available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

# Validation and verification of computer forensic software tools—Searching Function

*Yinghua Guo*, Jill Slay, Jason Beckett*

*Defence and Systems Institute, University of South Australia, Building F2-22, Mawson Lakes Campus, Mawson Lakes, Adelaide, SA, 5095 Australia*

*Keywords:*
Electronic evidence
Computer forensics
Validation
Verification
Searching

ABSTRACT

The process of using automated software has served law enforcement and the courts very well, and experienced detectives and investigators have been able to use their well-developed policing skills, in conjunction with the automated software, so as to provide sound evidence. However, 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 present a scientific and systemical description of the computer forensic discipline through mapping fundamental functions required in the computer forensic investigation process. Based on the function mapping, we propose a more detailed functionality orientated validation and verification framework of computer forensic tools. We focus this paper on the searching function. We specify the requirements and develop a corresponding reference set to test any tools that possess the searching function.


## 1. Introduction

Originating in the late 1980s as an *ad hoc* practice to meet the service demand from the law enforcement community, computer forensics has recently developed into a multi-domain discipline crossing the corporate, academic and law enforcement fields. While the definitions of computer forensics and its interacting elements vary and depend on the authors and their backgrounds (Lin, 2008), the core connotation of computer forensics can be concisely described as the process of identifying, preserving, analyzing and presenting digital evidence in a manner that is legally acceptable (McKemmish, 1999). In this work, we use the term Electronic Evidence (EE) which has been in common use by law enforcement and agencies in Australia and which subsumes, and includes, terms such as computer forensics, digital forensics and forensic computing.

Over the last decade, the world has experienced an explosive growth in IT technology and electronic crime. On one hand, the technology field has become very dynamic and the number of types of digital devices with processing and storage capacity in common usage, such as notebook computers, iPods, cameras and mobile phones, has grown incredibly rapidly. On the other hand, unfortunately, these continual advances in IT technology pose complex challenges to the electronic evidence discipline.

One of those challenges faced by EE practitioners is how to assure the reliability (or forensical soundness) of digital evidence acquired by EE investigation tools (NRC, 2009). As today's EE investigations heavily rely on automated software tools, the reliability of investigation outcomes is predominantly determined by the validity and correctness of such tools and their application process. Therefore, an insistent demand has been raised by law enforcement and other agencies to validate and verify EE tools to assure the reliability of digital evidence.

Another factor demanding the validation and verification of EE tools is the request to bring the EE discipline inline with

* *Corresponding author*.
E-mail address: Yinghua.Guo@unisa.edu.au (Y. Guo).
1742-2876/$ – see front matter 
doi:10.1016/j.diin.2009.06.015

other established forensic disciplines (e.g. DNA and ballistics) (Beckett and Slay, 2007). To achieve this goal, one main way is to gain external accreditation, such as ISO 17025 laboratory accreditation (ISO 17025E). EE laboratories and agencies are tested against developed criteria and have to satisfy the extensive requirements outlined within this document to gain accreditation. As a part of the accreditation, the EE tools and their utilization process need to be tested.

In this work, we propose a functionality orientated paradigm for EE tool validation and verification based on Beckett's work (Beckett and Slay, 2007). Within this paradigm, we dissect the EE discipline into several distinct functional categories, such as searching, data recovery and so on. For each functional category, we further identify its details, e.g. subcategories, components and etc. We call this dissection process *function mapping*. Our focus in this work is the *searching* function. We map the searching function, specify its requirements, and design a reference set for testing EE tools that possess searching functions.

The rest of this paper is organized as follows. Section 2 explains the necessity for EE tool validation and verification. It also reviews the related work of traditional EE tools testing in the EE discipline. Section 3 discusses the previous work and identifies their limitations. In Section 4, we present our functionality orientated testing paradigm in detail, which includes its fundamental methodology and unique features. Section 5 presents detailed searching function mapping. The requirements of searching function are identified in Section 6. Lastly, we develop a focused pilot reference set for testing the searching function in Section 7. This paper is finally concluded by Section 8.

## 2. Background and related work

### 2.1. Validation and verification of softwares

The methods and technologies that provide confidence in system software are commonly called software validation and verification (VV). There are two approaches to software VV: software inspection and software testing (Fisher, 2007). While software inspection takes place at all states of the software development life-cycle, inspecting requirements documents, design diagrams and program codes, software testing runs an implementation of the target software to check if the software is produced correctly or as intended. The VV work proposed in this paper falls into the software testing category.

Since introduced in early 1990s, the concept of validation and verification has been interpreted in a number of contexts. The followings are some examples.

1) In IEEE standard 1012–1998, validation is described as the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies requirements. Verification is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
2) ISO (17025E) describes validation as the confirmation by examination and the provision of objective evidence that the particular requirements for a specific intended use are fulfilled.
3) Boehm (1997), from the software engineering point of view, succinctly defines validation and verification as ''validation: Are we building the right product?'' and ''verification: Are we building the product right?''
4) The only available description of software validation in the EE discipline is given by the Scientific Working Group on Digital Evidence (SWGDE, 2004) as an evaluation to determine if a tool, technique or procedure functions correctly and as intended.

Taking into consideration all these definitions and keeping in mind the requirements of ISO 17025, we adopt the definitions of validation and verification of forensic tools (Beckett and Slay, 2007) as follows.

- **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 laboratories tools, techniques and procedures.

### 2.2. Demands of EE tools validation and verification

The process of using automated software has served law enforcement and the courts very well, and experienced detectives and investigators have been able to use their well-developed policing skills, in conjunction with the automated software, so as to provide sound evidence. However, the growth in the 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'. Another factor demanding EE tools validation and verification is for the EE discipline to move inline with other established forensic disciplines.

