

Low-Complexity and Secure Clustering-Based Similarity Detection for Private Files

Duaa Fadhel Najem¹, Nagham Abdulrasool Taha², Zaid Ameen Abduljabbar^{2,3,4},
Vincent Omollo Nyangaresi^{5,6}, Junchao Ma³, Dhafer G. Honi^{2,7}

¹ Department of Cyber Security, College of Computer Science and Information Technology,
University of Basrah, Basrah 61004, Iraq

² Department of Computer Science, College of Education for Pure Sciences,
University of Basrah, Basrah, 61004, Iraq

³ College of Big Data and Internet, Shenzhen Technology University, Shenzhen, 518118, China

⁴ Shenzhen Institute, Huazhong University of Science and Technology, Shenzhen 518000, China

⁵ Department of Computer Science and Software Engineering, Jaramogi Oginga Odinga
University of Science & Technology, Bondo 40601, Kenya;

⁶ Department of Applied Electronics, Saveetha School of Engineering, SIMATS,
Chennai, Tamil Nadu 600124, India

⁷ Department of IT, University of Debrecen, Debrecen, 4002, Hungary

Abstract – Detection of the similarity between files is a requirement for many practical applications, such as copyright protection, file management, plagiarism detection, and detecting duplicate submissions of scientific articles to multiple journals or conferences. Existing methods have not taken into consideration file privacy, which prevents their use in many delicate situations, for example when comparing two intellectual agencies' files where files are meant to be secured, to find file similarities. Over the last few years, encryption protocols have been developed with the aim of detecting similar files without compromising privacy. However, existing protocols tend to leak important data, and do not have low complexity costs. This paper addresses the issue of computing the similarity between two file collections belonging to two entities who desire to keep their contents private.

We propose a clustering-based approach that achieves 90% accuracy while significantly reducing the execution time. The protocols presented in this study are much more efficient than other secure protocols, and the alternatives are slower in terms of similarity detection for large file sets. Our system achieves a high level of security by using a vector space model to convert the files into vectors and by applying Paillier encryption to encrypt the elements of the vector separately, to protect privacy. The study uses the application of the Porter algorithm to the vocabulary set. Using a secure cosine similarity approach, a score for similar files was identified and the index of the similarity scores is returned to the other party, rather than the similar files themselves. The system is strengthened by using clustering for files, based on the k-means clustering technique, which makes it more efficient for large file sets.

DOI: 10.18421/TEM133-61

<https://doi.org/10.18421/TEM133-61>


Corresponding author: Zaid Ameen Abduljabbar,
Department of Computer Science, College of Education for
Pure Sciences, University of Basrah, Basrah, 61004, Iraq
Email: zaid.ameen@uobasrah.edu.iq

Received: 23 January 2024.

Revised: 11 May 2024.

Accepted: 18 May 2024.

Published: 27 August 2024.

 © 2024 Duaa Fadhel Najem et al;
published by UIKTEN. This work is licensed under the
Creative Commons Attribution-NonCommercial-NoDerivs
4.0 License.

The article is published with Open Access at
<https://www.temjournal.com/>

Keywords – File similarity, privacy, similarity
detection.

1. Introduction

File similarity detection techniques have begun to be used in many important applications since the first research in this field began in 1993 [1]. For example, this approach is used in a file management system, which can work more efficiently if similar files are identified. It is also used to improve the function of web crawlers in terms of detecting similar pages [2], [3], [4]. Finally, this method is used in applications related to plagiarism detection and copyright protection [5], [6].

The problem of security is considered very important in the process of data matching.

This scheme was not secure, as one party revealed all of the 3-grams to the other party. This can be considered a security weakness.

In [13], the authors suggested an efficient approach to evaluate the set similarity. The JS was used to measure the similarity between two private sets. Specifically, two approaches were applied to compute the similarity of the sets: the first was to compute the exact secure JS, while the second used the MinHash technique to reduce the computation and communication overheads. The PSI-CA protocol [19] was used in both approaches to specify the common 3-gram sets. The work in [20] used the MinHash technique, an efficient method of detecting the similarity of files in a secure manner. The authors of [20] used the SJCM protocol, where the frequency of each N-gram is computed using the JS during secure computation. Blundo *et al.* [16] used the secure algorithm presented by De Cristafaro *et al.* [13] to detect the secure similarity between two sets. More recently, in [22], Schoppmann *et al.* introduced a secure system for documents using classification (K-NN).

