For the feature selection, we first consider the simplest one, random projection (RP), and propose its two-step solution, SSDD-RP. We then present two discriminative feature selections and their solutions: SSDD-LF which selects a few dimensions locally frequent in the current querying vector and SSDD-GF which selects ones globally frequent in the set of all document vectors. We finally propose a hybrid one, SSDD-HF, which takes advantage of both SSDD-LF and SSDD-GF. We empirically show that the proposed two-step protocol significantly outperforms the previous one-step protocol by three or four orders of magnitude.

1. Introduction

Similar document detection is the problem of finding similar documents of two parties, Alice and Bob, and has been widely used in the version management of files, copyright protection, and plagiarism detection [1]. Secure similar document detection (SSDD) [2] has been recently introduced to identify similar documents while preserving the privacy of each party's documents, as shown in Figure 1. That is, SSDD finds similar document pairs whose cosine similarity [3] exceeds the given tolerance while not disclosing document vectors to the other party. SSDD is a typical example of privacy-preserving data mining (PPDM) [4–6] and has the following applications:

(i) Detection of double submissions [2, 7]: in two or more conferences that are not allowing double submissions, SSDD finds the double-submitted papers while not disclosing the papers to the other conference(s).

(ii) Detection of insurance fraud [2]: in an insurance fraud detection system, SSDD searches for similar accident cases of two or more insurance companies while not providing sensitive or private cases to the other company or companies.

(iii) Sharing of similar symptom cases [8]: doctors or patients of a hospital want to find similar symptom cases with the prescriptions or medical treatment histories of other hospitals without disclosing each other's medical or health information.

Jiang et al. [2] proposed a novel solution for SSDD by exploiting secure multiparty computations (SMCs) [9, 10] in a semihonest model. Their solution has preserved the privacy of two parties by using the secure scalar product in computing the cosine similarity between document vectors. They suggested the use of random matrix or homomorphic encryption methods [9], which are representative methods of

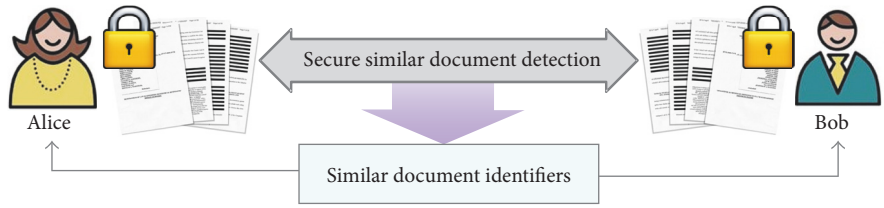


Figure 1: Concept of secure similar document detection.

secure scalar product. In this paper, we use the SSDD solution using the random matrix method by Jiang et al. as the base protocol, which we call SSDD-Base. However, SSDD-Base has a critical problem in that it incurs severe computation and communication overhead. Letting Alice's and Bob's document sets be $\mathbb{U}$ and $\mathbb{V}$, respectively, SSDD-Base requires $|\mathbb{U}||\mathbb{V}|$ secure scalar products. In many cases, the dimension n of the document vectors reaches tens of thousands or even hundreds of thousands in number, and SSDD-Base incurs a very high complexity of $O(n|\mathbb{U}||\mathbb{V}|)$, which is not practical to support a large volume of document databases. In particular, if there are many parties or frequent changes in document databases, the overhead becomes much more critical.

To alleviate the computation and communication overhead of SSDD-Base, in this paper, we present a two-step protocol that exploits the feature selection of a lower-dimensional transformation. The feature selection transforms high-dimensional document vectors into low-dimensional feature vectors, and in general, it selects tens to hundreds dimensions from thousands to tens of thousands of dimensions. We call this feature selection FS. Representative FS includes a random projection (RP) [11], the document frequency (DF), and a linear discriminant analysis (LDA) [12]. In this paper, we use RP and DF because they are known to be simple but efficient feature selections. To devise a two-step protocol, we need to find an upper bound of the cosine similarity for the filtering process. Thus, we first present an upper bound of FS and formally prove its correctness. Using the upper bound property of FS, we then propose a generic two-step protocol, called SSDD-FS. The proposed SSDD-FS works as follows: in the first *filtering* step, it converts n-dimensional vectors into f ($\ll n$)-dimensional vectors and applies the secure protocol to f-dimensional vectors to filter out nonsimilar n-dimensional vectors; in the second *postprocessing* step, it applies the base Protocol SSDD-Base to the nonfiltered n-dimensional vectors. In the filtering step, SSDD-FS prunes many nonsimilar *high*-dimensional vectors by comparing *low*-dimensional vectors with relatively less complexity of $O(f|\mathbb{U}||\mathbb{V}|)$, and thus, it significantly improves the performance compared with SSDD-Base.

For an efficient SSDD-FS, the FS should be highly discriminative; that is, the FS should filter out as many high-dimensional vectors as possible if they are nonsimilar. In this paper, we analyze SSDD protocols in detail and propose four different techniques as the discriminative FS implementation. An RP can first be considered the easiest way of implementing the FS. The RP randomly selects f dimensions from n dimensions. An RP is easy, but its filtering effect will be very low owing to the randomness. To solve this problem, we exploit the DF that selects the feature dimensions based on the frequencies in all document vectors. In particular, by referring to the DF concept, we present three variants of DF, called the local frequency (LF), global frequency (GF), and hybrid frequency (HF). First, the LF considers the term frequencies of Alice's current querying vector (which we call the *current vector*) and selects high-frequency dimensions from the current vector. The LF focuses on the *locality*, which means that considering the current vector only might be sufficient to decrease the upper bound of the cosine similarity. Second, the GF means the DF itself; that is, the GF counts the number of documents containing each term (dimension), constructs a frequency vector from those counts (which we call the *whole vector*), and selects high-frequency dimensions from the whole vector. The GF focuses on the *globality* because it considers all document vectors. To implement the GF, however, we need to make a secure protocol for obtaining the whole vector from both Alice's and Bob's document sets. For this, we propose a protocol, SecureDF, as a secure implementation of the DF. Third, the HF takes advantage of both locality of the LF and globality of the GF. The HF computes a *difference vector* between the current and whole vectors and selects high-valued dimensions from the difference vector. This is because the HF tries to maximize the value difference between Alice's and Bob's vectors for each selected dimension and eventually decreases the upper bound of the cosine similarity. Table 1 summarizes these four feature selections and their corresponding SSDD protocols, SSDD-RP, SSDD-LF, SSDD-GF, and SSDD-HF, which are described in Section 4.

In this paper, we empirically evaluate the base protocol, SSDD-Base, and our four SSDD-FS protocols (SSDD-RP, SSDD-LF, SSDD-GF, and SSDD-HF) using various datasets. The experimental results show that SSDD-FS protocols significantly outperform SSDD-Base. This means that the proposed two-step protocols effectively prune a large number of nonsimilar sequences early in the filtering step. In particular, SSDD-HF, which takes advantage of both the locality of SSDD-LF and the globality of SSDD-GF, shows the best performance. Compared with SSDD-Base, SSDD-HF significantly reduces the execution time of SSDD by three or four orders of magnitude.

The rest of this paper is organized as follows. Section 2 explains related work and background of the research. Section 3 presents the FS-based two-step protocol, SSDD-FS, and proves its correctness. Section 4 introduces four novel feature selections, RP, LF, GF, and HF, and proposes their

Table 1: Feature selection methods used for SSDD-FS.

Method	Description
SSDD-RP	Randomly select f dimensions from an n-dimensional vector.
SSDD-LF	Select highly frequent f dimensions from Alice's n-dimensional current vector.
SSDD-GF	Select highly frequent f dimensions from an n-dimensional whole vector.
SSDD-HF	Select high-valued f dimensions from the n-dimensional difference vector between current and whole vectors.

corresponding secure protocols. Section 5 explains experimental results on various datasets. We finally summarize and conclude the paper in Section 6.