#### 2.2.1. Trustworthiness of digital evidence

The validity and credibility (i.e. the ''trustworthiness'') of electronic evidence are of paramount importance given the forensic (for court) context of the discipline. Nowadays, the collection, preservation and analysis of electronic evidence in the EE process mainly rely on EE tools (hardware or software). If the EE tools or their application procedures are incorrect or not as intended, their results, i.e. digital evidence, will be questioned or may be inadmissible for court. In other words, the trustworthiness of digital evidence relies on the scientific application of the process, the analysis and the correct utilization and functioning of computer forensic tools.

However, the EE community is now facing a complex and dynamic environment with regard to EE tools. On one hand, the technology field has become very dynamic and the types of digital devices, such as notebook computers, iPods, cameras and mobile phones, have changed incredibly rapidly. And thus the digital evidence acquired from those devices has also changed. On the other hand, in such a dynamic technological environment, there is no individual tool that is able to meet all the needs of a particular investigation (Bogen and

Dampier, 2005). Therefore, the world has been witnessing an explosive boom in EE tools in the last decade. Although these EE tools are currently being used by law enforcement agencies and EE investigators, we must be aware that while some of them (e.g. EnCase, FTK) were originally developed for the forensic purpose, others were designed to meet the needs of particular interest groups (e.g. JkDefrag (Kessels) is a disk defragmenter and optimizer for Windows 2000/2003/XP/Vista/2008/X64). Hence, to guarantee that the digital evidence is forensically sound, EE investigators must validate and verify the EE tools that they are using to collect, preserve and analyze digital evidences.

#### 2.2.2. *Laboratory accreditation*

The establishment of digital forensic laboratories within Australia has predominantly been aligned with law enforcement agencies. While these laboratories or teams have worked successfully since their establishment, the discipline is now developing to a stage where the procedures, tools and people must be gauged against a quality and competency framework.

To achieve this goal, one main method is to comply with ISO 17025 Laboratory Accreditation standard. The ISO 17025 intends to specify the general requirements for the competence to carry out test and/or calibrations. It encompasses testing and calibration performed by the laboratory using standard methods, non-standard methods, and laboratory-developed methods. A laboratory complying with this standard will also meet the quality management system requirements of ISO 9001. Among these requirements (e.g. document control, internal audits and etc.), one content relating to the subject of this paper is "*test and calibration methods and method validation*". Due to the lack of verification and validation of EE tools, the EE branch may not be accredited by ISO 9001, like other law enforcement branches (e.g. DNA and ballistics) have already done. As a result, for the EE branch to avoid becoming the shortest wooden board (branch) of a bucket (law enforcement departments) calls for the verification and validation of EE tools.

### 2.3. *Existing works of EE tools validation and verification*

In the last a few years, although extensive research efforts have been conducted in the EE disciple ranging from generic frameworks and models (Reith et al., 2002; Bogen and Dampier, 2005; Brian, 2006) to practical guidelines and procedures (Beebe and Clark, 2005; Ruibin et al., 2005; Good practice guide), there is still very little on the validation and verification of digital evidence and EE tools.

Some efforts (Palmer, 2001; Bor-Wen Hsu and Laih, 2005) in the past have been made to investigate "*trustworthiness*" of digital evidence, that is the product of the process. In these works, the digital evidence (outcomes of the tools) are being examined rather than validating and verifying the tools themselves. The question can be asked as to why we should not validate the forensic development of such tools also.

The National Institute of Standards and Technology (NIST) is the one of the pioneers pursuing the validation and verification of computer forensic tools. Within NIST, the Computer Forensics Tool Testing (CFTT) project (NIST, 2002) was established to test the EE tools. The activities conducted in forensic investigations are separated into discrete functions or categories, such as write protection, disk imaging, string searching, etc. A test methodology is then developed for each category. So far, several functionalities and tools have been tested and documented, such as write blockers (NIST, 2003), disk imaging (NIST, 2004, 2005), string search (NIST, 2009) and mobile devices associated tools (NIST, 2008).

Developing extensive and exhaustive tests for digital investigation tools is a lengthy and complex process, which the CFTT project at NIST has taken on. To fill the gap between extensive tests from NIST and no public tests, Carrier (Brian, 2005) has been developing small test cases, called Digital Forensics Tool Testing Images (DFTT). These tests include keyword searching, data carving, extended partition and windows memory analysis.

Another research entity that is interested in the validation and verification of EE tools is the Scientific Working Group on Digital Evidence (SWGDE). Rather than developing specific test cases, the SWGDE recommended general guidelines for validation testing of EE tools (SWGDE, 2004). These guidelines include purpose and scope of testing, requirements to be tested, methodology, test scenario selection, test data and documenting test data used.

The validation and verification of EE tool can also been conducted by the vendors that produce these tools. For example, the Encase and FTK are two widely used digital forensic investigation tools in the world. Their developers, Guidance Software and Access Data have conducted some validation and verification work on Encase and FTK. These works can be found on their bulletin boards.

## 3. Discussion of existing works

### 3.1. *Various EE tools VV approaches*

As pointed out in the last section, a range of excellent work has been conducted on the validation and verification of EE tools, such as NIST/CFTT, DFTT and the VV work of EE tool vendors. In terms of the methods or methodologies that they utilize, these testing work can be categorized into two classes: tool orientated VV and functionality orientated VV.

#### 3.1.1. *Tool orientated VV approach*

The validation and verification work of EE tools conducted by the vendors (e.g. Encase from Guidance Software and FTK from Access data) falls into this category. Traditionally, in the digital forensic domain, the EE software tool, as an unseparated entity, is treated as the target of validation and verification. Usually, axiomatic proofs and/or reproducible experiments (testing) are required to perform the VV. To validate the target, the test cases need to be defined, the tests need to be run and the measured results need to be verified.

There are a few of points that need to be noted. First, vendor validation has been widely undocumented, and not proven publicly, except through rhetoric and hearsay on bulletin boards. Many published documents in this field discuss repeatability of process with other tools as the main

validation technique, but no documented record can be found in the discipline that expands on the notion of two tools being wrong (Beckett and Slay, 2007).

