Investigation Framework for Cloud Forensics using Dynamic Genetic-based Clustering

Mohammed Y. Alkhanafseh[1]
Department of Computer Science
Birzeit University
Ramallah, Palestine

Mohammad Qatawneh[2]
Department of Computer Science, KASIT
The University of Jordan
Amman, Jordan

Wesam Almobaideen[3]
EE and Computing Sciences Departmen
Rochester Institute of Technology-Dubai
Dubai, UAE

***Abstract*—Cloud computing allows a pool of resources, such as storage, computation power, communication bandwidth, to be shared and accessed by many users from different locations. High dependency on sharing resources among different cloud users allows some attacker to hide and commit crimes using cloud resources and as a result, cloud computing forensics become essential. Many solutions and frameworks for cloud computing forensics have been developed to deal with could based crimes. However, many problems and issues face the proposed solutions and frameworks. In this paper, a new framework for cloud computing forensics is proposed to enhance the investigation process performance and accuracy by adding a new stage to conventional stages. This new stage includes the implementation of a new way for matching based on the LSH algorithm. The proposed framework evaluation results show an improvement for matching and accurate cluster retrieval through the collection process.**

***Keywords*—Cloud computing forensics; genetic clustering algorithms; genetic dynamic clustering; forensics framework; digital forensics**

I. Introduction

Digital forensics is a science which is defined by many researchers, see [31][20], as a science of retrieving, examining, and analysing digital evidences collected from digital devices. A Digital evidence, according to [20], is defined as the information and data of investigative value that is stored on, received, or transmitted by a digital device. The goal of digital forensics is to detect and extract the evidences, analyze the collected evidences and preserve related ones in a format that can be presented in a court [24][18].

Cloud computing is one of the most popular technologies since different users and companies around the world are rapidly becoming more dependent on cloud computing. This means that millions or even billions of files have been uploaded to cloud computing resources. Security of resources over the cloud is a challenging issue that has to be addressed [43] [3]. Fog computing is similar to cloud computing but closer to the user which also has many security concerns [2]

Cloud computing forensics refers to one of the digital forensics branches This branch has gained popularity from the importance of today's cloud computing. Many researchers describe cloud computing forensics as an application for digital forensics for cloud computing resources to investigate the crimes that occurred in cloud computing resources. Text files and textual information represent a very high percentage of types of evidence, particularly in cloud computing [20]. Accordingly, and due to the extra high volume of data stored on the cloud, an enhanced solution is needed to handle textual evidence by extracting and encouraging the handling of textual evidence.

In recent years, the number of digital crimes that involve internet and computer has grown, which has encouraged a lot of companies to include measures in their products to assist law enforcement in using digital evidences to determine the perpetrators, methods used, timing, and victims of computer crimes. The process of applying digital forensics in the huge volume of files which stored in cloud computing resources is very hard. Therefore, the idea of clustering has been applied to facilitate the investigation process. Clustering is the process of grouping objects or documents related to each other in the same group. This is important in the investigation process, since the investigator needs to search for all documents in the storage space to define if there are any documents related to the original file [17] .

This paper presents a new approach to cloud computing forensics, which solves some of existing problems and challenges. The proposed solution consists of two main parts. The contribution of this paper can be summarized in the following points:

- A pre-investigation stage has been added to the standard forensics stages as part of the propose Cluster Based Investigation Framework (CBIF). Through this stage, a new hierarchical clustering algorithm capable of handling all types of evidence is added.
- Genetic Based Dynamic Clustering Algorithm (GBDCA) has been developed to divide a dataset into suitable number of clusters in cloud storage.
- A new search and matching technique based on using a locality-sensitive hashing algorithm has (LSH) to enhance the accuracy of the matching process.
- Additionally, the proposed idea can match parts of the files based on the concept of hierarchical matching and examining process.

This paper is organized as follows. Section 2 brings forth the related work. Section 3 presents a brief description of cloud computing and cloud forensics. Section 4 illustrates the proposed idea and foremost step of the proposed solution. Section 5 details the experimental implementation and results from the discussion. We conclude with Section 6.

II. Related Work

In the field of digital forensics, some frameworks have been suggested as a general frameworks while others were proposed for lower levels of digital forensics branches. Example of frameworks that are suggested as generic framework for digital forensics is an integrated digital forensics process model [41], which is a standard framework for digital forensics and can apply to all branches of digital forensics. This framework contains all standard stages of digital forensics with some changes in the level of combining evidence collection and the preservation stage into a single stage to make sure the identified evidence does not change through the collection stage.

Another model proposed as a generic framework is the integrated framework for digital forensics [48], where the proposed model is a standard model that helps the investigators to follow a uniform approach in digital forensics. The model contains all conventional stages of digital forensics. This framework is classified as a simple framework based on [31]. And other models that fall under general framework such as the model suggested through [40], [8], [32], and [7].

On the level of the computer forensics branch, many frameworks were suggested, such as the computer forensics investigation approach to evidence in cyber-crime [13], which aims to define a new approach to solve and enhance the stage of computer forensics examination. The model meets Italian legislation and could probably be used in other countries.

Numerous frameworks and solutions have been proposed in other digital forensics branches [31], [47]. One instance includes the frameworks and solutions proposed in the IoT forensics branch, which refer to one of the digital forensics branches, such as the one in [13]. The idea of the proposed system relates to using Block-Chain through the investigation process to improve the collection stage's security level and credibility.

The model proposed in [21] refers to a new model for reviewing and investigating cyber attacks, where digital

other framework was proposed in [37], which refers to a new framework for mobile forensics, since a lot of crimes was applied on the resources of mobile devices, specially with rapid development of wireless network and smart mobile. The idea that proposed here refers to depends on the clustering process to facilitate the investigation as all.

The model proposed in [6] refers to the first framework, which contains various safety principles proposed by the ISO standard. Another framework for investigating crimes committed in IoT devices is the IoT forensics framework for the smart environment [39]. The proposed model was specified to investigate the crimes committed in a smart environment, where the proposed model is a lightweight model that is well equipped to be compatible with IoT resources. Moreover, this model can classify as a generic model for all IoT crimes with a high level of privacy based on the principles mentioned for this point.

Other frameworks and solutions proposed in the other branches of digital forensics include the framework and solutions proposed to investigate the crimes committed in a cloud computing environment. One of these frameworks is presented in [26], where the main contribution for the proposed model is defining the difference between the evidence collection stage and the preservation stage, since many frameworks are merged between the two stages. In [42], a new framework for cloud computing with a fundamental change in the stages of conventional frameworks is illustrated. The change affects the proposed model of the identification and collection stages, and the rest of the stages did not change. The science of the proposed framework focuses on the first two stages. Additionally, an open could forensics model is also an example of a framework proposed for cloud computing forensics [51]. The framework consists of primary stages of digital forensics with an update on the levels of preservation and collection. The model merges between these two stages in a single stage to make sure that the evidence does not change through the collection stage, and merges between the analysis stage and examination stage in a new stage called the organization stage.

One solution proposed for the cloud forensics process is to secure logging as a service for cloud forensics [50]. The idea of this solution is to introduce secure logging as a service that allows the cloud service provider to store virtual machine logs and to provide access for the investigators while preserving the privacy for the tenant of the cloud. In [35], the goal of the proposed model is to provide as much information about each record, such as when any trigger occurs, as when a new record is added or deleted.

When the cloud computing environment contains multiple tenants for its resources, the privacy and security of evidence are essential. One of the proposed solutions to deal with this problem is presented in [5], where the idea of the proposed solution is to use a third party to check and evaluate data collected by investigators. Another solution to solve the multi-tenant problem is proposed in [4], where the idea depends on upholding the confidentiality and integrity for evidence that used through the investigation process.

Yet an additional issue facing the investigation process in cloud computing is the data gathering challenge, which faces the acquisition stage of the investigation process, because the data is distributed amongst different servers which ultimately decrease the performance of the investigation process. Specific frameworks and solutions have been proposed to deal with this problem, such as the framework proposed by Adams [44], which suggested a new cloud forensics model which consists of a planning stage, on-site survey stage, and acquisition stage. This applies to deal with the process of acquisition without any further intervention.

Numerous frameworks and approaches have been suggested to investigate crimes committed in computer networks. Frameworks that scales for a large-scale environment are called dynamic network forensics and have a specific property for investigation such as the framework presented in [25], which investigates the crimes that occur in network infrastructure. The framework contains the standard stages of any digital forensic framework with additional stages to enhance the investigation process, such as the evidence reduction stage, which can enhance the accuracy of the investigation process. Graphic network forensics have many frameworks are specialized for graphics-based network forensics crimes [49], [28], [38]. Many approaches and solutions have been proposed to

investigate soft-computing crimes specific to network analysis and monitoring. The systems proposed in this field are tailored to the environment that includes many attacks, because this type of network forensics can analyze the data collected and identify relevant attacks [9], [36].

