Available online at: http://brsj.cepsbasra.edu.iq

ISSN -1817 -2695

SECURE SIMILAR DETECTION FOR DOCUMENTS

[1] DUAA FADHEL NAJIM, [2]AYAD IBRAHIM ABDULSADA

Department of, Education Computer Science College for Pure Science, University of Basrah

E-mail: [1]duaa.najim@gmail.com , [2]ayad.abdulsada@uobasrah.edu.iq

ABSTRACT

Data similarity detection is an important for many applications such as document file management, document searching, plagiarism prevention, copyright protection. Most of the researches does not rely the preservation of privacy in detecting the similarity for documents because it is considered that the content of the document is general, which is reduced the use in certain applications that require the preservation of data when the detect the similarity, e.g., the conference submissions are treated as confidential and not revealing them to other program (in the process of similar document detection). Over the past few years, cryptologists have created protocols that preserve the privacy and protection of data while detecting similarities but it remains not secure. In this paper, we evaluate the similarity between two parties (Alice and Bob) without knowing any information about the content of their files, with maintaining efficiency and repairing the security problems of previous works. The cosine similarity is used to measure the similarity between the vectors of the document. Our proposal was applied on a real dataset through several experiments conducted to demonstrate the value and efficiency of proposed schemes in practice.

Keywords: *Document Similarity, Privacy, Homomorphic Encryption, Paillier Cryptosystem, Secure Multiparty Computing (SMC).*

1. INTRODUCTION

Since researcher presented in [1] the *document similarity detection* (DSD) have received wide attention. Like a model is frequently used in real-world applications. Specifically, file system management, in the context of system security and Web crawlers (detecting similar pages) [2]. Finally, among the most important applications that use this model are copyright protection and plagiarism detection. While document similarity increasingly employed as a good mechanism for detecting similar files, it is still undesirable for applications of sensitive data that must be subject to minimal exposure. This leads to develop new secure computation schemes over sensitive data where no information is

revealed to the participant parties except the outcome of the comparison.

For example, various health centers want to share reports on patients to verify similarities between reports to understand diseases well, but without violating the privacy of individuals. Furthermore, for academic conferences, its policy does not allow the double submission for the article itself. Therefore, it is necessary to use the secure similarity detection to check whether the article was submitted to more than one conference at the same time without revealing its content [3]. In fact, the best solution to protect data privacy is encryption. But it is considered a difficult task in traditional DSD methods. Therefore, an efficient method is required to measure similarity in the encrypted domain between two documents.

Most DSD models work by specifying distance assess and the text document representation In order to reduce the volume of data stored and reduce the cost of time when searching in the database. The stored data represent by either a vector (vector space model [4, 5]) or set substrings (N-gram model [6, 7]).

In our paper we adopt Vector space approach. It uses the concepts of information retrieval's (IR) [8] to detect the global similarity. In this approach represented each document as a vector of terms and each entry in that vector has number of occurrences for each term. Under this method, two documents are considered similar if they have common keywords with similar frequencies. Thus, two documents are referred to as similar with the same bag of words even if they are of different content. Cosine similarity metric is used to measure similarity between document term vectors.

We focused on the *syntactic similarity* notion to detect documents similarity. According to this notion, two documents consider to be similar if they contain similar keywords even if they have different meanings. On the other hand, semantic similarity notion uses intricate methods to consider meaning of the text during its matching process. The last notion is outside the extent of this paper.

Our work is very similar to information retrieval that preserves confidentiality [9, 10, 11, 12], but in fact there are many differences. Firstly, our work returns the most similar document, while the other scheme retrieves a set of documents ranked according to their similarity. Secondly, in our scheme, there are two parties; each party has documents that are supposed to be confidential. As for the other scheme, one party owns a set of documents and the other party provides the ability of storage computing (e.g. cloud server).Thirdly, our scheme is used for data that need high security (sensitive data).

Contributions in this work is to protect the scores of similarity that are revealed in previous works [4, 5], our scheme is of a general framework that is not specific, very effective and this it has been achieved through scientific experiments on a real database .

2. RELATED WORKS

The problem of the secure detection of documents was first proposed by the authors Jiang et al. [4]. The work in their studies was on a vector space model, each document represents a vector of normalizes term frequencies. To measure the similarity they used the cosine similarity. In their studies, two sides Alice and Bob agree together to identify the similarity of documents from their private collections.

The two sides work on pre-computes the term vector and then normalize vectors.

Secure dot product protocol is used to perform the cosine similarity. For Alice, to identify the closeness of a single document against Bob's collection, she will encrypt its elements with the public key and send it to Bob.

Bob calculates the similarity scores against all his vectors by a secure inner product and then sends the all encrypted similarity scores to Alice. Then, Alice decrypts the similarity scores and finds the similar documents.

The scheme [4] later improved its efficacy in [5] by used clustering algorithm (k-mean) to reduce the compared documents. The problem in [4] and [5] is detecting the all set of similarity scores for Alice. This leak can be used to infer the all collection of Bob. Our scheme solves this problem, where it is returning the most similar score only to the query party (Alice).

N-gram model is used in [6] to achieve secure DSD protocol. N-gram model has several characteristics. It is used to find the local similarity, easy, language independent, and less sensitive to changes to a document. The similarity under this model between two documents is commonly computed by Jaccard similarly (JS) [13]. When A and B are two documents, JS calculated as follows:

$$JS(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{(|A|+|B|-|A \cap B|)} \qquad (1)$$

The idea of [6] is illustrated as follows: firstly, each one of the two parties Alice and Bob calculates for each document the N-gram (used 3-gram) sets. Then Bob detect the all global 3-gram set to Alice. Alice and Bob use this global set to create a binary vector for each document. In the binary vector the entry is set to 1 if the corresponding 3-grams is be in the document; otherwise it is set to 0. Figure 1 shows the global 3-grams and binary vectors.