Secondly, this validation work treats the EE software package as a single unseparated entity. Tool orientated validation methods would usually invalidate a tool package when one of all the functions fails the validation even though all other functions pass the test. In most cases a forensic tool package is quite complex and provides hundreds of specific functions (Wilsdon and Slay, 2005), of which only a few may ever be used by an examiner. Therefore, according to the traditional tool orientated validation method, a digital forensic software suite, where most functions are valid and can be partially utilized, will not be utilized at all, or else must wait for the complete validation of the entire function set. Because the cost of purchasing such software is so great it would be infeasible to discount an entire package due to a single or small group of functions failing validation.

Following the second point is the cost and complexity issue of the tool orientated VV approach. Currently, to keep abreast of the broad range and rapid evolution of technology, many EE tools (and their updated versions) are constantly emerging. These tools either are designed solely for forensic purposes or are designed to meet the needs of particular interest groups. In such complex and diverse environments of EE tools, even trivial testing of all functions of a forensic tool for every version under all conditions, conservative estimates would indicate significant cost (Beckett and Slay, 2007).

#### 3.1.2. *Functionality orientated VV approach*

NIST/CFTT and DFTT perform the validation and verification of EE tools from another angle: functionality driven. Instead of targeting the EE software tool, they start the validation by looking at the EE discipline itself. They identify various activities required in forensic investigation procedures and separate them into functionalities or categories, such as write protection, disk imaging, string searching, etc. Then, they specify requirements that need to be fulfilled for each function category. Based on the requirements specification, testing cases are then designed to test functions of candidate EE tools.

The difference between the functionality orientated VV approach and the tool orientated VV approach is that the former does not treat a EE tool as a single entity. Instead, they parse an EE tool (or package) into various functions and test each function against the requirements specified by practitioners and expert advisory groups. For example, in the case of disk imaging testing (NIST, 2005), the EnCase LinEn 6.01 is selected as a test target and only its imaging function is tested. Clearly, the functionality orientated VV approach outperforms the tool orientated VV approach in terms of the effectiveness and cost.

### 3.2. *Open issues in previous work*

Despite the considerable achievements of previous EE work (including validation and verification of digital evidence and investigation tools), we discover two potential issues remaining unsolved, which motivate our proposed work.

The first open issue is that operational focus in the digital forensics domain to date has been to solve each problem as it presents and not to look at the process of analysis as a whole. For example, when dealing with the issue of analyzing an image obtained from one new device (e.g. new iPod), researchers and practitioners may design an investigation tool specifically working with this new device, rather than examining what impact will be on the digital forensics as a scientific discipline.

Digital forensics is very much an emerging discipline and has developed in an ad-hoc fashion (Beckett and Slay, 2007) without much of the scientific rigour of other scientific disciplines, such as DNA, ballistics, and fingerprints. Although the scientific foundations of EE field and the functions which together make up the EE process exist, they have never been formally or systematically mapped and specified (scientific foundations), or stated and characterized (functions). Though there have been recent efforts to formalize a definitive theory of digital forensics and research dissertations that focus on the process model have started to appear (Brian, 2006), there is still no adequate description of any depth of the specific functions of the discipline.

The second open issue regarding the validation and verification of EE tools is that methodologies proposed by NIST/CFTT and DFTT are broad and offer no conclusive detailed identification of what needs to be tested. In other words, there is still a lack of systematical and definitive description of the EE field as a scientistic discipline. For example, what basic procedures are in the EE investigation? What fundamental functionalities are needed in the EE investigation? What are the requirements of each functionality?

## 4. A new functionality orientated VV paradigm

### 4.1. *Proposed VV methodology*

Our methodology starts with a scientific and systematical 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. searching, data preservation, file identification, 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, a EE tool or its functions can be validated and verified independently.

In this work, we use the CFSAP (computer forensic-secure, analyze, present) model (Mohay et al., 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. In Beckett's previous work (Beckett and Slay, 2007), the processes of preservation and analysis are preliminarily dissected into

several fundamental functions at an abstract level. The functions in the data preservation procedure are forensic copy, verification, write protection and media sanitation. 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 Fig. 1.

In this work, we aim to complete the mapping of the functional categories of the field to a level of abstraction that would serve the purposes of a specification for a software developer, a technical trainer or educator, or for tool validation or verification. Specifically, we detail the specification of functional categories (e.g. searching, data preservation, file rendering and etc) and its sub-categories. For example, the *searching* function category can be further divided into three sub-categories, i.e. *searching target*, *searching mode* and *searching domain* which have a number of parameters need to be specified. We focus this work on the searching function, i.e. mapping the searching function, specifying the requirements of searching function and developing the reference set to validate and verify EE tools that possess the searching function.

For each mapped function, we specify its requirements. The following are some examples of searching function requirements.

- The tool shall find a keyword in the file slack.
- The tool shall find a keyword in a deleted file.
- The tool shall find a regular expression in a compressed file.

Based on the requirement specification, we then develop a reference set in which each test case (or scenario) is designed corresponding to one function requirement. With the reference set, a EE tool or its functions can be validated and verified independently.

Our proposed 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.

