

CRANFIELD UNIVERSITY

CHRISTOPHER JAMES HARGREAVES

ASSESSING THE RELIABILITY OF DIGITAL EVIDENCE FROM LIVE
INVESTIGATIONS INVOLVING ENCRYPTION

DEFENCE COLLEGE OF MANAGEMENT AND TECHNOLOGY

PHD THESIS

CRANFIELD UNIVERSITY

DEFENCE COLLEGE OF MANAGEMENT AND TECHNOLOGY

DEPARTMENT OF INFORMATICS AND SENSORS

PHD THESIS

2009

CHRISTOPHER JAMES HARGREAVES

ASSESSING THE RELIABILITY OF DIGITAL EVIDENCE FROM LIVE
INVESTIGATIONS INVOLVING ENCRYPTION

SUPERVISOR: PROFESSOR HOWARD CHIVERS

FEBRUARY 2009

ABSTRACT

The traditional approach to a digital investigation when a computer system is encountered in a running state is to remove the power, image the machine using a write blocker and then analyse the acquired image. This has the advantage of preserving the contents of the computer's hard disk at that point in time. However, the disadvantage of this approach is that the preservation of the disk is at the expense of volatile data such as that stored in memory, which does not remain once the power is disconnected. There are an increasing number of situations where this traditional approach of 'pulling the plug' is not ideal since volatile data is relevant to the investigation; one of these situations is when the machine under investigation is using encryption. If encrypted data is encountered on a live machine, a live investigation can be performed to preserve this evidence in a form that can be later analysed. However, there are a number of difficulties with using evidence obtained from live investigations that may cause the reliability of such evidence to be questioned. This research investigates whether digital evidence obtained from live investigations involving encryption can be considered to be reliable. To determine this, a means of assessing reliability is established, which involves evaluating digital evidence against a set of criteria; evidence should be authentic, accurate and complete. This research considers how traditional digital investigations satisfy these requirements and then determines the extent to which evidence from live investigations involving encryption can satisfy the same criteria. This research concludes that it is possible for live digital evidence to be considered to be reliable, but that reliability of digital evidence ultimately depends on the specific investigation and the importance of the decision being made. However, the research provides structured criteria that allow the reliability of digital evidence to be assessed, demonstrates the use of these criteria in the context of live digital investigations involving encryption, and shows the extent to which each can currently be met.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Prof. Howard Chivers for his support and guidance over the last three years; and also Prof. Tony Sammes and Dr. Mike Edwards who formed my thesis committee and have provided additional guidance at regular intervals. In addition, I would like to thank Marc Kirby for many interesting and insightful conversations about forensic computing and Lindy Sheppard for organising courses for me at the Centre for Forensic Computing. I would also like to thank Benoît Mangili for his help with aspects of *Linux*, miscellaneous programming problems and for introducing me to *Perl*, and also Alexeis Garcia-Perez and Jin Tong for many useful discussions about research methods. Also, final thanks to Catherine Hardie for taking the time to proof read this thesis, and also to my parents for their support throughout my seemingly perpetual education.

LIST OF CONTENTS

Abstract.....	i
Acknowledgements.....	ii
List of Contents.....	iii
List of Tables.....	vi
List of Figures.....	vii
List of Figures.....	vii
Chapter 1: Introduction.....	1
1.1 Introduction.....	1
1.2 Justification.....	2
1.3 Aim.....	3
1.4 Research Hypothesis.....	3
1.5 Research Methodology.....	3
1.6 Thesis Outline.....	6
1.7 Contributions.....	9
Chapter 2: Literature Review.....	11
2.1 Introduction.....	11
2.2 General Background.....	11
2.3 Encryption and Digital Investigations.....	25
2.4 Live Digital Investigations.....	35
2.5 Chapter Summary.....	56
Chapter 3: Assessing the Reliability of Digital Evidence.....	58
3.1 Introduction.....	58
3.2 Assessing the Reliability of Digital Evidence.....	59
3.3 Proposed Requirements for Digital Evidence.....	60
3.4 Existing Requirements for Digital Evidence.....	66
3.5 Satisfying these Requirements.....	73
Chapter 4: Completeness and Encryption.....	79

4.1 Introduction.....	79
4.2 Background.....	79
4.3 Methodology.....	81
4.4 Results.....	93
4.6 Conclusions.....	114
Chapter 5: Completeness and Intrusiveness	118
5.1 Introduction.....	118
5.2 Background.....	119
5.3 Methodology.....	125
5.4 Development of a System Monitoring Methodology	130
5.5 Results: Running Programs.....	142
5.6 Results: Connecting to a Live System	146
5.7 Results: Running Live Investigation Tools.....	150
5.8 Evaluation	155
5.9 Conclusions.....	160
Chapter 6: Accuracy	164
6.1 Introduction.....	164
6.2 Background.....	164
6.3 Methodology.....	173
6.4 GUI Based Key Recovery: <i>BitLocker</i>	175
6.5 Memory Image based Key Recovery: <i>TrueCrypt</i>	180
6.6 Evaluation	198
6.7 Conclusions.....	200
Chapter 7: Authenticity.....	203
7.1 Introduction.....	203
7.2 Background.....	204
7.3 Methodology.....	207
7.4 Development of Proof of Concept Tool.....	209
7.5 Results.....	215
7.6 Evaluation	218

7.7 Conclusions.....	220
Chapter 8: Evaluation	222
8.1 Introduction.....	222
8.2 Methodology Evaluation.....	222
8.3 Requirements Evaluation	223
8.4 Completeness Evaluation.....	226
8.5 Accuracy Evaluation.....	229
8.6 Authenticity Evaluation	232
8.7 Conclusions.....	233
Chapter 9: Conclusions	236
9.1 Conclusions.....	236
9.2 Contributions.....	236
9.3 Future Work	238
9.4 Final Summary	241
References.....	244
Appendix A.....	251

LIST OF TABLES

Table 1: Live tools necessary to investigate a <i>Windows</i> system (Mandia et al. 2003, p.97)	38
Table 2: Analysis techniques that can be performed using the <i>Volatility</i> framework	53
Table 3: A selection of ‘common findings’ from three of the fourteen different crime categories in NIJ (2001).	80
Table 4: Descriptions of whether the success of offline approaches to gaining access to encrypted digital evidence depends on the product category.	83
Table 5: Locations on disk that may assist in providing access to encrypted data.	87
Table 6: Predictions of availability of unencrypted data to an offline analysis (WinMagic, 2005).....	89
Table 7: Techniques used to test for the presence of plaintext data in various locations.	92
Table 8: Results from AxCrypt.	94
Table 9: Results from <i>GNU Privacy Guard</i>	96
Table 10: Results from Encrypt Files.	98
Table 11: Results from EFS.....	102
Table 12: Results from PGP.	105
Table 13: Results from <i>BestCrypt</i>	106
Table 14: Results from <i>Cryptainer</i>	107
Table 15: Results from <i>CompuSec</i>	109
Table 16: Modes of BitLocker.....	110
Table 17: Results from <i>BitLocker</i>	111
Table 18: Results from TrueCrypt.....	112
Table 19: Registry contents formatted to CSV format using the developed <i>Perl</i> script displayed in table form.	136
Table 20: Certainty scale described in Casey (2002a).....	169
Table 21: Positions and indexes of known plaintext in different modes of TrueCrypt.....	189
Table 22: Comparison of process models for digital investigations.....	252

LIST OF FIGURES

Figure 1: Screenshot of <i>FTK Imager</i>	40
Figure 2: Simplified structure of an EPROCESS block describing locations of important information in <i>Windows XP SP2</i>	46
Figure 3: The relationship between Page Directory, Page Tables, and Pages of physical memory.	48
Figure 4: PDB at offset 0x18 of the system process does not point to 0x00039000 but to 00319000 ...	50
Figure 5: Regular expression used in <i>Perl</i> script to locate 'System' process with command line output showing found system process.	50
Figure 6: Graphics from WinMagic (2005) showing the availability of locations on disk.....	88
Figure 7: Disk map of the test drive used, showing start and end sectors of the different partitions.....	90
Figure 8: <i>AxCrypt</i> and <i>Windows Explorer</i> right click integration providing the option for encryption and decryption.	93
Figure 9: <i>Windows Explorer</i> integration of <i>GNU Privacy Guard</i>	95
Figure 10: The <i>Encrypt Files</i> software providing access to files on disk and the option to encrypt or decrypt those files.....	97
Figure 11: Advanced attributes allowing the encryption of files using <i>EFS</i>	100
Figure 12: The structure of an <i>EFS</i> file (Microsoft, 2006d).	101
Figure 13: <i>PGP</i> interface for creating a new virtual disk.	104
Figure 14: <i>BestCrypt</i> interface for creating a new encrypted container.....	105
Figure 15: The <i>Cryptainer</i> interface.	107
Figure 16: <i>CompuSec</i> Pre-boot authentication.	109
Figure 17: The back of a PC with common ports highlighted.	129
Figure 18: <i>after.bat</i> , which is used to simplify creation of snapshots by copying the <i>.vmdk</i> and <i>.vmem</i> files to the 'after' subfolder.	132
Figure 19: Changes made between snapshots displayed with <i>fc.exe</i> (top) and the developed <i>Perl</i> script (bottom), the latter produces a cleaner, simpler list of changes.....	134
Figure 20: Registry contents extracted using <i>reg.pl</i>	135
Figure 21: Simplified architecture of fully automated results processing and report generation.	138
Figure 22: Main index page of the generated HTML report.....	139
Figure 23: Sample summary of file changes in the combined HTML report.	139
Figure 24: Results of running the built-in <i>manage-bde.wsf</i> script identifying encrypted volumes on a live system.	176
Figure 25: The 'Manage BitLocker Keys' graphical interface.	177
Figure 26: The format of <i>BitLocker</i> Recovery Keys.	177

Figure 27: XTS encryption of a single sub-block.....	183
Figure 28: The overall approach for demonstrating accuracy of live acquired images.	184
Figure 29: Depiction of the linear scan approach to key recovery. The keys used to perform test decryptions actually slide one byte at a time, rather than 256 as shown here.....	186
Figure 30: Parts of the keys displayed by the <i>TrueCrypt</i> GUI on creation of an encrypted container.	186
Figure 31: Part of a memory image showing the Primary and Secondary Master keys.	187
Figure 32: Known plaintext on FAT16 file systems.....	188
Figure 33: Keys recovered from a live memory dump.	191
Figure 34: MD5 hashes of live acquired container and offline decrypted container.	192
Figure 35: Differences between the live acquired Full Volume Encrypted drive and the offline decrypted version.....	193
Figure 36: Properties of a mounted encrypted container identified on a live system.	195
Figure 37: Tracing digital evidence artefacts back to a piece of physical evidence in a traditional digital investigation.	205
Figure 38: Tracing digital evidence artefacts back to a piece of physical evidence in a live investigation.	205
Figure 39: The overall methodology for demonstrating the origin of a piece of live acquired digital evidence.....	208
Figure 40: Displaying the MAC address under <i>Linux</i>	210
Figure 41: Volume Serial Number displayed with ‘dir’	211
Figure 42: Manufacturer’s serial number shown using <i>HD Tune</i>	212
Figure 43: The same serial number obtained from the label of the drive.	212
Figure 44: Calls to <i>dd</i> and <i>md5</i> from the <i>Perl</i> script.....	213
Figure 45: Code that recovers the manufacturer’s serial number.	214
Figure 46: Output from the live tool.....	214
Figure 47: Output from <i>hdparm -i /dev/sda</i> to obtain the manufacturer’s serial number.....	215
Figure 48: Photograph of the output from the live tool.	216
Figure 49: Photograph of the physical drive’s serial number.	216
Figure 50: Output of the offline tool which is the same as that produced on the live system (Figure 48), which links the live acquired disk image to the seized hardware.	217
Figure 51: Hashes produced do not match if the image was not acquired from the seized system.	217

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

As digital devices become ubiquitous, our day to day activities require more frequent interaction with digital systems, and as a result, more traces of our actions are left on these systems. Consequently, digital devices are often examined in order to infer what has happened in the real world. This process is referred to as a digital investigation.

A digital investigation is defined as *a process that formulates and tests hypotheses using digital evidence* (see Chapter 2). These hypotheses are tested by examining *digital evidence*, which is defined as *a set of reliable digital objects that support or refute a hypothesis* (see Chapter 2). Digital evidence can be used for a variety of purposes, from investigating violations of acceptable use policies to criminal offences. Many digital investigations involve the latter and as a result the term ‘digital investigation’ is often used interchangeably with ‘forensic computing’. This latter term can be referred to specifically as a ‘*forensic digital investigation*’, which is a digital investigation with the additional requirement that the obtained digital evidence needs eventually to be presented in court.

In digital investigations that involve seizing computer systems from the home or workplace of suspects during the course of the investigation, computer systems can be encountered while they are still powered on and running. The traditional approach to digital investigation has involved removing the power from these systems, i.e. ‘pulling the plug’. This has the advantage of preserving the contents of the computer’s hard disk at that point in time, since after the power is removed, no data can be written to the disk. However, this has the disadvantage that this preservation of the disk is at the expense of volatile data such as that stored in RAM, which does not remain once the power is disconnected.

There are an increasing number of situations where this traditional approach of ‘pulling the plug’ is not ideal, for example: cases where large volumes of data are involved; where systems are ‘mission critical’; when relevant digital evidence is stored in memory only; and also when the machine under investigation is using

encryption, i.e. data is stored in a form that cannot be understood without the correct decryption key. This research investigates the effectiveness of the traditional approach to digital investigations when encryption is involved, and examines the advantages and disadvantages of a live investigation as an alternative to this approach. Live investigations involve examining a digital system while it is still running and using the operating system (the software that runs on the hardware and allows other programs to run) of the machine being investigated to acquire, analyse or present digital evidence. Live investigations are permitted by Principle 2 of the ACPO Guidelines¹ and they are useful when encryption is involved since if physical access can be gained to a system at a point when the suspect is accessing the encrypted material, the investigator may be able to take control of the machine and will therefore also have access to the encrypted content.

1.2 JUSTIFICATION

As briefly described in the previous section and investigated in detail in Chapter 4, encryption poses a problem for the traditional approach to digital investigations and live investigations offer a simple mechanism to access the encrypted data in a form that can later be analysed. However, there are a number of difficulties with live investigations which are discussed in Section 2.4.4, for example, the difficulty in trusting the data supplied to live tools; the inherent intrusiveness of live techniques; the difficulty in verifying the output of live tools; and also ensuring that no evidence is missed. These difficulties mean that the reliability of evidence obtained using live investigation techniques could be called into question and, due to the lack of research and understanding of the subject, could result in digital evidence from live investigations being used when it should not be, or it not being used when it could be; either way, this could potentially result in an incorrect hypothesis being accepted.

¹ Principle 2 of the Association of Chief Police Officers' Good Practice Guide for Computer-Based Electronic Evidence states, "In circumstances where it is necessary to access original data held on a computer or storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions" (ACPO, 2007).

1.3 AIM

Encrypted evidence can cause problems for traditional digital investigations and live investigations provide a means to access evidence while it is still in its decrypted form. However, as described earlier, digital evidence from live investigations is potentially problematic since there are a number of challenges to using it. The aim of this research is therefore to determine the role that live digital investigations can play in investigations involving encrypted evidence.

1.4 RESEARCH HYPOTHESIS

Given that a digital investigation formulates and tests hypotheses by examining digital evidence, and that digital evidence is defined as a set of reliable digital objects that support or refute a hypothesis, this research is concerned with determining whether digital evidence recovered using live techniques from systems using encryption can be shown to be reliable and therefore accepted as digital evidence as part of a digital investigation. The research hypothesis is therefore:

Digital evidence obtained from live investigations involving encryption can be shown to be reliable.

1.5 RESEARCH METHODOLOGY

1.5.1 General Methodology

Digital evidence can be used for a variety of purposes and the decision of whether it is considered reliable depends on the situation and the person or persons making the judgement. This presents a problem in this research for assessing the reliability of digital evidence from live investigations, since adopting a subjective view of the reliability of digital evidence makes it extremely difficult to arrive at any conclusions. However, as described in Chapter 3, there are existing standards and requirements for

digital evidence² and therefore it is assumed that reliability can be assessed against objective requirements.

Based on this assumption, Chapter 3 proposes general requirements that can be used to assess the reliability of digital evidence. These are validated by comparing them to existing requirements and checking for consistency, and also demonstrating how current, accepted techniques for digital investigations satisfy them. Existing requirements are selected for comparison on the basis that they are well established, peer reviewed and/or used in practice. Requirements that disagree with those proposed are examined further to determine the cause of the discrepancy, since the difference may be due to existing requirements being specific versions of more general requirements. Also, since the definition for digital evidence is broad and accommodates its use in digital investigations as well as in the field of forensic computing, requirements that are specifically related to use of digital evidence in court are not considered to be appropriate for use in general requirements.

The requirements derived in Chapter 3 are then examined in Chapters 4-7, where the extent to which they can be satisfied for live investigations is determined. Each chapter contains its own methodology section which describes the approaches used. Chapters 8 and 9 evaluate and conclude about the extent to which digital evidence from live systems using encryption can meet the proposed requirements, and therefore be considered to be reliable.

In addition to this overall research strategy, there are also a number of research tools that are used throughout the testing of this hypothesis, which are described in the following sub-sections.

1.5.2 Use of Virtual Machines

Virtualisation is a technique that “lets you run multiple virtual machines on a single physical machine, sharing the resources of that single computer across multiple environments. Different virtual machines can run different operating systems and multiple applications on the same physical computer” (VMWare, 2009). These

² They are mostly in the form of principles for *forensic* digital investigations.

‘virtual machines’ are used throughout this research. Their multiple uses in digital investigations are discussed in detail in Pollitt *et al.* (2008), but in this research they are mainly used to allow virtual test systems (guests) to be quickly built in different configurations and run on a single physical machine (host) without the need for multiple pieces of hardware. Virtual machines also provide the advantage of quick access to the hard drives of the virtual machines without needing to spend long periods of time creating disk images of physical drives. This is possible since the disks of the virtual machines are represented on the physical system’s hard drive as one or more files (.vmdk files in *VMware*) which can be opened in any forensic software package and are treated as physical disk images. The memory of the virtual machine is also represented on the host system as a file (.vmem files in *VMware*) and can also be acquired in this way. There are some limitations to using virtual machines, specifically the inability to virtualise some hardware (e.g. Firewire) and some differences when analysing images of the virtualised memory. These are discussed in more detail later.

1.5.3 Use of Forensic Software to Examine Disk Images

Throughout this research, disk images (or the .vmdk files from *VMware* virtual machines) are examined. This is performed using ‘forensic software’ and there are a number of products from which to choose. *X-Ways Forensics* was chosen as the primary tool since it is a fraction of the cost of *EnCase* and *Forensic Toolkit (FTK)* and offers all the functionality needed for this research. *X-Ways Forensics* can interpret the file systems used in this research (FAT and NTFS), allowing traversing of these file systems and also the recovery of deleted files. It also offers a ‘Data Interpreter’ function that is useful for converting embedded dates and times and other values. *X-Ways Forensics* can also be used for file comparisons and text or hexadecimal searches and extractions (Casey, 2004b). While *EnCase* and *FTK* are more commonly used for performing ‘real’ digital investigations, they offer no advantages in this research.

1.5.4 Development of Software Tools

Also, throughout this research, software is written to perform a variety of tasks. No single language is used since each offer their own advantages and disadvantages. For example *Java* is used to develop Graphical User Interfaces, *C* is used where speed or low level access is a necessity, and *Perl* is used for text parsing and scripting routine tasks.

1.5.5 Testing for Randomness

On a number of occasions it is desirable to test for randomness, and in this case a variety of statistical tests are applied. Forster (2005) implements the Chi-Square test as part of an automated technique to identify encryption, since the technique is described as the only statistic that “was capable of isolating the pseudo-randomness of the encrypted file”. In this research an existing piece of software *ENT*, (Walker, 2008), is used for testing for randomness and detecting encrypted data. *ENT* performs a variety of statistical tests for randomness including, the Chi-Square test and others: entropy, the reduction in size though compression, the mean value, the Monte Carlo value for Pi and the serial correlation co-efficient. Where these are used, they are discussed in more detail.

1.6 THESIS OUTLINE

This section describes the structure of the thesis.

Chapter 2 provides a review of relevant literature and describes in greater detail some of the ideas introduced in this section. It discusses the differences between digital investigations and forensic digital investigations, which is important since additional requirements for evidence are imposed by the latter. It also defines digital evidence and discusses the importance of reliability. The traditional ‘pull the plug’ approach to digital investigations is also discussed, along with the challenges that this approach faces. One such challenge is encryption, which is discussed in detail, along with the approaches that can be used by investigators after the power has been removed to

attempt to gain access to encrypted evidence. The limitations of these approaches are also discussed. Live digital investigations are presented as an alternative approach, including the distinction between live acquisition and live analysis. Also, the problems with results obtained using live techniques are described.

Chapter 3 explains the need to determine basic requirements for digital evidence in order to assess reliability. It also describes existing requirements and shows how live digital investigations cannot meet some of those currently in use. However, it then shows that some of the existing requirements cannot be considered to apply to digital evidence in general, since they are either specific to law or can be shown to be technologically specific means of satisfying other more general requirements. It is shown that live investigations may also be able to satisfy these general requirements. The chapters that follow then investigate the extent to which the derived general requirements of completeness, accuracy and authenticity can be satisfied for live investigations.

Chapter 4 examines the completeness requirement and considers the likely success of existing offline approaches for attempting to access encrypted evidence. It also considers which of the approaches' success is affected by the amount of the disk that remains in unencrypted form after the power is removed. Encryption products are categorised based on the locations on disk they encrypt, and for those categories where offline access is unlikely using existing approaches, it considers if a live investigation could offer a more complete and therefore reliable set of digital evidence, which would support the overall hypothesis of this research.

Chapter 5 complements the previous chapter and examines how completeness could be adversely affected by performing a live investigation. Live tools are inherently intrusive and as a result could overwrite potentially relevant digital evidence. This chapter considers how to assess what evidence is lost by monitoring the changes caused to test systems when using various live investigation tools and techniques. The

results of testing in this way can be used to predict the data that will be overwritten and therefore the extent of the decrease in completeness of preserved digital evidence. This chapter identifies the limitations of current techniques for monitoring systems and develops a more comprehensive system monitoring methodology which is used to test a number of live tools and techniques, including running live acquisition and analysis tools, and also the effect of connecting to a system using various interfaces.

Chapter 6 considers how the accuracy of results obtained from live investigations can be assessed. In traditional digital investigations, accuracy can be demonstrated since the techniques used are repeatable and can be performed by multiple examiners on multiple copies of the same digital data. In this research a distinction is made between the acquisition and analysis of digital evidence from live systems. The consequence of this is that once evidence is acquired, the accuracy of the analysis stage of a live investigation can be demonstrated using the same tried and tested means as current investigations i.e. repeatability. This chapter therefore focuses on how to assess the accuracy of the acquisition stage of a live digital investigation. This is achieved by first considering the nature of error in digital investigations, which then allows methods to be developed to assess this error.

Chapter 7 examines how the authenticity of evidence obtained from live investigations can be demonstrated. In traditional digital investigations, the original physical evidence is always accessible and this contains the raw data from which digital evidence is extracted. Therefore, if the procedures used to recover digital evidence are thoroughly documented, it can always be shown how digital evidence was obtained from a physical piece of evidence that can be traced back to a person. In a live investigation, the original evidence may not be available after the power is removed; this section considers in this case how live acquired data can be demonstrated to originate from a particular piece of physical evidence.

Chapter 8 revisits the original requirements from Chapter 3 and evaluates the extent to which they have been satisfied in each of the previous chapters for live digital investigations involving encrypted evidence.

Chapter 9 summarises the conclusions and contributions of this research and describes future work.

1.7 CONTRIBUTIONS

This research tests the hypothesis that *digital evidence obtained from live investigations involving encryption can be shown to be reliable*, and demonstrates the strengths and weaknesses of performing live investigations of systems using encryption. The outcome of this is a set of requirements, which allows the reliability of digital evidence to be assessed. These requirements for digital evidence are clearly defined and the research as a whole acts as an example of how they can be used. These requirements could also be used in future to assess reliability of other types of digital evidence.

Also, in this research, categories of encryption products are validated and it is shown what affect these have on the locations on disk that become inaccessible when the power is removed. It is also shown how the categories affect offline approaches to attempting to gain access to encrypted digital evidence. The research therefore provides a demonstration of the increase in the amount of preserved evidence that a live investigation offers over the traditional approach, providing support for the use of live investigations.

This research also demonstrates the adverse affect that live investigations can have on the amount of preserved digital evidence. In the course of the research, a methodology and software tool is developed that simplifies the process of recording changes made to test systems. This allows the footprints of live tools to be determined, which is essential in minimising the loss of digital evidence due to actions of an investigator on a live system. This aspect of the research also has a number of additional future applications, including identifying locations of forensic

artefacts left by software, and also in computer security research for monitoring honey pots.

This research also provides a general definition for error in digital investigations, which is not available in current literature. This provides direction for the expression of error when presenting digital evidence. This definition of error is used to determine how error can be minimised in live investigations. The approach to this involves the development of a method that allows repeatability to be used to demonstrate the accuracy of live acquired copies of encrypted evidence. This involves acquiring specific information from the system at the same time as a decrypted copy of the encrypted evidence, which enables offline decryption of the static encrypted data. This is demonstrated in two ways: using the built in GUI of *BitLocker* and recovering decryption keys from a memory dump of a system running *TrueCrypt*.

Finally, it is shown how the physical origin of live acquired data can be demonstrated, even if the original data is unavailable, by integrating physical identifiers that are available before and after ‘pulling the plug’ into the acquisition process.

Many of these contributions have resulted in peer reviewed publications. Obtaining recovery keys in order to allow later access to *Bitlocker* encrypted data is discussed in Hargreaves and Chivers (2007) and Hargreaves *et al.* (2008). The latter also discusses the difficulty in gaining offline access to *EFS* encrypted files on *Windows Vista*. The key recovery approach to demonstrating accuracy of acquired digital evidence is discussed in Hargreaves and Chivers (2008b), where the ‘linear scan’ approach to key recovery is introduced. This key recovery approach is also used in Hargreaves and Chivers (2008a) to demonstrate how live imaging could be avoided in cases where it is impractical, such as when very large amounts of data are involved. Both papers on key recovery also include other aspects of this research, including the types of offline approaches that can be used for gaining access to encrypted evidence.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

This background chapter has three sections. First, general background is provided for digital investigations and digital evidence. This is necessary since if reliability of digital evidence from live investigations is to be assessed, then definitions for ‘digital evidence’ and related terms such as ‘digital investigation’ must be clear. This section also discusses the challenges to the traditional approach to digital investigations. Secondly, one of the challenges to traditional digital investigations is discussed in detail: the challenge of encryption. This is introduced and approaches are described that can be used in attempts to gain access to encrypted digital evidence during offline examinations. Also, the difficulties and limitations of the approaches are described. Finally, live digital investigations are defined and discussed, including reviewing existing live investigation techniques and the challenges they face.

2.2 GENERAL BACKGROUND

2.2.1 Introduction

This section introduces digital investigations and describes the specifics of forensic computing, both of which involve recovering digital evidence, which is also defined. This ‘back to basics’ section is necessary since terms such as ‘digital investigation’ and ‘forensic computing’ are often used interchangeably, even though there are important differences. The differences are particularly relevant in Chapter 3, where requirements for digital investigations are considered, and it becomes clear that different groups have different standards for judging the reliability of digital evidence. This section also describes the traditional ‘pull the plug’ approach to digital investigations since a live investigation is a different approach and it is important to clarify the differences. Finally, challenges to the ‘pull the plug’ approach are discussed which demonstrates its limitations and the necessity of a new approach.

2.2.2 Definitions and Digital Evidence Introduction

Carrier (2006a) makes the distinction between a digital investigation and a digital forensic investigation since “many corporations and intelligence agencies conduct investigations and collect evidence that will not be entered into a court of law”. Despite ‘forensic’ meaning “of or relating to courts of law” (Oxford, 2008), the terms ‘digital investigation’ and ‘forensic computing’³ are often used interchangeably. In this research a similar distinction is made as in Carrier (2006a). The following sections define the terms ‘digital investigation’, ‘forensic computing’ and ‘digital forensic investigation’.

Digital Investigation

Carrier (2006a) describes the goal of a digital investigation as to “make valid inferences about a computer’s history”, which is achieved by making observations and formulating hypotheses. Before hypotheses can be tested, digital data must be observed; but unlike the physical world it is not possible for us to view digital data directly and we rely on both hardware and software to report this information (Carrier, 2006a). Therefore, before higher level hypotheses are formed about computers’ histories, more basic hypotheses are made stating that the observed data (reported by hardware and software) is equal to the actual data (Carrier, 2006a). Furthermore, as described in Sammes and Jenkinson (2007 p. 63), patterns of bytes can represent anything; meaning is only derived when rules are applied to interpret this raw data. Therefore, hypotheses also need to be formed which state that not only that the actual data is equal to the observed data, but also that the interpretation of this observed data is “consistent with the interpretation used to establish the patterns” (Sammes and Jenkinson, 2007 p.63). As described in Carrier (2006a) “at the lowest levels of abstraction, hypotheses will be used to reconstruct events and to abstract data into

³ The term *computer forensics* is also used. However as described in Casey (2002 p.31), this is “a syntactical mess that uses the noun *computer* as an adjective and the adjective *forensic* as a noun, resulting in an imprecise term”. The term ‘computer forensics’ will therefore not be used in this research.

files and complex storage types. At higher levels of an investigation, hypotheses will be used to explain user actions and sequences of events”. Therefore, a digital investigation makes and tests both high and low level hypotheses.

The definition used in Carrier (2006a) for a digital investigation is “a process that formulates and tests hypotheses to answer questions about digital events or the state of digital data” (Carrier, 2006a). However, digital events occur on a system, often as a result of interactions with another digital device, or as a result of interactions with the real world. Since these interactions affect the state of the computer system, they too are part of the computer’s history and it is sometimes necessary to infer what these interactions were. For example, it is necessary to determine if a web site was visited because a user manually typed in the web address or it was opened automatically by visiting another site. As a result, a digital investigation may need to answer questions not only about “digital events or the state of digital data” (Carrier, 2006a), but also about what real world events or other interactions caused digital events on the computer system to occur and therefore digital data to have its current state. Consequently, the definition used in this research for a digital investigation removes references to answering specific questions about digital data. This change allows any hypothesis to be tested during a digital investigation. Instead, to define specifically a *digital* investigation, it is highlighted that to test these hypotheses, *digital data* is examined. In this research, a digital investigation is therefore defined as ‘a process that formulates and tests hypotheses using digital evidence’. Digital evidence is discussed later in this chapter.

Forensic Computing & Forensic Digital Investigation

In order to define the forensic computing field, due to its relatively recent conception, it is useful to begin with definitions from traditional forensic science: “strictly speaking, Forensic Science is the application of science to law and is ultimately defined by use in court” (Casey, 2004a). Forensic computing could be defined based on this to be ‘the application of computer science to law’. Casey (2004a p.21) takes the approach of broadening the definition of forensic science to “the application of

science to investigation and prosecution of crime, or the just resolution of conflict”. The purpose of using this broader definition is in order to “encourage corporate digital investigators to apply the principles of Forensic Science”. While this is accepted as a worthwhile goal, this research does not use this broader definition of forensic science since it contradicts the more accepted definition of ‘forensic’ (“of or relating to courts of law” (Oxford, 2008)) and therefore defines forensic computing as *the application of computer science to law*. Based on this definition, the term ‘digital forensic investigation’, used in Carrier (2006a) can be used to describe a digital investigation which has the ultimate purpose of recovering digital evidence that could be admitted to a court of law. Here the principle from the definition of forensic computing (application to law) is taken and applied to a digital investigation, giving the definition for ‘forensic digital investigation’ of *a process that formulates and tests hypotheses using digital evidence, where the results could be admitted to a court of law*. Forensic digital investigations are therefore specific instances of digital investigations, which have the additional requirement of the results being admissible in a court of law.

Digital Evidence

Definitions

Examining the definitions in the previous section, both digital investigations and forensic digital investigations use digital evidence to formulate and test hypotheses. This section considers alternative definitions for digital evidence and demonstrates why the definition in Carrier (2006a) is most appropriate. It is difficult to find a single, agreed upon definition of digital evidence, since it has a different meaning to different groups involved with digital investigations and those involved with the specifics of forensic computing. This difference is also evident when looking at multiple definitions of the word ‘evidence’; one is general: as a means to determine whether a belief or proposition is true; the other to establish facts in a legal investigation (Oxford, 2008). As discussed earlier, since the ‘forensic’ aspect is considered here to be a specific type of digital investigation, for ‘digital evidence’ to

be applicable to digital investigations and their forensic counterpart, a general definition of digital evidence is necessary. The following examples demonstrate the range of definitions that exist from different groups who conduct digital investigations. It also explains why a general definition of digital evidence is chosen.

The *Scientific Working Group on Digital Evidence* accepts members only from active law enforcement (Scientific Working Group on Digital Evidence, 2006) and their view of digital evidence tends towards the legal definition. Digital evidence is defined as “Information of probative value stored or transmitted in digital form” (Scientific Working Group on Digital Evidence, 2000). The word ‘probative’ is primarily a term in law meaning “affording provide proof or evidence” (Oxford, 2008) and as a result, this definition suggests that the term ‘digital evidence’ should be used only in a legal context.

Another definition that also focuses on the investigation of an offence is used in Casey (2004a p.12), “any data stored or transmitted using a computer that support or refute a theory of how an offence occurred or that address critical elements of the offence such as intent or alibi”. This definition also refers to evidence for legal purposes but does not restrict the use of digital evidence to proving an offence. It can also be used to support or further an investigation.

A definition with a wider scope comes from the United Kingdom’s *Association of Chief Police Officers*. It does not use the term ‘digital evidence’ specifically, but defines ‘computer based evidence’ as “information and data of investigative value that is stored on or transmitted by a computer” (ACPO, 2007). The wider scope of this definition is primarily due to the vague term ‘investigative value’, but nevertheless it is an all encompassing definition.

Sommer (1999) also states that in law, ‘evidence’ is “no more and no less than that which tends to persuade the court to a particular conclusion”. Even though this refers to persuading a court it does describe evidence as being used to come to a conclusion, which is more consistent with the general definition in Oxford (2008) of “determining whether a belief or proposition is true”. This is also supported elsewhere in Sommer (1999): “[evidence] is material which is used to establish the truth of a

particular fact or state of affairs” and in Miller (1992), “evidence is information used to decide whether disputed propositions are true”.

A more precise definition for digital evidence that agrees with this thinking and avoids referring only to its use in court is presented in Carrier (2006a), which compliments the earlier definitions of digital investigation and digital forensic investigation. In Carrier (2006a), digital evidence is defined as “digital data that supports or refutes a hypothesis about digital events or the state of digital data”. This definition uses the more general definition of evidence and takes into account that digital evidence may be used outside of a legal or criminal investigative context and simply used to determine the correctness of a belief or idea.

As discussed earlier, the specifics of hypotheses being about digital events or the state of digital data should be removed, since digital events are often representations of real events, e.g. the creation of a *Windows* Registry entry in TypedURLs is caused by the RegSetValue operation, which can be caused by a user typing text into the address bar of *Internet Explorer*. Therefore, hypotheses can be made not only about digital events but also the real world events that caused the digital events to occur.

Also, Carrier (2005 p.4) uses a similar but slightly different definition for digital evidence: “a digital object that contains reliable information that supports or refutes a hypothesis”. This definition explicitly states that digital objects must contain reliable information in order to be used as digital evidence. In this research, this requirement is considered to be a necessary constraint, since if evidence is used to support or refute a hypothesis, then the use of unreliable data could lead to an incorrect hypothesis being supported. Therefore, since this definition specifies that digital objects should contain reliable information, and it does not exclude hypotheses about the real world, in this research a definition based on Carrier (2005 p.4) is used. Digital evidence is therefore defined in this research as *a reliable digital object that*

*supports or refutes a hypothesis.*⁴ However, this definition does have a limitation, which is the lack of clarity of the term ‘reliable’. Reliability of digital evidence is difficult to define and this is discussed in detail in Chapter 3.

Abstracted nature

A property of digital evidence that is outside the scope of its definition but is important to discuss, is that digital evidence is a representation of some physical evidence. Carrier (2003) describes that this is more than a just a simple physical/digital divide and explains how digital investigation tools translate data through multiple layers of abstraction⁵. This is necessary because “all data, regardless of application, are represented on a disk or network in a generic format, bits that are set to one or zero”⁶ (Carrier, 2003) and as Casey (2004a p.16) summarised, “we never see the actual data but only a representation”. Therefore, all digital objects are abstractions of something physical and if digital evidence is *a reliable digital object that supports or refutes a hypothesis*, digital evidence must also be an abstraction of a physical piece of evidence.

⁴ This definition is for digital evidence (singular). Another definition of digital evidence is also sometimes used in this research, where digital evidence (plural) is defined as ‘a set of reliable digital objects that support or refute a hypothesis’.

⁵ In general computer science, levels of abstraction determine “the level of complexity by which a system is viewed” (TechWeb, 2008) and abstraction layers in digital forensics have the same function.

⁶ This statement itself is an abstraction layer and hides details of how the binary data is stored on disk. Ones and zeros are not stored on disk since “only a flux change can create a signal, so every bit needs to be implemented by some kind of flux change; usually a reversal of magnetisation” (Sammes and Jenkinson 2007 p.108) and as a result a number of different encoding schemes are used to represent patterns of ones and zeros on disks.

2.2.3 Traditional Forensic Computing Approach

The traditional philosophy of forensic computing can be best summarised by the four principles in the ACPO Good Practice Guide for Computer-Based Evidence (ACPO, 2007):

- *No action should change data held on a computer or storage media which may be relied on in court.*
- *In circumstances where it is necessary to access original data held on a computer or storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions.*
- *An audit trail or other record of all processes applied to computer based electronic evidence should be created and preserved. An independent third party should be able to examine those processes and achieve the same result.*
- *The person in charge of the investigation has overall responsibility for ensuring that the law and these principles are adhered to.*

The guide also contains further, more detailed instructions of how to conduct searches at crime scenes. The ACPO guidelines include advice for encountering computers that are switched on or computers that are switched off. If the computer is off then the guide states “do not under any circumstances switch the computer on”. This is because when a computer starts up a number of files and their metadata are changed, which could overwrite potential digital evidence.

If a computer is encountered in a powered on state then until recently (before ACPO Version 4 in 2007) the advice was “If no specialist advice is available, remove the power supply from the back of the computer without closing down any programs.”

(ACPO, 2003)⁷. This approach has become known as the ‘pull the plug’ method and is a simple and effective way to preserve the contents of the hard drive of a computer system. It is sometimes referred to in this research as a ‘traditional digital investigation’. This approach does have limitations: “It is accepted that the action of switching off the computer may mean that a small amount of evidence may be unrecoverable if it has not been saved to a storage medium but the integrity of the evidence already present will be retained” (ACPO, 2003). This will be discussed later.

Once the computer is in a powered off state (either encountered in that way, or the power was removed) the physical equipment can be removed along with any other material at the scene which may be relevant (diaries, notebooks, manuals etc. (ACPO, 2007)). Further steps can be found in guidelines from the National Institute of Justice (2004), that describe the detail of creating an exact duplicate of the hard drive of the seized equipment using a write blocker, which allows an investigator to “preserve and protect original evidence” (National Institute of Justice, 2004) by physically preventing any writes being made to the original hard drive. It is this duplicate that is then examined for digital evidence, since the duplicate can always be shown to be identical to the original. This is usually achieved using cryptographic hashes, where a mathematical function is applied to the whole data set to produce a fixed length bit string (Schneier, 1996 p.30). It is computationally infeasible to change the data in a way that will produce the same hash.

2.2.4 Digital Investigation Methods and Process Models

There are a number of process models for digital investigation based around both investigating computer security incidents, and also law enforcement procedures. There are also a number of abstract models that attempt to capture the general process of a digital investigation that can be applied equally to corporate investigations, incident response and law enforcement.

⁷ This is still in ACPO 2007 but also contains “Where possible, collect data that would otherwise be lost by removing the power supply e.g. running processes and information about the state of the network ports at that time”.

Carrier (2002) does not explicitly present a process model, however, digital forensics is described as having three major phases: acquisition, analysis, presentation. While this is not presented in Carrier (2002) as a process model for digital forensics, it can be used to broadly describe the process. Acquisition is concerned with “sav[ing] the state of a digital system so that it can later be analyzed”. In the traditional forensic computing approach described in the previous section, this phase can include locating the physical evidence upon which digital evidence resides, powering off the system to preserve the contents of the hard drive, seizing the machine and securely transporting to a lab and then acquiring a duplicate of the contents of the hard drive. Analysis “takes the acquired data and examines it to identify pieces of evidence” (Carrier, 2002). The presentation stage “presents the conclusions and corresponding evidence from the investigation” (Carrier, 2002) and the format of this presentation will vary based on the context of the digital investigation (corporate/law enforcement). However, it is on the presentation of evidence that a decision is likely to be made about whether hypotheses are believed to be correct.

There are also a significant number of more detailed and more complex models (Baryamureeba and Tushabe, 2004, Carrier and Spafford, 2003, Farmer and Venema, 2004, Mandia *et al.*, 2003, National Institute of Justice, 2004, Palmer, 2001, Reith *et al.*, 2002, Beebe and Clark, 2005, Ciardhuáin, 2004). However, these can be approximately mapped to the acquisition, analysis, presentation model (see Appendix A) where many of the models expand stages to provide more detail. For example, Reith *et al* (2002) has preservation and collection stages which are both concerned with acquisition. As a result, in this research, the higher level Carrier (2002) process model of acquisition, analysis and presentation is used.

2.2.5 Challenges to Traditional Digital Investigations

This section describes some of the major challenges to digital investigations and also how in some cases there is a move away from the ‘pull the plug’ approach described

in the earlier sections, towards carrying out actions on the live system (described in detail later in Section 2.4).

High Volume of Data

“The amount of data that exists in digital form is growing rapidly” (Craigier *et al.*, 2005) and this rapid increase in storage capacity is considered one of the greatest challenges in digital forensics (McKemmish, 1999). This is because “larger disk capacities increase the time required for analysis and the difficulty and expense of collecting all disk evidence” (Adelstein, 2006).

A related problem described in Roussev and Richard III (2004) is the exponential growth in storage capacity compared to the linear growth in input/output transfer speeds. This means that the imaging stage of an investigation remains an inherent bottleneck in the current forensic process.

In addition to the problem of the increased storage capacity of individual machines, another issue is the number of machines that could be included in an investigation. This is a problem due to the limited scalability of the traditional forensic approach (Sommer, 2004), since for each machine that is to be included in an investigation, the machine must have the power removed, the hard drive imaged and then analysed. Home users may now have more than one computer in a household, or more than one per individual, resulting in an increase in the resources needed to conduct an investigation. In enterprise environments where investigations could span tens or hundreds of machines, the traditional approach is simply not feasible.

Ubiquity of Digital Evidence

In addition to the increase in the number of computer systems that may need to be included in an investigation, digital evidence can also be found on an increasingly diverse range of devices. Since the definition of digital evidence is *a reliable digital object that supports or refutes a hypothesis*, any device capable of storing digital data may contain digital evidence. These can include ‘traditional’ computer systems, Personal Digital Assistant (PDAs), mobile phones, digital cameras, MP3 players,

digital photo frames, storage media such as USB sticks, Compact Flash and Secure Digital cards.

Different devices can store data in different ways and the traditional approach may not apply, for example, removing the power from a PDA may eventually cause a loss of data (ACPO, 2007).

Evidence in Memory Only

Another problem with the ‘pull the plug’ approach to digital investigations is that in some cases, evidence that is relevant to the investigation may not be stored on the hard disk of the computer system and may reside only in memory. Three examples of this are described in the following sections.

Messaging applications

Carvey (2004) describes a situation where a live messenger session is encountered in the course of responding to reports of missing children. In this case critical evidence from instant messaging applications could be lost if standard procedure was followed and the power was disconnected. By performing a live investigation, information can be retrieved that may be stored purely in memory such as IP addresses (as evidence of a direct connection initiated between instant messaging clients) and records of conversations which may not necessarily be logged to disk. Both types of evidence could be useful in furthering the investigation.

Malware in memory

It is possible for malware to reside only in the memory of a computer system. This is described in Burdach (2004): “sometimes the live procedure is the only way to acquire incident data because certain types of malicious code such as Loadable Kernel Module (LKM) based rootkits are loaded into memory only and don’t modify any files or directories.” While Burdach (2004) discusses mainly *Linux* systems, it also mentions that this also applies to *Windows* and “the Code Red Worm is a good [*Windows*] example where the malicious code was not saved to a file but was inserted into and then run directly from memory.” More recent examples are provided in Vidas

(2007), e.g. the *SQL Slammer worm*. While memory only malware may have limited effectiveness as it may not survive a reboot without compromising at least one software component that gets loaded on system boot (Hoglund and Butler, 2006 p.46), the possibility does exist that malware may reside in memory only, particularly on systems that remain powered on for extended periods of time.

It is also possible for defendants to use the ‘hacker’ or ‘trojan defence’ (Ghavalas and Philips, 2005, Vidas, 2007, Haagman and Ghavalas, 2005). For example, a suspect may deliberately download some malware “unrelated to material they are accused of possessing” (Vidas, 2007) and claim that the malware was responsible for that material. It may be possible to examine the hard disk and identify the nature of the malware and Kennedy (2006) describes how this can be done using a combination of antivirus software, MD5 hashes and a search for ‘triggers’ (events that will trigger its execution, e.g. Registry start-up locations). However, claims of memory only malware, or malware that has deleted itself are possible.

Privacy Mode of Browsers

Also, since the release of Google’s browser *Chrome*, which offers ‘Incognito Mode’ (Google, 2008) which prevents browsing and downloading histories from being logged, other browsers are also offering this functionality, including *Internet Explorer 8*’s ‘InPrivate’ mode (Zeigler, 2008) and *Firefox 3.1*’s ‘PrivateBrowsing’ (Mozilla, 2008). For *Internet Explorer 8*, Zeigler (2008) describes a number of pieces of data that are not recorded, e.g. addresses typed into the address bar. However, it also states that “new temporary Internet files will be deleted after the Private Browsing window is closed”. However, the approach in Mozilla (2008) specifically describes “not writ[ing] anything to disk” as one of the top level requirements, which implies that data needed for the session will be kept in memory only. Therefore, while some browsers are resorting to deleting data after the session, others are trying to implement a full memory only privacy mode and in these cases, pulling the plug would erase the only traces of recent browsing activity.

Size of Memory

The ACPO (2003) guidelines state that “It is accepted that the action of switching off the computer may mean that a small amount of evidence may be unrecoverable if it has not been saved to a storage medium but the integrity of the evidence already present will be retained”⁸. However, as the sizes of RAM increase, with computer systems sold to home users having capacities of 4GB (Sutherland *et al.*, 2008), the amount of potential evidence that is lost due to the traditional ‘pull the plug’ approach is significantly larger.

Encryption

Finally, another challenge facing the traditional digital investigation approach is the use of encryption as an attempt to conceal evidence from an investigation. This is discussed in detail in Section 2.3.

2.2.6 Summary

This section has defined a digital investigation as *a process that formulates and tests hypotheses using digital evidence*. This is performed by examining digital evidence, which is *a reliable digital object that supports or refutes a hypothesis*. Digital evidence must be shown to be reliable since if it is used to support or refute a hypothesis and is not reliable, this could result in an incorrect hypothesis being supported and ultimately an incorrect conclusion being drawn. This constraint of reliability is examined in Chapter 3.

This section has also discussed the traditional approach to digital investigations, where the power is removed from the system at the scene, preserving the contents of the disk at the expense of volatile memory. The advantages of this approach and how it addresses the need for digital evidence reliability are discussed later in Chapter 3. This section also presented the challenges that digital investigations currently face and specifically discussed situations where this ‘pull the plug’ approach

⁸ However, this is not in ACPO (2007)

is inadequate since the discarded evidence from memory is important to the investigation. These challenges included situations where evidence is available in memory only, e.g. instant messenger applications, memory only malware, and certain browsers' privacy modes. Finally, this section mentioned the use of encryption, which is the specific context in which live investigations are considered in this research, and is discussed in detail in the following section.

2.3 ENCRYPTION AND DIGITAL INVESTIGATIONS

2.3.1 Introduction

This section introduces encryption and describes how it causes problems for traditional digital investigations. The section also describes existing approaches that can be used to attempt to gain access to encrypted evidence. The section also explains how live investigations can help with the problem of encrypted evidence.

2.3.2 Background

According to Schneier (1996 p.1), cryptography is “the art and science of keeping messages secure”. However, it can be used not only to preserve the confidentiality of messages, but also of stored data. Encryption is a process which takes data (the plaintext) applies a mathematical function with a key and produces a ciphertext. The reverse process, decryption, takes that ciphertext, applies a mathematical function with a key and produces the original plaintext.

Schneier (1996 p.4) describes that there are two general types of key based cryptographic algorithms: symmetric and asymmetric (also known as ‘public key’). In symmetric algorithms, the encryption key can be calculated from the decryption key (in most cases they are the same). In asymmetric algorithms, it is computationally infeasible for the decryption key to be derived from the encryption key, allowing one of the keys to be public without compromising the other.

Using these principles it is possible to encrypt network traffic, specific communications technology such as e-mail and also to prevent access to files stored on a media (Denning, 1999 p.306). It is this latter use with which this research is

concerned. Cryptography has many legitimate uses, and as stated in Wolfe (2003), “no investigator should make any judgement as to innocence or guilt merely because the suspect has chosen to protect his or her privacy using data encryption”.

However, encryption can be used by the criminal elements of society to conceal evidence of crimes. Therefore, if encrypted material is encountered in the course of a digital investigation, it is usually desirable to gain access to the contents since “it is likely that many encrypted files will contain evidence as it is usually incriminating or unlawful material that suspects seek to hide in this way” (Forster, 2005). Since it is desirable to access all parts of the disk to search for digital evidence that supports or refutes hypotheses, the use of encryption by the suspect has the potential to impede or even stop an investigation and certainly “has the effect of frustrating enquiries in the immediate period following the arrest of suspects and the seizure of computer equipment” (Home Office, 2006). The point is also made in Denning and Baugh (1999) that “even when decrypted material has little or no investigative value, considerable resources are wasted in reaching that determination”.

Also, the use of encryption is increasing. A report from the Home Office (2006) states that in the two to three previous years “investigators have begun encountering encrypted and protected data with increasing frequency”. Several examples are provided where encrypted data has adversely affected investigations:

Suspect charged with possession of a collection of images including extreme level 4 images (penetrative adult abuse) of babies and some level 5 images (sadism and bestiality). Encrypted files were seized that the police cannot access, giving rise to concern they may contain worse material.

Suspect charged with possession of a huge amount of level 1 images (erotic posing with no sexual activity). These images were protected insecurely and were made intelligible. Other data remains encrypted and unintelligible.

Three individuals were convicted for possession and making of indecent images. All were in possession of encrypted data to which they claimed to have

forgotten their passwords. That protected data and the imagery contained in it remains unintelligible.

Mr A was convicted of attempting to procure a child aged 10 for sex and sentenced to three years imprisonment. He was in possession of encrypted files that remain unintelligible.

Mr B was suspected of possession of indecent images. He was found to be in possession of 27 encrypted disks, none of which could be opened.

Two individuals possessed a set of encrypted disks. Only a few of these could be accessed. They were sentenced on the basis of these. The rest remain unopened.

There are also a number of examples in Denning and Baugh (1999) of the use of cryptography in cases involving criminal activity and terrorism. Those described below specifically describe where investigations have been “derailed” by the use of encryption.

At one university, the investigation of a professor thought to be trafficking in child pornography was aborted because the campus police could not decrypt his files.

An employee of a company copied proprietary software to a floppy disk, took the disk home, and then stored the file on his computer encrypted under PGP. Evidently, his intention was to use the software to offer competing services, which were valued at tens of millions of dollars annually (the software itself cost over \$1 million to develop). At the time we heard about the case, the authorities had not determined the passphrase needed to decrypt the files. Information contained in logs had led them to suspect the file was the pilfered software.

At Senate hearings in September 1997, Jeffery Herig, special agent with the Florida Department of Law Enforcement, testified that they were unable to access protected files within a personal finance program in an embezzlement case at Florida State University. He said the files could possibly hold useful information concerning the location of the embezzled funds.

[It is] also reported that they had encountered unbreakable encryption in a US customs case involving an illegal, world-wide advanced fee scheme. At least 300 victims were allegedly bilked out of over \$60 million. Herig said they had encountered three different encryption systems. Although they were able to defeat the first two, they were unsuccessful with the third. The vendor told them that there were no back doors.

As these cases show, stored data that is encrypted may not be accessible to an investigator. The following section describes methods that can be used when attempting to gain access to encrypted evidence in the course of a digital investigation.

2.3.3 Addressing the Problem of Encrypted Evidence

This section discusses possible approaches that can be used by those carrying out traditional digital investigations to attempt to gain access to encrypted evidence. Countermeasures to each approach are described in order to demonstrate the difficulties and limitations.

Persuade or force the suspect to hand over their keys

“The simplest and easiest method of overcoming encryption is to ask the suspect for the password(s)” (Craiger *et al.*, 2005). As a result, any interview process should include asking the suspect for any passwords and encryption keys that are needed to access their system (Wolfe, 2003) as suspects may co-operate “as part of a plea bargain” (Denning and Baugh, 1999).

However, many suspects may decide not to reveal their passwords or could claim to have forgotten them (Barrett, 2005) and as a result in the UK, an offence has been created for failing to provide access to protected electronic information.⁹ This is described in Part III of the *Regulation of Investigatory Powers Act (2000) (RIPA)* which makes it a criminal offence not to provide decrypted versions of encrypted files or the keys¹⁰. Legislation such as this may have limited effectiveness since many of the crimes that could be concealed by encryption carry longer sentences than refusing to disclose encryption keys: the maximum sentence for which in cases of national security is five years, or two years in other cases. Furthermore, technical means such as duress keys can be used whereby two keys can be used to decrypt data; one will reveal the true content, whereas the second ‘duress’ key reveals some prearranged innocent content.

Locate unencrypted copies of the encrypted data

During the encryption process, if the original data is deleted rather than wiped it may be possible to recover parts of the original copy of a now encrypted file (Zimmermann, 1998 p.159). More subtly, an encrypted file may have been written to disk during memory swapping operations, backed up to another media or stored temporarily on the disk in an unencrypted form while being processed (Casey, 2002b). This is based on the premise that data cannot be processed while it is encrypted so must exist in a plaintext form to be manipulated in any complex way (Denning, 1999 p.309).

The success of this approach depends on the availability of locations in which unencrypted copies of data may be stored. As will be shown in Chapter 4, different

⁹ Barrett (2005) discusses legislation such as RIPA Part III as one of a number of policy options, and concludes that it is the best solution rather than “outlaw the use of strong encryption”, “[allow] only those forms of encryption which are sufficiently weak or are implemented with backdoors so as to allow law enforcement to gain access”, “allow strong encryption but require that pass-phrases (or the keys themselves) be lodged with some central, trusted escrow agent”.

¹⁰ Part III was originally not activated but came into force on 1st October 2007 under Regulation of Investigatory Powers Act 2000 (Commencement No. 4) Order 2007 and was used in a recent case *R. v S & A* [2008] EWCA Crim 2177

categories of encryption software can affect this, with some categories making the success of this approach unlikely.

Locate copies of the key or passphrase on the disk or in the surrounding area

An alternative to searching for the original data on the system is to attempt to locate the keys or passphrase on the disk or in the surrounding physical area. The success of this relies on the suspect recording the password somewhere to avoid forgetting it, encryption software writing the key to RAM which is then written to disk (Craiger *et al.*, 2005) or the key being written to a temporary file in some other way. Automated tools such as the *Forensic Toolkit (FTK)* from Access Data (2008a) can generate a full list of all keywords on a suspect disk which can be imported into the *Password Recovery Toolkit* (Access Data, 2008b) which will try all the extracted keywords as passwords for encrypted data. Using this approach “if the user purposefully or unintentionally stored their pass phrase on disk or an application wrote the pass phrase to disk, it will be available in the keyword list” (Casey, 2002b). Also, users often use the same password for several accounts (Craiger *et al.*, 2005) which may increase the chances that a residual copy could be found on the system and also “it may be useful for an agency to attempt to (legally) break passwords for other accounts to which the suspect has access to determine if the suspect uses a guessable password” (Craiger *et al.*, 2005). Keys can also be backed up to various media. For example, *BitLocker* recovery keys can be displayed on screen, printed, saved to a USB drive or any other folder (Microsoft, 2006b) and locating these would allow investigators access to encrypted data since they are provided for the purpose of recovering encrypted data in case a user loses their USB key or forgets their PIN (Microsoft, 2006a).

