Computer Forensic Functions Testing: Media Preparation, Write Protection and Verification

Yinghua Guo
School of Computer and Information Science
University of South Australia
yinghua.guo@unisa.edu.au

Jill Slay
School of Computer and Information Science
University of South Australia
jill.slay@unisa.edu.au

ABSTRACT

The growth in the computer forensic field has created a demand for new software (or increased functionality to existing software) and a means to verify that this software is truly forensic i.e. capable of meeting the requirements of the trier of fact. In this work, we review our previous work---a function oriented testing framework for validation and verification of computer forensic tools. This framework consists of three parts: function mapping, requirements specification and reference set development. Through function mapping, we give a scientific and systemized description of the fundamentals of computer forensic discipline, i.e. what functions are needed in the computer forensic investigation process. We focus this paper on the functions of media preparation, write protection and verification. Specifically, we complete the function mapping of these functions and specify their requirements. Based on this work, future work can be conducted to develop corresponding reference sets to test any tools that possess these functions.

Keywords: Computer forensics, validation, media preparation, write protection, verification

1. INTRODUCTION

Defined by Rodney (Rodney 1999), computer forensics is the process of identifying, preserving, analysing and presenting digital evidence in a manner that is legally acceptable. In this work, we use the terms Electronic Evidence (EE), computer forensics, digital forensics and forensic computing to refer to this discipline.

There is a critical need in the law enforcement community to ensure the reliability of computer forensic tools, which means forensic software tools consistently produce accurate and objective results. Hence, a demand of validating and verifying these tools has been raised recently (Jason 2007). Generally, the

validation and verification (VV) of softwares often refers to methods and technologies that provide confidence in system softwares. Since introduced in the early 1990s, the concept of validation and verification has been interpreted in a number of contexts by different organizations and communities, such as IEEE standard 1012-1998, ISO 17025 and the Scientific Working Group on Digital Evidence (SWGDE). Taking into consideration all these definitions and keeping in mind the requirements of ISO 17025 (e.g. *validation is the confirmation by examination and the provision of objective evidence that the particular requirements for a specific intended use are fulfilled*), we adopt the definitions of validation and verification of forensic tools proposed by Jason (Jason 2007).

- Validation is the confirmation by examination and the provision of objective evidence that a tool, technique or procedure functions correctly and as intended.
- Verification is the confirmation of a validation with a laboratories tools, techniques and procedures.

Two approaches, i.e. software inspection and software testing are widely used in the field of software validation and verification. While the former takes place at all stages of software development life-cycle, inspecting requirements documents, design diagrams and program codes, the latter runs an implementation of the target software to check if the software is produced correctly or as intended. Since our proposed work is to validate existing EE software tools, it falls into the software testing category.

EE software tool validation and verification is still in its embryonic stage, and there is limited work in this filed, such as the National Institute of Standards and Technology (NIST) project "Computer Forensics Tool Testing" (CFTT) (NIST 2009) and Brian Carrier's work Digital Forensics Tool Testing (DFTT) Images (Brian 2009).

In our previous work (Jason 2007), we proposed a function orientated framework for EE tool validation and verification. The core principle of our framework is function driving, and this framework conceptually consists of three parts: function mapping, requirement specification and reference set development. In this framework, we identify fundamental functions required in EE investigations, such as search, data recovery, forensic copy and so on. For each function, we further identify its details, e.g. sub-categories, components and etc. We call this process function mapping. Based on the function mapping, we specify each function's requirements and then develop a reference set against which EE tools can be tested. Following this work, we focused on and addressed two functions, i.e. "search" and "data recovery" in (Guo 2009) and (Guo 2010) respectively. For each function, we accomplished its function mapping, requirements specification and reference set development. In this work, we continue our "puzzle game" and focus on the functions of "media preparation", "write protection" and "(forensic

copy) verification". The related background information of our VV framework and detailed review of existing work of EE software tool validation and verification can be found in (Guo 2009). By addressing EE functions one by one, we eventually can accomplish the entire function oriented validation and verification of EE tools in the end.

The rest of this paper is organized as follows. In section 2, we review our function orientated VV framework. Section 3, 4, 5 address the functions of "media preparation", "write protection" and "verification" respectively, in terms of function mapping, requirements specification and reference development. Section 6 concludes this paper.

