Fuzzy Hashing for Digital Forensic Investigators

Dustin Hurlbut - AccessData

January 9, 2009

Abstract

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

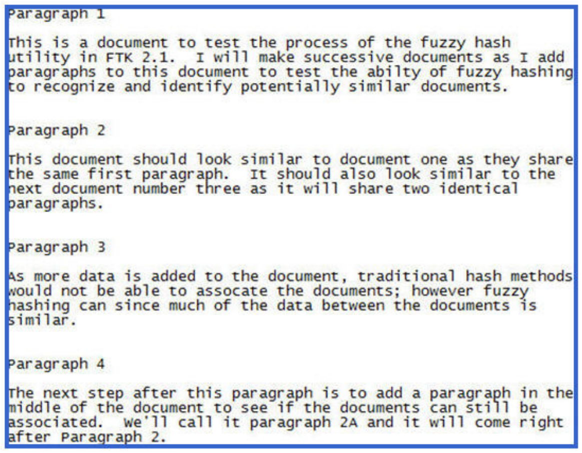
Introduction

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 use this new tool 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 and 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 perhaps 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.

Background

There are three types of hashes the investigator must be familiar with to understand the fuzzy hash utility; Cryptographic, 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 not to 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

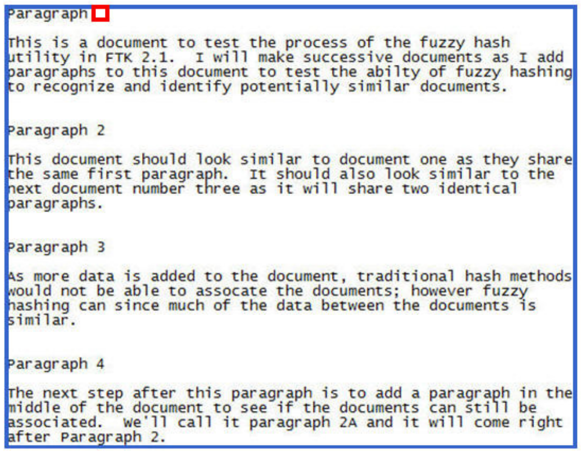
As more data is added to the document, traditional hash methods
would not be able to assocate the documents; however fuzzy
hashing can since much of the data between the documents is
similar.

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.

File 1

$\neq$



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 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" was removed and the two files were hashed using the MD5 cryptographic hash. The hashes are radically dissimilar, and there is no way to know the documents are virtually identical through this type of hash analysis.

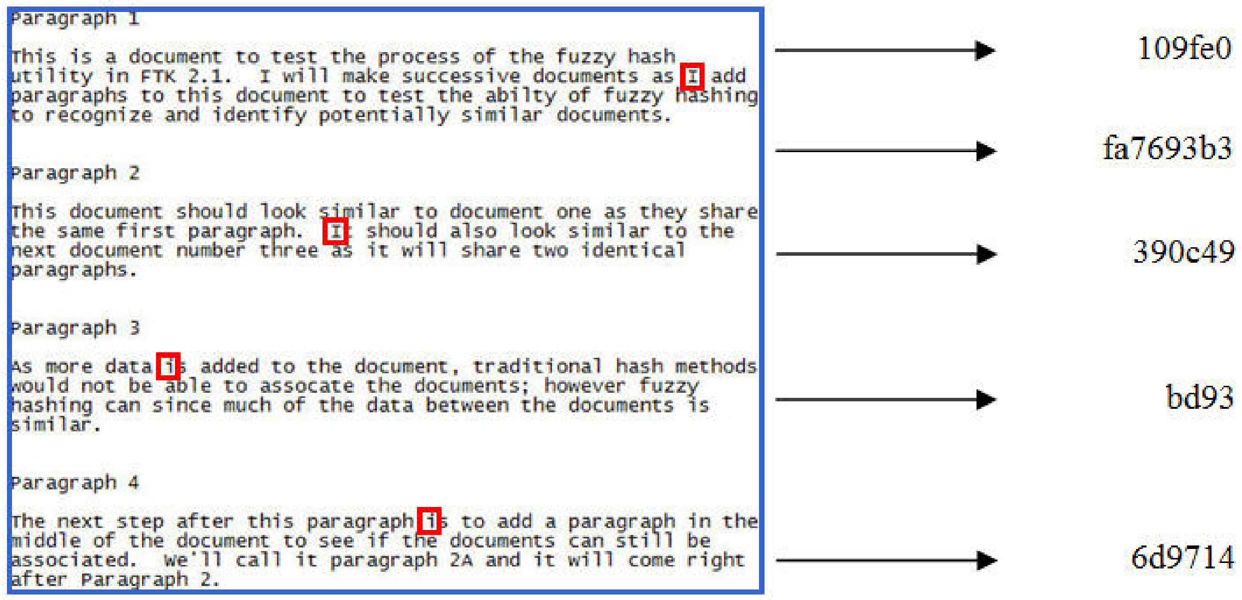
The fuzzy hash utility makes use of traditional hashing, but in pieces. The document is hashed in segments depending on the size of the document. These segments will contain fragments of traditional hashes joined together for comparative purposes. Before this segmented hash can be done, a rolling hash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

Rolling Hashes:	**A rolling hash is used to piece through the document. It uses a "trigger" value generated by the rolling hash engine to determine where these segments will be created. Once located, the segments 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 the trigger value in the rolling hash 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[1].