Global n-gram space and its mapping to integer domain

S	*aaa*	*aab*	*abb*	*abx*	*bbc*	*bcd*	*bxy*	*caa*	*cdd*	*ddf*	*xaa*	*xxx*	*xxy*	*xyy*	*yxy*	*yyx*	*yyy*	*yyz*
$M(S)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Binary vector representation of each document in the global n-gram space

u	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
v_1	1	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0
v_2	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	1
v_3	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Figure 1: Binary Vectors from [6]

Calculating JS between two private documents, Alice sends her encrypted binary vector to Bob. Bob and Alice running a secure dot product to calculate the common 3-grams, and then to compute JS they run a secure division protocol. However, this protocol is not secure since Bob detects all his 3-gram set to Alice. She can use this information to achieve whether Bob's collection involve a given word or not.

Lately, the authors of [7] proposed a scheme to find an efficient solution for the problem of privacy reserving evaluation of set similarity; the similarity is measured by the Jaccard similarity. Specifically, the authors proposed two schemes that use the private set intersection cardinality (PSI-CA) protocol [14] to identify the common 3-gram sets. The first one calculates the exact secure Jaccard similarity and the other one uses Min-hash technique to secure approximate it, which leads to a significant improvement in terms of communication and computation overheads.

The work in [15] employs sim-hash method, which is essentially used as a dimensional reduction tool [16], for generating small size document fingerprints by encoding all document terms and their frequencies into a fixed length binary vectors, and reducing the problem of document matching into to a

secure XOR computation between those small vectors [7].

Also MinHash technique is used in the [17] to design an advanced and efficient scheme for the secure document detection, Where the recent study of [13] is considered an improvement to the Jaccard similarity computation because it takes into account the calculation of the frequency of each element (N-gram).

3. PROBLEM AND SECURITY DEFINITION

3.1 Problem Definition

We have two parties. The first party is Alice. She has a document U that you want to measure the similarity with the collection of documents for Bob D, However, on the condition that each one's privacy is preserved, that is, the most similar document of the first party will be disclosed in strict confidence.

Alice and Bob want to find whether there are documents in Bob's collection that are similar to Alice's document (e.g., duplicate, near duplicate, somewhat close, etc.) without disclosing Alice's document to Bob and vice-versa. We compute document similarity under vector space model where each document is represented as vector of normalized term frequencies instead of comparing documents term by term. Let $D = \{D_1, D_2, \dots, D_m\}$ refers the set of m files in Bob's collection. Without compromising privacy of each party, our goal is to find the document in D that is most similar to U. We refer to our protocol as privacy preserving document similarity detection ($PPDSD$), which is formally defined as:

$$PPDSD(U, D) \rightarrow Ind$$

Where Ind is the returned index of the most similar document. Our scheme can be more general to meet the situation of threshold-based matching. In such a case, Alice can provide Bob with a secret threshold and asks him to return all indexes of documents that have similarity scores higher than the given threshold value.

A trivial solution to achieve $PPDSD$ is to utilize a trusted third party (TTP). Alice sends her document U to the TTP and Bob sends D to the TTP, and then TTP can investigate and inform Alice whether there are documents are similar to her query. However, in real life situations, finding a completely trustworthy third party is a difficult task. Our scheme achieves its goals without using any third party. Our proposed scheme is illustrated in Figure 2.

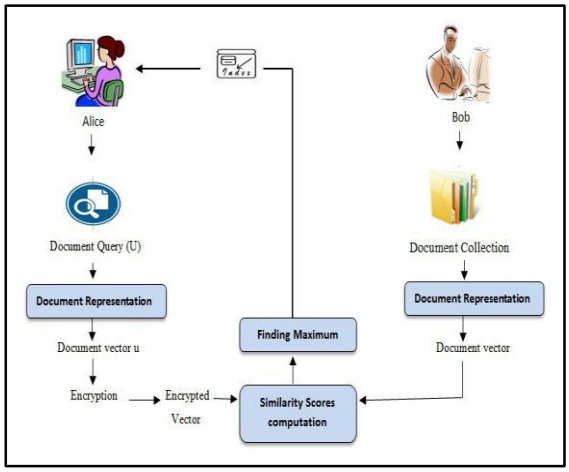

Figure 2: The Proposed Scheme Outline

3.2 Security Requirements

Our security definition follows the secure two computing definition of Goldreich et al. [18]. We use the standard security model, which assumes semi-honest (honest-but-curios) parties, where each one follows the scheme steps exactly. However, they may collect as much information as possible by looking at the received messages during the execution of the scheme to deduce private information. This model seems to be adopted widely by the community, since it offers an

acceptable level of security within reasonable computational and communication costs. The scheme is considered secure if whatever can be computed by a party participating in the protocol can be computed based on its input and output only. In other words, parties learn nothing from the execution of the scheme itself. Formally, let party's view represent what it sees during the execution of the scheme. If the view of each party is simulated by its input and output, and the real view is computationally indistinguishable from the simulated view, then the proposed scheme is said to be secure under the semi-hones model.

4. CRYPTOGRAPHIC PRELIMINARIES

In this section, we provide a high level description of the main cryptographic primitives that are used in this paper.

