A Study on Approximate Matching for Similarity Search: Techniques, Limitations and Improvements for Digital Forensic Investigations

Vitor Hugo Galhardo Moia[1]
Supervisor: Marco Aurélio A. Henriques[1]

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

vhgmoia,marco@dca.fee.unicamp.br

Abstract. *With the rapid increase in data storage capacity, the use of automated procedures to handle the massive volume of data available nowadays is required by digital forensic practitioners. The Known File Filtering is one of the techniques employed to reduce/separate data (based on their hash values) for analysis, using a list of interest objects. However, due to the limitation of hashes in this scenario (inability to detect similar objects), new methods have been designed. These functions, called Approximate Matching (AM), are possible candidates to perform such a process because they can identify similarity in a very efficient way by creating and comparing compact representations of objects. In this work, we show how to perform the similarity search task (using AM) more efficiently using the Similarity Digest Search Strategies and perform a detailed analysis. Given the limitations found, we also propose a new strategy. Furthermore, we address some limitations of AM tools regarding the similarity detection, where many matches pointed out as similar, were indeed false positives. Another improvement made was in the comparison function of one of the most known AM tools after a theoretical assessment of it. We identified and mitigated some limitations, proposing a new and more precise similarity measurement. New applications of AM are also presented and analyzed: One for fast file identification (using sampling) and another for efficient fingerprint identification. This text is a summary of a Doctor of Science thesis presented and approved in Feb./2020 at the School of Electrical and Computer Engineering, University of Campinas, São Paulo, Brazil, and submitted to the Thesis and Dissertations Competition of SBSeg 2020.*

1. Introduction

Digital forensics is a branch of forensics aiming at investigating digital devices in the search for crime evidence. One particular problem of this field is that, due to technology improvements, storage devices' capabilities have increased significantly in the past years. The growth is a result of the popularity of digital devices that became more accessible to people due to a decrease in costs. This trend imposes a severe challenge on forensic practitioners, who, even in ordinary investigations, have to handle terabytes of digital evidence. Consequently, the time and effort to undertake analysis on seized devices remain a

challenge, forcing practitioners to explore solutions to handle the massive volume of data in a short time.

One possible approach to deal with the problem mentioned above more efficiently is the Known File Filter (KFF) method, which separates relevant from irrelevant information in prior analysis. Practitioners can use lists of interest objects to remove known good files from the analysis (objects of operating systems, known software, and other inoffensive ones in a white list) and/or separate bad ones (illegal or suspicious objects in a blacklist). Cryptographic hash functions (e.g., MD5, SHA-1, SHA-2, etc.) are a straightforward technique to perform KFF. However, hashes do not perform well in this scenario since even small changes in an object will produce an entirely different hash, which will make the correlation of the original object and its modified version almost impossible.

Suitable candidates to mitigate hash limitations are the Approximate Matching (AM) functions. Using a small and compact data representation (digest), they can identify the similarity between objects in a way that similar objects will have similar digests; small changes in the input reflects in minor variations in the digest. When comparing two digests, AM produces a score related to the amount of content shared between them.

The downside of AM is that they are computationally more expensive than hash functions. Comparing two digests is not as straightforward as the string comparison of hashes. Specifically, AM requires a particular function to determine the similarity of two digests that is more complex and tool-dependent. Besides, current functions present other limitations regarding compression rates and/or precision. Each solution usually focuses on addressing satisfactorily one of these aspects to the detriment of others. Another obstacle of the AM adoption for KFF is that the straightforward comparison solution usually adopted is the brute force (all-against-all). Every reference list digest is compared to every digest created for the target device under analysis in this mode. This way, the complexity of this search is quadratic, and the whole process becomes very time-consuming.

Given the limitations of the field, new solutions are required to handle the massive volume of data found in investigations nowadays. In this work, we present new solutions to deal with current AM limitations. We show how digital forensic practitioners could benefit from using such functions and how to use them to perform a similarity search, where large data sets are compared to identify similar objects. We estimate the (theoretical) minimum similarity detected by AM functions and improve current AM tools to perform better over large data sets and produce more reliable results. Finally, we present new applications for AM functions.

2. Background

The main concepts in the Approximate Matching field are briefly presented in this section.