Therefore, in terms of countermeasures and precautions that a suspect could take, if large proportions of the disk have been encrypted and are inaccessible, it is less likely that copies of the passphrase or key will be found on the disk. If this is the case and keys have not been written down or backed up to insecure media then this approach is unlikely to be successful.

Intelligent password attacks

This approach is based on the theory that “most people do not construct their keys in a way that makes them difficult to guess. Their main concern is being able to remember the keys themselves” (Wolfe, 2002). Users will often use passwords that have personal meaning since these are the easiest to remember, e.g. birthdays, anniversaries, names of children or pets etc. (Craiger *et al.*, 2005). Automated tools can be used which will try common passphrases and passwords derived from a suspect’s personal details gathered during the investigation. These tools can also use standard and customised dictionaries in different languages. Many can be downloaded from the Internet that are specific to the user’s interests, e.g. sports teams, characters from TV, film and literature etc. (Craiger *et al.*, 2005) These passwords can also be tried in various combinations and permutations (Casey, 2002b).

Careful selection of passwords and passphrases will defeat intelligent password attacks and there is a great deal of literature on selecting appropriate hard to guess passwords such as Keith *et al.* (2006) which suggests not using dictionary words, or indeed anything that would be in any precompiled dictionaries and to make the password as long as possible. There are also encryption solutions that offer the facility to avoid using passwords or to supplement them with multi-factor authentication. For example, *TrueCrypt* allows the use of ‘key files’ where one or more files’ content is processed and combined with the user’s password to produce a key.

Exhaustive key search

It is possible to use automated tools to try all possible keys in an attempt to recover encrypted data. However, “as strong encryption becomes more widely used by criminals, it is infeasible to attack the encryption directly using brute force methods” (Casey, 2002b) and other methods should be attempted before resorting to a brute force approach (Casey, 2004a p.270). For example, the key size of the Advanced Encryption Standard (AES) is up to 256 bits, giving 1.16×10^{77} possible keys. Since no recent information could be found on time estimates for brute forcing modern

algorithms, a simple test was conducted and a brute force tool was developed with no optimisation and was capable of testing 267,000 keys per second¹¹, meaning that to try all possible keys would take 1.38×10^{64} years. Therefore, if a suitably large key has been used then the chance of the keys being identified in a time that is useful is extremely unlikely.

Vulnerabilities in an implementation

It may be possible to use vulnerabilities in a particular encryption product implementation to recover information from an encrypted file. Some products contain 'back doors' which allow the vendors to assist users in recovering data if their keys are lost (Wolfe, 2003). In the appeal of *United States v Hersh* [2002], the court heard that F-Secure provided law enforcement with partial source code allowing an encrypted container to be partially interpreted so that the names of the encrypted files were visible. In this case the names were consistent with the names of known child abuse images.

If the suspect has used Open Source Software, provided it is up-to-date, then the chances of there being undiscovered, unfixed vulnerabilities are much lower (Raymond, 2001 p.31). The use of Open Source Software also means that any deliberate code introduced into a piece of software to provide a 'backdoor', would be public and the software would either be avoided or fixed.

¹¹ The brute forcing program was developed in the course of this research in C and tried 28,836,257 keys in 1 minute 48 seconds on Windows XP, SP3 on an Intel Core 2, 1.86 GHz with 2GB of RAM.

Cryptanalysis

Cryptanalysis is described as “the science of recovering the plaintext of a message without access to the key” (Schneier, 1996 p.5). Schneier (1996 p.5) describes six general types of cryptanalysis which are summarised below.

Cipher text only: the cryptanalyst has access to several cipher texts of different plaintexts that have been encrypted with the same algorithm and key.

Known plaintext: the cryptanalyst has access to the cipher text of several messages and also the original plaintexts.

Chosen plaintext: The cryptanalyst is able to choose the plaintext that gets encrypted.

Adaptive chosen plaintext: The cryptanalyst is able to choose the plaintext that gets encrypted and then submit further plaintexts for encryption based on previous results.

Chosen cipher text: The cryptanalyst can choose different cipher texts to be decrypted and then has access to the plaintexts.

Chosen key: The cryptanalyst chooses a relationship between a pair of keys, but does not know the keys themselves. The same plaintext is encrypted with both keys.

Most current encryption products use public algorithms that have been, and are subjected to extensive research and scrutiny. For this reason, in this research, the encryption algorithms used are considered to be secure and cryptanalysis is not considered a viable option for data recovery.

Surveillance

Either hardware or software surveillance techniques can be used to monitor a system in order to record the pass phrase that allows access to encrypted data. This can be used when a suspect is highly unlikely to co-operate.

Software Surveillance

Software surveillance techniques include key loggers and screen scrapers which record keyboard entry or the output of a graphics card respectively. These can be used to capture passwords or record the values in drop down menu based pass-code entry. Software based surveillance can capture information only after they have been launched. Since most are installed at the operating system level they are unable to capture passwords entered early on in the boot process (Wolfe, 2002).

Hardware Surveillance

Hardware key loggers are physical devices that sit between the computer and the keyboard and will capture any keyboard entry no matter what state the system is in (Wolfe, 2002). The captured passwords can either be stored in the device or transmitted on a designated radio frequency (Wolfe, 2002). Such devices can be obtained easily online or even high street shops for under £50 (Maplins, 2008).

Wolfe (2002) also points out that surveillance can take place before or after the initial seizure. The example is given of the former is *United States v Scarfo* [2001] where a key logger was installed on his machine covertly before the seizure in order to capture encryption passwords. In the latter, the machine is seized, imaged and then returned to the suspect with an installed key logger. In an unspecified case: “within three hours of returning his machine, the authorities had the needed keys and were then able to unlock the evidentiary copy of the encrypted hard disk” (Wolfe, 2002).

In the post-seizure case, surveillance could be defeated by a suspect who is aware not to enter their passwords after law enforcement has had unrestricted access to their machine. In the pre-seizure case hardware surveillance could be countered by exercising vigilance for suspicious devices attached to the system. Software surveillance could be detected by monitoring for suspicious processes and using both keyboard and mouse entry for pass phrase entry. The use of multi-factor authentication may also reduce the effectiveness of the surveillance approach.

2.3.4 Summary

As described in previous sections, the use of encryption by a suspect could present difficulties for those conducting digital investigations. Also the use of encryption is believed to be increasing. As shown in the previous section, there are a number of approaches for attempting to gain access to encrypted digital evidence, but there are countermeasures for each of the approaches.

Another option available in some cases that was not covered in the previously discussed approaches is the use of 'live investigations'. Live digital investigations are discussed in detail in the following section.

2.4 LIVE DIGITAL INVESTIGATIONS

2.4.1 Introduction

This section defines live digital investigations. This is followed by examples of situations where live investigations are currently used and specific techniques are discussed in detail, including live acquisition and live analysis. In addition, when discussing analysis techniques, the nature of volatile memory and *Windows* operating system memory management are described.

2.4.2 Defining Live Digital Investigation

Section 2.2.3 described the traditional approach to a digital investigation, where the power is removed from a system in order to preserve the contents of the disk, but this

is at the expense of live memory. However, as previous sections have shown, some of that information located in memory could be useful or even essential for an investigation.

As described in Vidas (2007), “upon arriving on scene, a responder has two core choices, either interact with the system, or pull the plug”. A live investigation involves interacting with the system and takes into account the potential usefulness of volatile data that would be lost due to the ‘pull the plug’ approach. There are a number of definitions that highlight the difference between a live investigation and the traditional pull the plug approach (sometimes referred to as a ‘dead’ investigation, to complement a ‘live’ investigation).

According to Carrier (2005 p.5), “A dead analysis occurs when you are running trusted applications in a trusted operating system to find evidence” and a live analysis is defined as “when you use the operating system or other resources of the system being investigated to find evidence.” However, in some cases this second definition conflicts with the first. In the case of bootable *Linux* CDs, the hardware of the suspect system is used, i.e. the resources of the system being investigated, but the investigation is also being performed in a trusted operating system using trusted applications, so from these definitions it is unclear where bootable *Linux* CDs would fit.

Another definition from Mandia *et al.* (2003 p.27) simply states that “a live response is conducted when a computer system is still powered on and running.” This would suggest that a bootable *Linux* CD is indeed a live investigation.

Carrier (2006b) offers another explanation of the terms ‘live’ and ‘dead’ analysis. Here “live analysis techniques use software that existed on the system during the time frame being investigated.” Carrier comments that using this definition, due to the fact that many hardware devices contain software (hard disk firmware), even imaging a disk on a trusted analysis machine would constitute a live investigation. In this research, the software embedded in hard disk controllers is ignored and disk imaging is not treated as a live investigation. Therefore, if the use of hardware and the software embedded within it is not considered to be a live investigation, since these

are the ‘other resources’ that are used by a bootable *Linux* CD, it does not constitute a live investigation either. As a result, a modified definition of *using the operating system of the system being investigated to acquire, analyse or present digital evidence*¹² will be used to describe a live investigation.

2.4.3 Current Live Investigation Techniques

This section describes current live investigation techniques. Considering the high level view of the digital investigation process discussed in Section 2.2.4 (acquisition, analysis and presentation), first, tools are discussed that are referred to as ‘live investigation tools’, since they perform multiple stages of a digital investigation, acquiring and analysing data on the live machine. Following this, live acquisition tools are discussed, which just acquire data¹³. Live acquisition tools are discussed in terms of live disk acquisition and live memory acquisition. Also discussed are memory analysis tools, which are not usually run on the live system and are therefore not live tools, but they are important for analysing live acquired data from memory. However, analysis of live acquired disk images is not discussed as it uses the same analysis techniques as a standard digital investigation.

Live Investigation Tools

There are a number of live investigation tools that can be used to acquire and analyse information on a live system. The methodology for running such tools is best described in Wait (2008). This involves:

- 1) establishing a trusted command prompt,
- 2) establishing a method for transmitting and storing the collected information,
- 3) running various tools and creating hashes of the output.

¹² Note the removal of ‘or other resources’ from the Carrier (2005) definition and the substitution of ‘find evidence’ with ‘acquire, analyse and present digital evidence’. This is to reflect the high level process model of acquisition, analysis and presentation used to describe a digital investigation.

¹³ Live presentation tools are not discussed in this research as it is not a common term. However, by the definition used in this research, using VMware to boot a copy of a suspect’s machine to present as evidence in court would constitute a live presentation.

There are many tools that can be used in this way to gather information about a live system's configuration and they are often grouped together into toolkits. They can also be scripted and written to CD so that the same tools are run in the same order for each investigation, e.g. *FirstOnScene.vbs* (Monday, 2004). Mandia *et al.* (2003, p.97, p.127) describes the live investigation tools for both *Windows* and *Unix* that should form the basis of any incident response toolkit. Table 1 shows the tools that are described as being essential for a *Windows* incident response toolkit.

Tool	Description	Source
cmd.exe	A trusted copy of the command prompt. This will change for different versions of <i>Windows</i> .	Built in
PsLoggedOn	A utility that shows all users connected locally and remotely.	www.foundstone.com
Rasusers	A command that shows which users have remote-access privileges on the target system.	NT Resource Kit (NTRK)
Netstat	A system tool that enumerates all listening ports and all current connections to those ports.	Built in
Fport	A utility that enumerates all processes that opened any TCP/IP ports on a <i>Windows NT/2000</i> system.	www.foundstone.com
PsList	A utility that enumerates all running processes on the target system.	www.foundstone.com
ListDLLs	A utility that lists all running processes, their command line arguments, and the dynamically linked libraries (DLLs) on which each process depends.	www.foundstone.com
Nbtstat	A utility that lists the recent NetBIOS connections for approximately the last 10 minutes.	Built in
Arp	A system tool that shows the MAC addresses of systems that the target system has been communicating with, within the last minute.	Built in
Kill	A command that terminates a process.	NTRK
Md5sum	A utility that creates MD5 hashes for a given file.	www.cygwin.com
rmtshare	A command that displays the shares accessible on a remote machine.	NTRK

Table 1: Live tools necessary to investigate a *Windows* system (Mandia et al. 2003, p.97)

One of the most popular collections of incident response tools is the *Helix Live CD* (e-fense, 2008). This contains the tools described earlier and also specific toolkits that package together the tools. *Helix 1.9a* contains the following toolkits: *Windows Forensic Toolchest* (WFT), *First Responder's Utilities* (FRU), *Incident Response Collection Report*, *Agile Risk Management's Nigilant32* which run the tools listed in Table 2.1 and various others.

Live Acquisition: Disk

The tools described above report specific information about a system, whereas disk acquisitions are traditionally bit stream copies of the entire drive of a system. However, Turner (2006) discusses alternatives to *full disk acquisition* and describes that there are multiple selective acquisition techniques: *manual selective acquisition* (the investigator chooses individual files for acquisition), *semi-automatic selective imaging* (investigator decides file types to acquire), *automatic selective imaging* (investigator selects only source and destination and evidence is automatically acquired according to pre-configured parameters related to the investigation). These will be discussed further in Section 4.2.

There are a number of tools that can be used to acquire data from the disk of a live system. For example, on the *Helix Live CD* (e-fense, 2008), *dd* and *FTK Imager* are supplied. *dd* is a command line tool that can be used to acquire physical and logical drives. The version on *Helix* is part of George Garner's Forensic Acquisition Utilities (FAU) (Garner, 2007). *dd* can be invoked using commands such as those shown below, which acquire the entire physical drive, and the C:\ partition respectively.

```
dd if=\\.\PhysicalDrive0 of=E:\physical.dd
dd if=C:\ of=E:\logical.dd
```

FTK Imager is a tool from Access Data (Access Data, 2007) that provides a Graphical User Interface for acquiring physical drives, logical drives and the contents of folders, as shown in Figure 1.

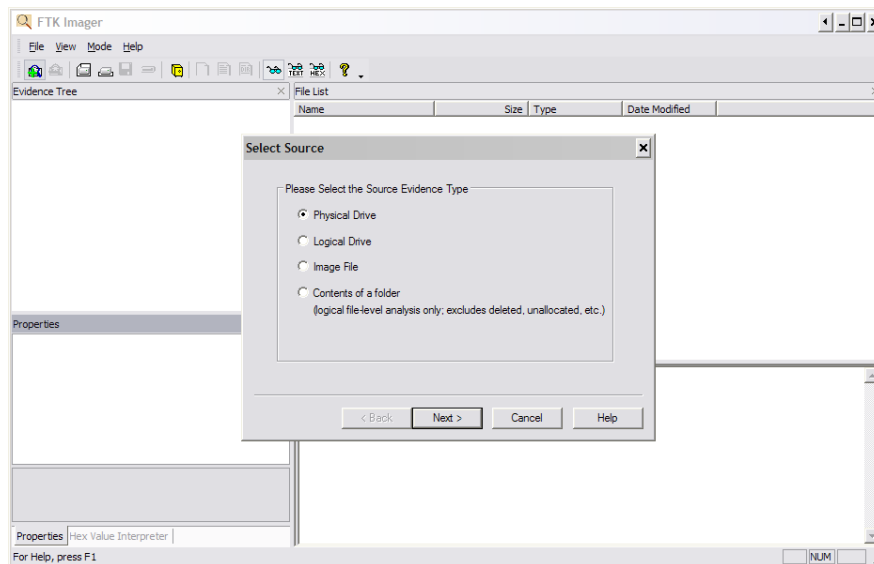


Figure 1: Screenshot of *FTK Imager*

There are also other versions of *dd* with purposes specifically described as forensic acquisition, for example *dcfldd* (US Department of Defence Computer Forensic Lab Version) (Harbour, 2006). However, no tool currently implements the Semi-automatic or Automatic Selective imaging techniques discussed in Turner (2006). These tools can also be used to image virtual file systems, but not mounted network drives (unless they are copied as ‘contents of a folder’ using *FTK Imager*).

It is also possible to use these live disk acquisition tools to obtain decrypted copies of data that would otherwise be encrypted. Due to the nature of a live investigation (‘uses the operating system of the system under investigation’), if encrypted data is available to the operating system then it will also be available to live investigation tools and can be copied to external media.

Live Acquisition: Memory

In addition to live acquisition of disks, it is also possible to acquire the live memory of a system. There are a number of techniques for achieving this:

\\.\PhysicalMemory (user mode): Similar to `\dev\memory` in *Linux*, this object provides access to the physical memory of *Windows XP* (Vidstrom, 2006a). This can be accessed and copied using user-mode programs such as a modified version of *dd* (Carvey, 2007b). This has the advantage that no software needs to be installed on the system under investigation (Schuster, 2005). However, the main practical problem with this approach is that it requires administrator privileges on the live suspect machine (Schuster, 2005). User mode access to the `\\.\PhysicalMemory` object is also not possible unavailable under *Windows Server 2003 SP1* onwards, including *Windows Vista* (Schuster, 2005).

\\.\PhysicalMemory (kernel mode): Recently a number of options have become available that overcome the problem of lack of user mode access to the `\\.\PhysicalMemory` object. These allow imaging of the memory of a *Windows Vista* machine using a kernel mode driver to access the `\\.\PhysicalMemory` object (Schuster, 2008b). There are a number of implementations of this (Schuster, 2008b):

WinEn: This is included with *EnCase* versions 6.11 onward. It is also included on the latest version (2.0) of the *Helix Live CD* (e-fense, 2008).

mdd (Stotts, 2008): The *Memory DD* tool from ManTech is open source and available on SourceForge (ManTech, 2008).

win32dd (Suiche, 2008b): This tool is also open source but uses more kernel mode functions, including writing the output file, rather than *mdd*, in which “the [kernel] driver is only used to get `\Device\PhysicalMemory` handle (Suiche, 2008a).

Firewire (IEEE 1394): Firewire devices use Direct Memory Access (DMA) meaning they can access the memory of a system without using the CPU (Carvey, 2007b). Bolieau (undated) describes a way to use this property to obtain an image of a machine's physical memory. This approach allows imaging of memory, even if a machine is locked. However, the technique is more difficult to configure and use than the `\\.\PhysicalMemory` tools and the target system must have a working Firewire port. There are also documented problems in Vidstrom (2006b), e.g. dumping the Upper Memory Area (UMA) and it can cause a fatal error and blue screen if non-existent memory addresses are accessed (Schuster, 2008a).

Process Memory Acquisition Tools: In addition to the acquisition of full memory dumps there are also software tools that can dump the memory used by a specific process. Examples of such tools include *pmdump* (Vidstrom, 2002) and *userdump* (Microsoft, 2007c).

Cooling + Reboot: This approach is described in Halderman *et al* (2008) and explains that even though data in RAM does decay when the power is removed, "retention times can be increased by cooling" (Halderman *et al.*, 2008). By cooling RAM chips to -50°C using an inverted can of compressed air and using a warm or cold reboot, the bit deterioration may be reduced sufficiently so that by rebooting the system to a custom operating system with a minimal memory footprint (network based or on USB) the contents of RAM can still be imaged, albeit with some bit errors. This has the advantage of providing a trusted operating system in which to perform imaging. At time of writing the tool from Halderman *et al* (2008) (*ram2usb*) was not available but an alternative that uses the same principles is available from McGrew (2008). There are also additional problems to the bit errors: it is possible that the machine has been configured not to boot to network or USB, preventing an operating system from being loaded that can perform the memory imaging. It is also possible that the machine may perform a destructive memory test when restarting. As a result, there are

a number of variables that affect whether data from memory can be recovered using this technique and at present this remains an experimental approach.

Cooling + Physical Removal of RAM Chips: This approach is also described in Halderman *et al.* (2008) and uses the same cooling approach. At -50°C data is described to persist for several minutes, allowing the RAM chips to be physically removed from the system and placed in a second machine. This second system would be booted to an operating system with a minimal memory footprint which may allow previous memory contents to be acquired. This solves the problem of the suspect machine not booting to network or USB and a system can be used that does not perform a destructive memory test. However, it would require a compatible system in which to place the RAM chips.

Crash Dumps: A system can be configured in advance through the *Windows* Registry or Start-up and Recovery settings to create a full dump of its memory to disk on a key press (on PS/2 keyboards¹⁴) (Microsoft, 2007e). This has the significant advantage that the entire system is halted when the contents of RAM are being written (Carvey, 2007b), this means that this is a true ‘image’ of memory rather than a ‘smear’, since the data is not constantly changing as it is being copied. However, it is necessary to reboot the system for the Registry change to take effect (Microsoft, 2007e) and as a result is unlikely to be practically of use since systems are unlikely to be found in this configuration. There is also a further limitation described in Huebner *et al.* (2007) that “the key sequence used to generate the crash dump is insecure and could be intercepted by an application program”.

Hibernation File: When a *Windows* system is put into hibernate mode, the system’s state is stored in hiberfil.sys file. Using the *Sandman Framework* the hibernation file can be converted to a flat, *dd* style image (Suiche and Ruff, 2008). Using the

¹⁴ It can also work with USB keyboards but only on *Windows Server 2003* and it is necessary to install a hotfix for Kbdhid.sys driver

hibernation file to obtain an image offers the significant advantage that the system is completely stopped, so the acquired memory image is completely coherent and does not suffer the same ‘smearing’ as other techniques. Since the hibernation file is examined offline, it is not possible for malware to hide from an analysis. It is also counterproductive for malware to prevent itself from being written to the hibernation file as the malware would then not resume running when the system restarts from the hibernation file

However, if the system’s power is disconnected and the hibernation file later imaged, the hibernation file may be significantly out of date; alternatively, the investigator intentionally putting the system to sleep will overwrite data in the current hibernation file on the disk. Ruff and Suiche (2007) also states that there is “no guarantee that 100% of physical memory has been saved”. Also, as will be seen in Chapter 4, the hibernation file is not available when certain types of encryption product are in use.

Hardware Devices: Carrier and Grand (2004) describes a PCI card that can be fitted to a PC which can dump memory to an external storage device. Since this approach is hardware based it does not rely on potentially untrusted code. However, the hardware needs to be installed before an incident occurs (Carvey, 2007b) and as a result this is unlikely to be an option.

Memory Image Analysis: Introduction to Memory and Memory Structures

While live investigation tools described earlier can be used on a live machine to acquire, analyse and present information about the system, more recent approaches to live investigations separate out the acquisition of data from live systems and the analysis and presentation of the information (Walters and Petroni, 2007). The state of the art of memory analysis techniques is discussed in the next section and this section provides the necessary background. This section describes the nature of Random Access Memory (RAM) in physical terms and also in terms of how data is organised logically by the *Windows* operating system.

Physical:

Random Access Memory (RAM) has two main types, static and dynamic. Dynamic RAM, which is used in modern computers as the main temporary storage consists of a collection of ‘cells’, where each cell contains a transistor and capacitor (Tanenbaum, 1999 p.152). These capacitors charge and discharge, representing the binary values of one and zero. The capacitors will discharge naturally and in order to maintain a state of one, each must be refreshed every few milliseconds (Tanenbaum, 1999 p.152). It is for this reason that when the power is disconnected from a system the data in memory is lost.

Logical Organisation:

Computers are “electronic devices capable of storing and processing information in accordance with a predetermined set of instructions” (Oxford, 2008). These predetermined set of instructions are referred to as ‘programs’ and when they are run on a computer they are organised in memory as processes (Russinovich and Solomon, 2005 p.6). Each process is a “container for a set of resources used when executing the instance of the program” (Russinovich and Solomon, 2005 p.6) and each is assigned memory in which to execute and store data.¹⁵

Processes do not generally access addresses in physical memory directly. Instead, each process is assigned its own ‘virtual memory’ space. This provides each process with “the illusion of having its own large, private address space” (Russinovich and Solomon, 2005 p.14) and allows the operating system to control and protect memory locations, ensuring that processes do not overwrite the operating system or each other’s data. When accessed, these virtual addresses are converted into the physical addresses of the computer’s memory by the system’s memory manager. The virtual address space is divided into blocks that are referred to as ‘pages’ (default size

¹⁵ Processes do not actually run, since only threads can run (Florio 2005). Processes contain one or more threads which are scheduled and executed by the system. However, processes are discussed in this section on memory structures since all threads of a process shares the process’s virtual address space (Russinovich and Solomon, 2005 p.13).

4k) and the references used to map virtual addresses to physical memory addresses are stored in Page Tables (Russinovich and Solomon, 2005 p.425).

These virtual addresses may map not only to physical memory, but can also reference data stored on disk in ‘page files’. These page files exist since “most systems have much less physical memory than the total virtual memory in use by the running processes” (Russinovich and Solomon, 2005 p.15). To overcome this problem, pages from memory that are not currently in use can be written to page files stored on disk which frees up pages in memory for processes currently executing. Stored data in page files can be paged back into memory when needed.

Each process is represented in memory by an Executive Process or EPROCESS block (Russinovich and Solomon, 2005 p.289). An EPROCESS block contains a number of pieces of information and also pointers to other data structures; full details are provided in Maclean (2006) and Russinovich and Solomon (2005 p.291-293). A summary of important offsets in the EPROCESS block that are referenced in this section is shown in Figure 2.

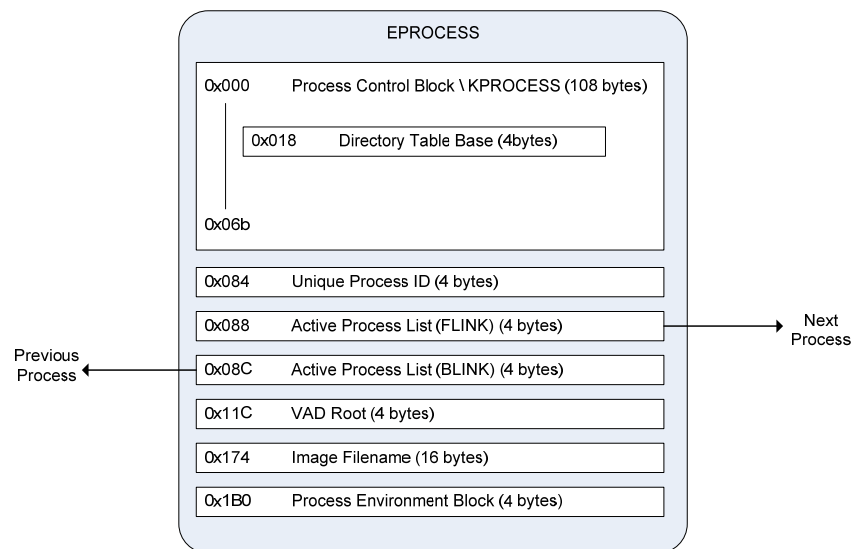


Figure 2: Simplified structure of an EPROCESS block describing locations of important information in *Windows XP SP2*.

As shown in Figure 2, the EPROCESS block contains various pieces of information about the process, including the Process ID and links to the previous and next process (a double linked list (Florio, 2005)). The first part of the EPROCESS block contains a

sub-structure, a Kernel Process (KPROCESS) block or Process Control Block (PCB). This also contains information about the process, including a pointer to the start of the process's Page Directory (Russinovich and Solomon, 2005 p.428).

Each process has a single Page Directory, which keeps track of all Page Tables for that process. The Page Directory contains Page Directory Entries (PDEs) which point to the locations of Page Tables for the process (Russinovich and Solomon, 2005 p.428). Page Tables then contain Page Table Entries (PTEs) which point to the correct page in physical memory. A 32-bit virtual memory address therefore has three separate components:

- 1) The Page Directory Index (10 bits) which finds the Page Directory Entry that points to the correct Page Table.
- 2) The Page Table Index (10 bits), which locates the correct Page Table Entry which points to the desired page in physical memory.
- 3) The Byte Index (12 bits), which points to the desired byte in the selected page in physical memory.

This is shown diagrammatically in Russinovich and Solomon (2005 p.427) and a simplified version is shown in Figure 3.

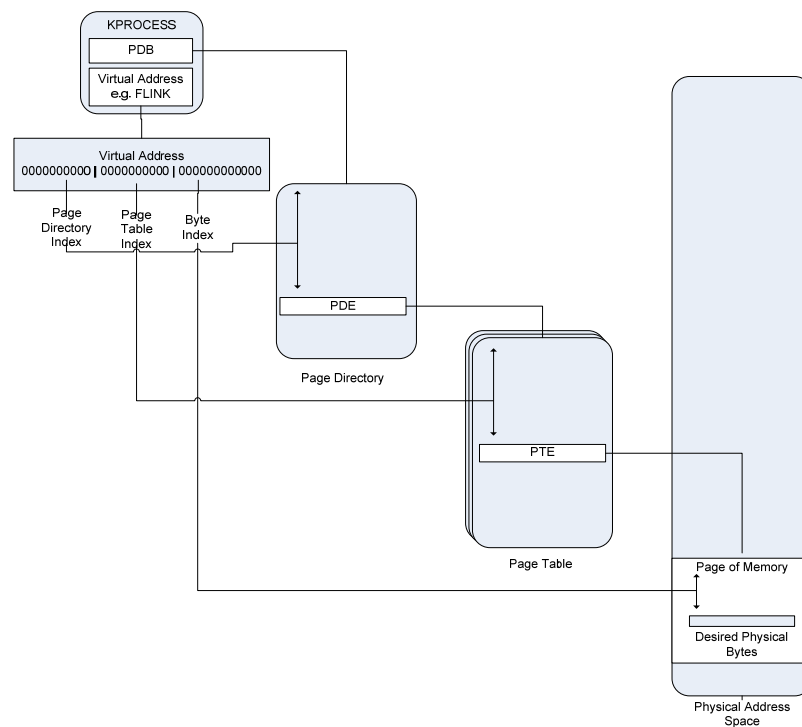


Figure 3: The relationship between Page Directory, Page Tables, and Pages of physical memory.

In addition to the Page Tables that keep track of a process's virtual memory, there is also another data structure that has a similar function. The **EPROCESS** block also contains references to a set of **Virtual Address Descriptors (VADs)** which keep track of "which virtual addresses have been reserved in the process's address space and which have not" (Russovich and Solomon, 2005 p.448). These are used to improve performance of the system by avoiding constructing Page Tables for allocated memory until the pages are accessed and a page fault occurs. Then the VADs are used to look up the accessed address range and to create a Page Table Entry.

Memory Image Analysis: Techniques

Recent approaches to live investigations separate the acquisition of data from live systems from the analysis of that data (Walters and Petroni, 2007). This involves a memory acquisition of the live system, followed by an offline analysis of the memory dump to recover information from it. This section describes some of the memory analysis techniques that are currently available.

String searches:

Early analyses of memory dumps consisted of simply extracting text strings (Carvey, 2007b p.88). This is achieved using tools such as *strings* (Russovich, 2007), *grep* (on *Linux*) or *bintext* (Foundstone, 2000) and enables searches for passwords, IP and e-mail addresses and other text strings. The difficulty with evidence obtained in this way is that it is difficult to attribute to a specific process (Carvey, 2007b p.89).

Process Enumeration:

As described in the previous section, processes are linked to each other by a double linked list. Therefore if one EPROCESS block can be found in memory, the Forward Link (FLINK) and Backwards Link (BLINK) pointers can be used to enumerate all processes in the memory dump.

This approach relies on locating an EPROCESS block. There are a number of methods described for achieving this. Burdach (2005) explains how to find process blocks by searching for two processes that link to each other. Burdach (2005) states that two processes that link to each other are *smss* and *csrss*¹⁶. These are found using a simple string search, checking for one which has a link to the other. However, when this was tested, more than 100 references to these strings were found making cross-checking difficult. Maclean (2006) describes an alternative approach, where the 'system' process's Page Directory Base is consistent whenever *Windows* boots, pointing to 0x00039000. However, in experiments conducted as part of this research and as shown in Figure 4, this was not always the case¹⁷.

¹⁶ smss – session manager subsystem, csrss – client server runtime server subsystem

¹⁷ The test was conducted on a Windows XP, SP2 system using VMware. The PDB at offset 0x18 points to 0x00319000

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
01BCC830	03	00	1B	00	00	00	00	00	38	C8	9C	81	38	C8	9C	81	8È 8È
01BCC840	40	C8	9C	81	40	C8	9C	81	00	90	31	00	00	00	00	00	@È @È 11
01BCC850	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
01BCC860	AC	20	00	00	00	00	00	00	51	02	00	00	00	00	00	00	~ Q
01BCC870	70	C8	9C	81	70	C8	9C	81	00	00	00	00	00	00	00	00	pÈ pÈ
01BCC880	68	C7	9C	81	58	EF	66	81	00	00	00	00	01	00	00	00	hÇ Xif
01BCC890	36	00	08	06	00	00	00	00	00	00	00	00	00	00	00	00	6
01BCC8A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
01BCC8B0	00	00	00	00	04	00	00	00	28	FE	7D	81	58	92	55	80	(b) X`U
01BCC8C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
01BCC8D0	00	00	00	00	00	00	00	00	07	00	00	00	00	70	29	00	p)
01BCC8E0	00	50	1D	00	00	00	00	00	00	00	00	00	00	00	00	00	P
01BCC8F0	00	00	00	00	C0	0C	00	E1	ED	17	00	E1	01	00	00	00	À áí á
01BCC900	54	B6	DF	F9	00	00	00	00	01	00	04	00	00	00	00	00	T#Bù
01BCC910	10	C9	9C	81	10	C9	9C	81	00	00	00	00	00	00	00	00	È È
01BCC920	01	00	00	00	58	B6	DF	F9	00	00	00	00	01	00	04	00	X#Bù
01BCC930	00	00	00	00	34	C9	9C	81	34	C9	9C	81	00	00	00	00	4È 4È
01BCC940	00	00	00	00	00	00	00	00	00	00	00	00	C0	C2	9C	81	ÀÀ
01BCC950	C0	C2	9C	81	00	00	00	00	03	00	00	00	00	00	00	00	ÀÀ
01BCC960	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
01BCC970	00	93	55	80	00	00	00	00	00	00	00	00	00	00	00	00	U
01BCC980	00	00	00	00	00	00	00	00	00	00	00	00	A8	10	00	E1	.. á
01BCC990	90	C9	9C	81	90	C9	9C	81	00	00	00	00	00	00	00	00	È È È
01BCC9A0	00	00	00	00	53	79	73	74	65	6D	00	00	00	00	00	00	System
01BCC9B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Figure 4: PDB at offset 0x18 of the system process does not point to 0x00039000 but to 00319000

An alternative approach exploits that the *system* process usually has a Process ID of 4 (at offset 0x84 of the EPROCESS block). Using a simple *Perl* script a memory dump can be scanned for this pattern and the ‘System’ EPROCESS block located, as shown in Figure 5.

```

C:\WINDOWS\system32\cmd.exe
%:\My Documents\Development\Memory Find System Process>perl FindSystemProcess.pl "example 1\memory.d
d"
I think the 'System' EPROCESS is at offset 29149640 (0x1bcc9c8), PDB: 00294000
%:\My Documents\Development\Memory Find System Process>_

```

/\x04[\x00-\xFF]ff\x53\x79\x73\x74\x65\x6d\x00\x00/gsm

Figure 5: Regular expression used in *Perl* script to locate ‘System’ process with command line output showing found system process.

Once a process is identified, all others can be enumerated from the EPROCESS FLINKs and BLINKs. However, these are virtual addresses, and as described in the previous section, need to be translated to the real physical addresses. There are also further complications in that data may have been paged out to disk. In this case the virtual memory address will point to an address in one of the pagefiles. This is described in detail in Kornblum (2007).

However, there are limitations to this enumeration approach. Processes that have ended will be unlinked from the list and while information in the EPROCESS blocks may still be available in memory, it will not be recoverable using this approach (Schuster, 2006). Also, processes could be deliberately unlinked from this list and would not appear in a list of enumerated processes (Vidas, 2007). This is an approach known as Direct Kernel Object Manipulation (DKOM) that can be used for hiding malware.

Process Carving:

Schuster (2006) provides an alternative to the process enumeration approach that is similar to file carving in disk images. This approach scans through a memory image testing for valid process and thread structures using a 20 rule criteria. This is implemented in *PTFinder.pl*. There is also another implementation of this approach in Carvey (2007b p.104), *lsproc.pl*, which is limited to identifying processes rather than threads.

VAD Tree Based Process Recovery:

Another useful memory analysis technique is the recovery of memory that belongs to a specific process. As discussed earlier, the VAD tree provides access to areas of memory assigned to a process. The root of the VAD tree is stored in the process's EPROCESS block at offset 0x11C. From this pointer the VAD tree can be traversed and the areas of memory assigned to the process can be extracted (Dolan-Gavitt, 2007).

Like the process enumeration approach described earlier, VAD nodes can be unlinked from the tree, since “memory reads appear to use the page directory to access memory first, and the VAD is only consulted if a page fault occurs” (Dolan-Gavitt, 2007), which will hide the nodes from an analysis such as this. The VAD tree is also only available in processes that are still running and the pointers to the VAD root are zeroed when the process exits (Dolan-Gavitt, 2007). Also, the VAD root offset has changed in *Windows Server 2003* and *Windows Vista* (Schuster, 2007).

Memory Image Analysis: Toolkits

Offline memory analysis is an extremely fast moving field and there are additional techniques that are available that have not been discussed in previous sections. These include identifying open ports, open files, recovering parts of an executable that has been run etc. These memory analysis techniques come from a variety of authors but are being combined into toolkits, meaning that an investigator does not need to rely on a collection of different tools in order to analyse memory dumps.

Responder is a commercial product that is “the industry's first live memory and runtime analysis platform for *Windows* operating systems”, (HBGary, 2008b) and allows an investigator to view the physical and virtual memory structures in a memory dump in a graphical environment.

The leading open source toolkit for memory analysis is the *Volatility Framework* which is “a completely open collection of tools, implemented in *Python* under the GNU General Public License, for the extraction of digital artefacts from volatile memory (RAM) samples” (Volatile Systems, 2008). *Volatility* allows a number of different analyses to be performed on a memory image from *Windows XP SP2* and *SP3* systems. These different analyses are shown in Table 2 (from *Volatility 1.3 Beta* help file).

Volatility command	Information Obtained
connections	Print list of open connections.
connscan	Scan for connection objects.
datetime	Get date/time information for image.
dlllist	Print list of loaded DLLs for each process.
dmp2raw	Convert a crash dump to a raw dump.
dmpchk	Dump crash dump information.
files	Print list of open files for each process.
hibinfo	Convert hibernation file to linear raw image.
ident	Identify image properties.
memdmp	Dump the addressable memory for a process.
memmap	Print the memory map.

modscan	Scan for modules.
modules	Print list of loaded modules.
procdump	Dump a process to an executable sample.
pslist	Print list of running processes.
psscan	Scan for EPROCESS objects.
raw2dmp	Convert a raw dump to a crash dump.
regobjkeys	Print list of open Registry keys for each process.
sockets	Print list of open sockets.
sockscan	Scan for socket objects.
strings	Match physical offsets to virtual addresses.
thrdscan	Scan for ETHREAD objects.
vaddump	Dump the VAD sections to files.
vadinfo	Dump the VAD info.
vadwalk	Walk the VAD tree.

Table 2: Analysis techniques that can be performed using the *Volatility* framework .

As can be seen in Table 2, the *Volatility Framework* allows much of the information obtained using live investigation tools such as *pslist* to be obtained from an acquired memory image. Separating acquisition from analysis in this way simplifies overcoming some of the challenges to live investigations described in the next section and in detail in Chapter 3.

2.4.4 Challenges to Live Digital Investigations

As previous sections have shown, there are limitations to the traditional approach to digital investigations. It has also been discussed how live investigations tools can preserve digital evidence that would otherwise be lost. However there are a number of challenges to using live digital investigations which are discussed in this section.

Trusting Results

One of the biggest difficulties in performing live investigations is the ability to trust results. Mohay *et al.* (2003) states that “any system being examined live should be considered to be hostile until proven otherwise.” Unfortunately some part of the

system will need to be trusted; at the very least, the software used to mount a CD of trusted binaries (Burdach 2004). Carrier (2006) goes as far as saying that “the only difference between live and dead analysis is the reliability of results.” While this is not the case based on the definition of live analysis used in this research, it certainly is one of the most difficult issues to address, particularly in *Windows* environments where such extensive use is made of *Dynamic Link Libraries* (Carvey 2004).

There are two main concerns regarding trust. The first is that the operating system could be modified in some way to provide false information (Kenneally and Brown 2005, Carrier 2006). This presents two possibilities: 1) a malicious root kit could be responsible for the creation of incriminatory evidence and hide all traces of itself, thus implicating an innocent user. 2) a root kit could be installed intentionally with the purpose of hiding parts of the disk from an investigator performing a live analysis. The second concern is that logic bombs, “booby traps” (Mohay *et al.* 2003, p.135) or “electronic mines” (Farmer and Venema 2004, p.5) could be placed on a system and used to destroy or corrupt evidence if triggered.

Intrusiveness of Techniques

A common concern with live investigations is that compared to a traditional investigation they are highly intrusive. Due to the inherent volatility of digital evidence it is very easy for it to become contaminated (Adelstein 2006) and the write blocking approach used in traditional forensics is not possible during live investigations (Nikkel 2005). As a result terms such as “modifying as little as possible on the system” (Carvey 2004) and “minimally invasive” (Hargreaves *et al.* 2006) are used when discussing live investigations.

Ultimately there is “no way to avoid making changes, since in order to conduct a live examination it is necessary to deploy tools on the live system to capture data, and such tools will make changes to the running system” (Sutherland *et al.*, 2008). The amount of change caused will vary, depending on hardware and software configurations (Vidas, 2007), but even when attempting one of the simplest of live responses, capturing a memory image, “no software tool is capable of capturing the

image of memory without, by the very act of its own execution, changing the content of memory” (Huebner *et al.*, 2007).

Verification of results

One of the fundamental principles of digital investigation described by Pollitt (1995) is that examination results should be verifiable and repeatable. Compared to the ease at which results can be verified in traditional forensics by repeating the same procedure on a duplicate image of the original evidence, verifying certain results from a live investigation is difficult.

As mentioned in earlier sections, recent approaches to live digital investigations separate acquisition from analysis. Once evidence has been acquired from a live system, the extracted data has the same properties as evidence obtained from a traditional investigation and can be exactly copied and any analysis techniques used can be repeated on duplicate copies by independent examiners. Therefore, the problem lies in the repeatability and verifiability of the acquisition stage of a live investigation since “the evidence gathered represents a snapshot of a dynamic system that cannot be reproduced at a later date” (Adelstein 2006). As a result the acquired image can be verified only against itself rather than the original media (Casey and Stanley 2004) which prevents the correctness of data from the acquisition stage being easily demonstrated. This could result in the challenges to the integrity of the acquired evidence, potentially affecting its weight in court or even preventing it from being admissible (Kenneally and Brown 2005).

Ensuring a Complete Set of Evidence

Another concern with live investigations is that it can involve selective file copying rather than creating a full image (Kenneally and Brown 2005). This could result in challenges claiming that a piece of digital evidence that proved innocence was not captured. Certainly putting first responders in a position where they need to identify relevant digital evidence is significantly different from their current role of identifying and preserving physical evidence upon which digital evidence may reside.

2.4.5 Summary

This section has defined a live digital investigation as *a digital investigation which uses the operating system of the system being investigated to acquire, analyse or present digital evidence*. It has reviewed a number of live investigation techniques, including the live acquisition of disks and memory. Specifically, it has discussed that if a system is encountered that contains encrypted data, then it is possible to use live acquisition techniques to acquire that data in an accessible form prior to the machine being powered down. In addition to disk acquisition techniques it has been shown that memory analysis techniques can be used in an offline or ‘dead’ environment to obtain information from memory images acquired from live systems. This means that it is possible to separate out the acquisition and analysis stages of a live investigation, where previously ‘incident response’ tools had to be used to perform both acquisition and analysis using the operating system of the system under investigation. The importance of this will be discussed later in Chapter 6. Finally, the challenges to live investigations were described which show that while the acquisition of encrypted data seems a simple solution there are a number of problems with results obtained using live techniques: the difficulty in trusting the results, its inherent intrusiveness, the difficulty in verifying results and ensuring that no evidence is missed.

2.5 CHAPTER SUMMARY

In summary, this chapter has covered a broad range of topics. The first section discussed how digital investigations are different from forensic digital investigations, which is an important distinction due to the additional requirements imposed by the latter, i.e. that evidence must be admissible in a court of law, which will be subject to localisation.

It has also been discussed that both digital investigations and their forensic counterpart are centred on the recovery of digital evidence which has been defined as *a set of reliable digital objects that support or refute a hypothesis*. The specific constraint of ‘reliable’ has been shown to be important since if digital data is used to

support or refute a hypothesis and is not reliable, then this could result in an incorrect hypothesis being supported and ultimately, an incorrect conclusion being drawn.

The traditional ‘pull the plug’ approach to digital investigations has also been discussed. This involves the power being removed from running systems at the scene, which preserves the contents of the disk, but at the expense of volatile memory. This section has also presented the challenges that digital investigations currently face and specifically discussed situations where the ‘pull the plug’ approach is inadequate. One situation in particular, the use of encryption, has been discussed in detail. It has been shown that the use of encryption is increasing and while there are a number of approaches for attempting to gain access to encrypted digital evidence, there are countermeasures for each of the approaches.

Live digital investigations were introduced as an alternative approach and were discussed in detail, including being defined as ‘a digital investigation which uses the operating system of the system in question to acquire, analyse or present digital evidence’. While discussing a number of live investigation techniques, including the live acquisition of disks and memory, it was shown that if a system is encountered that contains encrypted data then it is possible before the machine is powered down to use live investigation techniques to acquire data in a form that can later be analysed. Live investigation tools were also discussed and it was shown that much of this functionality can now be achieved using an offline analysis of live acquired memory images, meaning that the acquisition and analysis stages of a live investigation can be separated.

Finally in this chapter, while it has been described how acquiring encrypted data from live systems seems a simple solution, it has also been shown that there are a number of problems with results obtained using live techniques: the difficulty trusting results, their inherent intrusiveness, the difficulty in verifying results and ensuring that no evidence is missed. The significance of these challenges will be discussed in the next chapter where the reliability of digital evidence is considered.

CHAPTER 3: ASSESSING THE RELIABILITY OF DIGITAL EVIDENCE

3.1 INTRODUCTION

The previous chapter proposed definitions for digital investigations, live digital investigations and digital evidence. A digital investigation was defined as *a process that formulates and tests hypotheses using digital evidence*. It is also possible that a digital *forensic* investigation may be performed where there is also the additional need for the results to be admissible in court.

Both digital investigations and more specific forensic digital investigations are performed by examining digital evidence, which is defined as *a set of reliable digital objects that support or refute a hypothesis*. The limitation of this definition of digital evidence was explained in Chapter 2 to be that reliability is not defined. Due to the difficulty in defining and therefore assessing reliability of digital evidence directly, this chapter describes how reliability can be assessed using a set of proposed general requirements. These proposed requirements are validated by showing how they are either compatible with existing requirements or that existing requirements are specific means of satisfying those proposed.

The chapter is structured as follows: Section 3.2 explains why requirements are necessary in order to assess the reliability of digital evidence and Section 3.3 proposes general requirements that can be used to assess this reliability. These proposed requirements are then validated in Section 3.4 by comparing them to existing requirements. It is shown that while those that already exist are valid in a particular context, they cannot be considered to be general requirements, but they are specific ways of satisfying the proposed general requirements. Section 3.5 explains the proposed requirements further and discusses how they are satisfied by the technical and procedural measures used in traditional digital investigations. It also

revisits the challenges that live digital investigations face (described in the previous chapter) and determines how these relate to the identified requirements.

3.2 ASSESSING THE RELIABILITY OF DIGITAL EVIDENCE

Digital evidence was defined in Chapter 2 as *a set of reliable digital objects that support or refute a hypothesis*. Therefore, digital objects should only be used as evidence if they are reliable. Defining reliability is extremely difficult as dictionary definitions describe ‘rely’ as being “depend on with full trust”, where ‘trust’ is a “firm belief in someone or something” (Oxford, 2008). Since belief is subjective, this makes independent, objective judgements of trust and therefore reliability difficult.

In other literature, Casey (2002a) describes that “reliability refers to the consistency of a measuring or recording process. A perfectly reliable process will record the same value when repeated measurements of the same entity are taken.” This definition of reliability makes no reference to the process producing correct results, only that they are consistent. If a digital object is used to support or refute a hypothesis, more is needed than consistency. In this research reliability is not used as a measure of consistency, but as a measure of quality.

Rather than attempting to explicitly define reliability in the context of digital investigations, an alternative approach can be taken: “where reliability cannot be assessed directly, there must be some indirect way of assessing reliability” (Miller, 1992). An approach to achieve this is described in Pollitt (1995) which describes the purpose of developing standards for forensic computing as being “to ensure quality”, to “describe that which is the minimum acceptable level of performance” and to “serve as a guarantee to those not involved, of reliable results”. Therefore, it is assumed that the reliability of digital evidence can be assessed indirectly, by meeting certain standards or requirements, and is effectively defined in terms of those requirements.

3.3 PROPOSED REQUIREMENTS FOR DIGITAL EVIDENCE

3.3.1 Introduction

The previous section showed that the reliability of digital evidence can be assessed indirectly by assessing it against certain standards or requirements. There are a number of existing standards and requirements for digital evidence and they are discussed in Section 3.4. However, as will be shown later, none are suitable as general requirements for digital evidence. However, the requirements found in Miller (1992) for assessing the reliability of machine generated evidence e.g. a breathalyser or speedometer, are considered to have the potential to apply to digital evidence, and it is hypothesised that these principles can be adapted into general requirements to assess the reliability of digital evidence.

3.3.2 Requirements in Miller (1992)

Miller (1992) states that in cases where reliability of evidence cannot be assessed directly, the reliability of the source of the information is assessed instead. It also states that “in assessing the reliability of the source of information, several factors apply”, which are: “the source must be authentic” and “it must be possible to assess: a) the accuracy with which the source has recorded the information, b) whether the source accurately reproduces the information, and c) how complete the information is” (Miller, 1992). These are summarised below as authenticity, accuracy and completeness.

Authenticity: Miller (1992) does not define authenticity; however, the assessment of authenticity is divided into a number of questions, which are:

1. “How is it possible to verify the authenticity of input to a machine”?
2. “How is it possible to verify the authenticity of the output of a machine”?
3. “How can the question of deliberate tampering with information stored in a system be dealt with”?

According to Miller (1992), the authenticity of the output of a machine can be verified if a human being is “able to confirm that the output is from the machine in question” and “that the output is the result of one particular process if several processes are performed by the system”, i.e. it needs to be shown that the output was produced by running a particular process. Verifying the authenticity of the input to a machine is described as being more complex since information may be provided from other machines, large volumes of data may be involved and the sources may be human or machine. Miller (1992) gives the example of an accounting system which generates an auditing trail that records transactions carried out on the system. It is designed so that it is possible for a human to verify the authenticity of the information recorded by the system. However, there is no example that can be related to digital investigations. Also, tampering with evidence is not discussed in detail, only that “deliberate tampering with information stored in a system or the deliberate entry of false information into a system” needs to be considered.

Accuracy: Miller (1992) also divides this into two parts:

1. It must be possible to assess the accuracy of the information supplied to the machine
2. It must be possible to assess the accuracy of the information produced by the machine

In Miller (1992) ‘accuracy’ is not defined, but it is described that in accounting systems, manual procedures can be used to verify the accuracy of information supplied to a machine. However, the difficulty arises when more complicated systems are considered where the input to one machine may be the output from another.

Completeness: Miller (1992) states that “if a decision is to be made on the basis of incomplete information, the decision may prove to be incorrect. The completeness of the information depends, in part, upon what decision is to be made”, e.g. in some

cases a bank statement showing current balance is sufficient, but for accounting purposes more details may be needed. One difficulty described is that machine generated evidence “may be unfamiliar in format or presentation [so] it may be difficult for a non-expert to form an opinion about whether it is complete information for the purposes of adjudicating a dispute.”

3.3.3 Application of the Requirements to Digital Evidence

One difficulty in applying the requirements in Miller (1992) directly to digital evidence is that it is unclear what would constitute a ‘machine’. In Miller (1992) ‘machines’ are not explicitly defined, but they are described as being used to “process data” and “are not limited to computers or calculating equipment” and when describing the requirements of accuracy and authenticity of information, machines are described in terms of their input and output. It is also stated that they can obtain information from various sources, including information supplied by a human being; and information supplied by or obtained from another device. Examples include automatic video cameras and digital watches. Dictionary definitions of ‘machine’ also do not help in explaining what they may be, e.g. “an apparatus using mechanical power and having several parts for performing a particular task” (Oxford, 2008), since this excludes non-mechanical devices, i.e. digital devices. It is therefore necessary to consider how the machines in Miller (1992) relate to digital investigations.

As described in Chapter 2, a digital investigation is *a process that formulates and tests hypotheses using digital evidence*. It can also be considered as a series of smaller processes, for example, the acquisition, analysis and presentation of digital evidence. In this research, the machines in Miller (1992) are considered to be equivalent to the processes that make up a digital investigation, since the machines in Miller (1992) process data and have inputs and outputs, as do the processes that make up a digital investigation. The remainder of this section discusses how the requirements in Miller (1992) can be applied to digital evidence.

Authenticity

The dictionary definition of ‘authentic’ means “of undisputed origin; genuine” (Oxford, 2008) and therefore this requirement is concerned with being able to prove where a particular piece of digital evidence came from. Authenticity in Miller (1992) is broken into three parts and is first concerned with the authenticity, or origin, of the input to a process. The input to the first stage of a digital investigation (acquisition) is digital data which, as described in Chapter 2, is an abstraction of something physical e.g. data from a hard disk is actually an interpretation of changes in magnetisation on its surface (Sammes and Jenkinson, 2000 p.93-102). As a result, the ultimate origin of a piece of digital evidence is something physical and therefore digital evidence should be traceable back to an original physical piece of evidence.

The second part of the authenticity requirement in Miller (1992) is that it must also be possible to assess the authenticity of the output from a process. Since the origin of the output of a process is the process itself, this means that it should be possible to demonstrate that the output data was the result of performing a particular process, i.e. to demonstrate what process was used to produce a specific output. This is equivalent to the example given in Miller (1992) of a person being able to verify that the output is from the machine (process) in question.

Miller (1992) also states that “there is also always the question of how to deal with deliberate tampering with information stored in a system or the deliberate entry of false information into a system”. Therefore, accusations of tampering, i.e. “interfering without authority” (Oxford, 2008) with a piece of digital evidence should be refutable.

In this research the requirement of authenticity for digital evidence is therefore a combination of the three aspects of authenticity in Miller (1992) and can be summarised as: *it should be possible to demonstrate the origin of digital evidence, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.*

Accuracy

‘Accurate’ means “correct in all details” (Oxford, 2008), where ‘correct’ is defined as “free from error; true; right” (Oxford, 2008). This requirement has two parts in Miller (1992), where both the accuracy of the input and output of a process should be assessed. The requirement in Miller (1992) is not that the information supplied to or produced by a process needs to be accurate, i.e. free of error, but that the accuracy must be capable of being assessed. This is important given that proving absolute correctness is not possible since “all digital evidence has some degree of uncertainty” Casey (2002a), which is also supported in Palmer (2002) which describes that “there is error in every analytical method” and that “error rates in analysis are a fact. They should not be feared, but they must be measured”.

Given that digital evidence is an abstraction of some physical evidence that is translated through a number of layers of abstraction, and that error can be introduced at each abstraction layer (Carrier, 2003), it is important that at each abstraction layer the possible error is measured and understood. In addition, since this is a requirement that will be used to determine if digital evidence can be considered to be reliable, assessment on its own is insufficient; there must also be a measure of error that can be used to decide if a piece of digital evidence can be considered accurate, and therefore reliable. However, due to the different uses of digital evidence, the measure of error that is acceptable will depend on the context in which it is used and the decision to be made. Therefore, it is not possible to fix a measure of error that is acceptable. Therefore the requirement must be that the error must be acceptably small for the current investigation. Error in digital investigations is discussed in more detail in Chapter 6. In this research the requirement of accuracy means that: *it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation.*

Completeness

Rynearson (1989) cited in Carrier (2006a) points out that “everything is evidence of some event. The key is to identify and then capture evidence related to the incident in question”. A consequence of this is that due to the diverse range of investigations and types of digital evidence, the person performing the investigation is best placed to decide what to include to ensure that evidence is ‘complete’ and only they can justify this decision. This means that the ‘completeness of preserved evidence’ is ultimately subjective. However, using their previous experience and knowledge of the current case, the investigator should be well positioned to determine and to justify what evidence needed to be preserved for this particular case. Once the evidence is presented, it is then up to those making a decision about the evidence to determine if it is sufficient. Therefore, similarly to accuracy, the completeness requirement is: *it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved.*

3.3.4. Summary

This section has proposed general requirements for digital evidence. These were based on those described in Miller (1992) for ‘machine generated evidence’ but have been adapted by considering the processes that are part of a digital investigation to be equivalent to machines in Miller (1992). The three requirements in Miller (1992) have been discussed and it was explained how they apply to digital investigations. The proposed requirements are:

Authenticity: it should be possible to demonstrate the origin of digital evidence, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable,

Accuracy: it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation.