However, all of the systems described above detect the actual similarity scores for the other side. In contrast, our scheme encrypts the data and only sends back the index of the matching file to the other party. Furthermore, the scheme has low complexity due to the use of the k-means clustering technique, where the files are grouped into clusters, meaning that only the similarity scores for the n_k files inside the closest cluster are computed when the first party wants to inquire about a file.

3. Cryptographic Background

The following part provides a brief explanation of the basic tools that are used in this paper.

3.1. Homomorphic Encryption

Homomorphic encryption occupies the largest position among encryption systems in order to secure data and maintain its privacy. This approach has many valuable properties [22], [24]. Through homomorphic encryption, any mathematical operation on encrypted texts can be performed without the need to know the private key. In 2009, Gentry [25] was the first author to design a fully homomorphic scheme that supported multiplication operations, and many authors later improved on this technique [26], [27], [28]. The study employs an additive homomorphic encryption process for the scheme. Additive homomorphic encryption has the following properties:

$$1- D_{pr}(E_{pk}(x).E_{pk}(y)) = x + y.$$

where x and y are given integers, (pr, pk) are the public key pairs, $D_{pr}()$ is the decryption algorithm, and $E_{pk}()$ is the encryption algorithm.

$$2- E_{pk}(x)^c = x \cdot c$$

for any positive integer c and x in the message space.

$$3- \text{The multiplicative inverse } E_{pk}(y) = E_{pk}(y)^{-1} \text{ i.}$$

$$D_{pr}(E_{pk}(x).E_{pk}(y)^{-1}) = x - y$$

3.2. Paillier Cryptosystem

In 1999, Pascal Paillier [29] invented a strong public-key cryptosystem that supports additive homomorphic and multiplication homomorphic functions. Paillier's cryptosystem is semantically secure. The authors used the algorithms of this cryptosystem in the scheme which is explained in detail in [29].

3.3. DGK Encryption System

There are many protocols that can be used for comparing encrypted numbers, and these have been applied in numerous fields, including secure classification [30]. Veugen [31] created a protocol that is considered efficient for comparing two encrypted integers $[[a]]$, $[[b]]$ s.t. $[[0]] < [[a]]$ and $[[b]] \leq [[2^\ell]]$ without the need for decryption. The DGK encryption system compares two private numbers while preserving privacy [32], and is additively homomorphic used in our scheme.

The main idea underlying our scheme is to calculate $[[z]] = [[2^\ell]] - [[a]] + [[b]]$ for the $(\ell + 1)$ -bit and determine its most significant bit z_ℓ . $[[a]] \geq [[b]]$ if it is 1, and $[[a]] < [[b]]$ otherwise. Thus, to determine whether or not $[[a]] \geq [[b]]$, all that is needed is to compute the bit z_ℓ .

In Veugen's protocol [31], there are two parties: Alice inputs an encrypted number $[[a]]$, and Bob inputs an encrypted number $[[b]]$, where $[[\delta]] = ([[a]] < [[b]]) = [[1 - 2^\ell]]$ is the result. The power of this protocol lies in its ability to prevent the other party from knowing the true values of a , b , and the comparison bit δ . In [31], Veugens outlines the fundamental steps of this protocol.

4. Secure Comparison Scenario

The following problem is solved in this paper. Bob and Alice both wish to use a secure comparison to find similarities in their text files. Alice's private file is u , while Bob's input is F .

Bob:

$\forall j = 1, \dots, m$: For each file $f_j \in F$

- Vector ω_j of size n .

Alice:

- $[\omega'_j] \leftarrow E_{pk}(\omega'_j), \forall j = 1, \dots, n$
- $[\sum_{j=1}^n \omega'^2_j]$
- $[\omega'_j], j = 1, \dots, n$ (Transmit to Bob)
- $[\sum_{j=1}^n \omega'^2_j]$ (Transmit to Bob)
- Set $inx \leftarrow 1$

Bob:

- Secure dot product $[F_j] \forall j = 1, \dots, m$: // As shown in Equations 1 and 2.