2.1. Approximate matching

NIST [Breitinger et al. 2014b] defines approximate matching functions (AM) as a *"Promising technology designed to identify similarities between two digital artifacts. It is used to find objects that resemble each other or find objects that are contained in another object"*. They can be classified according to their operational level. The first level is bytewise, which relies on the byte sequence of the object. The syntactic considers some object's internal structure, while the semantic tries to interpret the object and extract some

contextual attributes. Our work focuses on the bytewise level for its interesting characteristics: Format independence and efficiency. We highlight it is not the focus of this work to deal with encrypted data, memory dumps, or malware since this sort of data requires extra procedures before we can use AM tools to search for similar content.

There is a vast range of applications for AM. One can use it to identify new versions of documents and software, embedded objects (e.g., jpg file inside a word document), objects in network packets (without reconstructing the packet flow), locate variants of malware families, code reuse (intellectual property protection and/or bug detection), and so on [Roussev 2011, Harichandran et al. 2016].

We can find many tools implementing AM concepts to perform the similarity identification using digests at the byte level. The more promising ones and target of our research are `ssdeep` [Kornblum 2006], `sdhash` [Roussev 2010], `mrsh-v2` [Breitinger and Baier 2013], and `TLSH` [Oliver et al. 2013].

2.2. Strategies for digest similarity digest search

The major bottleneck in digital forensic investigations when performing KFF based on AM is the similarity digest search. An examiner, who usually has a reference list containing objects of interest, needs to compare each object from this set to each one obtained from the target system under analysis. The goal here is to find similar objects, which can be efficiently done using one of the AM tools. However, the usual brute force mode performed in most investigations (an all-against-all comparison) could be too time-consuming when dealing with large data sets.

To cope with this problem, researchers have proposed techniques aiming to reduce the time involved in the similarity digest search, which we refer to Similarity Digest Search Strategies (SDSS). They are efficient approaches for comparing large data sets of objects. Some strategies proposed in the literature are `F2S2` [Winter et al. 2013], `MRSH-NET` [Breitinger et al. 2014a], and `HBFT` [Breitinger et al. 2014c].

3. FSDS: A new Similarity Digest Search Strategy

In this section, we present our first contributions to the field, related to the Similarity Digest Search Strategies (SDSS). They are an efficient alternative to brute force, which is too time-consuming and even prohibitive for dealing with large data sets. More details about our results can be found in Chapters 3 and 4 of the thesis. We performed a comparison of current SDSS to point out strengths and weaknesses [Moia and Henriques 2017d] [Moia and Henriques 2017a] [Moia and Henriques 2017c]. We also provided a detailed analysis of some strategies' operational costs, showing how they scale with different data set sizes and which performed best. Our results demonstrated that even though some strategies outperform others in some aspects, they fail in others. For instance, the `F2S2` strategy has low memory consumption but fails in the lookup complexity. The same happens for other strategies with different metrics. In conclusion, there is no suitable approach satisfying the most relevant requirements.

Another problem with current SDSS is that they are specific for a particular AM tool, being `ssdeep` and `sdhash` the most predominant choices. Other tools with interesting characteristics for investigations do not have better forms to perform the similarity search other than the brute force. For this reason, we propose the Fast Similarity Digest

Search (FSDS)[1] strategy [Moia and Henriques 2017b], aiming to perform efficiently over large amounts of data using the TLSH tool, which presents interesting characteristics for investigations, such as robustness to random changes and adversarial manipulations and space and time efficiency.

FSDS uses a particular kind of hash table to store digests, consisting of a central array (main table) and buckets. Each position in the main table, referred to as a bucket, can keep multiple entries composed by one field only, the ID, which links to the corresponding digest. When storing an object into its structure, the strategy first creates its digest using *TLSH*. Then it uses a distance function to map each digest to a position in the main table according to their distance to a reference point. It is expected that similar objects are mapped to near positions in the main table. This way, when performing a search, one needs to calculate a TLSH digest for the given object and compute its distance to the reference point. The resulting value will point to a position in the table where similar objects will be located. Using the TLSH comparison function with all digests located in the given position and with others stored around it (in the r near buckets), we can figure out whether or not similar items reside in the structure. We performed experiments showing the efficiency of our approach compared to brute force in three different scenarios, where a reduction of about 95% in time was observed with a minimum impact on recall (from 100% to 85%). *FSDS* also presented a small memory requirement, only 375 KiB to store 4457 objects (compression of 99,98%).