4.1 Homomorphic Encryption

Our approach strongly utilizes additively homomorphic public key encryption schemes. Given integers a and b, additive homomorphic encryption has the following property: given the private and public key pair (pr, pk),
$Dec_{pr}\,(Enc_{pk}\,(a).Enc_{pk}\,\,(b)) = a + b,$
where $Enc_{pk}\,()$ and $Dec_{pr}\,()$ are the encryption, decryption algorithms, respectively. As a consequence of additive homomorphism, any cipher text $Enc_{pk}\,(a\,)$ raised to the power of c results in an encryption of $c.m$, i.e. $Enc_{pk}\,(a\,)^c = a\,.c$ for any a in the message space, and positive integer c. The subtraction of two plain text integers can also be realized directly as follows: $Dec_{pr}\left(Enc_{pk}\,(a).Enc_{pk}\,(b)^{-1}\right) = a - b,$ Where $.Enc_{pk}\,(b)^{-1}$ denotes the multiplicative inverse of $Enc_{pk}\,(b)$. In particular, we adopt two additively homomorphic encryption semantically secure schemes: Paillier scheme [19] and the DGK scheme [20].

4.2 Paillier Cryptosystem

It is a popular public key cryptosystem with a cipher text modulus N^2, where N is RSA modules of k-bits. Where RSA is the cryptography algorithm (Rivest Shamir Adleman) with k is always in the range (1000-2048). N is produced by the product of two large prime numbers p and q.

The encryption of a message $x \in \mathbb{Z}_N$ is indicated as $[\![x]\!]$ and is computed by $[\![x]\!] = g^x.r^N \; mod \; N^2$, where $r \in Z_N$ is a random number, g is an element that order is multiple of N in $\mathbb{Z}^*_{N^2}$. For getting significant performance for encryption, we use $g = N + 1$ because it simplify g^x to $1 + xNmod\,N^2$ which saves one exponentiation. The resulting ciphertext can be "re randomized" using a fresh random value $r \in Z_N$ as $[\![x]\!].\mathrm{r}^{\mathrm{N}}$. The public key of Paillier cryptosystem consists of two elements $pk = \{N, g\}$; the secret key is the pair $pr = \{p, q\}$. It is easy to view that Paillier is additively homomorphic: given $[\![a]\!]$ and $[\![b]\!]$ we have that $[\![a + b]\!] = [\![a]\!].[\![b]\!] \; mod \; N^2$. The Paillier cryptosystem is efficient enough to be utilizing in practice. However, the cipher text is twice as long as the original plaintext.

4.3 Damg˚ard, Geisler and Krøigaard cryptosystem (DGK)

DGK cryptosystem was designed to deal with small message spaces and produce shorter ciphertext size than other probabilistic encryption schemes. It is used efficiently to compare two encrypted integers. The public key is defined as (N, g, h, u) and the private key is defined as (p, q, v_p, v_q), where N is the ciphertext modulus of k-bits, produced by multiplying two large primes p and q of $k/2$-bits, u defines the plaintext space $\mathbb{Z}_u$, it is a small prime divisor of both $p - 1$ and $q - 1$. Virtually, u have values that

very small, typically it is chosen as the minimal prime greater than ℓ +2. The additional parameters v_p and v_q are t-bit prime and $uv_p|(p-1)$ and $uv_q|(q-1)$. Parameters k,t and ℓ should satisfy $k > t > l$. Typically, these values might be k=1024, t=160, and ℓ ranges from 8 to 32 bits. The elements g, $h \in \mathbb{Z}_N^*$ such that g has order of uv_pv_q and h has order of v_pv_q Given a message $x \in \mathbb{Z}_u$, its DGK encryption is denoted by $[x]$ and it is calculated as $[x] = g^x.h^{\,r} mod\ N$, where r is a randomly selected integer of $2.5t$-bits. DGK is also additively homomorphic so $[x].[y] = [x+y]\ mod\ N$. The pretty feature of DGK is that to define whether x equal zero it is enough to check $x^{\,v_p\,v_q} mod\ N = 1$. Moreover, since $u < p$ it is even sufficient to check $x^{\,v_p\,v_q} mod\ p = 1$.

Secure comparison protocol for encrypted integers

Veugen [21] proposed an efficient protocol to compare two encrypted integers $x, y, 0 \le x, y < 2^\ell$. Such a protocol in turn uses as building block a resolution originally given by Damg˚ard et al [20] to compare two private integers. The idea behind this solution is to compute $z = 2^\ell - y + x$ (computations are done over encrypted data) of $(\ell+1)$ - bit value and check the most important bit z_ℓ of z. If it is 1, it means that $x \ge y$, else $x < y$. So, in order to determine if $x \ge y$ it suffice to calculate the bit z_ℓ. The bit z_ℓ can be obtained as the outcome of the division of z by 2ℓ. The protocol can be described briefly as follows: Bob has got the encryption of two numbers $[x]$ and $[y]$, and he is interested in learning the outcome comparison bit $\delta = (x < y) = 1 - z_\ell$. The security of the protocol requires that Alice should not learn the original values x, y, and the result of comparison should not be exposed to Bob. The protocol of [21] works as follows:

Bob blinds the value of z by adding it to a random value r, obtaining d. Bob sends d to Alice, and then they run jointly DGK-comparison protocol (described in the next section) so that Bob learns the comparison bit δ' between the private values α, β, where $\alpha = r\ mod\ 2^\ell$ and $\beta = d\ mod\ 2^\ell$. The value of δ' is useful to compute the bit δ as follows:

$$\delta = 1 - \left\lfloor \frac{z}{2^\ell} \right\rfloor = 1 - \left\lfloor \frac{d}{2^\ell} \right\rfloor + \left\lfloor \frac{r}{2^\ell} \right\rfloor + \delta' \qquad (2)$$

In our work, we use the public key Paillier cryptosystem to encrypt the inputs $[\![x]\!]$, $[\![y]\!]$ as well as the output result. We modify the protocol [20] in that we let Alice to learn δ. Protocol 1 shows to our modified version which compares two encrypted integers $[\![x]\!]$, $[\![y]\!]$ and return the comparison bit $(x < y)$ to Alice.

