A Secure Index for Document Similarity Detection
الفهرس الآمن لاكتشاف تشابه الملفات

Ayad Ibrahim Abdulsada
Department of Computer Science, College of Education, Basrah University, Basrah, 61004, Iraq.
Email: ayad.abdulsada@uobasrah.edu.iq

Abstract

The document similarity detection plays an essential role in many applications such as plagiarism detection, copyright protection, document management, and document searching. However, the current methods do not care to the privacy of the contents of documents outsourced on remote servers. Such limitation reduces the utilization of these methods. For example, plagiarism detection between two conferences should protect the privacy of the submitted papers. In this paper, we consider the problem of privacy-preserving similarity document detection. The proposed scheme allows comparing documents without disclosing them to the untrusted servers. For each document, the fingerprint set is computed. The inverted index is built based on the entire fingerprint set. The index is protected by Paillier cryptosystem before uploading it to the untrusted server. We have developed a secure yet efficient method to rank the retrieved documents. Several experiments are conducted to investigate the performance of the proposed scheme.

Key words: Document similarity, inverted index, security, document fingerprinting, Paillier encryption.

الخلاصة

يلعب اكتشاف تشابه الملفات دور أساسي في العديد من التطبيقات كاكتشاف الاستلال, حماية النسخ, إدارة الملفات, والبحث عن الملفات. الطرق الموجودة لا تولي أهمية لخصوصية الملفات المراد مطابقتها. هذا القصور يقلل من الاستفادة من هذه الطرق. على سبيل المثال, اكتشاف الاستلال بين مؤتمرين يجب أن يحفظ خصوصية البحوث المقدمة. في هذا البحث, قمنا بأخذ مسالة الحفاظ على خصوصية الملفات المراد مطابقتها بنظر الاعتبار. المشروع المقترح يسمح بمقارنة الملفات بدون إفشاء أي شي للخادم غير الموثوق. قمنا بحساب البصمة لكل ملف. تم إنشاء الفهرس من مجموعة البصمات. تم حماية الفهرس باستخدام شفرة Paillier قبل نقل الفهرس إلى الخادم. كما قمنا بتطوير طريقة كفؤ وآمنة لترتيب الملفات المسترجعة. تم إجراء العديد من التجارب لإظهار انجازية الطرية المقترحة

الكلمات المفتاحية: تشابه الملفات, الفهرس المعكوس, الأمنية, بصمة الملف, شفرة Paillier.

1. Introduction

The *document similarity detection* (DSD) is ubiquitous in many practical applications. To gain an efficient file access, similar files are bring together to form clusters. Such that once providing a document, all the similar documents are retrieved efficiently. DSD can also be applied to detect whether the newly submitted article to a journal includes plagiarized contents. However, all the current solutions of DSD assume that the document collection is public and thus do not care to the privacy of documents that need to be matched. Such limitation reduces the utilization of these methods in our real life applications.

There are many practical applications where detecting the similar documents to a given query document in a privacy-preserving manner is needed. For example, to better understand the spread diseases, a number of different health agencies want jointly to check the similarity between their reports. For privacy issues, no agency reveals its report to the others. Such that it's urgent need to compare the underlying reports without compromising the privacy. Furthermore, most journals prevent the double submission of the same article. Thus, privacy preserving DSD is needed to check

whether the same paper is submitted into more than one journal at the same time without comprising the privacy of each journal.

In fact, encryption represents the best method to protect the privacy of the stored documents [15]. But encryption makes the work of the traditional DSD methods a challenging task. Thus, it's necessary to find an efficient method that measures the similarity between two documents at the encrypted domain.

Basically, the current DSD methods can be classified into two approaches: hashing[1-5] and vector space[6, 7]. In hashing approach, a set of fixed length substrings is extracted from the document. Then the hash code of each substring is computed. Finally, a compressed yet descriptive fingerprint is generated from the hash codes. Two documents are considered similar if they contain a significant number of shared fingerprint terms more than a predefined threshold. Such approach is more suitable to catch the local similarities, i.e. finding the overlapped contents between two documents.

On the other hand, vector space approach employs the *information retrieval*'s (IR) concepts to identify the global similarity information. Under the vector space model, each document is represented as a vector of terms or words, and each entry of the vector indicates certain frequency information of the corresponding term. Two documents are considered similar under such a model if they have common terms with similar term frequencies. However, vector space model detects the similarity of the whole documents. Consequently, it marks two documents with the same bag of words to be similar even if they have different contents.

Our proposed scheme utilizes the hashing approach to generate the fingerprint for each document. Where fingerprint is a representative yet compressed set of numbers. From the fingerprint set of the entire document collection, we build the inverted index structure. Inverted index structure is extensively used in information retrieval community [8] to provide fast retrieval. To utilize the appealing features of the inverted index in the context of secure data, we build a secure inverted index and build a secure DSD scheme on top of this index, where a secret key is used to encrypt the index in such a way that allow to measure the similarity without leaking the contents of the underlying data. Without learning this key no one can generate a valid fingerprint for the documented wanted to be matched. In this paper, we adopt the *syntactic* similarity notion. Briefly, under such notion two documents with different keywords will not consider to be similar even they have the same meaning. On the other hand, *semantic* notion employs sophisticated methods to consider the document meaning during its matching process. The latter notion is outside the scope of this paper.

