Fuzzy Hashing for Digital Forensic Investigators

Dustin Hurlbut - AccessData

January 9, 2009

Abstract

Fuzzy hashing allows the discovery of potentially incriminating documents that may not be located using traditional hashing methods. The use of the fuzzy hash is much like the fuzzy logic search; it is looking for documents that are similar but not exactly the same, called homologous files. Homologous files have identical strings of binary data; however they are not exact duplicates. An example would be two identical word processor documents, with a new paragraph added in the middle of one. To locate homologous files, they must be hashed traditionally in segments to identify the strings of identical data.

Introduction

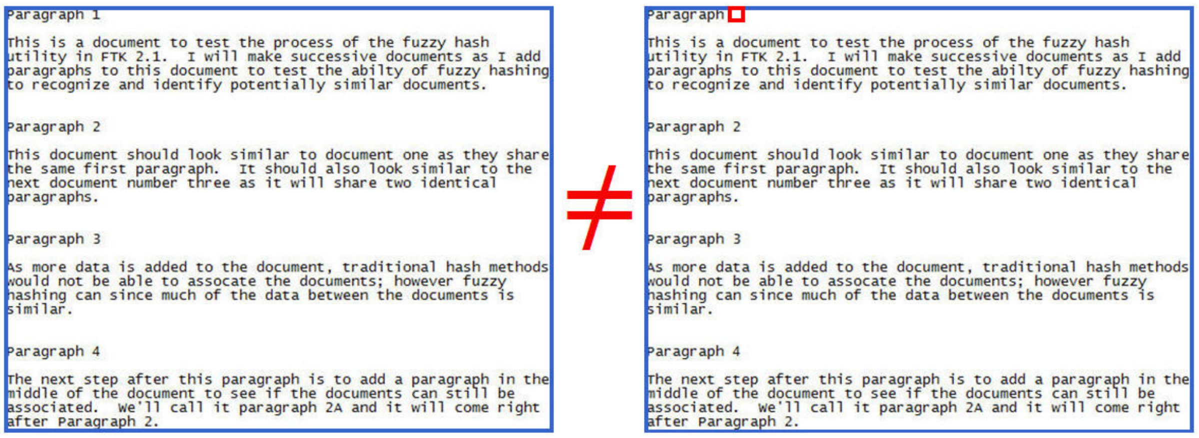
This document will discuss in lay terms what a fuzzy hash is and the theory behind it. FTK contains a fuzzy hash utility that will enable investigators to locate evidence that otherwise could take considerable time to track down. This would include trying to compare live documents active in the file system to partial files recovered from unallocated or slack space. This applies to the law enforcement analyst who is trying to determine if files were ever on a suspect's system, or the corporate investigator who is dealing with an employee who is removing proprietary documents and then making minor alterations to them to avoid detection through conventional hashing techniques.

Fuzzy hashing is not a new concept. In 1999, Dr. Andrew Tridgell wrote the rsync checksum which he later incorporated into his program called spamsum in 2002[1]. Spamsum using rsync is a fuzzy hash utility that attempts to identify e-mail that is similar to known spam[2]. In 2003, Stephen Payne added a utility called C-Hash to his forensic tool DataLifter.Net Bonus Tools that allowed the investigator to combine keywords and a hash of each sector of a file which could then be compared to other sectors both in allocated and unallocated space[3]. This was an effective tool to locate full and partial files from the media. Jessie Kornblum identified the potential presented by spamsum to be used forensically in 2006 in his paper "Identifying almost identical files using context triggered piecewise hashing"[1].

Background

There are three types of hashes the investigator must be familiar with to understand the fuzzy hash utility; Cryptographic (traditional), Rolling, and Context hashing.

Cryptographic Hashes: **Traditional hashing is used to validate media, to locate exact duplicate files, alert us to known files of interest, or let us skip files that are known to not contain evidence. Altering a single bit of data will radically alter the hash. There is no way a cryptographic hash by itself can show us associations with files that may have had even small alterations of their data.**



```
Paragraph 1

This is a document to test the process of the fuzzy hash
utility in FTK 2.1.  I will make successive documents as I add
paragraphs to this document to test the abilty of fuzzy hashing
to recognize and identify potentially similar documents.

Paragraph 2

This document should look similar to document one as they share
the same first paragraph.  It should also look similar to the
next document number three as it will share two identical
paragraphs.

Paragraph 3

As more data is added to the document, traditional hash methods
would not be able to assocate the documents; however fuzzy
hashing can since much of the data between the documents is
similar.

Paragraph 4

The next step after this paragraph is to add a paragraph in the
middle of the document to see if the documents can still be
associated.  We'll call it paragraph 2A and it will come right
after Paragraph 2.
```

$\neq$

```
Paragraph

This is a document to test the process of the fuzzy hash
utility in FTK 2.1.  I will make successive documents as I add
paragraphs to this document to test the abilty of fuzzy hashing
to recognize and identify potentially similar documents.

Paragraph 2

This document should look similar to document one as they share
the same first paragraph.  It should also look similar to the
next document number three as it will share two identical
paragraphs.

Paragraph 3

As more data is added to the document, traditional hash methods
would not be able to assocate the documents; however fuzzy
hashing can since much of the data between the documents is
similar.

Paragraph 4

The next step after this paragraph is to add a paragraph in the
middle of the document to see if the documents can still be
associated.  We'll call it paragraph 2A and it will come right
after Paragraph 2.
```

File 1 File 2

File 1 MD5 Hash = aa060a95f2732612f80dd80e5f33b0f6
File 2 MD5 Hash = 290423c55ee6f9f48c45e6e02d126420

Figure 1 – Traditional Cryptographic Hash Results with a Minor Data Change

In Figure 1, a traditional hash is used to compare two files. In the second file, the "1" in the top title was removed and the two files were hashed using the MD5 cryptographic hash. The hashes are radically dissimilar, and there is no way to ascertain that the documents are virtually identical through this type of hash analysis.

