DOI: http://dx.doi.org/10.5769/C2011015

Creation and Maintenance of MD5 Hash Libraries, and Their Application in Cases of Child Pornography

Alexandre Vrubel
Computer Forensics Laboratory
Criminalistics Institute of Curitiba - Paraná
Curitiba, Brazil
alexandre.vrubel@ic.pr.gov.br

***Abstract* — Forensic programs, such as *Guidance EnCase*, allow the use of hashes of known files, to ease the selection of files to be analyzed or ignored in Computer Forensics examinations. However, these same programs do not allow an effective maintenance of their hash libraries. To remedy this deficiency, the *HashManager* tool was developed, which is able to create, manage and use libraries of millions of MD5 hashes in seconds. *HashManager* allows the calculation of the union, intersection and difference of hash libraries. These libraries are stored and processed as simple text files, but *HashManager* is able to convert them to the *EnCase* format, if needed. Hash libraries of files related to child pornography have been built, with the aim of increasing productivity in these types of examination. These hash libraries, *HashManager* and other auxiliary tools are freely available to the community of Computer Forensics.**

Keywords — hash libraries, child pornography, pedophilia, MD5, HashManager

1. Introduction

The computer forensic examinations use routinely file hashes. Hashes are codes of fixed length, typically represented in hexadecimal notation, resulting from the application of algorithms (hash functions) on an original file of arbitrary size.

Two properties are important for the use of hashes. The first is that the hash functions are one-way, i.e. one cannot reconstruct the original file from its hash. The second is that given a file x is computationally infeasible to find another file y different from x that generates the same hash code. These properties make the hash of a file to work as a kind of fingerprint of it, i.e. a small code that uniquely identifies the file. A good introduction on hashes is provided on [1].

Forensic tools such as Guidance EnCase or AccessData FTK allow the use of hash libraries of known files, and their comparison with the hashes of the files present in the case being examined. There are two uses: files to be ignored (e.g. files shipped with the operating system), and suspicious files to be analyzed by the examiner (e.g. known pornographic images involving children).

Despite the effective use of hash libraries, the same forensic tools do not provide practical ways to manage custom libraries of forensic hashes. They allow the import or update of third party hash libraries, or the addition of hashes of files present in the case being examined to libraries. Operations such as intersection between libraries, or subtraction of one library from another, are not available. This means that the expert is dependent on third parties libraries, not possessing the means to easily create and maintain their own hash libraries.

To meet these needs, as well as to allow hash analysis of cases without the need for paid forensic tools, the *HashManager* tool was developed. It allowed the creation and maintenance of hash libraries related to child pornography, among other applications. It was developed and improved by crime lab experts to solve real problems found in everyday computer forensic examinations.

Section II analyzes the existing tools for the management of hashes. Section III presents *HashManager* and some auxiliary tools that increase its scope of use. Its application, especially in cases of child pornography, is presented in Section IV. Conclusions and future work are presented in Section V.

2. Existing Tools

The hash function adopted by *HashManager* is MD5. Although now no longer safe enough for cryptographic applications, it is widely used by forensic tools and hash libraries in the public domain. Besides, in comparison with other hash functions, its calculation time is one of the fastest [1].

There are several tools that can calculate MD5 hashes for both Windows and Linux systems. For Linux, there is the native command "*md5sum*", which calculates the MD5 hash of a single file. Used in combination with the native command "*find*", one can generate a text file containing the hashes of the files present in a folder and its subfolders. For the Windows system, there are the free applications "*fsum*" and "*corz checksum*", among others. Both of them generate text files compatible with those generated by the "*md5sum*" command. Another useful tool is "*md5deep*", which can be

used to create and check hashes on both Windows and Linux systems.

A forensic tool related to hash analysis is the command "*hfind*" from "*The Sleuth Kit*" package, and its "*Autopsy*" interface. This package offers a free set of forensic tools, and is mainly used in Linux. The "*hfind*" command allows the import and indexing (sorting) of hash libraries, and optimized searches of a list of hashes within a library previously indexed.

The problem with all those tools is that they were not designed with the idea of management of hash libraries in mind. Therefore, they are useful to create, check and calculate hashes of files, but not to make operations between hash libraries. What was wanted was a tool which understands hash libraries as sets, and performs basic operations between these sets, such as union, intersection and difference. i.e., given two input libraries, perform an operation between them and generate a new library as a result. No tool with these characteristics was found in our research.