2. Related Work and Background

With increasing need of preserving privacy of big data, there have been many efforts on PPDM [13]. PPDM solutions can be classified into four categories: data perturbation, k-anonymization, distributed privacy preservation, and privacy preservation of mining results [14]. Among these categories, secure multiparty computation (SMC) has been used in distributed privacy preservation. As a recent secure multiparty computation of principal component analysis (PCA), Won et al. [15] proposed S-PCA that computes PCA without disclosing the sensitive data of individual nodes. Also, SSDD can be regarded as an application of distributed privacy preservation. Shah and Joshi [16] summarized PPDM technologies in a distributed environment. Representative SMCs include secure comparison, secure sum, and secure scalar product, and SSDD is an interesting application of exploiting the secure scalar product.

Jiang et al. [2] proposed a novel solution for SSDD by exploiting SMCs in a semihonest model. Their solution has preserved the privacy of two parties by using the secure scalar product in computing the cosine similarity between document vectors. They also proposed n-gram based SSDD protocols that are effective in detecting local similarity unlike the existing vector space model, which is effective in detecting global similarity [17]. Buyrukbilen and Bakiras [7] introduced a solution based on *simhash* to reduce high computational and communication costs. Simhash is a dimensionality reduction technique that encodes all the document terms and their frequencies into a fixed-size bit vector and made it possible to reduce the similarity calculation to a secure XOR computation between two bit vectors.

We use the cosine similarity as the basic operation of similar document detection. The cosine similarity of two n-dimensional vectors $\vec{U} = \{u_1, \ldots, u_n\}$ and $\vec{V} = \{v_1, \ldots, v_n\}$, which are normalized to size 1, is computed as $\vec{U} \cdot \vec{V}$, where $\vec{U} \cdot \vec{V}$ is the scalar product of $\vec{U}$ and $\vec{V}$; that is, $\vec{U} \cdot \vec{V} = \sum_{i=1}^{n} u_i v_i$. If we can compute $\vec{U} \cdot \vec{V}$ securely in two parties, we can also compute $\cos(\vec{U}, \vec{V})$ securely. There are two representative methods for a secure scalar product [2]. The first is a random matrix method, where two parties share the same random matrix, which they use to securely compute the scalar product. The second is the homomorphic encryption method [18], where two parties use a homomorphic probability key system for the secure computation of scalar products. In this paper, we use the random matrix method because it is more efficient than the homomorphic encryption method; however, we can instead use the homomorphic encryption method for the protocols discussed later. Without a loss of generality, we assume that vectors $\vec{U}$ and $\vec{V}$ are normalized to size 1. That is, $\|U\| = \|V\| = 1$, and thus, simply $\cos(\vec{U}, \vec{V}) = \vec{U} \cdot \vec{V}$.

Algorithm 1 shows the protocol of SSDD-Base, the recent SSDD solution by Jiang et al. [2]. SSDD-Base uses a random matrix method for secure scalar products, where Alice and Bob share the same matrix $\mathbf{A}$ and securely determine whether two vectors $\vec{U}$ and $\vec{V}$ are similar. For the correctness and a detailed description of *Protocol* SSDD-Base, readers are referred to [2]. In SSDD, we apply SSDD-Base for each pair of document vectors. More formally, if $\mathbb{U}$ and $\mathbb{V}$ are sets of document vectors owned by Alice and Bob, respectively, we apply SSDD-Base for each pair $(\vec{U}, \vec{V})$, where $\vec{U} \in \mathbb{U}$ and $\vec{V} \in \mathbb{V}$, respectively. As we mentioned in Section 1, however, SSDD-Base incurs the severe computation and communication overhead of $O(n\|U\|\|V\|)$, which will be very serious if there are several parties or if a large number of documents are changed dynamically. To alleviate this critical overhead, in this paper, we discuss a two-step solution for SSDD.

In text and time-series mining, many lower-dimensional transformations have been proposed to solve the dimensionality curse problem [19–21] of high-dimensional vectors. We can classify lower-dimensional transformations into feature extractions and feature selections [22]. First, the feature extraction *creates* a few *new* features from an original high-dimensional vector. Representative examples of feature extractions include latent semantic indexing (LSI) [23], locality preserving indexing (LPI), discrete Fourier transform (DFT) [24], discrete Wavelet transform (DWT), and piecewise aggregate approximation (PAA) [25]. In contrast, the feature selection *selects* a few *discriminative* features from the original (or transformed) high-dimensional vectors. Representative examples of feature selections include RP, DF, LDA, and principal component analysis (PCA) [11, 12]. In this paper, we use RP and DF with the appropriate variations. This is because RP and DF are much simpler than other transformations, and accordingly, they are easily applied to SSDD with low complexity; on the other hand, LSI, LPI, LDA, and PCA may provide very accurate feature vectors, but they are too complex to be applied to SSDD. For a detailed description of lower-dimensional transformations for text mining, readers are referred to [22, 23].

Protocol SSDD-Base
(1) $\vec{U}$ and $\vec{V}$ are n-dimensional vectors owned by Alice and Bob, respectively.
(2) Assume that Alice and Bob maintain the same $n \times n/2$ matrix $\mathbf{A} = [a_{i,j}]$.
Alice:
(1) Generate $n/2$ random numbers $r_1, r_2, \ldots, r_{n/2}$;
(2) Compute an n-dimensional vector $\vec{Z}$, where; $z_i = u_i + a_{i,1} \cdot r_1 + \cdots + a_{i,n/2} \cdot r_{n/2}$;
(3) Send $\vec{Z}$ to Bob;
Bob:
(4) Compute a scalar value $S = \vec{Z} \cdot \vec{V}$;
(5) Compute an $n/2$ -dimensional vector $\vec{Y}$, where $y_i = a_{1,i} \cdot v_1 + \cdots + a_{n,i} \cdot v_n$;
(6) Send $\vec{Y}$ with S to Alice;
Alice:
(7) Compute a scalar value $S' = \vec{Y} \cdot \vec{R}$;
(8) Obtain $\delta = S - S'$; // Note that $\delta = \vec{U} \cdot \vec{V}$
(9) **if** $\delta \geq \varepsilon$ **then** Identify $\vec{U}$ and $\vec{V}$ as a similar one; // ε = user-specified tolerance

Algorithm 1: Protocol of SSDD-Base.

There have been many efforts on PPDM [13]. PPDM solutions can be classified into four categories: data perturbation, k-anonymization, distributed privacy preservation, and privacy preservation of mining results [14]. SSDD can be regarded as an application of distributed privacy privation. For the detailed explanation on problems and solutions of data perturbation and k-anonymization, readers are referred to survey papers [4, 13].

3. Feature Selection Based Secure Two-Step Protocol

We use the FS for the secure two-step protocol. To transform an n-dimensional vector into an f-dimensional vector, the FS chooses randomly or highly frequent f dimensions from n dimensions, and thus, its transformation process is very simple. In this section, we first assume that the FS can select f dimensions from n dimensions in a secure manner, and we then propose a secure two-step protocol of SSDD using the secure FS.

To use a lower-dimensional transformation F for SSDD, we need to find an upper bound function $\text{upper}(\overrightarrow{U^F}, \overrightarrow{V^F})$ that satisfies (1), where $\overrightarrow{U^F}$ and $\overrightarrow{V^F}$ are f-dimensional feature vectors transformed from n-dimensional vectors, $\vec{U}$ and $\vec{V}$, respectively, by the transformation F. In (1), $\vec{U} = \{u_1, \ldots, u_n\}$, $\vec{V} = \{v_1, \ldots, v_n\}$, $\overrightarrow{U^F} = \{u_1^F, \ldots, u_f^F\}$, and $\overrightarrow{V^F} = \{v_1^F, \ldots, v_f^F\}$.

$$\cos\left(\vec{U}, \vec{V}\right) \leq \text{upper}\left(\overrightarrow{U^F}, \overrightarrow{V^F}\right). \tag{1}$$

The reason why the transformation F should satisfy (1) is that the SSDD using F should not incur any false dismissal, which is known as Parseval's theorem (the lower bound property of the Euclidean distances) in time-series matching [20]. To obtain an upper bound of the lower-dimensional transformation F, we first define an upper bound of F as follows.