Numerous researchers identified cloud computing forensics [14], as an application for digital forensics in cloud computing as a subset of network forensics. NIST framework [29], defines it as an application of science to the identification, examination, and analysis of data while preserving the integrity of data and maintaining the chain of custody for the data through the investigation process. Any information or data stored or extracted from digital devices can be evidence or part of the evidence; these pieces of evidence are analyzed through the investigation process.

The goal of digital and cloud investigation is to ensure that the collected evidence is admissible in a court of law and to maintain that the chain of custody is essential and documented, and is thus stored throughout the entire investigation process [16]. The chain of custody is the process of recording and storing digital evidence as well as managing historical history to protect the custody chain from any alteration or modification or damage by any unauthorized user, and so, a high level of care is required.

The investigation of crimes occurring in the cloud computing environment is faced with many issues. These issue can be summarized through the Table I as follow

Based on the issues mentioned above, a solution for cloud computing forensics is required, as many of the proposed frameworks do not provide a general solution to the aforementioned problems. The solution suggested in this article aims to avoid all the above mentioned significant issues. The proposed solution can be applied in all branches of digital forensics in addition to cloud computing, such as applying the proposed solution in computer forensics, cloud forensics, network forensics, database forensics, and all others.

TABLE I. KEY ISSUE THAT FACES CLOUD COMPUTING FORENSICS PROCESS.

Key Issue Description	The Reason of Key Issue
Searching for Candidate Files	Cloud computing contains a considerable amount of massively distributed information from different users, which makes the searching for evidence a laborious process
Privacy of users documents	Cloud computing contains a considerable amount of massively distributed information from different users, which makes the searching for evidence a laborious process.
Time Needs for locating the target file	Searching for the target file or any of its pieces is time-consuming, based on the massive amount of files in the data center. On that basis, a new matching technique is required to improve the efficiency and accuracy of the matching process for evidence.
The Integrity of users data	In conventional ways, the investigator must check all files to check whether or not they relate to the target file, which may violate all data in the storage center if the investigator is a criminal. A technique is required to check only those files that are related to the target file.

III. PROPOSED IDEA

The idea of the proposed solution focuses on resolving shortcomings and problems that face previous related solutions such as these related to the used conventional clustering algorithm in [17] [33]. The solution which proposed in [17] depends on using conventional versions of k-means and k-medoid clustering algorithms during the pre-investigation stage to enhance the accuracy of investigation process. While in [33] the approach suggested is based on the usage of the hierarchical clustering algorithm in the pre-investigation process to enhance the investigation's accuracy and performance.

Another weak-point faces the previous suggested frameworks is related to the privacy of users' data through the investigation process. This is because many of proposed framework does not proposed any solution for the privacy problem [1] and [34].

The proposed framework consists of all stages of a conventional framework for digital investigation, with additional stages to achieve enhancement goals. The pre-investigation stage is essential and was added in the proposed framework to enhance the investigation process from in regards with the performance, accuracy, and security. The proposed solution consists of a set of stages as follows:

- Pre-investigation stage: this stage reduces the amount of information submitted to the investigator. The stage can improve the accuracy and efficiency of the investigation process.
- Evidence collection stage: in this stage the investigator is responsible for collecting and transferring to the investigator side only those clusters identified in the preceding stage which are related to the target files.
- Matching stage: conventional matching, such as sequential and carving based matching, is usually used in related framework found in the literature [1]. LSH based matching is suggested in the proposed framework in order to improve the matching process accuracy and efficiency.
- Analysis and presentation stage: in this stage the investigator is responsible for the analysis of the evidence gathered and finalizing the report to be presented in front of the court. .5

The improvement introduced in CBIF framework relates to the addition of the pre-investigation stage and to the improvement of the efficiency and accuracy of the investigation process

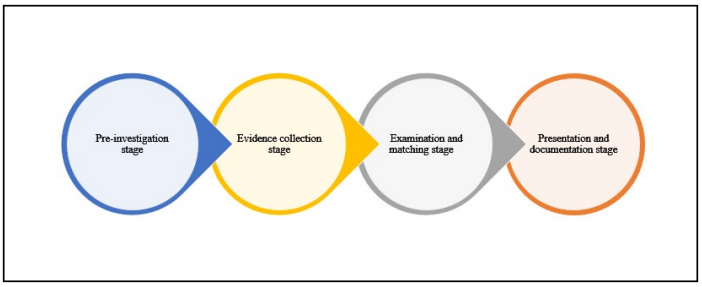


Fig. 1. Main Stages for the Cluster based Investigation Framework(CBIF).

by introducing the LSH based matching. Detailed discussion of each of the CBIF enhanced points will be presented in the subsections below.

A. Hierarchical Clustering used in the Pre-Investigation Stage

The objective of the hierarchical clustering algorithm used in the suggested idea is to improve the investigation process by enhancing the crime-related matching and related data collection within proper clusters. The idea of hierarchical clustering algorithm merges between a static clustering algorithm based on a genetic algorithm [23][27][30], and a dynamic clustering algorithm using an evolutionary algorithm [44][38].

Fig. 1 illustrates how the proposed hierarchical clustering algorithm operates. The algorithm starts by applying a genetic-based static clustering algorithm (GBSCA) to divide the storage center into a set of clusters based on files data-type. The storage center in Fig. 2 is divided into four main clusters; one for audio files, one for video files, one for text files and documents, and one for executable files. Afterward, another clustering round process is applied to each of the clusters formed. The second-level clustering algorithm is a genetic-based dynamic clustering algorithm (GBDCA). The objective of this second round is to divide each cluster generated into a suitable number of sub-clusters. The execution of all steps in Fig. 2 is the responsibility of the cloud service providers.

The evaluation metric which is used in the evaluation process refers to the Sum Square Error(SSE), whereas the equation for this evaluation metric is as follows.

$$SSE = \sum_{i=1}^{k} \sum_{i=1}^{|cf|} D(cj - fi)^2 ...(1)$$

In Equation (1) D refers to the Euclidean distance between Centroid file Cj and a specific file fi. The value of SSE must be minimized as much as possible.

B. Genetic based Dynamic Clustering Algorithm (GBDCA)

Dynamic clustering algorithm refers to a clustering strategy used to divide the storage center into a set of clusters [24], which is different from the conventional clustering algorithm. The goal of a dynamic clustering algorithm in CBIF is to divide the storage center into a proper number of clusters. Dynamic clustering algorithm consists of a set of iterations; the maximum number of iterations is defined before the algorithm is started. The iterations begin with an arbitrarily small number of clusters; at each iteration the number of clusters increases by one until the maximum number of iterations is reached. The number of clusters used is evaluated, based on two fitness functions, the internal fitness evaluation, and external fitness evaluation that is measured in each iteration.

The goal of the internal loop is to select the best centroid file which achieves the highest fitness value based on the SSE equation. fitness is achieved based on specific centroid files being compared with the best fitness value.

The results of applying the genetic-based dynamic clustering, as the second level in the hierarchical clustering algorithm, is getting the most suitable set of clusters based on the reported best fitness function of the external loop. The flowchart below shows the main stages of the GBDC algorithm.

The flowchart in Fig. 3 represents the main stages of the dynamic clustering algorithm, which can facilitate the investigation process by dividing the files into a suitable number of clusters. This means that the cluster which achieves a high level of similarity to the target file will contain all the files actually related to the target file indicated by low intra-distance which means high similarity between files in the same cluster. The equation for the ratio between inter-distance, i.e. the differences between files in different clusters, to intra-distance, which is proposed in [12] is mentioned in equation (2). This equation represents the external fitness function proposed in GBDCA, which is responsible for selecting the optimal number of clusters for the files in the original data set. The output of the external fitness evaluation helps in descending on the best number of related clusters:

$$R = \frac{InterDistance}{K-1} * \frac{IntraDistance}{n-K} \quad (1)$$

...(2)

In equation (2), "R" refers to the ratio between the inter-distance and intra-distance, "K" refers to the maximum number of clusters that are allowed in a solution, and "n" refers to the number of data instances, e.g. number of files. In the Algorithm 1, the similarity between data centers and all files in the cloud

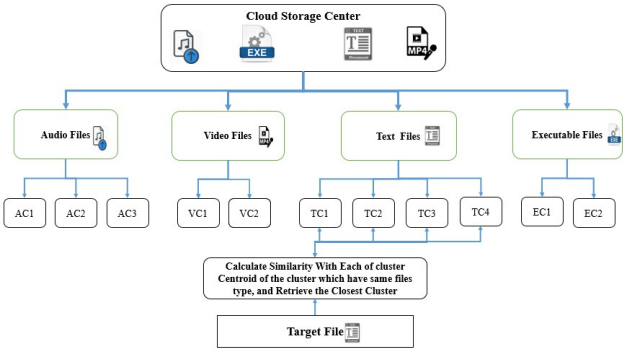


Fig. 2. The Main Steps of the Pre-investigation Step of the Proposed Idea to Divide the Cloud Storage Center into a Suitable Number of Clusters.