For the hash libraries themselves, there are NSRL and HashKeeper. The NSRL (National Software Reference Library) is a project of the U.S. Department of Defense that collects hashes of software installation media, and currently offers over 19 million hashes of known files. This library, by law, does not contain illegal files such as images and videos relating to child pornography. On the other hand, HashKeeper was a project of the U.S. National Drug Intelligence Center, which collected MD5 hashes of known files, even illegal, to assist criminal investigations. However, the project was discontinued. In practice, obtaining hash libraries of files relating to child pornography is not easy for forensic experts.

As for the forensic tools aimed at investigations of child pornography, some highlights are "*NuDetective*", from Brazilian Federal Police [2], and Bluebear's LACE (Law Enforcement against Child Exploitation). Both tools use hash libraries as part of the techniques for identifying images and videos related to child pornography.

3. Hash Manager

The *HashManager* tool was developed at the Computer Forensics Laboratory of the Criminalistics Institute of Curitiba-PR, in order to remedy the shortcomings cited in Section II. Since 2011, the tool became part of the standard procedures in several examinations, particularly those relating to child pornography, as will be presented in Section IV.

HashManager was developed in C++ as a command line application (without a GUI). This simplified the portability of it, with compiled versions for Windows and Linux available.

MD5 hash libraries are stored in text files, using the "*md5sum*" standard format adopted by the several tools cited in Section II. Basically, each line of text file begins with the 32 hexadecimal characters of the MD5 hash, optionally followed by the name of the file that generated the hash. Lines that do not follow this pattern are ignored and treated as comments. This format has several advantages: hash libraries can be created, read or edited manually in any text-file editor; and they are easily processed by other tools, such as operating system shell scripts or scripts executed by forensic tools such as Guidance EnCase (EnScripts).

HashManager treats the hash libraries as sets, allowing operations such as union, intersection and difference, among others. The *HashManager* command line syntax is as follows: "hashmgr <operation> <1 or 2 input files> <output file>". **Table I** lists the supported operations. According to the chosen operation, it takes one or two input text files, each containing a set of MD5 hashes. The last argument is always the name of the resulting text file. There is no restriction in using one of the input files as the target of the operation.

The resulting text file from the operations of *HashManager* has always its hashes sorted in ascending order. In cases of hashes with the same value, the line used comes from the first

TABLE I. Operations supported by HashManager.

Operation	Description
sort	Sorts the hashes of the input file in ascending order, eliminating duplicates
sort2	Sorts the hashes of the input file in ascending order, preserving duplicates
add or merge	Calculates the union of the hashes of two input files, generating an output file without duplicates
inboth or intersection	Computes the intersection of the hashes from two input files, generating an output file without duplicates
inboth2 or intersection2	Computes the intersection of the hashes from two input files, generating an output file that preserves the duplicates of the 1st input file
sub	Eliminates from the 1st input file hashes present in the 2nd input file, generating an output file without duplicates
sub2	Eliminates from the 1st input file hashes present the 2nd input file, generating an output file that preserves the duplicates of the 1st input file
encase	Converts the input file to a binary output file used by the forensic tool EnCase
hsh	Converts an input file in HSH format (HashKeeper) to an output file in text format
strip	Deletes the optional file names of each line of the input file

input text file. Input files are sorted automatically if necessary. Text files in Unicode format (where all characters are encoded in 16 bits) are still not supported, and must be first converted to UTF-8 encoding, for example.

A. Algorithmic Analysis

One of the design goals of *HashManager* was its ability to manage huge hash libraries, i.e. text files containing millions of lines. In addition, the tool should be fast enough to be of practical use.

The ordering of the input data is important to speed up operations of the tool. Consider two input files with n and m hashes, respectively. If these hashes were not ordered, we would need to compare each one of the n hashes of the first file with all m hashes the second file to perform the operations of intersection, difference and union, which it is an algorithm of quadratic time complexity $O(n \times m)$. On the other hand, for sorted input files all three operations examine each hash only once (single pass), in an algorithm of linear time complexity $O(n + m)$. This method of combining two sorted lists to generate the desired result of the operation is similar to the core of the *mergesort* algorithm [3].

The fact that MD5 hashes have a fixed length of 32 hexadecimal characters allowed the use of the *radix-sort* algorithm to sort the input hashes. This algorithm has linear time complexity $O(n)$, being asymptotically faster than the commonly used *quicksort* algorithm, which has time complexity $O(n \log n)$. Moreover, the worst case time complexities for the *radix-sort* and *quicksort* are respectively $O(n)$ and $O(n^2)$. In-depth analysis of the *radix-sort* and *quicksort* can be found in [3].