4. The impact of common blocks on Approximate Matching

The next contribution to the field is the improvement of the similarity detection process of AM tools and hence their effectiveness in producing valid results, as shown in Chapter 5 of our thesis. By simulating real-world investigations, we figured out that many matches pointed out as similar by current solutions were not similar when we visually inspected them and looked for content in common, such as paragraphs, figures, tables, and other elements created by users. We found out that matches were indeed a result of common blocks, i.e., common structures found in many objects of the same and different types. Since these structures may repeat in many different files, they are not suitable for assessing similarity in some contexts. Examples are header/footer information, color palettes, font specifications, or other data structures belonging to particular software vendors. Table 1 shows some of the blocks that repeated the most across different file types and includes its hash (FNV-1a), number of files having the block, and a brief description of its content.

In our work [Moia et al. 2020], we discuss how common blocks are spread across different objects and show their frequency. We came up with a classification of the matches according to the kind of similarity detected (e.g., user-generated content, application-generated content, and template content) and proposed a new version of sdhash that removes the common data among objects from the similarity digest, called NCF_sdhash[2]. Our results showed a significant reduction in the number of matches. Table 2 presents some of our findings. The number of matches is drastically decreased when changing the N value, responsible to define when a block should be considered common.

In a followed work [Moia et al. 2019], we measured how precision/recall rates are

[1]GitHub page: https://github.com/regras/fsds
[2]GitHub page: https://github.com/regras/cbamf

Table 1. Most repeated blocks per file type and their content.

File type	FNV-1a hash	Occurrences per number of files	Content
doc	c5e7aeb2482c56c0	442 / 533	Necessary stream of compound files, specific of Microsoft Office Word documents.
ppt	ef9a5a76d0df0c16	357 / 368	Part of a document summary information stream with application defined properties.
pdf	d5fb4ee41392d833	347 / 1,073	Piece of an indirect object of a pdf stream, belonging to RGB color space.
xls	b3310ce89e000aa4	226 / 250	Font specification.
jpeg	f0a05cdcac5796d4	108 / 362	RGB color palette.
html	cbac5aaf609ccf54	61 / 1,093	Sample of a well-known piece of java script code to make web pages have rollover images.
text	69c06bea6c3a3f10	18 / 711	Part of a template content.
gif	c91811dfd69ce32b	5 / 67	Global color table, which is a sequence of bytes representing RGB color triplets.

affected by common blocks and show how the recommended threshold score to identify similar matches used by sdhash is affected. Figs. 1 and 2 show one of the scenarios discussed in the paper, where we want to find all matches related to user-generated content and template. We can observe a small decrease in precision for NCF_sdhash (especially with $N = 20$) compared to a significant increase in recall as we decrease the threshold. The impact of such a reduction for sdhash shows how common blocks affect (negatively) the similarity assessment.

Table 2. Number of file matches by score range using sdhash and NCF_sdhash for ALL file types, discarding common features with occurrences $>$ N.

Score	sdhash	NCF_sdhash for N					
		3	5	10	20	50	100
= 1	2,992	65	93	152	195	253	311
≥ 1	9,220	409	622	1,188	1,541	2,123	2,371
≥ 10	1,795	241	356	745	963	1,249	1,262
≥ 21	1,038	181	267	563	799	925*	925*
≥ 50	459	79	114	237	414	475	472
≥ 90	86	20	21	55	58	85	85
= 100	18	6	6	15	15	30	30

5. Mitigating sdhash limitation with Jaccard Distance

In this section, related to Chapter 6 of the thesis, we present the results of a theoretical study of the sdhash detection capabilities. We show the conditions for which it works efficiently and the ones where it struggles. We figure out that, in general, the larger the object, the better the performance of sdhash; it requires 0.50% of similar content between two objects to detect similarity. On the other hand, it requires about 35% of similar data to produce valid scores for small objects. By identifying sdhash's limitation (inconsistency when dealing with small objects), we proposed a change in its comparison function. We developed an improved version of it where we use the Jaccard Similarity