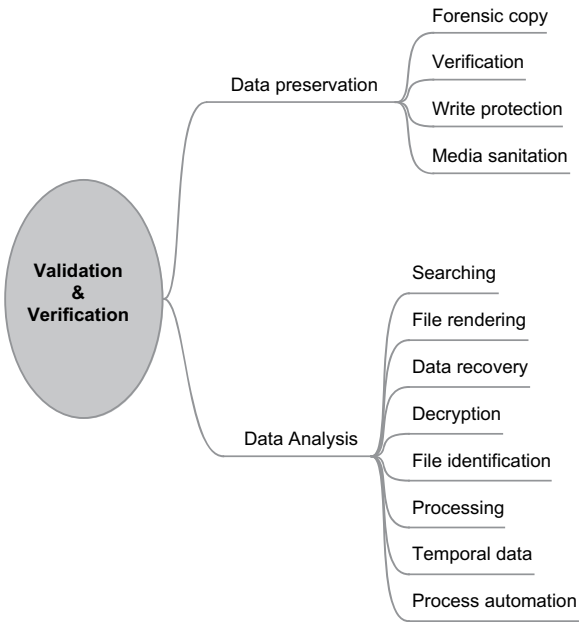


**Fig. 1 – Validation and verification top level mapping.**

### 4.2. *Features and benefits*

Our functionality orientated validation is performed on individual functionality of the software package rather than treating the package as a single entity. Validating each of the specific functions allows for a complex collection of tools to be partially utilized for active investigations rather than waiting for the complete validation of the complete set. Because of the function mapping, requirements specification and reference sets (which are described in detail in Section 5, Section 6 and Section 7), our validation approach has the following features.

- Detachability: a EE tool suite may possess a number of functions. Each function can be individually validated by our testing approach and the tool suite can be partially utilized rather than waiting for the complete validation of the complete function set.
- Extensibility: With a defined function, there exists a set of specifications for components that must be satisfied for the result of a function to be valid. That means as new specifications are found they can be added to a schema that defines the specification.
- Tool (tool version) Neutrality: If the results, or range of expected results, are known for a particular function, then it does not matter what tool is applied, but that results it returns for a known reference set can be measured. As a tool naturally develops over time, new versions are common, but the basic premise of this validation and verification paradigm means that the comments previously described for tool neutrality are also measurable.
- Transparency: A set of known references described as a schema are therefore auditable and independently testable.

Beside validating and verifying the EE tools, a standard test paradigm is also useful for proficiency testing (certification), training (competency) and development of procedures. According to a survey conducted by the National Institute of Justice (Appel and Pollitt, 2005), only 57% of agencies in US required specific training to duplicate, examine and analyze evidence and more than 70% of practitioners had no or minimal (less than a few hours) of training in this discipline. The situation in Australia is no different: new investigators, and the pool of seasoned detectives with advanced IT qualifications is drying up. Although some modern IT security certifications include aspects of forensic computing as part of incident response but there is no formal Australian certification, or established training standards for Electronic Evidence.

Through the development of these standard tests, the skills necessary to carry out a particular test may similarly be specified. For example, if we know that a piece of software must be able to search for a keyword from an image, then we can also specify that the investigator will only be certified as competent if he or she can use the software to analyze the image and find the same specified keyword.

## 5. Searching function mapping

For our validation and verification model to be developed, the discipline needs to be described in sufficient detail so that the discrete functions can be applied to the model. In this section, we complete the searching function mapping by detailing its sub-categories and various factors that need to be considered.

Generally speaking in the computer forensic domain, the *searching* relates to finding and locating the information of interest in digital devices. Naturally, several questions would be asked when performing the searching: what do we search? how to search? And where to search? To answer these questions, we divide the searching function category into three sub-categories: *searching target*, *searching mode* and *searching domain* as shown in Fig. 2.

The ''searching target'' in Fig. 3 addresses the question of ''what do we search for''. The search target could be a keyword, a regular expression, or a particular file. When we search for a keyword, we know exactly what we intend to find, such as ''terrorist'', ''computer'' etc. In another situation where we know what we want to search but are not sure about the contents (e.g. searching credit card number, phone number, address, post code, etc) we need regular expression searching, also called pattern searching. In both cases, there are several factors we need to consider. They are case sensitivity, written order (i.e. left-to-right or right-to-left) and character coding (e.g. Unicode, ASCII, code page, etc). Lastly, the EE investigation may need to find some particular files according to their attributes. These attributes include file size, file location, file name and file type. This type of searching is also called file filtering.

The ''searching mode'' is about how we perform the searching or formatting the query (Fig. 4). In the regular searching, we intend to find a single keyword or multiple keywords (i.e. sentence). However, basic and regular searching sometime may not able to meet the requirements of complex searching. Hence, some advanced searching modes are needed, such as *Boolean searching*, *proximity searching*, *Soundex searching*.

Using Boolean logic that describes certain logical operations of search terms, we are able to broaden and/or narrow the search results. Commonly used logical operations are *AND*, *OR*, *NOT*, *XOR* and wildcards. Proximity searching looks for documents where two or more separately matching term (keyword) occurrences are within a specified distance, where distance is the number of intermediate words or characters. For example, a search could be used to find ''red brick house'', and match phrases such as ''red house of brick'' or ''house made of red brick''. By limiting the proximity, these phrases can be matched while avoiding documents where the words are scattered or spread across a page or in unrelated articles in an anthology. Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English. The goal is for names with the same pronunciation to be encoded to the same representation so that they can be matched despite minor differences in spelling. With Soundex, the ''sound'' of names (the phonetic sound to be exact) is coded. This is of great help since it avoids most problems of misspellings or alternate spellings. For example: Scherman, Schurman, Sherman and Shireman and Shurman are indexed together as NARA Soundex Code ''S655''.

In the end, the ''searching domain'' answers the question of ''where to search''. From Fig. 6, if we look at the higher level (i.e. accessible by operating system), there are two places where targets can be searched: normal file data and metadata. In a computer system, data is stored and managed in the form of files. A large number of file types exist in today's computer system, such as document files (Word, PDF, TEX…), video files (MPEG, Quick Time…) sound files (WAV ASF, MP3…), graphic files (JPEG, PNG, BMP, TIFF…) and Internet files (HTTP, Cookies, ICQ…). The target needs to be searched not only in these original files, but also in the compressed files, deleted files and etc.