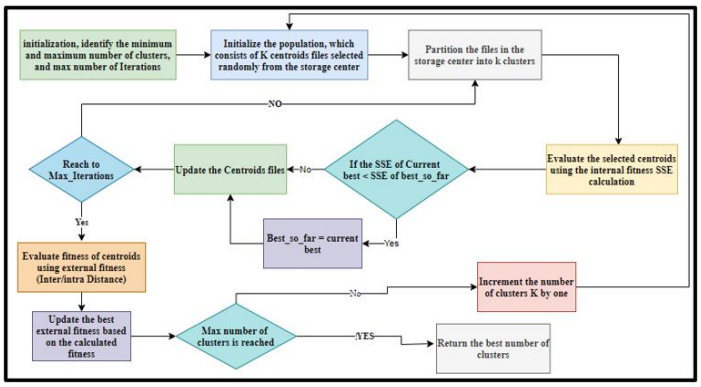


Fig. 3. Main Steps of Genetic-based Dynamic Clustering which used to Divide the Files in the Storage Center of the Cloud Computing.

Algorithm 1: Pseudo-code for dynamic clustering algorithm

```
Initialize Define G;
Define N;
Define LP;
Define T;
Define T1;
Define P[size];
Fill T with zeros;
Fill T1 with zeros;
Define I;
Generate permutation lists randomly;
for P = 0; P < L(P); P++ do
    check the values below for I = 0;
    I < size;
    I++ do
        Find the row which contains number i in
          permutation list;
        Let R in M(s) = I value;
    for C = I;
    C < File_Count;
    C++ do
        if R[c] = 1 then
            T[p][c] = i;
        if rowT[p][]contains-no-zeros then
            Break;
    for I = 1;
    I <= No.Permutation − Lists;
    I++ do
        Find the row which contains number I in
          permutation list;
        if row T1[p][] contains no zeros then
            setT1[p][1] = i
        if row T1[p][] contains no zeros then
            Break;
Return T;
Return T1;
```

storage center is determined using Euclidean distance, see [19], based on the following equation.

$$D = \sqrt{\sum(Cjr, fir)^2} \ldots (3)$$

Where D refers to the distance or the similarity between files. Two fitness functions are used through clustering based on genetic, internal fitness, and external fitness. The internal fitness function allows for the selection best centroid data item based on equation (1). The external fitness function has to do with the decision on the proper number of clusters based on the out come of equation (2).

As a summary of the actions and steps of the pre-investigation stage, it starts by dividing the storage center into a set of clusters based on the data types of the files. Afterward, another level of clustering round using genetic-based dynamic clustering, divides the generated clusters from static clustering step into a set of sub-clusters, which produces a set of small clusters for each of the first round's clusters. The next step in the proposed idea is to start the investigation process by retrieving the cluster that is highly similar to the tampered file based on ranking the clusters according to the similarity achieved through the internal loop of the genetic-based dynamic clustering using equations (1) and (3). Then comes the step of matching and finding the file or files related to the target file. The outcome of the pre-investigation step is to define the clusters that are very similar to the target file. The pre-investigation step can improve the performance and accuracy of the investigation process.

C. The LSH Algorithm Applied for the Matching and Examination Stage

Another step in the process of improving the matching and searching for evidence in cloud computing forensics depends on using the Locality Sensitive Hashing (LSH) algorithm. The LSH algorithm consists of three steps. The first step is the shingling step, which is responsible for building the shingle matrix of two dimensions. The second step is the implementation of the Min-hashing algorithm; this step is responsible for the compression of the shingle matrix. The last step is the implementation of the LSH algorithm [24], which is an enhanced matching algorithm that is implemented on the signature matrices generated to identify similar files in the previous step. LSH algorithm can improve the accuracy and efficiency of the investigation process based on reducing the size of the shingle matrix extracted from the original data. Fig. 4 shows the main stages of the LSH algorithm, where the steps begin after retrieving the clusters linked to the tampered file.

The following steps present detailed information on the improved examination and matching steps based on the LSH algorithm.

1) Shingling Step: The initial step of the LSH algorithm implementation is the extraction of shingles for each file of the retrieved cluster data, and developing a two-dimensional shingle matrix for all extracted files' shingles. The shingle matrix shown in Table II represents an example of a shingle matrix extracted from the files of the cluster. In the Table, a Shingle matrix ID in a row exists in documents, specified in column, where the flag bit is set.

Table II displays the shingle matrix structure. The shingle matrix building step is the initial step in the LSH algorithm procedure. All documents contents must be transformed into

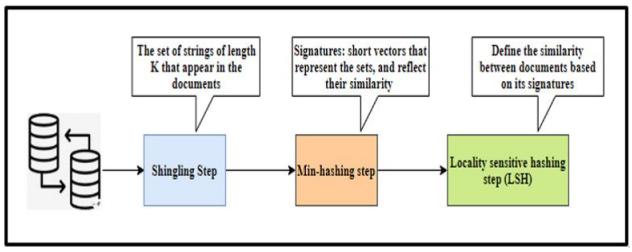


Fig. 4. Main Steps of Applying the LSH Algorithm to Enhance the Investigation Process in Cloud Computing

TABLE II. AN EXAMPLE OF A SHINGLE MATRIX EXTRACTED FOR A SET OF TEXT FILES

Shingle ID	First Document	Second Document	Third Document
Shingle 1	1	0	0
Shingle 2	0	1	0
shingle 3	0	0	1
shingle 4	1	0	0
Shingle 5	0	1	0
Shingle 6	0	0	1
Shingle 7	1	1	0
Shingle 8	0	0	1
Shingle 9	1	1	0
Shingle 10	0	0	1
Shingle 11	1	1	0
Shingle 12	0	0	1
Shingle 13	1	1	1
Shingle 14	1	1	0
Shingle 15	0	0	1

shingles that are added into a shingle matrix which represents the files in the cloud storage center.

The process of extracting shingles from documents is called the shingling extraction step where each document is represented by a string of characters. Shingles extraction is the process of creating a set of k-shingles for a document to be any sub string of length k found. As an example of the shingle extraction step, assuming the original document containing a, b, c, d, e, f, g, h, and k is 3 then some possible shingles are a, b, c, b, c, d, c, d, e, d, e, f, e, f, g, and f, g, h. The similarity of documents is measured based on the degree of overlapping between these shingles. Increasing the number of shingles overlapped between two files means that the two files are more similar to each other [11].

The similarity between documents is measured using Jaccard similarity metric [45], which depends on the degree of overlapping between shingles. matrix. The equation used to calculate Jaccard similarity is as follows

$$Sim(File1, File2) = \frac{|File1 \cap File2|}{|File1 \cup File2|}$$

... (4) By increasing the intersection ratio between the two files, the degree of similarity would increase. Based on this, the degree of similarity between the two files will increase when the amount of intersected characters or features between the two files is increased.

2) The Step of Compression using Min-hashing Algorithm : The signature matrix which is built through the first step of the pre-investigation stage will be usually very large due to the huge volume of files stored in cloud computing storage space. Accordingly, it is very complicated to handle the shingle matrix for the recovered clusters. Therefore, further processing is needed, which could be very time consuming to use the conventional method for performing the searching and matching steps.

It is necessary to compress the shingle matrix in a specific way while keeping the distance between the files in the original matrix. One of the best solutions is the Min-hashing algorithm, which can compress the shingle matrix into a small matrix called signature matrix "M". The most important feature of this algorithm is that it keeps the similarity of the underlying sets of shingles in the compressed version.