Completeness: it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved.

3.4 EXISTING REQUIREMENTS FOR DIGITAL EVIDENCE

3.4.1 Introduction

This section examines current requirements that are often used for digital evidence and digital investigations. It first shows that there are existing requirements that specifically agree with those proposed; however, the explanations of the principles differ. It also shows how some other existing requirements cannot be satisfied for live investigations, but that this is due to them assuming the use of traditional digital investigation approaches. It also shows that many of these current requirements are actually specific means of satisfying the general requirements proposed in the previous section. In order to show this, a number of sets of existing requirements are discussed. Subsection 3.4.2 discusses a set of requirements that specifically agree with those proposed. The remaining subsections then discuss a number of other sets of requirements, but since many present similar requirements they are divided here into the following subsections: evidence should not be altered, results should be accurate, processes should be repeatable, records of processes should be maintained, information should be authentic, and only what is authorised should be seized.

3.4.2 Requirements that Agree with those Proposed

One set of requirements that need to be mentioned individually are those in Sommer (1998) since they are also adapted and developed from Miller (1992). Sommer (1998) describes three general principles for evaluating evidence that state that evidence

should be authentic, accurate and complete. The explanations of these principles are as follows:

Authentic: It should be possible to show that evidence is “specifically linked to the circumstances and persons alleged – and produced by someone who can answer questions about such links.”

Accurate: It should be possible to show that evidence is “free from any reasonable doubt about the quality of procedures used to collect the material, analyse the material if that is appropriate and necessary and finally to introduce it into court – and produced by someone who can explain what has been done.”

Complete: It should be possible to show that evidence “tells within its own terms a complete story of particular set of circumstances or events.”

These are significantly different to the explanations proposed for using these as general requirements for digital evidence; however, the sentiments in each are similar. Linking evidence to the alleged persons is achieved by connecting digital evidence to some physical evidence, which is then connected to a person. Also, in the proposed requirements it is explicitly stated that accusations of tampering should be refutable, which is necessary since without it, it would be difficult to link digital evidence to the persons alleged. Sommer (1998) states that the evidence should be free from any reasonable doubt about the quality of procedures used to collect, analyse and introduce the material to court. This expression of the requirement demonstrates that these were written for forensic digital investigations, since ‘introduce it into court’ is used rather than ‘present’. Also, the measure of acceptable error used in Sommer (1998) is ‘free from any reasonable doubt’. For requirements to be truly general they must take into account the variety of uses of digital evidence, and as a result specifying that the ‘amount of error should be acceptable in the context of the current investigation’ is considered more appropriate. The completeness requirement in Sommer (1998) specifies that evidence should tell the complete story of a particular

set of circumstances or events. The proposed requirement states that it should be possible to assess what has been preserved and lost, which allows an assessment to be made about whether the evidence does tell a complete story.

Therefore, despite the different explanations of the same three requirements of authentic, accurate and complete, those proposed agree in principle with those in Sommer (1998). However, for the purposes of assessing reliability of digital evidence from live investigations, the proposed requirements provide more explicit criteria describing what is necessary for digital evidence to be considered reliable. In addition to this set of requirements, there are also other requirements that are found in various pieces of literature.

3.4.2 Evidence should not be altered

This requirement is found in various forms in Pollitt (1995), ACPO (2007) and Mocas (2004). This is a relatively simple requirement to satisfy for systems that are encountered in an offline state, since the hard drive of the machine to be examined can be connected to the imaging/analysis machine through a physical write blocker which allows access to the contents of the drive while preventing any writes being made to the disk (Lyle, 2006). However, for machines that are encountered in a running state, it is possible to satisfy this requirement if only the hard drive of the machine is considered to be capable of containing relevant digital evidence. If the memory of the system is considered to be a potential source of digital evidence then this requirement is impossible to satisfy for a live machine regardless of whether a live investigation is performed since ‘pulling the plug’ on a live machine will change and in most cases rapidly clear the contents of RAM (Vidas, 2007, Halderman *et al.*, 2008).

Chapter 2 discussed the necessity of live investigations and showed that in some cases digital evidence in memory can form an essential part of the investigation. As stated in Walters and Petroni (2007), “volatile memory is a critical component of the digital crime scene and as such, should also be integrated into every phase of the digital investigation process used to analyze that crime scene”. Once we accept that

the RAM of a machine is part of the digital crime scene and can contain potentially relevant digital evidence, performing any sort of live investigation makes it impossible to satisfy this requirement. This is because “there is no way to avoid making changes, since in order to conduct a live examination it is necessary to deploy tools on the live system to capture data, and such tools will make changes to the running system” (Sutherland *et al.*, 2008). The requirement to ‘change nothing’ is also heavily criticised in Casey (2007), which states that “conforming to such a standard may be impossible in some circumstances and, therefore, postulating this standard as the ‘best practice’ only opens digital evidence to criticisms that have no bearing on the issues under investigation”. Casey (2007) also draws comparisons with physical world forensics, citing an example of destructive DNA testing that is still considered to be forensically sound, and therefore the requirement of “change nothing is ... inconsistent with other forensic disciplines”. However, it does go on to accept that “the acquisition process should change the original evidence as little as possible and any changes should be documented and assessed in the context of the final analytical results”.

The difficulty in satisfying this requirement is acknowledged in both ACPO (2007) Principle 2 (“where a person finds it necessary to access original data held on a computer or on storage media, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions”) and in Mocas (2004) which states “changing some data of the target machine may be unavoidable”. Mocas (2004) goes on to describe types of interference with evidence as ‘non-interference’ and ‘identifiable interference’, where the former does not change the original data set and in the latter, the original data set is changed but the changes are identifiable.

Not altering evidence therefore should not be used as a general requirement for all digital investigations. However, by revisiting the requirements proposed in the previous section, the requirement to not alter evidence can be considered to be a specific means of satisfying two of the proposed general requirements. First, by not changing any evidence and being able to demonstrate this, it is simple to show the

authenticity of the digital evidence since it can be compared to the data on the original physical device and shown to be the same, thus demonstrating the origin of the digital evidence. Also, if evidence is not changed and this can be shown to be the case, then it is simple to make arguments about the completeness of the preserved evidence, since if nothing has been changed or overwritten then the evidence is as it was when it was first encountered and the maximum amount of evidence possible was preserved. Not altering any evidence is therefore considered as a specific mechanism to demonstrate authenticity and completeness.

3.4.3 Results should be accurate

Pollitt (1995) states that “examination results should be accurate”, but does not expand on this requirement. This could be interpreted as meaning that results need to be completely free from error. This is discussed in Casey (2002a) where accuracy relates to how closely data represents actual events and concludes that “all digital evidence has some degree of uncertainty and an expert should be capable of describing and estimating the level of certainty that can be placed in a given piece of evidence.” Accuracy is therefore accepted as being an important requirement, but only as described in the proposed requirements, where the accuracy must be capable of being assessed, thus allowing a human judgement to be made on whether the error is acceptably low, depending on the context in which the digital evidence is used. In Pollitt (1995) or any other requirements where one hundred percent accuracy is implied, it is a specific instance of an accuracy requirement, that is likely to be unachievable.

3.4.4 Processes should be repeatable

This requirement is described in various forms in Pollitt (1995), Sommer (1999), Mocas (2004) and ACPO (2007). This requirement is problematic to satisfy for many aspects of live digital investigation, particularly the acquisition stage. As discussed in Walters and Petroni (2007), for live investigations, “we can never reproduce the exact same inputs to the exact same tools, thereby making it difficult to prove the

correctness of any results that have been gathered”. To determine if this is a requirement it is necessary to consider what repeatability achieves; if a technique is repeatable, multiple parties can perform the same actions on the same data and show the results are consistent. This actually does not achieve or test accuracy but assesses the precision of the technique¹⁸ since it makes the assumption that the process produces correct results. Accurate results can be demonstrated by performing different processes¹⁹ on the same data and showing the results to be the same, thus increasing confidence that the results are correct. The requirement for the use of repeatable processes is therefore considered as a means of assessing the accuracy of the results and satisfying the accuracy requirement proposed earlier.

3.4.5 Records of processes should be maintained

ACPO (2007) states that “An audit trail or other record of all processes applied to computer-based electronic evidence should be created and preserved”. This allows applied processes to be repeated by others which demonstrates accuracy of results, as discussed in the previous section. It also allows the requirement of authenticity to be addressed since it records the processes that have been used to recover digital evidence from some physical piece of evidence.

A similar requirement stating that “there should be a clear chain of custody or continuity of evidence” is described in Sommer (1998, 1999), and can involve recording the physical items recovered from the original scene. This again demonstrates authenticity in terms of the origin of a piece of digital evidence since it demonstrates where the physical evidence from which digital evidence was recovered was obtained. It also involves recording persons who have had access to the evidence

¹⁸ Oxford (2008) describes the difference between precision and accuracy: “Strictly speaking, *precise* does not mean the same as *accurate*. Accurate means ‘correct in all details’, while precise contains a notion of trying to specify details exactly: if you say ‘It’s 4.04 and 12 seconds’ you are being precise, but not necessarily accurate (your watch might be slow)”

¹⁹ using a different tool or manually examining the original data (Carrier 2003)

which can be used to demonstrate authenticity by limiting those who have access to evidence and therefore reducing the risk of accusations of tampering.

Audit trails of processes applied and continuity of evidence can be used with digital evidence obtained from live investigations and are important for addressing the requirement of authenticity. However, since they provide evidence of authenticity they are not independent requirements.

3.4.6 Information should be authentic

This is a requirement in Mocas (2004), which explains that “it is often important to connect a person to a piece of information”. The need for digital evidence to be traceable back to an original piece of physical evidence was discussed earlier and shown to be part of the authenticity requirement.

3.4.7 Only that which is Authorised Should be Seized

Mocas (2004) describes the ‘minimisation’ requirement where in some cases “the law does not authorize the government to seize items which do not have evidentiary value”. This requirement is specific to forensic digital investigations and is also specific to particular regions’ legal systems. Therefore, this should not form part of general requirements for digital evidence. This example of a region specific requirement further demonstrates why it is necessary to develop requirements for digital investigations in general rather than for forensic digital investigations, since involving specific legal requirements for a particular region would make the reliability of digital evidence region specific. If necessary, it is easier to impose additional requirements on general ones in order to address the particulars of local legislation, rather than to write exceptions.

3.4.8 Summary

There are a number of different proposed requirements for digital evidence and digital investigations. However, they can be shown to be compatible with, or be specific mechanisms to satisfy the requirements proposed in Section 3.3.3 of authenticity,

accuracy and completeness. This supports the claim that the proposed requirements can be used as general requirements to indirectly assess the reliability of digital evidence.

3.5 SATISFYING THESE REQUIREMENTS

3.5.1 Introduction

This section discusses the requirements for digital evidence proposed in Section 3.3.3 and how traditional digital investigation techniques satisfy them. It also explains why these traditional approaches to satisfying these requirements cannot be used during live investigations. Digital investigations are discussed here as they are described in Chapter 2; where the stages of a digital investigation can be summarised as acquisition, analysis and presentation. In condensed form, a traditional digital investigation involves seizing a piece of physical evidence e.g. a computer, at some location, which is taken from the scene and stored at a secure location. At some point a disk image is created of the hard drive of the machine and verified against the actual disk contents using a cryptographic hash e.g. MD5 or SHA1. The disk image is then examined using forensic software e.g. *EnCase*, *FTK* etc. and the results presented in a report. The following subsection describes how the proposed requirements are satisfied by this traditional digital investigation process.

3.5.2 Authenticity

The requirement for authenticity described in Section 3.3.3 was *it should be possible to demonstrate the origin of digital evidence, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.* In a traditional digital investigation it is possible to demonstrate that digital evidence came from a particular piece of physical evidence since generally a full disk image of a drive is obtained and examined. The cryptographic hash, (e.g. MD5) of the disk image can be compared to that of the contents of the physical drive from which it came and

shown to be identical, thus demonstrating that the disk image and therefore digital evidence extracted from it originated from the seized physical evidence. This physical evidence can be traced back to a physical location by examining documentation of the original seizure. Accusations of tampering can be minimised through the principle of 'continuity of evidence', where it is documented who has had access to the physical evidence and at what stage. Also, any tampering with the evidence can be detected from the point at which a hash of the evidence is first recorded.

In a live digital investigation, continuity of evidence is still possible after the initial seizure. However, the proof that digital evidence came from a particular physical piece of evidence may not be possible in the same way. This is because data may have been volatile, and after acquisition at the scene from a live machine, no longer exists, e.g. it was wiped when the power was removed. Therefore, demonstrating authenticity of live acquired evidence currently relies on documenting the process and trusting those performing the seizure to report the origin correctly.

3.5.3 Accuracy

The requirement for accuracy described in Section 3.3.3 was *it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation*. In a traditional digital investigation the accuracy of the results can easily be assessed since the original physical evidence is accessible. This allows the accuracy of the acquisition stage of a digital investigation to be assessed by any number of people who can re-acquire the disk image and compare the cryptographic hash of the new image to that of existing disk images.

The accuracy of the analysis stage is more complicated and relies on multiple examiners (e.g. prosecution and defence) being able to perform the same analysis on the same raw data. They are able to use the same tools on the same raw data and check the results are the same, which ensures that the tools were used correctly and were operating normally. Also, accuracy can be determined for each interpretation of raw data through manual verification or using multiple tools (Carrier, 2003). From the

abstractions of the raw data, multiple examiners can perform their own high level analyses and come to their own conclusions about the evidence. These conclusions may or may not agree

In a live digital investigation, in many cases, the acquisition stage can be performed only once, meaning the assessment of accuracy using repeated acquisitions is not possible. Also, running tools that acquire, analyse and present results all on the live system means that no assessment of accuracy of any stage is possible through repeatability. This is because the output from the acquisition stage (i.e. the input to the analysis stage), is not preserved and therefore the raw data is not preserved to be inspected by multiple parties. As described in Chapter 2, this problem can be addressed by separating out the acquisition and analysis stages of a live digital investigation. For example, a memory image can be acquired from a live machine, followed by an analysis of that memory image which takes place ‘offline’ in a trusted environment. Here the accuracy of the results of analysis can be determined using the same repeatability method as a traditional digital investigation. However, the problem of determining the accuracy of the acquisition stage remains.

3.5.4 Completeness

The requirement for completeness described in Section 3.3.3 was *it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved*. For traditional digital investigations, this is satisfied by adhering to guidelines that predetermine the scope of what potentially relevant digital evidence should be preserved, e.g. preserving and acquiring the entire hard drive but discarding data in memory: “it is accepted that the action of switching off the computer may mean that a small amount of evidence may be unrecoverable if it has not been saved to a storage medium but the integrity of the evidence already present will be retained” (ACPO, 2003).

In a live digital investigation the preservation of potentially relevant digital evidence is more complicated. Since any tools that are used will make changes to the

system under investigation (Sutherland *et al.*, 2008) and it is difficult to determine what has been altered, it is therefore difficult to assess what has been preserved and lost, and therefore if the maximum amount of relevant digital evidence has been preserved. This requirement is therefore difficult to satisfy and there is presently no way of addressing this problem.

3.6 Challenges to Live Investigations

The challenges to live investigations identified in Chapter 2 were: the difficulty trusting results, inherent intrusiveness, difficulty in verifying results and ensuring no evidence is missed.

In Chapter 2, the concerns regarding trusting results were that the operating system could be modified to provide false information (either hiding malware that was responsible for incriminating material on the system, or an anti-forensic rootkit deliberately installed to hide data from a live investigation) or that ‘logic bombs’ could be placed on the system which could destroy evidence if triggered. Relating these to the proposed requirements, it can be seen that the former (OS modification) is concerned with the accuracy requirement for digital evidence, since both malware and anti-forensic rootkits could mean that the acquired data contains error. The latter (logic bombs) is concerned with the completeness of the preserved evidence since the use of a logic bomb would make it difficult to assess what evidence has been lost and could result in evidence being erased which would obviously decrease the completeness of the preserved evidence.

The inherent intrusiveness of live techniques is also a completeness problem, since as tools and techniques make changes to a system, it becomes more difficult to assess what data is preserved and what is lost, and this could result in a decrease in the amount of preserved relevant digital evidence.

Verification of results is challenging in live investigations since it is difficult to supply the exact same inputs to tools (Walters and Petroni, 2007), particularly at the acquisition stage. When related to the proposed requirements, the difficulty in

verifying results means that it is a problem assessing the accuracy of results of live investigations.

Ensuring that no evidence is missed could be problematic for live investigations if partial acquisitions are performed. Since the entire hard disk would not be preserved, it is possible that challenges could be raised about the completeness of the preserved digital evidence and that something that was relevant was not collected. However, this is only a challenge where not all data on a system is collected and partial acquisitions and live investigations are not synonymous.

3.7 Chapter Summary

This chapter has discussed the 'reliability' aspect of digital evidence; including the difficulty in defining reliability and also that measuring it directly may not be possible. It has also shown that the reliability of digital evidence can be measured indirectly by evaluating it against a number of requirements. This chapter has examined the requirements in Miller (1992) for machine generated evidence and shown how they can be applied to digital evidence. The proposed requirements and their explanations are:

Authenticity: it should be possible to demonstrate the origin of digital evidence, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.

Accuracy: it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation.

Completeness: it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved.

These are proposed as general requirements for digital evidence and have been validated by comparing them against a number of existing requirements which were shown either to be compatible with, or to be specific mechanisms of satisfying those proposed. This chapter has also explained how the proposed requirements are satisfied in traditional digital investigations and how live digital investigations cannot satisfy them in exactly the same way.

The remainder of this thesis considers the extent to which these three requirements can be satisfied by live investigations that involve encrypted evidence, and therefore determines to what extent digital evidence obtained from live investigations can be considered to be reliable.

CHAPTER 4: COMPLETENESS AND ENCRYPTION

4.1 INTRODUCTION

The previous section described the necessity for and details of the requirements used to assess the reliability of digital evidence. This section examines the ‘completeness’ requirement in the context of digital investigations involving encrypted evidence. Completeness was explained as *it should be possible to assess which evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved*. This chapter examines how the type of encryption on a system can affect the completeness of evidence recovered from an offline or ‘dead’ investigation. It considers if live investigations could increase the amount of evidence preserved and therefore increase the completeness and offer a more reliable set of digital evidence than traditional digital investigations.

4.2 BACKGROUND

This chapter examines the latter part of the completeness requirement described above and in Chapter 3 and therefore considers how the maximum amount of relevant digital evidence could be preserved. The difficulty with this requirement is identifying which digital evidence is potentially relevant to the investigation. There have been attempts to define the types and location of digital evidence that are specific to different types of investigations. For example, guidelines from the National Institute of Justice (2001) identify fourteen ‘crime categories’, a selection of which are shown in Table 3 to illustrate the types of digital evidence that can be examined.

Crime Category	Common findings
Child Exploitation	Chat logs, data time stamps, digital camera software, e-mail/notes/letters, games, graphic editing and viewing software, images, Internet activity logs, movie files, user created directory and file names that classify images.
Computer Intrusion	Address books, configuration files, e-mail/notes/letters, executable programs, Internet activity logs, Internet protocol (IP) address and usernames, Internet Relay Chat (IRC) logs, source code, text files.
E-mail Threats/ Harassment/Stalking	Address books, diaries, e-mail/notes/letters, financial/asset records, images, Internet activity logs, legal documents, telephone records, victim background research.

Table 3: A selection of ‘common findings’ from three of the fourteen different crime categories in NIJ (2001).

While these are useful broad starting points, since individual cases have specific requirements, these ‘crime categories’ can be used as guidelines only, and cannot contain exhaustive lists that define the scope or completeness of digital investigations.

The question of ‘completeness of evidence’ is particularly relevant when discussing the investigation of large volumes of data; particularly when the idea of ‘partial’ or ‘selective’ acquisition is suggested. A selective acquisition occurs when the decision is made “not to acquire all the possible information during the capture process” (Turner, 2006). In the case of selective acquisition, completeness is an issue since it is difficult to know “that you have captured everything relevant to the case under investigation or have not missed evidence of other offences” (Turner, 2006). It is also possible that evidence that proves the suspect’s innocence was missed.

Kenneally and Brown (2005) examines in more detail the potential problems of selective acquisition, e.g. data that is not collected is inaccessible to the defence and could be relevant. Kenneally and Brown (2005) heavily focuses on case law from the United States to argue in favour of what is described as ‘risk sensitive digital evidence collection’, i.e. a selective acquisition, and also uses examples from physical forensic science. It is suggested that ‘reasonableness’ should be used to determine the scope of an investigation; “just as it would be unreasonable to expect that investigators cordon-off an entire building, mercury fulminate hundreds of offices for

latent fingerprints and seize every file cabinet during the course of a robbery scene investigation”, “the reasonableness standard takes into account cost and capabilities, and does not require perfection.” This example of seizing entire buildings in physical forensic science examples is often used, “complete bit-by-bit captures of huge targets may be completely impractical, in the same sense that capturing the state of an entire building is impractical in a (non-digital) forensics investigation involving a murder” (Richard III and Roussev, 2006).

While the issue of the completeness of selective acquisitions is interesting and is likely to be highly relevant for the future of digital investigations, it is considered to be outside the scope of this research. This work assumes that access to encrypted evidence is desired to ensure completeness. While it does not go as far as Forster (2005), where it is assumed that data that has been encrypted is likely to be of evidential value since “it is usually incriminating or unlawful material that suspects seek to hide in this way”, it is assumed that encrypted data is potentially relevant and it is necessary to gain access in order to determine if it is, or is not relevant to the investigation. Therefore, if encrypted data from a system is preserved in a form that can be analysed rather than being lost or rendered inaccessible then completeness is taken to be increased.

4.3 METHODOLOGY

This section describes the methodology that is used to examine encryption software and determine how a traditional digital investigation would be affected if such software was in use. The general methodology section describes the way in which the different products are categorised, which allows generalisations to be made about the locations on disk in which evidence is rendered inaccessible by encryption software. This is followed by the experimental methodology section which describes how the amount of evidence that is left in an accessible form is quantified.

4.3.1 General Methodology

To determine whether the completeness of preserved digital evidence (that would otherwise be in encrypted form) will increase if a live investigation is performed, it is

necessary to know whether the encrypted evidence is likely to be accessible if a live investigation were not performed and offline or 'dead' approaches were used instead (described in Chapter 2). Since there are a large number of different encryption products available, in order to generalise about the success of offline approaches and therefore the need for a live investigation, it is necessary to categorise encryption products in some way.

An existing method of categorisation is described in WinMagic (2005), which separates encryption software into four categories. These categories are:

Manual File Encryption: A user selects a single file for encryption;

Folder Encryption: all files contained within a particular folder are automatically encrypted;

Virtual Drive Encryption: A virtual drive is created which is stored as a single file on the user's file system. All data stored on that virtual drive is automatically encrypted. Data is decrypted on a block basis rather than by file;

Disk Encryption: This encrypts all data on the disk including the operating system itself.

These categories can be considered to define the scope of the encryption, since the category distinctions are made based on how much data is encrypted, a single file, folder, virtual drive or the entire disk. As more of the disk is encrypted, a live investigation will allow the preservation of more data that would otherwise be lost. However, there are techniques that can be used to attempt to gain access to encrypted data offline without performing a live investigation, which were described in Section 2.3.3. The success of some of these approaches was dependant on what data on disk remained in unencrypted form. Since products are categorised based on this property, depending on which locations remain unencrypted, some product categories may not require a live investigation to be performed. Table 4.2 that follows discusses whether the success of each of the offline approaches is affected by the category of product in use.

Approach	Discussion	Success can be generalised depending on category
Obtain keys from suspect	This is dependent on the co-operation of the suspect, which is too case specific to generalise with regard to encryption product categories.	n
Locate unencrypted data	Since plaintext can be found on disk, the amount of the disk that is accessible to an offline investigation is relevant and the success of this approach may be able to be generalised based on the product category.	y
Locate copies of the key/password	Locating copies of the key is dependant on the key backup mechanism of the specific product in use, for example, <i>BitLocker</i> keys can be printed or stored on USB key or any other folder (Microsoft, 2007d), <i>TrueCrypt</i> full volume encryption requires that a <i>TrueCrypt</i> recovery CD is created (TrueCrypt, 2008e). However, as described in Chapter 2, the disk can be scanned for strings which may include any saved copies of the typed password, whether stored accidentally or on purpose. Therefore, the amount of disk available affects the amount strings that can be extracted and tried as possible passwords.	y
Intelligent password attacks	The success of this approach depends on the complexity and length of the password used and the technical capability of the suspect. Password attacks can be speeded up using rainbow tables, where information is pre-calculated for specific sets of passwords. However, for longer passwords this is not feasible and it is necessary to be selective about the passwords tested. Intelligent password attacks are therefore dependent on the availability of information upon which to select likely passwords. Therefore the amount of the disk that is accessible to the investigator will have an effect on the success of this approach.	y
Exhaustive keys search	As described in Chapter 2, brute force attacks are not feasible on modern algorithms. Therefore the success of this approach depends on a specific implementation using an insecure algorithm.	n
Vulnerabilities in implementations	This is, by definition, implementation specific.	n
Cryptanalysis	As described in Chapter 2, in this research, the algorithms used are considered not to be vulnerable to cryptanalysis. The success of this approach would be dependant on a specific implementation using an insecure algorithm.	n
Surveillance	This is case specific; however, the type of encryption may have an effect on the type of surveillance that can be used. For example pre-boot encryption such as <i>Bitlocker</i> prevent the use of software based key loggers but hardware versions can still be used. However, this is not considered in this research.	n

Table 4: Descriptions of whether the success of offline approaches to gaining access to encrypted digital evidence depends on the product category.

Therefore, the success of three of the approaches described is affected by the locations on disk that are accessible to an offline investigation, and therefore the type of encryption product in use.

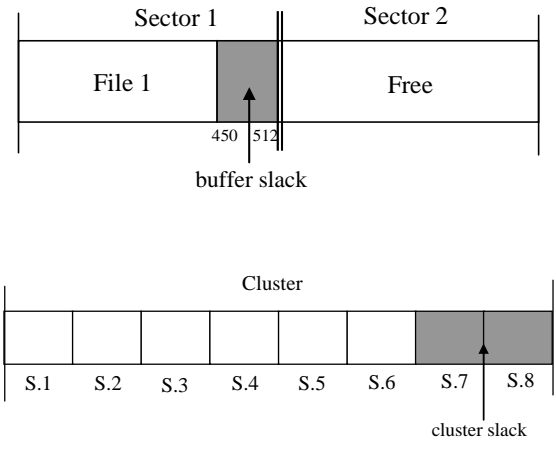
As can be seen in Table 4, the first approach that could be affected by product category is 'locating unencrypted data'. For this there are two situations to consider. The first is whether the encryption software itself leaves plaintext data on the disk. This can either be as a result of failing to erase the original plaintext or by the creation of temporary files when data is decrypted. The second situation to consider is whether other applications that use the encrypted data while it is in decrypted form create copies of plaintexts that will be later accessible to an offline investigation. Metadata about files may also be left, even if the contents are not. This will be specific to the application that opens the encrypted data, e.g. *notepad*, *Microsoft Word* etc. Despite this, it is possible to identify particular locations on disk that could contain evidence that could be useful when using the described approaches, which are shown in Table 5.

For the second approach, 'locating copies of passwords', there are a number of likely candidate locations that may provide other passwords used by the suspect. These include the pagefile and also saved passwords from Internet browsers etc. These are also described in Table 5.

The third and final approach 'intelligent password attacks' relies on collecting personal details about the suspect, from which likely passwords can be constructed. For this approach, a number of locations may be of use, for example browsing history and personal files. This is discussed in Table 5.

To determine whether the offline approaches are likely to be successful, it is investigated whether particular locations on disk remain accessible after the power is removed. Table 5 that follows describes different locations and content on disk that may be use in gaining access to the encrypted material on the disk. These locations are based on those in WinMagic (2005); however, some additional locations have been added that are specific to the encrypted data recovery techniques discussed in this chapter.

Location	Description
Temporary Files	<p>A temporary file is “a file created either in memory or on disk, by the operating system or some other program, to be used during a session and then discarded” (Microsoft, 2002). If a file that is encrypted is opened by some piece of software for editing or viewing then it is possible that a temporary file in decrypted form could be created which is not erased at the end of the session. Therefore, access to temporary files may mean access to copies of the plaintext.</p>
Pagefiles	<p>Pagefiles exist because RAM is a limited resource and when the total memory needed on a system exceeds what is available, data that is not immediately needed is ‘paged out’ of memory and stored on the disk (Microsoft, 2004). When the data is needed it is paged back into memory.</p> <p>Pagefiles may contain a number of different types of useful information; they may contain temporary decrypted copies of encrypted data, passwords from memory that have been paged to disk, or data from memory from other applications that may help with intelligent password attacks.</p>
Slack space	<p>There are two types of slack space (Carrier, 2005 p.187).</p> <ol style="list-style-type: none"> 1. Between the exact end of the file and the end of the sector in which the file ends, which can contain data from memory (buffer slack or RAM slack). 2. Between the last sector that contains part of the file and the last sector of the cluster, which can contain data from previous files that resided on that part of the disk (cluster slack). <p>This is shown below diagrammatically (Sammes and Jenkinson, 2007).</p>

	 <p>Slack space is important since it may provide access to deleted data which could be deleted plaintext or data that could be used to construct passwords.</p>
The Recycle Bin	<p>When files are deleted, they are first moved to the Recycle Bin. The file is renamed using the convention <DRIVELETTER><#>.<ORIGINALEXTENSION> e.g. 'd1.txt'. Its original name and path is stored in an INFO2 file in a folder named 'recycled' (Microsoft, 2007a). These files may be deleted plaintext or contain information that could be used to construct passwords.</p>
Deleted Files	<p>When the Recycle Bin is emptied, only then are files actually deleted. Even then, files are not actually erased. The space that the file occupied is marked as free and can be overwritten by new data stored to disk. The files are therefore still accessible after deletion for an undetermined but non-zero amount of time.</p>
The <i>Windows</i> Registry	<p>The <i>Windows</i> Registry is “a central hierarchical database used ... to store information that is necessary to configure the system for one or more users, applications and hardware devices” (Microsoft, 2002). The Registry is stored as a number of ‘hive’ files, which are “files that contain a Registry sub-tree” (Russovich and Solomon, 2005 p.263).</p> <p>The Registry could contain a number of useful pieces of information, including hashes of some passwords used by the suspect, programs run and recently accessed files (which may point to encrypted files).</p>
Users Folders, e.g.	<p>This is where the majority of user data is likely to be stored.</p>

C:\Documents and Settings\User	Also applications should write their configuration data (e.g. browser caches) to these folders (Gajic, 2008). Encryption software may therefore use this location to store temporary files. Also, user data can be used to obtain information upon which to base intelligent password attacks. Alternatively user data may contain passwords intentionally stored by users to assist in remembering them.
Hibernation file	The hibernation file contains the complete state of the system at a specific point in time, including the memory (Ruff and Suiche, 2007). Therefore, it could contain passwords or plaintext that was stored in memory at the time of the hibernation.
Hidden partitions	This is a “portion of the hard disk that an operating system, such as <i>Windows</i> , does not recognize or display a file system for” (WinMagic, 2005) and therefore could be used to conceal data since it is not accessible through normal use of the system.
Free space between partitions	Between partitions and at the end of the disk is free space, since partitions can be created with gaps in between. It is also possible to conceal data in these locations.

Table 5: Locations on disk that may assist in providing access to encrypted data.

WinMagic (2005) makes specific predictions about the availability of some of these locations for the different encryption scope categories. The diagrams used in WinMagic (2005) are shown in Figure 6, and are summarised in Table 6.

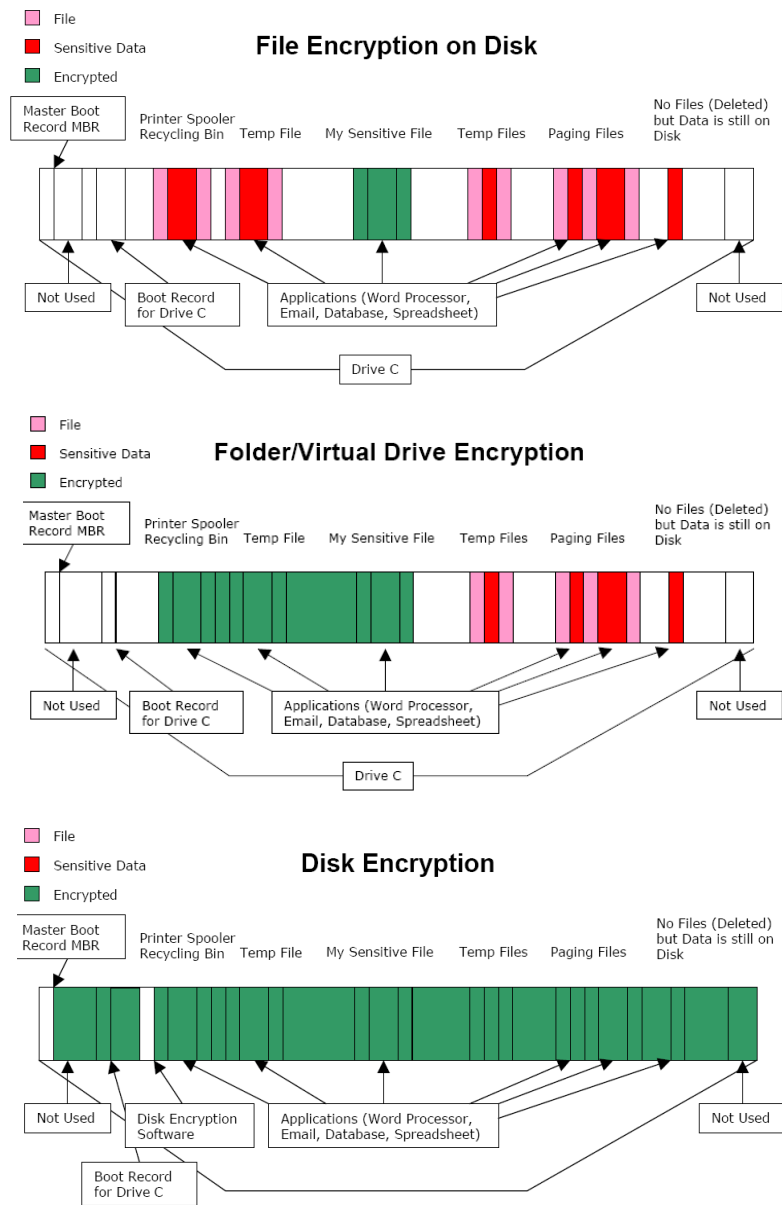


Figure 6: Graphics from WinMagic (2005) showing the availability of locations on disk

Location	Expected Availability of Location			
	Single File	Folder	Virtual Drive	Full Disk
Temporary Files	available	possible	possible	encrypted
Paging Files	available	available	available	encrypted
Slack Space	available	available	encrypted	encrypted
The Recycle Bin	available	encrypted	encrypted	encrypted
Deleted Files	available	available	encrypted	encrypted
User folders ²⁰	unspecified	unspecified	unspecified	unspecified
The Registry	available	available	available	encrypted
Hibernation and Sleep Files	available	unspecified	unspecified	encrypted
Hidden Partitions	available	unspecified	unspecified	encrypted
Free Space between Partitions	available	available	unspecified	encrypted

Table 6: Predictions of availability of unencrypted data to an offline analysis (WinMagic, 2005).

'Possible' is entered if data in these locations may or may not be encrypted e.g. for temporary files, it depends in where they are generated. Also, 'unspecified' is entered where there were no claims made about the availability of data in a particular location.

In order to determine the correctness of these predictions of availability of particular locations, experiments are set up to examine these locations on disk images from systems that have been running a variety of different encryption software from the different categories.

4.3.2 Experimental Methodology

For most categories examined, three products are used, with the exception being 'folder encryption' where only one product could be found to belong to the category. The selection of products is based on the extent of their use or if they are particularly of interest. Random sampling from a sample frame could be used but since statistical techniques are not being applied and only inductive conclusions are drawn, random sampling is not considered necessary. Furthermore, this testing provides additional information that may be of use to the community²¹ other than just for the purpose of

²⁰ This location was not discussed in WinMagic (2005) but has been added here since it may be useful for intelligent password attacks.

²¹ E.g. the location of temporary files generated by the encryption software

this research. If products were chosen for testing at random and they were not in popular use, then this additional information would be of little further use.

The tests are carried out on copies of a baseline virtual machine running *Windows XP SP2*. The use of virtual machines was discussed in Chapter 1. The baseline virtual machine used in this case has the following disk map:

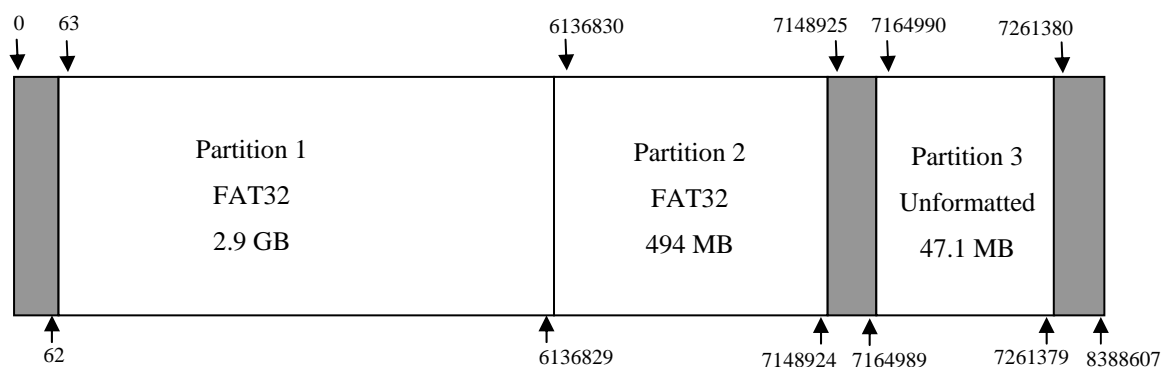


Figure 7: Disk map of the test drive used, showing start and end sectors of the different partitions.

Each product under test is examined in its own virtual machine. The general procedure used for each product examination is detailed below.

1. The baseline virtual machine is cloned to a new folder and booted.
2. The encryption product on test is installed on the machine (in the full disk encryption cases this also involves encrypting the disk and in the case of virtual disks, the creation, formatting and mounting of a virtual disk).
3. A new folder named ‘test’ is created on the virtual system (on the desktop for most but on the virtual drive in the appropriate cases).
4. The programs *gentest*²² and *gentemp*²³ are also copied to the test folder.
5. A command prompt is opened on the virtual system and *gentest* is run from the test folder with the parameters `gentest 600 b 1`, which produces a

²² The program *gentest* was written in C and produced a number of plaintexts of a specified size. The parameters passed are `gentest [size] [unit (m/k/b)] [number of files]`. The program produces files containing the text “This is the plaintext” preceded by a unique line number. This was inspired by (Farmer and Venema 2004 p.172)

²³ The program *gentemp* is also written in C and represents the worst case in terms of temporary file generation. This program simply creates an exact duplicate of the file it opens and places it in the folder from which the program was called. It then fails to delete the generated temporary file. The parameters passed are `gentemp [path + name of file to open]`

single 600 byte file, which is just larger than a sector (512 bytes) and has both buffer and cluster slack.

6. The virtual machine is paused (after a short time in order for writes to the virtual disk to take place).
7. The file representing the hard disk of the virtual system is examined in *X-Ways Forensics* and the disk examined for evidence of the plaintext file. If it is found then its location is documented.
8. Each of the techniques listed below are applied to test for the presence of plaintext data in each of the locations specified earlier. The design of this procedure allows all locations to be examined sequentially in the order described in Table 7 without needing to revert to the virtual machine to the baseline snapshot.

Location	Method Used
Temporary Files	To test if temporary files are accessible, the developed software <i>gentest</i> is used. This simply creates a duplicate of the opened file in the directory from which it is called. This represents the most extreme form of temporary file generation, where the entire file is duplicated. Temporary files are generated in the location of the test file and also in the root of the C:\ drive.
Paging Files	The file on disk representing the virtual machine's hard disk is loaded into <i>X-Ways Forensics</i> and the PageFile examined. If the contents are accessible (which is obvious since text is usually visible somewhere) then the Pagefile is considered to be 'accessible'. Also, where possible, attempts are made to use the encryption product under test to deliberately encrypt the pagefile.
Slack Space	As mentioned earlier, files are generated that are not multiples of the sector size. In these cases <code>gentest 600 b 1</code> is used to generate a 600 byte file that just stretches across two sectors. The buffer and cluster slack of the original (where possible) and the encrypted file are examined before and after encryption. Also the general accessibility of slack space on other files on the disk is determined.
The Recycle Bin	Where possible the plaintext file is sent to the Recycle Bin and is accessed offline through <i>X-Ways Forensics</i> . If the contents of the

	file can be read then the Recycle Bin is considered to be accessible. Encrypted files are also sent to the Recycle Bin and the contents examined.
Deleted Files	The Recycle Bin is emptied and the sectors in which the plaintext file previously existed examined and the 'this is the plaintext' string searched for. Also, in cases where the plaintext is erased by the encryption software, the sectors are examined and if non-text strings are encountered then the <i>ENT</i> program (Walker, 2008) used to test for statistical randomness (to check plaintext data was not simply permuted or substituted). Other files on the disk are also deleted to infer about the general accessibility of deleted files.
The Registry	The hive files that make up the <i>Windows</i> Registry (SYSTEM, SAM, SOFTWARE, DEFAULT, NTUSER) are opened in <i>X-Ways Forensics</i> . If the hives can be mounted and explored then they are considered to be accessible.
User folders	The folder C:\Documents and Settings\Chris is examined and if the folder can be browsed then it is considered to be accessible. A deliberate attempt is made to encrypt a file from this location. A sample file C:\Documents and Settings\Chris\Cookies\index.dat is used as a test case to determine whether it can be encrypted (this file is known to be reported as 'in use' by <i>Windows</i>).
Hibernation Files	The default <i>Windows</i> hibernation file is stored in C:\hiberfil.sys. If this file can be accessed in the disk image then the hibernation file is considered to be accessible.
Hidden Partitions	As described earlier, the disk of the virtual test system is set up so that there are several partitions. Partition 3 is unformatted and is therefore not visible to <i>Windows</i> but is manually filled with the test string "Hidden partition". <i>X-Ways Forensics</i> is used to attempt to view the partition and if this is successful then it is considered to be accessible.
Free Space between Partitions	As described earlier, the disk is set up with space between partitions. This is edited manually in the baseline image to contain the text "space between partitions". If after encryption this text is visible using <i>X-Ways Forensics</i> then this area of the disk is considered to be accessible.

Table 7: Techniques used to test for the presence of plaintext data in various locations.

4.4 RESULTS

4.4.1 Single File Encryption Results

AxCrypt

This is an open source, single file encryption product that uses AES encryption with 128 bit keys (Axantum Software, 2008). It integrates with *Windows Explorer* and encryption is performed by right clicking a file and selecting encrypt, as shown in Figure 8. An encrypted file is opened by double clicking which then decrypts the file and opens it with the default program.



Figure 8: *AxCrypt* and *Windows Explorer* right click integration providing the option for encryption and decryption.

The results for the offline examination of an *AxCrypt* system are described in Table 8.

Location	Results
Temporary Files	Temporary files were produced by both the decryption process and by the software <i>gentemp</i> . ‘Pulling the plug’ while the plaintext file was open revealed a temporary copy of the plaintext file in the C:\Documents and Settings\Chris\Local Settings\Temp\axcrypt\... Also, temporary files produced using the <i>gentemp</i> program were accessible.

Paging Files	An examination of the disk image showed that the pagefile was accessible to the offline analysis. Attempts to encrypt the pagefile with <i>AxCrypt</i> failed.
Slack Space	When encryption was applied to the test files, the buffer slack of the original plaintext was filled with zeros but the cluster slack was found to contain data from the files previously stored in that location. However, other than the manually encrypted files, other files' buffer slack and cluster slack was accessible.
The Recycle Bin	An examination of the disk image showed that the contents of the Recycle Bin were accessible to the offline analysis. Attempts to encrypt the Recycle Bin were not successful. However, encrypted files that were sent to the Recycle Bin remained encrypted.
Deleted Files	<i>AxCrypt</i> encrypts to a new file and by default erases the original plaintext. The original sectors containing the plaintext were examined and found to contain random data. However, other files on the disk that were not encrypted and were deleted were accessible.
The Registry	An examination of the disk image showed that the Registry hive files were accessible to the offline analysis. Attempts to encrypt the hive files were not successful.
User folders	Individual files in user folders could be encrypted but this had to be done manually. However, attempting to encrypt C:\Documents and Settings\Chris\Cookies\index.dat failed.
Hibernation Files	The hibernate file, hiberfil.sys was available to an offline analysis. Attempts to encrypt this file were not successful.
Hidden Partitions	Other partitions were available to the offline analysis and could not be encrypted.
Free Space between Partitions	Free space between partitions was available to an offline analysis.

Table 8: Results from *AxCrypt*.

GNU Privacy Guard (GNUPG) (for *Windows*)

This is a free, open source implementation of the *OpenPGP* standard (Koch, 2007) which includes a full replacement for *PGP* that can be used on messages or on files. The software also provides right click integration with *Windows Explorer* for file encryption, as shown in Figure 9. The results of the offline encryption of *GNU Privacy Guard* are shown in Table 9.

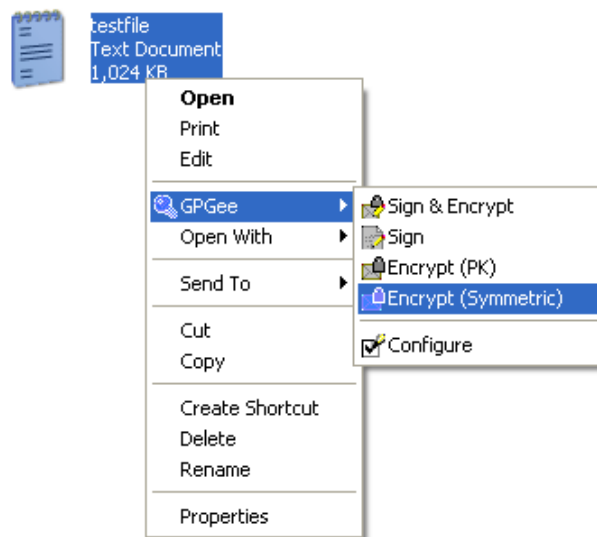


Figure 9: *Windows Explorer* integration of *GNU Privacy Guard*.

Location	Results
Temporary Files	When the encrypted file was accessed, a decrypted copy was created in the same directory as the encrypted file. When access to the file is no longer required, it needs to be either re-encrypted or manually erased. Temporary files were therefore accessible to an offline analysis. The same is true for any temporary files produced by software used to view the decrypted files.
Paging Files	The pagefile was accessible to an offline analysis and could not be encrypted.
Slack Space	The buffer slack of the encrypted file consisted of zeros and the cluster slack contained data from files that were previously stored in that location.

The Recycle Bin	The Recycle Bin was accessible to an offline analysis.
Deleted Files	<i>GNUPG</i> encrypted the files to a new location and left the originals in place. <i>GNUPG</i> does not provide file erasing capabilities. Therefore, if plaintext files are deleted, they are likely to be available to an offline analysis unless manually erased using separate software.
The Registry	The Registry hive files were available to an offline analysis.
User folders	Individual files in user folders could be encrypted but this had to be done manually. Attempting to encrypt C:\Documents and Settings\Chris\Cookies\index.dat failed.
Hibernation Files	Hiberfil.sys was available to an offline analysis.
Hidden Partitions	Additional partitions were available to an offline analysis.
Free Space between Partitions	This was accessible to an offline analysis.

Table 9: Results from *GNU Privacy Guard*.

‘Encrypt Files’

The mechanism by which this software operates is slightly different but it still involves manual single file encryption. Instead of *Windows Explorer* integration, it uses a separate program that accesses the files on the disk. Through this single interface, file encryption is performed, as shown in Figure 10. The results of the offline examination of *Encrypt Files* are shown in Table 10.

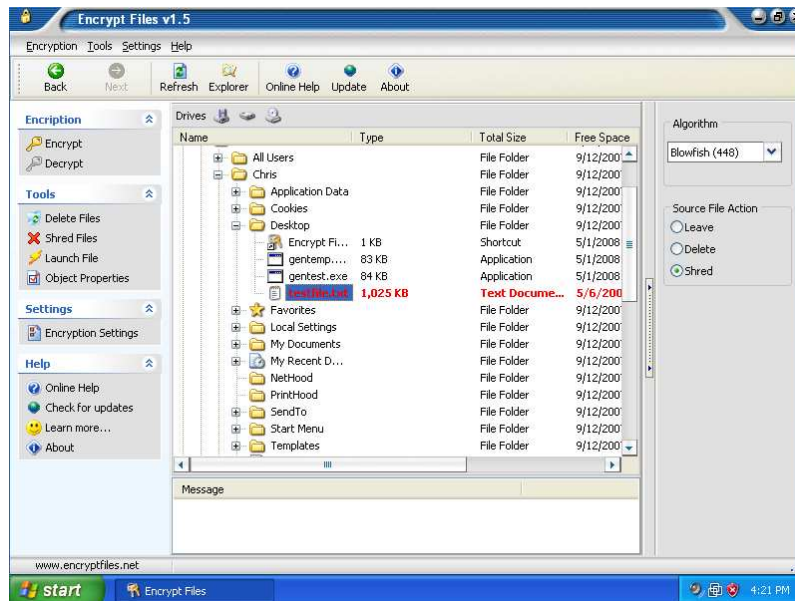


Figure 10: The *Encrypt Files* software providing access to files on disk and the option to encrypt or decrypt those files..

Location	Results
Temporary Files	The <i>Encrypt Files</i> software was used to convert files between encrypted and decrypted states. The availability of the plaintext depends on the 'Source File Action' option, used to re-encrypt the file each time, which is either 'leave', 'delete' or 'shred'. From inside the software, once a file was decrypted it could be opened with other software. Any temporary files generated by other software were in unencrypted form and were accessible unless these additional copies were manually encrypted or erased.
Paging Files	The pagefile was accessible to an offline analysis and could not be encrypted using <i>Encrypt Files</i> .
Slack Space	The buffer slack was zeroed and the cluster slack contained data previously stored at that location. In general the file slack of files on the disk was still accessible.
The Recycle Bin	The Recycle Bin was accessible to an offline analysis. However, using the interface of <i>Encrypt Files</i> , files that had been previously sent to the Recycle Bin could be manually encrypted.
Deleted Files	Several options were provided for plaintext: 'leave', 'delete' or 'shred'. The default option is 'shred' which overwrites the original plaintext. However, when the 'delete' option was used, it was found that deleted files could be accessed.

The Registry	The hive files that make up the Registry were available to an offline analysis and could not be encrypted using <i>Encrypt Files</i> .
User folders	Files in user folders could be manually encrypted but need manual decryption before they could be used. Index.dat could not be encrypted.
Hibernation Files	The hibernate file was available to an offline analysis and could not be encrypted using the software.
Hidden Partitions	Additional partitions were available to an offline analysis and could not be encrypted using <i>Encrypt Files</i> .
Free Space between Partitions	This was accessible to an offline analysis and could not be encrypted using <i>Encrypt Files</i> .

Table 10: Results from Encrypt Files.

Summary of Single File Encryption

For all the single file encryption products, the majority of locations remained accessible to an offline analysis. Files had to be manually encrypted and decrypted and existed fully on disk in one of these states. For *AxCrypt* and *GNU Privacy Guard* files were decrypted to temporary files in order to be accessed, and while *Encrypt Files* did use temporary files, the actual file's state changed between encrypted and decrypted and the availability of the previous state is dependent on the 'source file action' selected (shred, leave or delete). Any temporary files produced by other software were also accessible. The temporary files produced by the encryption software may or may not be erased after use, depending on the implementation. If the temporary files were not erased but deleted, then they would be accessible in unallocated space but are susceptible to being overwritten by new data. Also, only the logical file was encrypted and the cluster slack remained accessible.

AxCrypt erases the plaintext file after encrypting and *Encrypt Files* provides the option to 'shred' the plaintext file. However, *GNU Privacy Guard* had neither and the plaintext file needs to be either manually deleted or erased with other software. The availability of the original plaintext is therefore implementation specific.

Due to the manual encryption process, these programs could not be used to encrypt the pagefile, hibernation file or Registry. Also, since the encryption is

designed for single files, both hidden partitions and free space between partitions could not be encrypted with this type of encryption software. Some files in the user folders could be encrypted, but any files in use by *Windows* e.g. `\Cookies\index.dat` could not be encrypted because the manual file decryption does not allow the operating system transparent access to the file.

Regarding the approaches for attempting to access encrypted evidence discussed earlier, in terms of locating unencrypted copies of encrypted data, for these manual file encryption packages the availability of the original plaintext is implementation specific, depending on whether the original is deleted or wiped.

Encrypted files are decrypted in their entirety to files on disk while in use (either taking the form of temporary files (*AXCrypt*, *GNUPG*) or the file is changed to its decrypted form permanently (*EncryptFiles*)). The temporary files then are either deleted or erased after use. Temporary files generated by other applications are likely to be deleted only, since other applications are not aware of the sensitive nature of the files they have opened, and as a result may be useful for obtaining unencrypted copies of encrypted data.

In terms of locating passwords, due to limitations of the scope of the manual file encryption software, any files that have not been manually encrypted are accessible, and text from them can be used as possible passwords. The inability of this type of encryption software to encrypt locations such as the Registry hives means that these locations can be used to obtain possible passwords or personal data from which passwords can be derived.

4.4.2 Folder Based Encryption Results

Encrypting File System (EFS)

The *Encrypting File System (EFS)* provides the ability to encrypt files on NTFS file systems. When files in NTFS are flagged as encrypted, as shown in Figure 11, they are transparently encrypted without the need to enter a password or provide keys since this information is recovered from the Registry using the user's *Windows* logon password (Microsoft, 2006c).

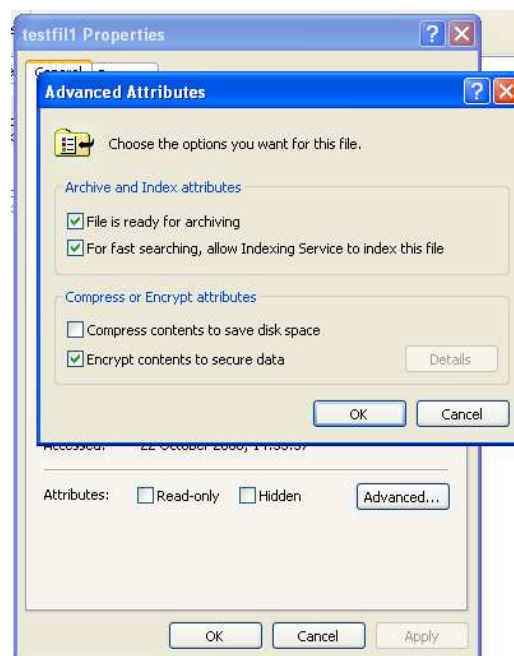
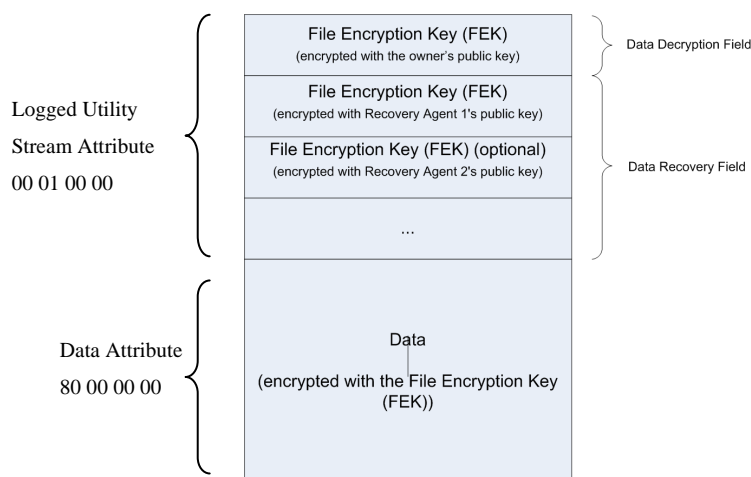


Figure 11: Advanced attributes allowing the encryption of files using *EFS*.