Bob:

- Select a random permutation ϵ over $\{1, \dots, m\}$
- Set $[maxv] \leftarrow [th]$
- $[F] = [0]$
- $\forall i = 1, \dots, m$:
 - **Bob:** Secure comparison of encrypted integers by a protocol as explained by Veugen [25] for $([maxv], [F_{\epsilon(i)}])$. The output is δ comparison bit.
 - **Alice:** Send $[\delta]$ to Bob
 - **Bob:** $[F] = [F] \cdot [\delta]^{2^{\epsilon(i)}}$.
- **Bob:** Send $[F]$ to Alice.
- **Alice:** Decrypt $[F]$; if all locations of binary form are one then the files are similar.

Protocol 1 is effective in terms of comparison, but it has a long computation time, as each file is compared with all the others (for a secure comparison). In other words, we compare each file with the whole database. For each comparison, we need a secure dot product, so to speed up the protocol; we can use a representative for each cluster of files. The comparison will then only involve the representatives, and a few clusters can be chosen for pairwise comparisons. To explain the protocol further, the comparison process takes place for each file with another file, and thus you need the computation of the secure cosine.

The proposed work is built to use a representative for each cluster of files, so only the representatives are compared for pairwise comparisons between a few clusters. In this scenario, the execution time will depend on the number of representatives.

The best way to select representatives without losing too much accuracy is by clustering the file collection into k (number of representatives).

Protocol 2 as seen in the Figure 1 that follows outlines the phases of the *SSFD* protocol, which is based on each party's clustering file dataset. In the first and second steps, the two parties must work for their files as a cluster into k clusters. In this protocol, assume that both sides create clusters of equal numbers. The findings for the k -means clustering technique are provided.

The distance function by these clustering algorithms with the conventional cosine similarity between the frequency vectors is employed. In addition, the mean frequency vector for the files in the cluster is chosen to determine the cluster centres. For n files and k representatives, $n - k$ merges have applied, which is accomplished by indiscriminately selecting the files that are the closest to one another for merging. The files are represented by the centre of the cluster when the two nearest files have been combined into a single cluster.

In Steps 1(b) and 2(b), the two parties create the representative vectors for the k clusters. In Step 3(a), a comparison is made between the representatives for the first party's files and the other party's representative vectors. When the similarity score (σ_i, j) is greater than the similarity threshold σ_{th} between the i -th and j -th clusters for Alice and Bob, files in clusters Alice A_i and Bob A_j are using a protocol 1 (*SSFD*) for securely compared. The clustering model reduces the number of computations involving the secure dot product, as not all of the files are compared. The accuracy remains intact if there are exactly as many files as there are representatives.

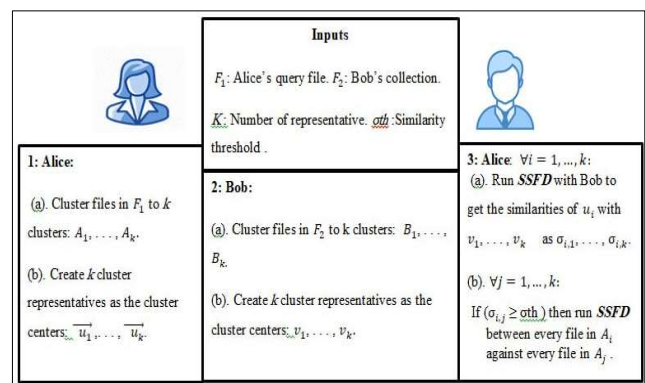


Figure 1. Protocol 2 (clustering-based *SSFD*)

5.1. Security Analysis

Our work is designed as a privacy-preserving computation (secure two-party computation) under the semi-honest model.

Table 2. Execution time pre-computation

Execution time (sec.)				
Size of file collection (m)	File vector encryption	Secure dot product computation	Finding similar files	Total running time
100	0.72	5.715	16.824	7.753
300	0.74	14.757	81.25	32.249
600	0.78	40.855	134.963	58.866
800	0.7	62.616	193.741	85.685
990	0.759	82.73	252.998	112.162

The sizes of the file collection m and the file vector n determine the computational cost for the secure dot product operation used in our system. The execution times for the secure similarity function, which increases linearly with the size of the file collection, are shown in Table 3. The execution time becomes longer when the larger vector's collection size is fixed. This is because clustering is not applied to the files, and thus Protocol 1 is slow for large datasets, despite its efficiency, as explained earlier.