Our contribution in this paper can be summarized as follows. First, we utilize the fingerprint approach, for the first time, to generate a secure index and build our secure DSD depending on this index. Second, we develop a secure solution to compute the common fingerprint terms of the provided document and the entire stored collection. Third, we have conducted several experimental results to show the performance of our approach.

The rest of this paper is organized as follows. Section 2 illustrates the related literature. Section 3 introduces document fingerprinting technique. Section 4 introduces the problem definition. Section 5 shows the proposed approach. In Section 6 we investigate the performance of our approach. Finally, Section 7 concludes the whole paper.

2. Related Work

The first practical scheme for detecting the similar documents is due to Manber [1]. Such scheme employs the hashing concept to measure the similarity of large document system. Where a set of fingerprints is calculated from the substrings of document and compared to detect the similarity. Later on, the notion of fingerprint is updated in [2] to be calculated on the sentence level. However, such scheme has been proven to be sensitive to the modification in the sentence unit. A system for plagiarism detection is proposed in [3], where the number of shared words in the same sentence is used to measure the similarity. Schleimer S. et. al [4] have presented the *Winnowing* algorithm; an efficient local fingerprinting algorithm to capture an essential property of any fingerprinting technique guaranteed to detect similar documents. The work of [5] has exploited the *Winnowing* algorithm to design a system for detecting the plagiarism among the text documents. Recently, the authors of [9] have proposed a compact document representation that can be used to efficiently prune duplicate and near-duplicate documents from result lists.

In the context of secure DSD, a little work is presented. The authors of [7] have used the vector space model to represent the documents, and adopt the cosine similarity to measure the global similarity between each document pairs. The main disadvantage of this approach is defining the global similarity but not the local similarity. Recently, the authors of [10] have used the *k*-gram technique to find the local similarity. They use the random share method to compute the distance between the *k*-gram sets in a privacy reserving manner. However, such method requires an extensive computational and storage cost. Moreover, such work lakes to the experimental results.

3. Document Fingerprinting

Fingerprinting is an efficient technique to detect full and partial document copies using hash codes. To get the fingerprint, document is first divided into a large set of *k*-grams. The *k*-gram set is hashed, and then compressed by selecting a subset of these codes to be the fingerprint. The main problem is which hash will be chosen to be included in the fingerprint. In this paper, we use *Winnowing* algorithm [4] which selects the smallest hash value from *w* window slides. With such hashed fingerprints, there are certain bounds for detecting similarity between copied and original documents. In what follow, we give a brief summery about the work of this algorithm.

Given the string *S* as sequence of *n* characters. The *k*-gram is substring of length *k*. For example, the 4-grams of the string *S*='to be or not to be' is {'tobe', 'obeo', 'beor', 'eorn', 'orno', 'rnot', 'nott', 'otto', 'ttob', 'tobe' } of length *n-k+1*. Hashing the *k*-gram set is achieved by Karp-Rabin's algorithm [11]. This algorithm allows the hash of the i+1st *k*-gram to be computed efficiently from the ith *k*-gram. Suppose that the first *k*-gram is set of $t_1 \ldots t_k$ numbers in the base *b*. Then we can hash these numbers as:

$$F_1 = (t_1 * b^{k-1} + t_2 * b^{k-2} * \ldots + t_{k-1} * b + t_k) \bmod M, \qquad \ldots(1)$$

where *M* is a constant defined by the user.
The second *k*-gram of $t_2 \ldots t_{k+1}$ numbers is computed efficiently as follows:

$$F_2 = ((F_1 - t_1 * b^k) * b + t_{k+1}) \bmod M$$

Generally, the ith *k*-gram is computed as:

$$F_i = ((F_{i-1} - t_1 * b^k) * b + t_{k+i-1}) \bmod M, \forall i = 2..n \qquad \ldots(2)$$

Now we describe how the *Winnowing* algorithm selects the fingerprint among the $F_1 \ldots F_{n-k+1}$ hash codes. Given two documents, this algorithm guarantees to find the substring matches between these documents that satisfies the following:

1- The length of the matched strings is not less than the *guarantee threshold, T.*
2- The length of the matched substrings is not exceeding the *noise threshold, k.*

Note that both $T > k$ and *k* are user defined values. Even so choosing a large value of *k* prevents the coincidental matching between two documents, it will limit the sensitivity to reordering of document contents, as we cannot detect the relocation of any substring of length less than *k*. So, it is necessary to choose a minimum value of *k* such that coincidental matches will be a negligible. The algorithm defines a window size as:

$$w = T - k + 1$$

Each position in the sequence $1 \le i \le (n-k+1)-w+1$ defines a window of hashes $F_i \cdots F_{i+w-1}$. In each window, we select the minimum hash value. If there is more than one hash with the minimum value in the same window, select the rightmost occurrence. The same hash value among successive windows will not be inserted in the fingerprint. All selected hashes are considered to be the fingerprint of document.