2. FUNCTION ORIENTED VV FAMEWORK

In this section, we review our proposed validation and verification paradigm. Our methodology starts with a scientific and systemized description of the EE field through a model and the function mapping. Components and processes of the EE discipline are defined in this model and fundamental functions in EE investigation process are specified (mapped), i.e. search, data recovery, file identification and etc. Based on the comprehensive and clear understanding of EE discipline, we then actually perform the validation and verification of EE tools as follows. First, for each mapped function, we specify its requirements. Then, we develop a reference set in which each test case (or scenario) is designed corresponding to one function requirement. With the reference set, an EE tool or its functions can be validated and verified independently.

In this work, we use the CFSAP (computer forensic-secure, analyze, present) model (George 2003) to describe the basic procedures of EE investigation. In this model, four fundamental procedures are identified: Identification, preservation, analysis and presentation. In the context of validation and verification, identification and presentation are skill based concepts, while preservation and analysis are predominately process, function and tool driven concepts and are therefore subject to tool validation and verification. The processes of preservation and analysis are preliminarily dissected into several fundamental functions at the high-level. The functions in the data preservation procedure are forensic copy, verification, write protection and media preparation. The data analysis procedure involves eight functions: searching, file rendering, data recovery, decryption, file identification, processing, temporal data and process automation. An ontology of such function mapping is shown in Figure 1.

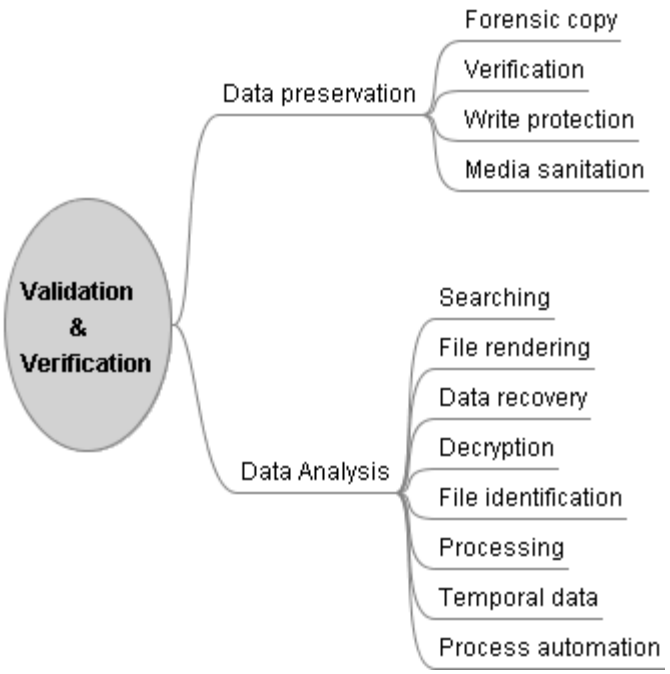

Fig. 1 A top-level ontology of computer forensic functions

Our function orientated VV methodology can be presented as the following. If the domain of computer forensic functions is known and the domain of expected results (i.e. requirements of each function) are known; that is, the range and specification of the results, then the process of validating any tool can be as simple as providing a set of references with known results. When a tool is tested, a set of metrics can also be derived to determine the fundamental scientific measurements of accuracy and precision. In summary, if the discipline can be mapped in terms of functions (and their specifications) and, for each function, the expected results are identified and mapped as a reference set, then any tool, regardless of its original design intention, can be validated against known elements.

3. MEDIA PREPARATION FUNCTION

Images, results of "forensic copy" process, must be accommodated in storage devices for future analysis. Three types of storage devices, i.e. magnetic disks (hard drives), optical disks (CD, DVD) and semiconductor devices (flash memory) are widely available on the market. These devices used by EE investigators for storing images could be either brand new or reused from one investigation to the next. In both cases, especially the latter one, an investigator needs to ensure the device is "clean", that is the device does not contain any data that could inadvertently become included in the current investigation. We, in this work, call the process of initiating storage devices forensically clean as

sanitisation. Although using the same name, we realize that the concept of sanitisation in the computer forensic context has slight differences from that in data security context. We will detail these differences in this section. After being sanitized, storage devices may need to be further processed (e.g. a hard disk may be partitioned and formatted). Hence, from the function point of view, we further dissect the function media preparation into three subcategories: *sanitisation*, *partitioning* and *formatting* as shown in Figure 2.