It is also possible to flag an entire folder as encrypted, meaning that files stored within that folder will also be automatically encrypted. When a file is encrypted, both symmetric and asymmetric encryption is used and when the file is stored with the structure shown in Figure 12 (Microsoft, 2006d).

Figure 12: The structure of an *EFS* file (Microsoft, 2006d).

The results for the offline examination of *EFS* systems are described below in Table 11.

Location	Results
Temporary Files	Temporary files generated inside the encrypted folder were also automatically encrypted. However, if the temporary files were generated in a folder that did not have the 'encrypted' attribute, then they were accessible to an offline analysis. In addition, there is a related implementation problem; when a single file is encrypted, a temporary copy is made of the plaintext named EFS0.TMP. When encryption is completed the file is deleted but not erased (Carrier, 2005 p.290). This can result in additional temporary copies of the plaintext being available on the disk.
Paging Files	The encrypted attribute could not be applied to the pagefile in <i>Windows XP</i> . However, " <i>Windows Vista</i> also supports encryption of items previously either impossible or not easily accomplished in <i>Windows XP</i> " (Morello, 2007). This includes the pagefile, and due to this significant difference between operating systems, another experiment was performed. It was eventually possible to encrypt the pagefile under <i>Windows Vista</i> using <i>EFS</i> .
Slack Space	The buffer slack of encrypted files consisted of random data, which is assumed to be encrypted. However, the cluster slack was not. The cluster slack was unchanged after encryption and contained the

	contents of previously deleted files.
The Recycle Bin	Contents of the Recycle Bin were generally accessible. When encrypted files were deleted they were moved to the Recycle Bin and the existence of the files and their metadata was available. However, the contents of the files remained encrypted. This is because the files were not actually moved and the encrypted data remained in the same location on disk, but the Parent ID of the entries in the Master File Table (MFT) were updated to reflect that they were now in the Recycle Bin.
Deleted Files	Once the Recycle Bin was emptied, the encrypted data from the deleted files was still present on the disk until overwritten by new data. However, recovery of deleted <i>EFS</i> files from unallocated space is difficult since carving is ineffective due to the content consisting of random data. Recovery therefore relies on finding the MFT entry for the deleted files.
The Registry	The Registry hive files could not be encrypted with <i>EFS</i> .
User folders	The encryption attributes were applied to the folder C:\Documents and Settings\Chris, but there were a number of files and folders that could not be encrypted because they were ‘currently in use’, including NTUSER.DAT, and index.dat in the cookies folder.
Hibernation Files	The hibernation file could not be encrypted with <i>EFS</i> .
Hidden Partitions	Any partitions that were not visible to <i>Windows</i> could not be encrypted with <i>EFS</i> .
Free Space between Partitions	Free space between partitions could not be encrypted with <i>EFS</i> since is not a file or folder and <i>EFS</i> encryption attributes could not be applied.

Table 11: Results from *EFS*.

Summary of Folder Based Encryption

Only one product was found to belong to this category without examining file system level encryption from other operating systems, e.g. *Private Folders* in *Ubuntu 8.10* or *FileVault* in *Mac OS X*. In terms of recovering unencrypted copies of encrypted data, the encryption mechanism may or may not produce temporary copies in decrypted form, depending on whether the ‘encrypted’ attribute is applied to a single file or the

folder in which a file is stored. Temporary unencrypted copies of the encrypted data that are generated by other applications may or may not be available depending on the location in which they are produced.

Due to limitations in the scope of *EFS*, a number of locations that could contain information for password attacks are available, including pagefile, deleted files, the Registry and certain files in user folders.

4.4.3 Virtual Disk Based Encryption Results

Pretty Good Privacy (PGP)

PGP Corporation (2008) provides a range of products that offer different combinations of features. One of the features is *PGP Virtual Disk* which allows files and folders to be stored in a single encrypted file which can be mounted as a regular drive letter. This feature was examined in *PGP Desktop 9.5.3*. A container file was created and mounted as G:\, as shown in Figure 13. The plaintext file was generated in the root of the virtual drive. The results of the examination of PGP Desktop are described in Table 12.

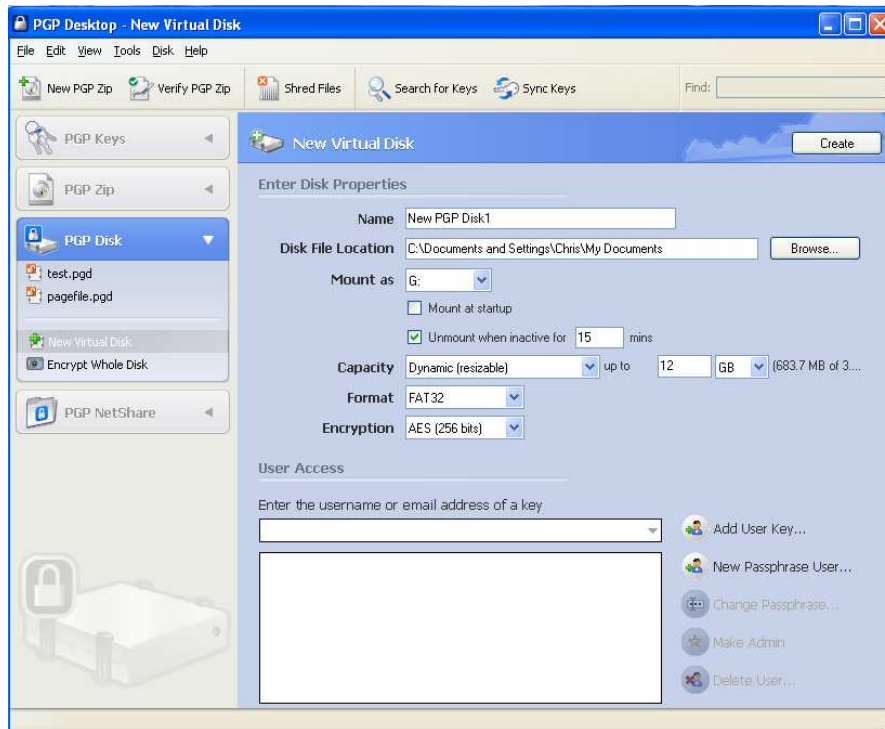


Figure 13: PGP interface for creating a new virtual disk.

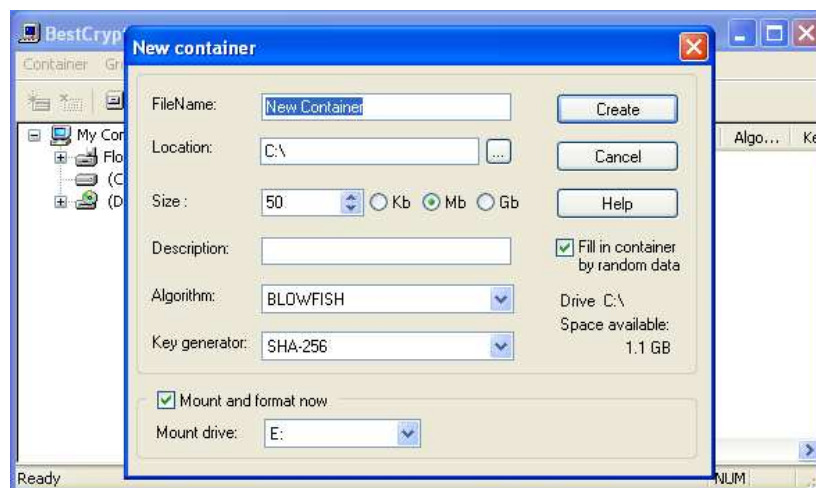
Location	Results
Temporary Files	When temporary files were generated on G:\ they were not accessible to an offline analysis using <i>X-Ways Forensics</i> . However, <i>gentemp</i> was also run from the Desktop and the temporary file produced there was accessible. Temporary files were therefore accessible if they were produced outside of the encrypted container.
Paging Files	The pagefile could not be successfully configured to reside on the virtual encrypted disk. Therefore, the pagefile was always outside the virtual disk and therefore accessible to offline analyses.
Slack Space	An examination with <i>X-Ways Forensics</i> revealed that both the buffer and cluster slack on the virtual disk were encrypted, but slack space on C:\ was still accessible.
The Recycle Bin	The Recycle Bin on the live system is a combination of the hidden 'Recycled' folders from all available hard disks (Microsoft, 2007a). Files deleted from the virtual drive appeared in the Recycle Bin on the live system. However, to an offline analysis, contents of the 'Recycled' folder on C:\ were accessible but not from the

	virtual drive.
Deleted Files	Files that were deleted on the virtual drive were in unallocated space on that drive. However, to an offline analysis, the unallocated space in the container was also encrypted and therefore deleted files from G:\ were inaccessible. However, deleted files on C:\ were accessible until they were overwritten.
The Registry	The hive files that make up the <i>Windows</i> Registry were accessible to an offline analysis and could not be encrypted.
User folders	The path for 'My Documents' could be changed to the mounted encrypted drive. However, only the Pictures and Music folders were moved and application data and other settings remained on C:\.
Hibernation Files	The hibernation file was accessible to an offline analysis.
Hidden Partitions	Hidden partitions were available.
Free Space between Partitions	Free space was also accessible.

Table 12: Results from PGP.

BestCrypt 7.20.2

Like *PGP*, *BestCrypt* “creates and supports encrypted virtual disks, which are visible as regular disks with corresponding drive letters” (Jetico, 2008). The *BestCrypt* interface is shown in Figure 14 and the results from experiments shown in Table 13.

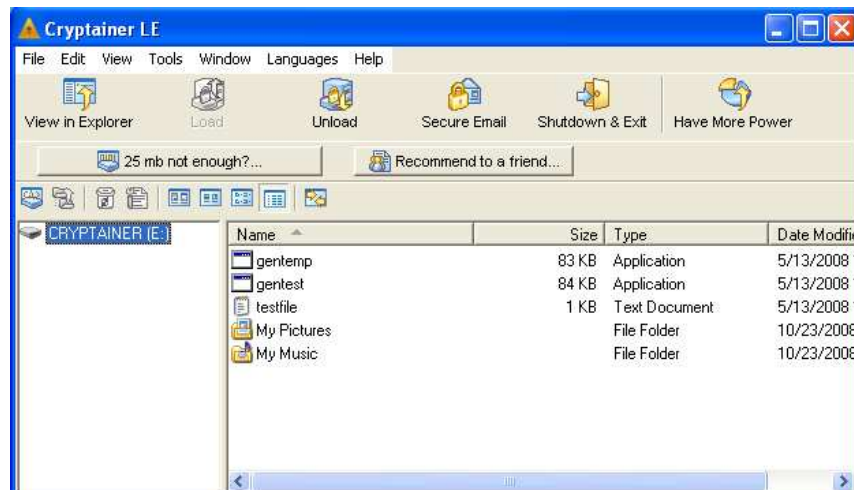
Figure 14: *BestCrypt* interface for creating a new encrypted container.

Location	Results
Temporary Files	Temporary files generated inside the virtual disk were not available to an offline analysis. However, plaintext was found if temporary files were generated on drive C:\.
Paging Files	It was not possible to successfully configure the pagefile to reside on the encrypted virtual drives. As a result, the pagefile was stored on C:\ and was therefore was accessible to an offline analysis.
Slack Space	The slack space of data stored on the virtual disk was encrypted and inaccessible but slack space on the remainder of the hard disk was available to an offline analysis.
The Recycle Bin	Deleted files went to the Recycle Bin on the live system. To an offline analysis the recycled folder on C:\ was accessible, but the recycled folder of the virtual drive was not accessible.
Deleted Files	Deleted files on the container were not accessible to an offline analysis.
The Registry	The hive files that make up the <i>Windows</i> Registry were accessible and could not be encrypted.
User folders	The path for 'My Documents' could be changed to the mounted encrypted drive. However, only the Pictures and Music folders were moved and application data and other settings remained on C:\.
Hibernation Files	The hibernation files were accessible.
Hidden Partitions	Hidden partitions were accessible.
Free Space between Partitions	Free space between partitions was accessible.

Table 13: Results from *BestCrypt*.

Cryptainer

This is a free encryption product that allows the creation of 25 MB container files, with a non-free option available that allows larger containers (Cypherix, 2008). *Cryptainer* can be run in 'portable' mode from a USB stick, meaning that it does not need installation. *Cryptainer* Virtual Drives are mounted as removable drives rather than fixed. The *Cryptainer* interface is shown in Figure 15 and the experimental results shown in Table 14.

Figure 15: The *Cryptainer* interface.

Location	Results
Temporary Files	Temporary files generated on the virtual drive were not found during an offline analysis. However, files generated on the rest of the hard disk were accessible.
Paging Files	The pagefile could not be set up on a removable drive.
Slack Space	The slack space of the container was encrypted but any slack space on the rest of the hard disk was accessible.
The Recycle Bin	The Recycle Bin of the hard disk was accessible but files deleted on the virtual removable drive did not go to the Recycle Bin (removable media do not have recycled folders (Fellows, 2005)).
Deleted Files	Deleted files could not be found during an offline analysis.
The Registry	The Registry was available during the offline analysis.
User folders	The path for 'My Documents' could be changed to the mounted encrypted drive, however, only the Pictures and Music folders were moved and application data and other settings remained on C:\.
Hibernation Files	The hibernation file was accessible during an offline analysis.
Hidden Partitions	The hidden partitions were available during an offline analysis.
Free Space between Partitions	The free space between partitions was available during an offline analysis.

Table 14: Results from *Cryptainer*.

Summary of Virtual Disk Encryption

All plaintext data created on the virtual disks was encrypted and no original plaintext was available. Also, when temporary files were produced on the encrypted virtual disk they were not accessible. However, any plaintext data duplicated outside of the encrypted virtual disk was accessible to an offline analysis.

Regarding the availability of data that could be used for password attacks, the pagefile could not be configured successfully to reside on the encrypted virtual disk. While the option could be set for it to be on a virtual drive, the pagefile was never actually generated since it was generated before the virtual disk was mounted. Therefore, data from the pagefile could be used for password attacks. The *Windows* Registry was also available, allowing attacks to be mounted to determine the *Windows* password which may help with determining the password of the virtual disk. It could also be used to identify the names and other metadata of files stored in an encrypted container from lists of recently accessed files. Also, while some paths in user folders, e.g. My Pictures, My Music could be moved to point to the virtual drive, Internet browser caches and other application data could not be moved to the virtual drive.

So while encrypted containers protect more locations and are less likely to leave plaintext on the disk, it is still possible for this to occur. Also many locations are accessible that could assist in identifying passwords to the encrypted virtual disk or metadata about the files on them.

4.4.4 Full Disk Encryption Results

CompuSec

CompuSec is a free Full Disk Encryption product for *Windows* and *Linux*. After installation and running through a wizard, the hard disk is encrypted. It also has pre-boot authentication where a username and password needs to be supplied before the system will boot, shown in Figure 16. The results from examining *CompuSec* are shown in Table 15.

Figure 16: *CompuSec* Pre-boot authentication.

Location	Results
Temporary Files	All temporary files generated were not available to an offline analysis.
Paging Files	The pagefile was not accessible.
Slack Space	Slack space (RAM or cluster) was not accessible.
The Recycle Bin	The recycled folder was not accessible.
Deleted Files	Deleted files were not accessible.
The Registry	The Registry was not accessible.
User folders	User folders were not accessible
Hibernation Files	The hibernation files were not accessible.
Hidden Partitions	The partition structure was visible but random data was found in all partitions.
Free Space between Partitions	Free space was also encrypted.

Table 15: Results from *CompuSec*.

BitLocker

BitLocker is the Full Volume Encryption feature built in to specific versions of *Windows Vista*. It uses the Advanced Encryption Standard (AES) to encrypt the system partition using the Full Volume Encryption Key (FVEK). The FVEK is

encrypted with AES using the Full Volume Master Key (FVMK). This key is then protected in a variety of means, depending on the mode in which *BitLocker* is used (Microsoft, 2006b). *BitLocker* operates in one of five modes, as shown in Table 16:

<i>TPM only</i>	This is the simplest scenario and the Volume Master Key is encrypted by a key protected by the Trusted Platform Module (TPM). The system will boot with no user intervention, but the disk is encrypted and will be inaccessible if moved to another system or viewed offline using another operating system (Microsoft, 2006b). This means that any disk image produced using standard techniques will produce an encrypted image.
<i>TPM & PIN</i>	Keys are protected by the TPM and a 4–20 digit PIN must also be entered with the function keys for every boot or when resuming from hibernation (Microsoft, 2006b).
<i>TPM & USB</i>	Keys are protected by the TPM and a USB storage device that contains a start-up key that must also be provided for each boot (Microsoft, 2006b).
<i>TPM & PIN & USB</i>	This mode is only available after <i>Windows Vista Service Pack 1</i> and offers “an additional multi-factor authentication method” (Microsoft, 2008b).
<i>USB only</i>	This can be used if a TPM is not enabled or not present. Startup keys are stored on a USB stick and must be provided in order for the system to boot. In this case the keys take the form of a 124 byte, hidden, read-only file, which by default has a file name of the format XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXXXXXX.BEK, where X is a hexadecimal digit (Microsoft, 2006c).

Table 16: Modes of BitLocker.

The following results in Table 17 are from running *Windows Vista Ultimate* with *BitLocker* on a virtual machine. Since the virtual machine does not have a TPM the system is configured in USB only mode. However, virtual machines also do not

recognise USB devices on start-up, and as a result the recovery key needs to be supplied to boot the machine. However, this does not affect the results.

Location	Results
Temporary Files	Temporary files generated on C:\ were not accessible. However, since only the C:\ partition was encrypted, if temporary files were generated on F:\ they were accessible to an offline analysis.
Paging Files	The pagefile on C:\ was not accessible. However, if a pagefile was generated on F:\ then it was accessible.
Slack Space	Slack space on C:\ was not accessible but could be accessed on unencrypted partitions.
The Recycle Bin	The Recycle Bin of the encrypted drive was not accessible but could be accessed on the unencrypted partitions.
Deleted Files	After files were emptied from the Recycle Bin, if they were originally on an encrypted partition then they were inaccessible, but it may be possible to access files deleted from an unencrypted partition if the data was not overwritten.
The Registry	The Registry was not accessible.
User folders	The user folders were not accessible.
Hibernation Files	The hibernation file was not accessible.
Hidden Partitions	The hidden partition could not be encrypted and was therefore accessible ²⁴ .
Free Space between Partitions	The free space between partitions could not be encrypted and was therefore accessible.

Table 17: Results from *BitLocker*.

TrueCrypt V6.0a

TrueCrypt is a “software system for establishing and maintaining an on-the-fly-encrypted volume” meaning that “data are automatically encrypted or decrypted right before they are loaded or saved, without any user intervention” (TrueCrypt, 2008d). *TrueCrypt* offers a number of advanced features, including hidden volumes, whereby

²⁴ This has changed as of *Windows Vista Service Pack 1* and other partitions can also be encrypted (Hynes 2008)

two passwords can be use to decrypt the volume; one decrypts prearranged innocent content, the other the real content. *TrueCrypt* has become a popular tool for encrypting data with over 8 million downloads (December 2008) (TrueCrypt, 2008a). At the time of writing *TrueCrypt* is at Version 6.1a, having last been updated in December 2008 (TrueCrypt, 2008f). From Version 5.0 onwards, *TrueCrypt* provided the option to encrypt the system partition/drive with pre-boot authentication (TrueCrypt, 2008f). The results from examining TrueCrypt 6.0a are shown in Table 18.

Location	Results
Temporary Files	All temporary files were not available to an offline analysis.
Paging Files	The pagefile was not accessible.
Slack Space	Slack space was not accessible.
The Recycle Bin	The Recycle Bin was not accessible.
Deleted Files	Deleted files were not accessible.
The Registry	The Registry was not accessible.
User folders	User folders were not accessible.
Hibernation Files	Hibernation files were not accessible.
Hidden Partitions	The partition structure was visible but all partitions contained random data.
Free Space between Partitions	Space between partitions contained random data.

Table 18: Results from TrueCrypt.

Summary of Full Disk Encryption

These tests revealed a subtle difference between Full Disk Encryption (FDE) and Full Volume Encryption (FVE), where in the case of *BitLocker* (a FVE product), only volumes/partitions are encrypted, meaning that some plaintext could be accessible to an offline analysis on other partitions and between partitions. Use of Full Disk Encryption meant that all partitions and the space between were encrypted. However, even the use of Full Disk Encryption did not mean the entire disk was encrypted, since code needed to decrypt the drive was accessible. Therefore, Full Disk Encryption encrypts all partitions and the space in between, whereas Full Volume

Encryption encrypts only the partitions. If this distinction is made between Full Disk and Full Volume Encryption then there are actually five categories of encryption product rather than four.

For all the Full Disk Encryption products examined, no plaintext data was found to be accessible to an offline analysis. Also, no locations were available that could have produced data that could be used to launch password attacks. However, with Full Volume Encryption products it is possible that plaintext could be located on the unencrypted partitions.

4.5 EVALUATION

This chapter has categorised encryption products into five types based on the scope of the encryption. It has considered how three of the eight approaches for gaining access to encrypted evidence that were discussed in Chapter 2 are affected by the scope of the encryption and therefore the category of product in use. The limited number of approaches considered is due to four of the five remaining approaches being dependent on specific product implementations or the individual investigation. Research into these could therefore not be generalised and in order to keep track of whether particular approaches would be successful at gaining access to encrypted information, it would be necessary to create a database of individual encryption products. This would need to include information such as whether a product uses an insecure algorithm, and could therefore be used to determine if offline access to encrypted data would be later possible. Producing and maintaining such a database has many potential problems including keeping the information up to date and controlling access. This is considered to be outside the scope of this research. There are also difficulties in generalising for approaches that are investigation specific e.g. persuade the suspect to provide decryption keys. In some cases this may be possible, but in others a suspect may be uncooperative. Predicting this is difficult and error prone and therefore is not useful in determining if a live investigation preserves more digital evidence in accessible form than relying on offline approaches.

The remaining approach of surveillance is likely to be generalisable based on the scope of the encryption in use, e.g. it may not be possible to install surveillance software when Full Disk Encryption is in use. However, this is not considered in this research since there is limited public information on software surveillance techniques in use, and in all cases hardware techniques are possible.

This chapter has also assumed that if encrypted evidence is preserved in a form that is accessible then completeness has been increased. However, this has the limitation of failing to consider loss of evidence due to live techniques applied, which is considered in Chapter 5.

4.6 CONCLUSIONS

This chapter has shown that there are different categories of encryption software that can be found on a system. Broadly speaking, each category leaves different locations available to an offline examination. It has examined the four categories in literature and found a subtle distinction that means Full Disk Encryption should be separated into Full Disk Encryption and Full Volume Encryption, which encrypt the entire disk²⁵ or entire partition respectively. Therefore, five categories of encryption product have been identified.

Section 4.4.1 showed that the use of manual file encryption means that any data that has not been manually encrypted can be examined by an investigator. Furthermore, the manual decryption process prevents many files from being encrypted in this way. If manual encryption is found on a system then a significant amount of information is available in order to attempt to gain access to the encrypted material. Furthermore, for the three products examined, if data was available to a live investigation, if the power was removed instead of performing a live acquisition, the unencrypted data would still be available. This was because the temporary copies of the encrypted data were stored as files on the disk of the system. However, it is

²⁵ With the caveats discussed earlier about the software needed to decrypt the drive remaining unencrypted.

possible that other single file encryption products could store data in memory instead of on disk. Therefore, it may be possible that a live investigation is not necessary for this category, but this is likely to be product specific.

Section 4.4.2 showed that folder based encryption allows a folder to be given the ‘encrypted’ attribute, meaning that all files created or moved to that folder are automatically encrypted. While there are limitations to the files and locations that can be encrypted in this way, the limitations are far fewer than for manual file encryption. Due to the folder encryption implementation that was examined (*EFS*) the keys needed to decrypt files are stored on the system under investigation²⁶, albeit encrypted using the user’s logon password as a key. Therefore the security of *EFS* protected files is dependant on the password used by the suspect. Password cracking software such as *OphCrack* (Objectif Sécurité, 2008) and *Cain* (Oxid.it, 2008) can be used to perform dictionary, brute force and rainbow table attacks on the password hash from the Registry in an attempt to recover it. Once the password is obtained, this can be used in conjunction with the encrypted version of the user’s private keys stored in the Registry to gain access to the encrypted files. Therefore, attacks on *EFS* encrypted files (the only folder based encryption examined) are possible, but are dependent on the strength of the password used and the specific settings of how the password hashes are stored²⁷.

Virtual disk based encryption prevents access to the plaintext of files stored on the virtual disk and the original copies of files are not accessible since files are automatically encrypted when they are created (see Section 4.4.3). Also no temporary files were generated by the decryption process since data is decrypted in blocks into memory as it is needed. However, temporary files produced by other applications, if generated outside of the container, may be accessible to an offline analysis. Since the pagefile cannot be encrypted in this way, unencrypted data may also be found here

²⁶ Only if the machine is not part of a domain, in which case there may be cached password hashes stored in HKLM\SECURITY\CACHE\NL\$1 to NL\$10 and a dictionary attack can still be used (Pilon 2005, Irongeek 2008)

²⁷ See Hargreaves *et al.* (2008) for details of the differences between password cracking on *Windows XP* and *Windows Vista*.

and in the hibernation file. There are also opportunities for password recovery from the pagefile and hibernation file, along with several areas in the user data folders, e.g. browser cache, which were not relocated when the 'My Documents' folder was moved to the encrypted virtual disk. It may be possible to move some application data to an encrypted location, but this would need to be manually configured inside the software in question. The *Windows* Registry was also available, potentially allowing names of files created inside the container to be obtained from lists of recently accessed files or programs run. Therefore, there is some information available that could be used to launch attacks on virtual disk based encryption products and success of these attacks is based on the password used for the virtual disk and on which applications have been used to open the encrypted data.

This research has highlighted a difference between Full Volume and Full Disk Encryption. In Section 4.4.4 it was shown that for Full Volume Encryption information on the partition that is encrypted is inaccessible. However, if there are multiple partitions and the other partitions are not encrypted then there may be temporary files available or information that can be used as the basis for intelligent password attacks. Volume slack is also available which may contain information from previously deleted partitions that could be recovered. However, it is also possible that Full Volume Encryption could be used with only a single partition which fills the disk. In this case the situation is the same as Full Disk Encryption.

Full Disk Encryption is also discussed in Section 4.4.4. Full Disk Encryption products prevent the offline approaches considered here from being used. In some cases a password attack could be launched, but a disk that has been fully encrypted cannot be used to recover information upon which to launch an intelligent password attack. The success of offline approaches is therefore much less likely when Full Disk Encryption products are in use. Even with the introduction of legislation requiring the disclosure of keys (United Kingdom, 2000), some products even offer duress key/hidden operating system functionality allowing one key to disclose the true operating system and another to reveal a false one (TrueCrypt, 2008c). There are other practical approaches, for example, locating unencrypted copies of backup data or

locating recovery keys on paper or CD that allow the decryption of the drive. However, these are case dependent, product specific, and are not guaranteed solutions.

Therefore, in some cases (particularly Full Volume and Full Disk Encryption) offline approaches have been demonstrated to be unlikely to succeed and in these cases the completeness of the preserved evidence will be significantly reduced. Even with other product categories, the success of offline approaches is dependent on what applications have been used to access encrypted content and the strength of the user's password. Therefore, predicting whether it is possible to access encrypted data offline involves many variables and is difficult and prone to error. As discussed in Chapter 2, live acquisitions can be used to acquire encrypted data in a form that is accessible. Live investigations are therefore an effective method of preserving evidence that may otherwise not be accessible, and therefore can increase the completeness of the preserved evidence. However, live investigations are not perfect solutions since live tools and techniques are intrusive, meaning that they make changes to the system under investigation. Since the requirement states that *it should be possible to assess which evidence is preserved and which is lost, also the maximum amount of digital evidence relevant to the investigation should be preserved*, the need to assess what evidence is overwritten by performing a live investigation is discussed in Chapter 5.

CHAPTER 5: COMPLETENESS AND INTRUSIVENESS

5.1 INTRODUCTION

The completeness requirement described in Chapter 3 states that *it should be possible to assess which evidence is preserved and which is lost, also the maximum amount of digital evidence relevant to the investigation should be preserved*. The previous chapter demonstrated that during digital investigations involving encrypted evidence, a live response can preserve significantly more evidence than a traditional ‘pull the plug’ investigation. However, if a live investigation is performed then changes will be made to the suspect’s system since live techniques are inherently intrusive. This will result in some data being overwritten and therefore lost. Since the requirement states that ‘it must be possible to assess which digital evidence is preserved and which is lost’ it is therefore necessary to be able to determine which data has been overwritten by using live investigation techniques on a system. The ability to assess this also has implications for preserving the maximum amount of relevant digital evidence on a system since different live techniques will make different changes to the system. As a result an investigator needs to know the likely changes that will be made to the system in order to determine the most appropriate technique that will overwrite the least relevant data in the current investigation. This chapter develops a method for monitoring the changes made to test systems by live tools and techniques. The results of such experiments can assist an investigator in assessing the changes made to a system post-live investigation, and can also provide the knowledge needed for investigators to determine the most appropriate course of action during a live investigation in order to preserve the maximum amount of relevant digital evidence.

5.2 BACKGROUND

5.2.1 Introduction

In addition to the explanation of the completeness requirement mentioned in the introduction, Chapter 3 also discussed that in a traditional digital investigation, completeness can be assessed by adhering to guidelines that predetermine the scope of the investigation, i.e. by removing the power from a live machine the contents of the hard disk are exactly preserved, but the contents of memory and other volatile data are lost. The previous chapter showed that performing a traditional digital investigation when encrypted evidence is present may not result in the maximum amount of relevant digital evidence being preserved if offline access to encrypted evidence is not possible. It also explained that performing a live investigation can preserve encrypted evidence in an accessible form, and can therefore increase the completeness of the preserved digital evidence. However, live investigations will cause changes to the live system and as a result some data will be overwritten and therefore lost. This is unavoidable since live digital investigation techniques are intrusive, meaning that they change or overwrite potentially relevant digital evidence. There is no simple way to prevent changes to a live machine, since the write blocking approach (that physically preventing writes from being made to the disk) used in traditional digital investigations cannot be used during live investigations. As a result there is “no way to avoid making changes, since in order to conduct a live examination it is necessary to deploy tools on the live system to capture data, and such tools will make changes to the running system” (Sutherland *et al.*, 2008). The amount of change caused will vary, depending on hardware and software configurations (Vidas, 2007), but even when just attempting one of the simplest of live responses, acquiring a memory image, “no software tool is capable of capturing the image of memory without, by the very act of its own execution, changing the content of memory” (Huebner *et al.*, 2007).

The completeness requirement from Chapter 3 specifies that it should be possible to assess what evidence has been lost, and also that the maximum amount of digital evidence that is relevant should be preserved. Therefore, the changes caused by

using live tools and techniques should be capable of being assessed. Also an investigator should be able to determine which live techniques will minimise the amount of potentially relevant digital evidence that is overwritten and therefore lost in individual investigations.

5.2.2 Existing Solutions

It is suggested in Request For Comments (RFC) 3227 that in order to minimise the loss of digital evidence, collection should be performed in ‘order of volatility’ (Network Working Group, 2002), collecting the most volatile first, working towards the least volatile. An example order of volatility is provided in RFC 3227 for a typical system:

- Registers and cache
- Routing table, arp cache, process table, kernel statistics, memory
- Temporary file systems
- Disk
- Remote logging and monitoring data that is relevant to the system in question
- Physical configuration, network topology
- Archival media

Collection in this way attempts to minimise the loss of digital evidence by acquiring data in a particular sequence. However, it does not address the need for an investigator to be able to assess which evidence has been preserved and which has been lost due to the techniques used, i.e. explain the consequence of their actions (ACPO, 2007).

The need to assess what has been lost is explained in ACPO (2007) as “[an investigator] must be able to give evidence explaining the relevance and the implications of their actions”. It also states that “by profiling the forensic footprint of trusted volatile data forensic tools, an investigator will be in a position to understand the impact of using such tools and will therefore consider this during the investigation

and when presenting evidence”. ACPO (2007) does not suggest any specific means for determining the footprint of tools; however, it can be achieved by monitoring the changes made in test environments. If other literature is consulted, there are a number of techniques that can be used to monitor a test system and to record changes made by running a tool or performing a particular technique. These are described in the following section.

5.2.3 System Monitoring Tools and Techniques

This section describes three current tools and techniques that can be used for monitoring changes to a system and that could be used in a test environment to determine the forensic footprint of live investigation tools and techniques. The section concludes with a summary of the limitations of these three techniques.

Live Logging Tools e.g. *Filemon*, *Regmon*, *Procmon*

The live system monitoring tools *Filemon* and *Regmon* (Russinovich and Cogswell, 2006a, b) can be run on a live system and used to record events relating to the file system and Registry respectively. The use of these tools has been described for monitoring changes to a system when dynamically analysing malicious software (Carvey, 2005). They can also be used during digital investigation research to investigate possible locations of artefacts left by particular pieces of software (Dickson, 2006a, b, c, 2007, van Dongen, 2007). These monitoring tools install drivers to log events, for example *Filemon* installs *filem.sys* which attaches to the device object for the mounted file system and intercepts and records file system requests (Russinovich and Solomon, 2005 p.706). Since they all install drivers, they therefore make their own changes to the system they are monitoring.

The successor to *Filemon* and *Regmon* is *Procmon* (Russinovich and Cogswell, 2008) which works in a similar way but offers a number of improvements, including simultaneously recording both file system and Registry changes. *Procmon* records a massive amount of data in the form of extensive logs containing details such as the time of the event, process name, operation performed, path, etc. To illustrate the

extent of these logs, a run on an idle *Windows XP* system for 1 minute generated a log containing over 14,000 events. To process these large logs, advanced filters which form part of the software can be used to reduce the logs and monitor individual events, files or any other detail of interest. *Procmon* was used in Evans (2007) and Sutherland *et al.* (2008) to document changes made to test systems by some live tools, including: memory acquisition tools, e.g. *dd*; network status tools, e.g. *fport*; and, system status tools, e.g. *psinfo*.

Forensic Package and Sort by Modification Date

Another technique that can be used to detect changes made to a system by tools, techniques or malware under examination is, after the event, to conduct an examination of the disk of the test system using forensic software and to sort files on the disk by their attached metadata, i.e. modification date. This can be used to highlight files created and modified on the system within the time period in which the test was conducted. This technique has the advantage of being unintrusive since it can be run retrospectively on a disk image from the system under examination.

InCtrl5

InCtrl5 is a piece of monitoring software that allows the recording of changes made to a system during the installation of new software (Rubenking, 2000). The *InCtrl5* program requires installation prior to monitoring a software installation, but is then able to record changes “by running installation programs from within its tracking system” (Rubenking, 2000). It can also be used to monitor other changes to a system by creating snapshots before and after a certain action. *InCtrl5* produces a HTML report that reports Registry and file changes, including the stated modification date and size.

Limitations of these techniques

Both the live logging tools and the snapshot based approach of *InCtrl5* are intrusive, since they run on the system being monitored. This means that to determine the

changes caused by the tools being tested, it is necessary to filter out the changes caused by the monitoring tools themselves. This can be performed either by the tools monitoring their own changes (*Procmon*), or by examining accompanying documentation²⁸ (*InCtrl5*). The other issue with intrusive monitoring tools is that if changes to the memory of the test system are also of interest, e.g. how much data in memory has changed, the running of live monitoring tools makes this impossible to ascertain without running additional tests.

The only un-intrusive technique described is the ‘sort by modification date’ technique, which has the separate problem of relying on Modified Accessed Created (MAC) times. This is a problem since, as Carrier (2005 p.12) points out, there is essential and non-essential data on a system, where essential data must be accurate in order for the system to function. Dates and times unfortunately fall into the non-essential category and can be easily modified without affecting the operation of the system. This is particularly problematic in malware analysis since deliberate alterations of dates and times could take place to avoid detection. The *InCtrl5* snapshot based monitoring technique also has the limitation of failing to record changes that are made after the first snapshot but are undone before the second, e.g. the creation and removal of temporary files between snapshots. Also *Procmon* has been found to sometimes miss certain changes to the system being monitored, e.g. files related to restore points (Hargreaves, 2007).

Since all monitoring methods can miss changes, another problem with these techniques is that it is difficult to validate the results from a single monitoring method. Also, no documented testing of any of the current techniques could be found, despite *Procmon* being used extensively for system monitoring, even in the forensic computing community. This lack of testing may be due to the challenges involved with attempting to test these monitoring techniques; if the tools are used concurrently on a system, attempting to compare the results is difficult since they produce output in very different formats that is difficult to cross-check. Also, since two of the

²⁸ The modifications caused by the installation of *InCtrl5* are listed in the accompanying readme.txt included with the installation program. However, *InCtrl5* does not monitor its own installation changes (Rubenking, 2000).

techniques are intrusive, if tools are run concurrently, it then becomes necessary to filter out two sets of changes from the reports generated by the three techniques.

Nevertheless, validation such as this should be performed since, as described in Carrier (2003), it is important to verify the results from any digital investigation tool, which can be done either manually or using a second tool. While these monitoring methods are not digital investigation tools, if they are used to profile changes made by live investigation tools and techniques, the results could be used to determine if the techniques overwrite potentially relevant digital evidence and to determine the best course of action for a live investigation. It is therefore important that the results are correct and comprehensive and they should therefore be verified to the same extent as results from digital investigation and forensic tools.

If monitoring methods fail to record changes that are made to the system, the investigator will be unaware of potential digital evidence that will be lost due to the live techniques and may perform an investigation and overwrite relevant evidence. Also, if changes are recorded that are caused by the monitoring techniques and are wrongly attributed to the live investigation techniques, then an investigator may choose not to perform a live investigation when it could have been used, and therefore digital evidence that could have been preserved using live techniques may be lost.

So, while these limitations are not as significant for previous uses of system monitoring (e.g. indicating where evidential artefacts of software may reside), for the purpose of profiling a live technique's footprint on a system these limitations are relevant and it is important to know that the changes recorded in tests are correct and comprehensive.

5.2.4 Summary

Live investigation techniques are inherently intrusive, i.e. they will make changes to the system under investigation. This could affect the completeness of the amount of potentially relevant digital evidence that is preserved. To minimise this loss, evidence can be collected in order of volatility. However, this does not address the need of an investigator to identify and quantify the evidence that has been lost due to the live

techniques used, i.e. to explain the consequences of their actions. It also does not address the subtle differences in the data that is overwritten by using different live techniques. In order to do this, monitoring techniques can be used on test systems to determine the changes that investigators are likely to make to a system due to the use of live tools and techniques.

Monitoring a system can be performed using a number of methods: live logging tools such as *Procmon*; sorting files by modification date; or using a snapshot approach such as *InCtrl5*. However, individually these techniques have limitations that could result either in changes failing to be recorded or additional changes being recorded that are due to the monitoring method itself. While these changes are not as significant for previous uses of system monitoring, when used for determining the changes made by live investigation tools and profiling their footprints, a more robust methodology is necessary.

5.3 METHODOLOGY

5.3.1 Introduction

This section describes the methodology used in this chapter. As shown in the previous section, it is necessary to determine the changes made to a system due to live investigation tools and techniques. This information enables an investigator to determine what course of action to take (i.e. which methods to use) during an investigation in order to preserve the maximum amount of potentially relevant digital evidence. It also assists an investigator in assessing what digital evidence has been lost after the live investigation has been performed. These aims can be achieved by profiling the footprint of live tools, i.e. determining what changes they make in a test environment. Existing methods of monitoring changes on a system were described in the previous section, but they were all shown to have limitations, including their intrusive nature and the difficulty comparing results from different methods. In this section, first the general methodology used to identify changes caused by live investigation tools and techniques is outlined, including the justification for the ‘footprinting’ approach. Following this, the selection of live tools and techniques that

are profiled is explained. The actual development of a new system monitoring technique is described later in Section 5.4.

5.3.2 General Methodology

This chapter has so far explained the need to determine the changes caused by live tools and techniques in order to assess what digital evidence has been lost, and also to allow the most appropriate course of action to be taken during a live investigation. Determining the best course of action in an investigation is a function best performed by the investigator, due to the specific requirements of each individual investigation. However, it has been shown that it is possible to monitor live tools in a test environment and document the changes they cause. Using the results of tests performed, an investigator will be able to combine the information gained from testing tools with their knowledge of the specific case to more effectively determine the best course of action in an investigation. Also, monitoring of live tools in a test environment will allow an investigator, post-live investigation, to more effectively determine what changes were made. This is because monitoring test environments will provide a greater understanding of changes that normally occur on a system and changes that are likely to be attributable to live tools. The research in this chapter therefore focuses on identifying changes caused to test systems by live tools.

As explained in the previous section, there are several existing methods for monitoring systems which could be used to identify changes caused by live tools. However, as explained earlier, they have limitations, including the possibility of missing changes, the results being difficult to correlate, and consequently the results being difficult to validate. So, while the research in this chapter is concerned with identifying changes to test systems, it specifically focuses on developing a new system monitoring methodology that overcomes the limitations of the existing techniques. Therefore, while individual results from testing live tools will be obtained in the course of evaluating the new methodology, the main aim is the production of a methodology that will allow changes to be identified in a test environment.

Although building a catalogue of the footprints of live tools is outside the scope of this research, a range of tools will be tested in order to evaluate the effectiveness of the methodology. This testing will also provide a starting point to the rigorous testing that is necessary for the extensive and ever changing range of live tools and techniques on a number of operating systems in a number of different configurations. The choice of initial test scenarios is described in the following section.

5.3.3 Testing Live Tools and Techniques

The methodology developed in Section 5.4 is used to profile the footprint of a number of live tools and techniques in a basic *Windows XP SP2* environment. This section describes the choice of live tools and techniques that are tested.

Live digital investigation techniques are tested based on the overall methodology for running live investigation tools described in Wait (2006), which involves:

- 1) establishing a trusted command prompt;
- 2) establishing a method for transmitting and storing the collected information;
- 3) running various tools and creating hashes of the output.

The first stage is establishing a trusted command prompt from which to launch tools. However, live tools may not necessarily be launched in this way since some tools are launched directly, e.g. the *EnCase Enterprise* servlet and *FTK Imager*. Therefore, a number of different means of launching a program are considered. In this research three techniques are investigated: double clicking an executable; using Start->Run and typing the path and program to be executed; and then finally opening a trusted command prompt and launching a program from it. Monitoring is performed from the point at which the prompt is already running, in order to see the effect of launching programs from the prompt, not of launching the prompt itself. All of these techniques are tested by running simple 'hello world' programs, which are compiled in both 32 and 16 bit environments using *Visual Studio* and *DJGPP* respectively.

Secondly, changes are investigated that are caused by connecting to the system in order to enable the transmission or storage of acquired data. Figure 17 shows the back of a PC and some common interfaces are highlighted. In this research the changes caused by connecting with USB and Firewire (IEEE 1394) are considered since USB is a popular interface for storing live acquired data and Firewire is of interest due to its direct memory access which can be used for memory imaging. Connection via USB in this research involves attaching a USB thumb drive to the monitored system. For the Firewire connection, a *Linux* based laptop is configured as it would be to perform Firewire memory imaging (emulating an *iPod*, as described in Chapter 2). In addition to making the Firewire connection, an image of the system's memory is also obtained over the connection. In addition to these connections, since live investigation tools should be run from a static media (Adelstein, 2006) the changes caused by inserting a CD-ROM into a system are investigated. Connection via Ethernet is not considered in this research since only limited testing is performed to demonstrate the use of the developed methodology, and since USB and CD-ROMs are popular tool delivery methods, and Firewire is of interest due to its potential for memory imaging, these interfaces were prioritised.

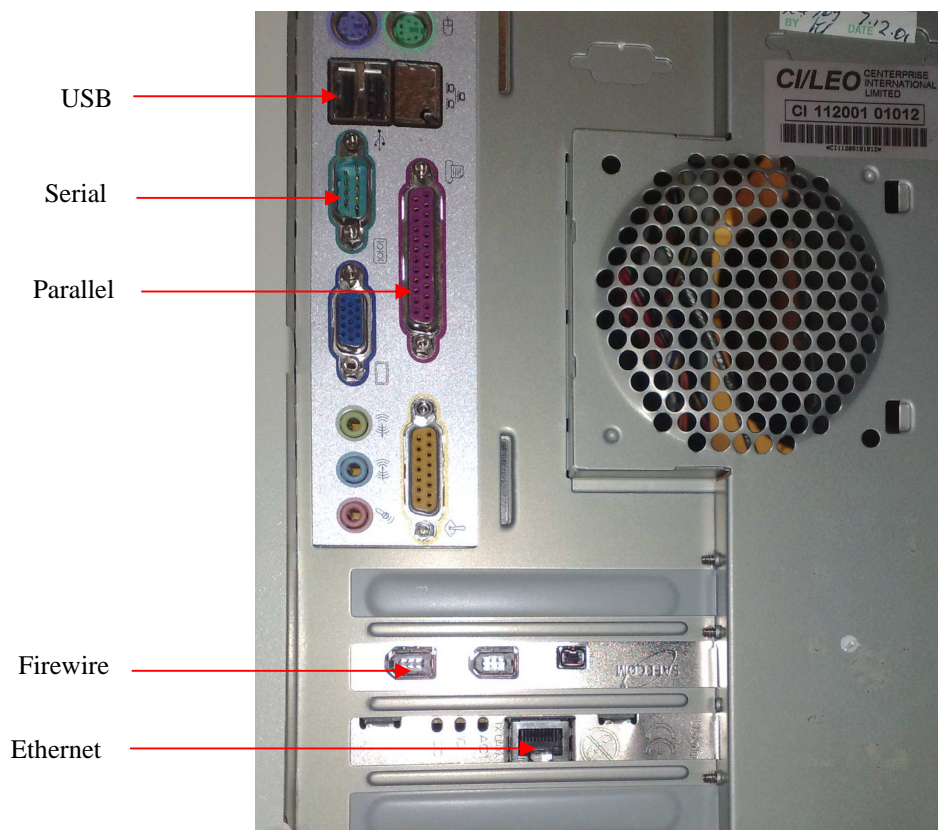


Figure 17: The back of a PC with common ports highlighted.

The final stage in Wait (2006) is running various tools to obtain information from the live system. Since changes caused by the operating system when programs are launched are investigated in an earlier stage, this focuses on changes caused by specific software that may be run as part of a live investigation. As described in Chapter 2, much of the information that can be obtained using live analysis tools such as *pstools* and *fport* etc. can now be obtained from acquired memory dumps of live systems. Therefore, these experiments include live memory acquisition tools such as *dd* and *Fast Dump*, and also results from monitoring some live analysis tools which are taken from the *Helix live CD*.

In addition, before these three live investigation stages are monitored, systems are examined in an idle state to determine background changes that occur normally on a system. For this, test systems are set up and left idle for 10 minutes, 1 hour and 24 hours. A list of background changes is produced and is used to exclude these background changes from the later tests.

5.4 DEVELOPMENT OF A SYSTEM MONITORING METHODOLOGY

5.4.1 Introduction

As mentioned in earlier sections there are a number of limitations of the currently available system monitoring techniques. These include the intrusiveness of the monitoring tools, which means it is necessary to separate out changes caused by the monitoring tools themselves and those caused by the techniques under test. They also include the risk of missing changes, either due to files being created and removed between snapshots, or relying on non-essential data. It is also difficult to compare the results between different methods. This section describes the development of a methodology that overcomes these limitations.

5.4.2 Overall Approach

The overall approach uses virtualisation to examine changes made by the live investigation techniques under test (the advantages of virtualisation were discussed in Chapter 1). The technique combines the approaches described earlier, with a significant modification to the snapshot based approach. This modification is that instead of creating snapshots on the live machine under test (as with *InCtrl5*), virtual machines (in this case *VMware*) are used so that snapshots can be created of the entire machine at particular states from outside the environment under test. This provides an unintrusive option for monitoring a test system. The modified snapshot approach still suffers from the limitation of missing changes that are made and undone between snapshots, however, this problem is addressed by also running *Procmon* inside the virtual machine between snapshots which will record all changes made, including the ones missed by the snapshot approach. This prevents the approach from being fully unintrusive, but since only one intrusive technique is used, and this can be monitored in separate tests, it is possible to filter out the changes caused by this single intrusive method. In addition, the snapshots²⁹ created in this way are much more

²⁹ The snapshots discussed here should not be confused with the VMWare 'Snapshot feature' which creates a reference point in a virtual machine's history, from which point changes are stored to a

comprehensive than other approaches since the entire disk and memory of the virtual machine is duplicated before and after the event being monitored. This full duplication of the disk allows different techniques to be applied retrospectively to determine the differences between the two snapshots, since it separates out data collection and analysis, which are discussed in Sections 5.4.2 and 5.4.3 respectively.

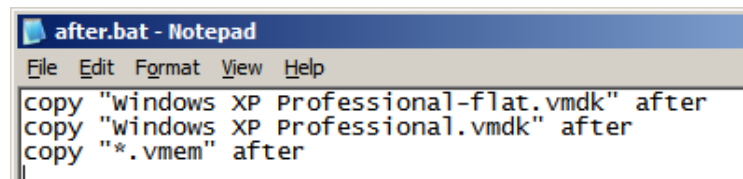
5.4.2 Data Collection

For each tool or technique to be monitored, a duplicate of a baseline virtual machine is created. The duplicated virtual machine is booted and configured to the state just prior to where changes to the system are to be monitored. The following steps describe the rest of the process.

1. At the point at which changes needed to be monitored *Procmon* is launched. However, it is not yet set to log changes. The virtual machine is paused.
2. A duplicate is created of the virtual hard disk file (.vmdk).
3. A duplicate is created of the virtual memory file (.vmem).
4. The virtual machine is resumed.
5. The *Procmon* logging is started.
6. The action/connection under test is performed.
7. The *Procmon* logging is paused.
8. The virtual machine is paused.
9. A duplicate is created of the virtual hard disk file (.vmdk).
10. A duplicate is created of the virtual memory file (.vmem).
11. The virtual machine is resumed.
12. The *Procmon* log is saved (including all events) as a *Procmon Monitoring Log* (PML) file.

separate file rather than the machine's virtual disk. The snapshots used here refer to manually created full duplicates of the files representing a virtual machine's disk and memory (.vmdk and .vmem).

For simplicity, the copying of the files needed for the before and after snapshots is performed by `before.bat` and `after.bat` batch files. The latter is shown in Figure 18.



```
after.bat - Notepad
File Edit Format View Help
copy "windows XP Professional-flat.vmdk" after
copy "windows XP Professional.vmdk" after
copy "*.vmem" after
```

Figure 18: `after.bat`, which is used to simplify creation of snapshots by copying the `.vmdk` and `.vmem` files to the 'after' subfolder.

The *Procmon* logs are recovered from the virtual machine by connecting a USB device and saving the `.PML` file to it, since changes caused by connecting the USB stick are not relevant at this stage since the 'after' snapshot disk image records the state of the machine before these changes are made. Therefore, for each test, the following data is produced.

<code>\before\Windows XP Professional-flat.vmdk</code>	The hard disk image before the event took place.
<code>\before\mem.vmem</code>	The memory image before the event took place.
<code>\after\Windows XP Professional-flat.vmdk</code>	The hard disk image after the event took place.
<code>\after\mem.vmem</code>	The memory image after the event took place.
<code>logfile.pml</code>	A <i>Procmon</i> log file of the live changes that took place on the system.

5.4.3 Data Analysis

Once the experimental data is collected, it needs to be processed to produce lists of changes caused to the system. *Procmon* logs (PML files) already consist of a list of changes; however, it is still necessary to process them in order to convert them into a format that can be combined with other methods, which simplifies later analyses. The disk images created before and after the event under test also require analysis in order to extract lists of changed files. The processing of the *Procmon* logs and also the two techniques that are used to extract file and Registry changes from the before and after disk images are discussed in the next three sub-sections.

***Procmon* Log Processing**

Due to the *Procmon* logs being saved in their complete form, i.e. including all events, it is necessary to filter them to highlight the details that are relevant. To do this, the saved PML logs are loaded into a version of *Procmon* on an analysis machine. *Procmon* filters are applied so that the logs are reduced to show only the relevant events. Different filter sets are used to separately show changes to files and changes to the Registry. File writes are filtered using the ‘Operation = WriteFile’ filter, which records any event where data is written to files. Registry changes are also filtered by the ‘Operation’ field, where events that cause writes to the Registry, e.g. creating keys and setting values, are included. The full list of Registry operations with descriptions is available in Microsoft (2008c) and the filters used to detect changes are: RegCopyTree, RegCreateKey, RegCreateKeyEx, RegCreateKeyTransacted, RegDeleteKey, RegDeleteKeyEx, RegDeleteKeyTransacted, RegDeleteKeyValue, RegDeleteTree, RegDeleteValue, RegFlushKey, RegLoadKey, RegRestoreKey, RegReplaceKey, RegSaveKey, RegSaveKeyEx, RegSetKeyValueEx, RegSetInfoKey, RegSetValue, RegSetValueEx and RegUnloadKey. Other than the filters mentioned here, the default *Procmon* filters, which exclude \$MFT, exclude Pagefile and others, are removed. After application of the filters, the subsets of the results are exported to Comma Separated Value (CSV) files, named files.csv and reg.csv. Since the logs record all changes as they occur, writes to an individual file or Registry entry can occur multiple times and therefore appear more than once in the logs. While the time of the event may be useful, for this method, only a summary list of files and Registry entries that have been changed is necessary. Therefore, the two CSV files are processed using a *Perl* script to remove fields that were not needed e.g. time of the event, sequence number, etc. and also to remove duplicate entries so that each file and Registry modification appears only once in the results.

Snapshot comparison technique

This technique identifies changes to the file system by traversing the file structure, calculating and outputting an MD5 hash of each file encountered. By generating these

lists of hashes it is possible to determine files that have changed between the snapshots. To obtain hashes, the disk images acquired before and after the event are mounted on the analysis machine and the tool *md5deep* (Kornblum, 2008) is used in recursive mode to output the hashes of the files. There are a number of ways to mount disk images: on a *Linux* system using the built in mount command, or on *Windows* using specialist software such as *Mount Image Pro* (GetData, 2008) or using *VMWare*'s disk mounting tool (VMWare, 2008)³⁰. In this research the latter is used.

The use of *md5deep* produces lists containing MD5 hashes followed by the full path of the file. These can be compared using *Windows*' *fc.exe* or *Linux*'s *diff*, but these tools are not specifically designed for outputting a list of files that have different hashes, and in the output of the tools, each change is sandwiched between lines that do not contain changes. Therefore a short *Perl* script was developed and used to compare the two lists and to report those files that have changed between the snapshots. The output of *fc.exe* and the developed script is shown in Figure 19.

```

***** J:\SYSTEM_CHANGE\HELIX_SOMETHING\RESULTS\BEFORE\hashes.txt
2f3cdc1d898fd25b2547f5bfeb01fd0d WINDOWS/winnt256.bmp
39f4afca2443fa9899ea3d6d1500391d WINDOWS/windowsUpdate.log
5a5cff37f1bd0f86b9bdaad7a9445882 WINDOWS/windowsShell.Manifest
***** J:\SYSTEM_CHANGE\HELIX_SOMETHING\RESULTS\AFTER\HASHES.TXT
2f3cdc1d898fd25b2547f5bfeb01fd0d WINDOWS/winnt256.bmp
b37ca325fe39a8d4b47be993b0990f2c WINDOWS/windowsUpdate.log
5a5cff37f1bd0f86b9bdaad7a9445882 WINDOWS/windowsShell.Manifest
*****

Documents and Settings\Chris\NTUSER.DAT
Documents and Settings\Chris\ntuser.dat.LOG
PAGEFILE.SYS
WINDOWS\system32\config\software.LOG
WINDOWS\system32\config\AppEvent.Evt
WINDOWS\system32\config\SOFTWARE
WINDOWS\system32\wbem\Logs\wmiprov.log
WINDOWS\windowsUpdate.log
WINDOWS\Prefetch\WUAUCLT.EXE-399A8E72.pf
WINDOWS\Prefetch\WMIIPRVSE.EXE-28F301A9.pf
WINDOWS\Prefetch\PROCMON.EXE-13F2CDD6.pf
WINDOWS\SoftwareDistribution\DataStore\Logs\edb.log
WINDOWS\SoftwareDistribution\DataStore\Logs\edb.chk
WINDOWS\SoftwareDistribution\DataStore\DataStore.edb

```

Figure 19: Changes made between snapshots displayed with *fc.exe* (top) and the developed *Perl* script (bottom), the latter produces a cleaner, simpler list of changes.

³⁰ This requires a workaround where, the *.vmdk* file that configures the virtual hard disk is duplicated and manually edited to reference the location of the new duplicate virtual disk.