For example, suppose we have the following hash codes: 77 72 42 17 98 50 17 98 8 88 67 39 77 72 42 17 98, and suppose that window size *w=4*. The entire windows will be (77, 72, 42, ***17***) (72, 42, 17, 98)(42, 17, 98, 50) (17, 98, 50, ***17***) (98, 50, 17, 98) (50, 17, 98, ***8***) (17, 98, 8, 88) (98, 8, 88, 67) (8, 88, 67, ***39***) (88, 67, 39, 77) (67, 39, 77, 72) (39, 77, 72, 42) (77, 72, 42, ***17***) (72, 42, 17, 98). According to the *Winnowing* algorithm, the fingerprint will be: 17 17 8 39 17, which are illustrated by the bold font.

4. Problem Definition

Consider a data owner Bob of a document collection $D=\{d_1, d_2, \ldots, d_m\}$ of size *m*. Bob outsourcers the storage and computing of his collection into a remotely server *S* to enjoy high quality services in an efficient cost. However, the server *S* is not trusted to see the contents of the stored collection. Such that Bob has to encrypt his collection before outsourcing. For efficient retrieval, Bob also builds a secure index from the document collection and uploads it together to the untrusted server *S*. The user Alice has a document *Q*. She wants to detect the similarity between its own document against all the *m* documents of Bob that are stored in *S* without either reveling *Q* to *S* or revealing *D* to Alice. To do so, Alice first extracts the fingerprint of its document, encrypts it, and then sends the result to the server *S*. Once receiving the secure fingerprint of the document *Q* from Alice, the server *S* matches securely the provided fingerprint against its secure index and then responds to Alice by the scores $\alpha_1, \alpha_2, \cdots \alpha_m$, which represent the matching score for all the stored *m* documents. Finally, Alice downloads the top-*h* documents. Figure 1 illustrates the basic architecture of proposed scheme.

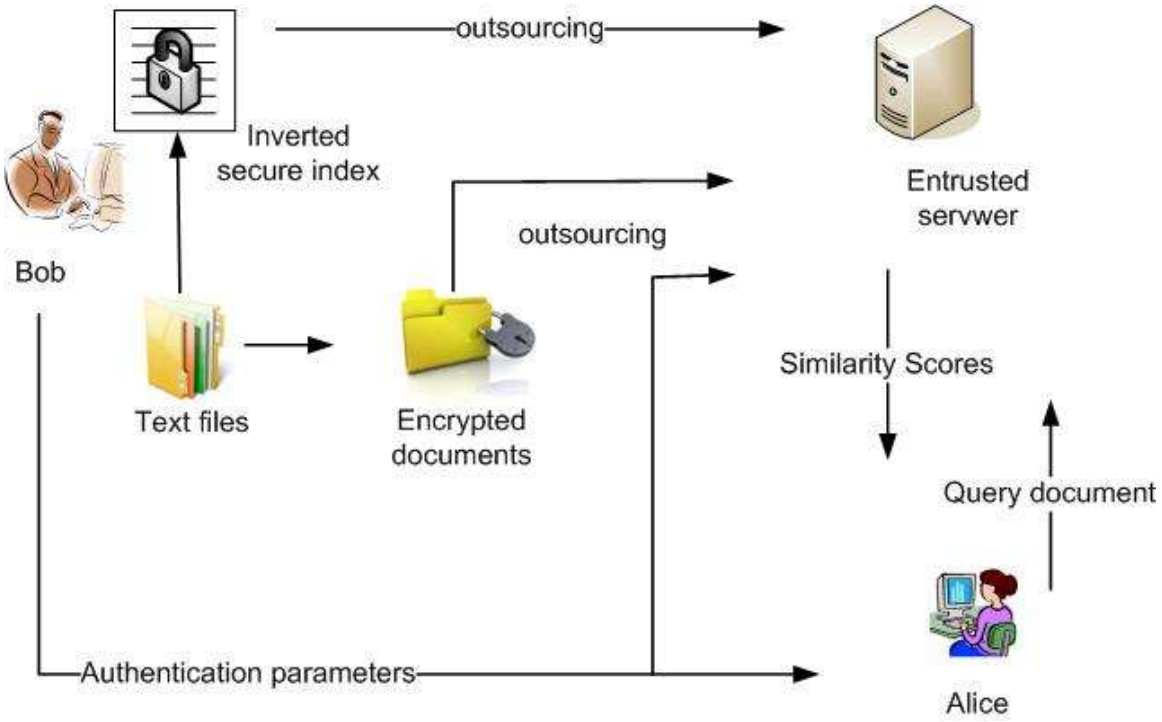

Figure 1: the proposed scheme

4-1 Security Requirements

In this paper, we assume that the *S* server is a *honest-but-curious* party, where it is trusted to follow the protocol, but at the same time it tries to learn as much information as possible from the stored data. The security requirements are illustrated as follows:

1- *Document collection*: The server *S* is not allowed to know the contents of Bob's collection.
2- *Query document*: Alice's document should not revealed to server *S* and without learning the secret key the server *S* could not generate a valid fingerprint.
3- *Index security*: The secure index does not leak anything about its contents.