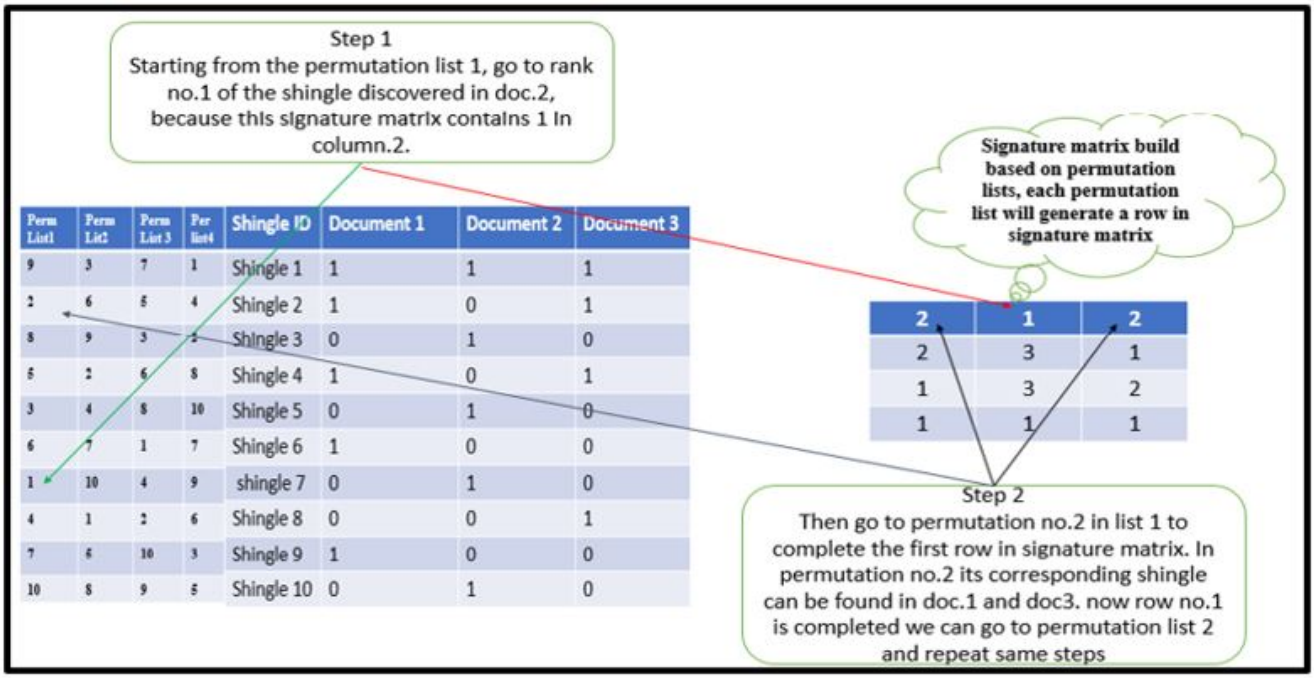
Min-hashing algorithm allows for the generating of a permutation list, which contains random numbers in the range from 1 to the number of shingles. The result of the Min-hashing process is stored in a signature matrix, where its rows are the Min-hashing value, and the column of the signature matrix is the file name. The number of permutation lists determines the accuracy of the signature matrix. Each permutation list produces a row in the signature matrix. Fig. 5 shows an example of the main step of converting the shingle matrix for a set of files into a signature matrix using 4 Min-hashing functions (4 permutation lists). The similarity between doc1 and doc3 in the shingle matrix is very close to the similarity between h(doc1) and h(doc2) in a signature matrix.
The pseudo code shown in Algorithm 1 contains the steps of min-hashing starting from the step of generation of a permutation list and all steps of compression for the shingle matrix until the production of the signature matrix and signature list. For more details on how to calculate the signature matrix out of the permutation lists please refer to [15].

3) Locality Sensitive Hashing (LSH): The last step after creating the signature matrix from the shingle matrix is applying locality-sensitive hashing to the produced signature matrix. LSH step is very helpful in dealing with parts of a file in order to discover criminal acts rather than dealing with he whole file imposed by other techniques. In the matching and examination forensic stage, we suggest to use LSH algorithm to enhance the investigation process based on the following factors:

1) The conventional way of matching byte-to-byte takes significantly long time carry out especially on the cloud.
2) High degree of accuracy in the matching process can be obtained. This aspect is essential and useful in a cloud computing environment since it contains a huge volume of files.
3) The LSH algorithm can handle all cases of the matching process, such as if the file was removed, partially modified, or fully updated.

Locality-sensitive hashing mainly depends on the division of the signature matrix from the shingle matrix into several bands. Specific hash function F(x) is used for each band of a signature matrix. The result of the hashing step maps to a specific bucket in a bucket list. Each column in the signature matrix is hashed multiple times based on splitting the column into band sets, and each band is hashed using a specific, preferably different, hashing function.

The Signature list will also be hashed multiple times based on splitting it into bands and hashing each band using a same has function that has been used for that band over the signature matrix. The bands with the same data will be hashed and mapped to the same bucket [22]. This means that the files which are shared in the bucket are exactly identical.
The flowchart in Fig. 6 shows the key steps of the LSH algorithm which are used in CBIF as the step of matching began by dividing the signature matrix into a specific number of bands. In order to determine that a set of files match, specific threshold has to be defined to determine which of the candidate set of files have contributed in creating the targeted document. The threshold in CBIF will be changed based on the corresponding results as shown in the flowchart in Fig. 6.



Perm List1	Perm List2	Perm List 3	Per list4	Shingle ID	Document 1	Document 2	Document 3
9	3	7	1	Shingle 1	1	1	1
2	6	5	4	Shingle 2	1	0	1
8	9	3	2	Shingle 3	0	1	0
5	2	6	8	Shingle 4	1	0	1
3	4	8	10	Shingle 5	0	1	0
6	7	1	7	Shingle 6	1	0	0
1	10	4	9	shingle 7	0	1	0
4	1	2	6	Shingle 8	0	0	1
7	8	10	3	Shingle 9	1	0	0
10	5	9	5	Shingle 10	0	1	0

2	1	2
2	3	1
1	3	2
1	1	1

Fig. 5. Main Steps of Building a Signature Matrix from the Shingle Matrix.

Algorithm 2: Pseudo code for the main steps of Min-hashing algorithm

```
Start Define shingle Matrix M(S);
Define shingle List L(x);
Define LP;
Define T;
Define T1;
Define P[size];
Fill T with zeros;
Fill T1 with zeros;
Define I;
Generate permutation lists randomly;
for P = 0; P < P.length; P++ do
    for I = 0; I < size; I++ do
        Find the row which contains number i in permutation list;
        Let R in M(s) = I;
    for C = I; C < File − Count; C++ do
        if R[c] = 1 then
            T[P][c] = i
        rowT[p][]contains − no − zeros Break;
for P = 1;P <= P.length;P++ do
    for I = 1; I <= LP; I++ do
        Find the row which contains number I in permutation list if rowT1[P][]containsnozeros then
            T1[P][1] = i;
        rowT1[P][]containsnozerosBreak;
Return T, T1;
```

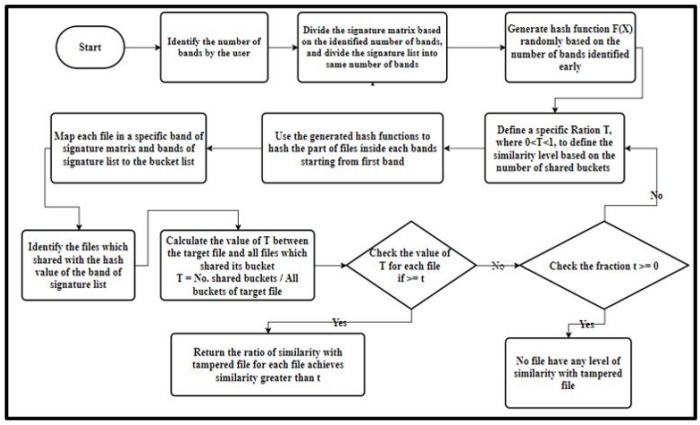


Fig. 6. Main Steps of Locality Sensitive Hashing Algorithm for Matching.

If there are no two files shared in a number of buckets based on the given threshold, the threshold's value will be reduced until it reaches zero. Therefore, it can be concluded that no file matches fully or partially the targeted file. Fig. 7 shows an example of how to apply LSH algorithm in the matching system to identify the set of matching documents.

IV. Experiments Implementation and Results Discussion

Two main experiments have been implemented throughout this research paper. The first one has to do with the implementation and evaluation of the clustering algorithm which makes use of CBIF. The second experiment has to do with the implementation and evaluation of LSH algorithm. All the experiments were implemented with Java Programming language using Net-Beans 8.1 on a PC with 64-bit Windows 10 operating system, and an Intel core i7-6500u CPU with 32GB of RAM.