Definition 1. If a lower-dimensional transformation F transforms n-dimensional vectors, $\vec{U}$ and $\vec{V}$, to f-dimensional vectors, $\overrightarrow{U^F}$ and $\overrightarrow{V^F}$, respectively, we define the *upper bound function* of F, denoted by $\text{upper}(\overrightarrow{U^F}, \overrightarrow{V^F})$, through

$$\text{upper}\left(\overrightarrow{U^F}, \overrightarrow{V^F}\right) = 1 - \frac{D^2\left(\overrightarrow{U^F}, \overrightarrow{V^F}\right)}{2}, \tag{2}$$

where $D^2(\overrightarrow{U^F}, \overrightarrow{V^F})$ is the squared Euclidean distance between $\overrightarrow{U^F}$ and $\overrightarrow{V^F}$; that is, $D^2(\overrightarrow{U^F}, \overrightarrow{V^F}) = \sum_{i=1}^{f} |u_i^F - v_i^F|^2$.

In this paper, we want to use the FS as a lower-dimensional transformation F, and thus, we formally prove that the upper bound function of the FS satisfies (1), the upper bound property of the cosine similarity.

Theorem 2. *If a feature selection FS transforms n-dimensional vectors, $\vec{U}$ and $\vec{V}$, to f-dimensional vectors, $\overrightarrow{U^{FS}}$ and $\overrightarrow{V^{FS}}$, respectively, $\text{upper}(\overrightarrow{U^{FS}}, \overrightarrow{V^{FS}})$ is an upper bound of $\cos(\vec{U}, \vec{V})$; that is, (3) holds.*

$$\cos\left(\vec{U}, \vec{V}\right) \leq \text{upper}\left(\overrightarrow{U^{FS}}, \overrightarrow{V^{FS}}\right). \tag{3}$$

Proof. Equation (3) is simply derived from (2), and we omit the detailed proof. □

Using the upper bound property of the FS, we now propose a generic two-step Protocol SSDD-FS. Algorithm 2 shows *Protocol* SSDD-FS. As shown in the protocol, SSDD-FS maintains f-dimensional $\overrightarrow{U^{FS}}$ and $\overrightarrow{V^{FS}}$ and n-dimensional $\vec{U}$ and $\vec{V}$ of SSDD-Base. In addition, Alice and Bob share $f \times f/2$ matrix $\mathbf{A}^{FS}$ and an $n \times n/2$ matrix $\mathbf{A}$ of SSDD-Base. Lines (1) through (7) of SSDD-FS are the first step of discarding nonsimilar n-dimensional vectors in the f-dimensional

Protocol SSDD-FS

(1) $\vec{U}$ and $\vec{V}$ are n-dimensional document vectors; $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$ are their f-dimensional feature vectors.
(2) Assume that Alice and Bob share the matrices $\mathbf{A}$ and $\mathbf{A}^{\mathrm{FS}}$ of sizes $n \times n/2$ and $f \times f/2$, respectively.
[1st step] Filtering step in the f-dimensional space
Alice:
(1) Execute Lines (1) to (3) of SSDD-Base for $\overrightarrow{U^{\mathrm{FS}}}$ and $\mathbf{A}^{\mathrm{FS}}$ instead of $\vec{U}$ and $\mathbf{A}$;
Bob:
(2) Execute Lines (4) to (6) of SSDD-Base for $\overrightarrow{V^{\mathrm{FS}}}$ and $\mathbf{A}^{\mathrm{FS}}$ instead of $\vec{V}$ and $\mathbf{A}$;
(3) Compute $\|\overrightarrow{V^{\mathrm{FS}}}\|$ and send it to Alice;
Alice:
(4) Execute Lines (7) and (8) of SSDD-Base; // $\delta = \overrightarrow{U^{\mathrm{FS}}} \cdot \overrightarrow{V^{\mathrm{FS}}}$
(5) Compute $\|\overrightarrow{U^{\mathrm{FS}}}\|$ and $\Delta = \|\overrightarrow{U^{\mathrm{FS}}}\| - 2\delta + \|\overrightarrow{V^{\mathrm{FS}}}\|$; // $\Delta = D^2(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}})$
(6) Compute $\upsilon = 1 - \Delta/2$; // $\upsilon = \mathrm{upper}(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}})$
(7) **if** $\upsilon < \varepsilon$ **then** Discard the pair $(\vec{U}, \vec{V})$ as a non-similar one; // υ = upper bound
[2nd step] Post-processing step in the n-dimensional space
Alice and Bob:
(8) Execute Lines (1) to (9) of SSDD-Base if $(\vec{U}, \vec{V})$ is not discarded in the 1st step;

Algorithm 2: Protocol of the generic two-step solution SSDD-FS.

space. First, Lines (1) through (4) securely compute the scalar product δ for f-dimensional vectors $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$. Except for using f-dimensional vectors instead of n-dimensional vectors, these steps are the same as those of SSDD-Base. The only difference from SSDD-Base is that Bob additionally sends $\|\overrightarrow{V^{\mathrm{FS}}}\|$ to Alice in Line (3) for computing $D^2(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}})$. In Line (5), Alice computes Δ (=$D^2(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}})$) using

$$D^2\left(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}}\right) = \left\|\overrightarrow{U^{\mathrm{FS}}}\right\| - 2 \cdot \overrightarrow{U^{\mathrm{FS}}} \cdot \overrightarrow{V^{\mathrm{FS}}} + \left\|\overrightarrow{V^{\mathrm{FS}}}\right\| = \left\|\overrightarrow{U^{\mathrm{FS}}}\right\| - 2\delta + \left\|\overrightarrow{V^{\mathrm{FS}}}\right\|. \quad (4)$$

Next, Alice computes an upper bound function of the FS, $\mathrm{upper}(\overrightarrow{U^{\mathrm{FS}}}, \overrightarrow{V^{\mathrm{FS}}})$, in Line (6), and in Line (7), we apply the filtering process by comparing the upper bound (υ) and the given tolerance (ϵ). If the upper bound is less than the tolerance, that is, if $\upsilon < \epsilon$, the actual cosine similarity will also be less than the tolerance, and we do not need to compute it in the next n-dimensional space. That is, if $\upsilon < \epsilon$, we can skip Line (8) of the second step. Thus, Line (8) is executed only if n-dimensional vectors of $(\vec{U}, \vec{V})$ are not filtered out by the upper bound. Finally, in Line (8), we compute the actual cosine similarity for $(\vec{U}, \vec{V})$ using SSDD-Base.

We note here how SSDD-FS improves the performance compared with SSDD-Base depending on how many n-dimensional vectors are discarded in the first step. This filtering effect depends largely on the discriminative power of the feature selection, that is, the efficiency of the FS. In other words, if the FS largely exploits the filtering effect, SSDD-FS can reduce the computation and communication overhead from $O(n|U||V|)$ to $O(f|U||V|)$. Based on this observation, we need to maximize the filtering effect of the FS, which can be seen as a problem of how we choose f dimensions from n dimensions for maximizing the discriminative power of the FS. Therefore, we propose efficient FS variants and their SSDD protocols in Section 4.