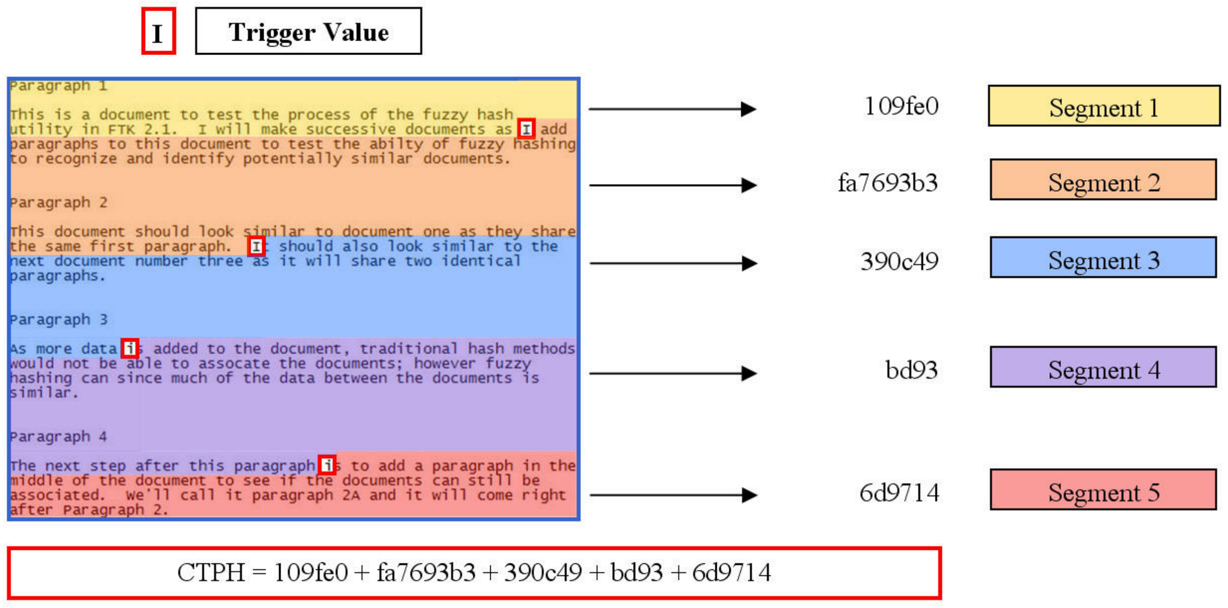
The fuzzy hash utility makes use of traditional hashing, but in segments. The document is hashed in sections defined in part by the size of the document. These hash segments contain fragments of traditional hashes that are joined together for comparative purposes. Before this segmented hash can be done, a rolling hash is used to begin the process.

Rolling Hashes: **A rolling hash is used to identify the segment boundaries. It uses a trigger value, or reset point, generated by the rolling hash engine to determine where these segments will be created. Once located, the data before and after each of the reset points will be processed to create sequences of traditional hash strings.**

In FTK, the trigger value is first based on the file size and on the number of trigger values in the document. Once each trigger value is determined, the traditional hash for that segment is generated and archived. Final hashes are computed based on dividing the file into two blocks and parsing between the trigger values[4] in each block.

The final hash is displayed using a colon to separate the hashes derived from the two blocks of data. This hash result is called a Context Triggered Piecewise Hash (CTPH) as postulated by Jessie Kornblum[1]. The CTPH makes use of the traditional and rolling hashes to create a segmented hash to evaluate documents for potential matching binary strings.

CTP Hashes: **The Context Triggered Piecewise Hash is based upon segments of traditional hashes. Comparisons can be drawn between documents with similar strings of data that are not exact duplicates.**



Paragraph 1

This is a document to test the process of the fuzzy hash utility in FTK 2.1. I will make successive documents as I add paragraphs to this document to test the abilty of fuzzy hashing to recognize and identify potentially similar documents.

Paragraph 2

This document should look similar to document one as they share the same first paragraph. It should also look similar to the next document number three as it will share two identical paragraphs.

Paragraph 3

As more data is added to the document, traditional hash methods would not be able to assocate the documents; however fuzzy hashing can since much of the data between the documents is similar.