Protocol 1: Secure comparison of encrypted integers

Input: Alice: Paillier cryptosystem key pairs *(pr, pk)*, DGK cryptosystem key pairs *(gr, gk)*, Bob: $[\![x]\!]$, $[\![y]\!]$: two encrypted integers, *pk, gk.*

Output: Alice: comparison bit $\delta = (x < y)$, Bob: $[\![\delta]\!]$.

Step 1: Alice and Bob reduce the problem from comparing two encrypted inputs $[\![x]\!]$ and $[\![y]\!]$ to comparing two private inputs.

Bob chooses a random r of $\ell + 2 + \sigma$ bits (where σ is a security parameter, and $\ell + 2 + \sigma < \log_2(N)$) and computes $[\![d]\!] \leftarrow [\![x - y + 2^\ell + r]\!] = [\![x]\!].[\![y]\!]^{-1}.[\![2^\ell + r]\!] \bmod N^2$ and sends $[\![d]\!]$ to Alice.

Bob calculates $\alpha = r \bmod 2^\ell$

Alice obtains d after decryption.

Alice computes $\beta = d \bmod 2^\ell$.

Step 2: Alice and Bob utilize DGK-comparison protocol (see next section) to securely compare their two

unencrypted ℓ-bit values α and β, to compute the encrypted bit [δ′] such that $\delta' = (\beta < \alpha)$

Step 3: Alice and Bob computes the final comparison bit δ as follows:

Alice encrypts $\left\lfloor \frac{d}{2^{\ell}} \right\rfloor$ and sends $\left[\left\lfloor \frac{d}{2^{\ell}} \right\rfloor\right]$ to Bob.

Bob computes $[\![\delta]\!] = \left[\!\left[1 + \left\lfloor \frac{r}{2^{\ell}} \right\rfloor\right]\!\right] \cdot \left[\!\left[\left\lfloor \frac{d}{2^{\ell}} \right\rfloor\right]\!\right]^{-1} \cdot [\![\delta']\!] \bmod N^2$.

Bob sends $[\![\delta]\!]$ to Alice.

Alice uses her private key to decrypt $[\![\delta]\!]$ and get δ

DGK comparison protocol

The problem of comparing two private unencrypted integers have been studied intensively and several protocols are proposed to solve such a problem. See [22] for more detail. We have utilized the solution of DGK [20] due to its efficiency. DGK comparison protocol enables two parties with private inputs β and $\alpha, 0 \leq \beta, \alpha, < 2^{\ell} < N$ to know the comparison bit encrypted bit [δ′], such that $\delta' = (\beta < \alpha)$. The intuition to DGK comparison is to scan both bits of the two integers from left to right searching for the first differing bit. The value of such bits decides the overall comparison of both integers. In more details, suppose both integers β and α include ℓ bits denoted by β_i and a_i respectively. The most significant bit is denoted by $\ell - 1$. The protocol computes the ciphertexts c_i, for all $i = 0, \dots, \ell$, such that these values equal to zero only when $\beta_i = a_i$ for each $j, i < j < l$ and at the same time $\beta_i \neq \alpha_i$. Particularly:

$$c_i = s + \alpha_i - \beta_i + 3 \sum_{j=i+1}^{\ell-1} (\alpha_j \oplus \beta_j) \quad , 0 \leq i < l \qquad (3)$$

The random s can be either 1 or -1. It is used to prevent the other party (Alice) from learning whether the protocol outputs the comparison result of $\beta < \alpha$ or $\beta > \alpha$. The summation of s, a_i, β_i may result false zeros, so weight 3 is used to prevent this. The basic steps of DGK comparison protocol is described in Protocol 4 (which is detailed in Appendix A for space limitation).

5. OUR CONSTRUCTION

In this section, we will present our system as a solution to the problem described above. The work in our system consists of two parts: initialization and matching, which includes three main steps.

1. ***Document representation***: Alice and Bob transform both of their documents into vectors. The size of its vocabulary is constant. This step is part of the initialization phase.

2. ***Similarity score computation:*** In the matching phase, we measure the similarity by the cosine metric between the query vector of Alice and all vectors owned by Bob.

3. ***Finding maximum score:*** The index of document which has the maximum similarity selects in this step.

Document representation

In our work we used the vector space model to represent documents. We consider the assumption that both Alice and Bob's collection have the same terms. Given the document collection D of m documents, we first extract all the distinct terms (keywords) from D denoted as the vocabulary $W = \{w_1, w_2, \dots, w_n\}$ of n terms. Each document $d_i \in D, i = 0, .., m$ is represented as a single vector $\overrightarrow{d_i} = \{occw_1, occw_2, \dots, occw_n\}$, where $occw_j$ denotes the number of occurrences of term w_j in document d_i. If the term does not exist in the document, its occurrence is zero.

Similarity score computation

To compute the similarity score of Alice's input document U against all documents of Bob's collection D, she first represents her documents as a vector $\vec{u} = \{u_1, \dots, u_n\}$ of size n, and then she protects the privacy of her vector by encrypting separately the n integer elements of the generated vector. We use Paillier cryptosystem explained in section 4.2 to perform such a task. Specifically, she sends $[\![\text{occw}_1]\!], \dots, [\![\text{occw}_n]\!]$ to Bob. Once receiving the encrypted vector, Bob could still perform some of the basic operations over the encrypted data without decryption. This is due to the homomorphic operations of the adopted cipher. To match the received encrypted vector, Bob needs to compute the similarity between this vector and all vectors $\overrightarrow{d_i}, i = 1 \dots m$ in collection D. For this work, we use cosine similarity metric. To implement such a metric securely and efficiently, we apply it over normalized vectors. The normalized vector u is calculated as:

