Sampling and similarity hashes in digital forensics: An efficient approach to find needles in a haystack

Vitor Hugo Galhardo Moia[1], **Marco Aurélio A. Henriques**[1]

[1]School of Electrical and Computer Engineering (FEEC)
University of Campinas (UNICAMP)
Campinas, SP, Brasil 13083-852

vhgmoia,marco@dca.fee.unicamp.br

Abstract. *The amount of data handled by forensics examiners has grown significantly over the past few years. Investigations involving digital devices used to store valuable information related to crimes are becoming very common. Methods aiming to reduce the time spent in each case and yet be effective in finding evidences are necessary. Hash-based functions appeared as a solution, but they are limited only to yes/no answers, failing in detecting similarities among objects. New approaches have been proposed to solve this problem as, for example, the similarity hash. This function can find small-scale similarities in objects with different contents. However, such flexibility comes with a price: The high cost compared to traditional approaches. In this paper, we propose and evaluate the combination of similarity hashes and sampling techniques, where only a small portion of a seized media is analyzed to produce reliable results for a triage process in a short period of time. We also propose to apply similarity hashes to the sector level and present a new approach of clustering sectors to reduce even more the investigation time. The proposed approach can reliably identify even small fragments of deleted files that still remains in a seized media.*

1. Introduction

With the increasing volume of data in today's world, forensics examiners face a significant challenge in analyzing seized devices. Cases where digital electronics are used to store data related to crime are increasing. Users store more and more data as storage capacity increases and prices get cheaper. On the other hand, forensic professionals normally do not have the resources to scale in the same way. Methods capable of evaluating seized devices in a reasonable time become necessary to deal with this trend.

Hash-based techniques are an efficient solution for identifying objects in investigations. Forensics examiners create hashes for the seized media objects and look up for matches in a database containing hashes of interest objects. However, hash functions fail in identifying similar objects. They only give yes/no answers. If two files are identical, their hashes will be precisely the same, but if they differ at least by one bit, their hashes will be completely different. This way, documents with small changes will not be detected, and examiners may lose important evidences.

To overcome such limitation, researchers have employed similarity hashes, which are similar to hash functions in using unique representations for data, but with the difference that small variations will reflect in small changes in the digest. They can create representations that allow us to compare two objects and give a similarity value for them.

In this work, we use sdhash as our similarity tool [Roussev 2010], although it is very expensive in relation to traditional hash functions. We present an approach to minimize its high cost, combining it with statistical sampling techniques to reduce the amount of data analyzed and hence the number of similarity hashes generated. Also, we work at the disk sector level (or equivalent) with the purpose of detecting even small fragments of objects that remained in the media after been deleted but not overwritten.

2. Related Work

A significant amount of work has been done with hash functions to reduce the volume of data analyzed when comparing data in a seized device with a database of known interest objects. This database can encompass bad objects, which can be a lead to follow, or good ones, eliminated from the analysis, like operating system data, well-known software objects etc. The comparison is in a hash level, using algorithms like MD5 or SHA-1. In this work, we will focus on the approach using a bad objects database for our experiments.

Hash functions are used to create a representation of an input with arbitrary size into a fixed-length size value. If a single bit of the input is changed, the output will be radically different. They are commonly used in integrity checks, digital signatures, and many other applications. Their security stands on the fact that it is computationally infeasible to find two different inputs with the same hash.

However, hashing data has some drawbacks in the forensics field. They only give binary answers. Malicious users can make small changes to objects to make this technique fail, since their hashes will be far different. To mitigate this problem, researchers adopted another approach to deal with it. This approach, known as piecewise hash, splits the object into fixed-size segments, generate hashes for each one of them and then make the comparison. Besides identifying whole objects, this approach could also be useful to deal with data in disk sectors. However, there are also some difficulties in using this technique. Choosing the appropriate object block size is one of them, due to alignment issues. If objects are modified, their stored fragments in the hard drive may not align with the ones in the database. This way, the hashes will be different, and the method will fail.