Contents of the Registries from snapshots can be extracted using the *reg.pl* script from Carvey (2007b p.134). This script extracts the contents from Registry hives in the form shown in Figure 20.

```

\$$$PROTO.HIV\Microsoft\windows\CurrentVersion
Lastwrite time: wed Sep 12 10:33:02 2007
--> DevicePath;REG_EXPAND_SZ;%SystemRoot%\inf
--> MediaPathUnexpanded;REG_EXPAND_SZ;%SystemRoot%\Media
--> SM_GamesName;REG_SZ;Games
--> SM_ConfigureProgramsName;REG_SZ;Set Program Access and Defaults
--> ProgramFilesDir;REG_SZ;C:\Program Files
--> CommonFilesDir;REG_SZ;C:\Program Files\Common Files
--> ProductId;REG_SZ;76487-338-5610986-22153
--> WallpaperDir;REG_EXPAND_SZ;%SystemRoot%\web\wallpaper
--> MediaPath;REG_SZ;C:\WINDOWS\Media
--> ProgramFilesPath;REG_EXPAND_SZ;%ProgramFiles%
--> SM_AccessoriesName;REG_SZ;Accessories
--> PF_AccessoriesName;REG_SZ;Accessories

```

Figure 20: Registry contents extracted using *reg.pl*.

As can be seen in Figure 20, the data extracted from the Registry keys is difficult to interpret and is considerably different to the format of the *Procmon* logs, where each key is listed in full (including the sub-key written). This difference makes comparison with the *Procmon* logs difficult, and as a result another developed *Perl* script is used to format the extracted Registry data. This script converts the extracted Registry data into Comma Separated Value form so that it can be more easily read and manipulated. The output is of the format *key, last written time, type, value*, and due to its CSV format it can easily be tabulated as shown in Table 19.

Key	Last Written	Type	Value
HKLM\Software\Microsoft\Windows\CurrentVersion	Wed Sep 12 10:33:02 2007		
HKLM\Software\Microsoft\Windows\CurrentVersion\DevicePath		REG_EXPAND_SZ	%SystemRoot%\inf
HKLM\Software\Microsoft\Windows\CurrentVersion\MediaPathUnexpanded		REG_EXPAND_SZ	%SystemRoot%\Media
HKLM\Software\Microsoft\Windows\CurrentVersion\SM_GamesName		REG_SZ	Games
HKLM\Software\Microsoft\Windows\CurrentVersion\SM_ConfigureProgramsName		REG_SZ	Set Program Access and Defaults
HKLM\Software\Microsoft\Windows\CurrentVersion\ProgramFilesDir		REG_SZ	C:\Program Files
HKLM\Software\Microsoft\Windows\CurrentVersion\CommonFilesDir		REG_SZ	C:\Program Files\Common Files

HKLM\Software\Microsoft\Windows\CurrentVersion\ProductId		REG_SZ	76487-338-5610986-22153
HKLM\Software\Microsoft\Windows\CurrentVersion\WallPaperDir		REG_EXP AND_SZ	%SystemRoot%\Web\Wallpaper
HKLM\Software\Microsoft\Windows\CurrentVersion\MediaPath		REG_SZ	C:\WINDOWS\Media
HKLM\Software\Microsoft\Windows\CurrentVersion\ProgramFilesPath		REG_EXP AND_SZ	%ProgramFiles%
HKLM\Software\Microsoft\Windows\CurrentVersion\SM_AccessoriesName		REG_SZ	Accessories

Table 19: Registry contents formatted to CSV format using the developed *Perl* script displayed in table form.

Using this simpler CSV format, the differences between the Registries extracted from the before and after disk images can be more easily identified using another script.

‘Sort by modification date’ based technique

As described earlier, it is also possible to use forensic software to sort files from a disk image by their modification date and to report those which changed during the period in which the test was performed. To determine the timeframe from which to report file changes, the time could be manually recorded from the system clock at the point at which event monitoring begins. However, this can also be automated since the virtual machine’s memory is duplicated in addition to the virtual disk. This is performed before and after the monitored event and using the *Volatility* toolkit (Walters and Petroni, 2007) it is possible to recover the system time from the memory images automatically (using the *datetime* function of the *Volatility* toolkit)³¹. A challenge to using this Modified Accessed Created (MAC) times based approach is that most forensic tools are used though a graphical interface which makes automation of the process and outputting a specific format report difficult. However, *The Sleuth Kit (TSK)* (Carrier, 2009) (originally *Linux* only but also now with a *Windows* version) provides a series of command line tools for recovering information from a disk image. Using these tools, the ‘after’ disk image can be analysed using the

³¹ The dates and times could also be taken from the *Procmon* log.

command `fls -F -u -r -p -o 63 -m`,³² which indexes all the files in the disk image to a text file along with their metadata, separated by '|', including MAC times, (with Modified time in column 12, and Created time in column 13). This file is processed with a *Perl* script to parse the output from `fls`, retrieve the times from the memory images using *Volatility*, convert them to the same format as in `fls`, and then filter the results by these times.

5.4.4 Data Correlation: Manual and Automated

As described above, all the outputs from the individual monitoring methods are either generated or formatted using *Perl* scripts. This means that the format of the outputs can be controlled so they are all similar and can therefore be compiled and compared to each other to produce comprehensive and validated results.

Since this analysis is repetitive, time consuming and error-prone, and since it needs to be performed for all the experimental data collected, the process is automated as much as possible. This is achieved by the development of an automated toolkit written in *Perl*, the structure of which is shown diagrammatically in Figure 21.

³² `fls` options do the following:

- F: files only
- u: display undeleted only
- r: recursive
- p: display full path
- o: image offset (63 sectors into physical disk image)
- m: display metadata, including MAC times

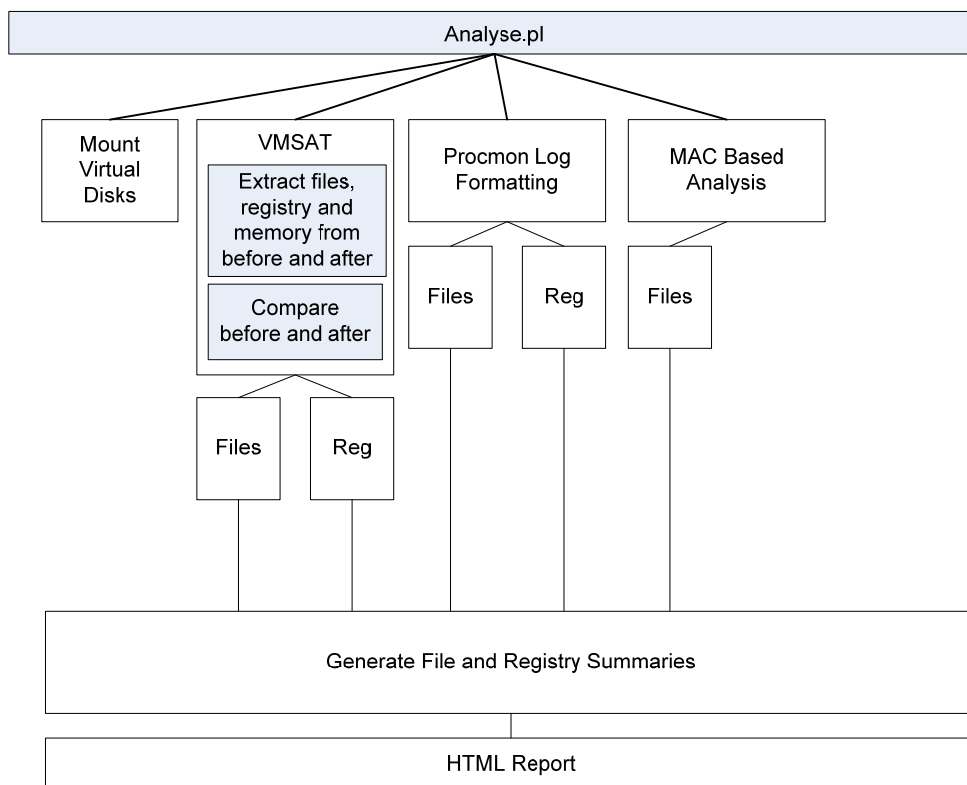


Figure 21: Simplified architecture of fully automated results processing and report generation.

The core of the automated analysis is the *Analyse.pl* script which reads a configuration file that contains a list of directories in which generated experimental data is stored. For each experiment directory, the script mounts the before and after virtual disks and calls the developed *Virtual Machine Snapshot Analysis Tool (VMSAT)* to extract the Registry and disk changes from the two snapshots. The *Analyse.pl* script also formats the *Procmon* logs and identifies changed files based on their ‘modified time’ metadata from the ‘after’ disk image. Finally, additional scripts are used to combine the results from the three methods and summarise them in an HTML report. This automation allows multiple sets of experimental data for file changes to be automatically processed and reports generated. Sample output is shown in Figure 22 and Figure 23.

System Monitoring Reports

File Reports

[File change report based on md5 hashes\(custom\) \[before\]\[after\]](#)

[File change report based on md5 hashes\(fc/diff\) \[before\]\[after\]](#)

[File change report from ProcMon \(simple\)](#)

[File change report from MAC times \(advanced\)](#)

[VSMTK Validation](#) [Procmon Validation](#)

[Summary](#)

Registry Reports

[DEFAULT\(new\) \[before\]\[after\]](#)

[SAM\(new\) \[before\]\[after\]](#)

[SECURITY\(new\) \[before\]\[after\]](#)

[SOFTWARE\(new\) \[before\]\[after\]](#)

[SYSTEM\(new\) \[before\]\[after\]](#)

[Registry change report from ProcMon \(simple\)](#)

[Registry change report \(custom\)](#)

[VSMTK Validation](#) [Procmon Validation](#)

[Summary](#)

[Summary \(exp\)](#)

Figure 22: Main index page of the generated HTML report.

Created Files	MAC	VSMTK	Procmon
WINDOWS\Prefetch\CMD.EXE-087B4001.pf	y	y	n

Modified Files	MAC	VSMTK	Procmon
WINDOWS\system32\config\software.LOG	y	y	y
WINDOWS\Prefetch\CMD.EXE-087B4001.pf	y	n	n
Documents and Settings\Chris\ntuser.dat.LOG	y	y	y
Documents and Settings\Chris\NTUSER.DAT	n	y	y
PAGEFILE.SYS	n	y	y
WINDOWS\system32\config\SOFTWARE	n	y	n
C:	n	n	y
WINDOWS\Prefetch\PLIST.EXE-08928D72.pf	n	n	y
DOCUME~1\Chris\LOCALS~1\Temp\Perflib_Perfdata_11c.dat	n	n	y

Figure 23: Sample summary of file changes in the combined HTML report.

5.4.5 Extending to Real Systems

One limitation of this methodology is that it can only be used on virtual machines. As described in Chapter 1, one of the limitations of virtual machines is that not all hardware can be virtualised, and in the case of this research, one relevant example is the Firewire port. As a result, it is necessary to adapt the technique for compatibility with real systems. The modified procedure requires tools to be run on the system under test in order to create duplicates of the disk and memory, and as a result this version of the technique is additionally intrusive. Also, since the images of memory and disk are obtained from a real system, it is not possible to ‘pause’ the machine as when virtual machines are used. Therefore, disk images acquired from real systems are not snapshots but are ‘smears’, where data may change between the start of acquiring an image and the end. Therefore, many of the advantages of the technique are lost when used on a real system. However, in a limited number of cases, this is necessary. The modified procedure is described below.

1. The test system is configured with two hard drives, one to contain the operating system and another to store images of disks and memory.
2. A baseline copy of *Windows XP* is installed to the system drive³³.
3. The system is booted to the point at which changes needed to be monitored and *Procmon* is launched (but is not yet set to log changes).
4. The *Procmon* logging is started³⁴.
5. The memory of the system is imaged, using *FastDump*, to the second drive.
6. The system hard drive is imaged live using *dd* to the second drive.
7. The action/connection under test is performed.
8. The memory of the system is imaged, using *FastDump*, to the second drive³⁵.

³³ For ease of testing, after *Windows* was installed, the baseline installation was imaged to the second drive so it could be easily restored after each test.

³⁴ In *Procmon*, the backing file for the log is set to the second drive rather than the pagefile (default).

³⁵ This can also be achieved using the Ctrl Scroll Lock method described in Chapter 2, although the *Procmon* log must be saved first.

9. The system hard drive is imaged live using *dd* to the second drive³⁶.
10. The *Procmon* logging is paused.
11. The *Procmon* log is saved (including all events) as a *Procmon* Monitoring Log (PML) file to the second drive.

In order for these disk images from real systems to be used with the automated analysis software they need to be mountable. This is achieved by converting them using *LiveView* (CERT, 2007), which converts them to VMware compatible virtual disks and allows the same analysis technique used for virtual disks to be used for disk images from real systems.

5.4.6 Summary

Due to the limitations of current system monitoring techniques and the difficulty in correlating results from multiple tools, in order to monitor the 'footprint' of live investigation tools, a more advanced system monitoring methodology is necessary. This section has described two ways in which virtualisation can be used to monitor systems externally in an unintrusive manner. First, the virtual machine can be monitored for changes using a snapshot based approach that determines whether files have changed based on their MD5 hashes calculated before and after an event occurs. Second, changes to the virtual machine's disk can be determined externally by examining the after snapshot for changed files by filtering the results by last modified date/time in files' metadata. However, the limitations of these individual approaches means that changes may be missed either due to files being created and deleted between snapshots or due to files being modified without updating the MAC times. As a result, these techniques have been supplemented by integrating a third, but intrusive technique (*Procmon*) into the monitoring process, which runs inside the virtual environment and logs changes made between the snapshots.

³⁶ This can also be achieved by powering off the system, booting to a CD such as *Helix* and imaging the hard drive, although reordering of the steps is necessary so this is the final stage. This may be desirable if examining malicious software since powering off provides a trusted operating system in which to acquire the second disk image.

The developed method also allows results from these three techniques to be automatically combined into a report allowing three-way comparison of the results. Despite the technique using virtual machines, it can also be extended to real systems. This is necessary for performing tests on hardware that cannot be virtualised, e.g. Firewire, or for analysis of malware which may detect that it is running in a virtual environment and not run correctly. However, the technique for real systems has additional limitations since it cannot be ‘paused’ and therefore, the live acquired images of disk and memory are ‘smears’ rather than snapshots. Nevertheless, this developed methodology provides a more comprehensive list of changes made to a system than using a single method.

5.5 RESULTS: RUNNING PROGRAMS

5.5.1 Introduction

Using the developed system monitoring technique the different mechanisms for running software on a system were examined. As described in the methodology section, programs can be launched by double clicking, using the Run command on the Start menu, and also launched from a trusted command prompt. These situations are examined in this section and the results described.

5.5.2 Running Programs: Double Click

In the first instance the ‘hello world’ programs were copied to the desktop and changes were logged when they were launched by double clicking. Both 16 and 32 bit versions of the program were run and this was found to affect the artefacts created. For the 32-bit versions of the ‘hello world’ program, prefetch files³⁷ for each of the ‘hello world’ programs were created in C:\Windows\Prefetch\.

Carvey (2007a)

³⁷ Prefetch files are created by the prefetcher which “tries to speed the boot process and application startup by monitoring the data and code accessed by boot and application startups and using that information at the beginning of a subsequent boot or application startup to read in the code and data” (Russinovich & Solomon, 2005).

describes that “XP can maintain up to 128 Prefetch files”, and during testing, after the creation of 128 prefetch files, no new .pf files were created.

For the 16-bit version of the program, prefetch files for ‘hello world’ were not created. However, there was a prefetch entry for NTVDM.exe which is the “*Windows support image*” that allows 16 bit processes to run under 32 bit *Windows* (Russinovich & Solomon 2005). This prefetch file contained references to the 16 bit programs that were run. In experiments running hello1.exe – hello10.exe only 8 entries were stored, i.e. running hello9.exe & hello10.exe overwrote the entries for hello1.exe & hello2.exe. Therefore, if running a 16 bit process on a live machine the potential exists to overwrite an entry for a previously run 16 bit process in the ntvdm.exe prefetch entry. For all the 16 bit programs, temporary files were also created in C:\Windows\Temp of the form scs#.tmp, where # is a hexadecimal digit. However, for both 16 and 32 bit programs, there were two Registry locations where artefacts were left that referred directly to the executed programs. These were:

```
HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache\  
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\..\
```

The first contained references for all the versions of hello[x].exe that were run with their full path. This is created by the Explorer.exe shell when the executable is run (Carvey 2005). The second location also contains entries for programs that have been run on the system, but is encoded using ROT13, which is trivial to interpret, since each character is simply shifted by 13 places. Each entry also had a binary value associated with it, the latter half being a *Windows* 64 bit hex value date and time describing the last time that the program was run (Farmer, 2007). These have not been found to be overwritten once a certain number is reached. However, further experimentation is necessary to guarantee this.

5.5.3 Running Programs: Start -> Run

Programs were also launched using the Run command from the *Windows* start menu. The artefacts produced were the same as double clicking to execute a program. However there were some additional artefacts found. Entries were created in:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU\
```

This key contains references to the full path of executables run and they are stored in sub keys named a-z, therefore there are 26 possible entries, after which previous ones are overwritten. Each new entry assigned a letter is added to

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU\MRUList
```

In addition, if the files are not typed directly into the Start -> Run option, but are browsed using the dialogue box, there are also entries created in sub-keys a-z (maximum 26 entries) in:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU\exe\
```

5.5.4 Running Programs: Trusted Command Prompt

Applications can also be launched by first opening a command prompt then running programs from there. In this case, first a prefetch entry was created for cmd.exe and the Registry artefacts described in the previous two subsections were created for cmd.exe. When launching programs from the trusted command prompt, in the case of 32 bit programs (only 32 bit processes were tested in this case), prefetch files were also created for each of the programs run from the command prompt. However, the following Registry keys contained references only to cmd.exe, not to the programs run from the command prompt:

```
HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache\
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\

---


```

5.5.5 Running Programs: Summary

Programs can be run in a number of different ways and each causes different disk and Registry changes to take place. For all programs run, prefetch entries were created on disk. In the case of 32 bit processes, one is created for each process run and in the case of 16 bit processes, they are run through NTVDM and therefore only this has a prefetch entry. However, in the latter, files are also produced in the *Windows* temporary folder for each program run. Registry changes also occur and record processes that have been run (e.g. the MUI Cache and UserAssist Registry keys). Running programs using Run from the Start Menu also added entries to RunMRU and, if the dialogue box was used to browse to another location, additional changes were made in the OpenSaveMRU key. When programs were launched from a command prompt, the prefetch files were created for each program run, but Registry entries were only created for cmd.exe, not the programs run from it.

There are a number of implications. All methods for launching software will create prefetch entries for the software run. However, during testing, after the creation of 128 prefetch files, no additional prefetch files were created, suggesting that evidence will not be overwritten in this way. Also, it has been shown that tools should ideally not be run using Start->Run since this will make additional entries in RunMRU which only stores a fixed number (26) of run programs and therefore may overwrite a record of a previously run application.

Further implications of these tests are that if it is necessary to run multiple tools and it is necessary or desirable to minimise entries in the Registry (in MUICache, UserAssist and RunMRU) then it is preferable to launch programs from a trusted command prompt since only one Registry entry will be created.

However, if just launching a single program, double clicking is preferable since it will make fewer prefetch entries. The possible need for an investigator to be able to launch a program by double clicking has implications for live tool design since it means that if parameters need to be passed to a tool, this should be achievable using means other than command line parameters, e.g. using a configuration file or designing programs with interactive shells.

5.6 RESULTS: CONNECTING TO A LIVE SYSTEM

5.6.1 Introduction

Using the developed system monitoring technique the different mechanisms for connecting to a system were examined. As described in the methodology section, inserting a CDROM, connecting a USB device and connecting via Firewire were considered. This section describes these results.

5.6.2 Mounting a CD

A CD was mounted containing the ‘hello world’ test programs used earlier. The changes made when the CD was inserted were monitored. Changes were made to the following Registry keys, but no identifiable information could be extracted:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Tracing\Imapi
HKLM\System\CurrentControlSet\Enum\Root\LEGACY_IMAPISERVICE\
HKLM\System\CurrentControlSet\Enum\IDE\CdRomHL-DT-ST_DVD+-RW_GSA-
H31L_____1.05____\30313030303030303030303030303030303030303130\Device
Parameters
```

Also, in the HKCU\Software\Microsoft\Windows\ShellNoRoam\MUICache Registry key, 8 entries were created, all of which began with @shell32.dll, and represent the tasks added to the user interface for an inserted CD. However, no maximum amount of entries stored in this key has yet been found.

@shell32.dll,-8504	REG_SZ	Auto&Play
@shell32.dll,-12589	REG_SZ	Files Currently on the CD
@shell32.dll,-12590	REG_SZ	Files Ready to Be Written to the CD
@shell32.dll,-31353	REG_SZ	CD Writing Tasks
@shell32.dll,-31355	REG_SZ	Write these files to CD
@shell32.dll,-31234	REG_SZ	These tasks apply to the files and folders you select

@shell32.dll,-31273 REG_SZ These links open other folders and take you quickly to useful places.

@shell32.dll,-31275 REG_SZ This section displays the size, file type, and other information about a selected item.

Locations containing data that specifically referenced the inserted CD are shown below.

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\TotalBytes
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\FreeBytes
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\Media
Type
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\UDF
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\Disc Label
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CD Burning\Current Media\Set
```

This information may be important if a CD needs to be ejected in order to load an investigator's toolkit on to the machine using another CD, since on removal of the CD the keys are deleted. Deleted Registry keys have not been investigated as part of this research but new values are written on inserting a different CD. It is possible that these may overwrite the details of the previously inserted CD. Note that these changes documented in this sections are for a CD inserted into a drive capable of writing CDs; there are far fewer made for standard CD drives.

5.6.3 Attaching a USB Device

Changes caused by connecting a USB device have been previously documented in Carvey and Altheide (2005) which described that on a *Windows XP* system, the `setupapi.log` in the system root is changed, as are the following Registry keys:

```
HKLM\System\CurrentControlSet\Enum\USB
HKLM\System\CurrentControlSet\Enum\USBStor
HKLM\System\MountedDevices
```

Tests were performed to validate these changes. If the USB device is the first to be inserted then the `usbstor.sys` driver is installed (25.9KB) which will overwrite data in unallocated space. During testing it was found that references were added to `setupapi.log`, including references to the USB device's Vendor ID, Product ID and serial number. However, these details were appended to `setupapi.log`, so an investigator adding a USB device will not overwrite any of the existing log. However, the changes added to the log totalled 1401 bytes of data, which may overwrite a small amount of data in slack and unallocated space. Also, a prefetch entry is created for `RUNDLL32.EXE` which executes DLLs and places them into memory, although an inspection of the prefetch entry with *bintext* (Foundstone, 2000) could not find any USB specific references. In addition to the Registry changes mentioned above, references to the attached USB stick were also found in:

```
HKLM\System\CurrentControlSet\Control\DeviceClasses\  
HKLM\System\CurrentControlSet\Enum\STORAGE\RemovableMedia\  
HKLM\System\MountedDevices
```

5.6.4 Connecting a Firewire Device

To examine the changes made by connecting a Firewire device the technique had to be modified for use on a real system instead of a virtual machine, as discussed in Section 5.4.5. A *Linux* based laptop was configured using the *pythonraw1394* scripts from Bolieau (2006), connected to the system and the system's RAM was imaged using the *1394memimage* script, also from Bolieau (2006).

On connecting via the Firewire port, since it was the first use of the port, Serial Bus Protocol 2 (SBP2) drivers were installed for the port:

```
WINDOWS\system32\dllcache\sbp2port.sys (43136 bytes)  
WINDOWS\SYSTEM32\DRIVERS\SBP2PORT.SYS (43136 bytes)
```

There were also entries added to `sysevent.evt` log reporting an error of the SBP2 driver:

112	19/11/2008 16:57:08	19/11/2008 16:57:08	4	Error 40	sbp2port	TEST-PC
113	19/11/2008 16:57:15	19/11/2008 16:57:15	4	Error 40	sbp2port	TEST-PC

There were also Registry changes related to the driver installation:

```
HKLM\System\CurrentControlSet\Control\Class\{4D36E967-E325-11CE-BFC1-08002BE10318}\0006
HKLM\System\CurrentControlSet\Control\Class\{D48179BE-EC20-11D1-B6B8-00C04FA372A7}\0000
HKLM\System\CurrentControlSet\Control\CriticalDeviceDatabase\gendisk
HKLM\System\CurrentControlSet\Enum\PCI\VEN_1033&DEV_00F2&SUBSYS_00CE1033&REV_01\
HKLM\System\CurrentControlSet\Services\EventLog\System\sbp2port
HKLM\System\CurrentControlSet\Services\sbp2port
```

Since the *Linux* machine simulates the connection of an iPod, other changes made when connecting the Firewire cable were similar to inserting a USB stick. In the *Windows* folder, the file `setupapi.log` had 3232 bytes appended to it, describing the installation of drivers for an iPod, and Registry entries for the installed 'iPod' could be found in:

```
HKLM\System\CurrentControlSet\Enum\1394\Apple_Computer__Inc.&iPod\ 80E000024C0000
HKLM\System\CurrentControlSet\Enum\SBP2\Apple_Computer__Inc.&iPod&LUN0\00004c0200000e08
```

5.6.6 Connecting to a Live System: Summary

There are a number of ways of connecting to a live system. Programs are often run on a system from an investigator's CD ROM and this is advised in Adelstein (2006) since it is a read only medium. During the test scenario for CDs, a number of changes were made to the Registry and if the drive is capable of writing CDs, among these changes is information about the currently inserted CD. The consequences of this are that if an investigator ejects a CD already in the machine then digital evidence of that CD being

in the drive may be lost. While this is only likely to be relevant in a small minority of cases it is still a point worth noting.

Another option for getting programs on to a system, and data from it, is the USB port. Connecting a USB device makes changes to files on disk (setupapi.log) and to the Registry. The information added to setupapi.log and to the Registry is always appended, as yet with no identified limit. Therefore, the risk of overwriting evidence is limited to data in slack and unallocated space, which is overwritten as data is added. Since data is appended, USB devices (preferably read-only for tool delivery) could be considered in situations where the contents of the CD drive may be of interest.

Connection via Firewire produced similar changes to connecting a USB device, although since Firewire devices are not as popular as USB it may be more likely that the Firewire driver needs to be installed; which will overwrite more data in unallocated space since SBP2PORT.SYS is created twice on the system and changes are also made to sysevent.evt. There were also several Registry keys created and modified. Specific iPod related changes could be found in setupapi.log and HKLM\System\CurrentControlSet\Enum.

There are also other connections that have not been considered in this research, for example Ethernet and e-SATA. Also, the specifics of connecting a USB hard drive rather than a thumb drive have not been examined. However, the developed methodology can be used in future to examine these interfaces on virtual and real systems.

5.7 RESULTS: RUNNING LIVE INVESTIGATION TOOLS

5.7.1 Introduction

Using the developed system monitoring technique, a number of live investigation tools were examined. Acquisition tools for both disk and memory were examined, as were some analysis tools, e.g. *pslist* and *psinfo*.

5.7.2 *dd* for acquiring disk

dd (from the Forensic Acquisition Utilities (FAU)) was executed from an already running command prompt and used to image a section of the hard disk to another drive. Running *dd* created only a single prefetch entry and single Registry change:

```
WINDOWS\Prefetch\DD.EXE-1D9BD197.pf
HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed
```

5.7.3 FTK Imager for acquiring disk

FTK Imager was executed and used to image the system drive of the virtual machine to a second drive. This created the following prefetch file:

```
WINDOWS\Prefetch\FTKIMAGER.EXE-00778F2F.pf
```

There were also several Registry changes made, including HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed. Also, The Registry key HKCU\Software\Smart Projects\AccessData Corp.\Version was created which describes the version of *FTK Imager* that was run. There were also Access Data specific entries created in HKCU\Software\AccessData. There were also a number of additional Registry entries created in this key that were deleted when the imaging was complete. These were:

```
HKCU\Software\AccessData\FTK Imager\ProfUIS240\Profiles\FTK Imager\ControlBar
HKCU\Software\AccessData\FTK Imager\ProfUIS240\Profiles\FTK Imager\ControlBar\data_size
HKCU\Software\AccessData\FTK Imager\ProfUIS240\Profiles\FTK Imager\ControlBar\data_integrity
HKCU\Software\AccessData\FTK Imager\ProfUIS240\Profiles\FTK
Imager\ControlBar\block_0x00000000\data_0x00000000
...
HKCU\Software\AccessData\FTK Imager\ProfUIS240\Profiles\FTK
Imager\ControlBar\block_0x00000000\data_0x00000058
```

Therefore, running *FTK Imager* made significantly more changes than using *dd*.

5.7.4 *dd* for acquiring memory

The modified version of *dd* obtained from the *Helix Live CD* was run from the command line and was used to image the memory of the test system. This produced the same results as the previous use of *dd* for disk imaging: a prefetch entry for *dd* and a change to the ...*Cryptography\RNG\Seed* Registry key.

5.7.5 Fast Dump for acquiring memory

FastDump (FD) (HBGary, 2008a) was run from the command line and used to acquire a memory image of the test system, and file changes consisted only of a prefetch entry: *WINDOWS\Prefetch\FD.EXE-062D3D04.pf*. There were no Registry changes detected.

5.7.6 *PSList*

PSList from SysInternals was run from the command line and listed processes running on the system. In addition to the prefetch entry caused by launching the tool (*WINDOWS\Prefetch\PSLIST.EXE-08928D72.pf*), one Registry entry was modified and another created. The created entry contained a flag to record that the End User License Agreement (EULA) for the software had been read and accepted on the first run of the program.

<i>HKLM\SOFTWARE\Microsoft\Cryptography\RNG\Seed</i>	(modified)
<i>HKCU\Software\Sysinternals\PsList</i>	(created)

5.7.6 *PSInfo*

After running *psinfo* from the command prompt a number of prefetch files were created, as was a *psinfo* specific Registry key regarding acceptance of the EULA:

WMIAPSRV.EXE-1E2270A5.pf
WMIPRVSE.EXE-28F301A9.pf
RUNDLL32.EXE-321A7019.pf
RUNDLL32.EXE.451FC2C0.pf

HKCU\Software\Sysinternals\PsWithInfo\EulaAccepted (created)

There were also 59 other Registry key changes, 20 of which were specifically attributed to the *psinfo* process by *Procmon*.

5.7.7 WinAudit from Helix Live CD

WinAudit produces a HTML report detailing various information about the system, including the system specification, the current date and time, up-time, size of RAM, size of disk, installed software, open ports, running programs and services. Executing *WinAudit* created one additional prefetch entry to the one for the *WinAudit* program:

WINDOWS\Prefetch\WMIPRVSE.EXE-28F301A9.pf

During testing there were also 132 Registry changes made, which is particularly significant when compared to the number of changes caused by disk and memory imaging. Particularly when much of the information obtained using *WinAudit* could be obtained from disk and memory images.

5.7.7 Running Live Investigation Tools: Summary

The changes reported in this sub-section exclude the Registry changes that are due to 'running a program' which were discussed in a previous section. The disk acquisitions performed with *dd* and *FTK Imager* made different numbers of changes, with *dd* making fewer changes to the system than *FTK Imager*, which is unsurprising since *FTK Imager* uses a graphical interface and offers more features.

The memory of the test systems was acquired using *dd* from the *Helix Live CD* and also *FastDump*. Both made very few changes. *dd* made one file and one Registry change and *FastDump* just a single file change, which was a prefetch entry for itself. Both these tools had to be launched from the command line, which, as shown in a previous section, is not ideal due to the extra prefetch entry created.

The results of memory acquisitions are particularly interesting when compared with the results from running live analysis tools. *Pslist* made fewer changes than expected: a single file change (prefetch) and two Registry changes, one related to EULA acceptance. *Psinfo* was also run, and created a number of prefetch and Registry entries, one Registry key specifically being related to EULA acceptance. Also *WinAudit* was run from the *Helix Live CD* which created two prefetch entries and a considerable number of Registry modifications.

While these live analysis tools were previously the only way to obtain information from a live system, the additional changes made by these live analysis tools could now be considered to be unnecessary since much of the information obtained by running these tools can be recovered from a memory dump of a live system. This is particularly true if the changes caused by 'launching programs' are considered, since live investigation tools usually perform a single task and many of them are run sequentially to obtain a broad amount of information from a system. This is a problem since each piece of software run will produce a prefetch entry, and at least one addition to the Registry. Considering that in many cases these live investigation tools can be replaced by a memory acquisition and an offline (but still at the scene) analysis of that image, changes to the suspect system can be minimised by taking the memory acquisition approach and minimising the reduction in the completeness of preserved digital evidence. Some information obtained using live tools cannot be recovered from memory images and requires information from disk. However, if *dd* was modified to perform selective acquisition and could be configured to obtain files from which this additional information could be recovered e.g. Registry hives, then these files could also be analysed at the scene, but offline. This would allow reduction in completeness to be minimised further. This selective acquisition approach remains future work.

5.8 EVALUATION

5.8.1 Methodology Evaluation

This chapter has investigated the extent to which the completeness requirement explained in Chapter 3 can be satisfied for live investigations. The requirement states that *it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved*. This means that the changes caused to a system due to live tools and techniques should be identifiable so that the evidence lost due to data being overwritten can be assessed post-live investigation. It is also important to know, before the investigation takes place, what changes are normally made by live tools, since this can assist an investigator in determining the most appropriate live technique to use in order to preserve the maximum amount of relevant digital evidence. The approach to achieve both of these was to develop a methodology to allow the monitoring of live tools in a test environment and to record changes made. This is now possible and a number of interesting results have been obtained.

However, there are limitations to this methodology. First, only a small selection of live techniques have been examined and there are many other live tools in use and many other methods of connecting to a live system. However, as discussed earlier in the chapter, the focus of this research was to develop a methodology to allow changes to be identified, rather than to provide a comprehensive testing of all available live techniques. This is therefore not considered to be a significant limitation. In addition, a broad range of different test situations have been examined, which demonstrates the versatility of the developed technique.

Another limitation is that the tests were all performed on a single operating system: *Windows XP Service Pack 2*; and while this is a popular operating system choice, the changes made by tools running on different operating systems may change and also need to be examined. However, again, this can be achieved using the developed methodology and remains future work. Also, the test systems are basic installs of the test operating system, i.e. free of any other installed software, e.g.

antivirus. Therefore, they are unlikely to represent real life systems that will be encountered in the course of live digital investigations. However, the research carried out can be considered to be the first stages of testing, whereas in future, test systems can be constructed that better represent those that may be encountered during live investigations. It is also possible, post-live investigation to build a test system in a similar configuration to that examined, and use it to identify changes that were likely to have been made during the real investigation.

Recording the footprint of live tools on test systems allows predictions of what changes will occur on real systems. However, the data that was changed on a system from a real investigation can be better inferred using information available on the system after the live investigation takes place. As a result, after a live investigation, it is still necessary to examine the system to determine what evidence has been lost. However, the results obtained from monitoring test systems also assist with this process, since an improved understanding is gained of the changes that are made to systems. This allows an investigator to draw conclusions about the cause of a particular artefact being found on the examined system (whether they were due to normal background activity or the live tools). Also, one of the implemented techniques (MAC times based approach) can be applied to acquired data from a live investigation, and changes made after the recorded time of the beginning of the investigation can be extracted. However, this MAC times based approach is insufficient on its own, since not all changes to the system will update MAC times. A complete methodology for identifying changes made post-investigation remains future work.

Despite these limitations, for a live investigation, this test system based approach, and the prediction of likely changes is extremely useful. The approach provides information that can be used by an investigator to understand likely changes that will be caused by the tools used. This allows them to make a decision about the best course of action in order to preserve the maximum amount of potentially relevant digital evidence. The examination of test systems also assists after the investigation has taken place, since it increases understanding of artefacts left on the system and

allows some of the detected changes to be attributed to normal background processes and the actual changes caused by live tools to be identified.

5.8.2 Monitoring Technique Evaluation

The system monitoring technique described in this chapter was developed to record changes made to test systems and to overcome the limitations of existing monitoring techniques. The three existing techniques for system monitoring described were: live logging tools, snapshot based approaches and MAC times based analysis. Live logging tools are intrusive techniques that intercept system events and record them to logs, and experiments have shown that these techniques can fail to record some events, and this research has not found a conclusive explanation for this. The existing snapshot based approach (*InCtrl5*) is also an intrusive monitoring technique that has the disadvantage of not recording changes that are made and undone in between snapshots, e.g. the creation and deletion of temporary files. The MAC times based approach is the only unintrusive technique, but this will also miss changes where the file metadata has not been updated.

The developed approach significantly modified the snapshot approach so that it was an unintrusive technique based on the use of virtual machines. It also combined it with the two other approaches into a much more comprehensive monitoring methodology, where a broader amount of changes are captured. However, the overall technique has remained intrusive due to the reliance on live logging techniques (*Procmon*). However, since only one intrusive technique is used, filtering out changes caused by *Procmon* is straightforward, as it can be monitored running on its own, the changes recorded and filtered out from future results. Also, in future, the combined technique could be used to monitor a virtual machine from a completely external perspective by removing the live logging tools, if the limitations of the snapshot approach could be overcome. *Procmon* is currently necessary due to the snapshot approach missing changes that are made and undone between snapshots. The impact of this limitation could be minimised, if in addition to the live file set, unallocated space in the two snapshots was also examined for changes, which would capture files

created and deleted between snapshots. It may also be possible to run the live logging tools on the host system and to translate writes to the virtual machine's disk file into changes made in the virtual file system. Alternatively, open source virtualisation software could be modified to record changes to the virtualised system as they occur. However, these improvements remain future work.

Alternatively, the *Procmon* monitoring could still be used, but separated out. This would mean conducting two experiments for each technique under test. Using the developed scripts for formatting tool output and combining them into a single report, the *Procmon* log from one test could be combined with the virtual system monitoring results from another. If the same baseline virtual machine was used, the results should not be significantly different. Alternatively, by combining results from systems in slightly different configurations, the robustness of the results could be increased. This separation of the live logging tools would mean that the snapshot and MAC address experiment was completely unintrusive, which would also enable the analysis of changes to the memory of the virtual machine to take place, since it would not be affected by monitoring tools. In this research, only changes to disk have been considered, without examining changes to the memory of the system. This is because it currently cannot be preformed correctly without additional experiments, due to the use of the intrusive *Procmon* tool. This is not a problem in this research, as changes to memory are not discussed since current techniques do not preserve memory at all, and therefore the intricacies of losing a small part of memory due to the use of a particular tool are not considered. However, this will need to be examined in future and the proposed modifications to the technique discussed in this section will allow this.

The developed approach also offers the advantage of combining the results from the three approaches into a single report. The purpose of this was to validate the results of each of the monitoring methods. However, this has not been fully achieved since the different methods often do not agree because each method misses certain changes. So, while the complete validation of the individual methods has not been possible, combining them produces a more comprehensive list of changes and highlights the need to correlate multiple monitoring techniques.

There are also some practical limitations of the developed monitoring technique to consider. One consequence of the thoroughness of the monitoring is that the space requirements are much greater: for each experiment, two additional copies of the virtual machine's hard disk and memory are made, in addition to the log from the live logging software. There is also a time issue, where creating duplicates of virtual hard drives for the 'before' and 'after' snapshots is more time consuming than the live logging approach alone. However, due to the low cost of hard disk storage, the amount of data generated has not been a significant problem, neither has the time needed to create duplicates of the virtual hard disks since the timescale is in minutes rather than hours.

There are also specific difficulties with the new snapshot based approach. When the virtual machine is paused immediately after performing some action or running a piece of software, the changes are sometimes not recorded due to caching of disk writes. This is not a problem in the overall approach since the live logging method does record them, and this has been addressed in this research by leaving a short time after the event to allow changes to be written. However, a more effective solution to this problem is still being sought.

Finally, it is still a challenge to efficiently filter out background changes. While changes that occurred on an idle system were documented and excluded from results manually, it is not possible to say with certainty that files that normally change in the background do not also change as a result of actions performed on the system. This is a weakness of the methodology that can currently only be resolved through manual inspection of all background changes listed by the techniques, which is repetitive, time consuming and error prone. One way in which this process could be made easier is by replacing the HTML based reporting of the current system with a full interactive user interface, designed specifically for examining reported files for changes. However, this also remains future work.

Despite these limitations, the virtual machine snapshot approach provides a realistic alternative to live logging tools, particularly if a method of identifying changes made and undone between snapshots can be devised. Also, the automation of

the report generation, while improvements are needed, allows the changes made to a number of test systems to be easily examined and allows a three way comparison of results that would not be feasible if performed manually.

5.9 CONCLUSIONS

5.9.1 Summary

The completeness requirement means that *it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence that is relevant to the investigation should be preserved*. Evidence lost can be easily assessed in a traditional digital investigation since it is pre-determined with procedure ('pull the plug') what will be preserved (the hard disk) and lost (memory). The previous chapter showed that in certain circumstances, e.g. Full Disk Encryption, this approach will not preserve the maximum amount of relevant digital evidence and therefore a live investigation should be performed.

Therefore, it is necessary to be able to assess which digital evidence is preserved and which is lost when a live investigation is performed. While in the Full Disk Encryption example a live investigation is capable of preserving more than a traditional investigation, there are subtleties within the live investigation: live tools are intrusive and therefore overwrite data and cause evidence to be lost, but some tools or techniques may make fewer changes and therefore preserve more digital evidence than others. Also, different tools make different changes and data that is not relevant in one investigation may be relevant in another. Therefore, a different choice of tool or technique in different investigations may allow the maximum amount of relevant digital evidence to be preserved for a particular investigation. It is therefore necessary to identify changes that are caused by performing different actions on systems, so that the most appropriate action can be chosen for the current circumstances.

This chapter has developed a methodology for monitoring live tools and techniques in a test environment to establish the changes that they make. This can be

achieved using existing methods: live logging tools, intrusive snapshots, or sorting files by their MAC times. However, each of these approaches has limitations that mean they may miss certain changes that are made to a system. Therefore, this could result in live investigations being performed that overwrite relevant data and consequently fail to preserve relevant digital evidence. The developed methodology combines the approaches into a single technique that uses each of the methods' strengths to overcome the weaknesses of others, providing a more comprehensive set of results.

This developed methodology was used to examine a number of live tools and techniques in test environments and to record the changes made. The tests carried out can be grouped into 'launching programs', 'connecting to a live system' and 'running live investigation tools'. A number of interesting results were found which are summarised in the following sub-sections.

5.9.2 Launching Programs

Changes caused by running programs in a variety of ways were recorded: launching by double clicking, using Run from the Start Menu, and also by launching from a trusted command prompt. The least intrusive way to launch programs depends on how many programs will be run. If more than one live tool will be run then a command prompt is better as it makes fewer changes to the Registry. However, if a single tool is to be run, then launching by double clicking is preferable since the Registry changes are the same but with one fewer prefetch file created. This is also an important point for developers of live tools since tools are often designed so that parameters are passed to live tools using the command line. If this is not always desirable then alternatives such as configuration files or interactive shells should be considered.

5.9.3 Connecting to a Live System

Options for connecting to a live machine were also considered. Information about the currently mounted CD was found in the Registry, but was deleted when the CD was removed e.g. for an investigator to load a toolkit. However, without investigating

deleted Registry keys it is not possible to know if information about previously inserted CDs would have been recoverable if an investigator's CD had not been inserted. Therefore, in a small number of cases where the currently inserted CD may be relevant, it is important not to eject the current CD to load live tools on to a system.

Attaching a USB mass storage device and Firewire device was also considered and both made a number of changes to the disk and Registry of the system. However, from the tests performed, the amount of potential digital evidence that could be lost due to connecting these devices is minimal since all data written is appended and does not replace existing values. Therefore, the loss of evidence is limited to data in unallocated space, slack space or deleted keys in the Registry.

5.9.4 Running Live Tools

A number of live acquisition tools were also investigated. Two disk imaging tools were tested and *dd* was found to cause fewer changes than *FTK Imager*. Also, memory was acquired using *dd* (from *Helix Live CD*) and *FastDump*, both of which produced prefetch files for themselves and *dd* caused an additional Registry change. More significant is the difference between the amount of change caused by these memory acquisition tools compared to live investigation tools, with the latter producing many more changes. This is important since much of the information obtained by running live analysis tools can now be recovered from a memory dump using tools such as *Volatility* (described in Chapter 2). Therefore, by using memory acquisition tools, which have a smaller footprint, and then obtaining information from the acquired image offline (but still at the scene), fewer changes to the suspect system can be made, which minimises the reduction in completeness of the preserved digital evidence.

5.9.5 Final Summary

It has been shown that it is possible to assess the changes made by live tools in a test environment and, with the assumption that similar changes will be made on systems during actual investigations, it is therefore possible to predict the changes that will be

made to systems during actual live digital investigations. Further experimentation using test systems that better represent 'real life' machines can be performed to make this assumption more valid, and will allow better understanding of the changes caused by live tools and techniques. This will allow the evidence preserved and lost as a result of live investigations to be assessed and will also assist an investigator to decide the best course of action at the scene that will preserve the maximum amount of relevant digital evidence. An example of this would be using a live memory acquisition followed by an offline analysis of that memory image, rather than using live investigation tools which obtain the same information, but make far more changes.

CHAPTER 6: ACCURACY

6.1 INTRODUCTION

In Chapter 3 it was explained that in a digital investigation *it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation*. This chapter considers the nature of error associated with digital evidence and how it can be assessed. It examines how repeatability can be used as a means of assessing the accuracy of techniques used to recover and process digital evidence, and shows that this is not usually possible for live digital investigations. However, this chapter also shows that in the context of live digital investigations involving encrypted evidence, other data can be recovered from a live machine that allows offline decryption of encrypted data. Since offline decryption is possible, this means that the acquisition of encrypted digital evidence can also be performed in a repeatable manner, allowing the accuracy of the processes used to be assessed.

6.2 BACKGROUND

6.2.1 Introduction

A limitation of the explanation provided for assessing the accuracy of digital evidence described above and in Chapter 3, is that error in the context of digital investigation techniques (acquisition, analysis and presentation) has not been defined. The technical definition of error is “a measure of the estimated difference between the observed or calculated value of a quantity and its true value” (Oxford, 2008). The difficulty in using this definition of error for digital evidence is that a piece of digital evidence is generally not a value and the error cannot be expressed as $x \pm y$. Also, since digital evidence is always an abstraction of something physical (see Chapter 2), and it is not possible to view the actual data directly, this would make it impossible to assess the error in any digital investigation, since the ‘true’ value of the data cannot be known.

This section explores how error associated with digital evidence can be assessed in a meaningful way.

6.2.2 Error in Specific Aspects of Digital Investigations

There are several papers that discuss error related to digital evidence and each focuses on error associated with a particular aspect of a digital investigation. Carrier (2003) considers the introduction of error due to analysis tools (which are used to “translate data through one or more layers of abstraction until it can be understood”). Two types of analysis tool error are described. One is *tool implementation error* which is due to programming and tool design errors. The other is *abstraction error*, which is introduced “because of simplification used to generate the layer of abstraction” and when “abstraction is not part of the original design”. An example of this in a traditional digital investigation would be the skin colour based detection feature of *X-Ways Forensics* (X-Ways, 2009), where files below a certain threshold are not displayed. This separation of files based on skin colour removes files from the view of the investigator in a way that is not part of the original operating system design. Since this approach for filtering relevant files is imperfect, this could filter out files that are important to the investigation, and the technique could therefore introduce abstraction error. However, abstraction layers do not necessarily introduce error and are described in Carrier (2003) as “lossless” (zero error)³⁸ or “lossy” (error greater than zero).

Carrier (2003) also mentions other types of error that are not due to analysis tools, including “errors introduced from the attacker covering his tracks, from faulty imaging tools, or from an investigator misinterpreting the results of a tool”. However, they are not discussed in detail.

Casey (2002a) also discusses error in digital investigations, focusing on error in digital evidence obtained from networks. Similarly to Oxford (2008), Casey (2002a) describes error as “the difference between the true value and the measured/recorded value”.

³⁸ ASCII is an example of a lossless abstraction layer

Casey (2002a) discusses ‘temporal uncertainty’, (where ‘uncertainty’ is described as the “probable upper bound of the error”) which can be caused by clock offset (either small drift or deliberate tampering) and by also limits in resolution, for example a record of “connections to/from a suspect’s computer that are totalled every ten minutes”. This means that from this data “it is not possible to distinguish between a Web site that the suspect accessed for ten minutes and a Web site that was only viewed for a few seconds.”

Casey (2002a) also discusses ‘uncertainty in origin’, which could be confused with the authenticity requirement used in this research. However, due to the way in which ‘uncertainty in origin’ is discussed in Casey (2002a), in this research, this term relates to the accuracy requirement. In the examples given in Casey (2002a), the term ‘uncertainty in origin’, rather than being concerned with identifying the physical evidence from which the digital object was obtained, relates to the events that caused the digital object to have its current value. For example, Casey (2002a) describes that the ‘from’ header in an e-mail could be falsified and has a high degree of uncertainty, and also that there is difficulty in using an IP address to determine an individual machine that is behind a Network Address Translation (NAT) device. In these examples, the digital evidence artefacts are the e-mail header and the IP address respectively. The origin of both, using definitions in this research, is considered to be the physical evidence from which it was obtained, likely a server. Showing that the digital objects were obtained from a particular server would therefore be part of the authenticity requirement. However, “uncertainty in origin” as referred to in Casey (2002a) i.e. determining the uncertainty in what actual events caused digital objects to have their current values, relates to accuracy. This is because, if there are multiple events that could cause the same state of digital data, there is an actual, true event that caused it, and one or more other events that did not. Considering why these digital objects have the states that they do, the ‘from’ entry in an e-mail header found on a server could have its value for a number of reasons, including that it came from that sender; or that it came from a different address and the header has been falsified. Equally, a reference found to the IP address of a NAT device could actually be caused

by connections from one of several machines behind that device. Discussing error in terms of a set of alternative reasons for a digital object having a particular value leads to a general definition of error for digital investigations.

6.2.3 Defining Error in Digital Investigations

In Chapter 2, it was explained how interactions with the real world and other digital devices can cause digital events to occur, which in turn create digital evidence artefacts, e.g. a person typing a URL into *Internet Explorer* (physical event) will trigger software code to execute, including an Application Programming Interface (API) call that creates a Registry entry (digital event) in the TypedURLs Registry key that contains the text typed into the address bar (digital evidence artefact). Chapter 2 also described that the aim of a digital investigation is to “make valid inferences about a computer’s history” (Carrier, 2006a), where a computer’s history is defined as “a sequence of its previous states and events” (Carrier, 2006a). Since digital data on a system is usually as a result of interaction with another digital device or the real world, it is often necessary in an investigation to infer about these past external interactions of the computer being examined. Therefore, a computer’s history is defined from Carrier (2006a) as “a sequence of its previous states and events”, and an event can be any “occurrence that changes the state of the system”. ‘Events’ can therefore include digital events on the system, e.g. API calls or automatic pop-ups, interactions with other digital devices, e.g. connection to a USB device or the receipt of an e-mail, and also real world events, e.g. a user launching a program, clicking an HTML link, or typing some text.

Since the history of a computer is not fully recorded (Carrier and Spafford, 2006), it is necessary during a digital investigation to infer about previous events in a computer’s history by formulating and testing hypotheses using the currently available digital evidence. Therefore, modifying the Oxford (2008) definition of error and using the main aim of a digital investigation and the idea that a computer has a history, the error associated with digital evidence can be defined as *the difference between the inferred history and the true history of the examined digital evidence*.

This error cannot be expressed as a definite value, e.g. $x \pm y$, but can be expressed as uncertainty (possible error) in the inferred events, i.e. *alternative possible hypothesised events that explain the current state of the examined digital evidence*. For example, cached images from a prohibited web site (digital evidence) could be found in a user's cache because the site was intentionally visited, or because it was opened automatically in the background by another site (alternative possible events).

While this explanation of the definition of error has discussed inferring incorrect events in a computer's history, these are errors only in high level hypotheses. In order to form these high level hypotheses about sequences of events, it is necessary to first form and test lower level hypotheses to "abstract data into files and complex data structures" (Carrier, 2006a), for example, the hypothesis that a particular piece of data embedded in a file represents the time it was modified, or that the typed URLs in the Registry need to be interpreted as Unicode characters. However, these are also included in the proposed definition of error since it is the difference between the inferred history and true history of the *examined* digital evidence. The history of the examined digital evidence includes events such as the acquisition of the digital data from the original physical evidence and the interpretation of this raw data to produce the digital objects in a form that can be analysed. Therefore, incorrect interpretations/abstractions of raw data can also be considered to be alternative hypotheses to that which assumes data structures are interpreted correctly and deterministically by the tools used to translate raw data into a form that can be understood.

6.2.4 Assessing Error in Analysis in Digital Investigations

Determining and explaining what caused the examined digital evidence to have its current state is part of the analysis stage of a digital investigation, e.g. determining whether pictures were intentionally downloaded or were part of an automatic pop up. Error in analysis is present if incorrect events are induced from the available digital evidence. Casey (2002a) proposes a means to specifically address the problem of

quantifying the certainty in digital evidence using a ‘Certainty Scale’ from C0 to C6. This is shown in Table 20.

Certainty level	Description/Indicators	Certainty
C0	Evidence contradicts known facts.	Erroneous/incorrect
C1	Evidence is highly questionable.	Highly uncertain
C2	Only one source of evidence that is not protected against tampering.	Somewhat uncertain
C3	The source(s) of evidence are more difficult to tamper with but there is not enough evidence to support a firm conclusion or there are unexplained inconsistencies in the available evidence.	Possible
C4	Evidence is protected against tampering or multiple, independent sources of evidence agree but evidence is not protected against tampering.	Probable
C5	Agreement of evidence from multiple, independent sources that are protected against tampering. However, small uncertainties exist (e.g. temporal error, data loss).	Almost certain
C6	The evidence is tamperproof and unquestionable.	Certain

Table 20: Certainty scale described in Casey (2002a).

A scale such as this is useful in estimating uncertainty in analysis and coming to conclusions; for example, if the Internet history on a machine contains references to prohibited web sites and a server log (outside the control of the suspect) records access to that site from the suspect’s machine, the hypothesis of the machine accessing that site would be given a certainty scale of C4-C5. However, the subtleties of this example, i.e. to determine whether the suspect intentionally accessed the site requires further digital objects to be examined e.g. typed URLs, bookmarks etc. Therefore, the Certainty Scale can be applied to a specific hypothesis, e.g. the machine accessed the prohibited site, and can also then be re-applied to new hypotheses as they arise, e.g. the suspect deliberately accessed the site. The scale is therefore best used to assess the certainty of specific conclusions.

However, this scale alone does not address the problem of assessing the accuracy of lower level hypotheses, since in order to use multiple independent sources

for low level analyses, it is necessary to be able to run different tools on the same data, in which case, what is necessary is repeatability. Repeatability also allows multiple, independent sources in terms of a different examiner running the same analysis tool to check the results, i.e. that an examiner used the tool correctly.

Uncertainty in high and low level hypothesis formation and testing, can therefore be addressed in traditional digital investigations since the data is preserved at a low level of abstraction. The raw data can be examined using multiple tools or manually to translate it into a form that can be understood. If multiple tools agree or the manual reconstruction of the information can be shown, this allows the alternative hypothesis of incorrect interpretation/abstraction to be ruled out from the history of the examined digital objects. Also, the analysis can be performed by multiple examiners who can repeat the low level analysis techniques to determine if they were performed correctly. They can also repeat the high level analysis, which allows investigators to form and test their own individual hypotheses about sequences of events that caused digital objects to have particular values. The data that supports or refutes these hypotheses can be evaluated against a Certainty Scale, either introspectively or using one such as that in Table 6.1 and a decision can be made about which hypothesis is most likely. The Certainty Scale also assists with making a decision about what error is acceptable in the current investigation. Therefore, assessing uncertainty in the analysis stage of a traditional digital investigation relies on a combination of repeatability and a scale against which to judge the certainty of conclusions drawn.

Both repeatability and Certainty Scales can also be used to assess accuracy in the analysis stage of a live digital investigation, but only if the acquisition and analysis stages are separated. This is because once an image is acquired from a live machine, it has the same properties as digital evidence acquired in a traditional digital investigation, i.e. the live image can be exactly copied in full and any analysis performed can be repeated on duplicate copies by independent examiners who can interpret the raw data and form and test different hypotheses. The difficulty in

determining the accuracy during live investigations therefore lies in the acquisition stage.

6.2.5 Assessing Error in Acquisition in Digital Investigations

A simple acquisition is an exact bit stream copy of the source data, i.e. the output should be the same as the input, and the input should equal the actual digital data. If any of these are not equal then this is an abnormal event in the history of the acquired and therefore examined digital object. In a traditional digital investigation it is not usually necessary to consider alternative histories of the acquisition since it is performed using tested, trusted software. Also, it can be verified at any time that the output is equal to the actual digital data, since the original evidence is still available. Furthermore, it is performed in a trusted environment, so there is no reason for the input provided to the acquisition process to differ from the actual data. It is also possible to repeat the acquisition many times and verify newly acquired data against existing images or the original evidence.

However, this is not the case during a live acquisition since it is performed on a machine that is running and the acquired evidence “represents a snapshot of a dynamic system that cannot be reproduced at a later date” (Adelstein, 2006). The acquired image can therefore be verified only against itself rather than against the original media (Casey and Stanley, 2004). Not only can live data change between consecutive acquisitions, but live acquisitions are particularly problematic in the context of this research i.e. when encryption is involved. This is because, as shown in Chapter 4, when acquiring encrypted evidence, ‘pulling the plug’ on a machine running file system, virtual disk or full disk encryption, means that the decrypted form of the evidence is no longer accessible and cannot be re-acquired to validate the live acquired data.

In a live acquisition, not only could the output of acquisition tools not equal the input, which introduces uncertainty, but also the operating system itself is untrusted (Carrier, 2006b, Kenneally and Brown, 2005). This means that the input data to the acquisition process may not equal the actual data. The normal behaviour of

an operating system is to provide the acquisition process with input that is equal to the actual data. However, acquired data can also have an alternative history where the data provided by the operating system as input to the acquisition process is manipulated in some way to add data, or more commonly to hide data. This is a common technique used by rootkits which can also be specifically designed for anti-forensics (Bilby, 2006) and can modify the operating system to hide files, folders, Registry entries and processes.

Therefore, by performing a live acquisition, extra elements of uncertainty are introduced. This uncertainty takes the form of an alternative history of the examined digital evidence where the output of the acquisition tool has not duplicated the input data correctly (faulty imaging tools, e.g. missed one or more sectors), and/or the operating system has supplied data to the acquisition process that does not represent the data on the system ('faulty' operating system). These elements of uncertainty currently cannot be ruled out, making the assessment of error and therefore assessment of accuracy difficult.

6.2.6 Summary

One of the requirements described in Chapter 3 is that during a digital investigation it should be possible to assess the amount of error associated with all techniques used. This section has defined the error associated with digital evidence based on its main purpose in a digital investigation, which is to "make valid inferences about a computer's history" (Carrier, 2006a). Therefore, based on this aim and the definition in Oxford (2008), error associated with digital evidence is defined as *the difference between the inferred history and the true history of the examined digital evidence*. The possible error or uncertainty is expressed as *alternative possible events that explain the current state of the examined digital evidence*. This definition and expression of error has the advantage that it can be used to assess error in both the analysis and acquisition stages of a digital investigation. This is because it considers events in the history of the *examined* digital evidence, which includes both its change in state on

the system in question, and its change in state from raw data on the physical evidence, to acquired raw data, to data in a form that can be understood.

Error in the analysis stage of a digital investigation has been discussed and it was shown that it can be assessed; this is because the data that is being analysed can be exactly duplicated and can be examined by multiple investigators who can form and test their own hypotheses and reach their own conclusions about the most probable events that created pieces of digital evidence. They can also qualify their conclusions based on the agreement of different sources that support that conclusion using a Certainty Scale such as that in Table 20.

The analysis stage does not present a particular problem for live investigations, since if the acquisition and analysis stages are separated, then data acquired from a live system has the same properties as data acquired in a traditional digital investigation. These properties are that data can be exactly duplicated and examined multiple times by multiple parties. However, uncertainty can be introduced during the acquisition stage of a live digital investigation, since it is difficult to demonstrate that in the history of the examined digital evidence, during the acquisition process, the actual data on the system was not captured incorrectly. This could either be due to acquisition tool error or the operating system supplying data to the acquisition process that does not represent the actual data on the system. The accuracy of a live investigation is therefore dependent on demonstrating the accuracy of the acquisition stage. As a result, this research examines how the accuracy of the acquisition of digital evidence from live systems using encryption can be assessed.

6.3 METHODOLOGY

This chapter examines how encrypted digital evidence from a live system can be acquired in a way that allows its accuracy to be demonstrated using repeatability, since this is used successfully in traditional digital investigations. The proposed approach involves recovering data from the live system that allows the encrypted data to be decrypted offline in a trusted environment. This means that since the decryption is done offline, data is static and the inputs to the decryption process are constant.

Decryption can be therefore performed in a repeatable manner in a trusted environment allowing the accuracy to be assessed and alternative hypotheses of faulty acquisition tools or manipulated operating systems to be ruled out. The only data used where the accuracy cannot be determined through repeatability is that which is used to allow offline decryption, since that data is acquired from a live system and will no longer exist after the power is removed. However, the accuracy of this particular piece of digital evidence can be assessed in another way: it can be demonstrated to be correct if it successfully decrypts the encrypted data, since if it were not correct then the data would not decrypt.

First it is shown how such information can be obtained from a live machine that is running an encryption product that allows recovery keys to be exported as part of the product's design. It is then shown how these keys can be used offline to recover the decrypted contents of an encrypted drive. This is demonstrated using *BitLocker* in *Windows Vista*. Secondly, since not all products offer such a key recovery feature, it is also shown how decryption keys can be recovered from the memory of a live system which are then used to decrypt encrypted data offline in a repeatable manner. This is demonstrated using *TrueCrypt*.

This approach for key recovery from memory differs from that discussed in Chapter 2 (locating copies of the key or passphrase on the disk or in the surrounding area) where keys are searched for on the powered down disk or on physical media at the crime scene. The approach described in this chapter is different since a live investigation is used specifically to recover keys prior to the power being removed, rather than hoping that they can be subsequently located post-seizure. The following two sections describe the GUI based key recovery using *BitLocker* and key recovery from a memory dump using *TrueCrypt*.

6.4 GUI BASED KEY RECOVERY: *BITLOCKER*

6.4.1 Introduction

This section demonstrates how the recovery keys of *Windows Vista's BitLocker* can be obtained by performing a live investigation. These keys can then be used offline to obtain a disk image of the decrypted data in a repeatable manner allowing the accuracy of the acquisition stage to be assessed using repeatability.

6.4.2 *BitLocker* Background

BitLocker is the Full Volume Encryption feature built into particular versions of *Windows Vista* (Enterprise and Ultimate). It offers five different modes which differ in how the decryption keys are protected: Trusted Platform Module (TPM) only, TPM & PIN, TPM & USB, TPM & PIN & USB and USB only. These were explained in detail earlier in Section 4.4.4. Regardless of the mode in use, *Windows Vista* will always provide the option for a recovery key. The purpose of this is to allow access to encrypted data if the decryption keys are lost, the PIN is forgotten or the encrypted volume is moved to another system (Microsoft, 2006b). Therefore, the recovery key will unlock the volume without the need for the PIN to be supplied or the USB stick or TPM to be present.

During the *BitLocker* life-cycle recovery keys are created prior to encrypting the drive and can be stored on USB drives, any other accessible folder, or printed (Microsoft, 2006b). Even though they are created prior to the encryption, it is also possible to create additional copies of these recovery keys at any point after the volume has been encrypted. This can be performed simply, using a graphical user interface in *Vista*, which is described in Section 6.4.4.

6.4.3 Identification of *BitLocker*

Before obtaining recovery keys for *BitLocker*, it first needs to be identified. As mentioned earlier, *BitLocker* is only available in *Enterprise* and *Ultimate* editions of

Vista (Microsoft, 2006f) so identification of other versions of *Vista* can rule out the presence of *BitLocker*³⁹. If *Enterprise* or *Ultimate* versions are installed, due to the requirement for a 1.5 Gigabyte system partition which must remain unencrypted (Microsoft, 2006b), drive partitioning such as this may be the first sign to indicate the presence of *BitLocker*. There are also scripts that can be run on a live system that report the status of each volume and will describe if *BitLocker* is running (HogFly, 2007, Microsoft, 2007b). The output of the built in Microsoft script is shown in Figure 24 and shows a protected volume.