Table 3. Execution times for the secure dot product operation

Duration (s)				
Size of file collection (m)	Query length (n)			
	100	200	300	400
200	8.685	9.023	9.847	9.185
500	25.941	28.249	36.952	37.01
800	44.228	45.105	45.564	47.747
990	62.981	64.522	69.099	69.11

To increase security, the size of the key in all of the experiments was 1024. Table 4 shows the variation in the execution time with an increase in the key size.

Table 4. Paillier key with different sizes k

Execution times (s)				
Paillier key size (bits)	File vector encryption	Secure calculation of dot products	Finding similar files	Total running time
128	0.026	3.093	39.892	14.337
256	0.042	6.677	54.007	20.242
512	0.107	20.979	97.938	39.674
1024	0.655	64.068	187.271	83.998

6.2. Effectiveness with Clustering Technique

In this stage of the experiments, the authors applied Protocol 2 to 900 files randomly selected from the original set of 990 files. This was the same dataset that was used in 20news collection [18].

From these 900 files, two collections are produced, each containing 495 files, and found that there were 150 files that were the same in both collections. As a result, each collection had 345 unique files and 150 perfectly identical files (with a cosine similarity score of 1.0). In total, if the cosine score was equal to or greater than 0.80, there were 210 unique pairs of files. This stage treats these 210 files as similar files.

Through this research, it is clear that the proposed clustering technique works more quickly to identify the matches, as fewer comparisons are made. This experiment calculates how long the k -means clustering procedure will take on the given file collection. The running times for the k -mean clustering algorithm are displayed in Table 5. Through this experiment, the number of clusters from 100 to 500 is varied. As shown in the table below, the execution times for the clustering step were much lower than for Protocol 1 SSFD.

Table 5. Running times for clustering of 500 files

Number of clusters	Running time (s)
100	45.18
200	62.1
300	65.81
400	69.31
500	110.5

Table 6 shows that the accuracy and the effect of the number of representations on it (Protocol 2 explains that in the Step 3(b) the precision does not reduce). The percentage of matches detected generally remained above 70% when using values from 0.5 to 0.9 for the threshold. As the threshold for similarity increases, some similar files are excluded, resulting in a percentage of similar files detected that is less than 100%.

Table 6. The accuracy and the effect of the number of representations

Similarity threshold	0.5	0.6	0.7	0.8	0.9	1
Number of representative files	% of matches found					
100	94	90	75	75	23	2
200	95	89	80	55	15	5
300	94	94	75	58	15	5
400	95	95	85	65	18	12
500	99	95	90	75	50	35

The k -means clustering method reduces the number of file comparisons (by reducing the number of computations of secure dot products).