To overcome this issue, a new method was proposed to compare objects. This mechanism, called similarity hash or similarity digest, can detect commonality among the binary structure of objects, providing a confidence measure of the similarity among them. The first idea of using such method was proposed by Rabin [Rabin et al. 1981], using random polynomials to create data fingerprinting. More recent research include the ssdeep algorithm [Kornblum 2006], which combines piecewise and rolling hash techniques. Data is broken into pieces using a rolling algorithm, and for each piece, a hash function is used to produce a digest. In the end, all of these digests are concatenated to generate the representation of the object. Other methods with the same purpose can be found in Martínez et al.' work [Martínez et al. 2014].

A more recent method for creating similarity hashes, called sdhash, was presented by Roussev [Roussev 2010]. This method will be discussed in the next section and used in this work due to its interesting characteristics and because of its availability, performance, and documentation. In another work, Roussev presents a comparison between sdhash and ssdeep and shows that the former has a better performance than the latter [Roussev 2011].

Another technique used in the present work is statistical sampling, with the purpose of reducing the amount of data analyzed in an investigation. Instead of examining all objects from the seized media, we extract random samples of it to make a statement about the entire population. We aim to implement the same idea of sampling discussed in Garfinkel's work [Garfinkel et al. 2010], but using similarity hashes instead of block hashes to increase the effectiveness of the search for evidences. We also use a sector level approach in order to detect even small parts of target data. This way, we expect to reduce the amount of time to pursue a triage process and increase the accuracy of the search, where evidences with small changes will not confuse the investigation.

3. SdHash: Similarity Digest Hash

The sdHash tool, developed by Roussev [Roussev 2010], has the purpose of selecting multiple features that are most likely to be unique to an object in order to produce a similarity representation. Fig. 1 illustrates the overall process performed by this tool. First, features (sequence of B bytes) are extracted from the object and have their entropy calculated. Then a filter selects the ones considered the best in representing the target. The next step is the creation of a sequence of Bloom Filters [Bloom 1970] to generate the object representation, as illustrated in Fig 2. In this process, up to f selected features are hashed (using SHA-1 algorithm) and the result is split into five sub-hashes. The 11 least significant bits of each sub-hash are taken to address the bits within the Bloom filter. The implementation described in the Roussev's paper uses B=64 and 256-byte Bloom filters, with up to $f = 128$ features mapped to each filter. New filters are created as needed.

