

Law Enforcement and Digital Evidence

J. Philip Craiger
Assistant Director for Digital Evidence
National Center for Forensic Science &
Department of Engineering Technology
University of Central Florida

Mark Pollitt
President
DigitalEvidencePro

&

Jeff Swauger
National Center for Forensic Science
University of Central Florida

Keywords: Digital evidence, law enforcement, digital forensics, computer forensics, computer crime, cybercrime, encryption, steganography, storage area network, software validation.

To appear in H. Bidgoli (Ed.), Handbook of Information Security. New York: John Wiley & Sons.

Table of Contents

- Digital evidence and digital forensics
- Who should read this chapter?
- Data obfuscation
 - Encryption
 - Encryption ciphers
 - What can a suspect encrypt?
 - Individual file encryption
 - File system encryption
 - How to determine whether files are encrypted
 - Cracking Encryption
 - Ask the Suspect and Social engineering
 - Automated tools
 - Rule-based attacks
 - Dictionary attacks
 - Brute force attacks
 - Commercial and freeware tools
 - Steganography
 - Steganographic demonstration
 - Dealing with steganography
 - Automated methods
 - Manual methods
 - Summary
- Digital forensic tool validation
 - Problems with validation testing
 - Black-box validation testing
 - Test samples
 - Erroneous results
 - Summary
- Forensic countermeasures
 - File wiping
 - Trace evidence locations
 - RAM
 - Swap and hibernation files
 - Unallocated space
 - Summary
- Growth and diversity of digital evidence
 - Cases & investigations
 - Data volume
 - Solutions for data reduction
 - Automated Methods
 - Storage Area Networks
- Diversity of digital devices and media
 - Summary
- A law enforcement view of the future of digital evidence
- Terms
- References

- Additional Readings

Abstract

Criminal investigations are more complex than ever, in part because of the growing use of computers in the conduct of illegal activities. The fact that computer technology has become a linchpin in our society means that criminal investigators must continually adapt their procedures and update their skills to facilitate the proper investigation of computer crimes. In this chapter, we focus on challenges that law enforcement faces with respect to digital evidence. These challenges include data obfuscation techniques, including encryption and steganography; forensic software tool validation; anti-forensic techniques; the exponential growth of data; and new digital devices. For each challenge, we describe potential solutions in line with law enforcements (typically) limited resources.

Digital evidence and digital forensics

One of the byproducts of the growth of information technology has been the proliferation of the ‘computer criminal.’ Forensic evidence at a crime scene that once was limited to physical items and attributes (carpet fibers, tool marks), and biological matter (hair, blood, fingerprints) now often includes *digital evidence*. In 1999, the Scientific Working Group on Digital Evidence (www.swgde.org) defined digital evidence as:

“Information of probative value stored or transmitted in binary form.”

Examples of digital evidence would include common application files (word processing, spreadsheets, etc.), graphical files, audio and video recordings and files, server logs, and application executables.

Like other forensic disciplines, digital forensics has its origins in the legal system. Digital forensics can be described as a technical solution to what is principally a legal problem. This statement is true for all of the forensic sciences. As a result, the law has had an equal, if not greater, influence on the practice of digital forensics than has computer science or engineering. The admissibility of digital forensic processes and products is the ultimate goal.

Law enforcement’s view of digital forensics is driven by the premise that anything done to an item collected as evidence is subject to presentation in court. This simple fact has tremendous impact, not only on law enforcement’s view of digital forensics, but also on the development and regulation of processes. Regardless of the scientific virtue of anything discovered, the value of these discoveries is constrained by the requirements of the legal system.

Who should read this chapter?

We wrote this chapter for semi-technically literate professionals at small or medium-sized law enforcement agencies (or businesses) who are involved in the processing of digital evidence. Agencies of this size make up the majority of law enforcement agencies in the United States.

From our many discussions with personnel from these agencies we have found that the person most often placed in charge of digital forensics examinations was selected because they were the most technically-literate person in the agency. It is atypical for these personnel to have degrees in IT-related fields (e.g., computer science, management information systems, or information technology) or years of experience in computer-related fields.

Furthermore, we did not assume that these agencies have access to digital forensics experts or that they have large budgets for training or technology, which is typical for the majority of small law enforcement agencies with which we are familiar. The descriptions and demonstrations in our chapter were written at a level that a semi-technically literate person should be able to follow without too much trouble. For the more technical

demonstrations, interested readers may wish to refer to the resources listed in our reference section for more information. A very good source on the use of the Linux operating system in digital evidence cases, written specifically for law enforcement, is Grundy (2004). Readers should also refer to other chapters in this volume including Computer Forensics Procedures and Methods, Forensic Computing, Forensic Analysis of Windows Systems, and Forensics Analysis of UNIX systems.

