THE JOURNAL OF
DIGITAL FORENSICS,
SECURITY AND LAW

Journal of Digital Forensics,
Security and Law

Volume 11 | Number 2

Article 6

2016

Towards Syntactic Approximate Matching - A Pre-Processing Experiment

Doowon Jeong
Korea University

Frank Breitinger
University of New Haven

Hari Kang
Korea University

Sangjin Lee
Korea University

Recommended Citation

Jeong, Doowon; Breitinger, Frank; Kang, Hari; and Lee, Sangjin (2016) "Towards Syntactic Approximate Matching - A Pre-Processing Experiment," *Journal of Digital Forensics, Security and Law*: Vol. 11 : No. 2 , Article 6.
DOI: https://doi.org/10.15394/jdfsl.2016.1381
Available at: https://commons.erau.edu/jdfsl/vol11/iss2/6

TOWARDS SYNTACTIC APPROXIMATE MATCHING – A PRE-PROCESSING EXPERIMENT

Doowon Jeong[1], Frank Breitinger[2], Hari Kang[1] and Sangjin Lee[1]

[1]Center for Information Security Technologies (CIST)
Korea University, 145 Anam-ro, Seongbuk-Gu, Seoul, Republic of Korea
{dwjung77, imstrong89}@gmail.com, {sangjin}@korea.ac.kr

[2]Cyber Forensics Research & Education Group (UNHcFREG)
Tagliatela College of Engineering, ECECS
University of New Haven, 300 Boston Post Rd, West Haven, CT 06516
{FBreitinger}@newhaven.edu

ABSTRACT

Over the past few years, the popularity of approximate matching algorithms (a.k.a. fuzzy hashing) has increased. Especially within the area of bytewise approximate matching, several algorithms were published, tested, and improved. It has been shown that these algorithms are powerful, however they are sometimes too precise for real world investigations. That is, even very small commonalities (e.g., in the header of a file) can cause a match. While this is a desired property, it may also lead to unwanted results. In this paper, we show that by using simple pre-processing, we significantly can influence the outcome. Although our test set is based on text-based file types (cause of an easy processing), this technique can be used for other, well-documented types as well. Our results show that it can be beneficial to focus on the content of files only (depending on the use-case). While for this experiment we utilized text files, Additionally, we present a small, self-created dataset that can be used in the future for approximate matching algorithms since it is labeled (we know which files are similar and how).

Keywords: Bytewise Approximate Matching, Pre-processing, Syntactic Similarity, Digital forensics.

1. INTRODUCTION

Nowadays, one of the biggest challenges in the digital forensic investigation process is that examiners are overwhelmed with data – a forensic case can easily consist of several 100 GBs. Finding the few relevant files for a specific case resembles finding a needle in a haystack. A common procedure for reducing the amount of data is known file filtering where files can either be filtered in (suspicious files) or filtered out (irrelevant files). Traditionally, this is solved using cryptographic hash functions which are very efficient but have the drawback of only identifying exact duplicates. In order to overcome this drawback, the community came up with approximate matching, which allows investigators to calculate the probability of similarity between two or more similar objects, such as media, stream, and other files. According to the definition from Breitinger, Guttman, McCarrin, Roussev, and White (2014), "approximate matching methods can be placed in one of three main categories":

Bytewise matching focuses on the complete underlying byte sequence that make up and digital object.

Syntactic matching is similar to bytewise but considers internal structures of the input, e.g., ignoring the header of a network packet.

Semantic matching operates on the contextual level and is therefore close to the cognitive abilities of humans, e.g., two images can be identical but have different byte structure (PNG vs. JPG).

Current research focuses on either the bytewise or the semantic level where especially the former approach comes with a significant problem. While bytewise approximate matching shows high reliability and accuracy for randomly generated inputs (Breitinger, Stivaktakis, & Roussev, 2014), there are drawbacks with real world scenarios. As discussed by Garfinkel and McCarrin (2015), many inputs have 'common blocks' which yield to *undesired matches*. Note, these are not false positives since there is similarity, however these are undesired matches since they are not wanted from an investigator perspective. For instance, "the most common block is the block of all NULLs, which is used to initialize blank media and is also found in many document and database files." Thus, the benefits for a practical investigation environment are rather low. In other words, existing metadata of inputs can lead to undesired results.