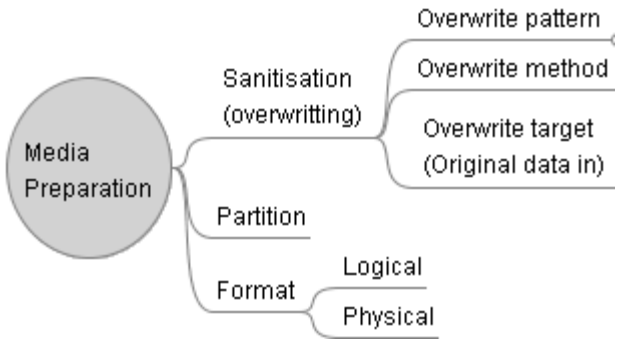

Fig. 2 Media preparation function mapping

3.1 Function mapping

Originally, the concept of sanitisation, or storage media sanitisation, stemmed from data security. In today's digital computing world, all digital data is maintained in storage devices, such as hard disk, CD, DVD and flash memory. When the owners retire their storage devices without proper data treatment, they will risk the unauthorized disclosure of confidential information contained in the disposed storage devices. To prevent it from happening, one method is used, that is storage media sanitisation. According to (Kissel 2006), sanitisation is referred to as "the general process of removing data from storage media, such that there is reasonable assurance, in proportion to the confidentiality of the data, that the data may not be retrieved and reconstructed". Four basic sanitization security levels and corresponding techniques are defined in (Hughes 2009): weak erase (deleting files), block erase (overwrite by external software), normal secure erase (current drives), and enhanced secure erase.

In the forensic computing context, sanitisation is generally referred to as the method to make storage device prepared for use/reuse in a forensically sound manner. Due to the difference of sanitisation in forensic computing and data security in terms of sanitisation purpose, not all techniques of data security sanitisation are applicable to the sanitisation in the forensic computing context. For example, deleting file and reformatting can be used as sanitisation techniques with certain security level in data security context. However, they are not forensically sound technique of sanitisation in the forensic computing context.

This is because deleting a file merely removes its name from the directory structure's special disk sectors. The user data remains in the drive data storage sectors and could be included to the next investigation. Reformatting a hard disk drive clears the file directory and severs the links between storage sectors, but the user data still remains. Therefore, overwriting, i.e. intentionally overlay the original data with arbitrary or random data, becomes the most secure technique for forensic computing sanitisation.

There are several factors investigators need to take into account when they perform the overwriting sanitisation, such as overwrite pattern, overwrite methods (Figure 3) and overwrite target (Figure 4). First, overwriting could be executed at either physical level or application level. By "physical level", we mean the overwriting is performed through using the built-in commands of a hard drive. A digital storage device may be attached to a host computer by one of several interfaces, such as ATA (AT Attachment), SATA (Serial ATA), SCSI (Small Computer System Interface), USB (Universal Serial Bus), and FireWire. For ATA and SATA hard drives, the SECURITY ERASE UNIT command overwrites a hard drive. This command instructs the drive's on-board controller to run a firmware routine that overwrites disk contents at the physical level, including any remapped bad sectors containing old data. The command is also supposed to move the drive head off track by 10% so that data between tracks is also overwritten. A similar command, ERASE, is defined for the SCSI interface. Besides these built-in facilities, there is a wide range of softwares and applications can implement the overwriting sanitisation as well. A survey of free and commercially available sanitisation tools can be found in (Garfinkel 2003, Roubos 2007).

Secondly and third, investigators need to select what to write (overwrite materials) to overlay original data (overwrite target). The selection of overwrite materials is the second difference of sanitisation in data security and forensic computing. In the data security context where the purpose of sanitisation is to overlay original data, the overwrite materials may take various forms of binary zeros, fixed data pattern or random data. However, the most overwrite material used for sanitisation in forensic computing is binary Zeros because it is required that there shall no possibility of inadvertent inclusion of unrelated data from a storage device into an investigation.