4. Discriminative Feature Selections for the Two-Step Protocol

In this section, we propose four methods to implement the FS of *Protocol* SSDD-FS. Figure 2 shows the procedure of SSDD-FS including the feature selection step. As shown in the figure, we first obtain $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$ from $\vec{U}$ and $\vec{V}$, respectively, through the feature selection, which should also be applied securely. As mentioned in Section 1, we presented the RP, LF, GF, and HF for the feature selection method, and in this section we describe how they work in detail. In Figure 2, the secure feature selection corresponds to Line (1) of *Protocol* SSDD-FS, and the other two steps correspond to the first (Lines (1) through (7)) and second (Line (8)) steps, respectively.

4.1. RP: Random Projection. An RP is the easiest way of implementing the FS, and it randomly selects f dimensions from n dimensions. We can consider two different methods for applying the RP to SSDD-FS. The first selects f dimensions dynamically for each document pair $(\vec{U}, \vec{V})$, and the second first determines f dimensions and then uses these predetermined dimensions for all document pairs.

To use the first RP method, Alice and Bob should share f indexes, $i_1, \ldots, i_f$ $(1 \le i_j \le n,\ j = 1, \ldots, f)$, of randomly selected f dimensions for each $(\vec{U}, \vec{V})$ before starting the first step of SSDD-FS. This sharing process can be implemented

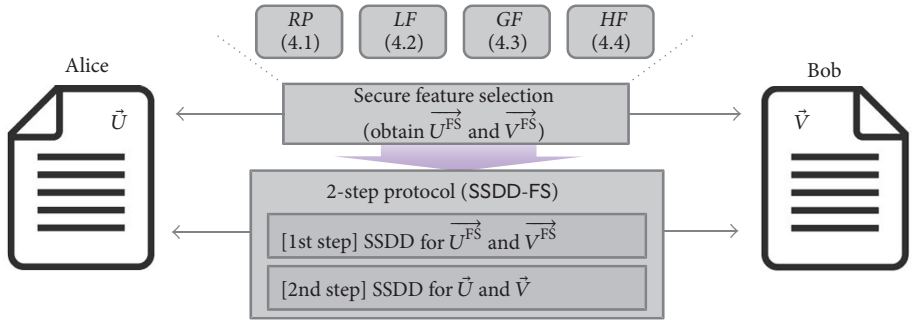


Figure 2: Feature selections in the SSDD-FS process.

(1-1) $\vec{U}$ and $\vec{V}$ are n-dimensional document vectors.
(1-2) Alice randomly chooses f dimensions, $i_1, \ldots, i_f$ $(1 \le i_j \le n)$, and sends them to Bob.
(1-3) $\overrightarrow{U^{FS}}$ and $\overrightarrow{V^{FS}}$ are f-dimensional feature vectors extracted from $\vec{U}$ and $\vec{V}$ by using those f indexes. // $u_j^{FS} = u_{i_j}$, $v_j^{FS} = v_{i_j}$, $j = 1, \ldots, f$.

Algorithm 3: Modification of Line (1) of SSDD-FS to implement SSDD-RP.

as Alice randomly selecting f dimensions and sending their indexes to Bob or as Alice and Bob sharing the same seed of a random function. That is, we can implement the first RP method by modifying Line (1) of *Protocol* SSDD-FS as Lines (1-1) through (1-3) of Algorithm 3.

The second RP method uses the same f dimensions for all $(\vec{U}, \vec{V})$ pairs. We can easily implement this method because Alice and Bob share the same f indexes only once before starting SSDD-FS. These first and second RP methods do not disclose any values of Alice's and Bob's document vectors and thus are considered secure. In addition, these two methods have the same effect in selecting f dimensions randomly. Thus, we use the second method, which we call SSDD-RP to differentiate it from SSDD-FS, because it is much simpler than the first.

4.2. LF: Local Frequency. SSDD-RP proposed in Section 4.1 has a problem of exploiting only a slight filtering effect during the first filtering step. This low filtering effect is due to the RP choosing features without any consideration of the characteristics of the document vectors. According to real experiments, SSDD-RP shows a very slight improvement in the SSDD performance compared with SSDD-Base. To solve the problem of SSDD-RP and enlarge the filtering effect, in this paper, we consider how frequent each term is in the document or document set; that is, we use the term frequency (TF) (in this paper, we use TF for simplicity, but we can also use TF-IDF (term frequency-inverse document frequency) instead of TF. Using which frequency among TF, TF-IDF, and other feature frequencies is orthogonal to our approach, and we use TF for easy understanding of the proposed concept). In general, we use the TF concept as follows: we first compute the number of occurrences (that is, the frequency) of each term throughout the whole dataset and then choose the highly frequent dimensions. We call this selection method document frequency (DF). We consider the TF (or DF) in SSDD-FS because if we select the highly frequent f dimensions, we can obtain relatively small upper bounds upper$(\overrightarrow{U^F}, \overrightarrow{V^F})$ by the relatively large $D^2(\overrightarrow{U^F}, \overrightarrow{V^F})$ of (2), and accordingly, we can largely exploit the filtering effect.

As a feature selection using term frequencies, we first consider how frequent each term is in an individual document rather than the whole document set; that is, we first propose the feature selection of exploiting the *locality* of each document. More precisely, for a pair of documents $(\vec{U}, \vec{V})$, the locality-based selection chooses f highly frequent dimensions in Alice's current vector $\vec{U}$. This selection is based on the simple intuition that, even without considering whole vectors of the document set, the current vector itself will have a significant influence on the upper bound upper$(\vec{U}, \vec{V})$. In this selection, we can instead use Bob's vector $\vec{V}$ rather than Alice's vector $\vec{U}$ as the current vector, or we can also use both Alice's and Bob's vectors $\vec{U}$ and $\vec{V}$. Using $\vec{V}$, however, incurs additional communication overhead, and thus, in this paper we consider a simple method of using Alice's $\vec{U}$ as the current vector. We call this selection method the local frequency (LF) because it considers individual (that is, local) documents rather than whole documents, and we denote the protocol of applying the LF to SSDD-FS as SSDD-LF. SSDD-LF exploits the locality by selecting f dimensions for each document at every start time. Algorithm 4 shows how we implement SSDD-LF by modifying Line (1) of SSDD-FS of Algorithm 2. In Line (1-2), Alice first selects the top-f frequent dimensions from her current vector. She sends those indexes of the selected f dimensions to Bob in Line (1-3). Thus, they can share the same indexes and obtain f-dimensional feature vectors using the same f indexes in Line (1-4).

(1-1) $\vec{U}$ and $\vec{V}$ are n-dimensional document vectors.
(1-2) Alice chooses f dimensions, $i_1, \ldots, i_f$ $(1 \le i_j \le n)$, whose TFs are larger than other dimensions.
(1-3) Alice sends those f indexes to Bob. // This can be done together with Line (3) of Algorithm 1
(1-4) $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$ are f-dimensional feature vectors extracted from $\vec{U}$ and $\vec{V}$ by using those f indexes. // $u_j^{\mathrm{FS}} = u_{i_j}$, $v_j^{\mathrm{FS}} = v_{i_j}$, $j = 1, \ldots, f$.

Algorithm 4: Modification of Line (1) of SSDD-FS to implement SSDD-LF.

We now analyze the computation and communication overhead of the feature selection in SSDD-LF. As shown in Algorithm 4, for each vector $\vec{U}$, Alice (1) chooses the top f frequent dimensions from n dimensions of $\vec{U}$, and (2) communicates with Bob to share those f indexes. First, Alice needs the additional computation overhead of $O(n \log f)$ to select the top f frequent dimensions from the current n-dimensional vector. Second, Alice and Bob need the additional communication overhead to share the f indexes. However, this communication process can be applied with Line (3) of SSDD-Base in Algorithm 1; that is, Alice can send f indexes together with the encrypted vector $\vec{Z}$ to Bob. The amount of f indexes is much smaller than that of the n-dimensional vector, and the overhead of f indexes can be negligible. Thus, we can say that SSDD-LF causes the computation overhead of $O(n \log f)$, but the communication overhead can be ignored. In particular, we compare each vector $\vec{U}$ of Alice with a large number of vectors $\vec{V}$ $(\in \mathbb{V})$ of Bob, and thus, the computation overhead of $O(n \log f)$ can also be ignored as a preprocessing step.