In this paper, we analyze the impact and effectiveness of the pre-processing of inputs where pre-processing separates the content from its metadata. Our tests focus on the four common file formats EML (e-mails), PDF, DOC and HTML. Our experimental results show that pre-processing can significantly impact the results of bytewise approximate matching. Additionally, we present a small self-created dataset which can be used for testing purposes and be downloaded from http://forensic.korea.ac.kr/prepocessing_testset.

The rest of the paper is organized as follows: The background and related work is discussed in Sec. 2. Next, we briefly explain the problem as well as the concept of our idea. In Sec. 4 we show our results.

2. BACKGROUND AND RELATED WORK

The usage of approximate matching with digital forensics (a.k.a. similarity hashing or fuzzy hashing) is a rather new domain and probably started with an idea from Kornblum (2006) named context-triggered piecewise hashing. Subsequently, a small community emerged around that field and presented new algorithms, finally coming up with a definition in 2014 – Special Publication 800-168 from the National Institute of Standards and Technology (NIST, Breitinger, Guttman, et al. (2014)).

As outlined in the introduction, the definition divides approximate matching algorithms into three categories: semantic, syntactic and bytewise matching. While there are several implementations for semantic and bytewise algorithms, the syntactic area is rather untouched. Syntactic algorithms operate on byte sequences but have the ability to consider the internal structure of inputs. For instance, syntactic approaches may ignore header / footer information or HTML-tags.

2.1 Bytewise approximate matching algorithms

Currently, there are three major implementations for bytewise approximate matching, which we will briefly discuss:

`ssdeep` was presented by Kornblum (2006) and is based on the concept of context-triggered piecewise hashing (called fuzzy hash). The idea is to divide an input into chunks, hash each chunk and concatenate the chunk-hashes. The peculiarity of `ssdeep` is that instead of dividing the input into chunks of an equal size (as it is done by `dcfldd`[1]), the implementation utilizes a rolling hash that slides through the input, byte by byte, and considers the current context (7 bytes). If the rolling hash matches a specific requirement, the end of a chunk is found.

`sdhash` was introduced by Roussev (2010). In contrast to `ssdeep`, this approach selects statistically improbable features in an input (sequences of 64 bytes) and hashes these features using SHA1. The SHA1 digests are then split into 5 sub-hashes where each sub-hash sets one bit in a Bloom filter[2] (Bloom, 1970). Each Bloom filter can hold a fixed number of features. Once this maximum is reached, a new filter is created and filled. Thus, the final similarity digest of a file is a sequence of 1 or more Bloom filters.

`mrsh-v2` is a combination of both aforementioned implementations and was published by Breitinger and Baier (2013). Like `ssdeep`, this approach utilizes a rolling hash to divide the input into chunks. After identifying all chunks, it stores the chunk hashes into Bloom filters.

Several analyses and comparisons of these algorithms showed that (a) `ssdeep` can be overcome by an active adversary (Baier & Breitinger, 2011) and (b) `sdhash` is slightly slower but more precise than `mrsh-v2` (Breitinger, Stivaktakis, & Roussev, 2014).

2.2 Approximate matching for digital investigations

The usage of approximate hash based matching (AHBM) for digital forensics and "how AHBM can be applied in digital investigations" was analyzed by Bjelland, Franke, and Årnes (2014). In their paper the authors discuss three modes of operation: Search, Streaming and Clustering.

Search is the traditional approach and des-cribes the scenario where an investigator is looking for matches of inputs against a set / database (the authors call them leads), e.g., compare file A against the set / database. *Streaming* mode describes the situation where continuous data is passed to the system (e.g., in case of an intrusion detection system). The difference between Search and Streaming mode is that in the former case the search space is rather large while in the latter case the search space should be small to work efficiently. *Clustering* mode can be utilized to organize a set and cluster similar data. Note, in all cases it requires $N \times M$ comparisons where N is the input size (e.g., network packets of the intrusion detection system) and M is the search space[3]. This also reveals a current problem

[1] http://forensicswiki.org/wiki/Dcfldd (last accessed July 25th, 2016).

[2] Bloom filters are probabilistic data structures that are commonly used to represent sets. A detailed presentation of Bloom filters is beyond the scope of this paper but information can easily be found online, e.g., Farrell, Garfinkel, and White (2008).

[3] For clustering mode $N = M$.

approximate matching faces – the runtime efficiency for large sets is impractical.

2.3 E-mail structure and their similarity

This subsections briefly explains the structure of e-mails followed by an explanation of similarities between them.

EML is a file extension for an e-mail message saved to a file[4] and is made up of a header and a text body text written by a user.

