A METHOD FOR PERFORMANCE VERIFICATION OF FREEWARE HASH UTITLIES USING MATLAB

by

SEAN ROBERT JACOBSON

B.S., University of Colorado Denver, 2011

A thesis submitted to the Faculty of the Graduate School of the University of Colorado in partial fulfillment of the requirements for the degree of Master of Science Recording Arts 2013 UMI Number: 1544273

All rights reserved

INFORMATION TO ALL USERS The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 1544273

Published by ProQuest LLC (2013). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC. All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC. 789 East Eisenhower Parkway P.O. Box 1346 Ann Arbor, MI 48106 - 1346 This thesis for the Master of Science degree by

Sean Robert Jacobson

has been approved for the

Recording Arts Program

by

Catalin Grigoras, Chair

Jeff Smith

Leslie M Gaston Bird

Date: 7/11/2013

Jacobson, Sean Robert, M.S., Recording Arts

A Method for Performance Verification of Freeware HASH Utilities Using MATLAB Thesis directed by Associate Professor Catalin Grigoras

ABSTRACT

HASH values are the result of a cryptographic HASH function that can be used for security or file authentication. Due to the rise of file exchange on the internet this value has become even more important when working with digital evidence. In recent years freeware programs have become available online for both commercial and personal use to check for this value. However, there has not yet been a set way to verify these programs and to check that the results they are giving is in fact accurate. This thesis proposes a series of tests that allows for a user or agency to check the program and see if it is indeed accurate.

The form and content of this abstract are approved. I recommend its publication.

Approved: Catalin Grigroas

ACKNOWLEDEMENTS

Firstly I want to say thank you to my mom and dad, Teri and David Jacobson. You put me through school and kept encouraging me. This thesis wouldn't be possible without your support. Secondly to my teachers Jeff Smith and Catalin Grigoras. For giving me great advice on where to find information, putting up with frustrations, and being there when I needed help, thanks. Leslie Gaston Bird a long time teacher for being on my committee and reviewing my multiple drafts. And to my co-workers at the Bookworm, for putting up with me during the final process and my stressful moments. Thanks so much for everything!

TABLE OF CONTENTS

CHA	PTEI	R	
I.		INTRODUCTION	1
		What are HASH Values	1
		Common HASH Algorithms	4
		Problems with HASH Values	6
		NIST and NIJ work with HASH	8
IJ	[.	PROPOSED FRAMEWORK AND EXPERIMENTAL PROCESS	10
Π	Π.	CONCLUSION	25
В	IBLI	OGRAPHY	26
А	PPE	NDIX	27

Table	
1: Excel Document MD5 Results	15
2: Excel Document SHA-1 Results	15
3: JPEG Image File MD5 Results	16
4: JPEG Image File SHA-1 Results	16
5: PDF File MD5 Results	17
6: PDF File SHA-1 Results	17
7: Text File MD5 Results	18
8: Text File SHA-1 Results	18
9: Word Document MD5 Results	19
10: Word Document SHA-1 Results	19
11: WAVE File MD5 Results	20
12: WAVE File SHA-1 Results	20
13: MP3 File MD5 Results	21
14: MP3 File SHA-1 Results	21
15: WMA File MD5 Results	22
16: WMA File SHA-1 Results	22
17: AVI Video File MD5 Results	23
18: AVI Video File SHA-1 Results	23
19: QuickTime Video File MD5 Results	24
20: QuickTime Video File SHA-1 Results	

LIST OF TABLES

LIST OF FIGURES

Figure	
1-1 The Original File and its HASH	2
1-2 The Copy of the Orignal File and its HASH	3
1-3 The Original File and its HASH	3
1-4 The Copied File Altered	4
1-5 WHIRLPOOL Algorithm Diagram	6
1-6 The Letter of Recommendation	7
1-7 The Altered Document	7
2-1 Excel Spreadsheet Example	12
2-2 All HASH Values Match	12
2-3 All HASH Values Match, columns	13
2-4 HASH Value Differences	13
2-5 HASH Value Differences, column	14

CHAPTER I

INTRODUCTION

In the emerging field of media forensics digital files are an important part of evidence analysis. When working with a file such as video or an image file it is important for a forensic examiner to know that the file that they are working with is the file that was originally collected at the crime scene. In order to verify evidence integrity upon seizure and throughout processing, HASH values should be calculated and compared.

In today's society the internet has opened up avenues to free exchange of information. This has also caused some problems with knowing whether or not the file received is the original file or if there have been alterations made to it. The HASH value is a tool used to determine whether or not the file's contents is the same or if it is different. There are many different kinds of programs that allow for a person to check the HASH value, some of them have been authenticated by the forensic community and some are freeware programs. In this paper eight different freeware programs are compared to two forensically sound programs to determine whether or not they function the same.

What are HASH Values?

A HASH value is a product of cryptographic HASH function used to authenticate a digital file [1]. The HASH function will take information from the digital file and run through an algorithm to generate the value. In digital media each digital file should have a unique HASH value. No other digital file should have the same HASH value unless it is a clone. [2]

A HASH value should be unique to the file as it currently is. If a person were to make alterations to the file the value would also be changed. In digital forensics the HASH value has come to be more important with the rise in file exchange sites and download utilities on the internet. Below is an example of this using a program called WinHex to check for the HASH value. In Figure 1-1 and Figure 1-2 an example of this is given. A text file called Test File was created and placed in WinHex to get its HASH value. Figure 1-1 is the original file and Figure 1-2 is its copy.