5. Proposed Scheme

In this section, we describe our proposed scheme. To enable efficient document similarity detection, Bob builds a secure index and outsourcers it to the remote server along with the encrypted documents. The server performs comparison on the index according to the queries of the data users without learning anything about the data other than what Bob allows an adversary to learn. In Part 5.1, the index structure is presented. In Part 5.2, the DSD scheme that is built on top of the index is described.

5.1 Secure Inverted Index

Our proposed secure DSD is based on the inverted index structure. Secure inverted index is constructed with the following 2 steps.

1. *Fingerprint generation*: given the document collection $D=\{d_1, d_2, \ldots, d_m\}$. The data owner Bob generates the fingerprint for each document d_i. Recall that the fingerprint is a set of integer numbers (terms). Define $F=\{f_1, f_2, \ldots, f_l\}$ to be the union set of all the fingerprint terms of the document collection. The inverted index I includes a set of fingerprint items f_j and their corresponding posting lists P_j i.e $I=\{(f_j,P_j), j=1\ldots l\}$. The posting list refers to the set of

document IDs that contains the term f_j. The posting list P_j is simply a bit vector of size m where m is the total number of Bob's collection D. Given ID(d_i) to be the identifier of the document d_i, the element vector P_j[ID(d_i)]=1 if and only if d_i includes the fingerprint term f_j. Table 1 shows a simple index of 5 fingerprint terms that derived from 14 documents. For example, the term 500 appears in five documents (d_1, d_2, d_4, d_6, d_{14}, and d_{15}).

Table 1: inverted index example

Term	Vector													
500	1	1	0	1	0	1	0	0	0	0	0	0	1	1
520	0	0	0	1	0	1	0	0	1	0	0	1	0	0
600	1	0	0	0	0	0	0	1	0	0	0	0	0	0
680	0	0	1	0	0	0	1	0	0	0	0	0	1	0
710	1	0	1	0	0	0	0	0	0	1	0	0	0	1

2- Inverted index encryption: in this step, we turn the inverted index into a secure index by encrypting the fingerprint terms and their corresponding posting vectors. Fingerprint terms(numbers) should be encrypted such that only the authorized users can generate valid queries. Otherwise, the adversary server can learn the fingerprint terms of a given document. Similarly, the posting vectors should be encrypted to hide the number of documents in a given fingerprint item, which may be used to conduct a frequency attack. Our solution to protect the fingerprint terms is to consider the parameter b and M of equations 1 and 2 as a secret key $k_1=(b,M)$. Such that only the authorized users who have the secret key k_1 can generate a valid fingerprint. Encrypting the posting vectors is more difficult. This is because, such vectors have to be encrypted while preserving their ability to rank the retrieved documents. In this paper, we utilize the appealing feature of the Paillier cryptosystem [12] to alleviate such challenge. Paillier is a secure semantically and additive homomorphic asymmetric encryption scheme. The semantically secure feature ensures that encrypting the same number (0 and 1 in our case) multiple times will generate different ciphers. Let Enc_{kpub} and Dec_{kpriv} be Paillier encryption and decryption functions with the public and private keys *kpub* and *kpriv*, respectively. So if $m_1=m_2$ are equal messages, then $Enc_{kpub}(m_1) \neq Enc_{kpub}(m_2)$. But $Dec_{kpriv}(m_1)=Dec_{kpriv}(m_2)$. The additive homomorphic property means that $Enc_{kpub}(m_1+m_2)=Enc_{kpub}(m_1)+ Enc_{kpub}(m_2)$. We use Paillier cryptosystem to encrypt each bit of the posting vectors. Such that if P_j[ID(d_i)]=1 then we store $Enc_{kpub}(1)$. Otherwise, we store $Enc_{kpub}(0)$. Note that, each encrypted zeros and ones are distinct due to the fact that Paillier is a semantically secure encryption scheme. Once encryption is done, we need to add some fake records into the index to hide the number of fingerprint items in the collection.

5.2 Secure Document Similarity Detection

The work of our scheme can be summarized as follows:

1- *key generation*: Bob generates the secret key $k_1=(b,M)$, *Kpub*, *Kpriv*, and *Kcoll*, where b is the base parameter and M is a constant.

2- *Index construction*: Bob uses the secret keys K_1 and *kpub* to build the inverted index from the collection D.

3- *Document encryption*: Bob encrypts his collection D with the secret key *Kcoll*. He then sends the encrypted collection to the server S. Once data is outsourced, data user should be able to match their documents with the remote server. To do so, Bob shares the following information with data users:

- *Kcoll* : secret key of data collection encryption
- K_1: secret keys of index construction.
- *Kpriv*: the private key of the Paillier decryption function.