An e-mail header is made up of different fields, e.g., Message-ID, Date, From, To, Subject, Cc, Mime-Version, Content-Type, Content-Transfer-Encoding, Bcc, X-Header. X-Headers are additional personalized information in the header that can be added. An example is given in the following:

```
Message-ID: <23335327.1075851772982.JavaMail.ev
ans@thyme>
Date: Sun, 7 Jan 2001 15:07:00 -0800 (PST)
From: gwdorsey@aol.com
To: jeffrey.a.shankman@enron.com
Subject: Re: fert
Cc: whalley@enron.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: whalley@enron.com
X-From: GWDORSEY@aol.com
X-To: Jeffrey.A.Shankman@enron.com
X-cc: whalley@enron.com
X-bcc:
X-Folder: \Jeffrey_Shankman_Nov2001\Notes Folde
rs\All documents
X-Origin: SHANKMAN-J
X-FileName: jshankm.nsf
```

As indicated by Bjelland et al. (2014), "the majority of the resulting matches fell into one of these three manually defined categories:

- Reply: When the number of matches where either e-mail is a follow up on another e-mail.
- Similar conversation: When e-mails with different subjects and content, sent to and from the same set of e-mail addresses.
- Different header: When identical e-mails found in different folders."

3. PROBLEM AND CONCEPT DESCRIPTION

Bytewise approximate matching considers the complete input and works very precisely – even small commonalities are usually identified and lead to a match. While this is a desired property, it may also lead to unwanted results. For instance, almost all files include metadata such as headers, footers, file signatures, information of options, structural information and so on, which is often common independent of the actual file content. In other words, metadata *can* influence the matching process and lead to unwanted results. We claim that for investigation purposes it can be beneficial to pre-process inputs, i.e., separating the metadata from the content.

For our testing, we use two independent test sets that are analyzed separately – one set consisting of DOC, PDF, HTML (more details see Sec. 4.1) and a second one consisting of EML files (see Sec. 4.2). Both sets contain metadata and were analyzed using the existing, well-established bytewise approximate matching tool `sdhash`[5] v3.4 from Roussev (2010) in regular mode, with a pre-processing step.

[4] See RFC2822 from Resnick (2001) and RFC1521 from Borenstein and Freed (1993).

[5] http://roussev.net/sdhash/sdhash.html (last accessed July 25th, 2016).

During our experiment we first compared the sets directly using `sdhash` and second, we applied a pre-processing to separate the metadata before running the comparison with the identical algorithm. We compared our results based on the parameters:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Fscore = 2 * \frac{Precision * Recall}{Precision + Recall}$$

where TP are true positives, TN are true negatives, FP are false positives and FN are false negatives. The F-score is the harmonic mean of precision and recall, so it can be seen as a weighted average of the precision and recall[6]. According to the tutorial from Roussev[7], "when applied to simple file types, such as text, scores as low as 5 *could* be significant." Therefore, a score of $5 \leq score \leq 100$ implies a positive match (TP and FP). The challenging part is to categorize the matches into one of the categorizes which is discussed in the corresponding subsection.

Note, we are not claiming that content matches are more important than metadata matches or vice versa. As will be discussed later, there can be scenarios where these matches are desired, e.g., identify photos that were taken with the same camera. However, we argue that investigators should be aware of the differences of results (pre-processing vs. no pre-processing).

We want to point out that if this pre-processing step can be done based on the byte level, this defines it as syntactic approximate matching[8]. Although for some of our tested file types the pre-processing is not based on the byte level, we will use the terms bytewise and syntactic approximate matching for the remainder of this paper.

4. EXPERIMENTAL RESULTS

This section discusses our experimental results. In Sec. 4.1 we describe our findings for the DOC, PDF and HTML. The following two sections present the results for EML where Sec. 4.2 describes the content based matches and Sec. 4.3 the metadata matches.

4.1 Assessment for DOC, PDF and HTML (content)

The first paragraph will detail our test-set while the second paragraph will outline our results.

Test-set description. Our first test set consists of three file types: DOC, PDF and HTML and is used to test the impact of pre-processing among different text file formats.

The procedure was as follows: (1) randomly select content for 15 files (.txt files) from the Open American National Corpus (OANC[9]), (2) manually copy and modify the content according to Table 1 (column one), (3) manually copy the modified version again and modify it (column two) and (4) replicate the files in all three formats which results in 45 files total (15 DOC, 15 PDF and 15 HTML).

For instance, 1-1 is the modified version of file 1; 1-2 is the modified version of file 1-1.