This chapter is not a step-by-step ‘how to’ guide, but rather a primer to create an understanding of the challenges, combined with descriptions and pointers on what is available, in terms of technologies and methods, to assist agencies in dealing with these challenges. Our reference section contains dozens of books and pointers that further elaborate on the solutions we describe in this chapter.

Challenges to Law Enforcement

In this chapter we provide an introduction to several challenges that agencies face during digital evidence investigations. We selected these challenges based on discussions with local, state, and national law enforcement agencies and our own personal experience. Due to space limitations we have not included every conceivable problem that agencies may encounter. Rather we selected a subset of problems that have demonstrated an ability to hinder computer crime investigations. In fact some of the problems may not be widespread at the moment. For instance, encryption is used in a small percentage of digital evidence cases; however, when used it can greatly impede an investigation.

We present these challenges in the following order:

- Data obfuscation
 - Encryption
 - Steganography
- Forensic tool validation
- Forensic countermeasures
- Large quantities of evidence
- Diversity of digital devices and media

For each challenge we describe potential solutions for dealing with the problem. We begin this chapter with the most technical subject first, techniques for obfuscating (hiding) digital files.

Data Obfuscation

Criminals who use computers to conduct illegal activities often attempt to hide the fruits of their crimes, and often hide the very tools used in the commission of the crime (e.g., software tools). Hiding files on a computer can take on many forms, ranging from simply changing a file’s three-letter extension to more technical methods such as file encryption and steganography. Each of these technologies has the potential to severely impede the recovery of digital evidence. We discuss encryption and steganography in turn below and describe potential methods of coping with each.

Encryption

This section provides sufficient detail on encryption so that the semi-technical reader can understand the problems that may be encountered during digital forensic examinations involving encryption. Readers interested in more detailed information on encryption are urged to review chapters in Volume II, Part 3 of the Handbook of Information Security: Encryption Basics, Symmetric-Key Encryption, Hashes and Message Digests, and Public-Key Algorithms. We include several important resources on encryption at the end of this chapter.

In simple terms, encryption takes a digital artifact (text, picture, audio, video, etc.) and transforms it so that it is unreadable. The transformation process requires a cipher -- mathematical encryption algorithm -- and a key. Frequently, the key is a password. The cipher takes the file and key and performs the necessary calculations to make the file unreadable. To decrypt requires reversing the transformation process so that the file is once again readable. This requires knowing which cipher was used, and most importantly, the key that was used in the transformation process.

Encryption is used legitimately and legally by business, industry, governments, military, and individuals, to ensure privacy of information by keeping it hidden from those not authorized to read the information. Unfortunately, it can also be used by criminals to hide the fruits of their crimes.

Encryption Ciphers

There are two major classes of encryption algorithms: symmetric (or secret key), and *asymmetric* (or public key). Symmetric key algorithms use a single key for encryption and decryption. Asymmetric key encryption algorithms use two keys: a public key and a *private* key. The public key is freely shared and is used by senders to encrypt a message that is meant for the recipient only.

Assuming the encryption works as intended, the encrypted message can only be decrypted with the recipient's private key. Often the keys are interchangeable; that is, if the public key encrypts the message, the private key can decrypt it, and if the private key encrypts a message, the public key can decrypt it. Although keys are often interchangeable, this is not required for asymmetric encryption. The private key may also be used in digital signatures as a means of authenticating the source of a message. A "digital signature" encrypted with the sender's private key may be attached to a message. Any message recipient can prove the message is authentic by decrypting the signature using the sender's public key, assuming the sender's private key has not been compromised.

A symmetric cipher (i.e., mathematical formula or algorithm) combines the password and the message and encrypts them into 'ciphertext,' essentially an unreadable scrambled message. To 'decipher' the message (i.e., recover the text) requires knowledge of the cipher.

With public key algorithms, a separate mathematical function is used to create the

public/private key pair. The private key is protected by encrypting it with a symmetric key cipher and a password. Knowledge of the encryption algorithm and the password used to encrypt the private key is required to recover the text.

Symmetric and public key algorithms are not directly comparable because the process underlying the encryption cipher differs dramatically. The current U.S. government standard for symmetric encryption is the Advanced Encryption Standard, which has key lengths of 128, 192, or 256 bits. In contrast, public key algorithms such as the RSA public key algorithm use keys containing 1024, 2048, or 4096 bits. The longer the key length, the more secure the encrypted message is for both symmetric and asymmetric encryption. As of 2004, symmetric key lengths of 128 bits or more are considered secure, as are public key lengths of 1024 bits or more.