```
Administrator: C:\Windows\System32\cmd.exe
C:\Windows\system32>cscript manage-bde.wsf -status
Microsoft (R) Windows Script Host Version 5.7
Copyright (C) Microsoft Corporation. All rights reserved.

Disk volumes that can be protected with
BitLocker Drive Encryption:
Volume C: [I
[OS Volume]

Size: 40.04 GB
Conversion Status: Fully Encrypted
Percentage Encrypted: 100%
Encryption Method: AES 128 with Diffuser
Protection Status: Protection On
Lock Status: Unlocked
Key Protectors:
  External Key
  Numerical Password

C:\Windows\system32>
```

Figure 24: Results of running the built-in *manage-bde.wsf* script identifying encrypted volumes on a live system.

6.4.4 Obtaining Recovery Keys

Once *BitLocker* has been detected, the recovery keys can be obtained using the graphical user interface. This can be accessed through the Control Panel and the option 'Manage *BitLocker* Keys' in the 'Security' sub-section. The interface is shown in Figure 25.

³⁹ This can be achieved on a live system using the *psinfo* tool or by examining the system properties through the Control Panel.

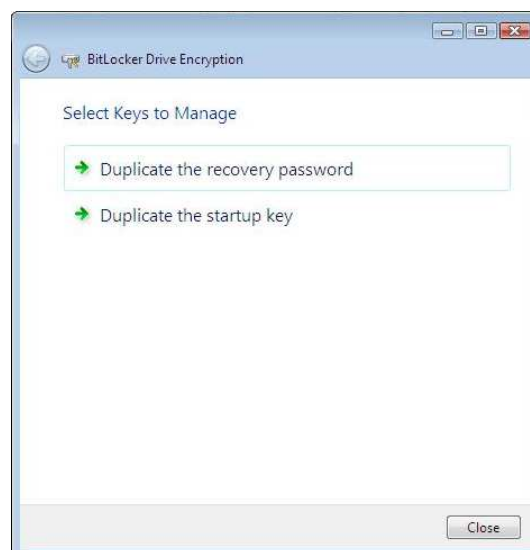


Figure 25: The 'Manage BitLocker Keys' graphical interface.

Using this interface, a copy of the recovery keys can be created on an attached USB device. Sample recovery keys are shown in Figure 26.

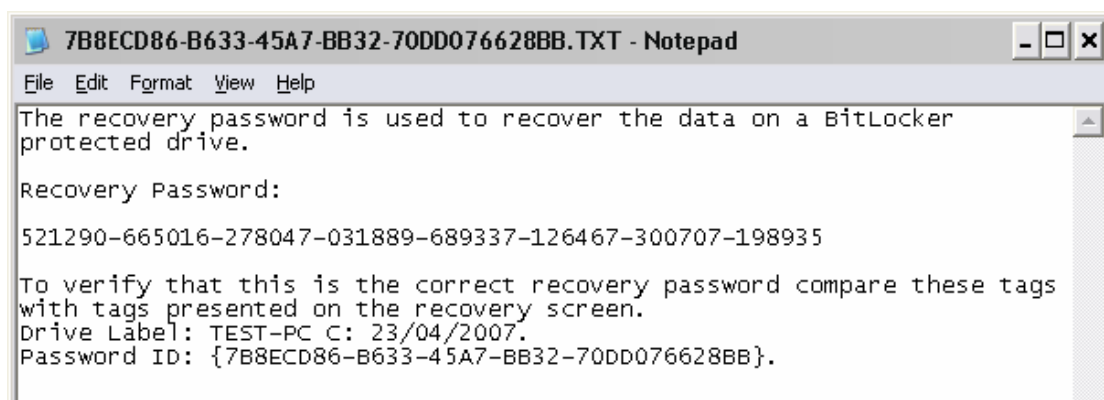


Figure 26: The format of *BitLocker* Recovery Keys.

6.4.5 Acquisition of a Powered Off System Using Recovery Keys

After seizure, in the TPM only scenario, the machine can be booted normally with no intervention and in the other scenarios the recovery keys can be provided (typed using the Function keys). After this, the system can be booted in order to create a logical disk image. However, this is not a desirable way in which to recover evidence since it boots an already powered off system in order to perform a live investigation which

makes unnecessary changes to the original evidence and will affect the completeness of the preserved evidence.

Changes to the original evidence can be avoided by creating a physical disk image of the original drive (in encrypted form) and converting it to a *VMware* virtual machine disk file using *LiveView* (Kaplan, 2007). At this point, the virtual machine disk image can be booted which causes *Vista* to enter recovery mode since the hardware has changed. Entering the recovery key boots a virtual version of the original system and makes it possible to create a logical image in unencrypted form. Note that in this case, even in the TPM only scenario, the recovery keys are necessary since the TPM is not present in the virtual machine used to boot the disk image. A disadvantage of this approach is that it relies on the suspect's operating system, which as described earlier could introduce uncertainty. There is also the problem that even after booting the virtual copy of the machine it is possible that a *Windows* password will then be necessary in order to log on to the system, which may not be available.

To resolve these problems a second virtual machine can be created⁴⁰, *Vista Ultimate* edition installed and the converted image from the original machine added to the new system as a second virtual drive. By booting into this freshly installed virtual *Vista* machine and using the *BitLocker* interface it is possible to unlock the drive using the recovery keys and produce a logical, unencrypted image of the original drive using a trusted operating system⁴¹. This method also has an advantage over connecting the original physical drive through a write blocker to a real *Vista* system to perform this drive unlocking, since the initial disk imaging stage remains the same as any other investigation. This disk image is then used for the entire analysis and recovery of evidence. This may be useful if those performing the analysis and attempting to gain access to the encrypted data are not the same as those who performed the initial disk imaging.

⁴⁰ Which needs to be done only once.

⁴¹ It is also possible to mount the virtual drive on a real system using software such as Mount Image Pro. Neither offer particular advantages and it depends on the software available.

6.4.6 Evaluation

There are a number of limitations to this approach; most significantly that few encryption implementations offer such an easy to access key recovery system, and as a result, this technique does not generalise well. There are also potential problems to using this approach for *BitLocker* on *Windows Vista*, if User Account Control (UAC) is enabled. The goal of this feature is to enable users to run standard privilege accounts, only escalating privileges to administrator when necessary (Microsoft, 2006e). UAC has implications for obtaining the recovery key since running the *BitLocker* management tool requires administrator privileges. If the account is an administrator then the keys can be obtained since running processes as administrator only requires clicking 'Allow' in the UAC prompt. However, if the account is a standard user then a password is required to access the mechanisms to create backups of *BitLocker* recovery keys.

Also, this method may violate the End User License Agreement from Microsoft, since even the *Windows Vista Ultimate* License states that *BitLocker* may not be used with Virtualization Technologies (Microsoft, 2008a). However, in a forensic computing situation it is unknown how this will affect law enforcement.

It is also conceivable that the suspect's operating system could be modified to supply a false recovery key that will not decrypt the drive. However, this is speculation and has not been investigated.

6.4.7 Summary

BitLocker Recovery Keys can be recovered from *Windows Vista* using the built in GUI regardless of the mode used. This allows a disk image of the encrypted system to be acquired in decrypted format in a trusted environment. This allows the accuracy of the acquisition to be demonstrated since the technique is repeatable. The live acquired data can be shown to be the same as the offline acquired data.

However, there are problems, including the difficulty obtaining keys if UAC is enabled and the user account does not have administrator privileges. Despite this, the principle of key recovery followed by an offline decryption to assess the accuracy of

live acquired images is sound and if a more flexible key recovery approach (one that would generalise) could be used then many of the limitations of the approach could be overcome.

6.5 MEMORY IMAGE BASED KEY RECOVERY: *TRUECRYPT*

6.5.1 Introduction

The previous section showed how decryption keys could be recovered from a system running *BitLocker* using the built-in interface. This section describes a technique for recovering the keys from a memory dump of a system, and uses them to demonstrate the accuracy of live acquired disk images. This is demonstrated using the popular, open source product *TrueCrypt*.

6.5.2 *TrueCrypt* Background

Introduction

TrueCrypt is a “software system for establishing and maintaining an on-the-fly-encrypted volume” meaning that “data are automatically encrypted or decrypted right before they are loaded or saved, without any user intervention” (TrueCrypt, 2008b). *TrueCrypt* has become a popular tool for encrypting data with over 8 million downloads (December 2008) (TrueCrypt, 2008a).

Version History

At time of writing *TrueCrypt* is at Version 6.1a, having last been updated in December 2008 (TrueCrypt, 2008f). Various features and bug fixes are implemented between versions. Also the default encryption scheme changes: Cipher Block Chaining (CBC) was used prior to V. 4.1, Liskov, Rivest & Wagner (LRW) for V.4.1 - 4.3a and it currently uses XOR Encrypt XOR based, Tweakable Codebook Mode, Ciphertext Stealing, (XTS) (V.5.0 onwards). Another important difference is that in

the most recent versions (V.5.0 onwards), *TrueCrypt* offers more advanced options than just creating an encrypted container⁴², which are discussed below.

Encrypted container: This is an encrypted file stored on disk that contains an entire file system that can be mounted as a drive letter in order to be accessed (Virtual Disk Encryption in Chapter 4).

Encrypted Volume: Here an entire partition on a drive is encrypted. However, the partition table is still accessible in the clear, even during a ‘dead’ analysis. The partition is visible to *Windows* as a drive letter but appears unformatted. The decrypted partition can then be mounted with *TrueCrypt* as a different drive letter.

Encrypted Drive: Here the entire drive is encrypted including the partition table. The entire drive from sector 0 onwards appears as random data. The drive is not accessible to *Windows* as a drive letter but can be seen through the Control Panel disk management tool.

System Partition or Drive: Here either the drive or the partition containing the operating system is encrypted.

The *TrueCrypt* Decryption Process

This section describes the operation of *TrueCrypt* and how containers are decrypted. This is necessary to understand the details of the key recovery technique used later.

TrueCrypt encrypted containers appear to contain nothing but random data and have no file signature. Prior to Version 6 of *TrueCrypt* the first 512 bytes of a *TrueCrypt* container are actually a header⁴³, but are encrypted using a Header Key so still appears to be random data. *TrueCrypt* decrypts the header using a user-supplied

⁴² Encrypted containers are referred to as Encrypted Virtual Drives in Chapter 4.

⁴³ The start of containers from Version 6 onward is still a header, but data begins at offset 131072 .

password or keyfile, salt⁴⁴ from offset 0-64 (bytes) and then the process of trial and error using different encryption and key derivation algorithms and modes of encryption (CBC, LRW, XTS etc.). Successful decryption of the header is when bytes 64-67 decrypt to the ASCII string 'TRUE'. The entire header is then decrypted which in the case of XTS mode, contains the Master Key and Secondary Master Key (Tweak Key) needed to decrypt the actual contents of the container, from the 'Data Area' which begins at offset 512 prior to Version 6.

XTS Encryption Mode

As mentioned in the version history, since *TrueCrypt* Version 5.0, LRW mode has been replaced with XTS. This section outlines this encryption mode's operation but full details are available in IEEE (2007). The data to be stored is divided into *data blocks* greater than 128-bits. Each of these *data blocks* is then divided into 128-bit *sub-blocks*. XTS uses two keys (key1 and key2) and each plaintext *sub-block* is encrypted with key1. However, before and after the actual encryption the *sub-block* is XORed with a tweak value calculated using the index of the block, the index of the sub-block and key2. This tweak value is calculated by encrypting the *data block* index with key2. This is then multiplied by 2 to the power of the *sub-block* index, modulo the polynomial $x^{128} + x^7 + x^2 + x + 1$. The overall process is depicted in Figure 27.

⁴⁴ Salt is used to prevent pre-computation of password hashes.

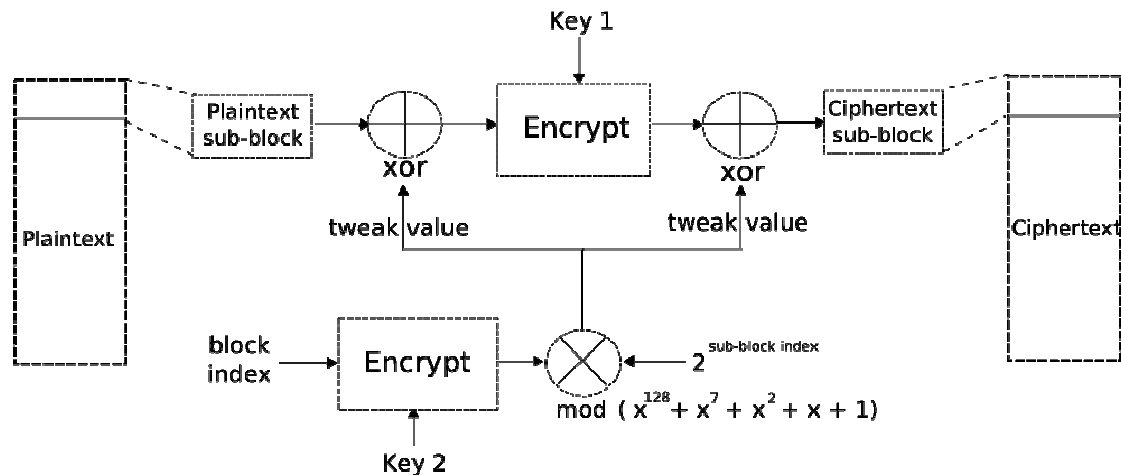


Figure 27: XTS encryption of a single sub-block.

6.5.3 Methodology

The technique described in this section demonstrates the accuracy of a live acquired disk image of encrypted data using key recovery from a memory dump. The encrypted data is referred to as an encrypted container throughout this section, but the same technique can be applied to containers, volumes and drives, including those containing the operating system.

The live acquired image of the contents of the encrypted container can be analysed but if it then becomes necessary to demonstrate the accuracy of the acquisition then an additional process can be carried out. Decryption keys can be recovered from the memory dump and used to decrypt the offline encrypted container. This can then be compared against the live acquired image and shown to be consistent, demonstrating the accuracy of the live acquired image since the decryption of the offline container was performed in a trusted environment and can be repeated.

This is illustrated in Figure 28 and the full methodology that allows the recovery of keys and demonstration of the accuracy of acquired images of encrypted containers, from the point of encountering a live system, is:

1. Identify the type of encryption software on the system, to determine whether the technique needs to be applied.

2. From the live machine, acquire the contents of the encrypted container.
3. From the live machine, acquire the physical memory.
4. Offline, recover keys from memory image.
5. Decrypt the offline encrypted data using the recovered keys.
6. Determine if the decrypted offline copy is consistent with the live acquired image.

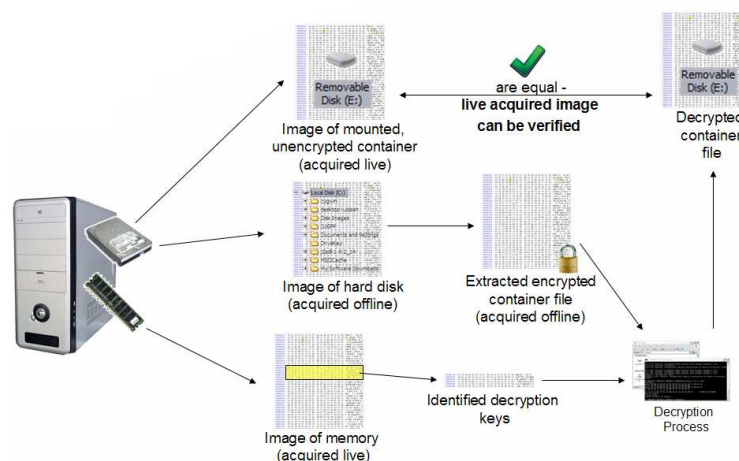


Figure 28: The overall approach for demonstrating accuracy of live acquired images.

6.5.3 Key Recovery Technique

One of the stages in the above methodology is the recovery of decryption keys from memory. There are a number of options for this, which are described in the following section. However, the specific technique used is not critical to the overall methodology.

Existing Key Recovery Techniques

There are a number of existing techniques to recover information from memory that may allow access to encrypted data. While not decryption keys, an anonymous work (anon, 2007) and Bolieau (undated) describe that some Full Disk Encryption packages cache plaintext passwords at offset 0x417 in memory images, including *PGP Desktop*. *TrueCrypt* can also cache plaintext passwords in the *TrueCrypt* driver memory if the

“cache passwords and key files in memory” option is selected (TrueCrypt, 2008d). However, these options are limited to particular products.

TrueCrypt key recovery from a *Linux* memory dump is described in Walters and Petroni (2007), where the operating system’s data structures are parsed and the master keys recovered from a clearly identifiable variable. This approach has also been extended to *Windows* in Kaplan (2008) where the *TrueCrypt* driver is located in memory and the keys extracted from particular offsets.

Also Halderman *et al.* (2008) describes a key recovery method that specifically takes into account bit errors introduced during cold boot memory acquisition techniques (See Chapter 2.3.4). The approach involves searching for data other than the key (the key schedule which stores pre-computed data for rounds of encryption for performance reasons) and using it to recover the key.

Since the approaches in Kaplan (2008) and Halderman *et al.* (2008) were not made public until late into this research, the following sub-sections describe the development of an additional key recovery technique.

Overview of Developed Key Recovery Approach

The overall developed key recovery approach uses a linear scan of a memory image, using each consecutive position in that image, extracting possible keys according to an identified pattern and attempting to decrypt the container. The correct keys are identified when the container successfully decrypts. In this sense the overall approach of the technique can be described as a dictionary attack on the key from a limited key space generated from the memory of the system. The overall key recovery approach is shown in Figure 29 and the following subsections describe each of the stages of developing the key recovery methodology.

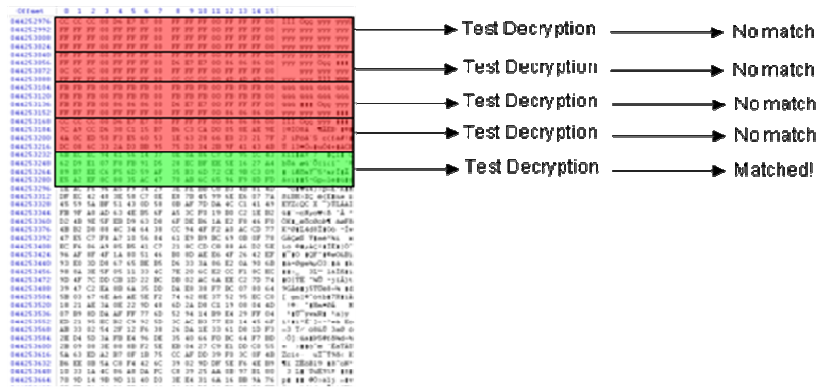


Figure 29: Depiction of the linear scan approach to key recovery. The keys used to perform test decryptions actually slide one byte at a time, rather than 256 as shown here.

Setup of test environment

For the development of this technique *VMware Workstation* was used to create a virtual *Windows XP Professional* machine. *TrueCrypt* was installed on the virtual machine and an encrypted container created with the password set to be ‘password’. After mounting the container the virtual machine was shut down and rebooted. The encrypted container was mounted using *TrueCrypt* and the appropriate password. With the container mounted, the virtual machine was paused and a copy of the .vmem file representing the virtual system’s RAM created.

Identifying patterns in memory

During the initial setup, when an encrypted container is created, the *TrueCrypt* graphical interface displays parts of the keys used to encrypt the container, as shown in Figure 30.

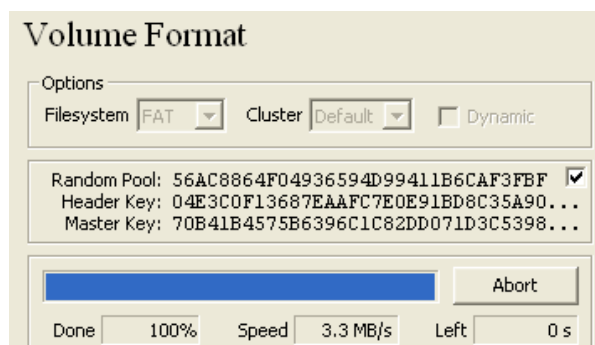


Figure 30: Parts of the keys displayed by the *TrueCrypt* GUI on creation of an encrypted container.

It is this property of *TrueCrypt*, rather than it being open source that allowed the pattern matching to be easily developed. Without this shortcut, establishing patterns in memory is much more complex and is discussed later in the evaluation section. Since the keys are known, they can be easily identified in the memory dump and a clear pattern in memory identified, as shown in Figure 31. Experiments showed the first 256 bit block to be the Master Key for the container and the second 256 bit block to be the Secondary Key for XTS (keys are reversed prior to Version 4.3a, see Hargreaves and Chivers (2008b)).

```

26018160 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26018176 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26018192 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26018208 | 70 B4 1B 45 75 B6 39 6C 1C 82 DD 07 1D 3C 53 98 p' Eu#91 |Ŷ <S|
26018224 | 87 C8 AC 29 04 5C DA CD 35 22 87 2C CF 20 7B 10 |È-) \Ůí5" |. Ĩ {
26018240 | 2D AB 90 43 DF 82 00 1D 50 B5 A6 54 8D 8C 0F DE -«|CB| Pp|T|| P
26018256 | 41 3B 5A 66 0D 4A C0 6A 9F 55 3F E1 EE C7 3E |C A:Zf JÀj|U?áíç:
26018272 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26018288 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
26018304 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Figure 31: Part of a memory image showing the Primary and Secondary Master keys.

It should be noted that the Header key cannot be found at all in memory. This is because the Header key is only necessary for *TrueCrypt* to decrypt the container header and extract the Master and Secondary keys, which are then used to decrypt the rest of the container. Therefore, once the Master and Secondary keys are stored in memory, the Header key is no longer needed and can be erased.

Once the patterns of the keys are identified, it is simply necessary to linearly scan memory, extracting keys in this pattern until the correct keys are found. However, it is first necessary to find a way to identify the correct keys.

Identifying Correct Keys

This key recovery technique uses known plaintext to identify the correct keys. During normal *TrueCrypt* operation the string ‘TRUE’ is used to show correct decryption of the header. From the header, the Master and Tweak keys are extracted, known to be

correct and are used to decrypt the data in the container. A similar known plaintext approach was developed to test for correct Master and Secondary keys; although, there are alternatives which are mentioned later in the future work section. Suitable plaintext was identified by creating, mounting and imaging several containers, and examining the images for consistent plaintexts. Offsets 3-7 of a mounted, decrypted 10 Megabyte FAT formatted container, decrypted to ASCII ‘MSDOS’, as shown in Figure 32. These correspond to offsets 515-519 of the encrypted container (skipping 0-512 which is the *TrueCrypt* header encrypted by the Header Key and not accessible).

00000000	EB 3C 90 4D 53 44 4F 53	35 2E 30 00 02 01 02 00	ë< MSDOS5.0
00000016	02 00 02 FF 4F F8 50 00	01 00 01 00 00 00 00 00	ÿOøP
00000032	00 00 00 00 00 00 29 9B	E1 46 47 4E 4F 20 4E 41)!±FGNO NA
00000048	4D 45 20 20 20 20 46 41	54 31 36 20 20 20 00 00	ME FAT16
00000064	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000080	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	

Figure 32: Known plaintext on FAT16 file systems.

Larger, FAT32 file system based containers were also examined and the known plaintext ‘MSDOS’ can still be used. However, NTFS containers have the string ‘NTFS’ at offsets 3-6 which needs be used to identify correct decryption of the data area of an NTFS formatted container.

There are also differences depending on whether a container, volume or disk is encrypted. As described above, for a container, the known plaintext file system data is located 3 bytes into Sector 1 (sector numbers begin at 0). For an encrypted volume the known plaintext is in Sector 64, Sector 1 for an encrypted Drive and Sector 63 for System Partitions and Drives. A summary of positions for the *TrueCrypt* headers and known plaintexts is shown in Table 21.

Mode	Sector on disk containing <i>TrueCrypt</i> header	Sector on disk containing known plaintext
Container	Variable	Variable ⁴⁵
Volume	63	64
Drive	0	1
System Partition	62	63
System Drive	62	63

Table 21: Positions and indexes of known plaintext in different modes of TrueCrypt.

Decrypting the container from the master keys

Software was developed in C that used sample AES decryption code from Devine (2006) which after implementing the modes of operation, allowed parts of the container to be decrypted from supplied keys. The developed software is compatible with LRW mode and the newer XTS encryption mode. Since the known plaintext strings ('MSDOS' and 'NTFS') are located at offsets 3-7 of the known-plaintext sector, and AES decrypts in blocks of 128-bits, the known plaintext resides in the first block of the data area of the encrypted container. This means that only one block needs to be decrypted.

Automating key recovery

Once a means of identifying correct keys was developed it was then necessary to automate the process of trying test keys so it could be applied to the entire memory dump. Software was developed in C, to scan through the whole of memory, using each 48 byte block as Master Keys and Secondary Keys in a fixed pattern, as shown earlier in Figure 29. The 'window' from which keys were obtained moves through memory one byte at a time, so for example in a 512 Megabyte memory image:

$$512 \times 1024 \times 1024 = 536,870,848 \text{ bytes}$$

$$536,870,848 - 64 \text{ ('window' size)} = 536,870,848$$

⁴⁵ Position is variable on disk since containers can be stored in various places in the file system. However, the TrueCrypt header is in Sector 0 of the container and the known plaintext is in Sector 1 of the container.

This means there are 536,870,848 possible key positions in the full 512 Megabyte memory dump.

As mentioned, the developed software only decrypts the first block of the data area of the container since that is all that is needed to determine if the keys are correct or not and allows significantly faster operation.

6.5.4 Results

Key Recovery

The developed software successfully recovers encryption keys from a memory dump of a live system. It has been tested on and successfully used with memory dumps obtained from *VMware* and using *dd*. Figure 33 shows the output of the *getkeys* program.

TrueCrypt header, which is then used to decrypt the master and secondary key stored in the header. It is not therefore possible to use the recovered master and secondary keys to directly open the container. However, as described in Walters and Petroni (2007) “with a few minor changes to the [*TrueCrypt*] mounter, we can use the extracted cryptographic information to mount the volume offline without the password.” Another approach is to extend the decryption code used to test for known plaintext and use it in a loop to decrypt the entire container or volume. This second approach is used to decrypt the container which can be analysed in the usual manner or compared against the already analysed live acquired image.

Comparison of two acquired images of a mounted encrypted container (one acquired from the live system using *dd*, the other acquired offline by decrypting the container using keys extracted from memory) showed the containers to be identical. This can be seen by the hashes in Figure 34.

```
H:\TrueCrypt 5\data>md5 disk<live>.dd
4EF0C370EED3FDB1DDE311AB3C25F971  disk<live>.dd

H:\TrueCrypt 5\data>md5 dec.bin
4EF0C370EED3FDB1DDE311AB3C25F971  dec.bin
```

Figure 34: MD5 hashes of live acquired container and offline decrypted container.

However, comparison of two acquired images of a system running *TrueCrypt* Full Volume Encryption showed the acquired images not to be identical. This is due to live systems constantly changing, as discussed in Chapter 5. However, using the same technique used in Section 5.4.3, *md5deep* (Kornblum, 2008) can be used to obtain hashes of individual files on the acquired image. The hashes of files in both acquired disk images can then be compared. The results from an acquisition of a Full Volume encrypted system are shown in Figure 35 and show that 4 files changed between the live acquisition and the system being powered off. The differences were identified using the developed software *md5listcompare*.



```
diffs.txt
Deleted files
-----

Created files
-----

Modified files
-----
WINDOWS/Prefetch/LOGON.SCR-151EFAEA.pf
WINDOWS/system32/wbem/Repository/FS/INDEX.MAP
WINDOWS/system32/wbem/Repository/FS/MAPPING.VER
pagefile.sys
```

Figure 35: Differences between the live acquired Full Volume Encrypted drive and the offline decrypted version.

For the files on the encrypted disk that are not listed in the figure above, their accuracy has been demonstrated since the offline acquired versions are consistent with those acquired from the live system. The offline acquisition can be performed multiple times using different acquisition tools which reduces the likelihood that the image is incorrect due to faulty imaging tools.

However, regarding the files that have changed between the images, these could be excluded from the investigation as the live acquired versions cannot be shown to be the same as those obtained in a repeatable manner using a trusted operating system. However, if the files are important to the investigation, the files can be inspected more closely and the parts of the files that are the same can be used, with just the parts of the files that have changed excluded. It may also be possible to explain the discrepancies in detail, for example the change to logon.scr (see Figure 35) occurred 10 minutes after the acquisition began which was when the screensaver on the live machine activated (in practice this activation of the screensaver should be prevented). The extent of the differences between the live and dead images therefore depends to a certain extent on the actions of the investigator, although in this case the changes caused by running *dd* (prefetch file, Registry changes, etc.) were recorded in both the live and dead acquisitions since the live imager acquired the parts of the disk that contained those changes after they were made.

However, other software running on the system may increase the number of differences, e.g. antivirus scans, Windows updates etc., although it should still be

possible to explain these differences. Also the longer the system is running before the power is removed the longer the system has to make changes to the drive. However, given that no differences were found with the live and dead acquired encrypted containers, differences are most likely to be a concern with Full Volume Encryption. In these cases background changes to the system are very small in relation to the total size of the data that is preserved. In this example a small 4 Gigabyte drive was used and still only 0.09% (3809953 bytes) was different.

Referring back to Section 6.2.5, it was explained that by performing a live acquisition, extra elements of uncertainty were introduced: the output of the acquisition tool could incorrectly replicate the input (faulty imaging tool), but also that the live operating system could provide the acquisition tool with incorrect data ('faulty' operating system, e.g. a rootkit). In addition to addressing the alternative hypothesis of a faulty imaging tool, by acquiring data offline using a trusted operating system, this eliminates the possibility of data being hidden due to the operating system behaving incorrectly (i.e. due to a rootkit) since rootkits cannot operate while the compromised operating system is not running and hidden files will be visible. This method therefore provides the opportunity to eliminate both of these alternative hypotheses since the encrypted data can be reacquired multiple times using multiple tools (addresses faulty acquisition tools), and it is possible to detect if the live operating system is incorrectly reporting its state, since when the offline acquisition is performed the suspect operating system is not running and therefore it is not possible for installed rootkits to run (addresses 'faulty' operating system).

6.5.5 Evaluation of this Key Recovery Approach

There are a number of limitations to this particular key recovery approach. One practical limitation is that the software has been developed only for AES in LRW and XTS mode. *TrueCrypt* supports a number of other algorithms including Serpent and Twofish. It could be argued that as the number of algorithms and modes increase, the number of combinations that need to be tried for each test key makes using this approach more of a challenge (Kaplan, 2008). Since the technique is simple and fast

due to only needing to decrypt a single block, even increasing the time by nine using three algorithms in three modes (AES, Serpent and Twofish with CBC, LRW and XTS), key recovery should still be possible in less than 5 hours. There are also many optimisations possible: sliding the key window more than 1 byte at a time; performing a simple entropy test on data prior to trying decryption, and rewriting the code to use multiple CPU cores or multiple machines; all of which will reduce the time taken to recover keys. In addition, at the stage where encryption is identified on a system, it may be possible to determine this information. *TrueCrypt*, for example, allows the properties of mounted containers to be viewed on a live machine, which describes the type of encryption (Container, Volume, Drive), the encryption algorithm and the mode used. This is shown in Figure 36.

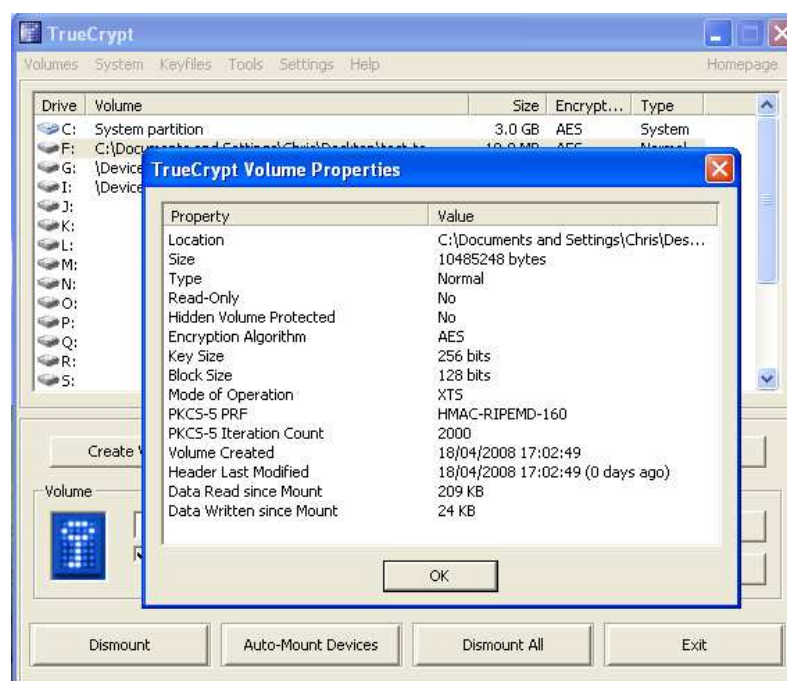


Figure 36: Properties of a mounted encrypted container identified on a live system.

A specific limitation of the linear scan approach is that it relies on keys being stored in consistent patterns in memory. It is conceivable that keys could be split in memory; however, this is simply a more complex pattern that would need to be identified. Introducing a random element to the storage location of keys is one way to hamper the

use of this technique. However, by design, keys need to be constantly accessible to on-the-fly encryption products, and even if the key was split over randomly spaced locations, the encryption software would need to keep track of these. In this situation, to recover keys it would be necessary to have a greater understanding of the internal operation of the software in question, but as described in Kaplan (2008), “given enough time, both the secret key and the exact details of each cryptosystem’s operation can be discovered”.

Also, the known plaintext used to identify successful decryption of the container is non-essential data, meaning that this can be changed without hampering the operation of the container. However, it is possible to change the scanning process so that the known plaintext used is essential data⁴⁶ or use statistical techniques to identify possible correct decryptions.

While this key recovery approach has been developed for only one product (albeit two versions), the work in Halderman *et al* (2008) has shown that keys are also available in memory for *BitLocker*, *FileVault* and *dm-crypt*. It also stands to reason that due to the inherent design of any on-the-fly encryption software, the keys have to be accessible in order to perform decryption, since the same keys are used for each block they need to be constantly accessible. Therefore, any product that does not decrypt all of the plaintext at once and decrypts data as it is needed is susceptible to some key recovery approach.

In addition to its generalisability, this approach overcomes the limitations of the GUI based approach where UAC could prevent keys from being recovered. While *dd* memory acquisition techniques require administrator privileges, a Firewire memory acquisition could be performed without needing to provide a password to UAC. It also is not affected by the use of duress keys, provided the seizure is performed when the real encrypted data is mounted, rather than the duress data. This is because the technique will recover the keys from memory that allow the currently mounted encrypted container to be decrypted offline.

⁴⁶ Such as the starting address of the root directory or number of FATs (Carrier 2005 p.214)

However, it is necessary to obtain the memory dump from the system while the encrypted data is in use i.e. in the case of container or volume encryption, it needs to be mounted. However, this is implementation specific since *TrueCrypt* securely wipes the keys once volumes are dismounted. Also, for any system volume, removing the keys from memory is not possible until the system is shut down.

The identification of key patterns in memory for this product was trivial, not due to the open source nature of *TrueCrypt* but because the GUI design reveals part of the keys during the creation of containers. However, any open source product could be modified to display this data in this way to assist in determining key patterns. For closed source products, a number of approaches are still being explored including reducing a memory dump to data that has changed before and after mounting a container and correlating memory dumps from multiple systems mounting the same container. Also, the use of a debugger may allow these keys to be viewed as the program executes. However, this is ongoing work.

Another limitation that is unrelated to the accuracy requirement but is important practically, is that it cannot be known in advance if the keys can be successfully extracted from memory, e.g. memory may not have been correctly acquired. The acquisition of and key recovery from memory is therefore presented as an additional step as well as acquisition of the mounted encrypted containers and volumes. The key recovery is used if necessary to defeat challenges about the accuracy of an otherwise unverifiable live container image. However, a solution to this limitation is provided in Hargreaves and Chivers (2008a), where keys are verified offline but still at the scene, allowing live imaging to be avoided if necessary; but this is outside the scope of this research.

Finally, as mentioned earlier, limitations of this specific key recovery approach are not critical to the overall methodology for demonstrating accuracy of acquired encrypted containers or volumes since alternative or multiple key recovery techniques can be substituted.

6.5.6 Key Recovery Summary

The accuracy of live acquired encrypted data has been demonstrated by recovering decryption keys from a dump of the system's memory which was acquired at the same time. These recovered keys can be used to decrypt the static, offline data in a repeatable manner in a trusted environment, and the decryption can be used to verify the live acquired image. The only data that cannot be verified through repeatability is data that is inconsistent due to unavoidable changes on the live system and also the acquired memory image. As discussed earlier, files that are inconsistent due to the live system changing files can be inspected more closely to determine the exact nature of the discrepancies. Regarding the memory image, the only part of it that is used as evidence is the decryption key, the accuracy of which is evident since it successfully decrypts the encrypted data. In the Certainty Scale in Casey (2002a), this particular digital evidence artefact (the decryption key) achieves C6 (the highest level of certainty), meaning it is "tamperproof and unquestionable". This key recovery approach has been demonstrated using the open source product *TrueCrypt* for all modes of operation: container, volume and drive encryption.

6.6 EVALUATION

Both methods of key recovery shown in this chapter, GUI based and linear memory scanning, have shown how encrypted data can be accessed offline in a trusted environment where the process is repeatable. Using repeatable techniques and trusted environments are existing and accepted ways in which accuracy of acquisitions can be assessed.

Therefore, while there are specific limitations of individual approaches, e.g. UAC for the GUI based approach on *Vista* and countermeasures for key recovery from memory, e.g. splitting the key, the multitude of approaches for obtaining keys means that if the encrypted data is being used on the live system, then keys must be stored somewhere and will be recoverable.

Allowing the accuracy of encryption to be assessed in this way reduces the possible error by eliminating the alternative hypotheses that the analysed digital

evidence has its value because the operating system supplied incorrect information to the live acquisition tool, e.g. due to the presence of a rootkit. This is because in the offline environment any malware on the evidence being examined cannot run. It can also eliminate possible error introduced due to the acquisition tool not operating correctly, since once the offline data is decrypted, multiple acquisition tools can be used to acquire data and it can be shown to be the same as the live acquired data.

Referring back to Casey's certainty levels (Section 6.2.4), use of this method increases the certainty in the live acquired encrypted disks from C2 ("only one source of evidence that is not protected against tampering") to C5 ("multiple independent sources that are protected against tampering, however small uncertainties exist" (in this case these small uncertainties are the differences between images due to unavoidable file changes on a live system)).

However, this method does not fully address the logic bomb problem, where software could be configured to erase data⁴⁷ on the system given certain conditions e.g. adding a USB stick or running a certain piece of software. However, systems that are used to store encrypted data still need to be usable and preventing the use of USB sticks will reduce the usability of the system. It may also be possible to address this by examining the system prior to inserting a USB stick or running software to search for such 'logic bombs' or to examine the acquired images for traces left by the use of logic bomb software. However, this remains future work.

A consequence of relying on offline repeatability to demonstrate accuracy of data from live acquisitions of encrypted data (i.e. performing a dead acquisition) is that this approach does not generalise to other live investigations e.g. demonstrating the accuracy of a memory dump or the accuracy of live investigation tools such as *psinfo*. However, an alternative approach is possible to assess the accuracy of acquisitions: the Certainty Scale in Casey (2002a), (see Section 6.2) uses multiple sources of evidence during the analysis stage to test a particular hypothesis about the history of digital data. However, this approach could also be applied to hypotheses

⁴⁷ Logic bombs could also lock the system, revert the data to its encrypt data or crash the system but this would not result in inaccuracy since that event would obviously have happened and would prevent the live investigation from progressing.

about the history of digital evidence during the acquisition stage i.e. whether the collected data has its value due to a manipulated operating system or a faulty imaging tool. To do this, memory could be acquired using multiple tools that use different sources to acquire an image, for example a *dd* based approach and the use of Firewire. Comparing the results could increase the certainty level from C2 (only one source of evidence that is not protected against tampering) to C4 (multiple, independent sources of evidence agree but evidence is not protected against tampering). This has the potential to allow a system to be screened for processes that may make it behave abnormally, therefore establishing if results from running further live tools on the system are likely to contain error due to the operating system misrepresenting its state. This is a promising area for future work, but generalising the assessment of accuracy of digital evidence for general live investigations, rather than those involving the acquisition of encrypted data, is outside the scope of this research.

6.7 CONCLUSIONS

One of the requirements in Chapter 3 was explained as *it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that amount of error should be acceptable in the context of the current investigation*. In this chapter, the error associated with digital evidence has been defined as *the difference between the inferred history and the true history of the examined digital evidence*, where possible error or uncertainty is expressed as *alternative events that explain the current state of the examined digital objects*.

This definition and expression of error has the advantage that it can be used to assess error in both the analysis and acquisition stages of a live digital investigation. This chapter has focused on the error in the acquisition stage of a digital investigation, since error in the analysis stage is concerned with interpretation of the acquired data and can be assessed by using multiple techniques that can be repeated by multiple examiners. If there are any differences in interpretation, the alternative hypotheses for the existence or state of digital evidence artefacts can be compared and the most probable decided on. It is then up to those making the decision to consider whether

the uncertainty is sufficiently small in the context of the current investigation to come to a decision.

The accuracy of the acquisition stage of a traditional digital investigation can also be assessed using repeatability since the original evidence is still accessible and the acquisition can be repeated in trusted environments using multiple tools and the results shown to be the same, which eliminates uncertainty about the acquisition methods or operating system functioning correctly. However, in a live acquisition, possible error, or uncertainty can be introduced either due to live acquisition tools operating incorrectly or by the operating system providing the acquisition tool with data that is not consistent with that on the system. This introduces a number of alternative hypotheses that explain the examined digital evidence having its current state. This uncertainty cannot normally be addressed for live investigations.

However, this chapter has shown that in the context of live digital investigations involving encryption it is possible to assess accuracy of live acquired copies of encrypted data. This is possible if at the same time as the live acquisition takes place, information is recovered from the live system that allows encrypted data to be decrypted offline in a trusted environment. This allows the accuracy of the acquisition to be assessed since offline decryption and offline acquisition are repeatable techniques, the output from which can then be used to validate the live acquired image. In this case, the only information used where the accuracy cannot be assessed through repeatability is data from the live disk that changed between being acquired and the system being powered off, and that which allows decryption of the encrypted data. However, the information that allows decryption is known to be correct since if it were not, then the encrypted data would not decrypt successfully.

This chapter has demonstrated recovery of this information from live systems and offline decryption in two ways. First, the built in graphical interface was used to export keys, which were then used to decrypt data offline. This was demonstrated using *BitLocker* on *Windows Vista*. The second approach involved obtaining a memory dump at the same time as the live acquisition of the mounted encrypted data. From this memory dump, decryption keys were extracted that allowed the offline

decryption of the encrypted data. The reason this is possible, and why it can be generalised, is that on-the-fly decryption is performed on a system as data is required, and the same key is used for all data. This means that the key needs to be stored somewhere for the encryption system to operate, and it can therefore be recovered.

Therefore, in summary, for live digital investigations involving encryption, the accuracy of the live acquisition of encrypted data can be assessed by recovering information that allows it to be also decrypted offline. This offline decrypted copy can then be acquired multiple times using multiple tools and compared to the live acquired data, eliminating the uncertainty that the operating system or live acquisition tool was behaving abnormally.

CHAPTER 7: AUTHENTICITY

7.1 INTRODUCTION

This chapter examines the authenticity requirement for digital evidence. This requirement specifies that *the origin of digital evidence should be provable*. The origin of digital evidence was explained in Chapter 3 to include demonstrating that it comes from a particular piece of physical evidence, since all digital evidence is an abstraction of something physical and ultimately the physical evidence can be connected to a real person. Also, authenticity is concerned with demonstrating what processes that have been used to obtain data from the physical evidence and then translate it through various layers of abstraction into a form that can be interpreted. The requirement also stipulated that accusations of tampering should be easily refutable.

These requirements are satisfied in a traditional investigation since the original physical evidence from where the digital evidence was obtained is still available and acquired evidence can be compared to the original. Also, records of all processes applied can be shown to be correct as they can all be repeated, and the same final result obtained. For a live investigation this is not the case since the original evidence may not be accessible, either because it was erased on removal of the power, e.g. memory or has reverted to a state that means it cannot be accessed e.g. encrypted data.

The chapter shows that two aspects of this requirement can be satisfied for live investigations using existing techniques: digital evidence can be shown to be produced by running particular processes using digital evidence bags; and that accusations of tampering of acquired evidence can be refuted using cryptographic hashes. The chapter therefore focuses on developing a method to demonstrate that digital evidence was obtained from a particular piece of physical evidence, which assists in allowing the recovered digital evidence to be traced to a person.

7.2 BACKGROUND

7.2.1 Authenticity in Traditional Digital Investigations

As described in Chapter 3, the authenticity requirement for digital evidence specifies that *the origin of digital evidence should be provable, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.*

Revisiting the traditional digital investigation process described in Chapter 2, after seizing the physical evidence, usually a full disk image is created of the contents of the seized computer's hard drive and this image is then analysed. The image that is analysed can be shown to be the same as the data on the seized physical evidence by computing a cryptographic hash of both, e.g. using MD5 or SHA1. This demonstrates that the acquired disk image, and therefore digital evidence extracted from it, originated from the seized physical evidence. Since the disk image can be shown to be the same as the original physical evidence, this prevents accusations of tampering with the disk image, since if the image is altered in any way, the hashes of the data stored on the physical evidence and on the disk image would not match. Accusations of tampering with the original physical evidence prior to imaging are countered using the principle of 'continuity of evidence', where it is documented who has had access to the physical evidence and at what stage. This is able to "provide continuity and assure provenance of the item from the time the item was seized to the time the item is used as evidence in court" (Turner, 2005). Demonstrating that digital evidence was obtained from the disk image by running particular processes is also achieved by thoroughly documenting actions performed. During the analysis, actions performed are recorded in detail, meaning that a third party could repeat those actions on the disk image and achieve the same results. This demonstrates that the final digital evidence artefacts obtained were as a result of running particular processes. This process of demonstrating authenticity in traditional digital investigations is shown diagrammatically in Figure 37.

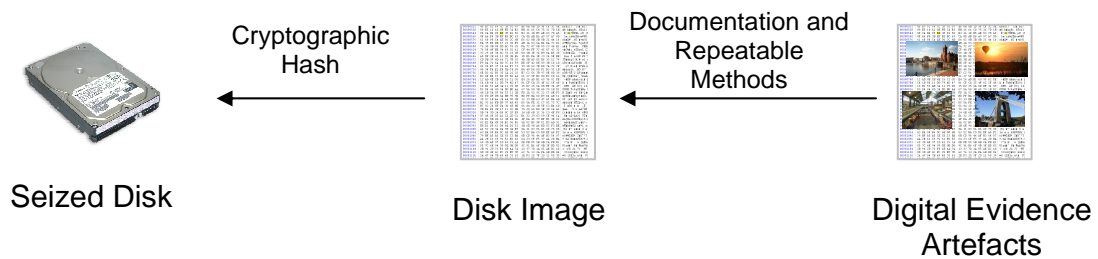


Figure 37: Tracing digital evidence artefacts back to a piece of physical evidence in a traditional digital investigation.

In a live investigation, if the acquisition and analysis stages are separated, documentation and repeatability can also be used to demonstrate that the obtained digital evidence artefacts are obtained as a result of running particular processes on the disk image. Therefore, the problem of demonstrating authenticity of digital evidence in a live digital investigation lies in the acquisition stage. This is because data acquired during a live investigation may not be accessible on the original physical evidence after the power is removed. Therefore cryptographic hashes cannot be used to demonstrate that the acquired evidence came from a particular piece of physical evidence because there is no original data to hash (memory) or it has reverted to a different state (encrypted data). This is shown diagrammatically in Figure 38.

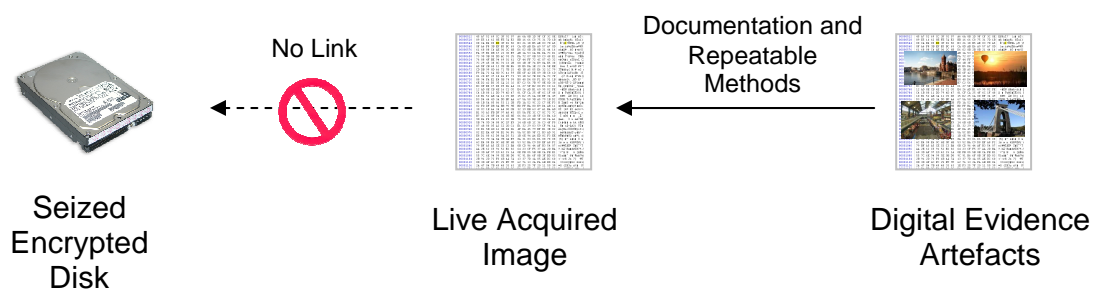


Figure 38: Tracing digital evidence artefacts back to a piece of physical evidence in a live investigation.