[6] https://en.wikipedia.org/wiki/F1_score (last accessed July 25th, 2016).

[7] http://roussev.net/sdhash/tutorial/03-quick.html (last accessed July 25th, 2016).

[8] As by the definition, syntactic approaches take file type specific structures into account but also work on the byte level, e.g., ignore the header of a TCP packet.

[9] Open ANC is a massive electronic text dataset written in American English and available at http://www.anc.org/data/oanc/download (last accessed July 25th, 2016)

Table 1. The modification details of DOC, PDF and HTML test set.

No.	First modification details		Second modification details	
1	1-1	removed table format	1-2	removed partial contents
2	2-1	removed table format	2-2	adjusted font style (size, bold)
3	3-1	changed a two-column layout to one column	3-2	removed partial contents and adjusted font style (bold)
4	4-1	changed a two-column layout to one column	4-2	removed partial contents and adjusted font style (bold)
5	5-1	removed partial contents and background color	5-2	changed template like item no. 7's modification
6	6-1	removed partial contents and background color	6-2	changed template like 9s'
7	7-1	removed partial contents and changed template like 5s'	7-2	changed template like 9-1s'
8	8-1	changed template like 3s'	8-2	removed and added partial contents
9	9-1	added table and changed template	9-2	removed partial contents
10	10-1	adjusted font style (italic, underline, bold) and removed partial contents	10-2	adjusted template like 9-1s' and contents' order
11	11-1	removed hyperlinks	11-2	adjusted font style (font, color)
12	12-1	adjusted font style (font, color)	12-2	removed hyperlinks
13	13-1	changed template	13-2	changed template
14	14-1	changed template like 13-1s'	14-2	changed template like 13-2s'
15	15-1	adjusted paragraph setup and added background and watermark	15-2	changed template and removed partial contents

On the other hand `file1.pdf`, `file1.html` and `file1.doc` share the same content but have a different file type. As all files are available in three types, there are 135 files (= 3 types × 3 modifications × 15 files) in total within the test set. We know that this is a small number, however, it provides a first outlook on how syntactic approaches can impact the results.

Pre-processing. In this case the pre-processing is done using Apache Tika[10] which can extract the content of text based files. Although we only created PDF, HTML, and DOC, Tika supports many other file types too and is open source.

As a very last step, we replaced identical consecutive spacing characters (i.e., 2 or more spaces / newlines) by a single one, e.g., two space are replaced by a single one. A space followed by a newline will remain as is.

Classification of the results. Since we manually created these files, we know which files are similar and should produce matches. Thus, calculating precision, recall and accuracy is straight forward.

[10] Tika, Apache Software Foundation, https://tika.apache.org (last accessed July 25th, 2016).

Table 2. Results of sdhash with and without pre-processing.

	*O*riginal set	*P*re-processed set
TP	71	433
TN	7614	8507
FP	586	0
FN	744	105
Precision	0.10807	1.00000
Recall	0.08712	0.80483
Accuracy	0.85247	0.98839
F-score	0.09647	0.89186

Test results. The all-against-all comparison of the results in 9045 comparisons (self-comparisons are eliminated). A summary of the results is given in Table 2. As can be seen, there is a huge difference for all the numbers.

The Original files yielded $(71 + 586 =)657$ matches while 433 matches were found in the *P*re-processed set. The reason for the false positives in *O*riginal set is the common bytes among the same format files, e.g., word documents share large low-entropy sequences in their header. It is notably that all positives matches (true positives and false positives) in the *O*-set are based on matches of the same file type. This results from a different encoding scheme of content text throughout the analyzed file-formats. Another eye-catching fact is that there are zero false positives in the *P*-set which results in a precision of 1.00. The number of false negatives is about 7 times lower for the *P*-set and thus also significantly better.

Next, we focus on the distribution of the similarity score for the matches (true positives + false positive). The true positives and false positive for the *O*-set are depicted in Fig. 1 and Fig. 2.

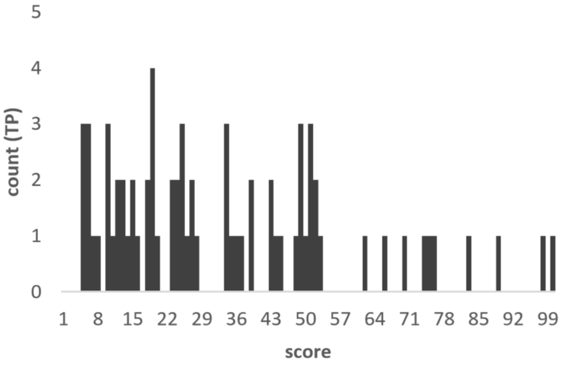

Figure 1. Distribution of the similarity score in TP for *O*-set (DOC, PDF, HTML).

