N-Gram based Secure Similar Document Detection

Wei Jiang and Bharath K. Samanthula

Department of Computer Science, Missouri S & T, Rolla, MO 65401
wjiang@mst.edu and bspq8@mst.edu

Abstract. Secure similar document detection (SSDD) plays an important role in many applications, such as justifying the need-to-know basis and facilitating communication between government agencies. The SSDD problem considers situations where Alice with a query document wants to find similar information from Bob's document collection. During this process, the content of the query document is not disclosed to Bob, and Bob's document collection is not disclosed to Alice. Existing SSDD protocols are developed under the vector space model, which has the advantage of identifying global similar information. To effectively and securely detect similar documents with overlapping text fragments, this paper proposes a novel n-gram based SSDD protocol.

Keywords: privacy, security, n-gram

1 Introduction

Textual information is ubiquitous and plays major roles in information dissemination. There are many practical situations where detecting documents that are similar to a given query document in a privacy-preserving way is beneficial. Consider an FBI agent looking for the data related to potential terrorist suspect. He or she may wish to check whether there are reports that are related to the suspect from local police databases. However, neither the FBI agent nor the local police wants to exchange their data unless there is a need to share. One way to identify such a need is to detect similarities or correlations between the the FBI's query (in form of textual document) and the local police's database of reports on criminal activities. Once the need for sharing information is verified, the FBI and the local police can exchange only the needed information. During the process of identifying similar or correlated information, it is the best interest for both parties not to disclose the query document and the database. Such a process is referred as secure similar document detection (SSDD).

The SSDD problem was first introduced in [6], where vector space model and Cosine Similarity are adopted to detect similar documents. 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. When the vectors contain normalized term frequency values, secure dot product protocols (e.g., [2]) can be used as the building block for an

SSDD protocol that identifies similar documents through Cosine Similarity. The same work was later extended in [8] to improve computation efficiency.

The vector space model is not the only way to compute document similarity, and n-gram based document representation [7] can also be adopted to detect similar documents. The vector space model has the advantage of detecting global similarity, e.g., a bag of similar terms. On the other hand, the n-gram model has the advantage of finding local similarity, e.g., overlapping of pieces of texts. Also, n-gram model is language independent and has simple representation. Plus, n-gram based document modeling is less sensitive to document modification. Thus, the goal of this paper is to propose a n-gram based SSDD protocol under the semi-honest model [3].

1.1 Related Work

The SSDD problem was first introduced in [6]. The key in their work is how to securely compute the similarity between two documents. Initially, both participating parties, Alice and Bob, compute local (their own) vector spaces for their document collections. Then each of their documents can be represented as a vector of terms or words. Cosine similarity is adopted to measure similarity between any two documents. The same work was later extended in [8] to improve computation efficiency by combining text clustering techniques.

2 The Proposed Method

We follow the same setting as in [6,8]. Let Alice and Bob represent two entities, each of whom has a collection of documents. Given a query document u from Alice, the goal of an SSDD protocol is to detect whether or not Bob's collection (denoted by $\mathcal{D} = \{v_1, \ldots, v_n\}$) contains a document similar to u without disclosing Bob's database to Alice and vice versa. SSDD is defined as follows [6]:

$$\text{SSDD}(u, \mathcal{D}) \rightarrow \sigma_1, \ldots, \sigma_n \tag{1}$$

Instead of returning the actual similar documents, SSDD returns n[1] similarity scores $\sigma_1, \ldots, \sigma_n$ to Alice. If one or more are particularly close, arrangements can be made (e.g., via agreed access control policies) to investigate further.

The SSDD protocol proposed in this paper is based on the n-gram model. In general, an n-gram is a subsequence (substring) of size n from a given sequence (string). An n-gram of size 1, 2 or 3 is referred as an uni-gram, bi-gram or tri-gram receptively. An n-gram of size 4 or more is simply called n-gram. If a text document is treated as a one big string, under the n-gram model, each document can be represented by a set of successive n-gram. Table 1 shows four pieces of texts and their corresponding unique tri-gram representations.

[1] Note that in this paper, n is used to indicate either Bob's document collection size or the length of the n-gram depends on the context.