$$u_j = \frac{u_j}{\|u\|}, \|u\| = \sqrt{\sum_{j=1}^{n} u_j^2} \qquad (4)$$

To make vectors suitable for encryption operations, we have multiplied all normalized vectors by 100 to get rounded integer values. Given two normalized vectors, their cosine similarity of two vectors is simply equivalent to perform the dot product operation between their elements. Thus, Bob can compute the encrypted similarity score $[\![D_i]\!]$ between normalized vector u and the stored vectors $\overrightarrow{d_i}$ as follows:

$$[\![D_i]\!] = \prod_{j=1}^{n} [\![u_j]\!]^{\overrightarrow{di}_j} \qquad (5)$$

To improve the efficiency of similarity score computation, Bob ignores every encrypted element in Alice's vector where the corresponding element value at his vector is zero. Such a trick reduces the number of not needed modular multiplications.

Finding maximum score

Once similarity scores $[\![D_1]\!] \dots . [\![D_m]\!]$ between the input vector and all the stored vectors of Bob are obtained. The similarity corresponds to the best matching term vector should be found and its index is returned to Alice. Alice and Bob engage in a secure protocol to find the maximum similarity score. To do so, we need to compare every similarity pairs, but this is too slow since it needs a quadratic number of comparisons. Instead, we utilize the protocol of [23] to achieve this goal, where it preserves a linear number of comparisons. At the beginning, Bob permutes his pairs by using random permutation to prevent Alice from learning the order of the similarity values. Then, Alice and Bob start to compare similarity scores $[\![D_1]\!]$ and $[\![D_2]\!]$ using protocol 1. The result is that Alice learns the index Ind of the maximum score. They compare $[\![D_{Ind}]\!]$ and $[\![D_3]\!]$ next. Repeating such process through m comparisons Alice will be able to know the index of the global maximum similarity score. Finally, Bob gets the original order of the obtained index by re-permutation. However, Bob can still learn the maximum of every two compared similarities. To prevent him from knowing this leakage of information, Alice re-randomizes the encryption of the maximum value after each comparison. The main steps of our scheme are highlighted in Protocol 2.

Protocol 2. Privacy-preserving document similarity detection (PPDSD)

Input: U: Alice's query document. $D = \{d_1,\dots,d_m\}$: Bob's collection.

Output: Alice and Bob: *Ind*: the index of best match.

Initialization phase:

Alice: Generates Pallier homomorphic encryption public key pair (pr, pk).

Generates DGK homomorphic encryption public key pair (gr, gk).

Sends pk and gk to Bob.

Represent U as a vector $\vec{u} = \{u_1, \dots, u_n\}$

Normalize the vector $\vec{u}$ using equation (4)

Bob: For each document $d_j \in D, j = 1, \dots, m$.

Represents the documentd_j as a vector $\overrightarrow{d_j}$ of size n.

Normalizes the result using equation (4)

Matching phase:

Alice: Encrypts the elements of vector $\vec{u}$ as: $[\![\overrightarrow{u_j}]\!] \leftarrow Enc_{pk}(u_j), j = 1, \dots, n$

Send $[\![u_j]\!], j = 1, \dots, n$ to Bob

Set ind $\leftarrow 1$

Bob: **{Similarity score computation}** Calculates the secure dot product $[\![D_j]\!], j = 1, \dots, m$,between Alice's encrypted vector and $\overrightarrow{d_j}$ $j = 1, \dots, n$ using equation (5).

{Finding maximum score}
Bob: Pick a random permutation π over $\{1, \dots, m\}$

Set $[\![max]\!] \leftarrow D_{\pi(1)}$

For $i = 2$ to m do

Bob: Runs protocol 1 with Alice on inputs $([\![D_{\pi(i)}]\!], [\![max]\!])$ to get the comparison bit δ.

Bob: Chooses two random integers $\rho_i, \sigma_i \in (0, 2^{\ell+\alpha})$

Bob: Sets $[\![max']\!] \leftarrow [\![max]\!].[\![\rho_i]\!]$

Bob: $[\![D']\!] \leftarrow [\![D_{\pi(i)}]\!].[\![\sigma_i]\!]$

Bob: Sends $[\![max']\!]$ and $[\![D']\!]$ to Alice

Alice: If δ=1 then ind $\leftarrow$ i, $[\![v]\!] \leftarrow$ re − randomize $([\![D']\!])$

Else $[\![v]\!] \leftarrow$ re − randomize$([\![max']\!])$

Alice: Sends $[\![v]\!]$ to Bob.

Bob:Computes $[\![max]\!] = [\![v]\!].([\![\delta]\!].[\![1]\!]^{-1})^{\rho_i}.([\![\delta]\!]^{\sigma_i})^{-1}$.

//max= v+($\delta - 1$). $\rho_i - \delta.\sigma_i$

End for//i

Alice: Sends Ind to Bob.

Bob: Sets *Ind* = $\pi^{-1}(\text{Ind})$, send Ind to Alice

In situation where Alice is interested to retrieve only results within specific threshold τ, Protocol 2 can be easily updated such that an additional comparison is performed between $[\![\tau]\!]$, and the maximum similarity score $D(\pi^{-1}(\text{Ind}))$. If the latter is higher than the provided threshold, a match is reported and the Ind is returned to Alice.

6. SYSTEM COMPLEXITY

In this section, we investigate the complexity of our proposed scheme in terms of computing and communication costs.

Computation cost. During the matching phase, Alice encrypts component-wise vector using Paillier cryptosystem; each encryption operation needs one rather complex exponential operation and two multiplications. Computing the encrypted similarities requires performing m secure similarity computation. Finding the maximum score performs $m - 1$ comparisons over encrypted integer of $\ell -$ bits, which needs 7 Paillier homomorphic operations per comparison. Moreover, each comparison calls DGK–comparison to compare two unencrypted integers, with $2\ell + 1$ DGK encryption operations, 2 Paillier encryption operations.