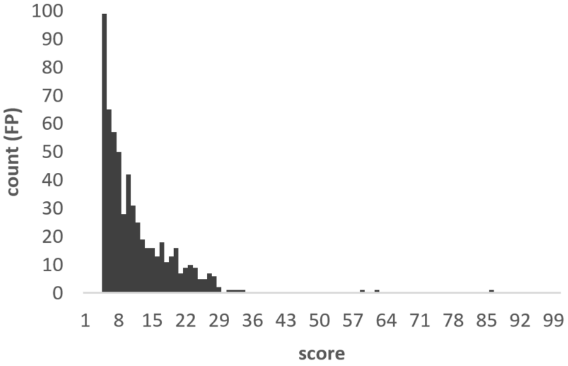

Figure 2. Distribution of the similarity score in FP for *O*-set (DOC, PDF, HTML).

While most of the scores for the false positives are rather low (almost all a 25 or less), the true positives scores are spread over the complete width. Thus, introducing a threshold (e.g., $t \geq 25$) is only partially effective – on the one hand it will eliminate the false positives and on the other it will also increase the false negatives.

Fig. 3 shows the distribution of the true positives for the *P*-set (since there are no FP, no figure is required). Compared to the TP from the *O*-set, the majority of matches have rather high scores (40 and higher). This is due to the comparison algorithm from sdhash which (roughly speaking) correlates the amount of overlap with the total input size.

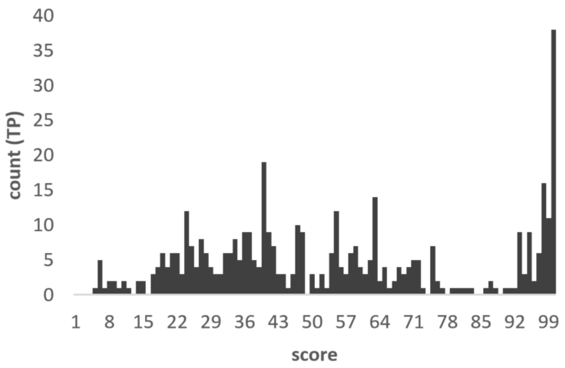

Figure 3. Distribution of the similarity score in TP for P-set (DOC, PDF, HTML).

4.2 Assessment for e-mail (content)

The first paragraph will detail our test-set while the second paragraph will outline our results.

Test-set description. Our second set consists of EML files from the Enron set[11] introduced by Klimt and Yang (2004). This set has a total of 680,579 real world e-mails from 150 users from the same company[12].

Since a lot of the evaluation is done manually, we first reduced the set to 17,831 e-mails from five different users. This subset contains e-mails from all directories such as incoming, outgoing, deleted, etc. Next, we randomly selected 30 e-mails that were compared against this subset which results in $534,930 = 17,831 \times 30$ comparisons.

Pre-processing. Our pre-processing simply separates the headers and the content (e-mail body). Note, `sdhash` requires a minimum input size of 512 bytes[13]. If the pre-processing of the EML file caused a smaller output, we randomly selected a different EML file.

Classification of the results. In contrast to the our first set which was manually created, the challenging part here is to differentiate between TP, FP, FN and TN (the set is not labeled). Therefore, the categorization process is based on two steps. First, we adopt the idea from Bjelland et al. (2014) and second, do a manual comparison. More precisely, we did the following:

Positives: If a match fell into the categories 'reply' and 'different header' (compare last paragraph Sec. 2.3), we rate this as a true positive. This decision was made based on the subjects of the e-mails – if the subject of two e-mails coincide, we rated this as a TP. We are aware that in some scenarios this might be incorrect (e.g., "RE: monthly report") but we believe this is the minority of emails.

If matches were based on 'similar conversation' (i.e., header information only caused the match[14]), we rated this as a false positive. For all remaining ones we manually analyzed the match and added it to the corresponding category: TP or FP.

Negatives: To verify that we do not miss matches (false negatives), we first analyzed the corpus with `ssdeep` to see if this algorithm identifies any additional matches. In a second step, we performed a keyword search throughout the e-mails. That is, we selected up to 5 keywords from each of our 30 e-mails subset and searched for them in the 17,831 e-mails. If one keyword was found, both e-mails were compared intensely (manually) to see if it is a false negative.