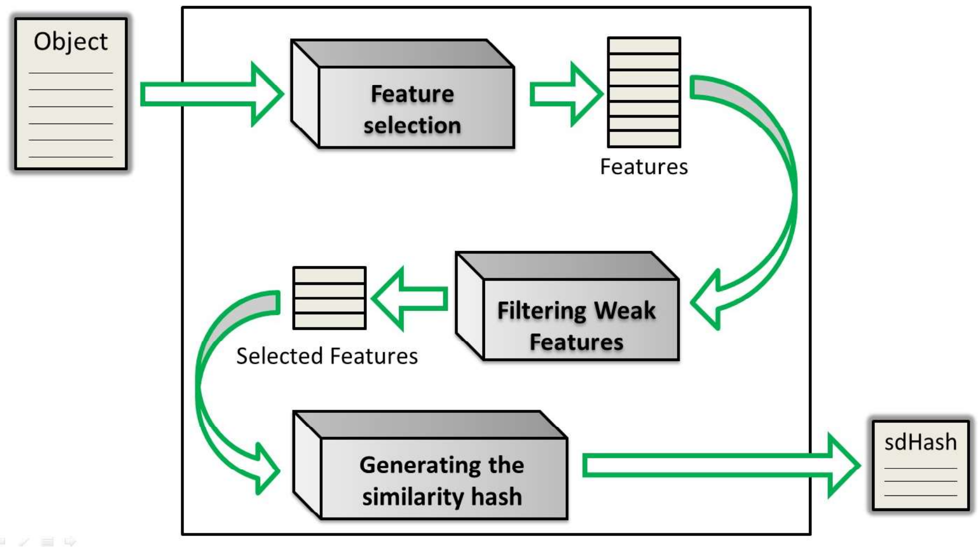

Figure 1. Overall process of generating a similarity hash with sdhash

Comparing two similarity hashes means comparing their Bloom filters. The first filter from the first object is compared to every filter from the second one. The maximum similarity score is selected. The process is repeated for all other features from the first object. Finally, an average of the results is calculated to produce a score which varies from -1 to 100 [Roussev 2010]. Roussev and Quates [Roussev and Quates 2012] point out that a result of 0 (zero) means that the objects are uncorrelated, while -1 (a rare occurrence) means that at least one of the digest have no enough features to produce a reliable comparison. Results ranging from 21-100 are reliable and indicate the existence

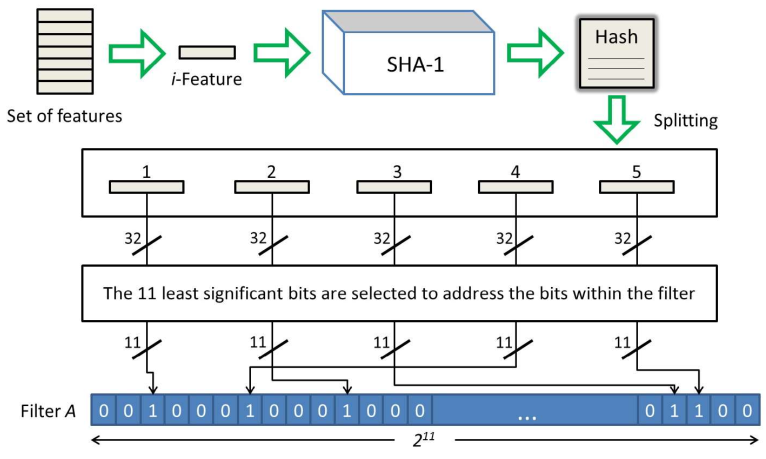

Figure 2. Process of mapping up to f features into each Bloom filter

of significant similarities between the objects. However, for simple object types, such as text, a 5 (five) score could be meaningful and worth further evaluation. The authors also highlight that a value of 100 does not guarantee that the objects are identical.

4. Similarity hashes and Digital Forensics

The sdhash tool has a great potential to be used in forensic investigations for identifying similar objects. Examiners can look in a seized media for objects of interest, being able to detect their targets even though they have been partially modified. Roussev [Roussev 2011] mention another application for this tool, which is finding embedded objects. In this case, we can look for a picture in a seized drive and find not only the picture itself but any occurrence of it even inside other objects.

However, besides the advantages and flexibility achieved with sdhash, its high cost compared to traditional approaches (hash functions) is a major problem. Although we can not estimate the cost of this tool easily, as it is file-dependent, we can calculate the number of features selected by looking at the metadata contained in the digest generated for the object [Roussev and Quates 2012]. For example, for a certain 1 MB object tested, a total of 15.981 features were used. As each feature needs to be hashed before inserted in the Bloom filter, a total of 15.981 hashes need to be performed. In this case, the sdhash will be 15.981 times more expensive than the traditional hash method (SHA-1). As the object size grows, so the number of hashes. Even small objects (1 MB) require thousands of hashes to build their representation. This shows how sdhash is expensive and the need for improving somehow its efficiency.

Another problem is the size of the similarity representation, as described by Roussev [Roussev and Quates 2012]. Hash functions produce a finite and well-known number of output bits, independent of object size. However, sdhash tool outputs a digest about 3% of the object size, according to empirical tests. This tool also requires more time to produce the digests, since it needs to create and consolidate several Bloom filters. Therefore, object size is a major problem in the use of such tool, as it can be a bottleneck in the triage process. For this reason, we propose a new way to deal with these problems by using statistical methods, presented in the next section.