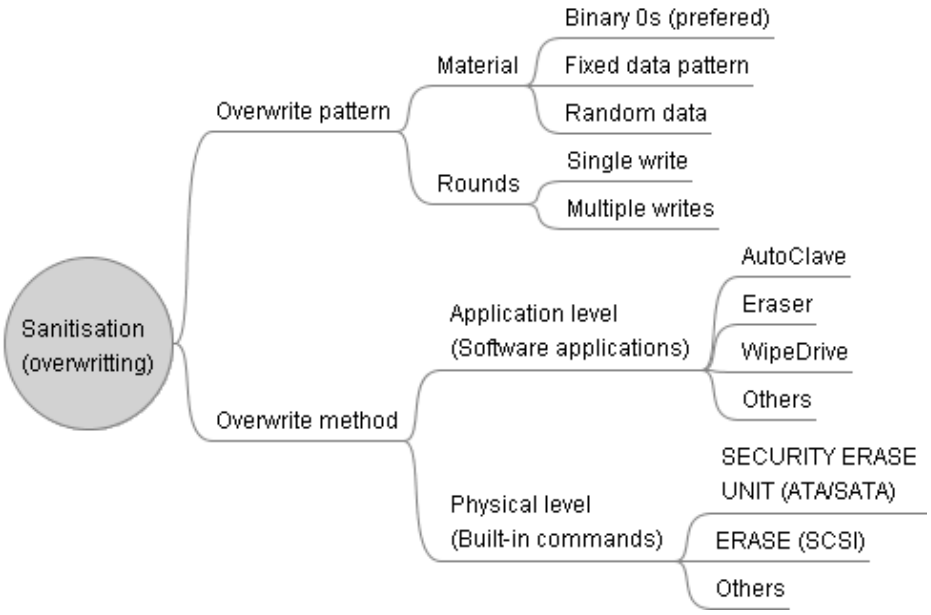

Fig. 3 Overwrite pattern and method mapping

At last, there is debate on the level of sanitisation that should take place, with the historical perspective that any wiping should consist of a DOD wipe (3 writes). The reason behind this has been justified by the release of a number of papers detailing recovery of remanent data. (Gutmann, 1996). Although it is a controversial endeavour, Wright claims a single write is substantial enough to remove data (Wright, 2008).

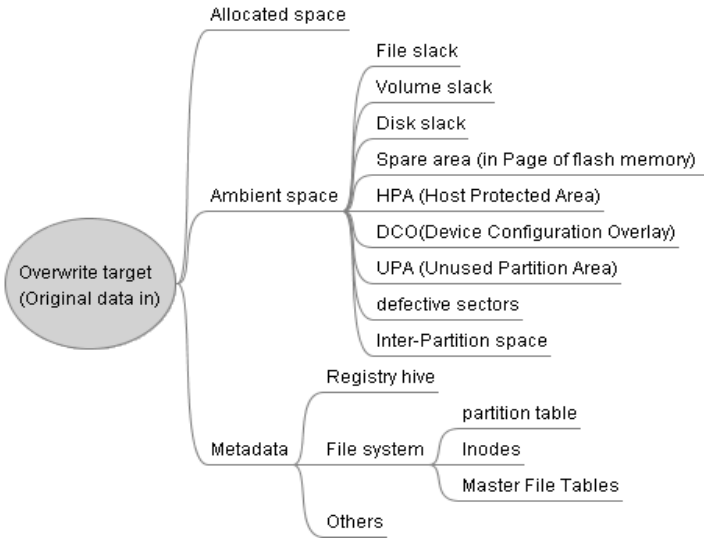

Fig. 4 Overwrite Target

Digital data contained in modern storage devices falls in one of the following categories: operating system, application programs, and user data stored in files. Drives also contain backing store for virtual memory, and operating system meta-information, such as directories, file attributes, and allocation tables. There is a range of places in storage devices where data could reside. For example, in hard disks, the *allocated space* is where data normally resides. Besides, the *ambient space* (i.e. unallocated space, or space orphaned from the operating or file system) may also contain remnants of previous files that were deleted but not completely overwritten, bytes at the end of partially filled directory blocks (sometimes called slack space), startup software that is not strictly part of the operating system (such as boot blocks), and virgin blocks that were initialized at the factory but never written. Hence, the ambient space may take the form of file slack, volume slack, HPA (Host Protected Area), DCO(Device Configuration Overlay) and so on. In order to obtain a completely "clean" storage device, all above space need to be overwritten.

Apart from sanitisation, partitioning and formatting may be needed to get storage devices prepared for accommodating digital evidence or images in some circumstances. The specification of partition and formatting, such as how many partitions in a hard disk, which file system is specified for each partition, is predominately depending on investigation.