The final hash is displayed using a colon to separate the two hashes derived from the two blocks of data. This hash result is called a Context Triggered Piecewise Hash (CTPH) as postulated by Jessie Kornblum[2]. The CTPH makes use of the traditional and rolling hashes to create a segmented hash to evaluate documents for 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

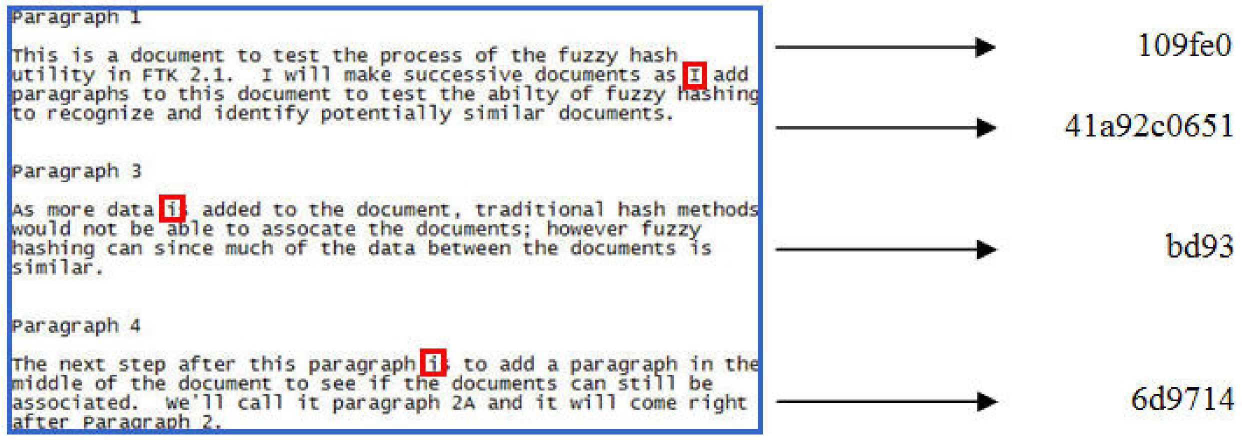
The next step after this paragraph is to add a paragraph in the middle of the document to see if the documents can still be associated. We'll call it paragraph 2A and it will come right after Paragraph 2.

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

Figure 2 – Context Triggered Piecewise Hash Example – Original Document

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

Figure 3 shows the same document as Figure 2, however in this example; the second paragraph has been removed.



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.

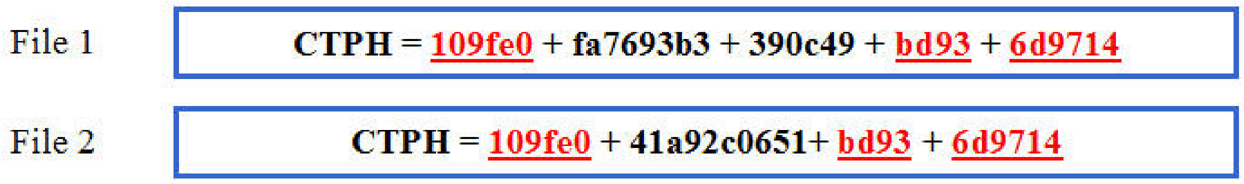
Paragraph 4

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 Example – Altered Document

Traditional hashing 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, a single paragraph has been removed. 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. Comparing these hashes shows a similarity between the two documents.

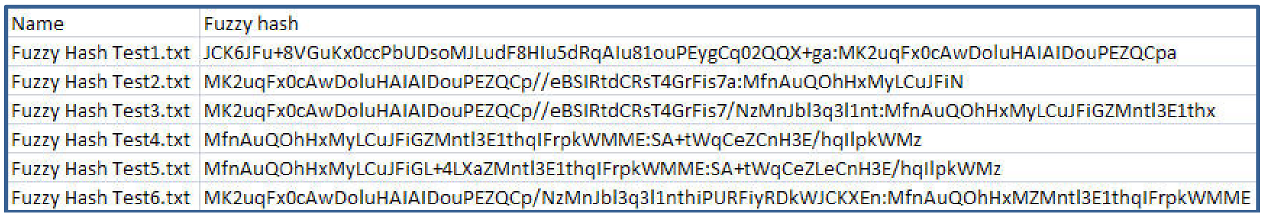


File 1 CTPH = 109fe0 + fa7693b3 + 390c49 + bd93 + 6d9714

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)[1]. 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 4, 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 had a paragraph added. Number 6 had Paragraph 2 removed. The second number four document (seen in Figure 5); "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.