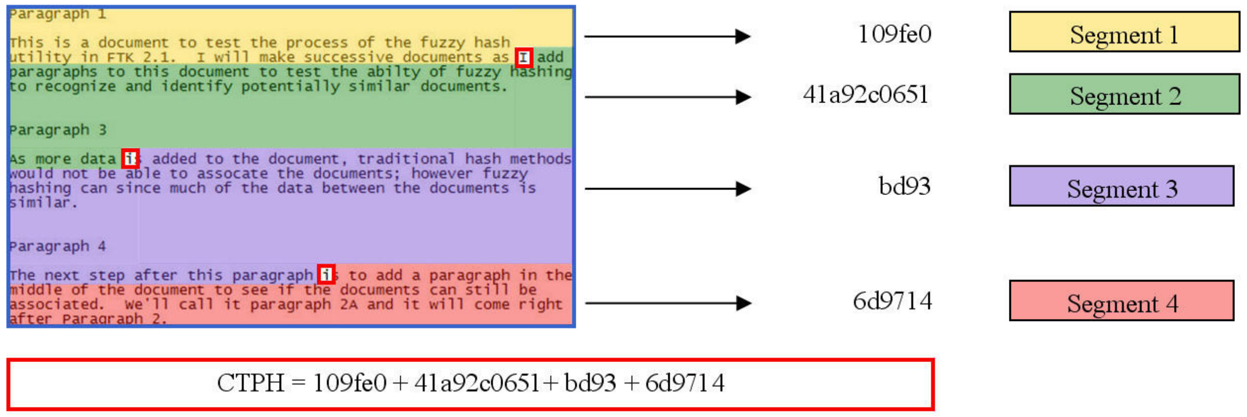
Paragraph 4

The next step after this paragraph is to add a paragraph in the middle of the document to see if the documents can still be associated. We'll call it paragraph 2A and it will come right after Paragraph 2.

CTPH = 109fe0 + fa7693b3 + 390c49 + bd93 + 6d9714

Figure 2 – Context Triggered Piecewise Hash Analogy – Original Document

Figure 2 shows an analogy of a CTPH string. In this example, there are four trigger values. Each trigger value is located by using the rolling hash. The data segments between each trigger value are hashed traditionally and stored.



Paragraph 1

This is a document to test the process of the fuzzy hash utility in FTK 2.1. I will make successive documents as I add paragraphs to this document to test the abilty of fuzzy hashing to recognize and identify potentially similar documents.

Paragraph 3

As more data is added to the document, traditional hash methods would not be able to assocate the documents; however fuzzy hashing can since much of the data between the documents is similar.

Paragraph 4

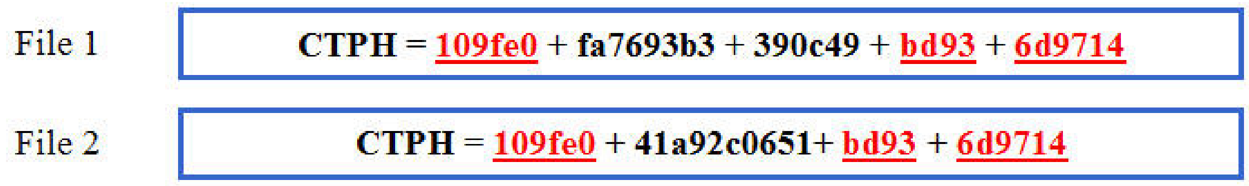
The next step after this paragraph is to add a paragraph in the middle of the document to see if the documents can still be associated. We'll call it paragraph 2A and it will come right after Paragraph 2.

CTPH = 109fe0 + 41a92c0651+ bd93 + 6d9714

Figure 3 – Context Triggered Piecewise Hash Analogy – Altered Document

Figure 3 shows the same document as Figure 2, however in this example; the second paragraph has been removed. The documents are not exact; however they are similar and should obtain a probability match when using the fuzzy hash utility.

Traditional hashing alone will not make any conclusion between the two documents as to similarity; however the CTPH will use the sequences to display potential homologous documents. In the example in Figures 2 and 3, the segmented traditional hashes can be compared for potential associations. Figure 4 shows an analogy of the two hashes from Figures 2 and 3 for comparative purposes. Evaluating these hashes shows a similarity of some identical binary streams between the two documents.

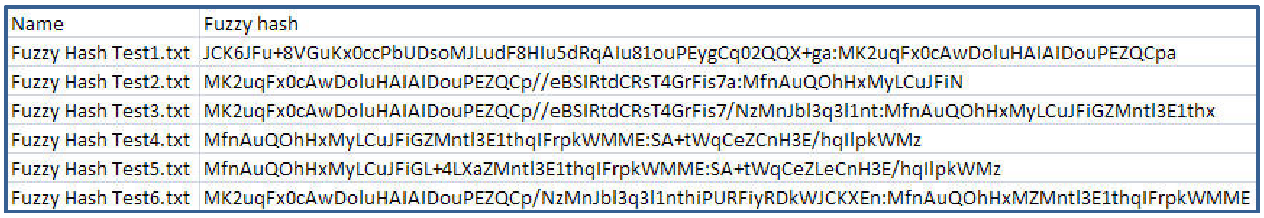


File 1 CTPH = 109fe0 + fa7693b3 + 390c49 + bd93 + 6d9714

File 2 CTPH = 109fe0 + 41a92c0651+ bd93 + 6d9714

Figure 4 – CTPH Comparisons from Figures 2 and 3

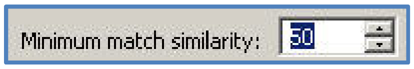
The comparison hashes are created by dividing the file into two blocks; BS1 and 2xBS1 (Where BS stands for Block Size)[4]. Each block will have a hash that is a compilation of the traditional hashes that were generated for each data stream between the trigger values. The two blocks will have a colon as a separator in the hash sequence. In Figure 5, note the similarities between the hash sequences. In the figure examples, a comparison was made of six text files. Each numbered file; 1, 2, 3, 4, and 5 were updated by adding a new paragraph. Number 6 had Paragraph 2 removed from the number 5 version. The second number four document (seen in Figure 6); “Trad Test Change the 1” had only a single byte of data changed between it and document number 4. This was the document shown in Figure 1 where traditional hashing did not show a match.