A. Testing and Evaluation of Clustering Algorithm

To define the performance of the genetic-based dynamic clustering algorithm used through the second level of clustering, an implementation of the algorithm over ten, real and artificial data-sets obtained from UCI [10] was performed. The data-sets used in this experiment have various numbers of features ranging between '3' to '13' features. It contains various numbers of clusters ranging from '2' to '6' clusters. A set of experiments have been designed to measure the accuracy and performance of GBDCA before being used in CBIF. Each experiment has been applied over the data-sets mentioned in Table III. GBDCA has been compared with K-means clustering algorithm which we have been also implemented to measure its accuracy and performance.

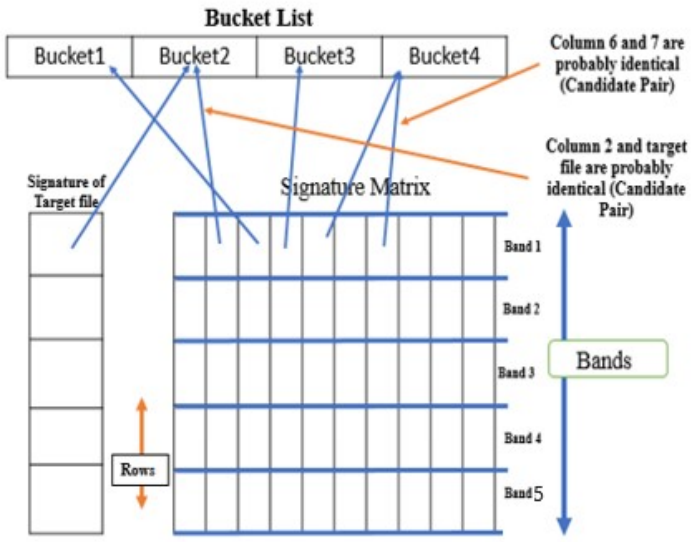


Fig. 7. A Locality-sensitive Hashing Algorithm (LSH) and How the Matching Process is Done based on the Map for One Band.

TABLE III. EVALUATION RESULTS OF THE PROPOSED DYNAMIC CLUSTERING ALGORITHM (GBDCA).

Dataset Name	Original No of Clusters	Avg. No of clusters GBDC	Avg. No of clusters CRO	Avg No.clusters Kmeans
E.coli DS	5	4.9	5.1	4.7
IRIS DS	3	3.43	3.63	4.23
Seed DS	3	3.367	3.53	4.134
Balance DS	3	3.23	3.36	4.1
Blood DS	2	2.5	2.6	2.934
Wine DS	3	2.8	2.53	3.567
Hepatitis DS	2	2.76	2.83	3.067
Vertebral DS	2	2.73	2.93	3.43
BC DS	2	2.36	2.57	3.034
Glass DS	6	5.033	5.63	4.863

Table III represents the number of clusters resulted from running each of the two clustering algorithms. The number of clusters achieved by each algorithm is compared with the original number of clusters for each data-set, and represented as the Error Rate. The results shown in Table III is the average value of ten experiment runs applied for each data-set. The main objective of these experiments and the comparison between GBDCA ad K-means algorithms were to prove that GBDCA is the better than K-means and can be used in the investigation process to enhance the forensics process of digital devices. The genetic based dynamic clustering algorithm depends on using the Inter/Intra ratio, as an external fitness function.

$$POE = \frac{ABS(OriginalNo - AchievedNo)}{Originalnumber} * 100$$

... (5) In the equation above, the POE refers to the percentage of error, ABS refers to the absolute value, which is calculated for the difference between the original number (OriginalNo.) of clusters for each algorithm and the average number of clusters achieved by the experiments applied on a specific data-set (AchievedNo.). Table IV shows the error rate for each competitive algorithm for the different data-set.

TABLE IV. ERRORS PERCENTAGE FOR EACH COMPETITIVE CLUSTERING ALGORITHM BASED ON THE RESULTS ILLUSTRATED IN TABLE 2

Dataset Description	Genetic Error Rate	CRO Error Rate	Kmeand Error Rate	Min Error Rate
E. coli DS	10%	14.67%	44.67%	10%-GA
IRIS DS	14.44%	21.11%	41.1%	14.44%-GA
Seed DS	12.22%	18%	37.78%	12.22%-GA
Balance DS	7.78%	12%	36.6%	7.78%-GA
Blood DS	25%	30%	46.67%	25%-GA
Wine DS	6.67%	16%	18.89%	6.67%-GA
Hepatitis DS	38.33%	42%	53.33%	38.33%-GA
Vertebral DS	36.67	47	53.33%	36.67%-GA
Breast Cancer	18.33%	28%	51.7%	18.33%-GA
Glass DS	16.11%t	6%	18.9%	6%-CRO
Average Results	19%	23%	40%	19%-GA

Table IV allows us to determine which is the best algorithm to be used through the step of dynamic clustering in the proposed hierarchical clustering algorithm. The average percentage of error rate achieved by the dynamic clustering-based genetic algorithm is 19%, and the percentage of error achieved by the dynamic conventional k-means algorithm is 40%. The results shown in Table 3 can be interpreted as a recommendation to use a genetic-based dynamic clustering algorithm in our solution, given the low percentage rate in comparison with the K-means algorithm.

B. Experimental Implementation for the LSH Algorithm

Several tests have been performed to determine the accuracy and reliability of CBIF and different data set sizes have been used. Furthermore, various structures for the tampered file were designed and used to test the proposed solution through a set of experiments. The data set used in our experiments is the Reuters data set consisting of 21000 files [46]. The experiments are divided into four main categories based on the structure of the tampered file used. Each category of experiments consists of 10 runs and the average value was calculated. In each run, the variables that controls the structure of tampered file were different. The details about the structures of tampered file will be shown in detail just before discussing the results of that experiment. The Reuters files were divided into sets and the series of experiments were applied to each of these sets; the size of these sets will be addressed below. The matching accuracy based on equation (5) has also been calculated.

The data set used in these experiments is divided into clusters based on the dynamic clustering algorithm decision proposed in the hierarchical clustering step. In most cases, the number of clusters achieved was 6 clusters. The files were distributed over the clusters based on the level of similarity between cluster centroid files and all other files.

1) Tampered File Description: The tampered file used in the experiments to evaluate the accuracy and efficacy of the CBIF has many structures, as follows.

- Structure-One: The first structure for the tampered file

consists of three random files from three random clusters. File "A", from a random cluster "A", is involved with 10% of the tampered file content. Random file "B", from another random cluster "B", is involved with 20% of the tampered file content. Finally, 70% of the tampered file creation process is from random file "C", which belongs to yet another cluster "C". The files and clusters "A", "B", and "C" are random in that they may vary from experiment to experiment. This also applies for the next structures. see Fig. 8A.

- Structure-Two: The second structure for the tampered file consists of three random files from three random clusters with different participation rates with 20%, 30%, and 50% for random files A, B, and C from random clusters A, B,and c respectively. Fig. 8B shows the percentage of participation for each file in the tampered file content data
- Structure-Three: Another structure for the tampered file was used that consists of two random files from two random clusters. Random file "A", from random cluster "A", is involved with 40% of the tampered file content, while random file "B", from random cluster "B", is involved with 60% of the tampered file content. Third part of Fig. 8C shows the percentage of participation for each file in the tampered file content.
- Structure-Four: The last structure of the tampered file used consists of two random files "A" and "B" from two random clusters "A" and "B" where each file is involved with 50% of the tampered file as shown in Fig. 8D.

2) Experimental Implementation for LSH Based Framework Using Different Tampered File's Structures : Multiple experiments were conducted based on the above mentioned tampered file structures. Each experiment was replicated sixty times with random file selected randomly from a different cluster each time.The AMA is the average results for '60' of the repeated experiments. Each experiment has different variable values for the structure of the tampered file. The purpose of repeating the experiments '60' times is to test CBIF's performance and reliability in order to deal with all cases of file creation being tampered with. The purpose of repeating the experiments is to test the CBIF's ability to recognize the original files in all cases of manipulated files.

For example, in the experiment defined for Structure-One of the tampered file structure, Cluster A can be randomly selected as Cluster '2', and File X can be randomly selected to be File '5', meaning that the tampered file has 10% of its content taken randomly from File number '5' in Cluster '2'. Cluster B can be cluster '4' and File Y can be File '1', this means that the randomly selected File '1' in Cluster '4' has participated with 20% of that tampered file content. Finally, Cluster C can be cluster '5' and File Z can be File '2', meaning that from Cluster '5' we have randomly selected File '2', which contributes to 70% of that same tampered file's content. Fig.9 consists of 4 main sub-figures A,B,C, and D, whereas each sub-figure of Fig. 9 is specified for an experiment based on a specific tampered file structure.