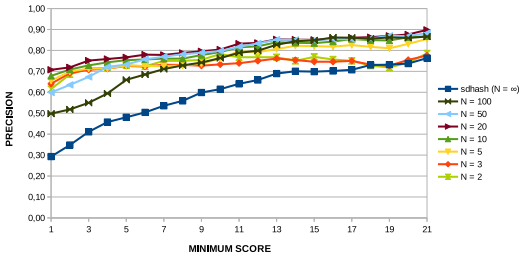

Figure 1. Common blocks removal: Precision vs. score

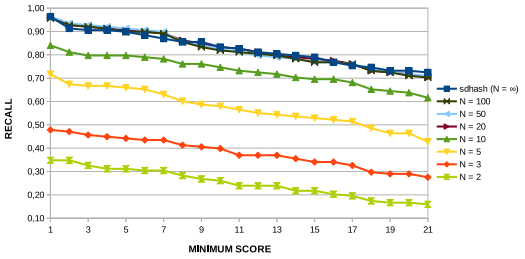

Figure 2. Common blocks removal: Recall vs. score

instead, called `J-sdhash`[3]. Our results showed that the new version generated an easier to interpret score (percentage value) that can be adjusted to work in different scenarios. Besides, the tool works well independent of the object size and, differently from other tools, its score reflects the real amount of bytes in common between objects.

Some of our findings can be found in Table 3. Here, we present recall and precision rates of `sdhash` and some versions of `J-sdhash`, where we change one of its internal parameters controlling the minimum percentage of commonality to the tool to detect similarity between objects. We also present the results for a new metric referred to as $diff$, an average value that measures how far the score produced by AM functions are from the real similarity found; the lesser the value, the better.

Metric	**sdhash**	**J-sdhash**			
		γ=30%	**γ=20%**	**γ=15%**	**γ=10%**
Recall	89.0%	90.91%	93.18%	93.18%	95.45%
Precision	100.0%	100.00%	36.94%	10.87%	1.93%
$diff$ **(all matches)**	4.16	1.55	1.35	0.86	1.65
$diff$ **(only true matches)**	4.16	1.55	1.19	1.20	1.31

Table 3. `J-sdhash` settings varying the minimal percentage of common content of similarity in the beginning of objects (44 similar matches).

6. New applications for Approximate Matching functions

Our last contribution to the field was towards new applications for AM functions, as presented in Chapter 7 of the thesis. We show how to use AM combined with sampling techniques for fast file identification [Moia and Henriques 2016]. We present how `sdhash` could be used and possible ways to reduce its overall costs for investigations using sampling techniques and clustering disk sectors. Besides reducing the number of comparisons and features generated, the new approach showed effectiveness in finding similar objects of interest and presented a lower false positive rate.

We also show how to apply AM for fingerprint identification where individuals can be identified over large data sets [Moia and Henriques 2018]. Our proposal, the `MCC-HBFT`[4], is a new fingerprint identification strategy that leverages the efficiency of

[3]GitHub page: `https://github.com/regras/J-sdhash`
[4]GitHub page: `https://github.com/regras/mcc-hbft`

the AM field and the accuracy of the state-of-the-art fingerprint representation MCC. We showed how our strategy works and outperforms a commonly used fingerprint indexing approach on public domain databases.

7. Conclusions

Digital forensic investigations suffer from the massive amount of data available nowadays, where time is a scarce resource. In this work, we presented Approximate Matching (AM) functions as a candidate to deal with such a problem when performing the similarity search. We presented the main concepts of AM and the limitations of current solutions. Our contributions to the field focused on new solutions to mitigate current limitations, as described below.

- Comparison of the similarity digest search strategies, pointing out their strengths and weaknesses. Identification of limitations of the field.
- Proposal of a new similarity digest search strategy, called `FSDS`.
- Proposal of a new way to identify and remove common blocks from the similarity assessment of AM, showing how they affect digital forensics investigations.
- Theoretical analysis of `sdhash` and identification of limitations in the similarity assessment process. Proposal of `J-sdhash` to mitigate these limitations.
- Proposal of new applications for AM: Fast file identification (using sampling techniques) and Fingerprint identification (when dealing with larger data sets).