3.2 Requirement Specification

Requirement specification is the second step of validating and verifying functions. Similar to our previous work, we specify requirements of the write preparation function in an extendable and custom-made way. From function mapping, we can see that there are a variety of diversifications we need to take into consideration when we specify the requirements. For example, the overwrite materials could be all binary zeros, fixed data pattern or random data. The overwrite target could be data in allocated space, ambient space or metadata. Hence, we use ***variables*** (in boldfaced and italic) to reflect these diversifications, and hence multifarious requirements can be refined to the follows statements. When one requirement needs change, people just need tailor (add, deleted, or modify) these variables.

1. The tool shall be able to overwrite ***overwrite target*** by ***overwrite means*** using ***overwrite pattern***.
2. The tool shall verify the success execution of overwriting
3. If the tool is not able to overwrite certain area in storage devices (e.g. defective sectors), the tool shall inform the user
4. If the tool support partition function, then the tool shall partition the storage device
5. If the tool support formatting function, then the tool shall format the storage device

This method of requirements specification is highly abstract and generalized. When it is needed for developing a specific test scenario in reference set, each of these requirements can be unwrapped. For example, the requirement "The tool shall be able to overwrite ***overwrite target*** by ***overwrite means*** using ***overwrite pattern***" can be unwrapped and instantiated as "The tool shall be able to overwrite data in file slack space by single writing 0s through ATA SECURITY ERASE UNIT command" or "The tool shall be able to overwrite data in allocated space by multiple writing random data through ATA SECURITY ERASE UNIT command " and etc.

4. WRITE PROTECTION FUNCTION

One of the forensic computing investigation rules identified by Rodney (Rodney 1999) is "*application of forensic computer processes during the examination of original data shall be kept to an absolute minimum*". In other words, during an investigation (e.g. acquisition or analysis), digital evidence stored in active system or a secondary storage device must be protected from being overwritten or altered. In modern computer systems, data is written to or read from a storage device via commands that are issued by the computer and transmitted from the computer's interface connection to the storage device's interface connection. Hence, the basic strategy for implementing a write protection is to place a filter between a host computer and a secondary storage device. This filter monitors I/O commands issued by the application. It blocks all commands that could directly or potentially cause alteration to the original data, and only allows commands to the device that make no changes to the device.

The filter sitting in the connection between a host computer and a storage device could be implemented either as hardware or software. Accordingly, two types of write protection techniques are developed, that are hardware write protection (blocking) and software write protection (blocking). Additionally, there is a third type of write protection that can be argued. It involves the practitioner adopting procedures and practices that reduce or eliminate un-intentional changes to data. In the following (as shown by Figure 5), we extrapolate these broad write protection functions further, detailing their constituent components.