Name	Fuzzy hash
Fuzzy Hash Test1.txt	JCK6JFu+8VGuKx0ccPbUDsoMJLudF8HIu5dRqAIu81ouPEygCq02QQX+ga:MK2uqFx0cAwDoluHAIAIDouPEZQCpa
Fuzzy Hash Test2.txt	MK2uqFx0cAwDoluHAIAIDouPEZQCp//eBSIRtdCRsT4GrFis7a:MfnAuQOhHxMyLCuJFiN
Fuzzy Hash Test3.txt	MK2uqFx0cAwDoluHAIAIDouPEZQCp//eBSIRtdCRsT4GrFis7/NzMnJbl3q3l1nt:MfnAuQOhHxMyLCuJFiGZMntl3E1thx
Fuzzy Hash Test4.txt	MfnAuQOhHxMyLCuJFiGZMntl3E1thqIFrpkWMME:SA+tWqCeZCnH3E/hqIlpkWMz
Fuzzy Hash Test5.txt	MfnAuQOhHxMyLCuJFiGL+4LXaZMntl3E1thqIFrpkWMME:SA+tWqCeZLeCnH3E/hqIlpkWMz
Fuzzy Hash Test6.txt	MK2uqFx0cAwDoluHAIAIDouPEZQCp/NzMnJbl3q3l1nthiPURFiyRDkWJCKXEn:MfnAuQOhHxMZMntl3E1thqIFrpkWMME

Figure 5 – CTPH Examples from FTK

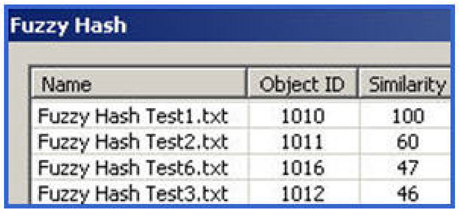
Associations are made by scoring from 1 to 100 with 1 being the lowest and 100 being the highest probability for similarity[5]. A “0” means that no meaningful sequence of data exists between the files. A comparative threshold can be set to further limit the comparative hits in FTK prior to conducting the search, such as setting the comparative value of “50” rather than the default “1”.



Minimum match similarity: 50

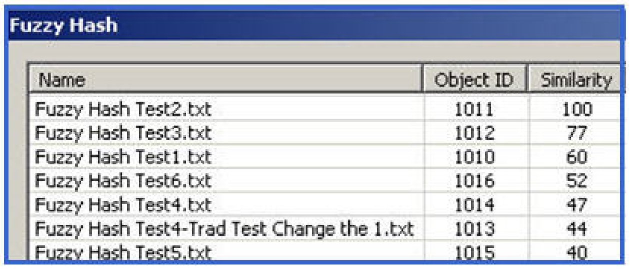
If files of interest are located, they will be displayed in descending order of probability. It is important to note that this process is not perfect. Documents may or may not show a probability match depending on a number of factors, including the type of document and the type of formatting it uses. Text documents have no inserted formatting codes so are more reliable in comparisons with fuzzy hash techniques than are documents created by a word processing program which have considerably more data than just the text that is entered. Differences between formats of graphics files can also alter the effectiveness of a fuzzy hash such as the difference between JPEG formatting procedures and Bitmaps.

Example A — Single paragraph text file


Fuzzy Hash

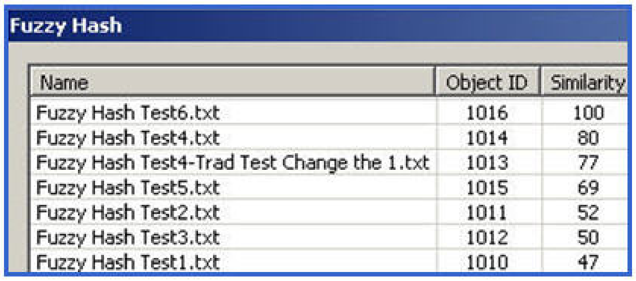
Name	Object ID	Similarity
Fuzzy Hash Test1.txt	1010	100
Fuzzy Hash Test2.txt	1011	60
Fuzzy Hash Test6.txt	1016	47
Fuzzy Hash Test3.txt	1012	46

Example B — Two paragraph text file


Fuzzy Hash

Name	Object ID	Similarity
Fuzzy Hash Test2.txt	1011	100
Fuzzy Hash Test3.txt	1012	77
Fuzzy Hash Test1.txt	1010	60
Fuzzy Hash Test6.txt	1016	52
Fuzzy Hash Test4.txt	1014	47
Fuzzy Hash Test4-Trad Test Change the 1.txt	1013	44
Fuzzy Hash Test5.txt	1015	40

Example C — Deleted paragraph


Fuzzy Hash

Name	Object ID	Similarity
Fuzzy Hash Test6.txt	1016	100
Fuzzy Hash Test4.txt	1014	80
Fuzzy Hash Test4-Trad Test Change the 1.txt	1013	77
Fuzzy Hash Test5.txt	1015	69
Fuzzy Hash Test2.txt	1011	52
Fuzzy Hash Test3.txt	1012	50
Fuzzy Hash Test1.txt	1010	47

Figure 6 – Fuzzy Hash Results of Modified Text Files

The results of FTK fuzzy hashing on these documents are shown in Figure 6. Each numbered file; 1, 2, 3, 4, and 5 were updated by adding a new paragraph. Number 6 had Paragraph 2 removed. The second number four document; "Trad Test Change the 1" had only a single byte of data changed between it and document number 4. This was the document shown in Figure 1 where traditional hashing did not show a match.