Thus, the overall time complexity of *HashManager* is a linear $O(2n + 2m)$ for any operation involving two input files, already taking into account the operations of sorting through *radix-sort* of both input files. If the input files are already pre-sorted, this condition is detected during their loading and sorting is omitted, speeding up the execution of the tool to a $O(n + m)$ time complexity.

To optimize disk access operations, the input files are read entirely into memory, and then analyzed. Despite limiting the maximum size of the input files, this approach causes no problems in practice, as a library with more than 19 million hashes has an input file of about 624 MiB of size, which represents a memory cost fully acceptable for current computers.

The adopted data structure consists of arrays of pointers, which point directly to the hash strings in the image of the input file saved in memory. For sorting by *radix-sort*, only the pointers switch positions in the vector, which prevents the movement of large blocks of memory. Likewise, results of the operations between two input files are represented by an array of pointers to the lines of the input text files already in memory. Thus, the resulting output file does not spend additional space in memory other than the array of pointers.

Runtime results of *HashManager* can be seen in **Table II**. The computer used in the tests was a PC with a Core i7 3.4 GHz processor, 16 GiB of RAM and a Windows 7 Home Premium 64-bit operating system. The code was compiled using Microsoft Visual C++ Express Edition 2010. It can be noted that even when performing operations on very large libraries, the response time of *HashManager* is fully acceptable. Its performance in real cases is even better, as a typical case usually has ten to one hundred times less hashes than the amount tested here.

TABLE II. Execution times of HashManager

Operation	Number of hashes		Time in seconds
	input	*output*	
sort	15,858,540	4,813,472	40.18
add	4,813,472 4,812,512	9,625,984	22.06
intersection	19,248,432 6,155,774	3,297,701	28.55
sub	6,155,774 19,248,432	2,858,073	29.63
encase	19,248,432	19,248,432	27.67
hsh	15,858,540	15,858,540	29.92
strip	19,248,432	19,248,432	27.91

B. Auxiliary Tools

HashManager is not used alone when performing the analysis of forensic cases. Usually, it is part of a procedure, and other additional tools were developed to further explore the potential of *HashManager*. They are:

1) *EnScript for EnCase called BookmarkFromHashList* (currently available for versions 4 and 5 of EnCase): This *EnScript* reads a text file containing hashes (possibly resulting from *HashManager*), and marks the files of the case that have the hashes present in the text file into a bookmarks folder specified by the user. It is often used when multiple files are extracted from EnCase for analysis in other applications. After this analysis outside of EnCase, some files may be pertinent to the case being investigated. Now there is the need to select these files again within the EnCase environment. To avoid the manual search for these files from the thousands of files commonly found in a typical case, the script *BookmarkFromHashList* is used;
2) *Export Option of EnCase*: This native option can become very useful when used in combination with *HashManager*. It allows the export of metadata of selected files in a case to an output text file. In this case, the fields exported are "*HashValue*" and "*FullPath*", which are exported for some (or all) files of the case. The resulting text file is in encoded in Unicode format, and after being converted to UTF-8 encoding (e.g. with Notepad), it can be used as a input parameter for *HashManager*;
3) *Shell script for Linux called "hashcp.sh"*: This script reads a text file (usually resulting from *HashManager*) that follows the "*md5sum*" convention, i.e. lines with a hash followed by the file name that originated it. The script

discards the hashes and comment lines, keeping only the paths and files names, and then copies these files to a specified folder. There are variations of this shell script, called "*hashrm.sh*" and "*hashmv.sh*", which respectively delete or move files listed within the text file instead of copying them. These three scripts are useful in procedures using *HashManager* for exclusion or separation of files, as described in Section IV.

4. Applications

Following are presented some types of examinations using *HashManager*, and their technical procedures as adopted in the Computer Forensics Laboratory of the Criminalistics Institute of Curitiba-PR.

C. Child Pornography

These examinations typically have a standard question that is: "*There are, in the equipments sent for examination, images or videos with pornographic content involving children or adolescents?*" If the answer to this question is yes, then other questions are answered, such as whether the files were shared on the Internet. There are two typical scenarios when these types of files are found: or they were created by the users of the equipment being analyzed, or they were obtained from third parties, usually through the Internet. Hash libraries are useful only for the second case, where the same files are copied between different users.