Communication and round complexities. Our scheme employs two cryptosystems: Paillier and DGK with cipher texts communication overhead of 2084, 1024-bit, respectively. The initialization phase requires only one round to share the public

keys between the two parties. The matching phase requires a fixed number of rounds. One round is needed to send the encrypted query vector and receiving the result, with a communication cost of $n+1$ Paillier cipher texts. Finding the maximum similarities requires $m-1$ secure comparisons of encrypted integers. Every comparison of two encrypted integers needs four rounds: two rounds with 3 Paillier cipher texts, and two rounds for performing Protocol 4, which is used to compare two unencrypted integers. Protocol 4 requires two rounds, with 2ℓ DGK communication cost. Thus the overall complexity of our scheme is $4(m-1)$ rounds.

7. SECURITY ANALYSIS

In this section we try to show that Alice should not learn any information about the collection holds by Bob except what is allowed to be discovered according to the system's functionality. At the same time Bob should not learn anything about Alice's vector.

We discuss the security of each step in our proposed scheme. Alice computes its vector and encrypts it using her public key, so nothing is leaked to Bob. Similarity computing is done completely at Bob's side without any interaction with Alice, so it is completely a secure step. Selectin the maximum similarity step requires collaboration between the two parties. All received messages of this step are encrypted by Alice public key, so Bob could not get anything. However, Alice could get useful information from these messages, as she knows the private key. We claim that Alice will not be getting more useful information beyond what is described by the scheme. This is because; all received messages are masked by Bob. In more detail, in protocol 2, Alice receives the comparison bit δ. However this is not useful since Bob changes the order of all similarities to be compared. During the execution of protocol 4, Alice reveals $c_i, i=1,\dots,\ell$ for each comparison. Again such information does not help Alice, as these values are masked by Bob and the comparison direction is protected by s value.

8. EXPERIMENTAL RESULTS

In this section, we report the experimental results of the proposed scheme on a real documents database containing 990 documents from 20news collection [24]. Our experiments were performed computer of 2.2 GHz Intel i5 CPU, with a Windows 7 operating system of 64-bits, and 4 GB RAM. Our scheme has been implemented in JAVA jre1.8.0_1911. Big numbers are implemented by BigNumber Java library. For Paillie and DGK cryptosystems we set k=1024. We use $\ell = 22$. For generating the vocabulary set we, remove all non-letters words and convert all words to small letters. After removing the stop words, we apply Porter algorithm to stem the remaining words. We used a stop word list including 418 words. The collection contains 13,826 unique terms (the size n of the vector space)

The first experiment reports the running time to compare the query document vector against a collection of m documents. Figure 3 illustrates that comparing the query document against 990 documents requires about 3.2 minutes. This time includes: encrypting query vector, similarity scores computation, and finding the maximum score.

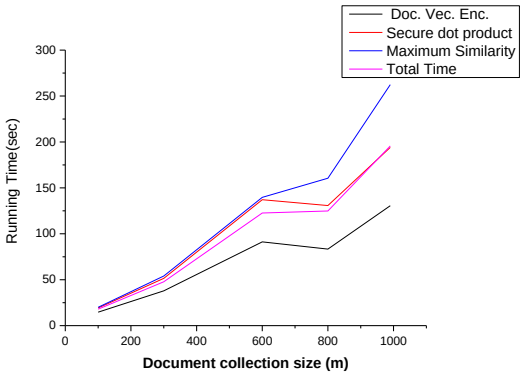

Figure 3: Running Time for Various Document Collection Size (m)

Recall that our scheme uses frequently two additively homomorphic encryption schemes: Paillier and DGK cryptosystems. The encryption operation of Paillier cryptosystem uses $r^N \, mod \, N^2$ to provide a sematic security, while DGK uses $h^N \, mod \, N^2$, where r is a fresh number. Such complex modular exponentiation can be pre-computed off-line in advance, as they do not depend on the provided query vector. Figure 4 shows that the matching time of the provided query vector against 990 documents is reduced to only 1.7 minutes.

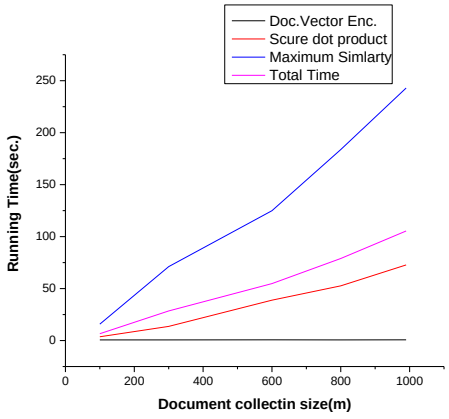

Figure 4: Running Time Pre-computing

The computational cost for secure dot product operation adopted in our scheme depends on the document collection size m and document vectors size n. Figure 5 shows that the running time for computing privacy preserving similarity function grows linearly as the document collection size increases. Furthermore, with a fixed collection size of larger vector, the running time becomes higher.

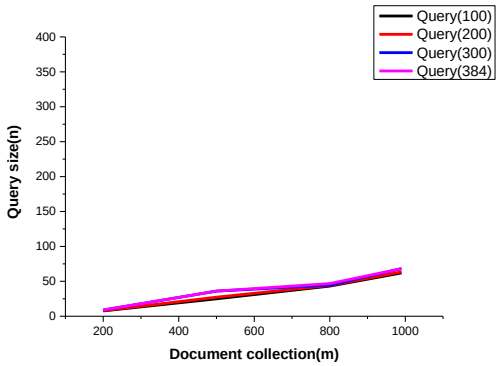

Figure 5: Computing Time for Secure Dot Product Operation

All the experiments listed above used the key size 1024 for Paillier cryptosystem, which is recommended for high security level. Figure 6 shows how the execution time of our protocol varies as the key size increases.