[11] https://www.cs.cmu.edu/~./enron (last accessed July 25th, 2016).

[12] To the best of our knowledge, this is the only freely available e-mail dataset and thus we could not mix the e-mails with e-mails from a different company.

[13] http://roussev.net/sdhash/tutorial/03-quick.html (last accessed July 25th, 2016).

[14] Of course, this is only possible for the O-set.

Test results. A summary of the results is given in Table 3. As can be seen, there is a significant difference for the detection rates which is due to the false positives.

Table 3. Results of `sdhash` with and without pre-processing based on the 30 e-mails.

	*O*riginal set	*P*re-processed set
TP	193	198
TN	534,425	534,702
FP	277	1
FN	35	31
Precision	0.41064	0.99498
Recall	0.84649	0.86463
Accuracy	0.99942	0.99994
Fscore	0.55301	0.92523

For the *O*-set, `sdhash` identifies a total of $(193 + 277 =)$ 470 matches while for the *P*-set there are only 199. While in both cases the outcome for TP, TN and FN is almost identical, there is a major difference for the false positives, which causes dropped precision rate. The minor difference in the accuracy rate originates from the large number of true negatives in both scenarios. Thus, in this case the F-score is a more significant measure of accuracy.

Fig. 4 and Fig. 5 shows the distribution of the similarity score for the *O*-set for the true positives and false positives, respectively.

Similar to our first test, the true positives are spread over the complete width while the false positives bunch up in the lower half (most match scores are ≤ 40). Again, it is not possible to identify an appropriate threshold to separate true and false positives because this will cause additional work for an investigator in a real world scenario.

Although the true positives for the *P*-set range from 5 to 100 (see Fig. 6), the majority of matches obtain high scores. On the other hand, the similarity score for the false positives was exactly 5.

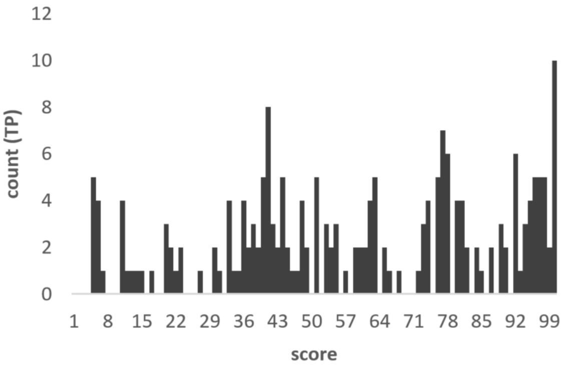

Figure 4. Distribution of the similarity score in TP for *O*-set (EML content).

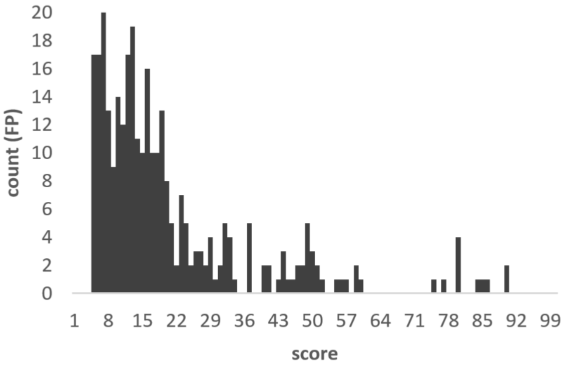

Figure 5. Distribution of the similarity score in FP for *O*-set (EML content).

4.3 Assessment for e-mail (metadata)

In this test scenario, we focus on the false positives from the previous section. As discussed by Bjelland et al. (2014), there are scenarios where headers share relevant information: “similar conversation” (i.e., caused by the ‘from’, ‘to’ and ‘cc’ fields in the header). Therefore, we changed our preprocessing so that now only header information is considered.

Test results. Running our test on the subset showed that many e-mail headers only

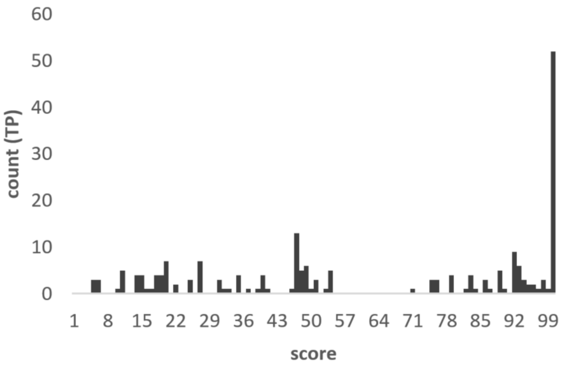

Figure 6. Distribution of the similarity score in TP for P-set (EML content).