For the classification of the biological maturation of individuals present in the images and videos, the experts use the criteria proposed by Bonjardim and Hegg [4], according to the stages presented by Tanner [5], and following the recommendations proposed by Amorim [6]. It should be noted that there is an age range (from about 14 to 18 years old), where the classification of person as a minor may not be conclusive. This happens because some individuals reach the last stage of biological maturity of the genital areas and the hairy distribution before 18 years of age.

Using *HashManager*, two hash libraries related to child pornography were created. The first ("*pedophilia.txt*") consists of hashes of image files or videos undoubtedly involving children or adolescents. Such files possess a conclusive and reliable classification, as they were found in real cases analyzed by the experts of the Criminalistics Institute of Curitiba-PR, with each file having been examined and classified individually.

The second library ("*suspicious.txt*") was built by merging hashes related to child pornography obtained from various sources, such as the "*NuDetective*" tool from the Brazilian Federal Police [2]. In addition to these third party hashes, files whose age classification is not conclusive are also found in this library. Despite possessing a huge amount of hashes (more than 1.2 million), some false positives can found when using this library, mainly due to the fact that most of these hashes have been obtained from third parties, without the analysis of image files or videos from which they were obtained. As they are detected in real cases, false positives are eliminated from the library, and files for which their classification is conclusive are deleted from the "*suspicious.txt*" library and added to the confirmed "*pedophilia.txt*" library. Both libraries have hashes of compressed files that have been shared on the Internet (ZIP, RAR, 7Z, etc.), containing relevant images or videos.

The library of known hashes from the NSRL (see Section II) was also converted to the format used by *HashManager* ("*nsrl.txt*"), to discard files that are not related to child pornography during examinations.

The procedures used in child pornography examinations are as follows. It should be noted that Linux and Windows tools are used simultaneously. This is accomplished using virtual machines running Windows over an Ubuntu Linux host. The main used tool is EnCase, after the acquisition of the case evidence files from the source devices.

1. Hashes of all files present in the case being analyzed are exported from EnCase (see Section III.B.2). The intersection of the hashes exported with the libraries "*pedophilia.txt*" and "*suspicious.txt*" is calculated, using the operation *inboth* of *HashManager*. If there are intersections, they are marked within EnCase with the EnScript *BookmarkFromHashList* (see Section III.B.1), and the bookmarked files are exported and analyzed. This first step already provides an idea of what will be found in the case under consideration. This sequence of operations is used because EnCase becomes very sluggish if the "suspicious" library, containing more than 1.2 million hashes, were used directly as a *Hash Set* within EnCase.
2. The other images and videos are exported from EnCase. To avoid analysis of non-relevant files, hashes of the exported files are computed with "*md5sum*" or any equivalent tool. Then the intersection of these exported hashes with the "*nsrl.txt*" library is calculated using the operation *inboth2* of *HashManager*. With the shell script "*hashrm.sh*" (see Section III.B.3), files known through NSRL are deleted, leaving only the files that need to be analyzed and classified individually.
3. The acquisition files in EnCase format (E01) are converted to a single file in the raw format (DD). This file serves as input to the "*foremost*" tool [7], used to recover deleted images and video. This step aims at detecting files embedded inside other files, and deleted files whose directory entries had already been overwritten, which hindered their recovery by EnCase.
4. To prevent the re-analysis of the files present in EnCase, the hashes of the files recovered through "*foremost*" are calculated with "*md5sum*" or any equivalent tool. Then the intersection between these "*foremost*" hashes and the EnCase hashes exported in step 1 is calculated using the operation *inboth2* of *HashManager*. With the shell script "*hashrm.sh*", the already examined files from EnCase are excluded from the output folder of "*foremost*". A similar procedure is performed to eliminate files possessing

known hashes from the NSRL library from the output folder of "*foremost*".