All these contributions are also documented in the papers published in national scientific conferences and some renowned international journals, as pointed out along this paper. Besides, a copy of the thesis described in this paper can be found in `http://repositorio.unicamp.br/jspui/handle/REPOSIP/346315`.

Acknowledgment

This work was partially supported by CAPES FORTE project (23038.007604/2014-69).

References

[Breitinger and Baier 2013] Breitinger, F. and Baier, H. (2013). Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In *ICDF2C 2012, Lafayette, IN, USA*, pages 167–182, Berlin, Heidelberg. Springer.

[Breitinger et al. 2014a] Breitinger, F., Baier, H., and White, D. (2014a). On the database lookup problem of approximate matching. *Digital Investigation*, 11:S1–S9.

[Breitinger et al. 2014b] Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., and White, D. (2014b). Approximate matching: definition and terminology. *NIST Special Publication*, 800:168.

[Breitinger et al. 2014c] Breitinger, F., Rathgeb, C., and Baier, H. (2014c). An efficient similarity digests database lookup-a logarithmic divide & conquer approach. *The Journal of Digital Forensics, Security and Law: JDFSL*, 9(2):155.

[Harichandran et al. 2016] Harichandran, V. S., Breitinger, F., and Baggili, I. (2016). Bytewise approximate matching: The good, the bad, and the unknown. *The Journal of Digital Forensics, Security and Law: JDFSL*, 11(2):59.

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

[Moia et al. 2019] Moia, V. H. G., Breitinger, F., and Henriques, M. A. A. (2019). Understanding the effects of removing common blocks on approximate matching scores under different scenarios for digital forensic investigations. In *XIX Brazilian Symposium on information and computational systems security*, pages 1–14, São Paulo-SP, Brazil. Brazilian Computer Society (SBC).

[Moia et al. 2020] Moia, V. H. G., Breitinger, F., and Henriques, M. A. A. (2020). The impact of excluding common blocks for approximate matching. *Computers & Security*, v.89:1–11. Available at: https://doi.org/10.1016/j.cose.2019.101676.

[Moia and Henriques 2016] Moia, V. H. G. and Henriques, M. A. A. (2016). Sampling and similarity hashes in digital forensics: An efficient approach to find needles in a haystack. In *XVI Brazilian Symposium on information and computational systems security*, pages 693–702, Niteroi-RJ, Brazil. Brazilian Computer Society (SBC).

[Moia and Henriques 2017a] Moia, V. H. G. and Henriques, M. A. A. (2017a). A comparative analysis about similarity search strategies for digital forensics investigations. In *XXXV XXXVII Brazilian Symposium on Telecommunications and Signal Processing (SBrT 2017)*, pages 462–466, Sao Pedro, Brazil.

[Moia and Henriques 2017b] Moia, V. H. G. and Henriques, M. A. A. (2017b). Fast similarity digest search: A new strategy for performing queries efficiently with approximate matching. In *XVII Brazilian Symposium on information and computational systems security*, Brasilia-DF, Brazil. Brazilian Computer Society (SBC).

[Moia and Henriques 2017c] Moia, V. H. G. and Henriques, M. A. A. (2017c). An operational costs analysis of similarity digest search strategies using approximate matching tools. In *XVII Brazilian Symposium on information and computational systems security*, Brasilia-DF, Brazil. Brazilian Computer Society (SBC).

[Moia and Henriques 2017d] Moia, V. H. G. and Henriques, M. A. A. (2017d). Similarity digest search: A survey and comparative analysis of strategies to perform known file filtering using approximate matching. *Security and Communication Networks*, 2017.

[Moia and Henriques 2018] Moia, V. H. G. and Henriques, M. A. A. (2018). A new similarity digest search strategy applied to minutia cylinder-codes for fingerprint identification. In *XVIII Brazilian Symposium on Information and Computational Systems Security*, pages 99–112, Natal-RN, Brazil. SBC.

[Oliver et al. 2013] Oliver, J., Cheng, C., and Chen, Y. (2013). TLSH–a locality sensitive hash. In *Cybercrime and Trustworthy Computing Workshop*, pages 7–13. IEEE.

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

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

[Winter et al. 2013] Winter, C., Schneider, M., and Yannikos, Y. (2013). F2s2: Fast forensic similarity search through indexing piecewise hash signatures. *Digital Investigation*, 10(4):361–371.