In Example A, the first document with a single paragraph was checked for possible similar files. It did not compare to all the related files as being homologous and is missing documents 4 and 5. The second and sixth document values shown in Examples B and C saw all the files as being potentially related.

When the testing is conducted on more complicated files, such as word documents or spreadsheets that have embedded formatting, the results are not always as expected. Tests that discover matches should be run on other potentially matching files to ensure all probable homologous documents are located.

This type of action can also be conducted on other types of files such as graphics. Viewing a set of twelve bitmap graphics that were analyzed as being related, did have a functional similarity; they all had the same border color around the graphic. See Figure 7.

Because the files are tested and a similarity is indicated won't always mean they are similar or related. The hash is not as distinctive as an MD5 or SHA hash. The more segments in a document, particularly one with many minor changes, can show the document as not being related even when it may be.

Forensic Issues

Creating the fuzzy hash sets takes processing time, more than that needed to create MD5 hashes for example. The use of the fuzzy hash utility may decrease performance by 7 to 10 times the normal time to process a case[6] if fuzzy hashing wasn't selected.

Obtaining a fuzzy hash of a file is limited to the file size. Smaller files under 100 bytes generally won't be able to generate a fuzzy hash, since they don't have sufficient size to find the requisite trigger values.

Identical files with the same MD5 hash will fuzzy hash with the same fuzzy hash string. They will show a 100 for matching when compared to their duplicate counterparts.

The strength of fuzzy hash sets are their ability to locate similar files. They can be used to match altered documents, such as multiple or incremental versions of the same document. Malicious alteration of a document to avoid detection such as stolen proprietary corporate files is difficult to detect but is apparent from fuzzy hash analysis. Fuzzy hashing will point to the documents that should be reviewed.

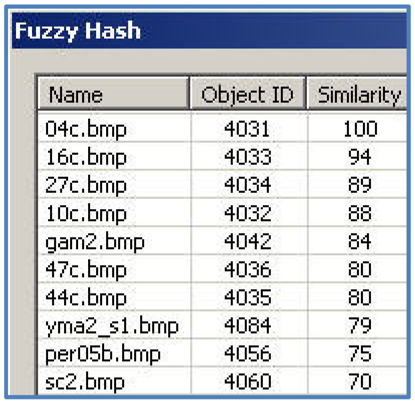
Fuzzy hashing can also be useful in examining partial files such as carved documents to compare to other documents of interest. Carving may retrieve partial documents that can be related to the original. Fuzzy hashes may also relate a document to a suspect when the incriminating document doesn't exist in the active file system. If the investigator has access to the original or suspected document in question, that document can be fuzzy hashed from outside of FTK by pointing to it. It can then be compared with carved items taken from within the image to determine if it was ever in the system at one time.

Partial graphics without headers may also be discovered in this manner. It seems to be common to search a drive for child pornographic photos only to discover no active documents. Other evidence may point to their existence at one time, but the files are not in the active file system. Data carving may reveal graphics, but if their headers aren't carved with them, they can't normally be viewed. Fuzzy hashes may uncover matches to partial graphics that will help determine if such files were ever in the system at one time.

Text documents make excellent candidates for fuzzy hash comparisons. Word processed documents, spreadsheets, and other types of documents that have extraneous formatting applied are not as reliable; however the strings of user inputted data will often be detected in the associations.

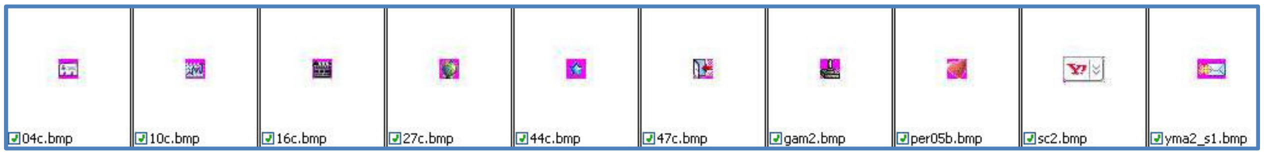
Graphics present different issues than documents that contain textual data. The potential of successful comparisons with graphic files will depend upon the type of graphic format that was used. A comparison was run on a small image (Mantooth32[7]) for matches to a particular Bitmap graphic as shown in Figure 7.