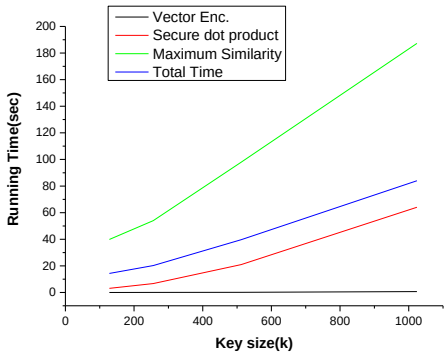

Figure 6: Effect of Paillier Key Size k

In the following experiment, we evaluate for different constructions document retrieval effectiveness to investigate the underlying models for document representation. We run 990 document queries are chosen from various topics (newsgroups). The documents are retrieved in database in a descending order according to their similarity with the

query. We used the precision-recall curve to better measure the Retrieval effectiveness, where (y-axis) is a plot of precision and (x-axis) is a recall for various thresholds. The retrieval effectiveness is better when precision value at a given recall value .Precision and recall metrics are explained as follows:

$$\text{precision} = \frac{|R \cap V|}{|R|} \quad , \text{recall} = \frac{|R \cap V|}{|V|}$$

Where, the set of retrieved documents is R and the set of relevant documents is V.
In this the experiment, relevant documents are defined to be those documents in the same group as the query. In Figure 7, we show the precision-recall curve for different constrictions: term-vector and N-gram. For the best CPU time, we select Min-hash-6 (P=6 hash functions). The best overall effectiveness for term vector representation and Min-hash approximations minimize some precision for best computation and communication costs.

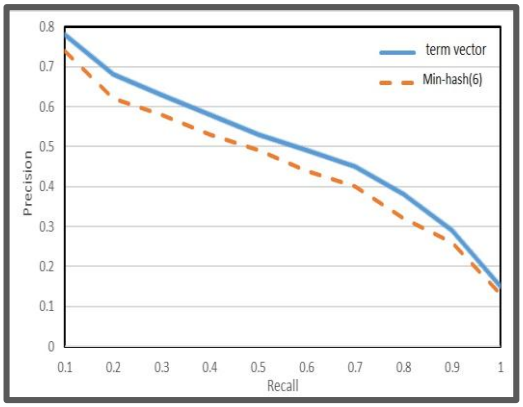

Figure 7: Precision-Recall Curve

Document similarity detection plays important roles in many real world applications. Current schemes assume that documents to be matched can be accessed freely. However, if parties are not willing to disclose their collections to each other, such as two security agencies whose collections are kept confidential, existing schemes are not desirable. Accordingly, in this paper, we have proposed a new scheme to solve this problem without compromising parties' privacy.

The proposed scheme is developed under the vector space model, which identifies the global similarity effectively. We adopted cosine similarity metric to measure the similarity of each two vectors, mainly because such measure can be incorporated easily in a privacy-preserving framework. As mentioned in Section 1, document similarity detection can be implemented using N-gram model, which has the advantage of identifying local similarities with overlapping text fragments. However, computing similarity under such a model requires an efficient yet secure scheme to identify the set of shared grams. Our current work is on exploring and implementing such schemes. For improving the efficiency of our proposed scheme, we try to employ clustering algorithms to significantly reduce the matching time while preserving high quality results, where similar documents are grouped together and representative is generated for each group. During the matching phase, the query vector is compared first against the small set of representatives to determine the closest group, then all documents belongs to that group are inspected.

REFERENCES

[1] U., Manber, *Finding Similar Files in a Large File System*, USENIX Association: San Francisco, California, (1994).

[2] G.S., Manku, A., Jain, and A.D., Sarma, *Detecting Near-duplicates for Web Crawling*, in Proceedings of the 16th international conference on World Wide Web, ACM: Banff, Alberta, Canada, (2007).

[3] S. Thandra, Unger, N., et al. Elxa: *Scalable Privacy-Preserving Plagiarism Detection*, ACM on Workshop on Privacy in the Electronic Society. Vienna, Austria, ACM, (2016).

[4] W., Jiang, et al., *Similar Document Detection with Limited Information Disclosure*, IEEE 24th International Conference on Data Engineering, (2008).

[5] M., Murugesan, et al., *Efficient privacy-preserving Similar Document Detection,* The VLDB Journal, **19**(4): p. 457-475 (2012).

[6] W., Jiang, and B.K. Samanthula, *N-Gram Based Secure Similar Document Detection*, Springer Berlin Heidelberg, (2011).

[7] C., Blundo, E. De Cristofaro, and P. Gasti, *EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity*, Springer Berlin Heidelberg, (2011).

[8] C.D., Manning, et al., *Introduction to Information Retrieval*, Cambridge University Press, 496, (2008).

[9] C., Wang, et al., *Secure Ranked Keyword Search over Encrypted Cloud Data*, IEEE 30th International Conference on Distributed Computing Systems, (2010).

[10] W., Zhang, et al., *Secure Ranked Multi-keyword Search for Multiple Data Owners in Cloud Computing*, 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, (2014).

[11] Z., Guo, et al., *Secure Multi-keyword Ranked Search over Encrypted Cloud Data for Multiple Data Owners*, Journal of Systems and Software, **137**: p. 380-395, (2018).

[12] C., Orencik, M., Alewiwi, and E., Savas, *Secure sketch search for document similarity*, (IEEE,edn.), pp. 1102-1107,(2015).