	Text	n-gram representation
u	aabbc abbde	{aab, abb, bbc, bca, cab, bbd, bde}
v_1	ca aabbc ddf	{caa, aaa, aab, abb, bbc, bcd, cdd, ddf}
v_2	xaab xyyyz	{xaa, aab, abx, bxy, xyy, yyy, yyz}
v_3	xxx yyy xyyz	{xxx, xxy, xyy, yyy, yyx, yxy, yyz}

Table 1. Sample Texts

Under the n-gram model, the similarity between any two documents can be calculated using set similarity. A commonly accepted way to measure set similarity is to use *Jaccard Coefficient* (JC). Suppose u and v are two documents under the n-gram model, then JC between u and v is given by:

$$\mathrm{JC}(u, v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|} \tag{2}$$

Referring to Table 1, assume u is a query document and $\{v_1, v_2, v_3\}$ is Bob's document collection. The similarity between u and v_1 can be calculated by $\mathrm{JC}(u, v_1) = \frac{3}{7+8-3} = 0.25$. JC will be adopted as the similarity measure in our proposed SSDD protocol.

2.1 Secure Similarity Computation

In order to return only the JC scores, our SSDD protocol consists of two stages. Each stage is summarized as follows:

- **Stage 1** - Computing Random Shares of $|u \cap v_i|$:
  At the end of this stage, Alice receives a random number a_{1i} and Bob receives a random number a_{2i}, such that $a_{1i} + a_{2i} \mod N = |u \cap v_i|$, where N is a security parameter or an encryption key.
- **Stage 2** - Computing JC Score:
  Alice sets $b_{1i} = |u| - a_{1i} \mod N$ and Bob sets $b_{2i} = |v_i| - a_{2i} \mod N$. Alice and Bob securely compute $\frac{a_{1i}+a_{2i} \mod N}{b_{1i}+b_{2i} \mod N}$, without disclosing a_{1i}, b_{1i} to Bob and a_{2i}, b_{2i} to Alice.

To implement the first stage, Bob first generates a global space of n-gram based on his document collection, denoted by S. Then S is mapped to an integer domain from 1 to $|S|$. Let M denote such a mapping function and S_j denote the j^{th} element in S, then $M(S_j) = j$. Under the domain of $M(S)$, each document v_i can be represented by a binary vector $\boldsymbol{v}_i$ (i.e., a vector with 0/1 entries). The following properties hold for each $\boldsymbol{v}_i$:

- $|\boldsymbol{v}_i| = |S|$
- $\boldsymbol{v}_i[j] = 1 \Rightarrow S_j \in v_i$ and $\boldsymbol{v}_i[j] = 0 \Rightarrow S_j \notin v_i$, where $\boldsymbol{v}_i[j]$ denotes the j^{th} entry or dimension of $\boldsymbol{v}_i$.

Example 1. Following the documents in Table 1, Table 2 shows a global n-gram space based on Bob's document collection and the vector representation of each document on the global space. Note that not all n-gram in u appear in S because S is only derived from Bob's document collection. However, this does not affect the set intersection computation. To compute the intersection between u and v_i, we merely calculate the dot product of $\boldsymbol{u}$ and $\boldsymbol{v}_i$, i.e., $|u \cap v_i| = \boldsymbol{u} \bullet \boldsymbol{v}_i$. □

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																		
$\boldsymbol{u}$	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$\boldsymbol{v}_1$	1	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0
$\boldsymbol{v}_2$	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	1
$\boldsymbol{v}_3$	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Table 2. Global n-gram space

The above example shows that to securely compute $|u \cap v_i|$, we can use a secure dot product protocol that computes $\boldsymbol{u} \bullet \boldsymbol{v}_i$. We will adopt the homomorphic encryption based secure dot product protocol given in [2].

2.2 Stage 1 - Securely Computing a_{1i} and a_{2i}

Let E_{pk} and D_{pr} be the encryption and decryption functions in Paillier's public-key homomorphic encryption system [9]. The encryption function has the following additive homomorphic property: $E_{pk}(x_1) * E_{pk}(x_2) = E_{pk}(x_1 + x_2)$.