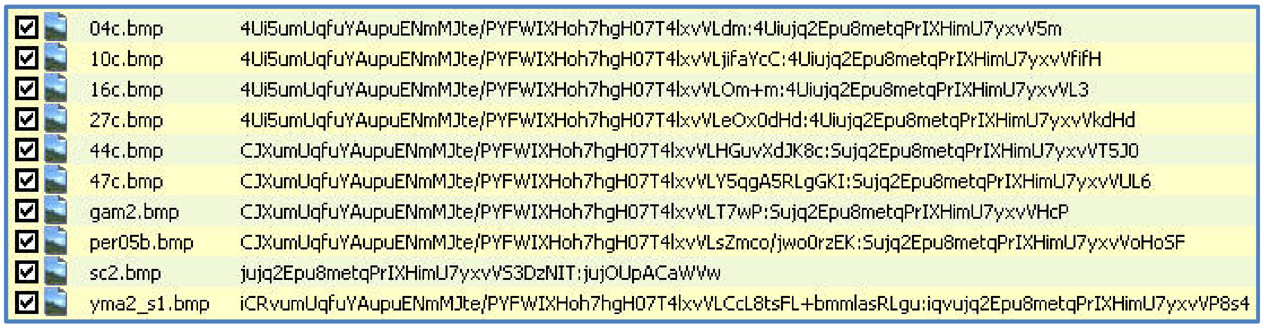
A visual inspection of the bitmaps will show they do have a similarity; their borders are all a similar color. These bitmaps were all icons from what appears to be a single website. Each had a similar size ranging from 1180 to 1464 bytes. It would appear from this example and others that the bitmap format lends itself to a reasonable chance of finding similar files based in this example on similar colors, sizes, and origin.



Fuzzy Hash

Name	Object ID	Similarity
04c.bmp	4031	100
16c.bmp	4033	94
27c.bmp	4034	89
10c.bmp	4032	88
gam2.bmp	4042	84
47c.bmp	4036	80
44c.bmp	4035	80
yma2_s1.bmp	4084	79
per05b.bmp	4056	75
sc2.bmp	4060	70

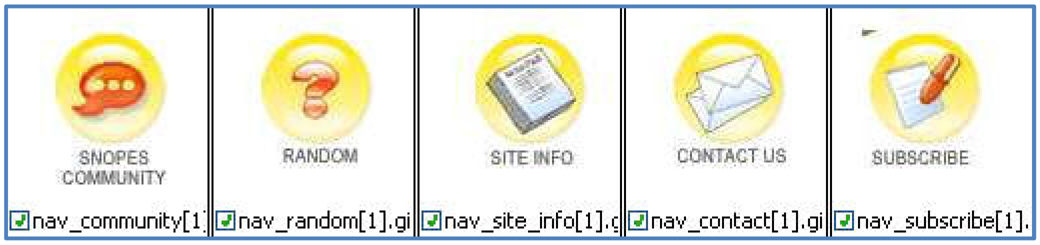


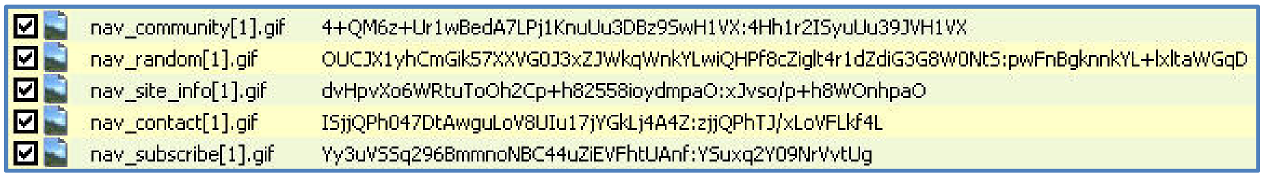


04c.bmp	4Ui5umUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLdm:4Uiujq2Epu8metqPrIXHimU7yxvV5m
10c.bmp	4Ui5umUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLjifaYcC:4Uiujq2Epu8metqPrIXHimU7yxvVfifH
16c.bmp	4Ui5umUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLOm+m:4Uiujq2Epu8metqPrIXHimU7yxvVL3
27c.bmp	4Ui5umUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLeOx0dHd:4Uiujq2Epu8metqPrIXHimU7yxvVkdHd
44c.bmp	CJXumUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLHGuvXdJK8c:Sujq2Epu8metqPrIXHimU7yxvVT5J0
47c.bmp	CJXumUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLY5qgA5RLgGKI:Sujq2Epu8metqPrIXHimU7yxvVUL6
gam2.bmp	CJXumUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLT7wP:Sujq2Epu8metqPrIXHimU7yxvVHcP
per05b.bmp	CJXumUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLsZmco/jwo0rzEK:Sujq2Epu8metqPrIXHimU7yxvVoHoSF
sc2.bmp	jujq2Epu8metqPrIXHimU7yxvVS3DzNIT:jujOUpACaWVw
yma2_s1.bmp	iCRvumUqfuYAupuENmMJte/PYFWIXHoh7hgH07T4lxvVLCcL8tsFL+bmmlasRLgu:iqvujq2Epu8metqPrIXHimU7yxvVP8s4

Figure 7 – Fuzzy Hash Results for .BMP Files

Comparisons of .GIF files that appeared similar did not display any matching characteristics like the bitmap graphics example. Figure 8 shows five .GIF files of similar size and appearance. None of these files exhibits any similar comparative hashes from a fuzzy hash comparison.