5. The text file containing the hashes of the "*foremost*" output folder (generated in step 4) is updated through the subtraction of the EnCase and NSRL hashes, using the *sub2* operation of *HashManager*. Then the intersection of the updated file with the "*suspicious.txt*" and "*pedophilia.txt*" libraries is calculated with *HashManager*, in order to identify illegal files in the output folder of "*foremost*". If any illegal files are found, they are moved to another folder with the shell script "*hashmv.sh*" (see Section III.B.3). The remaining files in the output folder of "*foremost*" are analyzed individually.
6. An optional step before the analysis of individual files is to remove duplicate files from the export folders of EnCase or "*foremost*". That can be accomplished through the sort operation of *HashManager*, which removes duplicates from a source hash list. The resulting hash list can then be used with "*hashmv.sh*" or "*hashcp.sh*" to respectively move or copy the "unique" files to another folder for examination.
7. Hashes of illegal files found during the analysis are added to the appropriate libraries with the *add* operation of *HashManager*. False positives or reclassifications of files also cause updates to the libraries.

Those procedures may seem complex, but the net result on the quality and productivity of child pornography examinations is very positive. Two complementary techniques of file recovery are performed (directory-based with EnCase, and file carving through "*foremost*"), and several possible types of known files ("*pedophilia.txt*", "*suspicious.txt*" and "*nsrl.txt*") are used to reduce the number of files that should be individually analyzed.

D. Search for specific files

The *HashManager* and other auxiliary tools presented in Section III work as a "Swiss army knife" for examiners, allowing their application in many cases, not necessarily just child pornography.

As presented in Section IV.A, *HashManager* can be used to eliminate known or already analyzed files from files recovered with the "*foremost*" tool, regardless of the case under consideration.

Another case where *HashManager* was applied was to search for specific files, sent in a CD, on different equipments. After calculation of the hashes of the media files and hard drives, the intersection operation of *HashManager* was used to perform the search.

5. Conclusion

This paper presented the features of *HashManager* and other auxiliary tools. Its main application has been the search of files relating to child pornography, as presented in section IV.A. Its use has increased efficiency in these examinations, allowing the automatic identification of files already found in previous cases. The tool is very versatile and is also used in other types of examinations.

Some possible future works are:

- The creation of a graphical interface for *HashManager*, making it friendlier to users;
- Add support to hash libraries larger than the amount of available memory;
- Allow the use of other hash algorithms such as SHA1 or SHA256;
- Enhance the NSRL library of known hashes with post-installation images and videos related to the operating system or other widely used applications, or the result of web browsing, found during the examinations;
- Integration of other existing libraries, related to either child pornography files or files to be ignored.

The *HashManager* tool is being supplied with its source code, along the hash libraries mentioned in the paper, free of charge. Just contact the author by the e-mail *alexandre.vrubel@ic.pr.gov.br*, requesting access to the tools and hash libraries. It is hoped this will contribute to improve the quality and efficiency of the computer forensics examinations.

Acknowledgment

The author would like to thank: the managers of the Computer Forensic Laboratory for their support to the creation and improvement of the tools and hash libraries presented in this paper; all the forensic experts of the Computer Forensic Laboratory, for their suggestions and help testing the tools; and Itamar Almeida de Carvalho, forensic expert from the Brazilian Federal Police, for sharing the MD5 hashes used in the "*NuDetective*" tool.

References

[1] RSA Laboratories, "Frequently asked questions about today's cryptography - Version 4.1", RSA Security Inc., 2000.
[2] P. Eleutério and M. Polastro, "Optimization of automatic nudity detection in high-resolution images with the use of NuDetective forensic tool", 5th ICoFCS: Brasília, 2010.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms", 3rd Ed MIT Press, 2009. ISBN 978-0262033848.
[4] E. Bonjardim and R. V. Hegg, "Crescimento e desenvolvimento pubertário em crianças e adolescentes brasileiros", São Paulo: Brasileira de Ciências, 1988.
[5] J. M. Tanner, "Growth at adolescence with a general consideration of the effects of hereditary and environmental factors upon growth and maturation from birth to maturity", 2nd ed. Oxford: Blackwell Scientific Publications, 1962.
[6] F. L. P. Amorim, "Definição de Parâmetros de Análise de Imagens Digitais Relacionados à Pedofilia", XX Congresso Nacional de Criminalística e III Congresso Internacional de Perícia Criminal: João Pessoa, 2009.
[7] J. S. Nascimento and K. S. Jerônimo, "Análise de Ferramentas Forenses de Recuperação de Dados", V Seminário Nacional de Perícias em Informática: Palmas, 2010.

Alexandre Vrubel holds a Master's degree in Computer Science from the Federal University of Paraná, and is also a graduate in Informatics from the Federal University of Paraná. Currently works as a forensic expert in the Computer Forensics Laboratory of the Criminalistics Institute of Curitiba - Paraná.