The Average Matching Accuracy (AMA) is a metric that we have used to measure the accuracy of the matching process and is calculated based on the percentage of matching between the tampered file and the original files which participated in the tampered file content. The equation which was used to calculate AMA between the tampered files and original files is equation (6). In this equation the Avg-Matching(Bx, . . . , By) refers to the average results of matching between the tampered file and all clusters participating with a certain percentage of the tampered file's construction. As an example, if equation (6) was used to calculate the average accuracy for a cluster involved with 50% of the tampered file content, then in this case, the AvgMatching(F.1, F.2, ..., F.n) is the average result of matching between the tampered file and the "n" files F.1, ..., F.n that are involved with 50% of the tampered file content. Here "n" is the number of runs the experiment has been repeated which is in our case equals 60.

$$AMA = 100 - \frac{ABS(AvgMatching(Bx, .., By) * 100 - Actualmatchingpercentage)}{ActualmatchingPercentage} * 100 \quad ...(6)$$

The results in Fig. 9 show how accurate CBIF is for the matching process, and presents how the AMA for the results of matching is close to the actual ratio of participation in all matching results.

Details of each result shown in Fig. 9 will be thoroughly explored next before comparing CBIF with other rival algorithms. The purpose of this is to reduce the complexity of discussing this comparison into two simpler phase. In the first phase we discuss the results obtained for CBIF and in the second phase we compare these results with results obtained for other protocols.

- First Experiment
  The tampered file used through the first experiment was built according to Structure-One of the tampered file structures shown in Fig. 9A, which consists of three files.
  Fig. 9-A consists of three columns where each column shows the average matching ratio between the tampered file and the original files in a specific cluster based on the percentage of participation in the tampered file content. The figure show the participation results of three clusters, thus we have three columns, since the participation ratios of other clusters were negligible and thus are not shown in the figure.
  The average accuracy of matching between the tampered files and original files which were involved in 10% of tampered file contents is 76.78%, the average accuracy of matching with the files involved in 20% of tampered file contents is 94.44%, and the average result of matching with the files participating in 70% of the content is 87.71%.
  The conclusion that can be drawn from Fig. 9A is that CBIF's has the ability to deal with the case when the target file in the investigation process was mixed with other files. This indicates that the file that the investigator was searching for and modified at a certain rate was found at a high accuracy rate.
- Second Experiment
  Another experiment was conducted based on the second structure of the tampered file that was referred to in Fig. 9B. This figure consists of three columns, each column is specified for the AMA between

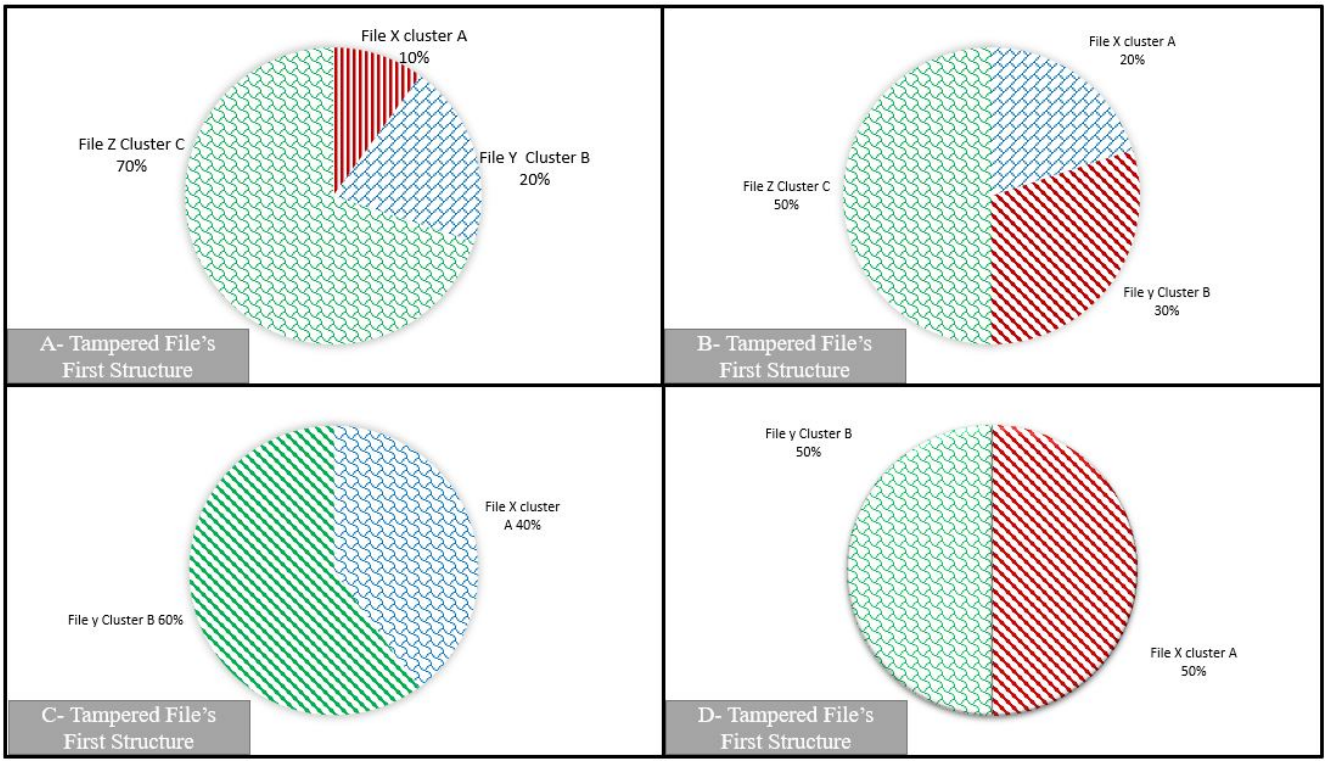


Fig. 8. Key Structures of Tampered File Content which were used by Experimental Implementation.

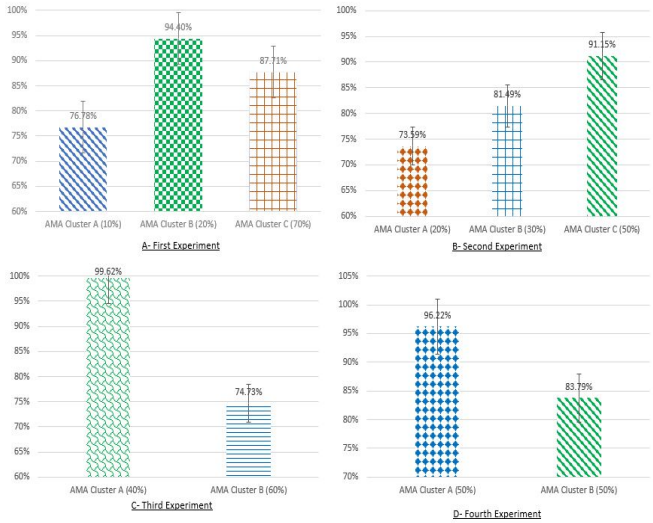


Fig. 9. Results for Set of Experiments based on Different Structures for Tampered Files.

tampered file and each of the original files which were involved in the creation process of the tampered file.

The files that participated in the tampered file's content creation achieves a high level of similarity if compared with other files, such that the accuracy of matching that was shown in Fig.9B depends on the values of actual matching results. The conclusion that can be drawn on the basis of the results shown in Fig.9B is that the proposed CBIF succeeds in retrieving the right files with a higher matching level than the other files in the storage space.

- Third Experiment The third experiments was applied based on the third structure of the tampered file mentioned earlier in Fig. 9C. This figure consists of two columns, each column represents the average accuracy of matching between the tampered files and original files which were involved in the construction process of the tampered file.
  The results shown in Fig. 9C represents the extent of accuracy and reliability of CBIF to deal with the third case of tampered file content mentioned earlier. This gives a strong argument for the benefits and importance of using the proposed CBIF.
- Fourth Experiment. The last experiment was applied based on the structure of the tampered file mentioned in Fig. 9D. The results shown in Fig. 9D illustrate the AMA for this experiment which consists of two bars, each bar shows the average accuracy of matching with each original file involved in creating the tampered file.
  The results shown in Fig. 9D show the accuracy of CBIF in retrieving the correct files which actually participated in the contents of the manipulated file.

3) Comparisons between the Proposed Solution and Related Cluster-based Solutions: Two of the related solutions have been implemented and compared with the proposed CBIF.