Table 4. Results of sdhash Information of 10 input e-mails headers where the header size is over 512 bytes.

Input #	Header size (bytes)	to	cc
1	2,845	27	0
2	1,612	13	0
3	1,150	17	0
4	1,532	8	3
5	7,486	64	1
6	5,170	57	0
7	2,388	28	0
8	1,815	30	0
9	2,231	21	0
10	3,795	32	2

had a short subject and a single recipient address. Thus, these headers did not fulfill the 512 byte requirement from sdhash.

Additional tests. We conducted additional test to verify our hypothesis of matches are caused by similar conversation. We randomly selected 10 e-mails from our e-mail subset where the header size was at least 512 bytes. Table 4 provides a brief summary of the selected e-mails where the 'to' and 'cc' column represent the number of addresses in 'to' and 'cc' header fields, respectively.

Next, sdhash was utilized to compare these 10 e-mails against the subset. The results are summarized by Table 5. Column 1 shows the e-mail identifier followed by the sdhash similarity score. Column 3 and 4 show the overlap of e-mail addresses in these fields. More specifically, let $H1_x$ denote the set of all e-mail addresses in e-mail header 1 and let $H2_x$ denote the set of all e-mail addresses in header 2 where $x \in \{to, cc\}$. Then, 'common to' is $|(H1_{to} \cap H2_{to})|$ and 'common cc' is $|(H1_{cc} \cap H2_{cc})|$. Accordingly, common rate (%) = $|H1 \cap H2| / |H1| \times 100$. The 'identical result count' is the amount of identical matches (columns on the left are identical).

For instance, let us focus on the input #2 block. Row 1 indicates that there was a header with 13 'to' e-mail address matches and none in the cc field. This corresponds a common rate of 100% (according to Table 4 e-mail 2 contains 13 addresses). sdhash returned a similarity score of 80. As indicated by the last column, there was only 1 match. Note, for input #2 row 5 there were 5 matches where sdhash output a similarity score of 6 but 'common to' was 0. The total amount of matches for a given input can be calculated by adding up the last row, e.g., for input #1 sdhash returned 15 matches.

The results show that it is possible to run algorithms on metadata, however there are several false positives where the actual overlap of e-mail addresses is 0 but sdhash outputs a match. Recall, this is not necessarily a false positive from the algorithm itself but from an investigation perspective – only irrelevant header data overlap. On the other hand, analyzing metadata can provide useful information, e.g., see inputs 4, 7 and 9, where it clearly identifies e-mails that were sent to the same user group.

5. CONCLUSION

While current bytewise approximate matching algorithms work precisely, they may be too precise for real-world scenarios. We therefore tested a pre-processing of inputs which can be seen as a step towards syntactic approximate matching. Our experiments demonstrated that simple preprocessing steps can significantly impact the quality of the results. This will be extremely helpful for formats that represent similar information but utilize a different char-set or encoding. On the other hand, as indicated by the metadata test, it also can be useful to compare metadata only (e.g., to identify e-mails that were sent to the same group of people).

A drawback of this procedure is the file type dependency. While the traditional algorithms work independent of the type, syntactic approaches require an awareness of the file type. Furthermore, it is an additional step that will slow down the overall runtime efficiency.

Despite this drawback, we suggest considering the internal structure for common file types and therefore go towards syntactic approximate matching. One possibility would be a two-step procedure: (1) known types are pre-processed while (2) unknown are processed by bytewise approximate matching.

ACKNOWLEDGMENTS

We would like to thank Vikram S. Harichandran for the support when finalizing this article. This research was supported by the Public Welfare and Safety Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2012M3A2A1051116).

REFERENCES

Baier, H., & Breitinger, F. (2011, May). Security Aspects of Piecewise Hashing in Computer Forensics. *IT Security Incident Management & IT Forensics (IMF)*, 21–36. doi: 10.1109/IMF.2011.16

Bjelland, P. C., Franke, K., & Årnes, A. (2014, May). Practical use of approximate hash based matching in digital investigations. *Digital Investigation*, *11*, 18–26. Retrieved from http://dx.doi.org/10.1016/j.diin.2014.03.003 doi: 10.1016/j.diin.2014.03.003

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

Borenstein, N. S., & Freed, N. (1993, September). *Mime (multipurpose internet mail extensions) — part one: Mechanisms for specifying and describing the format of internet message bodies* (Tech. Rep.). Internet RFC 1521.