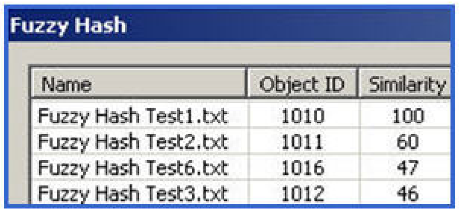
Name	Fuzzy hash
Fuzzy Hash Test1.txt	JCK6JFu+8VGuKx0ccPbUDsoMJLudF8HIu5dRqAIu81ouPEygCq02QQX+ga:MK2uqFx0cAwDoluHAIAIDouPEZQCpa
Fuzzy Hash Test2.txt	MK2uqFx0cAwDoluHAIAIDouPEZQCp//eBSIRtdCRsT4GrFis7a:MfnAuQOhHxMyLCuJFiN
Fuzzy Hash Test3.txt	MK2uqFx0cAwDoluHAIAIDouPEZQCp//eBSIRtdCRsT4GrFis7/NzMnJbl3q3l1nt:MfnAuQOhHxMyLCuJFiGZMntl3E1thx
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 4 – CTPH Examples from FTK

Associations are made by scoring from 0 to 100 with 0 being a low probability and 100 being a high probability for similarity[3]. FTK will make these hash comparisons to establish the likelihood of similarity.

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 match up depending on a number of factors.

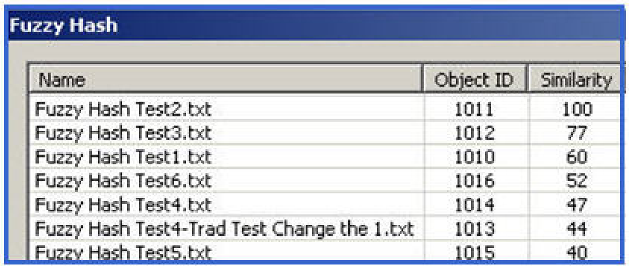
Example A — Single paragraph text file



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

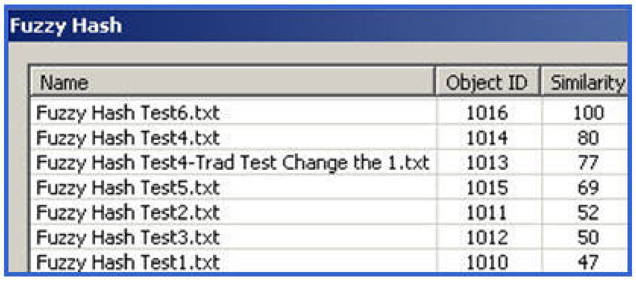
Example B — Two paragraph text file



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 5 – Fuzzy Hash of Modified Text Files

The results of FTK fuzzy hashing on these documents are shown in Figure 5. In Example A, the first document with a single paragraph was checked for possible similar files. It did not compute all the related files as being homologous and is missing both documents 4 and 5. However, 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 the other matched files to ensure all potentially 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 graphics that were analyzed as being similar, did have a functional similarity; they all had the same border color around the graphic.

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 to show relations. The more segments in a document, particularly one with many minor changes, can also show the document as not being related even when it may be.

Forensic Issues

The use of the fuzzy hash utility may decrease performance by 7 to 10 times the normal time to process a case[2, 4]. It would be advisable to use checked files in FTK for comparative purposes or use filters to limit the number of comparisons on large media.

The strength of the fuzzy hashes are their ability to locate similar files. It 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 is difficult to detect as discussed in the introduction. A thorough examination of documents is required. Fuzzy hashing can easily find documents altered in this manner. 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. 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 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, but the files are not in the file system. Data carving may reveal graphics, but if their headers aren't carved with them, they can't normally be viewed. Using fuzzy hashing may uncover partial graphics that will help determine if such graphics were in the system at one time.

FTK can create libraries of fuzzy hashes for specific document types[1,3], such as child pornography graphics or corporate proprietary documents. These libraries can then be applied to a seized media to see if potential matches exist without a close initial examination by the investigator of each document. A comparative threshold can be set to further limit the comparative hits. This utility has the potential to save considerable analysis time in locating matching documents.

References

[1] AccessData. Fuzzy Hashing. 2008.
[2] Kornblum Jesse. Identifying almost identical files using context triggered piecewise hashing. Digital Investigation*,* 3(S):91-97, Proceedings of the Digital Forensic Workshop, August 2006. http://dfrws.org/2006/proceedings/12-Kornblum.pdf.
[3]AccessData. FTK Users Guide. 2008.
[4]Kornblum Jesse. jessekornblum.com/research/**fuzzy**-**hashing**-cdfsl-2007.ppt.