Besides the normal data file, metadata is another important source to search targets. Metadata is ''data about data'' of any sort in any media. It gives information about data and help facilitate the understanding, characteristics, and management usage of data. Metadata is particularly important in legal environments where litigation can request this sensitive information (metadata) which can include many elements of private detrimental data. Metadata can be stored either internally (called embedded metadata) in the same file as the data, or externally in a separate file (out-of-band metadata). We classify metadata at three different levels: file system metadata, program metadata and file metadata. The file metadata, usually stored with the files, records the name of the person who created, edited the file, how many times the file has been printed, and even how many revisions have been made on the file. File system metadata is the data about files that is kept out-of-band: either in directory entries or in specialized structure. This type of metadata normally includes the name of the file, the type of file, the temporal information of file and etc.

What we discussed above is about searching for targets in allocated space (high level) in the storage media, i.e. files or

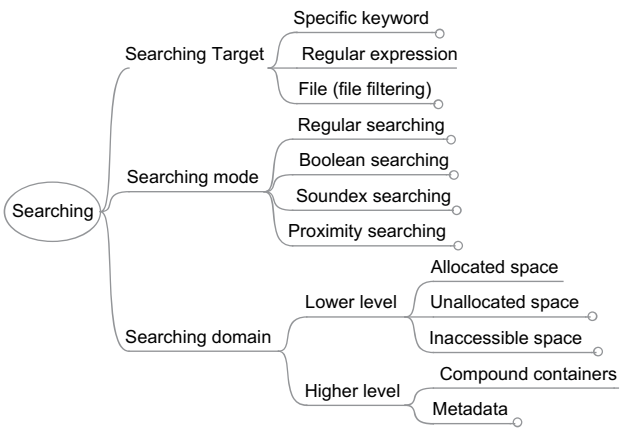


**Fig. 2 – An overview of searching function.**

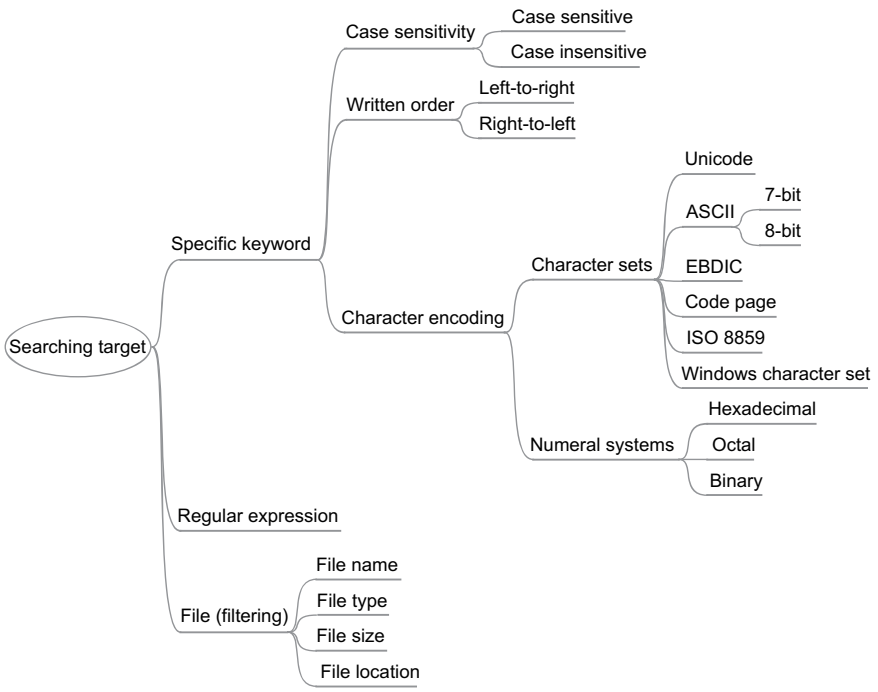


**Fig. 3 – The searching target mapping.**

metadata. Apart from the allocated space, there is a need to search for targets in unallocated space and inaccessible space (low level searching, Fig. 5) as well. By unallocated space, we mean disk space that can be accessed by the operating system but cannot be used by computer users. Some examples of unallocated space are file slack, volume slack and disk slack. The inaccessible space refers to the space in the disk that is reserved by the disk's vendor and cannot be seen by operating system, such as HPA (Host Protected Area), DCO (Device Configuration Overlay), UPA (Unused Partition Area) and etc.

## 6. Requirements specification of searching function

Through a comprehensive dissection, we have a systematically detailed understanding of the searching function. For our validation and verification model to work, requirements of the searching function need to be specified. Based on the specified requirements, the corresponding reference set (i.e. test cases) is developed (Section 7), against which EE tools and their individual functions can be tested to confirm their conformance to requirements.

We specify the requirements in an extendable and tailorable way. Within the searching function category, there are a variety of diversifications we need to take into consideration when we specify the requirements. For example, the searching target could be keywords, regular expression or files. The Soundex algorithms could be Daitch–Mokotoff Soundex, NYSIIS, or NARA soundex. The operating platforms (systems) could be Windows (98, 2000, XP or Vista), Mac OS, or Linux. The file systems could be FAT(12, 16,32), EXT(2,3), NTFS, HFS(+) or Raid.

Hence, we use *variables* (in boldfaced and italic) to reflect these diversifications, and multifarious requirements can be refined to the follows statements. When one requirement needs change, what people need to do is just tailor (add, deleted, or modify) these variables. The detailed description of these variables can be found in Figs. 2–6. The following is the requirements specification of searching function.

1) The tool shall operate in at least one **operation platform**
2) The tool shall execute searching in a **cylinder-aligned clone**, or in an **unaligned clone**, or in an **image** of a digital source, or provide the capability for the user to select and then operate searching
3) The tool shall execute searching in digital sources in each operation platform.
4) The tool shall execute searching in digital sources in each **file system**.
5) The tool shall execute searching by either full text searching, or index searching, or provide the capability for the user to select and then find the keyword by full text searching or index searching.
6) The tool shall find the **Target** in the **allocated space** of a **compound container**
7) The tool shall find the Target in the **unallocated space** of a compound container.
8) The tool shall find the Target in the **inaccessible space** of digital source.
9) The tool shall find the Target in the **metadata**.