The purpose of this comparison is to demonstrate matching accuracy level of CBIF relative other similar techniques. The first competitive solution relies on using the traditional k-means clustering algorithm [10]. The idea of the solution revolves around applying the k-means clustering algorithm in the investigation process to divide the documents in the storage center into groups and to focus on the group which is most similar to the target file. The second competitive solution is to use a k-medoid clustering algorithm to enhance the investigation process with respect to accuracy and performance [17]. Both solutions have been implemented and used with the data-set which was used to test CBIF. The structure of the tampered file which was used through the experiments of testing the competitive solutions is the same as the structure which mentioned earlier to test the proposed CBIF. All of experiments have been run 60 times. The average accuracy of matching for all experiments have been applied using equation 6 . The results for the experiments are as follows:

- Comparison between CBIF, a K-means based solution, and a k-medoid based solution using the first structure of the tampered file

  Fig. 10A shows the AMA for the three solutions based on structure-One of the tampered file. The figure consists of three groups of bars, each group contains three bars. Each bar refers to the average accuracy of matching between the tampered file and other files, exist in a specific cluster, using one of the three compared solutions. The first group which represents the AMA between the tampered file and a clustered file which participated in 10% of the tampered file content. CBIF has achieved higher AMA than the results obtained by the other solutions for the first tampered file structure.
  The AMA of the file which participated with 70% of tampered file content is greater in CBIF, close to 88%, compared to the other solutions, almost 75% and 65% for k-means and k-medoid, respectively. This demonstrates the ability of CBIF in improving the matching accuracy which is better than similar solutions. This helps in finding the correct file that actually participated in the contents of the tampered file.
  Another important enhancement of CBIF is related to the rank of the original files which participated in the tampered file's content. The files that were already involved in creating the file that was tampered with, in most experiments, have the highest degree of similarity with the tampered file. On the other hand, only in 72% of the total experiments did the original files that participated in the composition of the manipulated file obtain the highest similarity with the tampered file, which means that the original files did not get the highest similarity in a 28% of experiments.

- Comparison based on the second structure of the tampered file
  Other experiments have been designed and implemented for the related solutions to show how CBIF solution performs. AMA has been calculated for the 60 runs of experiment for each solution. The summary of these experiments can be shown in Fig. 10B.
  Results shown in Fig. 10B represent the AMA of the competitive solutions to retrieve the original participated files. CBIF achieved an accuracy level that is, also in this experiment, higher than other two competitive solutions for all participated files, as can be seen in the results shown in group three of Fig. 10B.
  The AMA achieved by CBIF is 91.15%, whereas the average accuracy achieved by the k-means based solution is close to 75%, and the average accuracy of matching achieved by the k-medoid based solution is close to 67%. This indicates the gap in the performance between CBIF solution and the other related solutions. This point is important to encourage the prefer the usage of the CBIF in investigation process over the other rival solutions.
  Another significant improvement accomplished by CBIF is the ranking of files based on the degree of similarity with the tampered file. File ranking here indicate the ability of the solution to rank the actual original files that have participated in the tampered file as the top ones. For example assume that file A and B have participated in creating a tampered file T with a certain percentage from each one. A solution that gives the highest percentages to files A and B, regardless of the accuracy results for each one, is assumed to reach 100% ranking accuracy. While a solution that gives a percentage for another file, e.g. file C, that is higher than A or B gets a lower ranking accuracy.
  In the experiments that are based on the Structure-One of the tampered file, using a solution based on k-means and a solution based on k-medoid, the files involved with the tampered file content do not reach the highest degree of similarity with the tampered file in some of the experiments. In the solution that depends on the k-medoid clustering algorithm, the ranking accuracy was close to 77% while in the solution that depends on the conventional k-means clustering algorithm, 80% was the ranking accuracy. CBIF has achieved 95% to 100% ranking accuracy in all of the experiments and for all tampered file structures.

- Comparison based on Structure-Three of the tampered file.
  Third part of Fig. 10C shows the results for the experimental implementation of competitive solutions based on Structure-Three of the tampered file mentioned earlier. For the 60 experimental runs, the average accuracy was calculated. The average level of accuracy obtained by CBIF was higher than the average accuracy achieved by the competitive solutions in the matching process with the files that participated in 40% of the tampered file contents. Although the difference is less than the case with 40% participation, CBIF achieved higher accuracy also for the 60% participation results. Additionally, for the ranking accuracy, in the k-means based solution 86% of all experiments. In the k-medoid based solution is 84% of all experiments.

- Comparison between the CBIF and other competitive solutions based on the Structure-Four of the tampered file

The results shown in last part of Fig. 10D represent the average accuracy of matching for the three competitive solutions to enhance the investigation process. CBIF achieves higher AMA in both clusters, which has participated with 50% of the tampered file, than the other compared solutions. Moreover, k-medoid based solution, shows ranking accuracy as 86% while solutions based on k-means results in 90% ranking accuracy which is much less than what CBIF has achieved.

To conclude, from the section of comparisons between the proposed CBIF and other rival solutions, it can be inferred that the proposed solution improves the average matching accuracy for the investigation process in all cases and for all of the files structures involved. This point is important in the forensics process, as many of the crimes are hidden based on mixing the original file with other files. Moreover, the point of enhancement in the accuracy of matching is what all frameworks and solutions are looking for to achieve; the results shown in the above figure indicate a noticeable improvement on average matching accuracy in most cases. This is supported by the results of ranking accuracy which prove that CBIF outperforms K-means and K-mediod algorithms in all the conducted experiments.

4) Comparison between CBIF and Related Solution on the Basis of Execution Time: Another comparison is between the competitive solutions based on the execution time for each algorithm. The comparison is based on the size of the data-set used. The execution time was measured for applying the three solutions for the 2000 file, 4000 file, 8000 file, 16000 file, and 21000 file data-sets. The results for the execution time are shown in Table V. The results in Table V refers to the time needed for the investigation to find a file similar to the target file. k-medoid clustering algorithm needs less time than k-means based investigation process which in turn takes less that proposed CBIF. This slight extra is justified by the higher average matching accuracy achieved by CBIF.

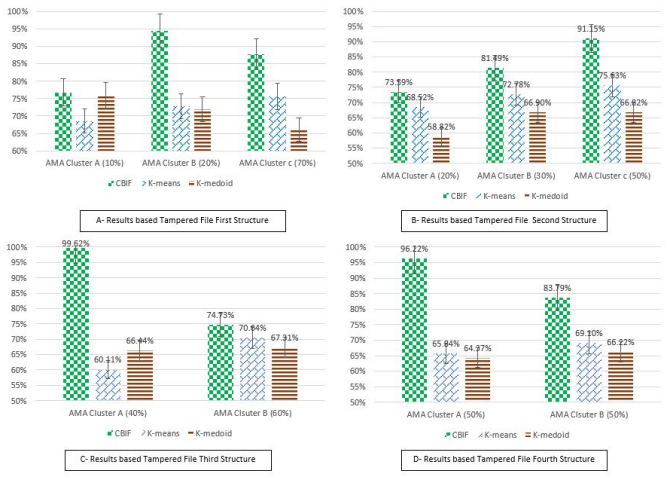


Fig. 10. Results for All Competitive Algorithms based on Different Structures for Tampered File Content.

TABLE V. RESULTS FOR THE EVALUATION PROCESS OF THE PROPOSED DYNAMIC CLUSTERING ALGORITHM

No.Files in a data-set	Execution time for CBIF in seconds	Execution time For k-means based investigation in seconds	Execution time for k-medoid in seconds
1000	189.2	150.388	135.9
2000	238.7	215.9	182.6
4000	1972.15	1213	600.356
8000	4370.16	4222.11	3446.462
16000	12325.2	11863.5	10695
21000	18735.5	17962	16818.5

V. CONCLUSION

Many frameworks and solutions have been proposed in the field of cloud computing forensics. Each of these frameworks have specific drawbacks and weak points in certain stages of the investigation process. In the paper, CBIF is proposed to focused on a specific issue of cloud computing forensics; namely, the performance and accuracy of the investigation process. CBIF adds a new stage called the pre-investigation stage, which is responsible for filtering and grouping the evidence and files in the cloud storage center into a set of groups based on using a hierarchical clustering approach. This stage enhances the average accuracy of the matching process by dividing the storage center into sets of groups.

The results achieved from the experimental implementation shows how the proposed CBIF enhances the average accuracy when compared with the related solutions, the k-means based solution and the k-medoid based solution. CBIF achieved higher accuracy matching levels and ranking accuracy than the competing solutions in most of the experiments.

The proposed CBIF solution can be enhanced in the future by enhancing the LSH step of current solution by changing the technique of selecting band size.

REFERENCES

[1] M Edington Alex and R Kishore. Forensics framework for cloud computing. *Computers & Electrical Engineering*, 60:193–205, 2017.

[2] Wesam Almobaideen and Muhyidean Altaramneh. Fog computing: survey on decoy information technology. *International Journal of Security and Networks*, 15(2):111–121, 2020.