Given two documents $\boldsymbol{u}$ and $\boldsymbol{v}_i$, represented under the global n-gram model, the main steps computing a_{1i} and a_{2i} is given in Algorithm 1. The algorithm takes $\boldsymbol{u}$ from Alice and $\boldsymbol{v}_i$ from Bob's document collection, and returns a_{1i} to Alice and a_{2i} to Bob, such that $a_{1i} + a_{2i} \mod N = \boldsymbol{u} \bullet \boldsymbol{v}_i$. During this process, Alice does not know anything that is computationally feasible about Bob's input vector, and vice versa. At the beginning, Alice encrypts her private input vector component-wise and sends the encrypted vector to Bob denoted as $\boldsymbol{z}$. Upon receiving $\boldsymbol{z}$, At step 2(b), Bob computes $E_{pk}(\boldsymbol{u} \bullet \boldsymbol{v}_i)$, the encryption of the dot product of $\boldsymbol{u}$ and $\boldsymbol{v}_i$, denoted by s. At step 2(c), Bob computes $E_{pk}(\boldsymbol{u} \bullet \boldsymbol{v}_i + r)$. At step 2(d), Bob computes his share a_{2i} and sends $E_{pk}(\boldsymbol{u} \bullet \boldsymbol{v}_i + r)$ to Alice. Alice can get her share a_{1i} by decrypting the value received from Bob. Note that Alice's private vector $\boldsymbol{u}$ is not disclosed to Bob because only the encrypted vector is sent to Bob. No one (except Alice) can decrypt the vector to get the actual values regarding $\boldsymbol{u}$. In addition, because $a_{1i} = \boldsymbol{u} \bullet \boldsymbol{v}_i + r$ and $a_{2i} = N - r$, it is certain that $a_{1i} + a_{2i} \mod N = \boldsymbol{u} \bullet \boldsymbol{v}_i$.

Algorithm 1 Computing_Random_Shares$(\boldsymbol{u}, \boldsymbol{v}_i) \rightarrow (a_{1i}, a_{2i})$

Require: Alice's input: Query Document $\boldsymbol{u}$ and (g, N); Bob's input: $\boldsymbol{v}_i$ and (g, N); $|\boldsymbol{u}| = |\boldsymbol{v}_i| = |S|$ (Note: the private key pr is only known to Alice)

1: Alice:

(a). Encrypt $\boldsymbol{u}$ component-wise: $\boldsymbol{z}[j] \leftarrow E_{pk}(\boldsymbol{u}[j])$ for $j = 1, \ldots, |S|$
(b). Send $\boldsymbol{z}$ to Bob

2: Bob:

(a). Receive $\boldsymbol{z}$ from Alice
(b). $s \leftarrow \prod_{j=1 \wedge \boldsymbol{v}_i[j]=1}^{|S|} \boldsymbol{z}[j]$
(c). $s \leftarrow s * E_{pk}(r)$, where r is randomly chosen from $\mathbb{Z}_N^*$
(d). $a_{2i} \leftarrow N - r$
(e). Send s to Alice

3: Alice:

(a). Receive s from Bob
(b). $a_{1i} \leftarrow D_{pr}(s)$

The protocol in Algorithm 1 is secure in the semi-honest model. The main reason is that the messages communicated during the execution of the protocol are random shares, which can be simulated based on the input and the output.

Complexity Analysis The complexity of Computing_Random_Shares consists of local computation cost and communication cost between the two parties. Since the protocol is asymmetric, the local computation cost is different for each party. The main computation cost for Alice is encrypting her input vector, and we calculate her computation complexity based on the number of encryptions she performs. According to Algorithm 1, Alice's computation cost is determined by step 1(a) where the number of encryptions performed is linearly bounded by the size of her input vector. Therefore, the computation complexity for Alice is bounded by $O(|S|)$ number of encryptions.

Step 2(b) of Algorithm 1 determines Bob's computation cost, and it requires at most $|S|$ multiplications of ciphertexts received from Alice. In general, one encryption operation (or exponentiation) under the Paillier's system is much more expensive than one multiplication (depends on the size of encryption key). Here, we assume that the time it takes to perform one encryption at step 2(c) is less than $|S|$ multiplications performed at step 2(b). Then, the computation complexity for Bob is bounded by $O(|S|)$ number of multiplications.

Let k denote the size of encryption key in number of bits. (In practice, the encryption key N should be at lest 1,024-bit long.) The communication complexity of the protocol is bounded by step 1(b). Because the size of each ciphertext is bounded by k, the communication complexity is given by $O(k \cdot |S|)$ in bits.