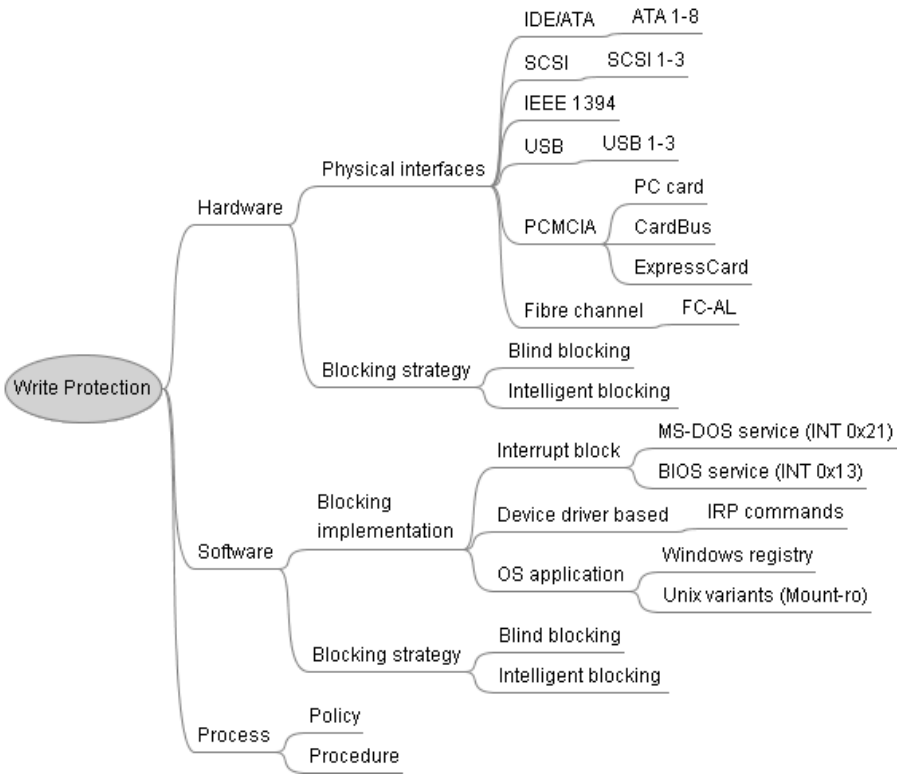

Fig. 5 Write Protection function mapping

4.1 Function mapping

4.1.1 Hardware write blocking

The hardware write blocking is implemented by a physical or mechanical device (known as hardware write blocker, HWB) that monitors the commands being issued and prevents the computer from writing data to the storage device. HWB is physically connected between the computer and a storage device, and hence breaks the bus used to attach a hard drive to a host computer into two segments. Once the blocking device is connected it can intercept a command from the host and select a desired course of action (e.g. allowing or blocking) for the command.

Various storage devices are attached to the host computer through certain physical interfaces. The common ones are the ATA and IDE (Integrated Drive Electronics) interfaces, including variants such as ATA-2, ATA-3 and EIDE (Enhanced IDE). Other physical interfaces include SATA, SCSI, IEEE 1394 and USB. The HWB intercepts all commands from the host to the storage device and only issues safe commands to the storage device.

4.1.2 Software write blocking

Compared to the HWB, the filter that blocks write commands could be also implemented in software way, that is software write blocker (SWB). Basically, three approaches are widely used for SWB: Interrupt based blocking, driver based blocking and OP applications.

Today's operating systems usually provide higher level access interfaces or services (compared to low level programming required for direct access drive through the interface controller) to execute drive related commands. For example, programs running in the DOS environment can use services: DOS service interface (Interrupt 0x21) or BIOS service interface (Interrupt 0x13). A SWB works by modifying interrupt table, which is used to locate the code for a given BIOS service (Brian 2005). The interrupt table has an entry for every service that the BIOS provides, and each entry contains the address where the service code can be found. For example, the entry for INT 13h will point to the code that will write or read data to or from the disk. A SWB modifies the interrupt table so that the table entry for interrupt 0x13 contains the address of the write blocker code instead of the BIOS code. When the operating system calls INT13h, the write blocker code is executed and examines which function is being requested.

A SWB can also be implemented as a segment code inserted into the device driver stack in operating systems that manages all access to a device. This code examines all the commands sent to a device through the stack. Any command that could directly or potentially cause modification to a protected drive is blocked, i.e., it is not passed on to lower layers of the stack. The third SWB approach utilises operating system facilitates to achieve write blocking. For example, in Windows XP, people can modify the registry HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StorageDevicePolicies to prevent or allow the writing command. In Unix, command (Mount-ro) is used for the same purpose.

4.1.3 Procedure based write blocking

Apart form HWB and SWB, the third write protection is based on abidance by certain procedures or rules. It involves the practitioner adopting procedures and practices that reduce or eliminate un-intentional changes to data. An example would include a practitioner using a DOS boot disk to access data on a hard disk, called controlled booting. The disk would boot a known operating system that the user would use tools known not to alter data.

4.2 Requirement specification

The CFTT project in NIST has covered the requirement specification of write protection (block) function very well (NIST 2009). They have already done extensive and comprehensive work on function requirement specification of hardware write block and software write block. Hence, we adopt their work and use them as requirement specification for our work. However, one concern could

be raised. That is, their tests may not be very repeatable for the average laboratory because it required sourcing a lot of devices, which is unwieldy and costly for smaller laboratories or even larger laboratories.

5. VERIFICATION FUNCTION

5.1 Function mapping

In the preservation phase of EE investigation, after forensic copy (Guo 2009) is completed, both the original and the copy of the original must be authenticated. The meaning of image authentication in computer forensic context is two-fold. First, it must be demonstrable that the copy is an exact, bona fide, copy of the original. This is raised by the requirement that any conclusions drawn from analysis of the copy are valid. The second meaning of image authentication is that there must be assurance of the continued integrity of the original. The image authentication is often referred to as (forensic copy) verification.

Forensic copy verification can be implemented on a number of different levels (Figure 6), and each has its own degree of reliability in their application. The simplest and lowest reliable verification is the visual inspection. In certain circumstances, investigators may verify image's integrity by visually inspecting and comparing the original data and its copy.