👯 WinHex - [Test File.txt]	
👯 File Edit Search Navigati	on View Tools Specialist Options Window Help
🗅 🚅 🖶 🎒 😭 🖉 👘	Be Be Be B: # # # \\$ \$ # → +D <= > _2 = ~ @ [] _ [
Fiļe E <u>d</u> it	Test File.bt Test File - Copy.bt
	Offset 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 1
	00000000 54 68 69 73 20 69 73 20 61 20 74 65 73 74 20 6
	00000016 69 6C 65
	MD5 (128 bit)
	for Test File.txt: DB26E313ED4A7CA6904B0E9369E5B957 Qose

Figure 1-1: The Original File and its HASH.

👯 WinHex - [Test File - Copy.txt]	Sean Jacobson Thesis Draft Final Draft.docx - Word	
File Edit Search Navigati	ion View Tools Specialist Options Window Help	
🗅 🖻 🖬 🎒 🕍 👘	≞ ≞ ™ % # # # \\$ % ₩ # → + + + ⇒ _> =	Sam 1
Fi <u>l</u> e E <u>d</u> it	Test File.txt Test File - Copy.txt	
	Offset 0 1 2 3 4 5 6 7 8 9 10 11 1	2 13
		374
	00000016 69 6C 65	
	MD5 (128 bit)	
	MD5 (128 bit)	
	for Test File - Copy.txt:	
	0B26E313ED4A7CA6904B0E9369E5B957	
	Close	

Figure 1-2: The Copy of the Orignal File and its HASH.

In Figure 1-3 and Figure 1-4 one of the file's contents have been changed. This has caused the HASH value of that file to have also changed.

Test File.txt Tes	t File	- Cop	y.bd														
Offset	0	1	2	3	4	5	6	- 7	8	9	10	11	12	13	14	15	
00000000				73	20	69	73	20	61	20	74	65	73	74	20	66	This is a test f
00000016	69	6C	65														ile
		м	05 (1	28 b	it)									х			
															n II.		
							fe	or Tes	st File.t	st:					11		
			DB	26E.3	13EF)447	CA69	904BC)E9369	E5B	957			٦l	11		
							GR 100								11		
															ш		
								Cļo)se								

Figure 1-3: The Original File and its HASH.

Test File.txt Tes	t File - Copy.txt												
Offset	0 1 2	3 4	5	6	7 8	9	10	11	12	13	14	15	
00000000	54 68 69	73 20	69	73	20 61	6E	20	61	6C	74	65	72	This is an alter
00000016	65 64 20	74 65	73	74	20 66	69	6C	65					ed test file
	MD5 (128	bit)							×				
	5393	8623B1EC		DCE2	File - Copy 24DCBE 44 2 (ose		4						

Figure 1-4: The Copied File Altered.

When working with digital information it is important for an examiner to know that the file they are working with is the original. A HASH value, when checked, gives the examiner the assurance that this is indeed a bit-for-bit copy and accurately reflects the original evidence. The tables in Chapter two are the results of testing for errors in multiple HASH checker programs.

Common HASH Algorithms

There are many different kinds of HASH algorithms used today. Some of the most common have been CRC32, Adler32, MD5, SHA1, and WHIRLPOOL.

The cyclic redundancy check (CRC) was first invented in 1961 by W. Wesley Peterson, an American mathematician and computer scientist. The CRC operates by working on multiple blocks of data at one time. However, the CRC is not a cryptographic function. It is a linear function. This means that there is a series of steps that the function needs to go through in order to find the errors. The following is an example of a CRC algorithm [3]:

$$F(X) = Xn-kG(X) + R(X) = Q(X)P(X),$$

The Adler32 algorithm was developed by Mark Adler in 1995. Here the algorithm calculates two 16-bit checksums and then links them together to form a 32-bit result. The problem with this is that if the data to be analyzed (called a message) is too small than errors will occur [4].

The Message-Digest-Algorithm 5 (MD5) is a cryptographic function designed by Professor Ronal Rivest in 1991 as a stronger version, more secure version of MD4. The algorithm breaks down the message into 512-bits and processes it through the following function [5]:

$$Hi+1 = f(Hi, Mi), 0 \le i \le t - 1.$$

The Secure HASH Algorithm (SHA-1) is another cryptographic function developed by the National Institute of Science and Technology (NIST) as a processing standard. The algorithm has been widely used in government and industry security checks. The algorithm also processes messages in 512-bit blocks. The following is an example of the SHA-1 algorithm [6]:

$$mi = (m_{i-3} \bigoplus m_{i-8} \bigoplus m_{i-14} \bigoplus m_{i-16}) << 1$$

WHIRLPOOL is one of the first freeware algorithms in the market today. Designed by Paulo S. L. M. Barreto, a cryptographer from Brazil, it is a 512-bit hashing function that works with messages less than 2^{256} in length. Figure 1-5 is a diagram of the algorithm [7]:

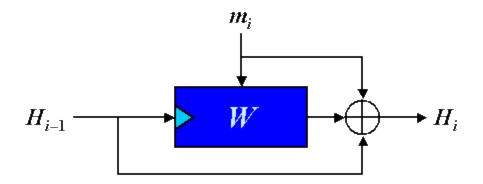


Figure 1-5: WHIRLPOOL Algorithm Diagram. Source: http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html

Problems with HASH Values

Some of the most commonly used HASH values are the MD5 and SHA-1. However, it is known that neither HASH algorithm is infallible. In 1993 collisions were found that caused some concern with the MD5's algorithm [8]. Collision means that two different documents when put through the HASH's algorithm will come up with the same value. Thus new algorithms have been created to improve upon security. Newer values like SHA-2, WHIRLPOOL and Tiger-192 have not caused problems just yet in testing.