7.2.2 Authenticity in Live Digital Investigations

It can be shown that data is produced as a result of running particular processes by documenting steps performed and by running known tools from read only media such as CD-ROMs. Combinations of tools are often combined into toolkits which execute a set of tools in a particular order and record a log of all actions performed. Toolkits such as these can be found on the *Helix Live CD*.

Also, these logs can be made tamperproof if hashes are calculated at the time of creation. Turner (2005) describes a format for Digital Evidence Bags (DEB), which allows information to be attached to acquired evidence in the form of a 'tag file'. The information is protected from tampering by incorporating a 'tag seal number' which is a hash of the current information in the tag.

Digital Evidence Bags can be used not only to attach information such as the name of the investigator and the date and time of capture to the acquired evidence, but also to allow 'real-time evidence capture' where command line instructions supplied to a live machine can be captured directly into Digital Evidence Bags along with the time, the name and hashes of the commands run and hashes of the output from the tools (Turner, 2007). This provides a means to demonstrate that the data acquired is as a result of running particular processes.

Since tampering accusations from the point of acquisition can be refuted by creating hashes of acquired evidence, and the processes run on the live system can be recorded by capturing command line instructions using a format such as Digital Evidence Bags, the remaining difficulty in demonstrating authenticity of live acquired evidence is showing that it came from a particular piece of physical hardware.

Some acquired data has embedded information that could be used to identify the physical origin, for example BIOS information such as make, model and serial number of the computer system can be obtained from memory dumps (Mcquown, 2008). However, live acquisitions may not necessarily be an acquisition of memory and such identifying information may not always be found in acquired data, e.g. an acquired mounted encrypted container will not contain such information.

It is possible to demonstrate the authenticity of live acquired containers using key recovery and offline decryption techniques described in Chapter 6, since the original evidence becomes accessible and the original physical origin can be demonstrated in the same manner as traditional digital investigations. However, in cases where the key patterns have not yet been identified, there is another approach that can be used to demonstrate the physical origin of live acquired digital evidence.

7.3 METHODOLOGY

7.3.1. Overall Technique

This research demonstrates that live acquired data can be shown to come from a particular piece of physical evidence even though the original data on the physical evidence is not longer available or accessible. The overall approach is to modify an acquisition process to also obtain unique system identifiers at the time of acquisition that will still be accessible after the power is removed from the system. At the time of the acquisition, the live acquired evidence can be cryptographically hashed with these identifiers. After the seizure, during the analysis stage, the live acquired image can be re-hashed with the seized physical evidence and the hashes shown to be the same as those produced from the live acquisition. This demonstrates that the live image was acquired on the seized physical evidence, which is shown diagrammatically in Figure 39.

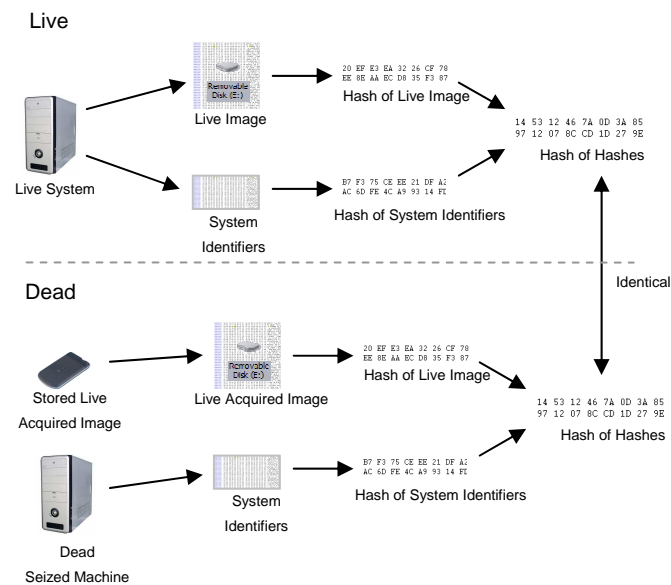


Figure 39: The overall methodology for demonstrating the origin of a piece of live acquired digital evidence.

7.3.2 Choice of Process for Proof of Concept

The technique described in the previous subsection is demonstrated in this research using a prototype, proof of concept implementation. This is achieved by modifying a live acquisition tool to include the hashes of system identifiers. There are a number of tools that could be used, for example tools for the acquisition of physical memory or for the acquisition of mounted encrypted containers. Both are of interest since, as described earlier it is possible to obtain a great deal of information from acquired memory images and are increasingly likely to be performed. Therefore, memory dumps would be a useful proof of concept tool. However, in the context of this research, the live acquisition of mounted encrypted evidence is particularly of interest. For the proof of concept development, *dd* from the *Helix Live CD* is used as an example since it can be used to acquire both physical memory and mounted encrypted containers.

7.4 DEVELOPMENT OF PROOF OF CONCEPT TOOL

7.4.1 Introduction

This section describes the development of a prototype tool that combines *dd* from the *Helix Live CD* with system identifiers in order to demonstrate the origin of live acquired data.

7.4.2 Overview: Two-Stage Methodology

The process of demonstrating the origin of live acquired data has two stages: live and dead. First, during the live investigation, a process is run that launches the acquisition tool *dd*, obtains some system identifiers, and hashes the acquired image with these identifiers. Then, later in an offline environment, the same system identifiers are obtained from the seized physical evidence. Offline, the live acquired image is then hashed with the system identifiers recovered from the powered off physical hardware. This latter process can be repeated at any time and the live acquired data can therefore be shown to have originated from the seized physical evidence.

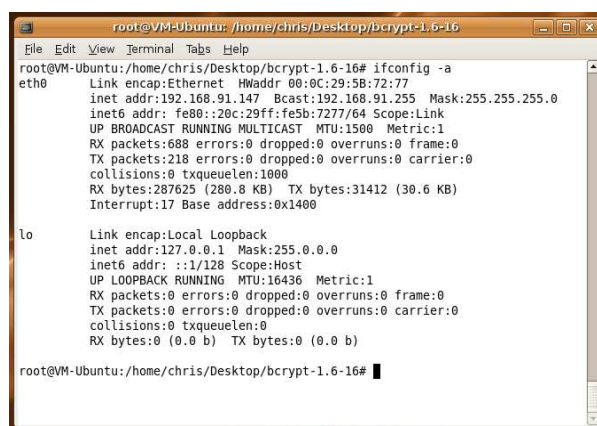
Due to this design, the system identifiers need to be accessible even after the power is removed, and even in cases where Full Disk Encryption is in use. Since in this case the contents of the drive will not be accessible after the power is removed, the use of software identifiers such as the *Windows Product Key* is not possible. It would be possible to generate a hash of some of the encrypted data that will be available after the power is removed and use this as a unique identifier. However, as described in Hargreaves and Chivers (2008a), in some cases it is difficult to obtain encrypted data from a live machine since the encryption software often transparently decrypts it. It is therefore desirable to use hardware identifiers, since these will be consistent when the machine is live and running the suspect's operating system, and when the system is accessed offline during the later analysis stage of the investigation. There are a number of additional requirements for the system identifiers that are used: they should be unique to the system and difficult or impossible to tamper with.

7.4.3 Choice of Hardware Identifiers

This section describes possible hardware identifiers that can be used to identify a machine.

MAC Address

One option is the Media Access Control (MAC) address of the computer. This is the physical hardware address attached to Network Interface Cards (NIC). The MAC address of the network card can be easily accessed on a live suspect *Windows* machine using `ipconfig /all`, and post-seizure, in a controlled environment⁴⁸ using `ifconfig -a`, see Figure 40. Also, the BIOS of some machines can report the MAC address of built in network cards.



```
root@VM-Ubuntu: /home/chris/Desktop/bcrypt-1.6-16# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0c:29:58:72:77
          inet addr:192.168.91.147  Bcast:192.168.91.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe5b:7277/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:688 errors:0 dropped:0 overruns:0 frame:0
          TX packets:218 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:287625 (280.8 KB)  TX bytes:31412 (30.6 KB)
          Interrupt:17 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

root@VM-Ubuntu: /home/chris/Desktop/bcrypt-1.6-16#
```

Figure 40: Displaying the MAC address under *Linux*.

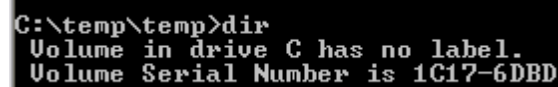
However, the MAC address of network cards can be changed under both *Windows* (Gorlani, 2008) and *Linux*. The changes to MAC addresses are not permanent and are specific to the running operating system. If the suspect is using a modified MAC address, this would have the consequence that during the live acquisition, the modified MAC address would be obtained, but during the offline analysis in a controlled environment, the true (but different) MAC address would be obtained.

⁴⁸ Controlled environment refers to booting the machine using a *Linux* CD such as *Helix*

Hard disk identifiers

A hard disk has more than one identifier, the Volume Serial Number, and the Manufacturer's Hard Drive Serial Number, which are discussed below.

Volume Serial Number: This serial number is a 32 bit number assigned to a partition. It is created when the partition is formatted and is derived from the time of the format (Wilson, 2005). Therefore, a drive with more than one partition would have more than one Volume Serial Number. This value is displayed when running a simple *dir* command, as shown in Figure 41.



```
C:\temp\temp>dir
Volume in drive C has no label.
Volume Serial Number is 1C17-6DBD
```

Figure 41: Volume Serial Number displayed with 'dir'.

Therefore, the Volume Serial Number can be easily retrieved from a live system. However, if the drive or volume has been encrypted, the Volume Serial Number will not be accessible to a later offline analysis, since the Volume Serial Number is stored within the partition itself (at offset 0x43 of sector 0 of the partition).

Also, the Volume Serial Number can be changed using the tool *VolumeID* (Rusinovich, 2006). However, unlike the changing of MAC addresses, these changes would still be present after a reboot and if encryption is not considered, could be used to link live acquired data to a specific machine. However, in this research, encryption does need to be considered.

Manufacturer's Hard Drive Serial Number: This is the serial number of the drive that is set during its manufacture. It can be retrieved using software such as *HD Tune* (EFD Software, 2008) and is often printed on the label of the drive (shown in Figure 42 and Figure 43). Unlike the MAC address, no technique has been identified that can be used to change the manufacturer's serial number.

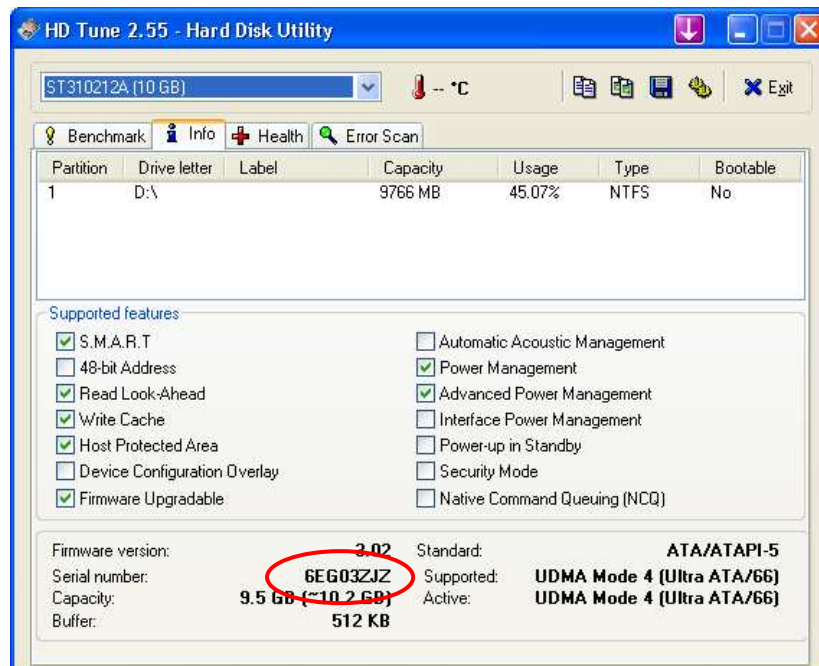


Figure 42: Manufacturer's serial number shown using *HD Tune*.

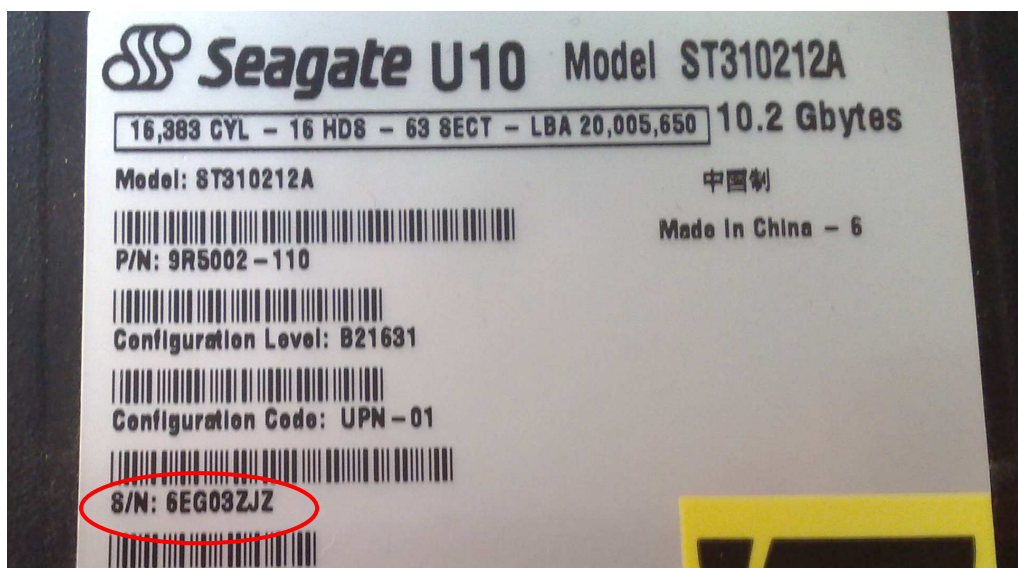


Figure 43: The same serial number obtained from the label of the drive.

Since no technique could be found to alter the manufacturer's hard drive serial number and it can be accessed during a live investigation and during a later offline analysis, this is currently considered to be the best choice to identify a particular

machine. The following two sub-sections describe the implementation of the two stages of the proof of concept tool.

7.4.4 The Live Side

As described earlier, the proof of concept implementation uses *dd* to acquire the contents of an encrypted container and combines the acquisition with the output of a tool that obtains the manufacturer's hard drive serial number. In this implementation, combining this functionality into a single tool is achieved using *Perl*. While *Perl* is an interpreted scripting language that requires software such as *ActivePerl* to be installed on a *Windows* machine to run scripts, it is possible to convert these scripts to self-contained executables using the tool *Perl2Exe* (IndigoSTAR Software, 2008). This was used to produce the executable *acquire_and_authenticate.exe* which makes calls to other software using the 'system' command or the 'backtick'⁴⁹, as shown in Figure 44. The contents of the mounted encrypted volume are acquired using *dd* from the Forensic Acquisition Utilities (Garner, 2007).

```
#-----
#   Get logical drive
#-----
system ("FAU\\dd if=\\\\.\\.$input of=$output conv=noerror");
print("Calculating MD5...\n");
$filehash = trim (`tools\\md5 -n $output`);
print("$filehash\n");
```

Figure 44: Calls to *dd* and *md5* from the *Perl* script.

The manufacturer's hard drive serial number is retrieved on a live machine using a *Visual Basic* script that uses code from Wilson (2006). This code is shown in Figure 45.

```
set svc = getobject ("winmgmts:root\cimv2")
set objEnum = svc.execQuery ("select * from win32_physicalMedia")
```

⁴⁹ 'System' returns the called program's exit code, whereas 'backticks' return the program's output.

```

for each obj in objEnum
    wscript.echo obj.GetObjectText_
next

```

Figure 45: Code that recovers the manufacturer's serial number.

The combined hash is obtained by calling a program that calculates the MD5 hash of input provided to it. First a hash is calculated of the acquired data⁵⁰, and then a hash of the serial number obtained using the earlier script. The output of the tool includes these two hashes and then a single hash of the combined two hashes, as shown in Figure 46. This hash can be documented and written to a text file.

```

Output F:\physical_origin_tool\image.dd (20709376 bytes)
5056+0 records in
5056+0 records out
Calculating MD5...
8AD04E3E1960FE8212527D7A48CC5CF9
14:38:23 6 Jan 2009
HD SERIAL: 6CR02348
-----
FILE HASH: 8AD04E3E1960FE8212527D7A48CC5CF9
HD SERIAL HASH: 703B0872885680EE90B787EA2202F3B4 -
-----
FINAL HASH: 019411AE6C88FED829DE5863F0ED0734 -
F:\physical_origin_tool>

```

Figure 46: Output from the live tool.

7.4.5 The Dead Side

With the combined hash recorded, and the live acquisition performed, the system can be powered off and seized. During the later analysis, the live acquired data can be demonstrated to originate from the seized hardware by recovering the manufacturer's serial number and re-hashing this serial with the live acquired image.

There are a number of options for recovering the manufacturer's serial number for a hard drive. For some drives it is printed on labels placed on the outside of the hard disk itself, and it can therefore be recovered easily. However, this is not the case for all drives and in these cases, it is possible to recover the serial number by booting to a *Linux* based CD (which allows the drive to be mounted as read only to prevent changes) and using: `hdparm -i /dev/sda`, the output of which is shown in Figure 47.

⁵⁰ Since the hash is calculated of the acquired data, not the actual data, there are no difficulties due to the data continuously changing.


```
ubuntu@ubuntu:/home$ sudo hdparm -i /dev/sda

/dev/sda:
Model=ST34313A , FwRev=3.03 , SerialNo=6CR02348
Config={ HardSect NotMFM HdSw>15uSec Fixed DTR>10Mbs RotSpdToL>5% }
RawCHS=8944/15/63, TrkSize=0, SectSize=0, ECCbytes=4
BuffType=unknown, BuffSize=512kB, MaxMultSect=32, MultSect=732?
CurCHS=8944/15/63, CurSects=8452080, LBA=yes, LBAsects=8452080
IORDY=on/off, tPIO={min:240,w/IORDY:120}, tDMA={min:120,rec:120}
PIO modes: pio0 pio1 pio2 pio3 pio4
DMA modes: mdma0 mdma1 mdma2
UDMA modes: udma0 udma1 udma2 udma3 *udma4
AdvancedPM=yes: unknown setting WriteCache=enabled
Drive conforms to: Unspecified: ATA/ATAPI-1,2,3,4,5

* signifies the current active mode

ubuntu@ubuntu:/home$
```

Figure 47: Output from `hdparm -i /dev/sda` to obtain the manufacturer's serial number.

With the serial number obtained, the offline verification of the acquired evidence's origin can be achieved using the developed executable, *offline_authenticate.exe*, which is another *Perl2Exe* converted *Perl* script that takes the path of the acquired image and the text string of the hard disk serial number and calculates the combined hash. If this is identical to the hash obtained during the live acquisition, then the live acquired image can be shown to have come from that piece of physical evidence.

7.5 RESULTS

The prototype tool was tested on a live system running *BestCrypt* (where key recovery from memory is not yet possible). A test system was built⁵¹ and an encrypted container was created using *BestCrypt*.

The *Perl* wrapped version of *dd* was used to acquire the contents of the encrypted container and the hashes were displayed on screen. The hashes were documented and the screen photographed (shown in Figure 48). The system was then powered off.

⁵¹ In this case a real system was built rather than using virtual machines so that the hard drive serial number could be photographed.

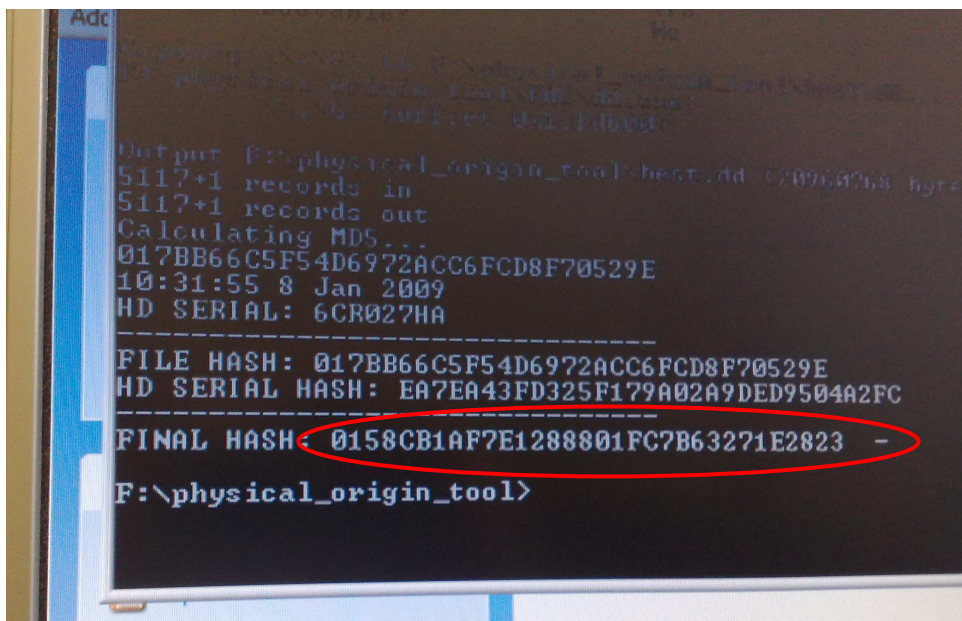


Figure 48: Photograph of the output from the live tool.

For the offline stage, the hard drive was removed from the system and the manufacturer's hard drive serial number recorded from the label (Figure 49). This serial number was entered into the offline authentication software as shown in Figure 50 and the outputted hashes compared to those obtained during the live investigation.

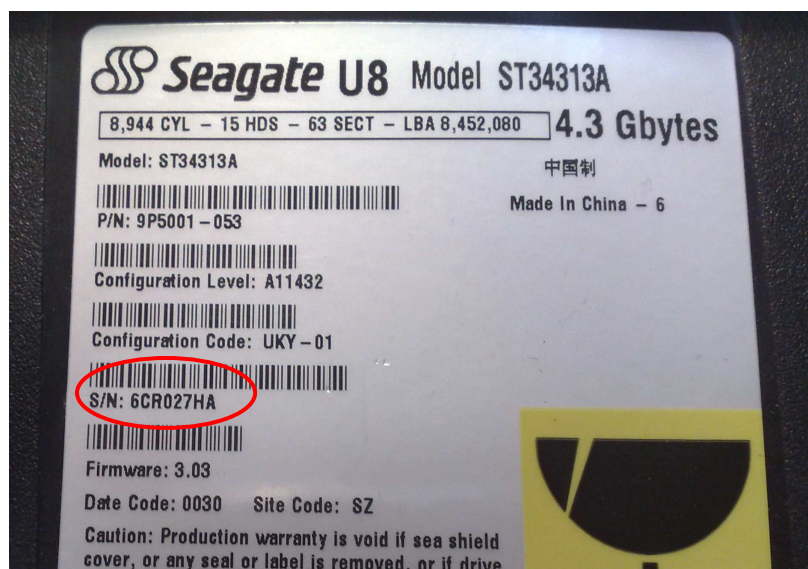
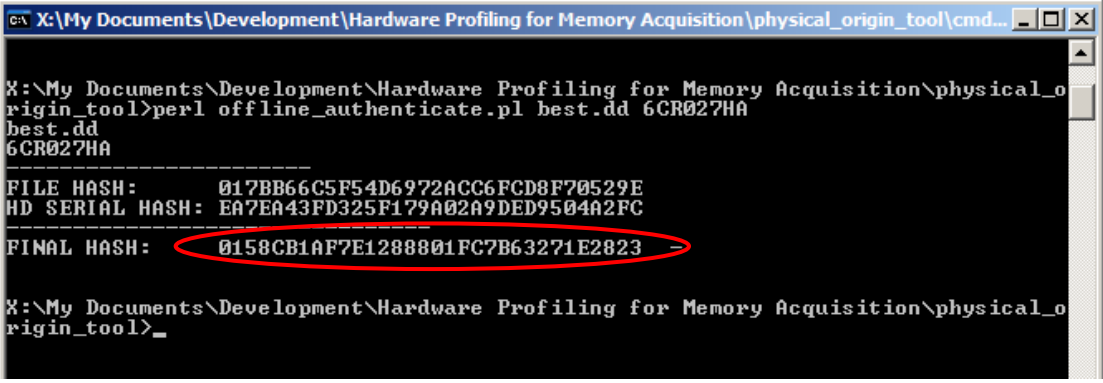


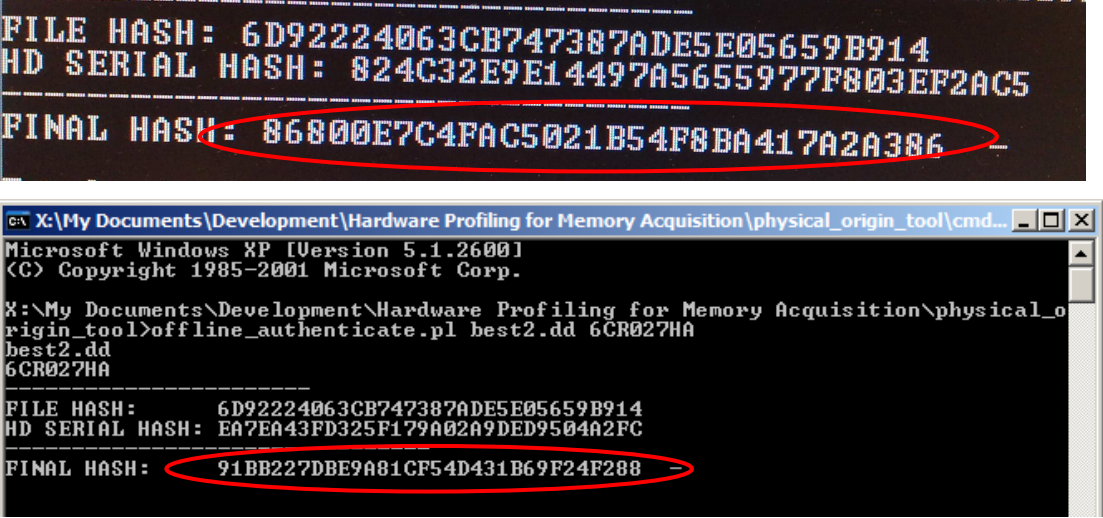
Figure 49: Photograph of the physical drive's serial number.



```
X:\My Documents\Development\Hardware Profiling for Memory Acquisition\physical_origin_tool\cmd...
X:\My Documents\Development\Hardware Profiling for Memory Acquisition\physical_o
rigin_tool>perl offline_authenticate.pl best.dd 6CR027HA
best.dd
6CR027HA
-----
FILE HASH:      017BB66C5F54D6972ACC6FCD8F70529E
HD SERIAL HASH: EA7EA43FD325F179A02A9DED9504A2FC
-----
FINAL HASH:    0158CB1AF7E1288801FC7B63271E2823 -
X:\My Documents\Development\Hardware Profiling for Memory Acquisition\physical_o
rigin_tool>
```

Figure 50: Output of the offline tool which is the same as that produced on the live system (Figure 48), which links the live acquired disk image to the seized hardware.

As can be seen in Figure 48 and Figure 50, the hashes produced at the time of the live investigation by the acquisition tool which were documented and photographed agree with those produced using the seized hardware that is available for repeated inspection in controlled environments. If the image was acquired from a different system to the one seized, the hashes would not match as shown in Figure 51.



```
FILE HASH: 6D92224063CB747387ADE5E05659B914
HD SERIAL HASH: 824C32E9E14497A5655977F803EF2AC5
-----
FINAL HASH: 86800E7C4FAC5021B54F8BA417A2A386 -

X:\My Documents\Development\Hardware Profiling for Memory Acquisition\physical_o
rigin_tool>offline_authenticate.pl best2.dd 6CR027HA
best2.dd
6CR027HA
-----
FILE HASH:      6D92224063CB747387ADE5E05659B914
HD SERIAL HASH: EA7EA43FD325F179A02A9DED9504A2FC
-----
FINAL HASH:    91BB227DBE9A81CF54D431B69F24F288 -
```

Figure 51: Hashes produced do not match if the image was not acquired from the seized system.

This demonstrates that the live acquired image came from the hardware in the possession of the investigator that can be linked to a suspect using traditional means.

7.6 EVALUATION

This developed approach allows live acquired data to be demonstrated to come from a particular piece of physical evidence. This is part of the authenticity requirement, as explained in Chapter 3: *the origin of digital evidence should be provable, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.* In this research only the physical origin has been considered, since running particular processes can be demonstrated using the principles used in Digital Evidence Bags (Turner, 2006, , 2007) and it may also be possible to determine processes run on a system during a live investigation by the changes made to the system by the processes run (see Section 9.3). Also, if the acquisition and analysis stage of digital investigations are separated then accusations of tampering with digital evidence can be refuted since the hash of the analysed data can be shown at all stages to be the same as when it was first acquired.

There are some limitations to this approach. First, administrator privileges are required to obtain the hard drive serial number, and these may not be available. In this case it would be necessary to revert to using the MAC address or other hardware configuration information such as disk sizes to establish a link between the live acquired data and the seized physical evidence.

Also, as described in the previous chapter, the operating system may not provide accurate information to live investigation tools. If the operating system behaves abnormally and provides false information e.g. a false hard drive serial number, when the offline analysis is performed in a controlled environment and the system does behave normally, the hashes will not match and the origin of the live acquired data will be difficult to prove. However, this would involve modifying the operating system to return a modified hard drive manufacturer's serial number, and a means of achieving this has not been found. Even if this is achieved, it is still possible to use multiple identifying factors to counter this. Hashes could be obtained of the volume serial number, the MAC address and even hard disk sizes. While some could be changed, or rendered inaccessible offline by encryption, being able to access and

hash acquired data with several of the multiple factors could increase confidence that the acquired data came from the seized hardware. As a result, future work involves exploring further options available to link live acquired data to physical evidence. It also may be possible to use the real-time capture implementation of Digital Evidence Bags and integrate the hashing of physical evidence identifiers to demonstrate the physical origin of acquired digital evidence into the DEB framework. However, prototype DEB tools are not yet available and this remains future work.

There are also limitations of the particular implementation: the *Perl* proof of concept is clumsy since it makes calls to other software to perform much of the functionality. If the technique is developed into a real tool then obtaining hardware identifiers and computing combined hashes should be integrated into the acquisition tool itself. Also, in this implementation, the hard drive serial numbers can only be obtained for IDE drives using the current script. However, code has now been found to obtain SATA serials (Napalm, 2006) but has not yet been integrated into the developed authentication programs.

Also, referring back to Chapter 5, the changes caused to a system by *dd* were determined to be minimal (single prefetch entry and single Registry change). Future work will involve examining the changes caused by the additional functionality of calculating MD5 hashes and obtaining hardware identifiers. Identifying these changes is necessary for any ‘real’ implementation of this prototype tool.

Also, it is important to emphasise that this ‘physical identifier approach’ is not sufficient on its own to demonstrate authenticity. It is possible to ‘cheat’ the system by using a modified version of the *acquire_and_authenticate.exe* that would take a fake memory dump as input that contains some fabricated evidence. This modified *acquire_and_authenticate.exe* could be run on the suspect’s system instead and would produce a combined hash of the faked evidence with the suspect’s hardware. When this is examined offline, this fake evidence will be authenticated as coming from the suspect’s machine. Therefore, the approach described in this chapter needs to be used in conjunction with a technique such as digital evidence bags which records the name and hash of the process run on the suspect machine. This means that it can be shown

that the evidence was obtained using the ‘standard build’ of `acquire_and_authenticate.exe`, which is known to acquire data from the machine on which it is running; not to take any custom input and generate a combined hash. The question still remains of how to demonstrate that the Digital Evidence Bag software has not been tampered with, but using procedural measures such as running from a read only medium such as a CDROM, multiple investigators signing documents to certify that certain software was run, or even videoing the procedure live, it is possible to demonstrate the authenticity of live acquired data.

Finally, while this ‘hardware hashing’ approach is useful in the context of this research (acquiring live encrypted data), and can be extended to apply to memory acquisitions and other live acquisitions saved to removable storage media, it is not yet known how this approach could be applied to demonstrate the physical origin of other types of digital evidence, for example packet capture on a network. Nor can it be applied to a memory image acquired over a Firewire connection since the physical identifiers of the source system cannot be obtained in the same way. However, live acquisitions of disk or memory to a USB storage device or other removable media represents a significant proportion of live acquisitions and this is therefore a useful technique. Also, as described earlier, BIOS information and other identifying material may be used to determine the origin of acquired memory dumps, including in the case of Firewire acquisitions. Nevertheless, demonstrating physical origins of other types of digital evidence remains future work.

7.7 CONCLUSIONS

The authenticity requirement means that *the origin of digital evidence should be provable, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.* It has been shown that two aspects of the authenticity requirement can be satisfied using existing techniques. Specifically that digital evidence can be shown to be obtained as a result of running particular processes using Digital Evidence Bags, which can create a tamperproof record of

processes run and their output. Also, accusations of tampering with the acquired evidence can be refuted if cryptographic hashes are created of the acquired evidence, which can be checked throughout the life of a piece of digital evidence.

However, demonstrating the physical origin of live acquired evidence is difficult, particularly in the case of encrypted data, since the original data is not accessible once the power is removed. This chapter has shown that by hashing live acquired evidence with some unique physical property of the computer system, in this case the manufacturer's hard drive serial number, the physical origin can be demonstrated, even without access to the original data in unencrypted form.

CHAPTER 8: EVALUATION

8.1 INTRODUCTION

This chapter reviews the research performed, and evaluates it against the original research hypothesis of *digital evidence obtained from live investigations involving encryption can be shown to be reliable*. Chapters 4 to 7 have examined each of the requirements used to assess the reliability of digital evidence and each chapter has evaluated the extent to which the requirement can be satisfied. This chapter provides a summary of the conclusions of the previous chapters, and evaluates them against the original research hypothesis. The chapter first considers the overall methodology used to test the proposed hypotheses, followed by evaluations of the methodologies used to examine each of the requirements that are used to assess the reliability of digital evidence, and also to determine if they support the original hypothesis.

8.2 METHODOLOGY EVALUATION

Digital evidence was defined in Chapter 2 as *a set of reliable digital objects that support or refute a hypothesis*. Therefore, this research was concerned with determining whether digital objects recovered using live techniques from systems using encryption could be considered to be reliable, and therefore used as digital evidence. The research hypothesis was that *digital evidence obtained from live investigations involving encryption can be shown to be reliable*. In order to test this hypothesis a measure of reliability was needed. As discussed in Chapter 2, digital evidence is used for digital investigations and forensic digital investigations, and each demand different levels of reliability of digital evidence. A higher degree of digital evidence reliability is needed to convict someone in a criminal court than in a corporate environment to come to a decision about a violation of an acceptable use policy, where the consequences of the decision are very different. Even within a forensic digital investigation, there are differences between the standard of evidence necessary for a decision in civil and criminal cases, where *balance of probabilities*

and *beyond all reasonable doubt* are used respectively. Therefore, whether digital evidence is convincing enough to come to a decision is dependant on the decision to be made and the person making the decision. As shown in Chapter 1, the reliability of digital evidence is therefore context sensitive and subjective. This presented a problem for this research since adopting a subjective view of digital evidence reliability means that this hypothesis could not be tested. However, in Chapter 3 it was shown that there is an alternative approach, where standards or requirements can be used to “ensure quality” and to “guarantee to those not involved of reliable results” (Pollitt, 1995). It was shown in Chapter 3 that there are number of existing standards or requirements that are currently used to assess the reliability of digital evidence, e.g. the ACPO guidelines, and since reliability is already assessed in this way, it was assumed that reliability of digital evidence can be assessed using a set of standards or requirements. Once this was established, it was then necessary to identify appropriate requirements.

8.3 REQUIREMENTS EVALUATION

Based on the assumption that the reliability of digital evidence could be assessed using standards or requirements, it was necessary to identify appropriate requirements. Existing requirements were examined but they were found to have limitations. Many were produced by those involved in *forensic* digital investigations and as a result they contained legal specific terminology, for example “chain of custody” (Sommer, 1998). This was not appropriate in this research since general requirements for digital evidence were needed, since digital evidence is used for forensic digital investigations and also general digital investigations, where the results do not need to be presented in court. Also, some of the existing requirements were written before live investigations became necessary and as a result the requirements were based around an approach where only the disk is considered to contain evidence. The consequence of building requirements on the assumption of evidence being only on disk, is that requirements such as “evidence should not be altered” (Pollitt, 1995) can then be used, since discarding memory by disconnecting the power is not

considered to be altering evidence. Once it is accepted that memory can contain relevant digital evidence, requirements such as this become impossible to satisfy.

As a result, existing requirements were not considered appropriate and it was necessary to produce technology neutral and general requirements for assessing the reliability of digital evidence. As a basis for this, a set of requirements for assessing the reliability of machine generated evidence was found in Miller (1992) that had the potential to be applied to assessing the reliability of digital evidence. These requirements were adapted to apply to digital evidence by considering the processes that make up a digital investigation to be equivalent to the machines in Miller (1992). From these, a set of general requirements for assessing the reliability of digital evidence were proposed, which were:

Authenticity: it should be possible to demonstrate the origin of digital evidence, both in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable;

Accuracy: it should be possible to assess the amount of error associated with all techniques used to obtain digital evidence, and that amount of error should be acceptable in the context of the current investigation;

Completeness: it should be possible to assess which evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved.

However, the hypothesis that these are *the* requirements for assessing the reliability of digital evidence has the limitation that it can never be demonstrated conclusively that it is correct, only that a counterexample has not yet been found. Therefore, to determine if the requirements could be shown to be incorrect, existing requirements and standards were examined and compared to those proposed, since if the proposed

requirements were incorrect, then there would be significant inconsistencies with existing requirements. Requirements were selected to be examined that were in current use, taken from peer reviewed literature, or produced by experts in the field. These were discussed in Chapter 3 and found either to agree with the proposed general requirements, or it was shown how they were specific means of satisfying the proposed requirements e.g. “chain of custody” (Sommer, 1998) is shown as a way of demonstrating authenticity. Other requirements such as “evidence should not be altered” (Pollitt, 1995) were shown to be inappropriate for reasons discussed earlier. Also, the requirements that were specific to law, for example, only seizing evidence allowed by law (Mocas, 2004), were shown not to apply to all digital investigations and therefore inappropriate for general requirements for assessing the reliability of digital evidence.

From this examination of existing requirements it was not possible to find a valid counterexample, thus supporting the hypothesis that these requirements were suitable as general requirements for assessing the reliability of digital evidence. However, one of the challenges of proposing general requirements for assessing the reliability of digital evidence is that there is an extremely broad range of sources of digital evidence, including PCs, network devices, mobile phones etc. Therefore, it is possible that counterexamples may be found in other specific sub-categories of digital investigations. However, given the sample of existing requirements to which these proposed requirements were compared and shown to be compatible with, it is believed that any changes that are found to be necessary over time to these requirements, due to their generalised nature, will be minor adjustments to phrasing and explanation of the requirements.

Therefore, since contradictory examples to assessing reliability of digital evidence using these three requirements were not found, in the remainder of the research, the reliability of digital evidence obtained from live investigations involving encryption was assessed against the three criteria of completeness, accuracy and authenticity.

8.4 COMPLETENESS EVALUATION

The completeness requirement stated that *it should be possible to assess which digital evidence is preserved and which is lost, and the maximum amount of digital evidence relevant to the investigation should be preserved*. This requirement was examined over two chapters, Chapters 4 and 5. The requirement was first examined from the perspective of whether a live digital investigation or a traditional digital investigation should be performed to preserve the maximum amount of relevant digital evidence. In order to assess relatively what was preserved and lost by performing a particular type of investigation, it was considered whether offline access to encrypted data was likely using the approaches described in Chapter 2. Several of these techniques for gaining access to encrypted evidence were case specific, e.g. persuade the suspect to provide decryption keys, and therefore it was not possible to generalise about whether offline access to encrypted evidence would be possible for certain types of product, and therefore whether a live investigation would increase the completeness of the preserved evidence. Also, these case specific approaches make assumptions, e.g. that the suspect would co-operate and provide keys. These factors are difficult to predict and therefore were not of use in determining whether a live investigation should be performed. There were also approaches that were product implementation specific, e.g. find a vulnerability in an algorithm in use. The likely success of these specific approaches also could not be generalised and as a result were not considered in this research. The limitation of not considering product specific approaches is that offline access to encrypted evidence may be possible for particular implementations. Therefore, without taking product specific approaches into account, a live investigation could be performed when it was not necessary. However, in order to address this problem, a database of encryption products, including any vulnerabilities that allow offline access, would need to be developed. This would require a team of researchers to maintain, including keeping it up-to-date and controlling access. Therefore, this was not considered to be a feasible approach in this research.

However, the remaining offline approaches were explored as to whether general conclusions could be drawn about the likelihood of successfully gaining

access to encrypted evidence. It was found that the success of three of the approaches was affected by the amount of the disk that remains accessible after the power is removed: locating copies of data in unencrypted form; locating copies of the password or key; and intelligent password attacks. Encryption products were therefore categorised based on this property (the amount of the disk that remains accessible) and the categories identified were:

Manual file encryption	A user selects a single file for encryption.
Folder encryption	All files contained within a particular folder are automatically encrypted.
Virtual drive encryption	A virtual drive is created which is stored as a single file on the user's file system and all data stored to this virtual drive is automatically encrypted.
Full Volume Encryption ⁵²	An entire partition is encrypted, but other partitions and the partition structure are accessible.
Full Disk Encryption	The entire disk is encrypted, including the partition table.

By examining which locations were left on the disk after a traditional digital investigation approach was used ('pull the plug'), for each category, the effect on the investigation in terms of the completeness of the digital evidence preserved could be determined, with some categories preserving less digital evidence than others. The areas of the disk rendered inaccessible by pulling the plug may or may not contain relevant digital evidence. However, in this research it was assumed that the encrypted evidence was relevant and therefore access was needed to it. This is a valid assumption because even if the encrypted content was not relevant to the investigation, access would be needed to it in order to determine this. As a result, if

⁵² This category was identified during the research. The initial categories used were based on WinMagic (2005) which proposed four original categories, which did not include a distinction between Full Volume Encryption and Full Disk Encryption.

data could not be accessed because it was encrypted then completeness was assumed to have decreased.

In addition to which locations were encrypted and therefore inaccessible after the power is removed, it was also considered how useful the locations remaining would be in assisting with offline approaches to gaining access. The different product categories were examined and some left areas of the disk accessible that could contain information that would be useful in obtaining access to encrypted data on the disk during an offline examination, e.g. C:\temp, or the *Windows Registry*. However, other factors also affect the likelihood of gaining access using these approaches. These include the suspect's technical ability, the complexity of their password, and their understanding of the precautions necessary when using encryption, e.g. erasing temporary files. It was therefore found that if a system is encountered in a live state and encrypted data is accessible, given the variables involved in attempting to predict whether offline access will be possible, the most effective method to obtain encrypted data in a form that is accessible is to perform a live acquisition.

However, there are more complexities to the completeness requirement once the decision has been made to embark upon a live investigation. These complexities were explored in Chapter 5. This was necessary since any live investigation is inherently intrusive and all actions performed on a live system will make changes to the system under investigation, which will overwrite data and decrease the completeness of the preserved digital evidence. Chapter 5 examined how these changes caused by live techniques could be assessed. Understanding changes caused by live tools and techniques makes it possible to predict which particular live technique should be used in an investigation to attempt to maximise the preservation of potentially relevant digital evidence. Since different tools overwrite different evidence, this means that depending on what needs to be preserved for the current investigation, techniques can be chosen and used that overwrite only data that is known to be irrelevant. Due to the diversity of system configurations, predictions may not be exactly correct and therefore being able to assess changes post-live

investigation is also necessary in order to determine which data has been overwritten and on the system in question after the live investigation has been performed.

The approach used to assess the evidence preserved and lost was to develop a methodology to monitor a test system and record the changes made to it. This monitoring methodology was used to record the changes made by running programs (including live investigation tools) and also connecting to the system in a variety of ways. However, the limitation of this approach is that these are records of changes made to simple test systems only, and further work is necessary to extend these predictions, and consider changes that are made to systems in configurations that are encountered in real investigations. However, the developed techniques and methodology will assist in performing this future work.

Therefore, completeness can be increased and decreased by performing a live investigation. If encrypted evidence is encountered on a live system, it is difficult to predict if offline approaches for gaining access to this encrypted digital evidence will be successful, whereas performing a live acquisition can preserve this information in a form that can be analysed. However, performing a live investigation will overwrite data on the system and decrease completeness. The assessment of this decrease in completeness caused by performing a live investigation can be achieved by testing tools in advance using the developed methodology. This allows changes to be predicted and therefore the best course of action decided upon for the current investigation, which attempts to minimise the loss of relevant digital evidence. It also assists in demonstrating post-live investigation, what changes were actually made and what evidence was lost.

8.5 ACCURACY EVALUATION

Chapter 6 examined the accuracy requirement for digital evidence, which was explained as *it should be possible to assess the amount of error associated with all techniques used to obtain and process digital evidence, and that error should be acceptable in the context of the current investigation*. However, this explanation has a limitation, since 'error' in the context of digital investigations was not defined.

Therefore, the concept of error in digital investigations was reviewed in Chapter 6 and then defined as *the difference between the inferred history and true history of the examined digital evidence*, where error is expressed as *alternative events that explain the current state of the examined digital objects*. This definition can be used to assess error in the acquisition and analysis stages of a digital investigation. However, this research did not consider error in the analysis stage of live investigations, since if the acquisition and analysis stages are separated; once live data is acquired, the remainder of the investigation process is no different to data acquired in a traditional investigation. This means that once data is acquired from a live system to a storage medium, it has the same properties as digital evidence from a traditional digital investigation, in that it can be exactly duplicated and examined multiple times by multiple examiners. Therefore, this research has concentrated on assessing the accuracy of the acquisition stage of a live investigation involving encrypted evidence.

This research showed that it is possible to assess the accuracy of live acquired data from systems using encryption, if, at the time of seizure, in addition to the mounted encrypted data, other information is acquired that allows offline decryption of the encrypted data. This was demonstrated by obtaining recovery keys from *BitLocker* in *Windows Vista*, and also recovering decryption keys from a memory dump of *TrueCrypt*. Both of these techniques allowed the encrypted data to be decrypted offline in a repeatable manner in a trusted environment. This offline acquired copy was then used to demonstrate the accuracy of the live acquired data since it eliminates alternative hypotheses that examined digital objects (the acquired data) have their values due to the operating system misrepresenting its state or the acquisition tool being faulty. The only digital object that is used as evidence, whose accuracy cannot be demonstrated in this way, is the data that allows offline decryption, and the accuracy of this is proven by its ability to decrypt the data.

There are however, two main limitations to this approach (excluding limitations of specific offline decryption approaches, e.g. *TrueCrypt* key recovery only being implemented for certain algorithms). First, is that it does not address the logic bomb problem, i.e. a piece of software could be installed so that when a certain

action is performed e.g. an investigator plugs in a USB stick, this causes data to be erased or manipulated, the system to be locked or the system crashed so that data reverts to its encrypted state. Only the first of these is an accuracy problem since it changes the history of the examined digital objects by changing their state, whereas the others prevent the data from being acquired at all. The second limitation is that the accuracy of a live acquisition has not been assessed using only live techniques, and an offline decryption and acquisition was necessary to allow accuracy to be assessed using repeatable methods. This is possible only for live acquired *encrypted* evidence, since after ‘pulling the plug’ the original data is inaccessible rather than permanently erased. So while the accuracy of digital evidence from live investigations involving encryption can be assessed by comparing it to the offline decrypted data, this approach can not be generalised to other live investigations, e.g. acquisition of memory or the output of live tools such as *pslist*.

However, a general approach to assessing accuracy may be possible using an alternative method. As described in the conclusions section of Chapter 6, the Certainty Scale in Casey (2002a), which can be used during the analysis stage of an investigation to compare multiple sources of evidence to test and describe confidence in a particular hypothesis, could also be applied to live acquisitions. In this case multiple live acquisition tools that acquire data using different sources could be used and the results compared e.g. a Firewire acquisition and a *dd* based acquisition. If mapped to the Certainty Scale in Casey (2002a), by using multiple tools, this would increase the certainty in the acquired data since data could be verified by multiple sources. Also, if the accuracy of a memory image can be assessed, it is then possible to begin to determine whether the system is ‘behaving normally’ i.e. there are no suspicious processes whose function cannot be explained. This would allow an investigator to search for traces of logic bombs, determine if the system is behaving normally for the purposes of determining changes caused by live tools, and also screen a system for processes that may affect the results of other live tools later run on the system. However, this remains future work, but highlights the importance of memory acquisitions in demonstrating accuracy of results obtained from live systems.

Therefore, if live acquisition is separated from analysis and presentation, the error associated with processes that analyse and present live acquired evidence can be assessed in the same way as in a traditional investigation. Also, the accuracy of the live acquisition of encrypted evidence can be demonstrated if information is recovered from the live system that allows offline decryption. This allows the accuracy of the live acquired copy to be verified by comparing it to the offline acquired copy, which was obtained in a trusted environment where the process and can be repeated, which therefore eliminates alternative hypotheses of the acquisition tool being faulty or the operating system providing false information to the acquisition tool. However, limitations remain, since there is possible error due to logic bombs erasing relevant data, which may or may not be significant depending on the individual investigation. In addition, this could be addressed with future work.

8.6 AUTHENTICITY EVALUATION

The authenticity of digital evidence from live investigations involving encryption was examined in Chapter 7. This requirement stated that *it should be possible to demonstrate the origin of digital evidence, in terms of coming from a particular piece of physical evidence and also being produced by running particular processes. In addition, accusations of tampering should be easily refutable.* Of the three aspects of the authenticity requirement, two could already be addressed for live investigations using existing techniques. First it can be demonstrated that data is produced as a result of running particular processes by maintaining a record of processes run and the output captured using a technology such as digital evidence bags. Also, accusations of tampering after acquisition can be refuted by creating hashes of acquired evidence which can be recalculated at any time and the evidence shown to be unchanged. The limitation of this is that accusations of tampering prior to acquisition cannot be refuted using technology. However, this is the case regardless of whether a live investigation is performed, since in a ‘pull the plug’ investigation, evidence could be manipulated prior to the power being removed, and tampering at the scene is possible in real-world forensics. This problem can be addressed procedurally using multi-person teams of

investigators and it would also be possible to video the entire seizure to record all actions performed at the scene. Therefore, accusations of tampering prior to acquisition were not considered in this research.

Since two aspects of the authenticity requirement could already be addressed, this research focused on demonstrating that live acquired data came from a particular piece of physical evidence. This was achieved by modifying the acquisition process to obtain unique physical identifiers of the system (in this case the hard drive's manufacturer's serial number) and to cryptographically hash the acquired data with these identifiers. Since these identifiers are available before and after the power is removed from the system, during the later analysis, even though the original data is not available (e.g. it is encrypted or erased when the power is removed) it is still possible to obtain the physical identifiers from the seized evidence. It is then possible to perform the same hashing operation and show that the live acquired evidence came from the seized piece of hardware, which can be connected to the suspect.

Therefore, authenticity can be demonstrated for live acquisitions using a combination of technological and procedural techniques. It can be demonstrated that data is produced as a result of running particular processes, either procedurally or using a technology such as Digital Evidence Bags, accusations of tampering after acquisition can be refuted by creating hashes of acquired evidence, which can be recalculated at any time and the evidence shown to be unchanged, and the physical origin can be demonstrated by hashing evidence with physical identifiers of the system which can be repeated at any time which demonstrates that the acquired digital evidence came from a particular piece of physical evidence.

8.7 CONCLUSIONS

In summary, the original research hypothesis was *digital evidence obtained from live investigations involving encryption can be shown to be reliable* and this research has proposed that reliability of digital evidence can be assessed in terms of three criteria: authenticity, accuracy and completeness. It is been shown that for a live investigation, authenticity can be satisfied by recording the processes run, either procedurally or

using a technology such as Digital Evidence Bags; that acquired evidence has not been tampered with after acquisition by creating cryptographic hashes of the acquired evidence; and that the acquired evidence came from a particular piece of physical evidence by hashing acquired digital evidence with physical identifiers such as manufacturer's hard drive serial number. It has also shown that the accuracy of digital evidence from live investigations is dependant on demonstrating the accuracy of the acquisition stage. This was shown to be possible by acquiring specific information, in addition to the mounted encrypted data, which later allows the static encrypted data that remains when the power is removed to be decrypted offline in a repeatable manner in a trusted environment and compared to the live acquired copy. This additional information can be in the form of recovery keys or a memory dump, from which decryption keys can be extracted. While this approach cannot be extended to general live acquisitions, in the context of live investigations involving encryption, this technique allows digital evidence to be acquired in a form that can be analysed and the accuracy of that acquisition to be assessed. It has also been shown that live investigations can increase the completeness of the preserved digital evidence, and assuming the encrypted evidence is considered relevant to the investigation, will preserve the maximum amount of digital evidence relevant to the investigation. It has also been shown that it is possible to assess the evidence that is preserved and lost by monitoring live tools and techniques in test environments and recording the changes made. This testing assists an investigator in determining the best course of action during a live investigation using predictions about what will be overwritten by particular live tools and techniques. Also, the results obtained from footprinting live tools and also other software that is found on systems e.g. antivirus, increases investigators' understanding of the changes that occur on a system, which assists with the analysis of the machine post-live investigation, to identify changes made and therefore potential digital evidence that was not preserved. While the test environments examined do not yet truly reflect the real systems on which live tools are run, the developed methodology makes this future work possible.

Therefore, referring back to the original hypothesis of *digital evidence obtained from live investigations involving encryption can be shown to be reliable*, despite the use of these requirements to assess reliability and the success of the implemented solutions to satisfy them, it is important to remember that in reality, the reliability of digital evidence is subjective and context sensitive, as discussed at the beginning of this chapter and in Chapter 3. So while reliability can be assessed against requirements, it is necessary for those requirements to address the context sensitivity of digital evidence reliability. The requirements proposed allow for this, for example, the requirement for accuracy states that it must be possible to assess error, and that this error should be acceptable in the context of the current investigation. Also, the completeness requirement states that the maximum amount of relevant evidence should be preserved, where what is considered relevant digital evidence will change depending on the investigation. Also, when considering authenticity, it is possible for the person collecting evidence to subvert the collection process, by introducing, altering or removing evidence. It is therefore necessary for decisions about these factors to be made for individual investigations: whether the error is acceptable, whether something relevant was not preserved, or whether the person who collected the evidence performed the evidence collection properly. This is the responsibility of those making the decision, which will in turn depend on the decision to be made. Therefore, it is not possible to broadly say whether digital evidence obtained from live investigations involving encryption is reliable, because it depends on the circumstances in which it is used. However, this research has provided structured criteria that allow this reliability to be assessed and has also demonstrated the use of these criteria in the context of live investigations involving encryption and shown the extent to which each can currently be met if the most reliable evidence possible is aspired to.

CHAPTER 9: CONCLUSIONS

9.1 CONCLUSIONS

This final chapter provides a summary of the contributions of this thesis, discusses future work and provides a summary of the work performed and the conclusions drawn.

9.2 CONTRIBUTIONS

This research has tested the hypothesis of *digital evidence obtained from live investigations involving encryption can be shown to be reliable*, which has involved investigating the strengths and weaknesses of performing live investigations of systems that use encryption. While it is not possible to say that digital evidence from live investigations involving encryption is reliable, since this is investigation dependent, it has been possible to produce a set of criteria, against which reliability can be assessed. The explanations of these requirements for digital evidence have been clearly defined and the research as a whole acts as an example of how they can be used.

Also, categorisations of encryption product have been validated and it has been shown how these affect the locations on disk that become inaccessible. It has also been shown how these categories affect offline approaches to attempting to gain access to encrypted digital evidence. This research showed that it is difficult to predict the success of offline approaches and therefore offline access may or may not be possible. However, live investigations allow data to be preserved in all cases and particularly in the case of Full Disk or Full Volume Encryption, are likely to offer a significant increase in completeness.

The adverse affect on completeness by performing live investigations has also been explored. A methodology and software tool has been developed that simplifies the process of recording changes made to test systems. These allow the footprints of live tools to be determined. Testing also produced some specific results, including the advantages of acquiring an image of memory followed by extracting information such

as processes running from the image, rather than using live investigation tools such as *pslist* to produce the same information.