2.3 Stage 2 - Securely Computing JC Scores

Once Alice and Bob obtain a_{1i} and a_{2i} respectively, Alice sets $b_{1i} = |u| - a_{1i} \mod N$ and Bob sets $b_{2i} = |v_i| - a_{2i} \mod N$. Alice and Bob can securely compute $\frac{a_{1i}+a_{2i} \mod N}{b_{1i}+b_{2i} \mod N}$, without disclosing a_{1i}, b_{1i} to Bob and a_{2i}, b_{2i} to Alice. Suppose Bob's collection contains n documents, at the end, Alice and Bob needs to compute n JC scores, one for each pair of u and v_i. Algorithm 2 lists the main steps to achieve this. The inputs to the algorithm are α_1 and α_2, from Alice and Bob respectively. α_1, privately owned by Alice, contains n random shares generated from stage 1. Similarly, α_2, privately owned by Bob, contains n random shares generated from stage 1. Initially, Alice and Bob independently compute β_1 and β_2, so for $1 \leq i \leq n$, JC score of u and v_i is given by $\sigma_i = \frac{a_{1i}+a_{2i} \mod N}{b_{1i}+b_{2i} \mod N} = \frac{|u \cap v_i|}{|u \cup v_i|}$. To securely compute σ_i (without disclosing α_1 to Bob and α_2 to Alice), step 3 of Algorithm 2 adopts the appropriate Secure_Division protocol proposed in [1]. The protocol takes private input (a_{1i}, b_{1i}) from Alice and private input (a_{2i}, b_{2i}) from Bob. It returns $\sigma_i = \frac{a_{1i}+a_{2i} \mod N}{b_{1i}+b_{2i} \mod N}$ to either or both parties.

Algorithm 2 Computing_JC_Scores$(\alpha_1, \alpha_2) \rightarrow \sigma_1, \ldots, \sigma_n$

Require: Alice's input: $\alpha_1 = \langle a_{11}, \ldots, a_{1n} \rangle$; Bob's input: $\alpha_2 = \langle a_{21}, \ldots, a_{2n} \rangle$, and $|\alpha_1| = |\alpha_2| = n$

1: Alice: Compute $\beta_1 = \langle b_{11}, \ldots, b_{1n} \rangle$, where $b_{1i} = |u| - a_{1i} \mod N$ for $1 \leq i \leq n$
2: Bob: Compute $\beta_2 = \langle b_{21}, \ldots, b_{2n} \rangle$, where $b_{2i} = |v_i| - a_{2i} \mod N$ for $1 \leq i \leq n$
3: Alice (with input α_1, β_1) and Bob (with input α_2, β_2): For $1 \leq i \leq n$, compute $\sigma_i \leftarrow$ Secure_Division$((a_{1i}, b_{1i}), (a_{2i}, b_{2i}))$

Complexity Analysis The secure dot product protocol can be easily implemented using Paillier's crypto system as suggested in [2]. Without giving further details, the computation cost of Secure_Division for both party is bounded by O(1) number of encryptions. The communication cost of Secure_Division is bounded by O(k) bits, where k is the size of an encryption key in bits. Computing_JC_Scores requires n executions of Secure_Division, so its computation complexity is bounded by O(n) encryptions for each party, and the communication complexity is bounded by O($k \cdot n$) bits.

2.4 N-Gram based SSDD

Once we know how to securely implement Stage 1 and Stage 2, we can combine the two stages together to derive a n-gram based SSDD protocol. The main steps of the protocol are highlighted in Algorithm 3. Initially, Bob generates the global n-gram space S from his document collection $\mathcal{D}$. Then a one-to-one mapping $M(S)$ is created to map S to $\{1, \ldots, |S|\}$. At step 1(c) of Algorithm 3, Bob, according to S and $M(S)$, computes the vector representation of each

document in his document collection $\mathcal{D}$ under the n-gram model. Once Alice receives S and $M(S)$, Alice generates $\boldsymbol{u}$, the vector representation of her query document u under the n-gram model. After that Alice and Bob can proceed to Stage 1 and Stage 2. At the end of stage 2, Alice receives a set of similarity scores, or these scores can be shared by both parties. The security of the protocol is determined by the Secure_Division protocol adopted at stage 2.