A famous example of HASH collision is "The Story of Alice and her Boss," created by Magnus Daum and Stefan Lucks as a way of illustrating how two files can have the same HASH value. Alice is an intern with Caesar and brings a letter of recommendation for Caesar to sign [9]. The HASH value is shown in Figure 1-6.

letter_of_rec.ps																	
Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
000000000	25	21	50	53	20	41	64	6.5	62	65	2D	31	2E	30	0D	ΟA	%!PS-Adobe-1.0
00000 MD5	(128	bit)									X		78	ЗA	20	30	%%BoundingBox: 0
00000					-								20	20	20	20	0 612 792
00000				,		,							20	0D	ΟA	28	(
00000				10	r lette	r_or_	rec.p	IS:					E5	9C	C1	A7	èB¦jÞM àÕ∎_øå∎ÁS
00000	425F7	FOB2	29EEI	DB 39	68C8	6073	8533	A4BS)				B1	FB	7F	08	∕Ê·∎ F~ªÀ T>±û 📄
00000									-				13	ÀЗ	78	52	Em3 üSà[ï 🛛 £xR
00000													52	84	79	EΒ	íZ3Î6∎ ∎ nEZR∎yë
00000													OF	0B	D9	CA	/½ ∎õWåv:쿪 ÙÊ
00000					C	lose		1					D4	36	1E	20	µsYÈ2ô} ,¹£vÔ6
00000													86	D8	6B	C8	ýï;∎iò}ëÊ6 S∎ØkÈ
00000				-	-		-						94	B5	03	90	ôlôNa ll`Ïolµ —
00000192	29	28	E8	42	Α6	6A	DE	4D	00	E0	D5	89	5F	F8	E5	9C)(èB¦jÞM àÕ∎_øå∎
00000208	C1	Α7	2F	CA	В7	97	ΟA	46	7E	AA	CO	03	54	ЗE	B1	FB	ÁS∕Ê∙∎ F~ªÀ T>±û
00000224	7F	08	45	6D	33	05	01	FC	53	E0	5B	EF	ΑO	83	13	АЗ	Em3 üSà[ï 🛛 £
00000240	78	52	ED	5A	33	CE	36	99	OD	9C	07	6E	45	5A	52	84	xRíZ3Î6∎ ∎ nEZR∎
00000256	79	FR	2도	BD	ገፑ	٩Ę	ፍና	57	됴드	76	зY	FC	RF	λλ	ባፑ	ΠR	##Z/W ∎%Må#++}/a

Figure 1-6: The Letter of Recommendation.

In the scenario Alice also wants to have a security clearance and sets up an algorithm so that two documents will have the same MD5. The altered document for security is shown below in Figure 1-7.

letter_of_rec.ps	order	.ps															
Offset	0	1	2	3	4	5	6	- 7	8	9	10	11	12	13	14	15	A
0 MD5 (128	hit)		1000	-		100	-	100		x		31	2E	30	0D	ΟA	%!PS-Adobe-1.0
00	Dity		100	-	1000	-	1.00	1.00	-			6F	78	ЗA	20	30	%%BoundingBox: 0
00											1.11	20	20	20	20	20	0 612 792
00				for c	order.	ps:						20	20	0D	ΟA	28	(
	7F0B2		מכסו		2072	0500						F8	E5	9C	C1	A7	èB¦jÞM àÕ∎_øå∎ÁS
	TUDZ	JEEU	10.330		507.5	00000	403					ЗE	B1	FB	7F	08	∕Ê· F~ªÀ T>±û _≡
00											1 H	83	13	23	79	52	Em3 üSà[ï ∎ #yR
00											- 11	DA	52	84	79	EΒ	íZ3Î6∎ ∎ nEÚR∎yë
00				~							- 11	AA	0F	0B	D9	CA	/½ ∎õWåv:쿪 ÙÊ
00				q	ose						- 11	76	D4	36	1E	20	µsYH2ô} ,¹£vÔ6
				_								53	86	58	6B	С8	ýï;∎iò}ëÊ6_S∎XkÈ
00000176	F 4	94	F4	4E	61	IF	70	84	80	60	UF	EF	94	B5	03	90	ô∥ôNa ∥∥`Ïï∥µ —
00000192	29	28	E8	42	Α6	6A	DE	4D	00	ΕO	D5	89	5F	F8	E5	9C)(èB¦jÞM àÕ∎_øå∎
00000208	C1	Α7	2F	CA	В7	97	ΟÀ	46	7E	ÅΆ	C0	03	54	ЗE	B1	FB	ÁS∕Ê∙∎ F~ªÀ T>±û
00000224	7F	08	45	6D	33	05	01	FC	53	ΕO	5B	EF	ΑO	83	13	АЗ	Em3_üSà[ï∎£
I 00000240	78	52	FD	5۵	33	CF	36	99	ΠD	90	07	6F	45	5∆	52	84	▼R173Î6∎ ∎ nF7R∎

Figure 1-7: The Altered Document.

How could this happen? Two files are not supposed to have the same HASH value right?

In theory a HASH algorithm is supposed to be collision proof. Two cryptographers, Bert von Boer and Antoon Bosselaers, published a paper in 1994 explaining an algorithm that could find collisions within the MD5 algorithm. In the paper they talk about how the search algorithm can search for collisions in the MD5 algorithm by going through four different rounds [9]. After its publication a group of scientists took it a step further.

Xiaoyun Wang and Hongbo Yu are both scientists at Shandong University China. In 2005 they published a paper about the different ways to create collisions and break HASH functions such as MD5. Their results showed how easily the algorithm could be broken by putting a file through different test rounds and generating the result [5]. Wang and Yu went on to publish another paper about collision attacks against SHA-1 with Yiqun Lisa Yin later in 2005 [6].