4- *Fingerprint construction*: Suppose that Alice wants to compare its document Q with the collection D. She first uses the secret key k_1 to generate the secure fingerprint items $Qf=\{qf_1, qf_2, \ldots, qf_c\}$ as in equations 1 and 2. Once doing that she sends Qf to the remote server S.

5- *Search*: given the fingerprint set *Qf*, the server *S* searches its index to find the matched terms. For each matching, *S* retrieves the corresponding posting vector i.e. retrieve $P_j=[e_{1j},....e_{mj}]$ such that $(f_j, P_j) \in I$ and $f_j=qf_j$ for all $j=1..c$, where e_{ij} is the encrypted bit of the document *i* corresponding to the fingerprint term *j*. Once doing that *S* has to calculate the score for each document d_i. To do so, *S* uses the additive property of Paillier cryptosystem to get the score for each document as follows: $\alpha(d_i) = e_{i1} + e_{i2} + ... + e_{ih}$, where *h* is the number of matched fingerprint items. Finally, the server *S* sends the scores $\alpha(d_1), \alpha(d_2), \ldots, \alpha(d_m)$ to Alice.

6- *Score decryption*: Alice decrypts the received score values $\alpha(d_i)$ for all $i=1...m$ by using the secret key *Kpriv* to know which document is similar to its own document *Q*.

7- *Document retrieval*: Alice asks the server *S* to retrieve the documents of the top-*h* scores. After retrieving the most similar documents, she decrypts them by *Kcoll* key to perform the manual investigation.

Continuing with our index illustrated in Table 1, suppose that the terms of the query document are: 400, 500, 600, 710, and 800. Thus the scores of the 14 documents are calculated as in Table 2. We see that document 1 is the most similar with 3 scores followed by document 14 with 2 scores and so on.

Table 2: Score calculation

Term	Vector													
500	1	1	0	1	0	1	0	0	0	0	0	0	1	1
600	1	0	0	0	0	0	0	1	0	0	0	0	0	0
710	1	0	1	0	0	0	0	0	0	1	0	0	0	1
Score	**3**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**2**

6. System Evaluation

In this section, we present the experimental evaluation of the proposed scheme. We have download 1000 text documents from the real dataset *Request For Comments* (RFC)[13]. During the fingerprint generation, we use 5-grams, and set the guarantee threshold *T* to 10. The results are the average of 10 runs. The document collection is encrypted by AES algorithm with 128-bit key. Our experiments were conducted on a 2.5GHz Intel i5-3210m processor, Windows 7 operating system of 64-bits, with a RAM of 4GB. We used MATLAB R2008a to implement our experiments. We used Java class to implement Paillier cryptosystem. The inverted index is stored in a hash table to provide *O(1)* access time.

6.1 Retrieval Evaluation

In this experiment, we test the success of our proposed scheme to retrieve the required documents. We use the precision evaluation metric to achieve this task. We have selected 100 random documents to be the query set. We compare our scheme with the resemblance measure for computing the similarity between two documents. The resemblance measure is defined as [14]:

$$\text{Resemblance} = \frac{|\text{Inter}|}{|\text{Union}|} \quad \text{...(3)}$$

Where *Inter* set represents the intersection set of two fingerprint sets, while the *Union* represents the union set. Recall that our ranking method employs only the intersection set without using the union set to measure the resemblance.

The precision of the query document *q* is defined as:

$$\text{Precision(q)} = \frac{|\text{R} \cap \text{A}|}{|\text{A}|} \quad \text{…(4)}$$

Where *R* is the retrieved documents under our scheme, while *A* is set of retrieved documents under equation 3. Given the set $Q = \{q_1, q_2, \ldots, q_x\}$ of *x* queries, we can compute the average precision as:

$$avg\_press = \frac{\sum_{i=1}^{x} precision(q_i)}{x} \quad \text{…(5)}$$

Figure 2 shows the average precision as the top-*h* documents increased. As expected increasing the *h* value will retrieve dissimilar documents.

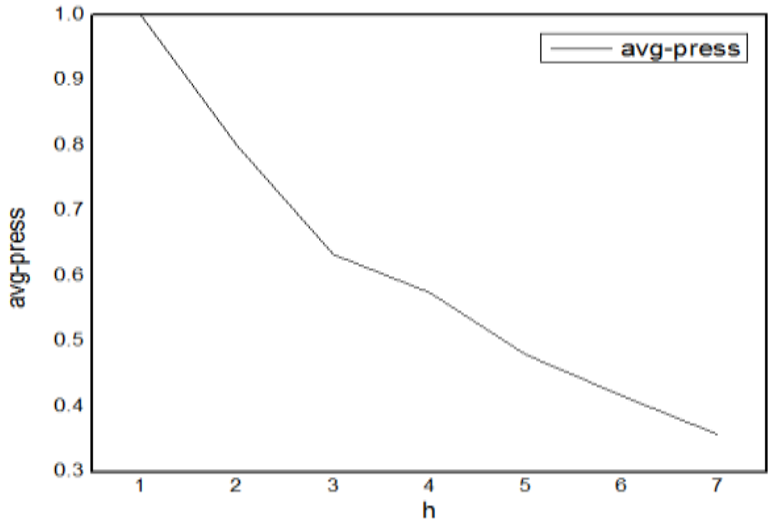

Figure 2: Retrieval evaluation

6.2 Effectiveness

In this experiment, we evaluate the ability of our proposed scheme to recover the inserted noise in the provided query. During the test, we have inserted different amounts of noise to the provided query ranging from 2% to 7% from the length of the query document. Noise is inserted into the query document at random locations. Figure 3 illustrates that the precision decreases as increasing the amount of the inserted noise.