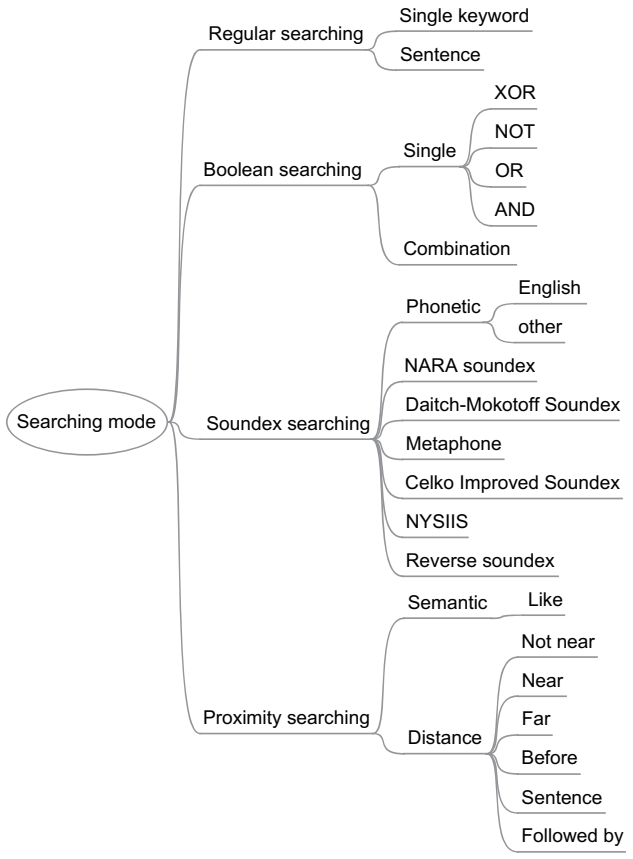


**Fig. 4 – The searching mode mapping.**

10) The tool shall find the Target by means of **Soundex** searching in the allocated space of a compound container.
11) The tool shall find the Target by means of Soundex searching in the unallocated space of a compound container.
12) The tool shall find multiple Targets by means of **Boolean searching** in the allocated space of a compound container.
13) The tool shall find multiple Targets by means of **Proximity searching** in the allocated space of a compound container.
14) The tool shall find multiple Targets by means of Boolean searching in metadata.
15) The tool shall find multiple Targets by means of Proximity searching in metadata.
16) The tool shall filter files by their **attributes**.

## 7. Development and use of reference set

The concept of reference sets is not new. In the field of EE tools validation and verification, a few of experimental sample sets have been developed. An example is the DFTT project (www.dftt.sourceforge.net). This project has produced only 13 basic tests at the time of writing. The NIST/CFTT project also conducted reference sets for limited functions, such as disk imaging and write blocker.

Essentially, a reference set consists of test scenarios (cases) against which a EE tool or its individual function is validated. The development of test scenarios is based on the specification of function requirements. With the requirements of searching function specified in Section 6, we are able to establish a reference set to test the searching function of various EE tools in this section. Since the functions requirements are specified in a extensible way in our work, the corresponding reference set is also extensible. This will enable practitioners, tool developers, and researchers to identify critical needs and target deterministic reference sets.

We identify 16 requirements for searching function. Since each requirement has several variables that lead to variations, we need to design multiple test scenarios for each requirement. Each scenario represents a requirement variation. Taking into account that there are a large number of test scenarios and the space limitation, we present some focused pilot samples for the illustration purpose in this section. A complete reference set can be found in (Guo et al., 2009).

For the purpose of reference set development, the following terms and definition apply.

- **Homogeneous cluster-crossing space**: starts at the allocated space of the cluster in a file and ends in the next consecutive cluster, which is allocated to the same file.
- **Heterogeneous cluster-crossing space**: starts at the allocated space of the cluster in a file and ends in the next consecutive cluster, which is allocated to the a different file.
- **Fragmented cluster space**: starts at the allocated space of the cluster in a file and ends in the next cluster of the file. The two clusters are fragmented.
- **Unallocated-crossing-unallocated space**: starts at the unallocated cluster and ends in the next consecutive cluster, which is also not allocated.