NIST and NIJ work with HASH

There has been significant work done with HASH values aside from testing for collisions. The National Institute of Justice (NIJ) began work with the National Software Reference Library (NSRL) to create a program that would help a computer forensic examiner [10]. When working on a case with digital files a computer forensic specialist must determine what files are the most important for analysis. The new Reference Data Set (RDS) contains software profiles that can help an examiner find these files. In this dataset a file is given a profile and a HASH value unique to the dataset allowing for faster results. This is an ongoing project and is continually updated [11].

Because of the need for security, there is a need for "collision proof" algorithms. The National Institute of Science and Technology (NIST) had a competition in 2007 for the

generation of the SHA algorithm, SHA-3. Sixty-four submissions were received for the first round and were narrowed down to five finalists with the help of public opinion. The hope was that after the selection the algorithm would be able to be collision-proof for at least twenty years. The finalists were placed through a series of tests that would test for strength and compatibility. The algorithms not only had to be able to handle large messages, but shorter ones as well [12]. In 2012 the algorithm Keccak, designed by Guido Bertoni et al., was chosen as the new SHA-3 algorithm.

When using a HASH algorithm through a software program it is important for the user to know that they are getting accurate results. There is some good forensically sound software, like WinHex, that is used by law enforcement when analyzing cases. Unfortunately, this software is often limited to only law enforcement use. Freeware software allows for personal and professional use. This paper proposes a way to test some of these programs for validity.

CHAPTER II

PROPOSED FRAMEWORK AND EXPERIMENTAL PROCESS

In the forensic sciences, it necessary to employ techniques and tools that are known to generate repeatable and reproducible results. This means that the testing procedure must be able to generate the same results when done by another person. During testing a series is run more than once in order to make sure that accuracy is maintained without bias to allow for reliability and consistency.

The following is a proposal when testing different freeware HASH programs for validity and accuracy.

In this series of tests ten different kinds of file types were used. The files were chosen based on commonality and what can be easily found in a personal computer. For each file type one hundred files were created. The files were numbered from one to one hundred. Example: Book001.xlsx, this is the first file for the Excel Documents. The file types made were:

- Excel Documents
- Word Documents
- JPEG Image Files
- Notepad Text File
- PDF Document
- MP3 Audio File
- WAVE Audio Fie
- WMA File

- AVI Video File
- QuickTime Video File

Once each file was created ten different kinds of HASH checker utilities were used to check both the MD5 and SHA-1 HASH values. Eight of the programs were freeware programs that can be found and downloaded from the internet and two programs were forensically sound programs commonly used by forensic examiners. The reason for this was to provide the ground truth when comparing HASH values to each other. These values will be called root values. The programs used were:

- FTK, forensically sound program
- WinHex, forensically sound program
- Advanced Hash Calculator (AHC)
- Arpoon
- Febooti Hash-CRC
- HashTab
- HashGenerator
- MD5-SHA1 Hash Utility
- IgorWare Hasher
- SFVNinja

Each of the different file types were put into a program one at a time and then run three times to for later comparison. The value was copied and placed into an Excel spreadsheet. Three columns were labeled Test 1, Test 2, and Test 3 for each of the different runs.

4	Α	В	C	D
1	File Name	Test 1	Test 2	Test 3
2	Book001	a2363b2057eeb8f5f33a2150ee380f14	a2363b2057eeb8f5f33a2150ee380f14	a2363b2057eeb8f5f33a2150ee380f14
3	Book002	3c5fb96eed88d02aa6c800351a5fd32e	3c5fb96eed88d02aa6c800351a5fd32e	3c5fb96eed88d02aa6c800351a5fd32e
4	Book003	0294a1546a21f54368f58b0e25153925	0294a1546a21f54368f58b0e25153925	0294a1546a21f54368f58b0e25153925
5	Book004	61dc6135505a045c3f496e93546b9db1	61dc6135505a045c3f496e93546b9db1	61dc6135505a045c3f496e93546b9db1
5	Book005	8e60cea5d634646bbd69794275084d13	8e60cea5d634646bbd69794275084d13	8e60cea5d634646bbd69794275084d13
7	Book006	1fa78edadab9c018ae42ade44e3f6246	1fa78edadab9c018ae42ade44e3f6246	1fa78edadab9c018ae42ade44e3f6246
3	Book007	281d1ac905c116d497eeb1ae0e6872f7	281d1ac905c116d497eeb1ae0e6872f7	281d1ac905c116d497eeb1ae0e6872f7
9	Book008	ef49e68d7f1da886dcdfcf6837931867	ef49e68d7f1da886dcdfcf6837931867	ef49e68d7f1da886dcdfcf6837931867
0	Book009	2921c6466bb35e4005b0dfaf9b17861a	2921c6466bb35e4005b0dfaf9b17861a	2921c6466bb35e4005b0dfaf9b17861a
4	D1-010	242462-501	242462-501	242462-501

Figure 2-1: Excel Spreadsheet Example.

After all the HASH values were gathered and placed into Excel MATLab was used to compare each of the value to ensure that they were the same. A script created by Catalin Grigoras was used for this. The full script can be found in the Appendix.

In MATLab each freeware value was compared to the two root values, first FTK then WinHex. The Excel documents were loaded in MATLab and the script was run. First the three columns were compared to see if the values matched. If they did a 1 would appear in the row and a message "All the HASH values match" would appear.

```
EDU>> SeanThesis01
All the c11 HASH values match.
All the c12 HASH values match.
All the c13 HASH values match.
All the c21 HASH values match.
All the c22 HASH values match.
All the c23 HASH values match.
All the HASH values match.
```

Figure 2-2: All HASH Values Match.

c1	=		
	1	1	1
	1	1	1
	1	1	1
	1	1	1
	1	1 1 1	1
	1	1	1
	1	1	1
	1	1 1 1	1
	1	1	1
	1	1	1
	1	1	1
		1	1
	1 1	1 1 1	1 1 1 1 1 1 1 1
	-	-	-

Figure 2-3: All HASH Values Match, columns.