Another point to consider in SSDD-LF is whether its feature selection process is secure. That is, there should be no privacy disclosure when Alice selects f indexes and shares them with Bob. Fortunately, Alice sends only indexes i_j to Bob rather than entry values u_{i_j} of $\vec{U}$, and the sensitive values u_{i_j} are not disclosed during the selection process. Unfortunately, however, the information regarding which f dimensions are frequent in $\vec{U}$ is revealed to Bob. If a user cannot be allowed even this limited disclosure of information, we recommend using the previous SSDD-RP or the next SSDD-GF or SSDD-HF as a more secure protocol.

4.3. GF: Global Frequency. SSDD-LF described above has a problem in considering only Alice's current vector and ignoring all other vectors of Bob. Due to this problem, SSDD-LF exploits the filtering effect for only a part of Bob's vectors but does not do so for most of the other vectors. To overcome this problem, in this section, we propose another feature selection that uses the whole vector whose elements represent the number of documents containing the corresponding term. Unlike the LF focusing on the current vector only, it considers all document vectors and has the characteristics of globality. We call this feature selection the global frequency (GF) and denote the GF-based secure protocol as SSDD-GF. Actually, the GF is the same as the DF, which has been widely used as the representative feature selection, and it works as follows. First, let $\vec{A} = \{a_1, \ldots, a_n\}$ be a whole vector, where a_k is the number of documents containing the kth term; that is, a_k is the DF value of the kth term. Then, to reduce the number of dimensions from n to f, the GF simply selects f dimensions whose DF values are larger than those of the other $(n - f)$ dimensions. We can obtain the whole vector by scanning all of the document vectors a single time. The traditional DF constructs the whole vector based on the assumption that all document vectors are maintained in a single computer. In SSDD, however, the document vectors are distributed to Alice and Bob, each of whom does not want to provide their own vectors to the other. Thus, to use the GF in SSDD, we first need to present a secure protocol for constructing the whole vector from the document vectors stored in a distributive manner by Alice and Bob.

Algorithm 5 shows *Protocol* SecureDF that securely constructs a whole vector $\vec{A}$ from Alice's and Bob's document vectors and obtains f frequent dimensions from $\vec{A}$. In Lines (1) through (8), Alice and Bob compute their own whole vectors independently. That is, Alice computes her own whole vector $\overrightarrow{A^{\mathrm{Alice}}}$ from her own document set $\mathbb{U}$, and Bob obtains $\overrightarrow{A^{\mathrm{Bob}}}$ from $\mathbb{V}$. In Lines (4) and (8), they share those whole vectors $\overrightarrow{A^{\mathrm{Alice}}}$ and $\overrightarrow{A^{\mathrm{Bob}}}$ with each other. In Lines (9) through (11), they then compute the aggregated whole vector $\vec{A}$ from those vectors. After obtaining the whole vector $\vec{A}$, Alice and Bob can select f-frequent dimensions from $\vec{A}$. We note that Alice sends $\overrightarrow{A^{\mathrm{Alice}}}$ to Bob in Line (4), and Bob sends $\overrightarrow{A^{\mathrm{Bob}}}$ to Alice in Line (8). Vectors $\overrightarrow{A^{\mathrm{Alice}}}$ and $\overrightarrow{A^{\mathrm{Bob}}}$, however, are not the exact values of the document vectors but simple statistics, and thus we can say that SecureDF does not reveal any privacy of individual documents. The computation and communication complexities of SecureDF are merely $O(n|\mathbb{U}| + n|\mathbb{V}|)$ and $O(n)$, respectively. In addition, SecureDF can be seen as a preprocessing step that is executed only once for all document vectors of Alice and Bob. Thus, its complexity can be negligible compared with the complexity $O(n|\mathbb{U}||\mathbb{V}|)$ of SSDD-Base.

We now describe SSDD-GF, which exploits SecureDF as the feature selection. Algorithm 6 shows how we modify Line (1) of Algorithm 2 to convert SSDD-FS into SSDD-GF. In Line (1-0), we first apply SecureDF to obtain the whole vector $\vec{A}$ and determine f indexes that are most frequent found in $\vec{A}$. For the current n-dimensional vectors $\vec{U}$ and $\vec{V}$, Alice and Bob obtain f-dimensional vectors $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$

Protocol SecureDF
(1) **U** is a set of n-dimensional vector owned by Alice.
(2) **V** is a set of n-dimensional vectors owned by Bob.
Alice:
(1) **for each** dimension d **do**
(2) Compute a_k^{Alice} = the number of occurrences in **U**;
(3) **end-for**
(4) Send $\overrightarrow{A_k^{\text{Alice}}}$ to Bob;
Bob:
(5) **for each** dimension k **do**
(6) Compute a_k^{Bob} = the number of occurrences in **V**;
(7) **end-for**
(8) Send $\overrightarrow{A_k^{\text{Bob}}}$ to Alice;
Alice and Bob:
(9) **for each** dimension k **do**
(10) Compute $a_k = a_k^{\text{Alice}} + a_k^{\text{Bob}}$; // a_k = DF value of the kth dimension
(11) **end-for**
(12) Choose f dimensions, $i_1, \ldots, i_f$, whose DF values of $\vec{A}$ are larger than the other dimensions;

Algorithm 5: Secure protocol for constructing a whole vector.

(1-0) Obtain $\vec{A}$ through SecureDF and determine f dimensions, $i_1, \ldots, i_f$ ($1 \le i_j \le n$), from $\vec{A}$.
(1-1) $\vec{U}$ and $\vec{V}$ are n-dimensional document vectors.
(1-2) $\overrightarrow{U^{\text{FS}}}$ and $\overrightarrow{V^{\text{FS}}}$ are f-dimensional feature vectors extracted from $\vec{U}$ and $\vec{V}$ by using those f indexes. // $u_j^{\text{FS}} = u_{i_j}$, $v_j^{\text{FS}} = v_{i_j}$, $j = 1, \ldots, f$.

Algorithm 6: Modification of Line (1) of SSDD-FS to implement SSDD-GF.

using the determined f indexes. As shown in Algorithm 6, the current vectors and even their term frequencies are not disclosed to each other, and thus, we can say that SSDD-GF is a secure protocol of SSDD.

4.4. HF: Hybrid Frequency. The LF and GF have the following different characteristics from the viewpoint of a filtering effect. First, the LF considers Alice's current vector $\vec{U}$ only, and thus, the filtering effect will be large for only a portion of Bob's vectors whose TF patterns differ greatly from the current vector; however, this effect is exploited less for most of the other vectors. In other words, the LF can exploit a better filtering effect than the GF when Alice's current vector differs significantly from the whole vector in the TF patterns. Second, the GF considers the whole vector $\vec{A}$ without considering the current vector, and it can thus exploit the filtering effect relatively evenly on many of Bob's document vectors. That is, the GF can exploit a better filtering effect than the LF when Alice's current vector has similar characteristics to the whole vector in the TF patterns.