- [7]. Unger, N., Thandra, S., & Goldberg, I. (2016). Elxa: Scalable privacy-preserving plagiarism detection. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 153-164.
- [8]. Al Sibabee, M.A., Lu, S., Abduljabbar, Z.A., et al. (2018). Efficient encrypted image retrieval in IoT-cloud with multi-user authentication. *International Journal of Distributed Sensor Networks*, 14(2). Doi: 10.1177/1550147718761814
- [9]. Abduljabbar, Z.A., Jin, H., Ibrahim, A., Hussien, Z.A., Hussain, M.A., Abbdal, S.H., & Zou, D. (2016). SEPIM: Secure and Efficient Private Image Matching. *Appl. Sci.*, 6, 213. Doi: 10.3390/app6080213
- [10]. Jiang, W., Murugesan, M., Clifton, C., & Si, L. (2008, April). Similar document detection with limited information disclosure. In *2008 IEEE 24th International Conference on Data Engineering*, 735-743. IEEE.
- [11]. Murugesan, M., Jiang, W., Clifton, C., Si, L., & Vaidya, J. (2010). Efficient privacy-preserving similar document detection. *The VLDB Journal*, 19(4), 457-475.
- [12]. Jiang, W., & Samanthula, B. K. (2011). N-gram based secure similar document detection. In *Data and Applications Security and Privacy XXV: 25th Annual IFIP WG 11.3 Conference, DBSec 2011, Richmond, VA, USA, July 11-13, 2011. Proceedings 25*, 239-246. Springer Berlin Heidelberg.
- [13]. Blundo, C., De Cristofaro, E., & Gasti, P. (2012). EsPRESSo: efficient privacy-preserving evaluation of sample set similarity. In *International Workshop on Data Privacy Management*, 89-103. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [14]. Manning, C.D., et al. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- [15]. Wang, C., Cao, N., Li, J., Ren, K., & Lou, W. (2010). Secure ranked keyword search over encrypted cloud data. In *2010 IEEE 30th international conference on distributed computing systems*, 253-262. IEEE.
- [16]. Samanthula, B. K., & Jiang, W. (2015). Secure multiset intersection cardinality and its application to jaccard coefficient. *IEEE Transactions on Dependable and Secure Computing*, 13(5), 591-604. Doi: 10.1109/TDSC.2015.2415482.
- [17]. Liao, W. (2013). Parallel k-Means Data Clustering. Retrieved from: <http://www.ece.northwestern.edu/wkliao/Kmeans/index.html> [accessed: 5 December 2013].
- [18]. Porter, M. (2006). An algorithm for suffix stripping. *Program: electronic library and information systems*, 40(3), 211-218. Doi: 10.1108/00330330610681286.
- [19]. De Cristofaro, E., Gasti, P., & Tsudik, G. (2012). Fast and private computation of cardinality of set intersection and union. In *International Conference on Cryptology and Network Security*, 218-231. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [20]. Yu, X., Chen, X., Shi, J., Shen, L., & Wang, D. (2017). Efficient and scalable privacy-preserving similar document detection. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 1-7. IEEE.
- [21]. Bost, R., Popa, R., Tu, S., & Goldwasser, S. (2015). Machine learning classification over encrypted data. In *The Network and Distributed System Security Symposium*, 1-14.
- [22]. Schoppmann, P., Vogelsang, L., Gascón, A., & Balle, B. (2020). Secure and Scalable Document Similarity on Distributed Databases: Differential Privacy to the Rescue. *Proceedings on Privacy Enhancing Technologies 2020*, 2020(2), 209-229. Doi: 10.2478/popets-2020-0024.
- [23]. UCI Machine Learning Repository. (n.d.). *Twenty Newsgroups dataset*. UCI Machine Learning Repository. Retrieved from: <https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups> [accessed: 02 January 2024].
- [24]. Ramaiah, Y. G., & Kumari, G. V. (2012). Efficient public key homomorphic encryption over integer plaintexts. In *2012 International Conference on Information Security and Intelligent Contr*, 123-128. IEEE.
- [25]. Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 169-178.
- [26]. Chillotti, I., Gama, N., Georgieva, M., & Izabachene, M. (2016). Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International conference on the theory and application of cryptology and information security*, 3-33.
- [27]. Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory*, 6(3), 1-36. Doi: 10.1145/2090236.2090262.
- [28]. Martins, P., Sousa, L., & Mariano, A. (2017). A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys*, 50(6), 1-33. Doi: 10.1145/3124441.
- [29]. Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, 223-238. Berlin, Heidelberg: Springer Berlin Heidelberg.
- [30]. Li, P., Li, T., Yao, Z. A., Tang, C. M., & Li, J. (2017). Privacy-preserving outsourcing of image feature extraction in cloud computing. *Soft Computing*, 21, 4349-4359. Doi: 10.1007/s00500-016-2066-5.
- [31]. Veugen, T. (2012). Improving the DGK comparison protocol. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, 49-54.
- [32]. Damgård, I., Geisler, M., & Kroigaard, M. (2007). Efficient and secure comparison for on-line auctions. In *Information Security and Privacy: 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Proceedings 12*, 416-430. Springer Berlin Heidelberg.
- [33]. Lindell, Y., & Pinkas, B. (2009). Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1), 59-98. Doi: 10.29012/jpc.v1i1.566.