Breitinger, F., & Baier, H. (2013). Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In M. Rogers & K. Seigfried-Spellar (Eds.), *Digital forensics and cyber crime* (Vol. 114, pp. 167–182). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-39891-9_11 doi: 10.1007/978-3-642-39891-9_11

Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., & White, D. (2014, May). *Approximate matching: Definition and terminology* (Special Publication 800-168). National Institute of Standards and Technologies. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-168

Breitinger, F., Stivaktakis, G., & Roussev, V. (2014, June). Evaluating detection error trade-offs for bytewise approximate matching algorithms. *Digital Investigation*, *11*(2), 81–89. Retrieved from http://dx.doi.org/10.1016/j.diin.2014.05.002 doi: 10.1016/j.diin.2014.05.002

Farrell, P., Garfinkel, S. L., & White, D. (2008). Practical applications of bloom filters to the nist rds and hard drive triage. In *Computer security applications conference, 2008. acsac 2008. annual* (pp. 13–22).

Garfinkel, S. L., & McCarrin, M. (2015). Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb. *Digital Investigation*, *14*, S95–S105.

Klimt, B., & Yang, Y. (2004). The enron corpus: A new dataset for email classification research. In *Machine learning: Ecml 2004* (pp. 217–226). Springer.

Kornblum, J. (2006, September). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3*, 91–97. Retrieved from http://dx.doi.org/10.1016/j.diin.2006.06.015 doi: 10.1016/j.diin.2006.06.015

Resnick, P. (2001). *RFC 2822: Internet message format* (Tech. Rep.). IETF. Retrieved from http://www.rfc-archive.org/getrfc.php?rfc=2822

Roussev, V. (2010). Data fingerprinting with similarity digests. In K.-P. Chow & S. Shenoi (Eds.), *Advances in digital forensics vi* (Vol. 337, pp. 207–226). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-15506-2_15 doi: 10.1007/978-3-642-15506-2_15

Table 5. Matched e-mails list with 10 input e-mails and the number of addresses that both matched and input e-mail have in common.

#	matched e-mail header					#	matched e-mail header				
	similarity score	common to	common cc	common rate (%)	ident. result count		similarity score	common to	common cc	common rate (%)	ident. result count
1	8	0	-	0	4	6	33	22	-	38.60	1
	7	0	-	0	3		31	22	-	38.60	1
	5	1	-	3.70	2		24	0	-	0	1
	5	0	-	0	8		20	0	-	0	1
2	80	13	-	100	1		15	0	-	0	4
	77	13	-	100	1		14	0	-	0	2
	26	12	-	92.31	1		12	0	-	0	1
	10	0	-	0	1		10	0	-	0	4
	6	0	-	0	5		6	0	-	0	3
	5	0	-	0	1		5	0	-	0	3
	4	0	-	0	1	7	92	28	-	100	1
3	93	17	-	100	1		88	28	-	100	4
	86	17	-	100	1		87	28	-	100	1
	17	17	-	100	2		85	28	-	100	1
	15	1	-	5.88	1		84	28	-	100	12
	15	0	-	0	2		73	28	-	100	1
	10	0	-	0	2		68	28	-	100	1
	10	1	-	5.88	3	8	95	30	-	100	1
	9	0	-	0	1		72	30	-	100	2
	5	0	-	0	6		8	8	-	26.67	1
	4	0	-	0	7		6	6	-	20	1
4	93	8	3	100	1		5	5	-	16.67	1
	61	7	3	90.91	2	9	79	21	-	100	1
	42	7	3	90.91	1		78	21	-	100	1
	17	7	3	90.91	1		76	21	-	100	2
	10	8	3	100	1		67	21	-	100	1
	7	7	3	90.91	1		64	21	-	100	2
5	97	64	1	100	1		62	21	-	100	1
	10	0	0	0	1		9	13	-	61.90	1
	8	1	0	1.54	1	10	66	32	1	97.06	1
	7	1	0	1.54	1		31	23	0	67.65	1
	6	0	0	0	1		25	20	1	61.76	1
	5	1	0	1.54	1		21	20	1	61.76	1
	4	0	0	0	1		19	20	1	61.76	1
							5	22	0	64.71	1