A general definition for error in digital investigations has also been proposed, which was lacking in current literature. A clear definition of error in digital investigations based around alternative hypotheses for digital objects having their current state provides direction for the expression of error when presenting digital evidence. After defining error, it was clear that in a live investigation, the alternative hypotheses for acquired data having its current state were the operating system providing false information to the acquisition tool, or the acquisition tool obtaining data incorrectly. To address this problem, a method was developed that used repeatability and the use of trusted operating systems as means of demonstrating the accuracy of live acquired copies of encrypted evidence. This involved acquiring specific information from the system at the same time as a decrypted copy of the encrypted evidence, which allowed offline decryption of the static encrypted data. This was demonstrated in two ways: using the built in GUI of *BitLocker* and recovering decryption keys from a memory dump of a system running *TrueCrypt*. This approach can be extended for all on-the-fly encryption systems.

Finally, it has also been shown how physical origin of live acquired data can be demonstrated by integrating physical identifiers that are available before and after ‘pulling the plug’ into the acquisition process.

Many of these contributions have resulted in peer reviewed publications. Obtaining recovery keys in order to allow later access to *Windows Vista Bitlocker* encrypted data was discussed in Hargreaves and Chivers (2007) and Hargreaves *et al.* (Hargreaves *et al.*, 2008). The latter also discussed the difficulty in gaining offline access to *EFS* encrypted files on *Windows Vista*. The key recovery approach to demonstrating accuracy of acquired digital evidence was discussed in Hargreaves and Chivers (2008b), where the ‘linear scan’ approach to key recovery was introduced. This key recovery approach was also used in Hargreaves and Chivers (2008a) to demonstrate how live imaging could be avoided in cases where it is impractical, such as when very large amounts of data are involved. Both papers on key recovery also

included other aspects of this research, including the types of offline approaches for gaining access to encrypted evidence.

9.3 FUTURE WORK

This research has also opened up many opportunities for further work. First, the criteria proposed for assessing the reliability of digital evidence could be applied to other types of digital evidence. An obvious example would be mobile phones since evidence is often obtained using a live investigation, i.e. using the operating system of the system under investigation to recover evidence, and no literature could be found on the reliability of digital evidence obtained from mobile phones.

One of the most interesting areas for future work is determining the footprint of live investigation tools and techniques. The developed methodology can be used to identify changes made to test systems and can be used to predict the locations of artefacts left by a live investigation. However, an individual post-live investigation analysis of a machine is still necessary for each case. Developing an optimised and standard methodology for performing this later analysis would speed up an investigator's ability to assess the changes made by the live tools used and determine which digital evidence may have been lost. Standardising this part of the analysis does not suffer from the same difficulties as attempts to standardise general digital investigations (including problems such as the diversity in investigations and the number of different questions to be answered), since only a single question is being asked – what changes were made to the system due to the investigators actions?

The actual methodology and software developed for live tool testing also enables other future work; firstly, they can both can be significantly improved. The methodology could be changed so that the live logging tools are not used, which would make the monitoring completely unintrusive. However, for this to be possible, other changes would need to be made, as described in Chapter 5, for example, the disk caching problem overcome, and an inspection of changes in unallocated space. Making the method completely unintrusive offers the advantage that changes to memory could be monitored simultaneously, since the memory of the system would

not be modified by the monitoring tools. The analysis of changes caused could also be improved, with more experiments into ‘background changes’, and more significantly, a modification of the reporting environment from a simple HTML report to a full GUI that allows recorded changes to be easily inspected to determine if they are relevant. Also, there is the potential for further automation. *VMware* offers an API which has not yet been fully explored, but at least allows virtual machines to be paused and resumed from the command line. This is a small optimisation but it may allow one-click snapshot generation, simplifying the collection of test data.

The developed system monitoring methodology, improved or otherwise, could be used to examine additional live investigation tools and techniques e.g. other memory acquisition tools, connecting via Ethernet, etc. It could also be used to examine them in greater detail: repeating the tests, using systems with different background software running e.g. antivirus, and on different service packs\operating systems. These may or may not make significant differences to the changes caused but experimentation is needed to determine this.

Identifying changes caused by live tools also has implications for the authenticity requirement; specifically that by identifying the changes made by live tools, it may be possible to use the artefacts that remain on a system after a live investigation to support investigators’ records of their actions performed on the live system.

Also, the methodology and tools developed could be used to significantly ease a popular area of digital investigation research: determining the forensic artefacts left by pieces of software. Using the developed methodology and tools it is possible to generate comprehensive reports detailing the changes caused by performing actions on a system, e.g. running *Skype* from a USB stick. If the tools are improved in the manner described earlier, recorded changes could be inspected through an interface designed for highlighting these changes, allowing relevant changes to be easily identified. The automated nature of the tools, particularly if the *VMware* API can be utilised, allows these reports to be very easily generated, allowing the investigator/researcher to concentrate on the analysis of the recorded changes.

This also raises a question about how to store and present these changes. This problem applies to changes caused by live investigation tools and also artefacts left by pieces of software. One of the difficulties is presenting the results in a form that is useful. Currently this achieved by interspersing file or Registry paths between text that explains the cause of the change. A standard, structured format for storing and disseminating this type of information would enable querying and visualisation tools to be built on top of this standard format, which would allow different methods for displaying this information to be developed. This has the potential to improve the process of sharing this type of information, which is extremely common in digital investigation research.

Additional future work is possible due to the research into accuracy of digital evidence. The demonstration of accuracy through key recovery was only performed for two products. Key recovery from memory has already become a popular area for research, with alternative approaches to the developed linear scan technique already published. There are an increasing number of on-the-fly encryption products available and key recovery approaches will be possible until keys are stored securely in hardware. However, developers of on-the-fly encryption software are aware of key recovery approaches and are modifying the way in which keys are stored in memory to defeat simple approaches. Therefore, developing key recovery techniques is likely to be a continuous source of future work. Also, research can be performed into demonstrating accuracy in live investigations that do not involve encrypted evidence, i.e. where offline data recovery is not possible. Applying a Certainty Scale such as that in Casey (2002a) has the potential to allow accuracy of digital evidence obtained from live acquisitions to be assessed by acquiring data from multiple sources and correlating the results. Exploring the anti-forensic techniques for different memory acquisition techniques would increase the understanding of what could cause acquired data from memory to have its state, and given the known technical expertise of the suspect, likelihood could be provided by an experienced investigator of these alternative explanations.

Finally, future work that is possible due to the research into demonstrating the authenticity of live acquired digital evidence includes exploring the procedural mechanisms used to demonstrate how live evidence was obtained, including requiring technological solutions such as video capture and Digital Evidence Bags, and how these could be used to increase confidence in the origin of live acquired digital evidence.

9.4 FINAL SUMMARY

Traditional digital investigations, i.e. ‘pulling the plug’ have the advantage of preserving the contents of the computer’s hard drive at a specific point in time, since while the power is removed no data can be written to the disk. However, when the power is removed, this means that volatile data, including data in memory is lost. This also has implications for investigating systems that are using encryption since while decrypted content may be accessible when the system is running, once the power is removed, the decrypted content, which may include all of the drive, may revert to its encrypted state and therefore become inaccessible.

As a result, this has led to the use of live investigations, where the computer system is investigated while it is still running, using the operating system of the suspect’s machine. Such investigations are useful when encryption is involved since when a live investigation is performed, the investigator has the same access to the system as the suspect had prior to the investigator taking physical control of the machine. Therefore, if encrypted data was accessible to the suspect at the time of the seizure, then the investigator would also have access to the decrypted content.

However, there are a number of difficulties with live investigations, including the difficulty in trusting the data supplied to the live tools; the inherent intrusiveness of live techniques; the difficulty in verifying the output of live tools; and also ensuring that no evidence is missed. Due to these difficulties it is possible that digital evidence from live investigations is used when it should not be, or is not used when it could be; either way potentially resulting in an incorrect decision being made. Given that live investigations are a useful technique for addressing the problem of encrypted

evidence, but have associated difficulties, the aim of this research was therefore to determine the role that live investigations could play when encrypted evidence is involved. The research hypothesis that was tested was *digital evidence obtained from live investigations involving encryption can be shown to be reliable*.

To test this hypothesis, this research first defined reliability as being assessed using three requirements: authenticity, accuracy and completeness. The remainder of the research evaluated evidence from live investigations against these three criteria. The requirement of authenticity was discussed in Chapter 7 and was shown to be satisfied by recording the processes run, using hashes to demonstrate that acquired digital evidence has not been tampered with, and hashing live acquired evidence with hardware identifiers to demonstrate its physical origin. Accuracy was discussed in Chapter 6 and it was demonstrated how certain sources of error in live acquisitions of encrypted data could be eliminated by also obtaining other specific information at the time of the live acquisition that allowed the static encrypted data to be decrypted later during an offline examination in a repeatable manner using a trusted operating system. This eliminates alternative hypotheses that acquired data contains error due to a manipulated operating system or faulty acquisition tools. However, potential error remains in that the investigators actions triggered a ‘logic bomb’ that manipulated evidence in some way prior to acquisition. This may or may not be significant depending on the specific investigation. Completeness was discussed in Chapters 4 and 5 and it was shown to be significantly increased by performing a live investigation, since this can preserve data in a form that can be analysed rather than being encrypted. However, live investigations also overwrite data on the live machine, and the significance of this will depend on the specific investigation. A software monitoring tool and a methodology has been developed that assists in predicting these changes and identifying which evidence has been overwritten on the system after the live investigation has been performed.

Therefore, it is possible for digital evidence from live investigations involving encryption to be considered to be reliable, but as discussed in the evaluation chapter, reliability of digital evidence depends on the specific investigation and the importance

of the decision being made. However, this research has provided structured criteria that allow the reliability of digital evidence to be assessed and the research as a whole has demonstrated the use of these criteria in the context of live digital investigations involving encryption and shown the extent to which each can currently be met.

REFERENCES

- ACCESS DATA (2007) FTK IMAGER,
[HTTP://DOWNLOADS.ACCESSDATA.COM/CURRENT_RELEASES/IMAGER/IMAGER-FTK_IMAGER-2.5.3.EXE](http://downloads.accessdata.com/current_releases/imager/imager-ftk_imager-2.5.3.exe)
- ACCESS DATA (2008A) FORENSIC TOOLKIT PRODUCT DESCRIPTION,
[HTTP://WWW.ACCESSDATA.COM/FORENSICTOOLKIT.HTML](http://www.accessdata.com/forensictoolkit.html)
- ACCESS DATA (2008B) PASSWORD RECOVERY TOOLKIT (PRTK) PRODUCT DESCRIPTION,
[HTTP://WWW.ACCESSDATA.COM/DECRYPTIONTOOL.HTML#PASSWORDRECOVERYTOOLKIT](http://www.accessdata.com/decryptiontool.html#passwordrecoverytoolkit)
- ACPO (2003) *GOOD PRACTICE GUIDE FOR COMPUTER BASED ELECTRONIC EVIDENCE V.3*, ASSOCIATION OF CHIEF POLICE OFFICERS.
- ACPO (2007) *GOOD PRACTICE GUIDE FOR COMPUTER BASED ELECTRONIC EVIDENCE V.4*, ASSOCIATION OF CHIEF POLICE OFFICERS.
- ADELSTEIN, F. (2006) LIVE FORENSICS: DIAGNOSING YOUR SYSTEM WITHOUT KILLING IT FIRST. *COMMUNICATIONS OF THE ACM*, 49, 63-66.
- ANON (2007) "DEFEATING" WHOLE DISK ENCRYPTION - PART 1, [HTTP://BREACH-INV.BLOGSPOT.COM/2007/05/DEFEATING-WHOLE-DISK-ENCRYPTION-PART-1.HTML](http://breach-inv.blogspot.com/2007/05/defeating-whole-disk-encryption-part-1.html)
- AXANTUM SOFTWARE (2008) AxCRYPT FILE ENCRYPTION FOR WINDOWS,
[HTTP://WWW.AXANTUM.COM/AXCRYPT/](http://www.axantum.com/axcrypt/)
- BARRETT, N. (2005) HOW TO DEAL WITH STRONG ENCRYPTION?,
[HTTP://SOFTWARE.SILICON.COM/SECURITY/0,39024655,39153438,00.HTM](http://software.silicon.com/security/0,39024655,39153438,00.htm)
- BARYAMUREEBA, V. & TUSHABE, F. (2004) THE ENHANCED DIGITAL INVESTIGATION PROCESS MODEL. *DIGITAL FORENSIC RESEARCH WORKSHOP*, 2004.
- BEEBE, N. L. & CLARK, J. G. (2005) A HIERARCHICAL, OBJECTIVES-BASED FRAMEWORK FOR THE DIGITAL INVESTIGATIONS PROCESS. *DIGITAL INVESTIGATION*, 2, 147-167.
- BILBY, D. (2006) LOW DOWN AND DIRTY: ANTI-FORENSIC ROOTKITS,
[HTTP://WWW.BLACKHAT.COM/PRESENTATIONS/BH-JP-06/BH-JP-06-BILBY-UP.PDF](http://www.blackhat.com/presentations/BH-JP-06/BH-JP-06-Bilby-up.pdf)
- BOLIEAU, A. (2006) FIREWIRE, DMA & WINDOWS, [HTTP://WWW.STORM.NET.NZ/PROJECTS/16](http://www.storm.net.nz/projects/16)
- BOLIEAU, A. (UNDATED) BIOSKBSNARF, [HTTP://WWW.STORM.NET.NZ/STATIC/FILES/BIOSKBSNARF](http://www.storm.net.nz/static/files/bioskbsnarf)
- BURDACH, M. (2004) FORENSIC ANALYSIS OF A LIVE LINUX SYSTEM: PART 1.
- BURDACH, M. (2005) AN INTRODUCTION TO WINDOWS MEMORY FORENSIC,
[HTTP://FORENSIC.SECCURE.NET/PDF/INTRODUCTION_TO_WINDOWS_MEMORY_FORENSIC.PDF](http://forensic.seccure.net/pdf/introduction_to_windows_memory_forensic.pdf)
- CARRIER, B. (2002) OPEN SOURCE DIGITAL FORENSICS TOOLS: THE LEGAL ARGUMENT. @STAKE RESEARCH REPORT.
- CARRIER, B. (2003) DEFINING DIGITAL FORENSIC EXAMINATION AND ANALYSIS TOOLS USING ABSTRACTION LAYERS. *INTERNATIONAL JOURNAL OF DIGITAL EVIDENCE*, 1.
- CARRIER, B. (2005) *FILE SYSTEM FORENSIC ANALYSIS*, ADDISON WESLEY.
- CARRIER, B. (2006A) A HYPOTHESIS BASED APPROACH TO DIGITAL FORENSIC INVESTIGATIONS. PURDUE UNIVERSITY.
- CARRIER, B. (2006B) RISKS OF LIVE DIGITAL FORENSIC ANALYSIS. *COMMUNICATIONS OF THE ACM*, 49, 56-61.
- CARRIER, B. (2009) THE SLEUTH KIT, [HTTP://WWW.SLEUTHKIT.ORG/](http://www.sleuthkit.org/)
- CARRIER, B. & GRAND, J. (2004) A HARDWARE BASED MEMORY ACQUISITION PROCEDURE FOR DIGITAL INVESTIGATIONS. *DIGITAL INVESTIGATION*, 1, 50-60.
- CARRIER, B. & SPAFFORD, E. (2003) GETTING PHYSICAL WITH THE DIGITAL INVESTIGATION PROCESS. *INTERNATIONAL JOURNAL OF DIGITAL EVIDENCE*, 2, 1-20.
- CARRIER, B. & SPAFFORD, E. (2006) CATEGORIES OF DIGITAL INVESTIGATION ANALYSIS TECHNIQUES BASED ON THE COMPUTER HISTORY MODEL. *DIGITAL INVESTIGATION* 3, 121-130.
- CARVEY (2007A) PREFETCH ANALYSIS, [HTTP://WINDOWSIR.BLOGSPOT.COM/2007/05/PREFETCH-ANALYSIS.HTML](http://windowsir.blogspot.com/2007/05/prefetch-analysis.html)
- CARVEY, H. (2004) INSTANT MESSAGING INVESTIGATIONS ON A LIVE WINDOWS XP SYSTEM. *DIGITAL INVESTIGATION*, 1, 256-260.

-
- CARVEY, H. (2005) MALWARE ANALYSIS FOR WINDOWS ADMINISTRATORS. *DIGITAL INVESTIGATION*, 2, 19-22.
- CARVEY, H. (2007B) WINDOWS FORENSIC ANALYSIS.
- CARVEY, H. & ALTHEIDE, C. (2005) TRACKING USB STORAGE: ANALYSIS OF WINDOWS ARTIFACTS GENERATED BY USB STORAGE DEVICES. *DIGITAL INVESTIGATION*, 2, 94-100.
- CASEY, E. (2002A) ERROR, UNCERTAINTY, AND LOSS IN DIGITAL EVIDENCE. *INTERNATIONAL JOURNAL FOR DIGITAL EVIDENCE*, 1.
- CASEY, E. (2002B) PRACTICAL APPROACHES TO RECOVERING ENCRYPTED DIGITAL EVIDENCE. *INTERNATIONAL JOURNAL FOR DIGITAL EVIDENCE*, 1.
- CASEY, E. (2004A) *DIGITAL EVIDENCE AND COMPUTER CRIME*, ACADEMIC PRESS.
- CASEY, E. (2004B) TOOL REVIEW - WINHEX. *DIGITAL INVESTIGATION*, 1, 114-128.
- CASEY, E. (2007) WHAT DOES "FORENSICALLY SOUND" REALLY MEAN? *DIGITAL INVESTIGATION*, 4, 49-50.
- CASEY, E. & STANLEY, A. (2004) TOOL REVIEW - REMOTE FORENSIC PRESERVATION AND EXAMINATION TOOLS. *DIGITAL INVESTIGATION*, 1, 284-297.
- CERT (2007) LIVEVIEW 0.6, [HTTP://LIVEVIEW.SOURCEFORGE.NET/](http://liveview.sourceforge.net/)
- CIARDHUÁIN, S. (2004) AN EXTENDED MODEL OF CYBERCRIME INVESTIGATIONS. *INTERNATIONAL JOURNAL OF DIGITAL EVIDENCE*, 3, 1-22.
- CRAIGER, J. P., POLLITT, M. & SWAUGER, J. (2005) LAW ENFORCEMENT AND DIGITAL EVIDENCE, [HTTP://NCFS.ORG/CRAIGER.DELF.REVISION.PDF](http://ncfs.org/craiger_delf_revision.pdf)
- CYPHERIX (2008) CRYPTAINER LE - FREE 128BIT ENCRYPTION SOFTWARE, [HTTP://WWW.CYPHERIX.COM/CRYPTAINERLE/](http://www.cipherix.com/cryptainerle/)
- DENNING, D. E. (1999) *INFORMATION WARFARE AND SECURITY*, ADDISON WESLEY.
- DENNING, D. E. & BAUGH, W. E. (1999) HIDING CRIMES IN CYBERSPACE. *INFORMATION, COMMUNICATION & SOCIETY*, 2, 251-276.
- DEVINE, C. (2006) FIPS-197 COMPLIANT AES IMPLEMENTATION, [HTTP://FILES.CODES-SOURCES.COM/FICHER.ASPX?ID=44080&F=LIB%5CAES.C](http://files.codes-sources.com/fichier.aspx?id=44080&f=lib%5caes.c)
- DICKSON, M. (2006A) AN EXAMINATION INTO AOL INSTANT MESSENGER 5.5 CONTACT IDENTIFICATION. *DIGITAL INVESTIGATION*, 3, 227-237.
- DICKSON, M. (2006B) AN EXAMINATION INTO MSN MESSENGER 7.5 CONTACT IDENTIFICATION. *DIGITAL INVESTIGATION*, 3, 79-83.
- DICKSON, M. (2006C) AN EXAMINATION INTO YAHOO MESSENGER 7.0 CONTACT IDENTIFICATION. *DIGITAL INVESTIGATION*, 3, 159-165.
- DICKSON, M. (2007) AN EXAMINATION INTO TRILLIAN BASIC 3.X CONTACT IDENTIFICATION. *DIGITAL INVESTIGATION*, 4, 36-45.
- DOLAN-GAVITT, B. (2007) THE VAD TREE: A PROCESS-EYE VIEW OF PHYSICAL MEMORY. *DIGITAL INVESTIGATION*, 4, 62-64.
- E-FENSE (2008) HELIX LIVE CD v2.0,
- EFD SOFTWARE (2008) HD TUNE, [HTTP://WWW.HDTUNE.COM/](http://www.hdtune.com/)
- EVANS, J. R. (2007) VOLATILE DATA ACQUISITION AND ANALYSIS: TOOLS AND TECHNIQUES. UNIVERSITY OF GLAMORGAN.
- FARMER, D. & VENEMA, W. (2004) *FORENSIC DISCOVERY*, ADDISON WESLEY.
- FARMER, D. J. (2007) A WINDOWS REGISTRY QUICK REFERENCE: FOR THE EVERYDAY EXAMINER, [HTTP://WWW.FORENSICFOCUS.COM/DOWNLOADS/WINDOWS-REGISTRY-QUICK-REFERENCE.PDF](http://www.forensicfocus.com/downloads/windows-registry-quick-reference.pdf)
- FELLOWS, G. H. (2005) THE JOYS OF COMPLEXITY AND THE DELETED FILE. *DIGITAL INVESTIGATION*, 2, 89-93.
- FLORIO, E. (2005) WHEN MALWARE MEETS ROOTKITS, [WWW.SYMANTEC.COM/AVCENTER/REFERENCE/WHEN.MALWARE.MEETS.ROOTKITS.PDF](http://www.symantec.com/avcenter/reference/when.malware.meets.rootkits.pdf)
- FORSTER, P. (2005) THE AUTOMATED IDENTIFICATION OF ENCRYPTION DURING FORENSIC COMPUTING EXAMINATIONS. *DEPARTMENT OF INFORMATION SYSTEMS*. CRANFIELD UNIVERSITY.
- FOUNDSTONE (2000) BINTXT, [HTTP://WWW.FOUNDSTONE.COM/US/RESOURCES/PRODDESC/BINTEXT.HTM](http://www.foundstone.com/us/resources/proddesc/bintxt.htm)
- GAJIC, Z. (2008) STORE USER AND APPLICATION DATA IN THE CORRECT LOCATION, [HTTP://DELPHI.ABOUT.COM/OD/KBWINSHELL/A/SHGETFOLDERPATH.HTM](http://delphi.about.com/od/kbwinshell/a/shgetfolderpath.htm)
- GARNER, G. (2007) FORENSIC ACQUISITION UTILITIES. [HTTP://WWW.GMGSYSTEMSINC.COM/FAU/](http://www.gmgsystemsinc.com/FAU/)
-

-
- GETDATA (2008) MOUNT IMAGE PRO, [HTTP://WWW.MOUNTIMAGE.COM/](http://www.mountimage.com/)
- GHAVALAS, B. & PHILIPS, A. (2005) TROJAN DEFENCE: A FORENSIC VIEW PART II. *DIGITAL INVESTIGATION*, 2, 133-136.
- GOOGLE (2008) EXPLORE GOOGLE CHROME FEATURES: INCOGNITO MODE, [HTTP://WWW.GOOGLE.COM/SUPPORT/CHROME/BIN/ANSWER.PY?HL=EN&ANSWER=95464](http://www.google.com/support/chrome/bin/answer.py?hl=en&answer=95464)
- GORLANI, M. (2008) MAC MAKEUP 1.95D, [HTTP://WWW.GORLANI.COM/PUBLICPRJ/MACMAKEUP/MACMAKEUP.ASP](http://www.gorlani.com/publicprj/macmakeup/macmakeup.asp)
- HAAGMAN, D. & GHAVALAS, B. (2005) TROJAN DEFENCE: A FORENSIC VIEW. *DIGITAL INVESTIGATION*, 2, 23-30.
- HALDERMAN, J. A., SCHOEN, S. D., HENINGER, N., CLARKSON, W., PAUL, W., CALANDRINO, J. A., FELDMAN, A. J., APPELBAUM, J. & FELTEN, E. W. (2008) LEST WE REMEMBER: COLD BOOT ATTACKS ON ENCRYPTION KEYS, [HTTP://CFTP.PRINCETON.EDU/MEMORY/](http://cftp.princeton.edu/memory/)
- HARBOUR, N. (2006) DCFLDD - LATEST VERSION 1.3.4-1 [HTTP://DCFLDD.SOURCEFORGE.NET/](http://dcfldd.sourceforge.net/)
- HARGREAVES, C. (2007) MONITORING THE MONITORING TOOLS: A ZERO-FOOTPRINT LIVE DISK MONITORING TECHNIQUE USING VMWARE, LINUX AND MD5DEEP. *F3 ANNUAL CONFERENCE 2007*.
- HARGREAVES, C. & CHIVERS, H. (2007) POTENTIAL IMPACTS OF WINDOWS VISTA ON DIGITAL INVESTIGATIONS. *PROCEEDINGS FROM 2ND ADVANCES IN COMPUTER SECURITY AND FORENSICS*. LIVERPOOL, UK.
- HARGREAVES, C. & CHIVERS, H. (2008A) AVOIDING LIVE IMAGING OF LARGE ENCRYPTED VOLUMES BY RECOVERING KEYS FROM MEMORY. *PROCEEDINGS FROM 3RD ADVANCES IN COMPUTER SECURITY AND FORENSICS*. LIVERPOOL, UK.
- HARGREAVES, C. & CHIVERS, H. (2008B) RECOVERY OF ENCRYPTION KEYS FROM MEMORY USING A LINEAR SCAN. *THE INTERNATIONAL WORKSHOP ON DIGITAL FORENSICS*. BARCELONA, SPAIN.
- HARGREAVES, C., CHIVERS, H. & TITHERIDGE, D. (2008) WINDOWS VISTA AND DIGITAL INVESTIGATIONS. *DIGITAL INVESTIGATION*, 5, 34-48.
- HBGARY (2008A) FASTDUMP - A MEMORY DUMPING TOOL
- HBGARY (2008B) RESPONDER PROFESSIONAL, [HTTP://WWW.HBGARY.COM/RESPONDER_PRO.HTML](http://www.hbgary.com/responder_pro.html)
- HOGFLY (2007) DETECTING BITLOCKER. [HTTP://FORENSICIR.BLOGSPOT.COM/2007/03/DETECTING-BITLOCKER.HTML](http://forensicir.blogspot.com/2007/03/detecting-bitlocker.html).
- HOGLUND, G. & BUTLER, J. (2006) *ROOTKITS: SUBVERTING THE WINDOWS KERNEL*, PEARSON EDUCATION.
- HOME OFFICE (2006) INVESTIGATION OF PROTECTED ELECTRONIC INFORMATION: A PUBLIC CONSULTATION, [HTTP://WWW.HOMEOFFICE.GOV.UK/DOCUMENTS/CONS-2006-RIPA-PART3/RIPA-PART3.PDF](http://www.homeoffice.gov.uk/documents/cons-2006-ripa-part3/ripa-part3.pdf)
- HUEBNER, E., BEM, D., HENSKENS, F. & WALLIS, M. (2007) PERSISTENT SYSTEMS TECHNIQUES IN FORENSIC ACQUISITION OF MEMORY. *DIGITAL INVESTIGATION*, 4, 129-137.
- HYNES, B. (2008) ADVANCES IN BITLOCKER DRIVE ENCRYPTION, [HTTP://TECHNET.MICROSOFT.COM/EN-US/MAGAZINE/CC510321.ASPX](http://technet.microsoft.com/en-us/magazine/cc510321.aspx)
- IEEE (2007) P1619/D18: DRAFT STANDARD FOR CRYPTOGRAPHIC PROTECTION OF DATA ON BLOCK-ORIENTED STORAGE DEVICES, [HTTP://IEEEXPLORE.IEEE.ORG/SERVLET/OPAC?PUNUMBER=4375276](http://ieeexplore.ieee.org/servlet/opac?punumber=4375276)
- INDIGOSTAR SOFTWARE (2008) PERL2EXE HOME PAGE, [HTTP://WWW.INDIGOSTAR.COM/PERL2EXE.HTM](http://www.indigostar.com/perl2exe.htm)
- IRONGEEK (2007) CRACKING CACHED DOMAIN/ACTIVE DIRECTORY PASSWORDS ON WINDOWS XP/2000/2003 [HTTP://WWW.IRONGEEK.COM/I.PHP?PAGE=SECURITY/CACHECRACK](http://www.irongEEK.com/i.php?page=security/cachecrack)
- JETICO (2008) BESTCRYPT FOR WINDOWS, [HTTP://WWW.JETICO.COM/BCRYPT8.HTM](http://www.jetico.com/bcrypt8.htm)
- KAPLAN, B. (2007) LIVE VIEW. [HTTP://LIVEVIEW.SOURCEFORGE.NET/](http://liveview.sourceforge.net/).
- KAPLAN, B. (2008) RAM IS KEY: EXTRACTING DISK ENCRYPTION KEYS FROM VOLATILE MEMORY. CARNegie MELLON UNIVERSITY.
- KEITH, M., SHAO, B. & STEINBART, P. J. (2006) THE USABILITY OF PASSPHRASES FOR AUTHENTICATION: AN EMPIRICAL FIELD STUDY. *INTERNATIONAL JOURNAL OF HUMAN COMPUTER STUDIES*.
-

-
- KENNEALLY, E. E. & BROWN, C. L. T. (2005) RISK SENSITIVE DIGITAL EVIDENCE COLLECTION. *DIGITAL INVESTIGATION*, 2, 101-119.
- KENNEDY, I. (2006) IT WAS THE BIG WOODEN HORSE, YOUR HONOUR, [HTTP://WWW.BCS.ORG/SERVER.PHP?SHOW=CONWEBDOC.6232](http://www.bcs.org/server.php?show=conwebdoc.6232)
- KOCH, W. (2007) THE GNU PRIVACY GUARD, [HTTP://WWW.GNUPG.ORG/](http://www.gnupg.org/)
- KORNBLUM (2008) MD5DEEP, [HTTP://MD5DEEP.SOURCEFORGE.NET/](http://md5deep.sourceforge.net/)
- KORNBLUM, J. (2007) USING EVERY PART OF THE BUFFALO IN WINDOWS MEMORY ANALYSIS. *DIGITAL INVESTIGATION*, 4, 24-29.
- LYLE, J. (2006) A STRATEGY FOR TESTING HARDWARE WRITE BLOCK DEVICES. *DIGITAL INVESTIGATION*, 3, 3-9.
- MACLEAN, N. P. (2006) ACQUISITION AND ANALYSIS OF WINDOWS MEMORY. UNIVERSITY OF STRATHCLYDE.
- MANDIA, K., PROSISE, C. & PEPE, M. (2003) *INCIDENT RESPONSE AND COMPUTER FORENSICS*, OSBORNE MCGRAW-HILL.
- MANTECH (2008) MEMORY DD, [HTTP://WWW.MANTECH.COM/MSMA/MDD.ASP](http://www.mantech.com/msma/mdd.asp)
- MAPLINS (2008) PS/2 KEYSHARK™, [HTTP://WWW.MAPLIN.CO.UK/MODULE.ASPX?MODULENO=220174](http://www.maplin.co.uk/module.aspx?moduleNo=220174)
- MCGREW, W. (2008) MSRAMDMP: MCGREW SECURITY RAM DUMPER, [HTTP://WWW.MCGREWSECURITY.COM/TOOLS/MSRAMDMP/](http://www.mcgrewsecurity.com/tools/msramdmp/)
- MCKEMMISH, R. (1999) WHAT IS FORENSIC COMPUTING? *TRENDS AND ISSUES IN CRIME AND CRIMINAL JUSTICE*, 1-6.
- MCQUOWN, R. (2008) BIOS MAGIC NUMBERS IN RAM (BETA), [HTTP://FORENSICZONE.BLOGSPOT.COM/2008/05/BIOS-MAGIC-NUMBERS-IN-RAM-BETA.HTML](http://forensiczone.blogspot.com/2008/05/bios-magic-numbers-in-ram-beta.html)
- MICROSOFT (2002) *MICROSOFT COMPUTER DICTIONARY, FIFTH EDITION*.
- MICROSOFT (2004) RAM, VIRTUAL MEMORY, PAGEFILE AND ALL THAT STUFF, [HTTP://SUPPORT.MICROSOFT.COM/KB/555223](http://support.microsoft.com/kb/555223)
- MICROSOFT (2006A) BITLOCKER DRIVE ENCRYPTION HARDWARE ENHANCED DATA PROTECTION, [HTTP://DOWNLOAD.MICROSOFT.COM/DOWNLOAD/5/B/9/5B97017B-E28A-4BAE-BA48-174CF47D23CD/CPA064_WH06.PPT](http://download.microsoft.com/download/5/b/9/5b97017b-e28a-4bae-ba48-174cf47d23cd/cpa064_wh06.ppt)
- MICROSOFT (2006B) BITLOCKER DRIVE ENCRYPTION: TECHNICAL OVERVIEW, [HTTP://WWW.MICROSOFT.COM/TECHNET/WINDOWSVISTA/SECURITY/BITTECH.MSPX](http://www.microsoft.com/technet/windowsvista/security/bittech.msp)
- MICROSOFT (2006C) ENCRYPTING FILE SYSTEM OVERVIEW, [HTTP://TECHNET.MICROSOFT.COM/EN-US/LIBRARY/CC700811.ASPX](http://technet.microsoft.com/en-us/library/cc700811.aspx)
- MICROSOFT (2006D) How EFS WORKS, [HTTP://WWW.MICROSOFT.COM/TECHNET/PRODTECHNOL/WINDOWS2000SERV/RESKIT/DISTRIB/D SCK_EFS_DUWF.MSPX?MFR=TRUE](http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/distrib/dsck_efs_duwf.msp?mfr=true)
- MICROSOFT (2006E) USER ACCOUNT CONTROL OVERVIEW. [HTTP://TECHNET.MICROSOFT.COM/EN-US/WINDOWSVISTA/AA906021.ASPX](http://technet.microsoft.com/en-us/windowsvista/aa906021.aspx).
- MICROSOFT (2006F) WINDOWS VISTA PRODUCT GUIDE, [HTTP://WWW.MICROSOFT.COM/DOWNLOADS/DETAILS.ASPX?FAMILYID=BBC16EBF-4823-4A12-AFE1-5B40B2AD3725&DISPLAYLANG=EN](http://www.microsoft.com/downloads/details.aspx?familyID=BBC16EBF-4823-4A12-AFE1-5B40B2AD3725&displaylang=en)
- MICROSOFT (2007A) HOW THE RECYCLE BIN STORES FILES, [HTTP://SUPPORT.MICROSOFT.COM/KB/136517](http://support.microsoft.com/kb/136517)
- MICROSOFT (2007B) HOW TO ENCRYPT DATA VOLUMES IN WINDOWS VISTA. [HTTP://SUPPORT.MICROSOFT.COM/KB/933637](http://support.microsoft.com/kb/933637).
- MICROSOFT (2007C) USER MODE PROCESS DUMPER VERSION 8.1, [HTTP://WWW.MICROSOFT.COM/DOWNLOADS/DETAILS.ASPX?FAMILYID=E089CA41-6A87-40C8-BF69-28AC08570B7E&DISPLAYLANG=EN](http://www.microsoft.com/downloads/details.aspx?familyID=E089CA41-6A87-40C8-BF69-28AC08570B7E&displaylang=en)
- MICROSOFT (2007D) WINDOWS BITLOCKER DRIVE ENCRYPTION STEP-BY-STEP GUIDE, [HTTP://TECHNET2.MICROSOFT.COM/WINDOWSVISTA/EN/LIBRARY/C61F2A12-8AE6-4957-B031-97B4D762CF311033.MSP?MFR=TRUE](http://technet2.microsoft.com/windowsvista/en/library/c61f2a12-8ae6-4957-b031-97b4d762cf311033.msp?mfr=true)
- MICROSOFT (2007E) WINDOWS FEATURE LETS YOU GENERATE A MEMORY DUMP FILE BY USING THE KEYBOARD, [HTTP://SUPPORT.MICROSOFT.COM/KB/244139](http://support.microsoft.com/kb/244139)
- MICROSOFT (2008A) LICENSE TERMS FOR SOFTWARE LICENSED FROM MICROSOFT, [HTTP://WWW.MICROSOFT.COM/ABOUT/LEGAL/USERTERMS/DEFAULT.ASPX](http://www.microsoft.com/about/legal/useterms/default.aspx)
-

-
- MICROSOFT (2008B) NOTABLE CHANGES IN WINDOWS VISTA SERVICE PACK 1, [HTTP://TECHNET.MICROSOFT.COM/EN-US/LIBRARY/CC709618.ASPX](http://technet.microsoft.com/en-us/library/cc709618.aspx)
- MICROSOFT (2008C) REGISTRY FUNCTIONS, [HTTP://MSDN.MICROSOFT.COM/EN-US/LIBRARY/MS724875\(VS.85\).ASPX](http://msdn.microsoft.com/en-us/library/ms724875(v5.85).aspx)
- MILLER, C. (1992) ELECTRONIC EVIDENCE - CAN YOU PROVE THE TRANSACTION TOOK PLACE. *COMPUTER LAWYER*, 9, 21-33.
- MOCAS, S. (2004) BUILDING THEORETICAL UNDERPINNINGS FOR DIGITAL FORENSIC RESEARCH. *DIGITAL INVESTIGATION*, 1, 61-68.
- MONDAY, B. (2004) FIRSTONSCENE.VBS: THE 10-SECOND FORENSIC DATA GATHERER, [HTTP://WWW.BMONDAY.COM/ARTICLES/975.ASPX](http://www.bmonday.com/articles/975.aspx)
- MORELLO, J. (2007) DEPLOYING EFS: PART 2, [HTTP://TECHNET.MICROSOFT.COM/EN-US/MAGAZINE/CC162489.ASPX](http://technet.microsoft.com/en-us/magazine/cc162489.aspx)
- MOZILLA (2008) PRIVATEBROWSING, [HTTPS://WIKI.MOZILLA.ORG/PRIVATEBROWSING](https://wiki.mozilla.org/PrivateBrowsing)
- NAPALM (2006) SATA DRIVE SERIAL NUMBER, [HTTP://FORUM.SYSINTERNALS.COM/FORUM_POSTS.ASP?TID=6643&PN=2](http://forum.sysinternals.com/forum_posts.asp?TID=6643&PN=2)
- NATIONAL INSTITUTE OF JUSTICE (2001) ELECTRONIC CRIME SCENE INVESTIGATION: A GUIDE FOR FIRST RESPONDERS. NATIONAL INSTITUTE OF JUSTICE.
- NATIONAL INSTITUTE OF JUSTICE (2004) *FORENSIC EXAMINATION OF DIGITAL EVIDENCE: A GUIDE FOR LAW ENFORCEMENT*, U.S. DEPARTMENT OF JUSTICE.
- NETWORK WORKING GROUP (2002) RFC3227 - GUIDELINES FOR EVIDENCE COLLECTION AND ARCHIVING, [HTTP://WWW.FAQS.ORG/RFC3227.HTML](http://www.faqs.org/rfcs/rfc3227.html)
- OBJECTIF SÉCURITÉ (2008) OPHCRACK, [HTTP://OPHCRACK.SOURCEFORGE.NET/](http://ophcrack.sourceforge.net/)
- OXFORD (2008) *CONCISE OXFORD ENGLISH DICTIONARY*, OXFORD UNIVERSITY PRESS.
- OXID.IT (2008) CAIN & ABEL, [HTTP://WWW.OXID.IT/CAIN.HTML](http://www.oxid.it/cain.html)
- PALMER, G. (2001) A ROAD MAP FOR DIGITAL FORENSIC RESEARCH. DIGITAL FORENSICS RESEARCH WORKSHOP.
- PALMER, G. (2002) FORENSIC ANALYSIS IN A DIGITAL WORLD. *INTERNATIONAL JOURNAL FOR DIGITAL EVIDENCE*, 1.
- PGP CORPORATION (2008) PGP PRODUCTS, [HTTP://WWW.PGP.COM/PRODUCTS/INDEX.HTML](http://www.pgp.com/products/index.html)
- PILON, A. (2005) CACHE_DUMP - RECOVERING WINDOWS PASSWORD CACHE ENTRIES, [HTTP://WWW.SECURITEAM.COM/TOOLS/5JP0I2KFPA.HTML](http://www.securiteam.com/tools/5JP0I2KFPA.html)
- POLLITT, M., NANCE, K., HAY, B., DODGE, R. C., CRAIGER, P., BURKE, P., MARBERRY, C. & BRUBAKER, B. (2008) VIRUTALIZATION AND DIGITAL FORENSICS: A RESEARCH AND EDUCATION AGENDA. *JOURNAL OF DIGITAL FORENSIC PRACTICE*, 2, 62-73.
- POLLITT, M. M. (1995) PRINCIPLES, PRACTICES, AND PROCEDURES: AN APPROACH TO STANDARDS IN COMPUTER FORENSICS. *SECOND INTERNATIONAL CONFERENCE ON COMPUTER EVIDENCE*. BALTIMORE.
- RAYMOND, E. S. (2001) *THE CATHEDRAL AND THE BAZAAR*, O'REILLY.
- REITH, M., CARR, C. & GUNSCH, G. (2002) AN EXAMINATION OF DIGITAL FORENSIC MODELS. *INTERNATIONAL JOURNAL FOR DIGITAL EVIDENCE*, 1.
- RICHARD_III, G. & ROUSSEV, V. (2006) NEXT-GENERATION DIGITAL FORENSICS. *COMMUNICATIONS OF THE ACM*, 49, 76-80.
- ROUSSEV, V. & RICHARD III, G. G. (2004) BREAKING THE PERFORMANCE WALL: THE CASE FOR DISTRIBUTED DIGITAL FORENSICS. *PROCEEDINGS OF THE 2004 DIGITAL FORENSICS RESEARCH WORKSHOP*.
- RUBENKING, N. J. (2000) STAY IN CONTROL: INCTRL5 [HTTP://WWW.PCMAG.COM/ARTICLE2/0,2817,9882,00.ASP](http://www.pcmag.com/article2/0,2817,9882,00.asp)
- RUFF, N. & SUICHE, M. (2007) ENTER SANDMAN. *PACSEC*. JAPAN.
- RUSSINOVICH, M. (2006) VOLUMEID v2.0, [HTTP://WWW.MICROSOFT.COM/TECHNET/SYSINTERNALS/FILEANDDISK/VOLUMEID.MSPX](http://www.microsoft.com/technet/sysinternals/fileanddisk/volumeid.msp)
- RUSSINOVICH, M. (2007) STRINGS v2.40, [HTTP://TECHNET.MICROSOFT.COM/EN-US/SYSINTERNALS/BB897439.ASPX](http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx)
- RUSSINOVICH, M. & COGSWELL, B. (2006A) FILEMON FOR WINDOWS v7.04, [HTTP://TECHNET.MICROSOFT.COM/EN-US/SYSINTERNALS/BB896642.ASPX](http://technet.microsoft.com/en-us/sysinternals/bb896642.aspx)
-

- RUSSINOVICH, M. & COGSWELL, B. (2006B) REGMON FOR WINDOWS v7.04, [HTTP://TECHNET.MICROSOFT.COM/EN-GB/SYSINTERNALS/BB896652.ASPX](http://technet.microsoft.com/en-gb/sysinternals/bb896652.aspx)
- RUSSINOVICH, M. & COGSWELL, B. (2008) PROCESS MONITOR v2.02, [HTTP://TECHNET.MICROSOFT.COM/EN-US/SYSINTERNALS/BB896645.ASPX](http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx)
- RUSSINOVICH, M. & SOLOMON (2005) *MICROSOFT WINDOWS INTERNALS*, MICROSOFT PRESS.
- RYNEARSON, J. (1989) *EVIDENCE AND CRIME SCENE RECONSTRUCTION*.
- SAMMES, T. & JENKINSON, B. (2000) *FORENSIC COMPUTING: A PRACTITIONER'S GUIDE*, SPRINGER.
- SAMMES, T. & JENKINSON, B. (2007) *FORENSIC COMPUTING: A PRACTITIONERS GUIDE, SECOND EDITION*, SPRINGER.
- SCHNEIER, B. (1996) *APPLIED CRYPTOGRAPHY*, JOHN WILEY & SONS INC.
- SCHUSTER, A. (2005) ACQUISITION (1): DD, [HTTP://COMPUTER.FORENSIKBLOG.DE/EN/2005/10/ACQUISITION_1_DD.HTML](http://computer.forensikblog.de/en/2005/10/acquisition_1_dd.html)
- SCHUSTER, A. (2006) SEARCHING FOR PROCESSES AND THREADS IN MICROSOFT WINDOWS MEMORY DUMPS. *DIGITAL INVESTIGATION*, 3S, 10-16.
- SCHUSTER, A. (2007) _EPROCESS VERSION 6.0.6000.16386, [HTTP://COMPUTER.FORENSIKBLOG.DE/EN/2007/01/EPROCESS_6_0_6000_16386.HTML](http://computer.forensikblog.de/en/2007/01/eprocess_6_0_6000_16386.html)
- SCHUSTER, A. (2008A) ACQUISITION (5): FIREWIRE, [HTTP://COMPUTER.FORENSIKBLOG.DE/EN/2008/02/ACQUISITION_5_FIREWIRE.HTML](http://computer.forensikblog.de/en/2008/02/acquisition_5_firewire.html)
- SCHUSTER, A. (2008B) NEW PHYSICAL MEMORY IMAGERS, [HTTP://COMPUTER.FORENSIKBLOG.DE/EN/2008/06/NEW_PHYSICAL_MEMORY_IMAGERS.HTML](http://computer.forensikblog.de/en/2008/06/new_physical_memory_imagers.html)
- SCIENTIFIC WORKING GROUP ON DIGITAL EVIDENCE (2000) DIGITAL EVIDENCE: STANDARDS AND PRINCIPLES. *FORENSIC SCIENCE COMMUNICATIONS*, 2.
- SCIENTIFIC WORKING GROUP ON DIGITAL EVIDENCE (2006) BECOME A MEMBER, [HTTP://NCFS.ORG/SWGDE/MEMBERSHIP.HTML](http://ncfs.org/swgde/membership.html)
- SOMMER, P. (1998) DIGITAL FOOTPRINTS: ASSESSING COMPUTER EVIDENCE. *CRIMINAL LAW REVIEW SPECIAL EDITION*, DECEMBER, 61-78.
- SOMMER, P. (1999) INTRUSION DETECTION SYSTEMS AS EVIDENCE. *COMPUTER NETWORKS*, 31, 2477-2487.
- SOMMER, P. (2004) THE CHALLENGES OF LARGE COMPUTER EVIDENCE CASES. *DIGITAL INVESTIGATION*, 1, 16-17.
- STOTTS, B. (2008) MDD, [HTTPS://SOURCEFORGE.NET/PROJECTS/MDD/](https://sourceforge.net/projects/mdd/)
- SUICHE, M. (2008A) CAPTURE MEMORY UNDER WIN2K3 OR VISTA WITH WIN32DD!, [HTTP://WWW.MSUICHE.NET/2008/06/14/CAPTURE-MEMORY-UNDER-WIN2K3-OR-VISTA-WITH-WIN32DD/](http://www.msuciche.net/2008/06/14/capture-memory-under-win2k3-or-vista-with-win32dd/)
- SUICHE, M. (2008B) WIN32DD, [HTTP://WIN32DD.MSUICHE.NET/](http://win32dd.msuciche.net/)
- SUICHE, M. & RUFF, N. (2008) SANDMAN PROJECT, [HTTP://SANDMAN.MSUICHE.NET/](http://sandman.msuciche.net/)
- SUTHERLAND, I., EVANS, J., TRYFONAS, T. & BLYTH, A. (2008) ACQUIRING VOLATILE OPERATING SYSTEM DATA TOOLS AND TECHNIQUES. *SIGOPS OPERATING SYSTEMS REVIEW*, 42, 65-73.
- TANENBAUM, A. S. (1999) *STRUCTURED COMPUTER ORGANIZATION*.
- TECHWEB (2008) TECHENCYCLOPEDIA: LEVEL OF ABSTRACTION, [HTTP://WWW.TECHWEB.COM/ENCYCLOPEDIA/DEFINETERM.JHTML?TERM=LEVEL+OF+ABSTRACTION](http://www.techweb.com/encyclopedia/defineterm.jhtml?term=level+of+abstraction)
- TRUECRYPT (2008A) STATISTICS, [HTTP://WWW.TRUECRYPT.ORG/STATISTICS.PHP](http://www.truecrypt.org/statistics.php)
- TRUECRYPT (2008B) TRUECRYPT DOCUMENTATION, [HTTP://WWW.TRUECRYPT.ORG/DOCS/](http://www.truecrypt.org/docs/)
- TRUECRYPT (2008C) TRUECRYPT DOCUMENTATION: HIDDEN OPERATING SYSTEM, [HTTP://WWW.TRUECRYPT.ORG/DOCS/?S=HIDDEN-OPERATING-SYSTEM](http://www.truecrypt.org/docs/?s=hidden-operating-system)
- TRUECRYPT (2008D) TRUECRYPT HOME PAGE, [HTTP://WWW.TRUECRYPT.ORG](http://www.truecrypt.org)
- TRUECRYPT (2008E) TRUECRYPT RESCUE DISK, [HTTP://WWW.TRUECRYPT.ORG/DOCS/RESCUE-DISK.PHP](http://www.truecrypt.org/docs/rescue-disk.php)
- TRUECRYPT (2008F) TRUECRYPT VERSION HISTORY, [HTTP://WWW.TRUECRYPT.ORG/DOCS/?S=VERSION-HISTORY](http://www.truecrypt.org/docs/?s=version-history)
- TURNER, P. (2005) UNIFICATION OF DIGITAL EVIDENCE FROM DISPARATE SOURCES (DIGITAL EVIDENCE BAGS). *DIGITAL INVESTIGATION*, 2, 223-228.

-
- TURNER, P. (2006) SELECTIVE AND INTELLIGENT IMAGING USING DIGITAL EVIDENCE BAGS. *DIGITAL INVESTIGATION*, 3, 59-64.
- TURNER, P. (2007) APPLYING A FORENSIC APPROACH TO INCIDENT RESPONSE, NETWORK INVESTIGATION AND SYSTEM ADMINISTRATION USING DIGITAL EVIDENCE BAGS. *DIGITAL INVESTIGATION*, 4, 30-35.
- UNITED KINGDOM (2000) REGULATION OF INVESTIGATORY POWERS ACT 2000, [HTTP://WWW.OPSI.GOV.UK/ACTS/ACTS2000/20000023.HTM](http://www.opsi.gov.uk/acts/acts2000/20000023.htm)
- VAN DONGEN, W. S. (2007) FORENSIC ARTEFACTS LEFT BY WINDOWS LIVE MESSENGER 8.0. *DIGITAL INVESTIGATION*, 4, 73-87.
- VIDAS, T. (2007) THE ACQUISITION AND ANALYSIS OF RANDOM ACCESS MEMORY. *JOURNAL OF DIGITAL FORENSIC PRACTICE*, 1, 315-323.
- VIDSTROM, A. (2002) PMDUMP, [HTTP://WWW.NTSECURITY.NU/TOOLBOX/PMDUMP/](http://www.ntsecurity.nu/toolbox/pmdump/)
- VIDSTROM, A. (2006A) FORENSIC RAM DUMPING.
- VIDSTROM, A. (2006B) MEMORY DUMPING OVER FIREWIRE - UMA ISSUES, [HTTP://NTSECURITY.NU/ONMYMIND/2006/2006-09-02.HTML](http://ntsecurity.nu/onmymind/2006/2006-09-02.html)
- VMWARE (2008) VMWARE WORKSTATION 5.5 DISK MOUNT UTILITY, [HTTP://WWW.VMWARE.COM/DOWNLOAD/EULA/DISKMOUNT_WS_V55.HTML](http://www.vmware.com/download/eula/diskmount_ws_v55.html)
- VMWARE (2009) VIRTUALIZATION BASICS, [HTTP://WWW.VMWARE.COM/VIRTUALIZATION/](http://www.vmware.com/virtualization/)
- VOLATILE SYSTEMS (2008) THE VOLATILITY FRAMEWORK: VOLATILE MEMORY ARTIFACT EXTRACTION UTILITY FRAMEWORK.
- WAIT, P. (2006) 'LIVE' FORENSICS IS THE FUTURE FOR LAW ENFORCEMENT. GOVERNMENT COMPUTER NEWS [HTTP://WWW.GCN.COM/PRINT/25_22/41502-1.HTML](http://www.gcn.com/print/25_22/41502-1.html) (INTERNET).
- WALKER, J. (2008) ENT - A PSEUDORANDOM NUMBER SEQUENCE TEST PROGRAM, [HTTP://WWW.FOURMILAB.CH/RANDOM/](http://www.fourmilab.ch/random/)
- WALTERS, A. & PETRONI, N. (2007) VOLATOOLS: INTEGRATING VOLATILE MEMORY FORENSICS INTO THE DIGITAL INVESTIGATION PROCESS, [HTTP://WWW.KOMOKU.COM/FORENSICS/BASIC/BH-FED-07-WALTERS-PAPER.PDF](http://www.komoku.com/forensics/basic/bh-fed-07-walters-paper.pdf)
- WILSON, C. (2005) VOLUME SERIAL NUMBERS AND FORMAT VERIFICATION DATE/TIME, [WWW.DIGITAL-DETECTIVE.CO.UK/DOCUMENTS/VOLUME%20SERIAL%20NUMBERS.PDF](http://www.digital-detective.co.uk/documents/volume%20serial%20numbers.pdf)
- WILSON, L. (2006) PROGRAMMATICALLY OBTAIN THE HARD DISK'S SERIAL NUMBER FROM VBA?, [HTTP://BYTES.COM/GROUPS/MS-ACCESS/465109-PROGRAMMATICALLY-OBTAIN-HARD-DISKS-SERIAL-NUMBER-VBA](http://bytes.com/groups/ms-access/465109-programmatically-obtain-hard-disks-serial-number-vba)
- WINMAGIC (2005) WHITE PAPER: DISK ENCRYPTION PRODUCTS, [HTTP://WWW.WINMAGIC.COM/DOWNLOADS/DISKENCRYPTION_WHITEPAPER.PDF](http://www.winmagic.com/downloads/diskencryption_whitepaper.pdf)
- WOLFE, H. (2003) ENCOUNTERING ENCRYPTION. *COMPUTERS AND SECURITY*, 22, 388-391.
- WOLFE, H. B. (2002) ENCOUNTERING ENCRYPTED EVIDENCE (POTENTIAL). *PROCEEDINGS OF THE 4TH CONFERENCE ON INFORMATION TECHNOLOGY CURRICULUM*.
- X-WAYS (2009) X-WAYS FORENSICS, [HTTP://WWW.X-WAYS.NET/](http://www.x-ways.net/)
- ZEIGLER, A. (2008) IE8 AND PRIVACY, [HTTP://BLOGS.MSDN.COM/IE/ARCHIVE/2008/08/25/IE8-AND-PRIVACY.ASPX](http://blogs.msdn.com/ie/archive/2008/08/25/ie8-and-privacy.aspx)
- ZIMMERMANN, P. (1998) PGP USER'S GUIDE. PGP.

APPENDIX A

There are many process models that are used to represent a digital investigation. This appendix summarises them in table form. In this research, the simplest process model from Carrier (2003) is used, which is acquisition, analysis and presentation. Using this model does not consider preparation for performing a digital investigation, since a process model of the digital investigation itself is sought.

Carrier (2002)	Farmer (1999)	Palmer/DFRWS(2001)	Mandia <i>et al</i> (2003)	NIJ (2001)	Reith <i>et al</i> (2002)	Carrier & Spafford (2003)	Baryamureeba & Tushabe (2004)	O' Ciardhuain(2004)	Beebe & Clark (2005)
			Pre-Incident Preparation Detection Initial Response Response Formulation		Identification Preparation Approach <i>Strategy</i>	Operational Readiness Infrastructure Readiness Detection and Notification Confirmation and Authorisation	Operational Readiness Infrastructure Readiness Detection	Awareness Authorisation Planning Notification	Preparation Incident Response
Acquisition	Secure and Isolate Record the scene Search for evidence Collect and package evidence Maintain chain of custody	Identification Preservation Collection	Duplication	Collection	Preservation Collection	Physical Crime Scene Phase Preservation	Physical Crime Scene Investigation (1) Digital Crime Scene Investigation (1)	Search and Identify Collection Transport Storage	Data Collection
Analysis		Examination Analysis	Investigation Security Measure Implementation Recovery	Examination Analysis	Examination Analysis	Survey Documentation Search and collection	Confirmation Submission Digital Crime Scene Investigation (2) (Identification) Authorisation Physical Crime Scene Investigation (2) Digital Crime Scene Investigation(3) Reconstruction	Examination Hypothesis	Data Analysis
Presentation		Presentation	Reporting Follow up	Reporting	Reporting Return Evidence	Reconstruction Presentation Review	Communication Review	Presentation Proof/defence Dissemination	Findings Presentation Incident Close

Table 22: Comparison of process models for digital investigations.