To take advantage of both the locality of LF and the globality of GF, we now propose a hybrid feature selection, called hybrid frequency (HF). That is, the HF uses the current vector for exploiting the locality of LF and at the same time also uses the whole vector for exploiting the globality of GF. We then present an advanced secure protocol, SSDD-HF, by applying the HF to the SSDD-FS. Simply speaking, the HF compares the current and whole vectors and selects the feature dimensions whose differences are larger than those of the other dimensions. In more detail, we select feature dimensions that have one of the following two characteristics: (1) dimensions that frequently occur in Alice's current vector but seldom occur in the whole vector (i.e., whose values are relatively large in the current vector but relatively small in the whole vector) or, on the contrary, (2) the dimensions that seldom occur in Alice's current vector but frequently occur in the whole vector. This is because the larger $|u_i^F - v_i^F|$ is (that is, the difference between the values of the selected feature dimension), the smaller upper$(\overrightarrow{U^F}, \overrightarrow{V^F})$ is or, in other words, the larger $D^2(\overrightarrow{U^F}, \overrightarrow{V^F})$ of (2) is, which exploits the larger filtering effect.

However, we cannot directly compare Alice's current vector $\vec{U}$ and the whole vector $\vec{A}$ using SecureDF. The reason is that $\vec{U}$ represents "frequencies of terms" in a single vector, whereas $\vec{A}$ represents the "frequencies of documents" containing those terms. That is, the meaning of the frequencies in $\vec{U}$ differs from that of $\vec{A}$, and thus, their scales are also different. To resolve this problem, before comparing the two vectors $\vec{U}$ and $\vec{A}$, we first normalize them using their mean (μ) and standard deviation (σ). More precisely, we first normalize

(1-0) Obtain $\vec{A}$ through SecureDF.
(1-1) $\vec{U}$ and $\vec{V}$ are n-dimensional document vectors.
(1-2) Alice normalizes $\vec{U}$ and $\vec{A}$ to $\overrightarrow{\overline{U}}$ and $\overrightarrow{\overline{A}}$ by using Eq. (5).
(1-3) Alice obtains the difference vector $\vec{D}$ from $\overrightarrow{\overline{U}}$ and $\overrightarrow{\overline{A}}$ by $d_i = |\overline{u}_i - \overline{a}_i|$.
(1-4) Alice chooses f dimensions, $i_1, \ldots, i_f$, whose d_i values are larger than the other dimensions.
(1-5) Alice sends those f indexes to Bob. // This can be done together with Line (3) of Algorithm 1
(1-6) $\overrightarrow{U^{\mathrm{FS}}}$ and $\overrightarrow{V^{\mathrm{FS}}}$ are f-dimensional feature vectors extracted from $\vec{U}$ and $\vec{V}$ by using those f indexes. // $u_j^{\mathrm{FS}} = u_{i_j}$, $v_j^{\mathrm{FS}} = v_{i_j}$, $j = 1, \ldots, f$.

Algorithm 7: Modification of Line (1) of SSDD-FS to implement SSDD-HF.

$\vec{U}$ and $\vec{A}$ as $\overrightarrow{\overline{U}}$ and $\overrightarrow{\overline{A}}$ by (5), and we next obtain the difference vector $\vec{D} = \{d_1 = |\overline{u_1} - \overline{a_1}|, \ldots, d_n = |\overline{u_n} - \overline{a_n}|\}$. We then select the largest f dimensions from $\vec{D}$ and use them as the features of SSDD-HF.

$$\overline{u_i} = \frac{u_i - \mu\left(\vec{U}\right)}{\sigma\left(\vec{U}\right)}, \qquad \overline{a_i} = \frac{a_i - \mu\left(\vec{A}\right)}{\sigma\left(\vec{A}\right)}. \tag{5}$$

Algorithm 7 shows how we modify Line (1) of SSDD-FS in Algorithm 2 to implement SSDD-HF. First, as in SSDD-GF, Line (1-0) constructs $\vec{A}$ by executing SecureDF. Next, in Lines (1-2) and (1-3), we normalize the current and whole vectors and obtain the difference vector $\vec{D}$ from those normalized vectors. Finally, in Lines (1-4) through (1-6), Alice chooses f dimensions from $\vec{D}$ and shares those dimensions with Bob. That is, Lines (1-4) through (1-6) are the same as Lines (1-2) through (1-4) of SSDD-LF in Algorithm 4 except that SSDD-LF uses the current vector $\vec{U}$ whereas SSDD-HF uses the difference vector $\vec{D}$.

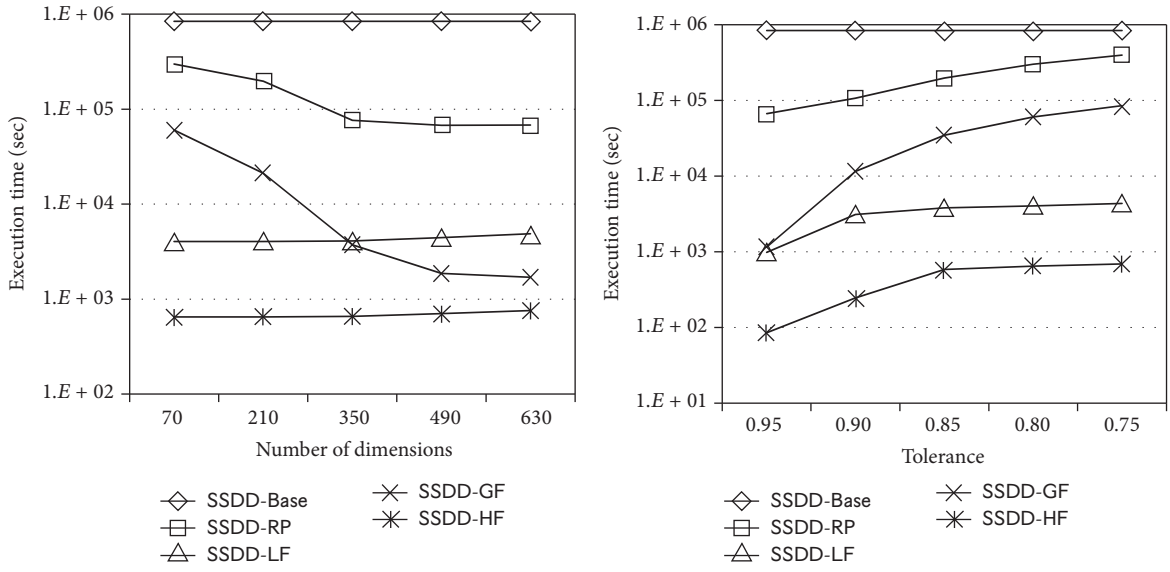
The overhead of the feature selection in SSDD-HF can be seen as a summation of those in SSDD-LF and SSDD-GF. That is, like SSDD-GF, it has the overhead of applying SecureDF to obtain the whole vector $\vec{A}$, and at the same time, like SSDD-LF, it has the overhead of choosing the largest f dimensions from the n-dimensional difference vector $\vec{D}$. Such overhead, however, can be negligible for the following reasons: (1) as we described with SSDD-GF earlier, SecureDF having computation and communication complexities of $O(n|\mathbb{U}| + n|\mathbb{V}|)$ and $O(n)$ can be seen as a preprocessing step executed only once for all document vectors, and its overhead can be negligible throughout the whole SSDD process; (2) as we described for SSDD-LF earlier, the computation complexity $O(n \log f)$ of choosing f dimensions from an n-dimensional vector can be ignored because it can also be seen as a preprocessing step. One more notable point is that SSDD-HF is a secure protocol like SSDD-GF because it uses SecureDF and the difference vector, which are secure and do not disclose any original values or sensitive indexes of individual vectors.

5. Performance Evaluation

5.1. Experimental Data and Environment. In this section, we empirically evaluate the feature selection-based SSDD protocols. We used three datasets obtained from the document sets of the UCI repository [26]. The first dataset consists of KOS blog entries collected from dailykos.com (*KOS* in short). KOS consists of 3,430 documents with 6,906 different terms (dimensions) and has a total 467,714 terms. The second dataset contains full papers published for the Neural Information Processing Systems Conference (*NIPS* in short). NIPS consists of 1,500 documents with 12,419 different terms and has about 1.9 million terms in total. The third dataset contains email messages of Enron (*EMAILS* in short). EMAILS consists of 39,861 emails with 28,102 different terms, and has about 6.4 million terms in total.