Another verification method is the *checksum*. It checks for errors in digital data. Typically a 16- or 32-bit polynomial is applied to each byte of digital data. The result is a small integer value that is 16 or 32 bits in length and represents the concatenation of the data. At any point in the future the same polynomial can be applied to the data and then compared with the original result. If the results match some level of integrity exists.

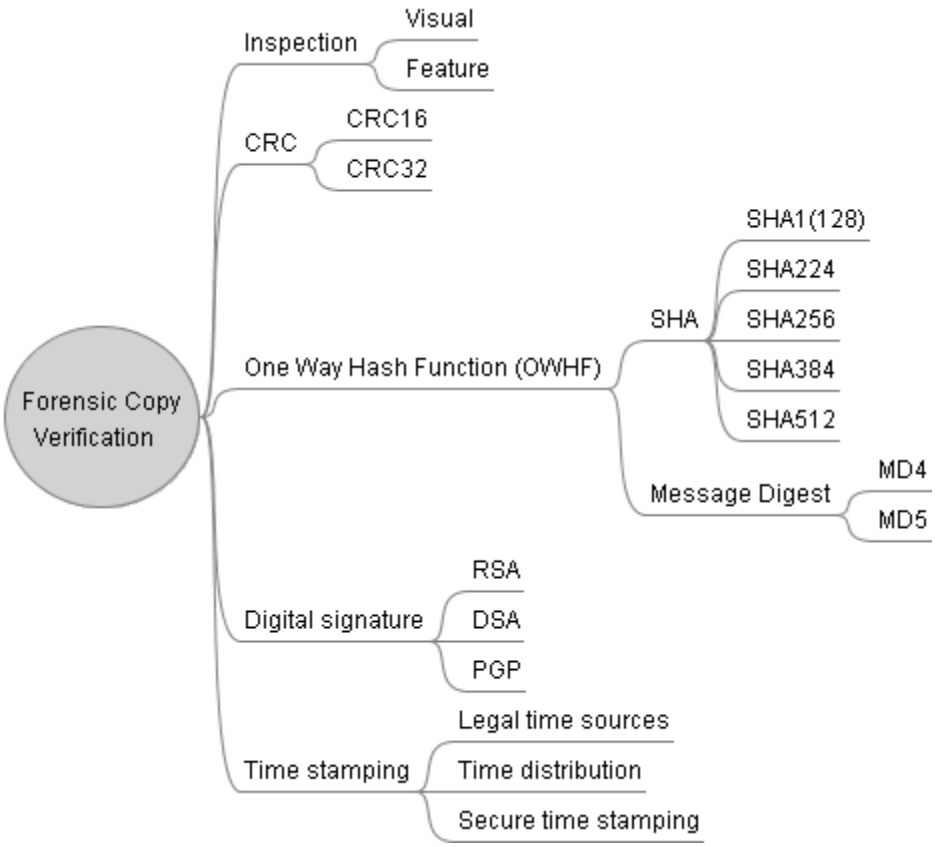

Fig. 6 Forensic copy verification function mapping

The most commonly used verification method is the cryptographic one way hash function (OWHF). OWHF make use of block cipher algorithms to calculate a digest (typically a value represented by a 128-bit string or longer) of a file or disk. This value is sensitive to the change of even a single bit in the original dada. After or during a forensic copy, acquisition tools will calculate a hash value for later reference. When verification is needed, investigators calculate the hash value again, and compare the two hash values. A mismatch indicates the integrity breach. MD5 and SHA-1 are widely used in OWHF verification method.

The digital signature method can provide the highest degree of reliability. It binds the identity of the signer with digital data integrity methods (e.g. one-way hash values). These methods use a public key crypto-system where the signer uses a secret key to generate a digital signature. Anyone can then validate the signature generated by using the published public key certificate of the signer. DSA and RSA are typical examples of this method (Charlie 2002).