Algorithm 3 NGram_SSDD$(u, \mathcal{D}) \rightarrow \sigma_1, \ldots, \sigma_n$

Require: Alice's input: u; Bob's input: $\mathcal{D}$

1: Bob:

(a). Compute S and $M(S)$ from $\mathcal{D}$
(b). Compute $\boldsymbol{v}_i$ from $\mathcal{D}, S, M(S)$, for $1 \leq i \leq |\mathcal{D}|$
(c). Send $S, M(S)$ to Alice

2: Alice: Compute $\boldsymbol{u}$ from $u, S, M(S)$
3: Alice and Bob:

(a). **Stage 1** - Computing_Random_Shares$(\boldsymbol{u}, \boldsymbol{v}_i)$, for $1 \leq i \leq |\mathcal{D}|$ (At the end, Alice obtains $\alpha_1 = \langle a_{11}, \ldots, a_{1n} \rangle$, and Bob obtains $\alpha_2 = \langle a_{21}, \ldots, a_{2n} \rangle$)
(b). **Stage 2** - Computing_JC_Scores(α_1, α_2)

Complexity Analysis The main cost of the protocol occurs at step 3 of Algorithm 3. Stage 1 initiates n executions of Computing_Random_Shares protocol. From Alice's point of view, based on the computation complexity analysis of Computing_Random_Shares, Alice needs to encrypt her query document $\boldsymbol{u}$ n times. However, during the actual implementation, Alice only needs to encrypt $\boldsymbol{u}$ once, and uses the same encrypted $\boldsymbol{u}$ for each of the n instantiations of the Computing_Random_Shares protocol. As a result, the computation cost for Alice at stage 1 is $O(|S|)$ number of encryptions. Since each instantiation of Computing_Random_Shares requires Bob to perform $O(|S|)$ multiplications, the computation cost for Bob at stage 1 is $O(n \cdot |S|)$ multiplications. Alice needs to send the encrypted $\boldsymbol{u}$ once, so the communication complexity for stage 1 is $O(k \cdot |S|)$ bits where k is the encryption key size in Paillier's system.

Stage 2 only calls the Computing_JC_Scores once; thus, its complexity analysis is the same as that of Computing_JC_Scores. Combining the two stages together, the computation complexity of NGram_SSDD for Alice is bounded by $O(|S|+n)$ number of encryptions. The computation complexity of NGram_SSDD for Bob is bounded by $O(n \cdot |S|)$ number of multiplications plus $O(n)$ number of encryptions. The total communication complexity of NGram_SSDD is bounded by $O(k \cdot |S| + k \cdot n)$ bits.

3 Future Work

We will empirically study the performance of the proposed protocol. To further improve the efficiency and instead of fixing the global n-gram space based on Bob's entire document collection, we could apply winnowing techniques [10] on each Bob's document and select representative n-gram (or fingerprints) for each document. Since winnowing selects few fingerprints by adjusting window size and noise threshold, it is possible to reduce the global space to a larger extent thereby reducing the computational cost.

Acknowledgment

This material is based upon work supported by the Office of Naval Research under Award No. N000141110256.

References

1. M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, pages 103–114, October 2004.
2. B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen. On secure scalar product computation for privacy-preserving data mining. In C. Park and S. Chee, editors, *The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004)*, pages 104–120, Seoul, Korea, Dec. 2-3 2004.
3. O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.
4. O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter Encryption Schemes. Cambridge University Press, 2004.
5. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, Rhode Island, U.S.A., May 6-8 1985.
6. W. Jiang, M. Murugesan, C. Clifton, and L. Si. Similar document detection with limited information disclosure. In *Proceedings of the 24th International Conference on Data Engineering (ICDE 2008)*, Cancun, Mexico, Apr. 7-12 2008.
7. U. Manber. Finding similar files in a large file system. Technical Report TR 93-33, Department of Computer Science, The University of Arizona, Tucson, Arizona, Oct. 1993.
8. M. Murugesan, W. Jiang, C. Clifton, L. Si, and J. Vaidya. Efficient privacy-preserving similar document detection. *The VLDB Journal*, January 16 2010.
9. P. Paillier. Public key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - Eurocrypt '99 Proceedings, LNCS 1592*, pages 223–238, Prague, Czech Republic, May 2-6 1999. Springer-Verlag.
10. S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 76–85, San Diego, California, United States, June 9-12 2003. ACM.