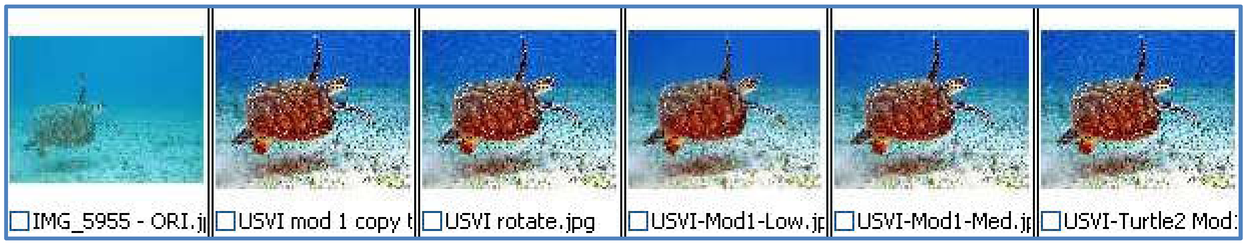


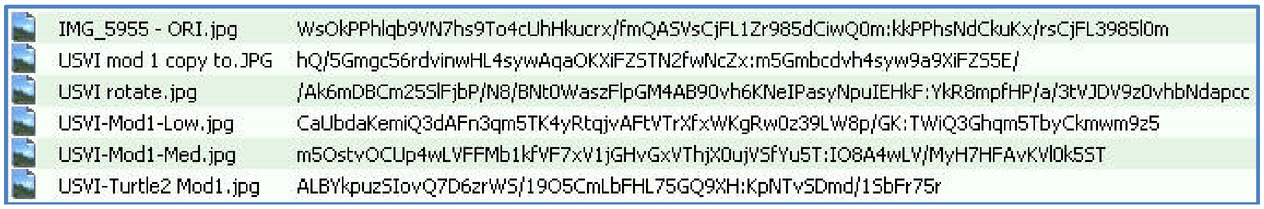


nav_community[1].gif	4+QM6z+Ur1wBedA7LPj1KnuUu3DBz9SwH1VX:4Hh1r2ISyuUu39JVH1VX
nav_random[1].gif	OUCJX1yhCmGik57XXVG0J3xZJWkqWnkYLwiQHPf8cZiglt4r1dZdiG3G8W0NtS:pwFnBgknnkYL+lxltaWGqD
nav_site_info[1].gif	dvHpvXo6WRtuToOh2Cp+h82558ioydmpaO:xJvso/p+h8WOnhpaO
nav_contact[1].gif	ISjjQPh047DtAwguLoV8UIu17jYGkLj4A4Z:zjjQPhTJ/xLoVFLkf4L
nav_subscribe[1].gif	Yy3uVSSq296BmmnoNBC44uZiEVFhtUAnf:YSuxq2Y09NrVvtUg

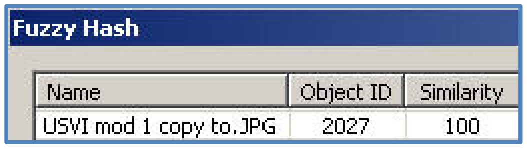
Figure 8 – Fuzzy Hash Results for .GIF Files

The JPEG graphic format also presents unique issues based upon its format design that uses lossy compression. Each time a JPEG is changed, it compresses portions of the data, specifically the least significant bit(s) of pixel data. Saving a .JPEG file using the Save As dialog, Copy To, or any minor changes such as rotating the graphic and rotating it back can change the compression. This will lead to an inability of fuzzy hashing to detect similarities when the human eye can clearly detect what appear to be identical graphics. There are so many minute changes that occur across the document that fuzzy hashing can't derive a similarity between the segments.





IMG_5955 - ORI.jpg	WsOkPPhlqb9VN7hs9To4cUhHkucrx/fmQASVsCjFL1Zr985dCiwQ0m:kkPPhsNdCkuKx/rsCjFL3985l0m
USVI mod 1 copy to.JPG	hQ/5Gmgc56rdvinwHL4sywAqaOKXiFZSTN2fwNcZx:m5Gmbcdvh4syw9a9XiFZS5E/
USVI rotate.jpg	/Ak6mDBCm25SlFjbP/N8/BNt0WaszFlpGM4AB90vh6KNeIPasyNpuIEHkF:YkR8mpfHP/a/3tVJDV9z0vhbNdapcc
USVI-Mod1-Low.jpg	CaUbdaKemiQ3dAFn3qm5TK4yRtqjvAFtVTrXfxWKgRw0z39LW8p/GK:TWiQ3Ghqm5TbyCkmwm9z5
USVI-Mod1-Med.jpg	m5OstvOCUp4wLVFFMb1kfVF7xV1jGHvGxVThjX0ujVSfYu5T:IO8A4wLV/MyH7HFAvKVl0k5ST
USVI-Turtle2 Mod1.jpg	ALBYkpuzSIovQ7D6zrWS/19O5CmLbFHL75GQ9XH:KpNTvSDmd/1SbFr75r



Fuzzy Hash

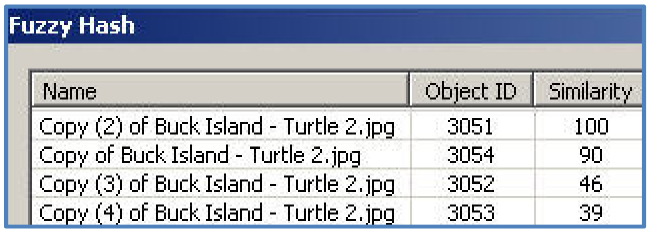
Name	Object ID	Similarity
USVI mod 1 copy to.JPG	2027	100

Figure 9 – Fuzzy Hash Results for .JPG Files

Figure 9 shows the same photograph as it was modified, Copied To, Saved As, rotated and rotated back, and had resolution changes. None of these graphics obtain a match using the fuzzy logic utility. Each graphic was altered and the MD5 hash sets show no matches. Even minor alterations will radically change the format of JPEG graphics; hence, they are not a good candidate for comparative analysis using fuzzy hashing techniques. The only matches obtained were to the document itself.