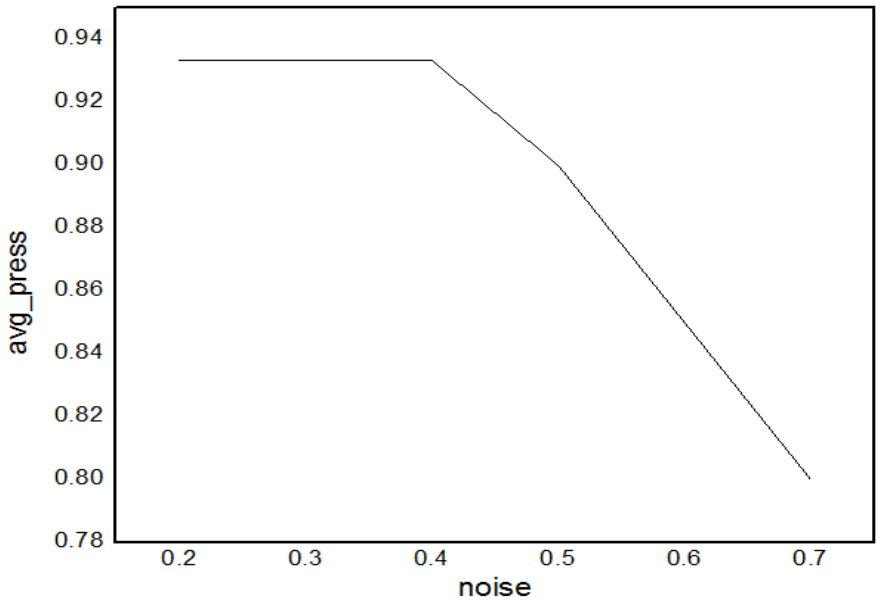

Figure 3: Effectiveness of the proposed scheme

6.3 Index Building

Building the inverted index by the data owner, Bob, includes the fingerprint generating step and encrypting the posting vectors. In this experiment, we show the effect of the collection size, *m*, and the *k* value for the *k*-gram on the index building time and the total number of individual fingerprint terms. Figure 4a shows that increasing *k* during computing the *k*-gram sets leads to increase the fingerprint terms. Similarly, increasing *k* requires more time to compute the fingerprint terms as explained in Figure 4b. Similarity, Figure 5a, and 5b shows the effect of increasing the document

collection size on the total number of fingerprint terms and indexing time, respectively. Again increasing the collection size leads to increase the two latter terms.

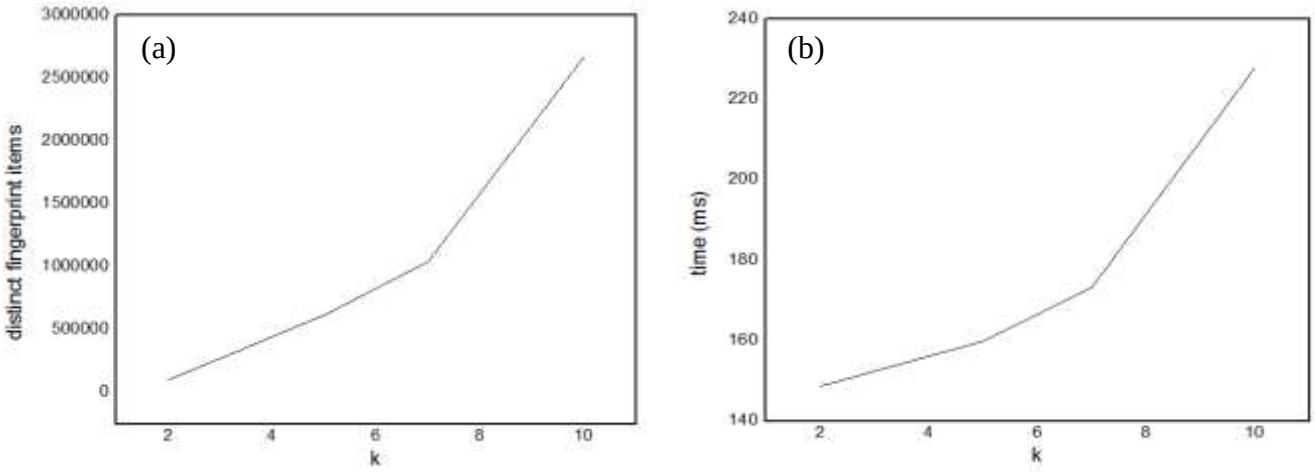

Figure 4: *k* effect on: (a): number of fingerprint items, (b): time of generating the fingerprint items