If the values did not match, a 0 would appear in the work and a message "Check the HASH

differences" would appear.

```
EDU>> SeanThesis01
All the c11 HASH values match.
All the c12 HASH values match.
All the c13 HASH values match.
All the c21 HASH values match.
All the c22 HASH values match.
All the c23 HASH values match.
Check the HASH differences.
```

Figure 2-4: HASH Value Differences.

c1 =		
o	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Figure 2-5: HASH Value Differences, column.

If a difference was detected the file would be run again through that HASH program and then run through MATLab again. If a value of 0 appeared again then the file was run through the program once more as well as MATLab. If the value was again 0 then it would be assumed that there was something wrong within the freeware HASH checker and it would fail the test.

Once all three columns had matching values the freeware file values were compared to the root value. If the values all matched a value of 1 was given and a message "All the HASH values match" would appear. If the values did not match a value of 0 was given and a message "Check the HASH differences would appear." If a difference was detected the file would be run again through that HASH program and then run through MATLab again. If a value of 0 appeared again then the file was run through the program once more as well as MATLab. If the value was again 0 then it would be assumed that there was something

wrong within the freeware HASH checker. Results were recorded in an Excel spreadsheet. A green X was used to indicate that all files matched and a red O was used to indicate where files did not match despite being tested three different times.

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	X	x
АНС	X	x
Arpoon	X	x
Febooti Hash-CRC	X	x
HashTab	X	x
HashGenerator	x	x
MD5 SHA1	x	x
IgorWare Hasher	X	x
SFVNinja	X	x

Table 1: Excel Document MD5 Results

Table 2: Excel Document SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	x	x
АНС	x	x
Arpoon	x	x
Febooti Hash-CRC	x	X
HashTab	X	X
HashGenerator	X	X
MD5 SHA1	X	X
IgorWare Hasher	x	x
SFVNinja	X	X

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	x	x
АНС	X	X
Arpoon	X	X
Febooti Hash-CRC	X	х
HashTab	X	х
HashGenerator	X	х
MD5 SHA1	X	х
IgorWare Hasher	X	х
SFVNinja	X	x

Table 3: JPEG Image File MD5 Results

Table 4: JPEG Image File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	x	x
AHC	x	X
Arpoon	X	X
Febooti Hash-CRC	X	x
HashTab	X	x
HashGenerator	X	x
MD5 SHA1	X	x
IgorWare Hasher	X	x
SFVNinja	X	x

Table 5: PDF File MD5 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	X
WinHex	X	X
АНС	x	X
Arpoon	x	X
Febooti Hash-CRC	x	X
HashTab	X	X
HashGenerator	x	X
MD5 SHA1	x	x
IgorWare Hasher	X	X
SFVNinja	X	X

Table 6: PDF File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	X	x
АНС	X	x
Arpoon	X	x
Febooti Hash-CRC	x	x
HashTab	x	x
HashGenerator	X	x
MD5 SHA1	X	x
IgorWare Hasher	X	x
SFVNinja	X	x

Table 7: Text File MD5 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	X
WinHex	x	x
AHC	x	X
Arpoon	x	X
Febooti Hash-CRC	x	X
HashTab	x	X
HashGenerator	x	X
MD5 SHA1	x	x
IgorWare Hasher	x	X
SFVNinja	x	x

Table 8: Text File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	X	x
АНС	x	x
Arpoon	x	x
Febooti Hash-CRC	x	x
HashTab	X	x
HashGenerator	X	x
MD5 SHA1	X	x
IgorWare Hasher	X	х
SFVNinja	X	X

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	x	x
АНС	x	X
Arpoon	X	X
Febooti Hash-CRC	X	X
HashTab	X	X
HashGenerator	X	Х
MD5 SHA1	X	х
IgorWare Hasher	x	х
SFVNinja	X	X

Table 9: Word Document MD5 Results

Table 10: Word Document SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	X	x
АНС	X	x
Arpoon	x	x
Febooti Hash-CRC	x	x
HashTab	x	X
HashGenerator	x	X
MD5 SHA1	x	X
IgorWare Hasher	x	X
SFVNinja	x	X

Table 11: WAVE File MD5 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	X	x
АНС	X	x
Arpoon	X	x
Febooti Hash-CRC	X	x
HashTab	X	x
HashGenerator	X	x
MD5 SHA1	X	x
IgorWare Hasher	X	x
SFVNinja	X	x

Table 12: WAVE File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	x	x
АНС	x	x
Arpoon	X	x
Febooti Hash-CRC	X	X
HashTab	X	х
HashGenerator	X	х
MD5 SHA1	x	x
IgorWare Hasher	x	x
SFVNinja	X	x

Table 13: MP3 File MD5 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	x
WinHex	X	x
АНС	X	x
Arpoon	X	x
Febooti Hash-CRC	X	x
HashTab	X	x
HashGenerator	X	x
MD5 SHA1	x	x
IgorWare Hasher	X	x
SFVNinja	X	x

Table 14: MP3 File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	x	x
АНС	x	x
Arpoon	X	x
Febooti Hash-CRC	X	X
HashTab	X	х
HashGenerator	X	х
MD5 SHA1	x	x
IgorWare Hasher	x	x
SFVNinja	X	x

Table 15: WMA File MD5 Results

_

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	X	X
WinHex	X	X
АНС	X	X
Arpoon	x	x
Febooti Hash-CRC	x	X
HashTab	X	X
HashGenerator	X	X X
MD5 SHA1	X	
IgorWare Hasher	X	X
SFVNinja	X	X