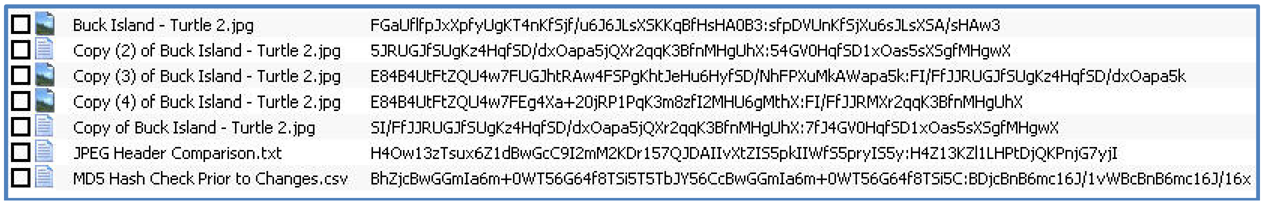
Considering the issues with the JPEG format, the potential still exists to locate matches based on partial files obtained from data carving. The test shown in Figure 10 involved making identical copies of the same graphic (no changes were made through lossy compression). The graphics were in turn modified by having just the header removed, then the front, middle and back one third of the document removed. Clicking on the original unmodified graphic didn't match to the others. However, the modified documents did match to the other files.

Libraries of known fuzzy hashed graphics could be used against a suspect's machine to locate partial files carved from unallocated space that are no longer active in the file system. This assuming no changes from the lossy compression algorithm which would not normally occur from a download or simple copy.



Fuzzy Hash

Name	Object ID	Similarity
Copy (2) of Buck Island - Turtle 2.jpg	3051	100
Copy of Buck Island - Turtle 2.jpg	3054	90
Copy (3) of Buck Island - Turtle 2.jpg	3052	46
Copy (4) of Buck Island - Turtle 2.jpg	3053	39



Buck Island - Turtle 2.jpg	FGaUflfpJxXpfyUgKT4nKfSjf/u6J6JLsXSKKqBfHsHA0B3:sfpDVUnKfSjXu6sJLsXSA/sHAw3
Copy (2) of Buck Island - Turtle 2.jpg	5JRUGJfSUgKz4HqfSD/dxOapa5jQXr2qqK3BfnMHgUhX:54GV0HqfSD1xOas5sXSgfMHgwX
Copy (3) of Buck Island - Turtle 2.jpg	E84B4UtFtZQU4w7FUGJhtRAw4FSPgKhtJeHu6HyfSD/NhFPXuMkAWapa5k:FI/FfJJRUGJfSUgKz4HqfSD/dxOapa5k
Copy (4) of Buck Island - Turtle 2.jpg	E84B4UtFtZQU4w7FEg4Xa+20jRP1PqK3m8zfI2MHU6gMthX:FI/FfJJRMXr2qqK3BfnMHgUhX
Copy of Buck Island - Turtle 2.jpg	SI/FfJJRUGJfSUgKz4HqfSD/dxOapa5jQXr2qqK3BfnMHgUhX:7fJ4GV0HqfSD1xOas5sXSgfMHgwX
JPEG Header Comparison.txt	H4Ow13zTsux6Z1dBwGcC9I2mM2KDr157QJDAIIvXtZIS5pkIIWfS5pryIS5y:H4Z13KZl1LHPtDjQKPnjG7yjI
MD5 Hash Check Prior to Changes.csv	BhZjcBwGGmIa6m+0WT56G64f8TSi5T5TbJY56CcBwGGmIa6m+0WT56G64f8TSi5C:BDjcBnB6mc16J/1vWBcBnB6mc16J/16x

Figure 10 – Fuzzy Hash Results for Altered .JPEG Files

Tests were also run on .ZIP files (WinZip Version 11.1). Hits were obtained when different Zip files contained combinations of the same archived files. Zip archiving appears to compress in a predictable manner and zipped documents were successfully compared to partially carved zip files for matches.

FTK can create libraries of fuzzy hashes for specific document types[4,5] such as proprietary corporate documents or known child pornographic files. These libraries can be applied to a seized media to see if potential matches exist without a close initial examination of each document. Letting the fuzzy hashing utility make initial comparisons has the potential to save considerable analysis time in locating matching documents.

Use of the fuzzy hash to locate comparative matches can be helpful; however, a complete reliance on them could result in missing similar files in an investigation. Many file types lend themselves to comparative matches such as text files, but some documents, like JPEG graphics do not. Investigators should test the specific file types of interest in a case before relying on a completely automated result returned from a fuzzy hash utility. It would appear the Mark II eyeball of the investigator will still be required to make the final decision on whether certain documents in a case are similar or not.

References

[1] Kornblum Jesse. Identifying almost identical files using context triggered piecewise hashing. August 2006. http://dfrws.org/2006/proceedings/12-Kornblum.pdf. Published by Elsevier Ltd.
[2] Tridgell Andrew. Readme. http://www.samba.org/ftp/unpacked/junkcode/spamsum/README.
[3] DataLifter. http://www.datalifter.com/products.htm.
[4] AccessData. Fuzzy Hashing. 2008.
[5] AccessData. FTK Users Guide. 2008.
[6] Kornblum Jesse. jessekornblum.com/research/fuzzy-hashing-cdfsl-2007.ppt.
[7] AccessData. BootCamp classroom Image Mantooth32.