Another component of verification function is identified by reviewing what the digital integrity is. According to Alfred (Alfred 2001), digital integrity can be

defined as "the property whereby digital data has not been altered in an unauthorized manner since the *time* it was created, transmitted, or stored by an authorized source". While above verification methods address questions of "who" (the signer) and "what" (the digital data), they don't provide the answer to the question of "when". Specifically, when did the signing of the digital evidence occur? How long after the evidence was seized, was its integrity protected? How long can we prove the integrity of the digital evidence that we signed? A secure and auditable time stamping mechanism or function is a solution to these questions (Hosmer 2002). When the forensic copy is done, a time stamp that is resistant to manipulation and provides an authenticated audit trail is created. It then can be electronically "bind" to digital evidence so that they can be verified by a third party. A number of issues need to be considered in time stamping, such as traceability to legal time sources, time distribution, secure digital time stamping and etc.

5.2 Requirements specification

1. The tool shall accurately perform hash functions and calculate hash values of verification objects.
2. The tool shall support multiple hash functions.
3. The tool shall support or provide time stamping.
4. The tool shall not mutate the verification objects.
5. The tool shall verify the correctness of hash values

6. CONCLUSION

In this work, we focus on and address the validation and verification of functions, media preparation, write protection and forensic copy verification. Specifically, we complete their function mapping and specify their requirements. Based on this work, future work can be conducted to develop corresponding reference sets to test any tools that possess these functions.

To complete the entire validation paradigm, more work need to be carried out in the future. First, although the proposed methodology holds promise, we realize that it needs to be tested at least using one tool in order to evaluate the methodology and work out any potential weakness or shortcomings. Hence, some tests will be implemented against some real tools, such as EnCase and FTK. Secondly, a quantitative model is required to evaluate the results of validation and verification. For example, specific metrics are needed to measure the accuracy and precision of testing results. Then, we need to design judgement rules of validity of EE tools. How to judge if a tool is validated or not? Is a tool validated only when it passes all the test cases, or is a tool validated in certain scenarios where it pass these test cases?

REFERENCES

Alfred J. M., Paul C. O., and Vanstone S. A., (2001) "Handbook of Applied Cryptography", Fifth printing, CRC Press.

Jason B., and Jill S., (2007) "Digital Forensics: Validation and Verification in a Dynamic Work Environment", Proceedings of the 40th Annual Hawaii International Conference on System Sciences, Hawaii.

Brian C., (2005) "File System Forensic Analysis", Addison-Wesley Upper Saddle River, New York.

Brian C., (2009) "Digital Forensics Tool Testing Images", http://dftt.sourceforge.net/, Sep. 1 2009.

Charlie K., Radia P., and Speciner M., (2002) "Network Security: Private Communication in a Public World", Second Edition. in computer networking and distributed systems, Prentice Hall PTR.

Garfinkel S.L., and Shelat, A. (2003), "Remembrance of data passed: a study of disk sanitization practices", IEEE Security and Privacy, Vol.1 (Issue 1): Page 17-27.

George M., Alison A., Byron C., Olivier D. V., and Rodney M., (2003) "Computer and intrusion forensics", Artech House, Boston

Yinghua G., Jill S., and Jason B., (2009), "Validation and verification of computer forensic software tools--Searching Function", Digital Investigation, Vol. 6: PageS12-S22.

Yinghua G., and Jill S., (2010) "Data Recovery Function Testing for Computer Forensics Investigation Tools", Advances in Digital Forensics VI (Springer, 2010).

Gutmann P., (1996), "Secure Deletion of Data from Magnetic and Solid-State Memory" http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html, June 2007

Hosmer C., (2002) "Proving the Integrity of Digital Evidence with Time," International Journal of Digital Evidence Vol.1 (1).

Hughes G. F., Coughlin, T., and Commins D. M., (2009), "Disposal of Disk and Tape Data by Secure Sanitization," IEEE Security and Privacy, Vol.7 (Issue 4): Page 29-34.

Kissel R., Scholl M., Skolochenko S., and Li X., (2006), "Guidelines for media sanitization", NIST SP 800-88.

NIST, (2009) "Computer Forensics Tool Testing (CFTT)", www.cftt.nist.gov, Oct. 11 2009.

Rodney M., (1999) "What is Forensic Computing?", Australian Institute of Criminology, Trends and Issues Technical Report.

Wright C., Kleiman D., and Shyaam S. R. S., (2008). "Overwriting Hard Drive Data: The Great Wiping Controversy", Lecture Notes in Computer Science (Springer Berlin / Heidelberg).

Roubos D., Palmieri L., Kachur R. L., Herath S., Herath A., and Constantino D., (2007). "A study of information privacy and data sanitization problems: student paper", Journal of Comput. Small Coll. 22, 4 (Apr. 2007), 212-219.