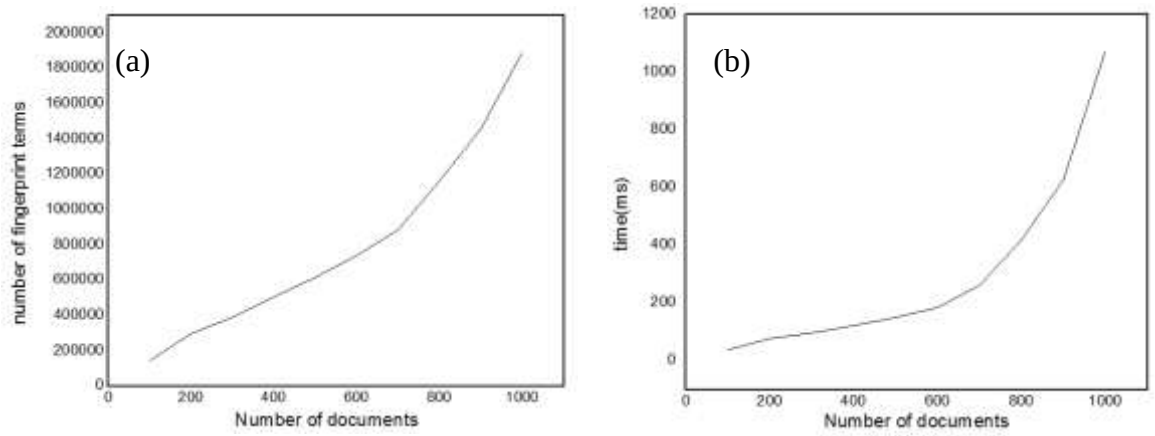

Figure 5: effect of collection size. (a) : fingerprint size, (b): indexing time.

6.4 Fingerprint Security

Recall that the fingerprint terms are protected by the secret key *K1=(b, M)*, where *b* is the base number used in equations 1 and 2. Without learning *b* no party can generate a valid fingerprint terms. In this experiment, we compute the resemblance as in (3) between a fingerprint set generated by the valid secret key, *b*=6, and 100 fingerprints generated by 100 different *b* values. One of these values is equal to the original one. Figure 6 shows that only the valid *b* value has the maximum resemblance.

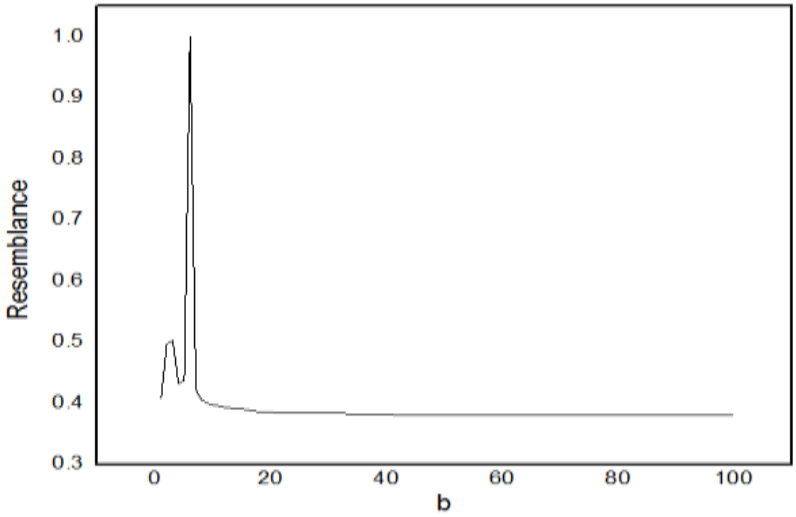

Figure 6: Effect of *b* on security

6.5 Ranking time

During the ranking time, our scheme sums the posting vectors of the matched fingerprint terms. Recall that all vectors are of a fixed length equals to the collection size *m*. Figure 7a shows that more matched fingerprint terms requires more ranking time. We have fixed the document collection size of this experiment to 1000. Also, Figure 7b illustrates that increasing the posting vector requires more ranking time. This is because, long vectors need more time to be added.