We experimented using five SSDD protocols: SSDD-Base as the basic one and the four proposed protocols, SSDD-RP, SSDD-LF, SSDD-GF, and SSDD-HF. In the experiment, we basically measured the elapsed time of executing SSDD for each protocol. In the first experiment, we varied the number of dimensions for a fixed tolerance, where the number of dimensions is f, that is, the number of features *selected* by the feature selection. In the second experiment, we varied the tolerance for a fixed number of dimensions. For these two experiments, we used KOS and NIPS, which have relatively small numbers of documents compared with EMAILS. On the other hand, the third experiment was conducted to test the scalability of each protocol, and thus, we used EMAILS because it is much larger than those of KOS and NIPS.

The hardware platform is an HP ProLiant ML110 G7 workstation equipped with an E31220 3.10 GHz Intel Xeon Quad Core CPU, with 16 GB of RAM and a 250 GB HDD. CentOS 6.5 Linux was used as the software platform. We also used C language for implementing all of the protocols. We applied SSDD in a single machine using a local loop for network communication. The reason why we use the local loop is because we want to intentionally ignore the network speed because different network speeds or environments may largely distort the actual execution time of each protocol. We measured the execution time spent for Alice to sends each document to Bob and securely identify its similarity. More precisely, we stored the whole dataset in Bob and selected



(a) Different numbers of dimensions (tolerance = 0.80)

(b) Different tolerances (number of dimensions = 70)

Figure 3: Experimental results for KOS.

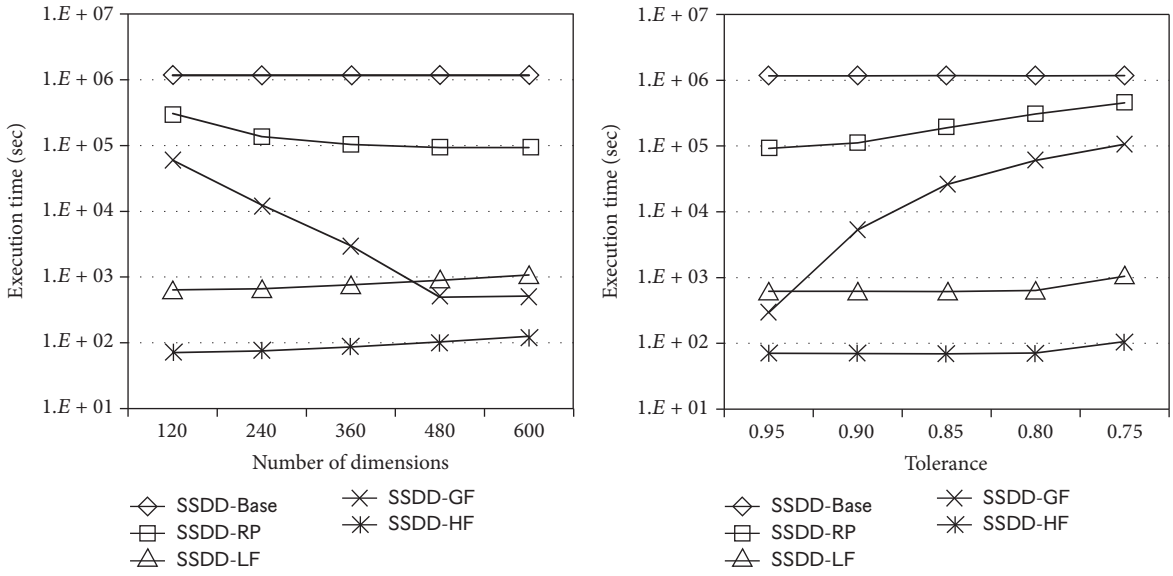
ten query documents for Alice. We then executed each SSDD protocol for these ten query documents and used their sum as the experimental result.

5.2. Experimental Results. Figure 3 shows the experimental results for KOS. First, in Figure 3(a), we set the tolerance to 0.80 and varied the number of documents by 70, 210, 350, 490, and 640, which correspond to 1%, 3%, 5%, 7%, and 9% of the number of KOS documents. In the figure, the x-axis shows the number of (selected) dimensions, and the y-axis shows the actual execution time. Note that the y-axis is at the log scale.

Figure 3(a) shows that all of the proposed protocols significantly outperform the basic SSDD-Base. Even SSDD-RP for selecting features randomly beats SSDD-Base by exploiting the filtering effect in the first step of the two-step protocol. Next, SSDD-GF shows a better performance than SSDD-RP because it selects the features frequently occurring throughout the whole dataset using DF. In the case of SSDD-RP and SSDD-GF, we note that as the number of dimensions increases, the execution time decreases. This is because the larger the number of dimensions we use, the larger the filtering effect we can exploit. SSDD-LF for using the locality of the current vector also outperforms both SSDD-RP and SSDD-Base. Note that SSDD-LF is better than SSDD-GF in dimension 350 (= about 5% of the number of documents) and worse than SSDD-GF in dimension 350 and beyond. This changing point of dimension 350 is reported by the actual experiment. This is because only a small number of dimensions have a significant influence on the locality of the current vector. Finally, SSDD-HF used for taking advantage of both SSDD-LF and SSDD-GF shows the best performance for all dimensions. In Figure 3(a), we note that the execution times of SSDD-LF and SSDD-HF slightly increase as the number of dimensions increases. The reason for this is that as the number of dimensions f increases, the filtering effect increases relatively slowly, but the overhead in obtaining a current/difference vector and choosing f dimensions from that vector increases relatively quickly.

Second, in Figure 3(b), we set the number of dimensions to 70 (1% of the total dimensions) and vary the tolerance from 0.95 to 0.75 by decreasing it by 0.05. Note that the closer the tolerance is to 1.0, the stronger the similarity we use. As shown in Figure 3(b), all of the proposed protocols significantly improve the performance compared with SSDD-Base. In particular, SSDD-LF and SSDD-HF, which exploit the locality, show a better performance than the other two proposed protocols. We note here that as the tolerance decreases, the execution times of all proposed protocols gradually increase. This is because the smaller the tolerance we use, the more similar documents we obtain. In the summary of Figure 3, the proposed SSDD-LF and SSDD-HF significantly outperform SSDD-Base by up to 726.6 and 9,858 times, respectively.

Figure 4 shows the experimental results for NIPS. Like Figure 3 of KOS, we measure the execution time of SSDD by varying the number of dimensions and the tolerance. In Figure 4(a), we set the tolerance to 0.80 and increase the number of dimensions from 120 (1%) to 600 (5%) by steps of 120 (1%), where 120 indicates 1% of a total of 12,419 documents. Next, in Figure 4(b), we set the number of dimensions to 120 and decrease the tolerance from 0.95 to 0.75 by steps of 0.05. The experimental results in Figures 4(a) and 4(b) show a very similar trend with those in Figures 3(a) and 3(b). That is, all of the proposed protocols significantly outperform SSDD-Base, and SSDD-HF shows the best performance. In Figure 4, SSDD-HF significantly improves the performance compared with SSDD-Base by up to 16,620 times.



(a) Different numbers of dimensions (tolerance = 0.80)

(b) Different tolerances (number of dimensions = 120)

Figure 4: Experimental results for NIPS.