Table 16: WMA File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTК	X	x
WinHex	X	x
АНС	x	x
Arpoon	x	x
Febooti Hash-CRC	X	x
HashTab	X	x
HashGenerator	X	х
MD5 SHA1	X	х
IgorWare Hasher	X	X
SFVNinja	X	x

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	x	x
АНС	X	X
Arpoon	x	X
Febooti Hash-CRC	X	X
HashTab	X	X
HashGenerator	X	x
MD5 SHA1	X	x
IgorWare Hasher	x	x
SFVNinja	x	X

Table 17: AVI Video File MD5 Results

Table 18: AVI Video File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>	
FTK	x	x	
WinHex	x	x	
АНС	x	X	
Arpoon	X	X	
Febooti Hash-CRC	x	X	
HashTab	X	X	
HashGenerator	X	x	
MD5 SHA1	X	x	
IgorWare Hasher	x	x	
SFVNinja	x	X	

Checker Name	<u>FTK</u>	<u>WinHex</u>	
FTK	x	x	
WinHex	x	x	
АНС	x	X	
Arpoon	X	X	
Febooti Hash-CRC	X	X	
HashTab	X	X	
HashGenerator	X	х	
MD5 SHA1	X	х	
IgorWare Hasher	X	x	
SFVNinja	X	X	

Table 19: QuickTime Video File MD5 Results

Table 20: QuickTime Video File SHA-1 Results

Checker Name	<u>FTK</u>	<u>WinHex</u>
FTK	x	x
WinHex	x	x
AHC	x	X
Arpoon	X	X
Febooti Hash-CRC	X	X
HashTab	X	X
HashGenerator	X	x
MD5 SHA1	x	x
IgorWare Hasher	X	x
SFVNinja	x	x

CHAPTER III

CONCLUSION

As seen in the previous tables there were no errors when gathering the HASH values from these freeware programs. However, this represents only a fraction of the different kinds of programs available online for download off of the internet. It is recommended that new programs are tested in a similar way, if not the same way, before they are considered to be verified. Testing other HASH algorithms, such as WHIRLPOOL, can help in the verification process. As a reminder MD5 and SHA-1 used together helps to safeguard against errors when safeguards are checked.

Freeware software is something that can be used to verify digital evidence from a working case. Proper protocol for handling digital files should still be followed depending on the Standard Operating Procedure (SOP) setup by a department. It is still recommended to double check the HASH values with a forensically based program like FTK, but when gathering evidence at a crime scene freeware programs can be used.

This thesis has proposed a way of testing freeware software for errors, but not for collisions. Future tests can be done with the dataset to see if collisions occur when the HASH of each file is compared to the other.

BIBLIOGRAPHY

- 1. *What is a Hash Function*? 2012, EMC Corporation. 7 Jul 2013 <<u>http://www.rsa.com/rsalabs/node.asp?id=2176</u>>
- 2. Lu, Wenjun, Avinash L. Varna and Min Wu. "Forensic Hash for Multimedia Information." *Media Forensics and Secutiry II* 7541 (2010).
- 3. Peterson, William Wesley, and Daniel T. Brown. "Cyclic codes for error detection." *Proceedings of the IRE* 49.1, 1961: 228-235.
- 4. Maxino, Theresa. "Revisiting Fletcher and Adler Checksums." (2006).
- 5. Wang, Xiayon, and HongBo Yu. "How to Break MD5 and Other Hash Functions." *Lecture Notes in Computer Science* 3494 (2005): 19-35
- 6. Wang, Xiaoyun, Yiqun Lisa Yin, and Hongbo Yu. "Finding collisions in the full SHA-1." *Advances in Cryptology–CRYPTO 2005*. Springer Berlin Heidelberg, 2005.
- 7. *The WHIRLPOOL Hash Function*. Ed. Paulo S. L. M. Barreto. 2008. 7 Jul 2013 http://www.larc.usp.br/~pbarreto/WhirlpoolPage.html
- 8. den Boer, Bert, and Antoon Bosselaers. "Collisions for the compression function of MD5." *Lecture Notes in Computer Science* 765 (1994): 293-304
- 9. Daum, Magnus and Stefan Lucks. Hash Collisions (The Poisoned Message Attack): "The Story of Alice and her Boss". 26 May 2013 < http://th.informatik.unimannheim.de/people/lucks/HashCollisions/>
- National Institute of Standards and Technology (NIST), and United States of America. "Forensic Examination of Digital Evidence: A Guide for Law Enforcement." 2004
- 11. *National Software Reference Library*. 20 Aug. 2003. National Institute of Software and Technology. 7 Jul. 2013. ">http://www.nsrl.nist.gov/>
- 12. Turan, Meltem Sonmez et al. *Status Report on the Second Round of the SHA-3 Cryptographic Hash Algorithm Competition*. NIST Interagency Report 7764, 2011.