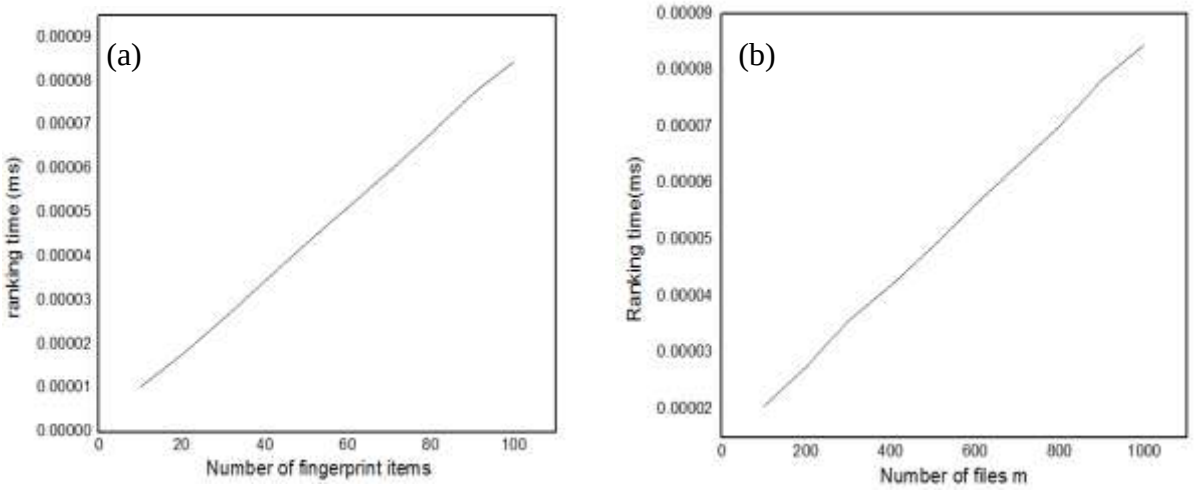

Figure 7: Ranking time: (a) increasing the fingerprint items, (b) increasing the document collection

7. Conclusion

In this paper, we have addressed the problem of secure similarity document detection. An efficient solution is presented for this problem. Our scheme uses the Winnowing algorithm to compute the fingerprint for each document. Fingerprints are a compressed version of the *k*-gram set with the ability to detect the full and partial copy. An inverted index is builds based on the fingerprint set to provide an efficient matching. Such index is protected by using Paillier homomorphic cryptosystem. Such cryptosystem allows summing the posting vectors without decryption. We have conducted several experiments to show the practical value of our solution.

References

[1] Manber, U.: Finding similar documents in a large document system. Department of Computer Science, The University of Arizona, Tucson, Arizona, Tech. Rep. TR 93-33. ftp://ftp.cs.arizona.edu/reports/1993/ TR93-33.pdf, 1993.

[2] Brin, S., Davis, J., Garcia-Molina, H.: Copy detection mechanisms for digital documents. In: Proceedings of the 1995 ACMSIGMOD Conference on Management of Data, pp. 398–409. ACM, San Jose, 1995.

[3] Collberg, C., Kobourov, S., Louie, J., Slattery, T.: SPlaT: a system for self-plagiarism detection. In: Proceedings of IADIS International Conference WWW/INTERNET 2003, Algarve, Portugal, pp. 508–514, Nov 5–8, 2003.

[4] Schleimer, S.,Wilkerson, D.S., Aiken, A.:Winnowing: Local algorithms for document fingerprinting. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 76–85, June 9–12, 2003. ACM, San Diego, 2003.

[5] Sorokin, D., Gehrke, J.,Warner, S., Ginsparg, P.: Plagiarism detection in arXiv. In: Proceedings of the Sixth IEEE International Conference on Data Mining (ICDM06), Hong Kong, China, pp. 1070–1075, Dec 18– 12, 2006.

[6] Shivakumar, N., Garcia-Molina, H.: Building a scalable and accurate copy detection mechanism. In: Proceedings of the First ACM International Conference on Digital libraries, Bethesda, MD,USA, pp. 160–168, Mar 20–23, 1996.

[7] Murugesan, M., Jiang, W., Clifton, C., Si, L., Vaidya, J.: Efficient privacy preserving similar document detection. The VLDB Journal, January 16, 2010.

[8] Manning, C. D., Raghavan, P., and Schütze, H.: Introduction to Information Retrieval, Reading, MA: Cambridge UP, 2008.

[9] Bernstein, Y., Shokouhi, M., Zobel, J.: Compact features for detection of near-duplicates in distributed retrieval. In: SPIRE,Glasgow, UK, pp. 110–121, Oct 11–13, 2006.

[10] Jiang, W., Samanthula , B. K.: N-Gram Based Secure Similar Document Detection, In: Proceedings of the 25th Annual WG 11.3 Conference on Data and Applications Security, Richmond, Virginia, July 11-13, 2011.

[11] Karp, R.M., Rabin, M.O.: Efficient randomized pattern-matching algorithm, In IBM J. Res. Dev. 31(2):249-260, 1987.

[12] Paillier, P: Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the 17th international conference on Theory and application of cryptographic techniques, ser. EUROCRYPT' 99. Berlin, Heidelberg: Springer-Verlag, pp. 223–238, 1999.

[13] RFC, Request For Comments Database, http://www.ietf.org/rfc.html.

[14] Broder, A.Z.: On the resemblance and containment of documents. In: Compression and Complexity of Sequences, pp. 21–29, 1997.

[15] Stallings, W.: Cryptography and Network Security: Principles and Practice, fifth ed., Pearson Education, 2002.