5. Sampling in digital forensics

In order to reduce the time taken for an examiner to classify a seized media as worth or not for further and deeper analysis, we could reduce the amount of data actually processed. This could be done following the ideas presented by Garfinkel et al. [Garfinkel et al. 2010]. The authors show how to adapt the classical "Urn Problem without replacement" in the forensics context. They say that taking enough random samples from a set of objects, there is a good chance that this sample represents the entire dataset. The amount of samples required can be obtained from the following equation, which calculates the probability of missing one of the objects of interest.

$$p = \prod_{i=1}^{n} \frac{((N-(i-1))-M)}{(N-(i-1))} \tag{1}$$

Here N represents the total number of objects in the set, M the number of targets (objects of interest), and n the number of objects required to be sampled. We will use this idea combined with similarity hashes in our work to reduce the time for a triage in investigations. Our goal is to verify whether a media contains at least one object from a database of interest objects. If so, we can take this media for further analysis.

In the calculus, the variable N is the seized media size. As we work at the disk sector level, the value for N will be the number of media sectors. For determining the M value, we can use different approaches. The first one considers the case where we want to find a particular object, which is just using its size (in sectors) as M value. Other scenario consists in figuring out whether a seized drive contains any object of a database or not. In such case, we need to adapt the Urn Problem to this context, since we do not know whether the media contains evidences or not. We have to consider that at least one object of size M (estimated value) will be present in the media in order to comply with one of the basic assumptions of the problem: The presence of an object inside the "urn", or in our case, the media. Then, we calculate the formula using the estimated M value (in sectors) to obtain n for a chosen probability rate. This means that we need to take n random samples from the media and with probability $1-p$ we will find objects with this size or larger (we can adjust this value as needed, but the n value will change). We highlight that the examiner can control the M value according to the investigation, using the average size of the objects from the database or a common object size for the type of searched objects for that particular scenario. High values for M increase the chances of missing smaller objects but decreases the required triage time. Therefore, prior knowledge about the search can potentially help in determining the M value.

Using the adapted Urn Problem equation and controlling its error rate, examiners can look for objects using random samples with high probabilities of success. For example, considering a 2 TB seized disk (which contains approximately 500 million 4 KB sectors) and that we want to find a 100 MB object (25,000 sectors) on it. Using the Eq. 1 with N = 500,000,000, M = 25,000 and fixing a success rate of 99%, we need to take around 92,500 samples from the hard disk drive in order to find at least one fragment from the desired object.

6. Using sampling to reduce sdhash cost

In order to minimize the time to analyze a large volume of data, we propose an approach to combine the use of similarity hashes, using the sdhash tool, with sampling techniques. We will also work with object fragments (disk sectors) instead of whole objects since we want to encompass scenarios where the file system is corrupted, and no metadata about its structure is available. Besides, taking sectors directly from a hard drive give us the possibility of finding data that were deleted but some of its fragments were not overwritten.

However, using the sampling technique with object fragments can also be costly. The number of similarity hashes generated and of comparisons between the datasets increase as their sizes grow, even though we use samples. This is also compounded by the fact that we are working with fragments instead of whole objects (it increases the number of objects). With this in mind, we propose the use of clusters to decrease this cost.

6.1. Clustering approach

We propose the use of clusters to reduce sdhash cost. We take a determined number of sectors from the seized media being analyzed and gather them in a single object, which will have the similarity hash generated and will be used in the comparison. The fragments can and most of the times will be from different objects and yet the sdhash tool will be able to identify similarity due to its characteristics.

Our objective is to figure out whether a seized media contains objects of interest based on a comparison to a database, in the shortest possible time. To this end, we randomly sample sectors from the disk, gather them into clusters of j sectors and generate their similarity hash representation using sdhash. Then, we compare these digests to a database in order to find common interest objects.