APPENDIX

HASH Programs

WinHex

WinHex - [Thesis Audio 00 File Edit Search Nav		s Specialist Options Window Help	
	ю в 🛱 в 		17.0 SF
	Thesis Audio 00		17:0 51
iļe E <u>d</u> it			
	Offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	[unregistere
	0000000	49 44 33 04 00 00 00 00 00 17 54 53 53 45 00 00 ID3 TSSE	Thesis Audio 001.mp3
	00000016	00 0D 00 00 03 4C 61 76 66 35 32 2E 39 33 2E 30 Lavf52.93.0	J:\Thesis\Test_Documents
	00000032	00 FF FB B0 44 00 0C 83 Å9 54 43 13 09 1Å F0 79 ÿå*D ∎@TC öy	File size: 235
	00000048	8F 07 A0 3C 63 6E 52 B5 8B 18 4C A5 0D C2 4B 3C (cnRµ] L¥ ÅK(File size. 235 240,777 by
	00000084	E0 18 F3 19 B8 01 01 95 1D 2D FA C0 29 A0 50 06 à ó , I -úÅ) P 62 2A 05 CE 84 B0 C0 44 63 40 60 6C 01 9F 41 71 b≭ ÎI*ÅDc@`l IAg	
	00000096	23 6F 54 40 81 8F 6D D5 A3 7A 91 6E 33 92 8C 2E #oT@ nŐ£z'n3'].	DOS name: THESIS~1.M
	00000112	8E 90 23 47 3E BB 75 08 4F D0 40 02 00 08 4E B9 #G>wu OĐ@ N1	Default Edit Mode
	00000128	C2 E0 B3 FA EE EF A0 01 3F F9 D0 EF C4 CF 4D DC Åà ³ úîî ?ùĐiĂĬMÜ	State: origi
	00000144	DC 5B 92 22 10 71 67 F1 38 95 DC 42 21 39 C4 24 U[1" qqf88 UB!9Å\$	Undo level:
	00000160	44 FC FD DD CE BB BB 96 9F A7 1D DC 38 BF 0F C3 DüýÝÍ.»»↓ \$ Ü8¿ Ã	Undo reverses:
	00000176	13 EE CB BF AA 50 E7 FF E7 35 8F C5 3A 8D FB F6 1Ê∂Pçÿç5 Å: úö	GIUGTEVEISES.
	00000192	35 7C 7F 7F 4A 53 4E EE 94 0C 5B C4 42 40 82 0B 5 JSN1 [AB0]	Creation time: 04/12/2
	00000208	E0 41 0E 06 2F 0E 06 EE 6E 7B 9F A2 44 E2 7F 00 àA / în{ cDâ	15:04
	00000224	C5 FB BC 2F 4D F4 44 43 FD FD DC BC AB FD 08 22 Âû¼/MôDCýýܼ«ý "	Last write time: 04/09/2
	00000240	87 03 73 88 06 06 06 2C CA C2 EB BB 81 BB 9B 9F s ,ÉAë» »	20:53
	00000256	FB BB A7 B9 D7 70 E2 09 DC E7 E9 45 9E 84 88 6F ûxS'xpâ ÜçéEIIIo	Attributes:
	00000272	FA 7F AO 00 82 C2 26 E7 FC 28 8E 6E 61 04 17 D3 ú Å&çü(na Ó	Autoutes.
	00000288	88 4E 7F F5 E8 18 18 1B FC 4D 3D 1F 44 BE E6 EE N õè üN= D%asî	icons.
	00000304	EE 10 00 40 0B 57 9D 07 82 B2 B8 A9 8B A8 59 F2 î @ V 2,@ `Yò	Mode: hexadeci
	00000320	DE B1 94 AC 6E E9 9A A9 97 E3 66 8A 04 A6 06 1B b± =né © ãf ;	Character set: CP 12
	00000336	12 OF 94 6D 22 86 5B 2E 52 C8 E2 26 0A 12 A9 91 [n"[[.RÊÂ& @'	Offsets: deci
	00000352	6B 08 D9 44 F5 8F A8 82 79 27 C2 4D CA 0D EE DD k ÙDS "↓y'ÅMÉ IÝ 21 53 FE C3 E1 0A B6 1C EA 59 15 11 6C 15 FE C5 !Sbãá ¶ é¥ l bå	Bytes per page: 39x16=
	00000388	11 C4 41 39 56 E3 4E B1 AA 2B 10 4E 3D E2 68 4A ÅA9VäN± ² + N=åhJ	Window #:
	00000384	26 61 95 4D 23 8A 1C BD 17 56 6D 77 64 D9 CA FO &a M# # WwwdDE8	No. of windows:
	00000416	3A DA 26 52 D5 D1 4F 10 6C BE 8A 5B 22 AC B4 19 :Ú&RÔÑO 1¾[["~'	
	00000432	71 8F EB 96 77 47 72 AD 3A DE 13 6A 84 C7 05 96 g ë wGr-: Þ j C	Cipboard: availa
	00000448	07 0A 82 22 1A 3B FE 9D 03 F6 0D 08 01 12 C3 8C ";b ö Å	TEMP folder: 448 GB f
	00000464	EA 2E 2D EC 43 9E B3 C5 7C 4D CD 33 07 D6 E1 16 ê\4C 3Å M13 Öá	NJA~1\AppData\Local\Te
	00000480	3A 11 4F 6B 64 B6 D0 19 ED 4D 4A CE 81 8E 08 4F : OkdMÐ iMJÍ O	
	00000496	AC E2 OC 99 AO C3 OE 2C 71 8E A3 12 54 35 F5 05 -4 A .qle T56	
	00000512	45 99 E9 9D A2 ED ED 46 20 7B DD B6 1C 45 05 6A Elé ¢11F (ÝN E j	
	00000528	47 1C CF 79 A3 D6 A1 D1 A4 6E EF 29 CE 87 EE 83 G Ïy£ÖIѤnï)Î i	
	00000544	B4 B4 C4 DD 67 7C 35 E7 77 72 3F 34 AF AB EB 79 '`ÄŸg 5çwr?4 «ëy	
	00000560	BD 68 77 F7 F7 65 E1 33 FC 36 D1 0D 86 83 2F 4A %hv++eá3ü6Ñ /J	Data Interpreter
	00000576	A2 37 36 F5 EA 61 B6 3A AD 0B 08 BB AB C1 06 E9 ¢768êa¶:- >><á é	Data Interpreter
	00000592	FE 35 2D A9 7D 8C 39 9E CA 3B 21 1C 00 40 14 92 b5-@} 9 Ê;! @ '	8 Bit (±): 73
	00000608	72 05 B0 46 C7 F1 C2 4D E0 42 2B A9 9E 8F 58 93 r "FÇñÅMàB+© X	16 Bit (±): 17481
	Page 1 of 386	Offset: 0 = 73 Block:	n/a Size: 32 Bit (±): 70468681