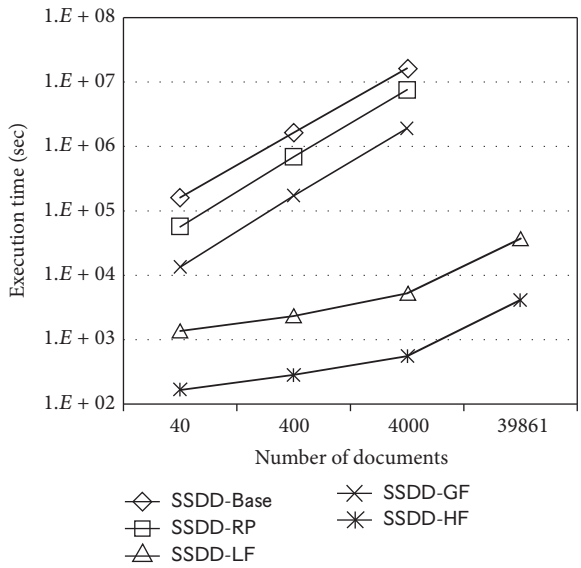


Figure 5: Experimental results of the scalability test for EMAILS.

Figure 5 shows the results for the scalability test using the large volume high-dimensional dataset, EMAILS. We set the tolerance and number of dimensions to 0.80 and 70, respectively, and increase the number of documents (emails) from 40 (0.1%) to 39,861 (100%) by tenfold. In this experiment, we exclude the results of SSDD-Base, SSDD-RP, and SSDD-GF for the case of 39,861 documents owing to the excessive execution time. As shown in Figure 5, as with the results of KOS and NIPS, our feature selection based protocols outperform SSDD-Base in all cases, and in particular, SSDD-LF and SSDD-HF show the best performance regardless of the number of documents. We also note that all of the proposed protocols show a pseudo-linear trend on the number of documents (note that the x- and y-axes are all at the log scales). That is, the protocols are pseudo-linear solutions on the number of documents, and we can state that they are excellent in terms of both scalability and performance.

6. Conclusions

In this paper, we addressed an efficient method for significantly reducing the computation and communication overhead in the area of secure similar document detection. The contributions of the paper can be summarized as follows. First, we thoroughly analyzed the previous one-step protocol and pointed out that it incurs a serious performance overhead for high-dimensional document vectors. Second, to alleviate the overhead, we presented the feature selection-based two-step protocol and formally proved its correctness. Third, to improve the filtering efficiency of the two-step protocol, we proposed four feature selections: (1) the RP for selecting features randomly, (2) the LF for exploiting the locality of a current vector, (3) the GF for exploiting the globality of all document vectors, and (4) the HF for considering both the locality and globality. Fourth, for each feature selection, we presented a formal protocol and analyzed its secureness and complexity. Fifth, through experiments on three real datasets, we showed that all of the proposed protocols significantly outperformed the base protocol, and in particular, the HF-based secure protocol improved the performance by up to three to four orders of magnitude. As future work, we will consider two issues: (1) the use of feature extraction (feature creation) instead of feature selection for a dimensionality reduction and (2) the use of homomorphic encryption rather than a random matrix for a secure scalar product.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (no. R7117-16-0214, Development of an Intelligent Sampling and Filtering Techniques for Purifying Data Streams).

References

[1] D. Sorokina, J. Gehrke, S. Warner, and P. Ginsparg, "Plagiarism detection in arXiv," in *Proceedings of the 6th IEEE International Conference on Data Mining*, pp. 1070–1075, Hong Kong, December 2006.

[2] W. Jiang, M. Murugesan, C. Clifton, and L. Si, "Similar document detection with limited information disclosure," in *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE '08)*, pp. 735–743, Cancun, Mexico, April 2008.

[3] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and semantic issues of text mining," *SIGMOD Record*, vol. 36, no. 3, pp. 23–34, 2007.

[4] C. C. Aggarwal and P. S. Yu, "Privacy-preserving data mining: a survey," *Handbook of Database Security: Applications and Trends*, pp. 431–460, 2008.

[5] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 29, no. 2, pp. 439–450, 2000.

[6] P. R. Bhaladhare and D. C. Jinwala, "Novel approaches for privacy preserving data mining in k-anonymity model," *Journal of Information Science and Engineering*, vol. 32, no. 1, pp. 63–78, 2016.

[7] S. Buyrukbilen and S. Bakiras, "Secure similar document detection with simhash," in *Proceedings of the Workshop on VLDB-Secure Data Management (SDM '13)*, pp. 61–75, Trento, Italy, August 2013.

[8] Y. Peng, G. Kou, Y. Shi, and Z. Chen, "Privacy-preserving data mining for medical data: application of data partition methods," in *Communications and Discoveries from Multidisciplinary Data*, vol. 123 of *Studies in Computational Intelligence*, pp. 331–340, Springer, Berlin, Germany, 2008.

[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[10] B. Pinkas, "Cryptography techniques for privacy-preserving data mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 12–19, 2002.

[11] E. Bingam and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 245–250, ACM SIGKDD, San Francisco, Calif, USA, August 2001.

[12] D. Cai, X. He, and J. Han, "SRDA: an efficient algorithm for large-scale discriminant analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 1–12, 2008.

[13] E. Bertino, D. Lin, and W. Jiang, "A survey of quantification of privacy preserving data mining algorithms," in *Privacy-Preserving Data Mining*, C. C. Aggarwal and P. S. Yu, Eds., vol. 34, pp. 183–205, Kluwer Academic, Norwell, Mass, USA, 2008.

[14] Y.-S. Moon, H.-S. Kim, S.-P. Kim, and E. Bertino, "Publishing time-series data under preservation of privacy and distance orders," in *Proceedings of the 21st International Conference on Database and Expert Systems Applications, Part II*, pp. 17–31, Publishing Time-Series Data Under Preservation of Privacy and Distance Orders, Bilbao, Spain, August 2010.

[15] H.-S. Won, S.-P. Kim, S. Lee, M.-J. Choi, and Y.-S. Moon, "Secure principal component analysis in multiple distributed nodes," *Security and Communication Networks*, vol. 9, no. 14, pp. 2348–2358, 2016.

[16] M. Shah and H. D. Joshi, "Privacy preserving data mining techniques in a distributed environment," *International Journal of Computer Applications*, vol. 94, no. 6, pp. 21–27, 2014.

[17] W. Jiang and B. K. Samanthula, "N-gram based secure similar document detection," in *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 239–246, Richmond, Va, USA, July 2011.

[18] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On secure scalar product computation for privacy-preserving data mining," in *Proceedings of the 7th Annual International Conference in Information Security & Cryptology*, pp. 104–120, Seoul, Republic of Korea, December 2004.

[19] S. Berchtold, C. Böhm, and H. Kriegal, "The pyramid-technique," *ACM SIGMOD Record*, vol. 27, no. 2, pp. 142–153, 1998.

[20] Y.-S. Moon, B.-S. Kim, M. S. Kim, and K.-Y. Whang, "Scaling-invariant boundary image matching using time-series matching techniques," *Data and Knowledge Engineering*, vol. 69, no. 10, pp. 1022–1042, 2010.

[21] Y.-S. Moon and W.-K. Loh, "Triangular inequality-based rotation-invariant boundary image matching for smart devices," *Multimedia Systems*, vol. 21, no. 1, pp. 15–28, 2014.

[22] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[23] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning (ICML '97)*, pp. 412–420, Nashville, Tenn, USA, July 1997.

[24] Y.-S. Moon and B. S. Lee, "Safe MBR-transformation in similar sequence matching," *Information Sciences*, vol. 270, pp. 28–40, 2014.

[25] W. Han, J. Lee, Y. Moon, S. Hwang, and H. Yu, "A new approach for processing ranked subsequence matching based on ranked union," in *Proceedings of the ACM SIGMOD International Conference on Management of data (SIGMOD '11)*, pp. 457–468, Athens, Greece, June 2011.

[26] Bag of Words Data Sets, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Bag+of+Words.