One question that stands is how to take the samples. We propose two different ways to perform this operation: diffuse or contiguous. In the first one (Fig. 3), we select objects (sectors) randomly around the seized media and group them into clusters, which will be sdhashed and compared to a database of interest objects. The second method consists in taking contiguous fragments around some randomly chosen sectors in order to build the clusters (Fig. 4). It is important to highlight that for fragmented disks both methods seems to work fine, while for a defragmented one, the second method will not get fragments from many different files. In this work, we will only use the diffuse mode for sampling. The other method will be covered in future studies.

6.2. Experiments

We conducted experiments to validate our ideas using sdhash tool and sampling techniques. To this end, we simulated specific scenarios to measure the efficiency of this approach in each one, in order to find evidences in a triage process. Our goal is to find at least one object fragment from a database of interest objects present in the seized media being analyzed. If so, this media will be selected for further and deeper analysis.

We first compared our approach of using similarity hashes and sampling to the common one where a similarity representation is generated for every object in seized media. Then, we made experiments with the first approach exploring different scenarios and evaluating the impact of using the clustering method.

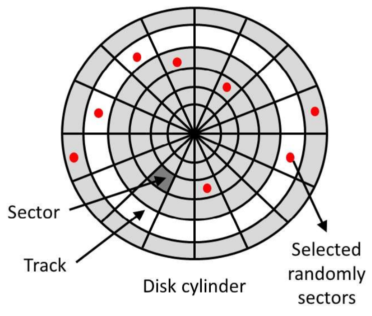

Figure 3. Diffuse method of selecting objects for a cluster.

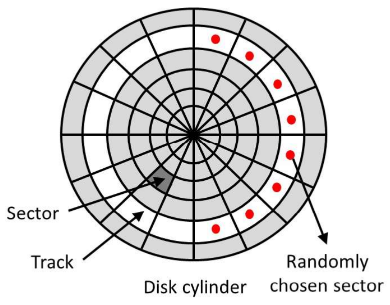

Figure 4. Contiguous method for clustering

In our experiments we selected two datasets and labeled them as DBIO (Database of Interest Objects) and SM (Seized Media). The DBIO is the largest one, with 2204 different files (218458 sectors), while the second have 549 (74929 sectors). The datasets were taken from the $t5$ corpus [Roussev 2011] and encompass different file formats: text, pdf, html, doc, ppt, jpg, xls and gif. For the purpose of our experiments, the DBIO and SM datasets have only four files in common (each one encompass several objects fragments), with the following format and size: F_1: doc (1039 sectors), F_2: ppt (469 sectors), F_3: text (87 sectors) and F_4: pdf (273 sectors). Our goal is to find at least one object in common in the shortest time possible. All files in the SM dataset were fragmented (broken into 4 KB objects) to simulate disk sectors. For the DBIO dataset, we disposed the objects in three ways to simulate different scenarios: fragmented (4 KB), whole file and image file.

We first evaluated the approach of using sdhash without sampling. For this end, we followed the steps below:

1. Creation of a similarity hash representation for every object in the DBIO.
2. Creation of a similarity hash representation for every object of the SM.
3. Comparison of similarity hashes between DBIO and SM objects.
4. Evaluation of the results.

The next experiment evaluated the inclusion of sampling in the process. The first step is to make a sample of the SM dataset. To this end, we used the approach described in section 5. We first estimated the object size we wanted to search and applied the Urn Problem equation using this value and the SM total size (74929 sectors). In our experiments, we chose as file size of 1039 sectors (the size of the greatest common file available in both datasets) and adopted a 99,9% success rate. Using the equation, we got a result of 550, which is the required number of samples we have to take from the SM.

Then, we made several experiments focusing on the described approach with different scenarios in order to evaluate the impact of clustering and determine the best conditions for using it. In general, we did our search following the steps below:

1. Creation of a similarity hash representation for every object in the DBIO.
2. Sampling the SM dataset.
3. Creation of a similarity hash representation for the samples/clusters of SM.
4. Comparison of similarity hashes between DBIO objects and SM samples/clusters.
5. Evaluation of the results.