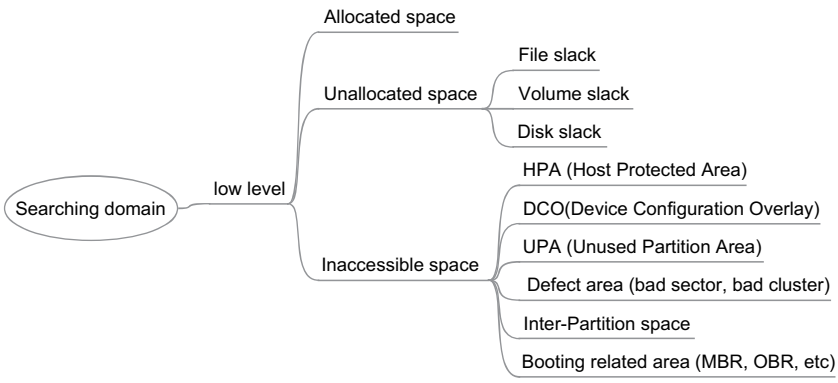


**Fig. 5 – The searching domain mapping (low level).**

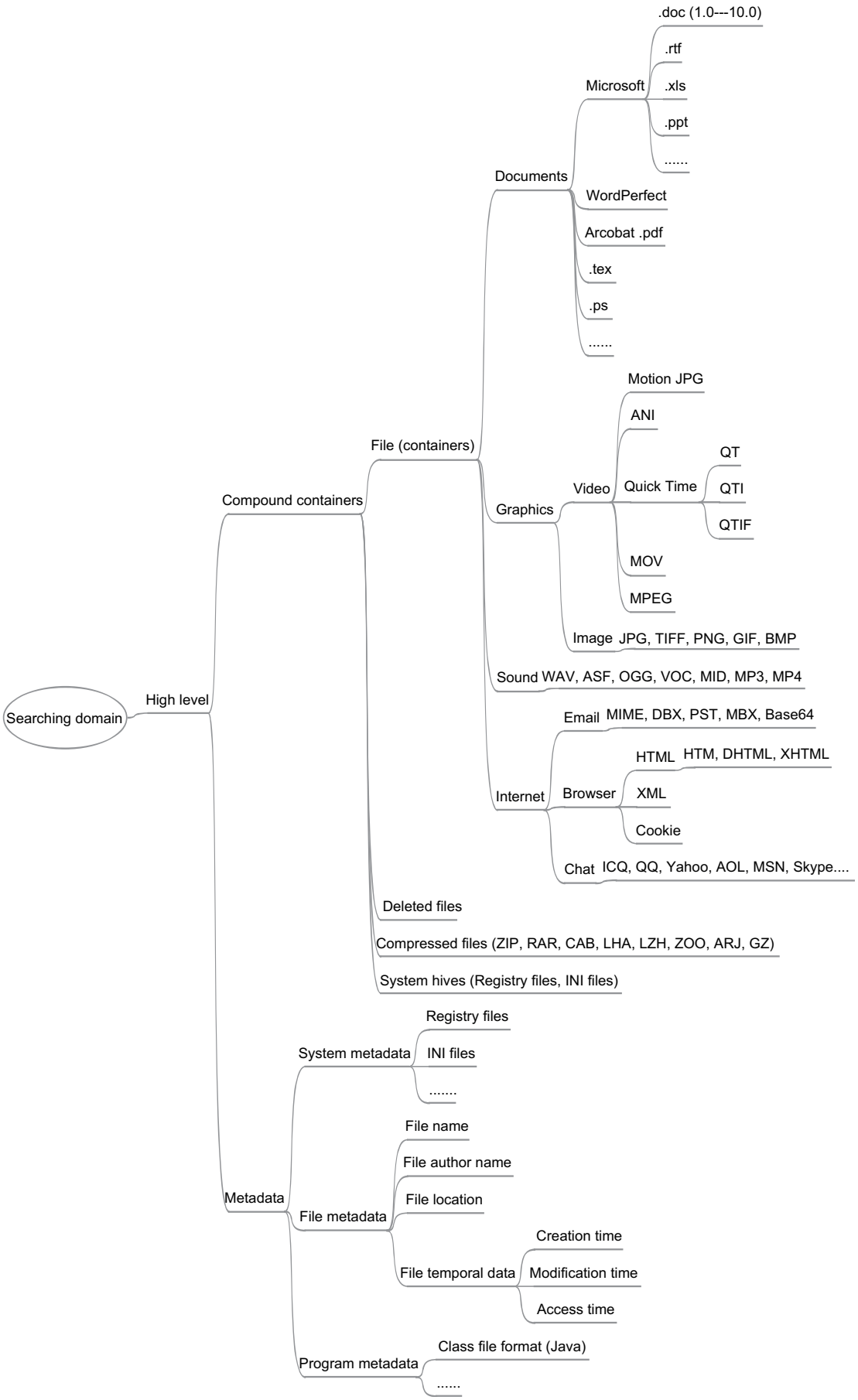


**Fig. 6 – The searching domain mapping (high level).**

- **Unallocated-crossing-allocated space**: starts at the unallocated cluster and ends in the next consecutive cluster, which is allocated a file.
- **File slack space**: starts and ends in the slack space of a file.
- **Allocated-crossing-slack space**: starts at the allocated space of one cluster in a file and ends in the slack space of the same cluster.
- **Slack-crossing-allocated space**: starts at the slack space of a cluster allocated to a file and ends in the allocated space of the next consecutive cluster allocated to a different file.
- **Slack-crossing-allocated space**: starts at the slack space of a cluster allocated to a file and ends in an unallocated cluster.

The following are some pilot samples of the reference set for the searching function.

1) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the allocated space of a cluster of a MS word file (Version 5.0).
2) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the homogeneous cluster-crossing space of a MS word file (Version 5.0).
3) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the heterogeneous cluster-crossing space of a MS word file (Version 5.0).
4) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the fragmented cluster space of a MS word file (Version 5.0).
5) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the unallocated space in one cluster.
6) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the unallocated-crossing-unallocated space.
7) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the unallocated-crossing-allocated space of a MS word file (Version 5.0).
8) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the file slack of a MS word file (Version 5.0).
9) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the allocated-crossing-slack space of a MS word file (Version 5.0).
10) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in the slack-crossing-allocated space of a MS word file (Version 5.0).
11) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in a deleted MS word 2007 file.
12) The case sensitive, left-to-right, ASCII keyword ''evidence'' is in a compressed MS word 2007 file.
13) etc.

Until now, we have mapped the searching function, specified the requirements and developed a reference set. We now know what need to be tested and what are the expectations. Hence, validating a EE tool that professes to have a search function can be as simple as testing this tool against the reference set and applying metrics (accuracy and precision) to determine the accuracy and precision of the results. Accuracy is the degree to which the results of the tool being tested match the specification for each of the functions. Precision is the degree to which the results of many test give the same result.

## 8. Conclusion

Electronic Evidence is not a discipline which has grown in the same way as other forensic sciences such as DNA and ballistics. A primary problem faced by researchers and practitioners is that the scientific foundations of the electronic evidence field have never been mapped and specified, and the functions which together make up the electronic evidence process have not been stated and characterized. In this work, we present a scientific and systemical description of the EE discipline through mapping fundamental functions required in the EE investigation process.

With the function mapping, we propose a new functionality orientated validation and verification paradigm of EE tools. Focusing on the searching function, we specify its requirements and develop a corresponding reference set. In the end, validating a EE tool can be as simple as testing this tool against the reference set. Compared to the traditional testing methods, our testing paradigm is extensible, tool and tool version neutral and transparent.