[3] Wesam Almobaideen and Ola Malkawi. Application based caching in fog computing to improve quality of service. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 20–27. IEEE, 2020.

[4] Arushi Arora, Sumit Kumar Yadav, and Kavita Sharma. Denial-of-service (dos) attack and botnet: Network analysis, research tactics, and mitigation. In *Handbook of Research on Network Forensics and Analysis Techniques*, pages 117–141. IGI Global, 2018.

[5] Mustafa Aydin and Jeremy Jacob. A comparison of major issues for the development of forensics in cloud computing. In *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*, pages 77–82. IEEE, 2013.

[6] Leonardo Babun, Amit Kumar Sikder, Abbas Acar, and A Selcuk Uluagac. Iotdots: A digital forensics framework for smart environments. *arXiv preprint arXiv:1809.00745*, 2018.

[7] Stacey O Baror, Hein S Venter, and Richard Adeyemi. A natural human language framework for digital forensic readiness in the public cloud. *Australian Journal of Forensic Sciences*, 53(5):566–591, 2021.

[8] Venansius Baryamureeba and Florence Tushabe. The enhanced digital investigation process model. *Digital Investigation*, 2004.

[9] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering. In *NDSS*, volume 9, pages 8–11. Citeseer, 2009.

[10] Catherine Blake. Uci repository of machine learning databases. *http://www. ics. uci. edu/˜ mlearn/MLRepository. html*, 1998.

[11] Andrei Z Broder. Identifying and filtering near-duplicate documents. In *Annual Symposium on Combinatorial Pattern Matching*, pages 1–10. Springer, 2000.

[12] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[13] Mumin Cebe, Enes Erdin, Kemal Akkaya, Hidayet Aksu, and Selcuk Uluagac. Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles. *IEEE Communications Magazine*, 56(10):50–57, 2018.

[14] Yu-Jia Chen and Li-Chun Wang. A security framework of group location-based mobile applications in cloud computing. In *2011 40th International Conference on Parallel Processing Workshops*, pages 184–190. IEEE, 2011.

[15] Ondrej Chum, Michal Perd'och, and Jiri Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24. IEEE, 2009.

[16] Jasmin Ćosić, Zoran Ćosić, and Miroslav Baća. An ontological approach to study and manage digital chain of custody of digital evidence. *Journal of Information and Organizational Sciences*, 35(1):1–13, 2011.

[17] Luis Filipe da Cruz Nassif and Eduardo Raul Hruschka. Document clustering for forensic analysis: An approach for improving computer inspection. *IEEE transactions on information forensics and security*, 8(1):46–54, 2012.

[18] Larry Daniel. *Digital forensics for legal professionals: understanding digital evidence from the warrant to the courtroom*. Elsevier, 2011.

[19] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980.

[20] Sergio Decherchi, Simone Tacconi, Judith Redi, Alessio Leoncini, Fabio Sangiacomo, and Rodolfo Zunino. Text clustering for digital forensics analysis. In *Computational Intelligence in Security for Information Systems*, pages 29–36. Springer, 2009.

[21] Athanasios Dimitriadis, Nenad Ivezic, Boonserm Kulvatunyou, and Ioannis Mavridis. D4i - digital forensics framework for reviewing and investigating cyber attacks. *Array*, 5:100015, 2020.

[22] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[23] David E Goldberg. Genetic algorithms in search. *Optimization, and MachineLearning*, 1989.

[24] Christopher Hargreaves and Jonathan Patterson. An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation*, 9:S69–S79, 2012.

[25] Ben Hitchcock, Nhien-An Le-Khac, and Mark Scanlon. Tiered forensic methodology model for digital field triage by non-digital evidence specialists. *Digital investigation*, 16:S75–S85, 2016.

[26] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. Nist cloud computing standards roadmap. *NIST Special Publication*, 35:6–11, 2011.

[27] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[28] Liu Jiang, Guiyan Tian, and Shidong Zhu. Design and implementation of network forensic system based on intrusion detection analysis. In *2012 International Conference on Control Engineering and Communication Technology*, pages 689–692. IEEE, 2012.

[29] Karen Kent, Suzanne Chevalier, Tim Grance, and Hung Dang. Guide to integrating forensic techniques into incident response. *NIST Special Publication*, 10(14):800–86, 2006.

[30] Mohammad Khanafsa, Ola Surakhi, and Sami Sarhan. A parallel face detection method using genetic & cro algorithms on multi-core platform. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pages 1–6. IEEE, 2019.

[31] Mohammed Khanafseh, Mohammad Qatawneh, and Wesam Almobaideen. A survey of various frameworks and solutions in all branches of digital forensics with a focus on cloud forensics. *Int. J. Adv. Comput. Sci. Appl*, 10(8):610–629, 2019.

[32] Manish Kumar, M Hanumanthappa, and TV Suresh Kumar. Network intrusion forensic analysis using intrusion detection system. *Int. J. Comp. Tech. Appl*, 2(3):612–618, 2011.

[33] Chanjin Lee and Mokdong Chung. Digital forensic for location information using hierarchical clustering and k-means algorithm. *Multimedia Society Journal*, 19(1):30–40, 2016.

[34] Sheik Khadar Ahmad Manoj, D Lalitha Bhaskari, et al. Cloud forensics-a framework for investigating cyber attacks in cloud environment. *Procedia Computer Science*, 85:149–154, 2016.

[35] Raffael Marty. Cloud application logging for forensics. In *proceedings of the 2011 ACM Symposium on Applied Computing*, pages 178–184, 2011.

[36] Matthias Neugschwandtner, Paolo Milani Comparetti, Gregoire Jacob, and Christopher Kruegel. Forecast: skimming off the malware cream. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 11–20, 2011.

[37] Liwen Peng, Xiaolin Zhu, and Peng Zhang. A framework for mobile forensics based on clustering of big data. In *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, pages 1300–1303, 2021.

[38] Mohammad Qatawneh, Ahmad Alamoush, and Ja'far Alqatawna. Section based hex-cell routing algorithm (sbhcr). *International Journal of Computer Networks & Communications*, 7(1):167, 2015.

[39] Mohammad Qatawneh et al. Multilayer hex-cells: A new class of hex-cell interconnection networks for massively parallel systems. *IJCNS*, 4(11):704–708, 2011.

[40] Mark Reith, Clint Carr, and Gregg Gunsch. An examination of digital forensic models. *International Journal of Digital Evidence*, 1(3):1–12, 2002.

[41] M Rogers. Dcsa: a practical approach to digital crime scene analysis. *West Lafayette, Purdue University*, 2006.

[42] Keyun Ruan, Joe Carthy, Tahar Kechadi, and Ibrahim Baggili. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation*, 10(1):34–43, 2013.

[43] Huda K Saadeh, Wesam Almobaideen, and Khair Eddin Sabri. Ppustman: Privacy-aware publish/subscribe iot mvc architecture using information centric networking. *Modern Applied Science*, 12(5):128, 2018.

[44] Sarah Shukri, Hossam Faris, Ibrahim Aljarah, Seyedali Mirjalili, and Ajith Abraham. Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. *Engineering Applications of Artificial Intelligence*, 72:54–66, 2018.

[45] Jatsada Singthongchai and Suphakit Niwattanakul. A method for measuring keywords similarity by applying jaccard's, n-gram and vector space. *Lecture Notes on Information Theory*, 1(4), 2013.

[46] Kilian Stoffel, Paul Cotofrei, and Dong Han. Fuzzy methods for forensic data analysis. In *2010 International Conference of Soft Computing and Pattern Recognition*, pages 23–28. IEEE, 2010.

[47] Akash A Thakar, Kapil Kumar, and Baldev Patel. Next generation digital forensic investigation model (ngdfim)-enhanced, time reducing and comprehensive framework. In *Journal of Physics: Conference Series*, volume 1767, page 012054. IOP Publishing, 2021.

[48] Sebastiaan Von Solms, Cecil Louwrens, Colette Reekie, and Talania Grobler. A control framework for digital forensics. In *IFIP International Conference on Digital Forensics*, pages 343–355. Springer, 2006.

[49] Wei Wang and Thomas E Daniels. A graph based approach toward network forensics analysis. *ACM Transactions on Information and System Security (TISSEC)*, 12(1):1–33, 2008.

[50] Shams Zawoad, Amit Kumar Dutta, and Ragib Hasan. Seclaas: secure logging-as-a-service for cloud forensics. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pages 219–230, 2013.

[51] Tanveer Zia, Peng Liu, and Weili Han. Application-specific digital forensics investigative model in internet of things (iot). In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–7, 2017.