Our first experiment involved comparing the objects from the SM with those in DBIO dataset in the fragment level, in order to simulate disk sectors (4 KB). We also conducted this experiment using our idea of clustering. We choose the cluster size c a value of 100 sectors since it was a reasonable value got from empirical tests. For larger values, we identified an increasing number of false positives (results pointing out similarity between unrelated files) while smaller ones did not present a significant reduction but only more comparisons to make. In the next experiment, we evaluated the use of whole files representing the database instead of their fragments. Again, we did it using clusters and without them. The last experiment compared the SM fragments to a DBIO image file. To this end, we created a similarity hash using the entire dataset in order to create a single representation of it. For default, the sdhash breaks large files in 128 MB pieces and creates a similarity hash for each one of them [Roussev and Quates 2013].

7. Discussion

The results of our experiments are presented in tables 1 and 2. In the first one, we present a comparison of sdhash with and without sampling, using DBIO and SM in the fragmented level. However, there is a difference between the maximum comparison number of both techniques (second column) to the expected value (16.368.839.482 without sampling and 120.151.900 with it). This is due to sdhash does not work with objects smaller than 512 bytes and the datasets have a few ones which do not have this minimum size.

When we analyze the impact of adding the sampling technique, a significant reduction in the number of comparisons is noticed, requiring about 136 times fewer operations. Also, we present the number of objects of interest found by each technique. Both found more objects than there really are. This exceeding value represents the false positives generated by them, minimized by the use of sampling.

The cost of each technique is presented in the last column of table 1. It is represented by the number of hashes functions (SHA-1) required to generate the representation of each dataset object. For the traditional approach using a hash-based method, the value is equal to the objects number: each one only requires a single hash to create its representation. However, a high cost is observed using sdhash (about 224 times more expensive than the traditional technique), which is minimized by the adoption of sampling (reduced to 167 times). This way, it is evident that using sdhash is very expensive and its combination with statistical methods becomes essential towards a practical approach. To this end, we focus on this combination and propose the use of clusters to reduce even more the costs, evaluating different scenarios in order to find the best ones in which we have the greatest reduction in costs.

Table 1. Experiments comparing sdhash with and without sampling on the fragment level, measuring their efficiency and the cost based on hash functions

Technique	#Max. Comparisons	#Objs of interest found	#Hashes functions (SHA-1)		
			DBIO	SM	Total
sdhash	16.343.700.471	9114 / 1868	48.947.005	16.791.207	65.738.212
sdhash & Sampling	120.031.450	62 / 22	48.947.005	124.979	49.071.984

We present in Table 2 the results of sdhash and sampling techniques for different scenarios, evaluating the use of our clustering approach. With this technique, the number of comparisons between DBIO and SM were reduced by a factor of c (size of the cluster). However, at the same time, in some experiments as cases #3 and #5, this technique is ineffective since no matches were found when comparing the object fragments with the DBIO in whole file and image file formats. The objects encompassing the cluster interfere in the creation of the similarity representation and thwart the comparison (the similarity among them will be slight). Only case #1 when the comparison is made considering the DBIO fragmented, we had significant results. Also, in the scenarios where we did not use cluster and consider the DBIO dataset in whole file and image file formats (#4 and #6), the comparisons decreased, but the number of false positives proportional to the comparisons is elevated in relation to the other scenarios (#1 and #2). Also, given an object fragment, it is harder to detect similarity comparing this small datum to a database image file as the size proportional between them increase. This makes our goal of identifying fragments of deleted data fail.