IgorWare Hasher

🕽 IgorWare Hashe		
Options Help		
File	Text	
		Browse
Verification data	3	
SHA-1:		Verification file not found
MD5:		Verification file not found
CRC32:		Verification file not found
Hash Result		
V SHA-1:		
MD5:		CRC32: Copy Save
	Ready	Calculate

MD5-SHA1 Utility

MD5 &	SHA-1 Checksum Utility 1.1	
Help		
Genera	te Hash	
File:		Browse
MD5:		Copy MD5
SHA-1:		Copy SHA-1
	Verify Hash with Generated Hash (MD5 or SHA-1)	Copy All
Hash:		Paste
	Verify	

Advanced Hash Calculator

Advanced Hash Calculator						
le Actio	on Options	Help				
	2	X		- 🦀	?	
ew List	Add Files	Clear	Calculate	Settings	About	
elected F	ïles					Hash
ag and drop files from Windows Explorer or click [Add Files] button Hash: SHA-1						

Febooti HASH-CRC

1	Book001.xlsx Properties
-	General Hash / CRC File Hashes Security Details Previous Versions
	Name: Book001.xlsx Location: J:\Thesis\Test_Documents\Excel_Files\
	CRC32
	Show more
	915.92 KB/s View file: Book001.xlsx Compute Copy More
	FileTweak module from Get more modules
L	OK Cancel Apply

HashTab

General Hash / CRC File Hashes Security Details Previou						
Name	Hash Value					
CRC32	CF87EA63 A2363B2057EEB8F5F33A2150EE380F14 2E742DFE45EB3A1C2DBA99DE332A6930FEC234					
MD5 SHA-1						
	arison -					
Settings Hash Comp I						
<u>Settings</u> Hash Comp [arison: Compare a file					

Hash Generator

Hash Generator - www.Se	urityXploded.com	-	
# H		Generator in-one Hash Generator Software	Show Help About
Select Input Type	I File ○ Text		Show Help About
Select File Location	C:\windows\notepad.exe		3
			Generate Hash
Hash Type H	sh Length 🛛 Hash Value		
Download More Free Tool	from SecurityXploded		Report

MATLab Script

```
clear
% cd directory
cd J:\Thesis\Test Documents\
%cd J:\Thesis\Test Documents\Excel Files\Excel Program HASHes\SHA1
% cd
%Get root file
[ExcelRoot, Root] = uigetfile ('*.xlsx', 'Select Root Excel File');
addpath (Root);
%Get comparison file
[ExcelCompare, Compare] = uigetfile('*.xlsx', 'Select Comparison Excel
File');
addpath (Compare)
% read the Excel files
%[sean01,txt01]=xlsread('Checksums.xlt');
%[sean02,txt02]=xlsread('HASH CRC.xlt');
%[sean01,txt01]=xlsread('Excel File MD5 HASH FTK.xls');
%[sean02,txt02]=xlsread('Excel File MD5 HASH WinHex.xls');
%MD5
%[sean01,txt01]=xlsread('Excel File MD5 WinHex');
%[sean02,txt02]=xlsread(ExcelCompare);
%SHA1
[sean01,txt01]=xlsread(ExcelRoot);
[sean02,txt02]=xlsread(ExcelCompare);
% compute the tables lines and columns numbers
[a01,b01]=size(txt01);
[a02,b02]=size(txt01);
% compare the HASH values of the Test1...Test3 columns
% txt01
for k1=2:a01
    c11(k1-1)=strcmpi(txt01(k1,2),txt01(k1,3));
    c12(k1-1)=strcmpi(txt01(k1,2),txt01(k1,4));
    c13(k1-1)=strcmpi(txt01(k1,3),txt01(k1,4));
end
if prod(c11(:)) == 1
    disp('All the cl1 HASH values match.')
elseif prod(c11(:))==0
    disp('Check the cl1 HASH differences.')
end
if prod(c12(:)) ==1
    disp('All the c12 HASH values match.')
```

```
elseif prod(c12(:))==0
    disp('Check the c12 HASH differences.')
end
if prod(c13(:)) ==1
    disp('All the c13 HASH values match.')
elseif prod(c13(:))==0
    disp('Check the c13 HASH differences.')
end
% txt02
for k1=2:a02
    c21(k1-1)=strcmpi(txt02(k1,2),txt02(k1,3));
    c22(k1-1)=strcmpi(txt02(k1,2),txt02(k1,4));
    c23(k1-1)=strcmpi(txt02(k1,3),txt02(k1,4));
end
if prod(c21(:)) ==1
    disp('All the c21 HASH values match.')
elseif prod(c21(:)) == 0
    disp('Check the c21 HASH differences.')
end
if prod(c22(:)) ==1
    disp('All the c22 HASH values match.')
elseif prod(c22(:))==0
    disp('Check the c22 HASH differences.')
end
if prod(c23(:)) ==1
    disp('All the c23 HASH values match.')
elseif prod(c23(:))==0
    disp('Check the c23 HASH differences.')
end
% compare the HASH values of two different Excel files
for k1=2:min(a01,a02)
    for k2=2:min(b01,b02)
        c1(k1-1,k2-1)=strcmpi(txt01(k1,k2),txt02(k1,k2));
    end
end
if prod(c1(:))==1
    disp('All the HASH values match.')
elseif prod(c1(:))==0
    disp('Check the HASH differences.')
end
```