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 a tool validated in certain scenarios where it pass these test cases.

## REFERENCES

Appel EJ, Pollitt MM. Report on the digital evidence needs survey of state, local and tribal law enforcement. National Institute of Justice, US Department of Justice, Tech. Rep.; 2005.

Beckett J, Slay J. Digital forensics: validation and verification in a dynamic work environment 2007. p.266a.

Beebe NL, Clark JG. A hierarchical, objectives based framework for digital investigations process. Digital Investigation 2005;2:147–67.

Boehm BW. Software engineering: R and D trends and defence needs. In: Proceedings of the conference on research directions in software technology; Oct. 1997.

Bogen C, Dampier D. Unifying computer forensics modeling approaches: a software engineering perspective. In: Proceedings of the first international workshop on systematic approaches to digital forensic engineering. Taipei: 7–9 November 2005.

Bor-Wen Hsu I-CC, Laih CS. A dct quantization-based image authentication system for digital forensics. In: SADFE '05: proceedings of the first international workshop on systematic approaches to digital forensic engineering on systematic approaches to digital forensic engineering. Washington, DC, USA: IEEE Computer Society; 2005. p. 223.

Brian C. Digital forensics tool testing images, http://dftt.sourceforge.net. 2005 [Visit Dec. 2008].

Brian C. A hypothesis-based approach to digital forensic investigations, Ph.D. Dissertation, Purdue University; 2006.
Fisher MS. Software verification and validation: an engineering and scientific approach. Springer; 2007.
Good practice guide for computer based electronic evidence. Available from: www.nhtcu.org. Tech. Rep., [Visited Sep. 2008].
Guo Y, Backett J, Slay J. Report on reference set of searching function. Defence and Systems Institute, University of South Australia, Tech. Rep.; 2009.
IEEE(1012–2004), Draft standard for software verification and validation IEEE P1012/D12. IEEE.
ISO (17025E). In general requirements for the competence of testing and calibration laboratories. ISO, Geneva, Switzerland.
Kessels J. jkDefrag, http://www.kessels.com/ [visit March 2009].
Lin YC. Study of computer forensics from a cross-cultural perspective: Australia and Taiwan, Ph.D. Dissertation, University of South Australia; 2008.
McKemmish R. What is forensic computing? trends and issues, paper No. 118, Australian Institute of Criminology; June 1999.
Mohay GM, Anderson A, Collie B, McKemmish RD, de Vel O. Computer and intrusion forensics. Norwood, MA, USA: Artech House, Inc.; 2003.
NIST. Computer Forensics Tool Testing (CFTT) Project. Available from: http://www.cftt.nist.gov/index.html; 2002 [Visit Dec. 2008].
NIST. Software write block tool specification and test plan (version 3 September 1, 2003). Available from:. National Institute of Standards and Technology US Department of Commerce www.cftt.nist.gov; 2003. Tech. Rep., [Visited December 2008].
NIST. Digital data acquisition tool specification (draft 1 of version 4.0, October 4, 2004). Available from:. National Institute of Standards and Technology US Department of Commerce www.cftt.nist.gov; 2004. Tech. Rep., [Visited Dec. 2008].
NIST. Digital data acquisition tool test assertions and test plan (draft 1 of version 1.0, Nov. 10, 2005). Available from:. National Institute of Standards and Technology US Department of Commerce www.cftt.nist.gov; 2005. Tech. Rep., [Visited Dec. 2008].
NIST. Gsm mobile device and associated media tool specification and test plan (version 1.1, May 5, 2008). National Institute of Standards and Technology US Department of Commerce, www.cftt.nist.gov; 2008. Tech. Rep., [Visit Dec. 2008].
NIST. Forensic string search specification, V1, draft 1. March 2009. Available at: http://www.cftt.nist.gov/StringSearch.html; 2009.
NRC. Strengthening forensic science in the United States: a path forward (free executive summary) 2009; 2009.
Palmer G. A road map for digital forensics research. Report from the first Digital Forensics Research Workshop (DFRWS), DTR-T001–01. Research Site, Tech. Rep.. Rome: Air Force Research Laboratory; 2001
Reith C, Carr M, Gunsch G. An examination of digital forensic models. International Journal of Digital Evidence 2002;1(3):1–12.
Ruibin G, Yun CK, Gaertner M. Case-relevance information investigation: binding computer intelligence to current computer forensic framework. International Journal of Digital Evidence 2005;4(1):147–67.
SWGDE. Recommended guidelines for validation testing version 1. Scientific Working Group on Digital Evidence, www.swgde.org; July 2004. Tech. Rep., [Visited December 2008].
Wilsdon T, Slay J. Digital forensics: exploring validation, verification and certification. In: SADFE '05: Proceedings of the first international workshop on systematic approaches to digital forensic engineering. Washington, DC, USA; 2005. p. 48.

**Yinghua Guo** received his B.Eng. in telecommunication engineering in 1999, and M.Eng. in information engineering in 2002 from Xidian University, Xi'an, China. In 2004, He joined the Institute for Telecommunications Research (ITR), University of South Australia as a PhD candidate majoring network security. After receiving his PhD in network security in 2008, he commenced the role of research fellow in the Defence and Systems Institute (DASI). His main research focuses on the network security and forensic computing. Other research interests include computer security, intrusion detection system and wireless networking.

**Jill Slay** is the leader of Systems for Safeguarding Australia Research Centre and the Forensic Computing Lab, Defence and Systems Institute at the University of South Australia, Adelaide, Australia. Her research interests include information assurance, digital forensics, critical infrastructure protection and complex system modeling.

**Jason Beckett** is a PhD student at the Defence and Systems Institute Safeguarding Australia Research Laboratory at the University of South Australia, Adelaide, Australia. His research interests include Forensic Computing validation and verification.