Table 2. Experiments with sdhash using different object formats

#Exp.	#Max. Comparisons	#Matches 20-100	Clusters	Database format	Objs of interest		#Features generated
					F_1	F_2	
1	1.309.434	107	Yes	Fragmented	11(4)/4	20(15)/18	48.985.337
2	120.031.450	238	No	Fragmented	17(4)/4	45(15)/18	49.071.984
3	12.870	0	Yes	Whole file	0/4	0/18	14.363.578
4	1.179.750	149	No	Whole file	5(4)/4	15/18	14.450.225
5	42	0	Yes	Image file	0/4	0/18	10.587.068
6	3850	288	No	Image file	4/4	17(15)/18	10.673.715

Table 2 also presents the total number of matches between the datasets according to a range of scores got from sdhash, where the results express the similarity between objects in the SM sample to those in the DBIO. We knew beforehand that there were only 22 objects in common between the datasets after the sampling, where 18 belongs to file F_1 and 4 to F_2. The other two objects (F_3 and F_4) had no fragments selected in the sample. We also show the number of matches restricted to the objects selected and the precision in the experiments on finding them, as well as the number of false positives by showing the total number of matches and the number of true positive (TP) matches (parentheses). For example, in experiment #2, object F_2 had 45 matches for only 18 TP possible. In the parentheses, we show the number 15 which is the number of TP matches. We highlight that only scores of similarity ≥ 20 were counted, since they are the ones significant (reliable) [Roussev and Quates 2012]. Object F_1 had 100% of cover in experiments #1, #2, #4 and #6, while F_2 a total of 83.34% for the same cases.

Comparing the expected number of matches to the one got from the total matches (third column), we can see a high number of false positives results (considering results with score ≥ 20). The elevated number will be analyzed in future studies, as well as measures to properly address them, but we believe that it happens due to common structure between objects of the same type. Furthermore, Young et. al [Young et al. 2012] mention that the cause can also be due to common NULL (0x00) blocks.

8. Conclusions

Digital forensic investigation is becoming a critical field as its techniques needs to scale to follow the fast increase in media storage capacity. In this work, we presented a proposal for using similarity hash tools, such as sdhash, combined with sampling techniques to reduce the time of a triage process. We showed how expensive sdhash is and possible ways to reduce the number of similarity hashes generated and of comparisons, using sampling techniques and clustering disk sectors. Besides the reduction of the number of comparisons and features generated, this approach showed effective in finding similar objects of interest and presented a smaller number of false positives. Also, our technique gives examiners more flexibility and allow them to find even objects that were deleted but some of its fragments remain on disk. We also showed limitations of clustering by evaluating different scenarios, which proved to be ineffective when comparing clusters to database image files and to whole files. On the other hand, we had good results comparing them to a fragmented database.

Although our experiments have shown that sdhash could identify objects of interest with a small set of data taken randomly from a seized media, there is a lack of understanding about the false positives generated and how to treat them, a subject that will be covered in a future work.

References

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426.

Garfinkel, S., Nelson, A., White, D., and Roussev, V. (2010). Using purpose-built functions and block hashes to enable small block and sub-file forensics. *Digital investigation*, 7:S13–S23.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97.

Martínez, V. G., Álvarez, F. H., and Encinas, L. H. (2014). State of the art in similarity preserving hashing functions. In *Proceedings of the International Conference on Security and Management (SAM)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Rabin, M. O. et al. (1981). *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Lab., Univ.

Roussev, V. (2010). Data fingerprinting with similarity digests. In *IFIP International Conf. on Digital Forensics*, pages 207–226. Springer.

Roussev, V. (2011). An evaluation of forensic similarity hashes. *Digital investigation*, 8:34–41.

Roussev, V. and Quates, C. (2012). Content triage with similarity digests: The m57 case study. *Digital Investigation*, 9:S60–S68.

Roussev, V. and Quates, C. (2013). sdhash tutorial: Release 0.8. `http://roussev.net/sdhash/tutorial/sdhash-tutorial.pdf`. Accessed 2016 Set 13.

Young, J., Foster, K., Garfinkel, S., and Fairbanks, K. (2012). Distinct sector hashes for target file detection. *Computer*, 45(12):28–35.