[13] B.K., Samanthula, and W., Jiang, *Secure multiset intersection cardinality and its application to jaccard coefficient*, IEEE Transactions on Dependable and Secure Computing, 13, (5), pp. 591-604, (2015).

[14] E., De Cristofaro, P., Gasti, and G., Tsudik, *Fast and Private Computation of Cardinality of Set Intersection and Union*, Springer Berlin Heidelberg, (2012).

[15] S., Buyrukbilen, and S., Bakiras, *Secure similar document detection with simhash*, (Springer, edn.), pp. 61-75, (2013).

[16] C., Sadowski, and G., Levin, *Simhash: Hash-based similarity detection*, (Technical report ,Google), (2007).

[17] X., Yu, X., Chen, J., Shi, L., Shen, and D., Wang, *Efficient and Scalable Privacy-Preserving Similar Document Detection*, (IEEE, edn.), pp. 1-7, (2017).

[18] G., Oded, *Foundations of Cryptography*, Volume 2, Basic Applications, Cambridge University Press, 452, (2009).

[19] P., Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Springer Berlin Heidelberg (1999).

[20] I., Damgård, M. Geisler, and M. Krøigaard, *Efficient and Secure Comparison for On-Line Auctions*, in

Information Security and Privacy, Springer Berlin Heidelberg, (2007).

[21] T.,Veugen, *Improving the DGK Comparison Protocol*, IEEE International Workshop on Information Forensics and Security (WIFS), (2012).

[22] T., Veugen, et al., *Secure Comparison Protocols in the Semi-Honest Model,* IEEE Journal of Selected Topics in Signal Processing, **9**(7): p. 1217-1228, (2015).

[23] R., Bost, Tu, S., Popa, S., Goldwasser, *Machine Learning Classification over Encrypted Data*, US, California, (2015).

[24]https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

APPENDIX A

Protocol 4: DGK comparison protocol.

Input: Alice: β, DGK cryptosystem key pairs (gp, gk). Bob: α, gk

Output: Alice: no thing, Bob: $[\![\delta']\!]$

Step1: Alice uses DGK cryptosystem to encrypt the individual ℓ −bits of her input β, that is $[\beta_0], \dots, [\beta_{\ell-1}]$, and sends them to Bob.

Step2: For each $i = 0, \dots, \ell - 1$ Bob computes the XOR of the encrypted bits $[\alpha_i \oplus \beta_i]$ as follows: If $\alpha_i = 0$ then $[\alpha_i \oplus \beta_i] = [\beta_i]$ else $[\alpha_i \oplus \beta_i]=[1].[\beta_i]^{-1} \bmod N$.

Step3: Bob randomly chooses the number s $\in\{1,-1\}$ and compute the following values:

$$[c_i] = [s].[\alpha_i].[\beta_i]^{-1}.(\prod_{j=i+1}^{\ell-1}[\alpha_i \oplus \beta_i])^3, i = 0, \dots, \ell - 1 .$$

Step4: Bob masks the numbers $[c_i]$ by raising them to a random exponent r of 2t bits: $[c_i] = [c_i]^{r_i} \bmod N$, and sends them in random order to Alice.

Step5: Alice checks whether one of the received ci values is decrypted to zero. If so, Alice sends $[\![\mathcal{E}]\!] = [\![0]\!]$. Otherwise she sends$[\![\mathcal{E}]\!] = [\![1]\!]$.

Step6: Bob uses the value of s and the encrypted bit τ to generate the encrypted comparison bit $[\![\delta']\!]$: $[\![\delta']\!] = [\![\beta < \alpha]\!] = \begin{cases} [\![\mathcal{E}]\!], & if\ s = 1 \\ [\![1]\!].[\![\mathcal{E}]\!]^{-1}, & if\ s = 0 \end{cases}$

كشف التشابه الامن للملفات

دعاء فضل نجم , اياد ابراهيم عبد السادة
قسم علوم الحاسوب , كلية التربية للعلوم الصرفة , جامعة البصرة
E-mail: [1]duaa.najim@gmail.com , [2]ayad.abdulsada@uobasrah.edu.iq

الخلاصة:

يلعب اكتشاف تشابه الملفات دورا اساسيا في العديد من التطبيقات الحيوية كحماية الملكية، ادارة الملفات، اكتشاف الاستلال، واكتشاف التقديم المتكرر للمقالات العلمية. الطرق الموجودة لمعالجة هذه الحالات عادة ما تفترض ان محتوى الوثائق النصية يكون غير خاص. وهذا يقلل من الاستفادة من تلك الطرق لتعمل فقط ضمن بيئات تكون فيها الملفات معلنة. اساسا، هذا الافتراض يحد من الكثير من التطبيقات، مثلا اكتشاف تشابه الوثائق النصية بين مؤسستين امنيتين، حيث تكون الوثائق سرية. في السنوات الاخيرة، طوّر مختصي التشفير بروتوكولات لاكتشاف تشابه الوثائق بدون الحاق الضرر بالخصوصية. على كل حال، البروتوكولات الموجودة غير امنة. في هذا العمل، عنونّا مسالة حساب تشابه مجموعتين من الملفات النصية تابعة لطرفين، لا يرغبان بكشف محتوى وثائقهم للطرف الاخر. تم توظف نموذج فضاء المتجهات لصياغة الملفات النصية بشكل متجهات واستخدام مقياس الجيب التمام الامن لقياس تشابه تلك المتجهات، واخيرا يسترجع دليل الوثيقة التي لها اكثر تشابه فقط. تم اجراء العديد من التجارب العملية على بيانات حقيقية .

الكلمات المفتاحية : تشابه الملفات , الخصوصية , تشفير هومورفك , تشفير بايلر , حساب متعدد الامن .