Public key encryption is a slow process compared to symmetric key encryption because more computation is required. Because of this, a third form of encryption is sometimes used. This form of encryption is a hybrid technique that combines symmetric and public key encryption. Examples include PGP (Pretty Good Privacy, <http://www.pgpi.org>) and its open source counterpart, GPG (GNU Privacy Guard, <http://www.gnupg.org>). These hybrid algorithms create randomly generated session keys that are used to encrypt messages with a symmetric cipher (DES, Triple-DES, CAST, AES, etc.) The session key is then encrypted with a public key, and the encrypted session key is sent, with the encrypted message to the recipient of the message. The recipient uses his or her private key to decrypt the encrypted session key, which may then be used to decrypt the message. Hybrid methods are attractive because they combine the best of both symmetric and public key encryption algorithms. As an additional measure of security, the recipient's private key is encrypted with a symmetric cipher and requires a password to decrypt. This extra layer of encryption of the private key provides an added layer of security.

What information can be encrypted?

Suspects can encrypt any digital files, from an individual file to an entire hard drive. Regardless of what data is encrypted, it is necessary to know both the cipher used for encryption and the password in order to decrypt the ciphertext.

Individual file encryption

Before investigators can decrypt and analyze encrypted information, the encrypted files must be detected and identified for further analysis. Many digital forensic tools can determine whether a file has been encrypted by evaluating the file's header information. Header information is digital information contained within the beginning of a file that indicates the file type. Unfortunately, this method only works if the file headers have not been modified, and whether the file has a recognizable header.

There is no single, simple, 100 percent accurate way of determining whether an individual file is encrypted, or whether the file merely resembles an encrypted file. In the following example, four of the files in the directory (Figure 1) are encrypted:

- request2.doc: GPG Public-key ASCII encrypted
- football.sch GPG Public-key binary encrypted
- WORMPAPER2.pdf GPG Symmetric key encrypted file
- shoppinglist.txt PGP Symmetric key encrypted file

Additionally the file hotels.doc is a compressed -- using zip compression -- file whose header information was removed. When the header information is removed from a file, applications can no longer correctly determine the file type. It also means that the file is corrupted and most likely cannot be opened by the application that created it.

We ran the UNIX file command against all of the files in this directory. The file command reads a file's header information and compares it against a known list of headers to determine the type of file. As Figure 1 demonstrates, the file command contains enough information to conclude that request2.doc is as an encrypted file of type PGP armored. For the remaining encrypted files, however, the headers do not contain enough information to determine the type of file by using the file command.

Insert Figure 1 Here

Figure 1. UNIX file command run against several files

We replicated this experiment using the same files and a commercial forensics tool. As Figure 2 demonstrates, we achieved the same results (under the column 'File Type').

Insert Figure 2 Here

Figure 2. Results for commercial forensic toolkit

It is impossible to determine whether a file is encrypted merely by 'eyeballing' its contents. As Figures 3-5 demonstrate, the contents of an encrypted file may appear very similar to other files such as binary executables and compressed files (e.g., zipped files). To demonstrate, we created a symmetric key encrypted file (Figure 3), a compressed file (Figure 4), and a binary executable (Figure 5). Clearly, there are no visually identifiable differences that allow an investigator to determine the type of file through a mere visual examination.

Insert Figure 3 Here

Figure 3. Symmetric cipher encrypted file

Insert Figure 4 Here

Figure 4. Compressed file with modified header

Insert Figure 5 Here

Figure 5. Binary executable without header

File System-level Encryption

There exist several applications that can encrypt a volume (partition) or even an entire hard drive. These applications are available for several operating system including Windows, Linux, and Macintosh. These applications work the same as individual file encryption, using a cipher and a key to encrypt data.

Some of these applications can leave visible clues that a suspect is running hard drive encryption. For instance, we found that several Windows-based encryption applications display their icon in the Window's taskbar. Figure 6 shows the icon displayed -- floppy disk with yellow key underneath -- for the BestCrypt encryption application (www.jetico.com). (There is no guarantee that a volume or disk encryption application will place its icon in the Window's taskbar. Therefore, lack of an icon doesn't guarantee that the file system is not encrypted. An investigator can be certain a file system or disk is encrypted if the investigator attempts to access files and is presented with dialog box requesting a password for file access.)

Insert Figure 6 Here

Figure 6. Icon displayed in task bar for BestCrypt encryption program

When the user attempts to access a file on an unmounted encrypted file system, the encryption program will ask the user for the password. If the user does not have the correct password, the encrypted file system remains unmounted and the files cannot be accessed. However, if the encrypted file system is mounted, then the files can be accessed. (Mounting a file system makes the files on the media accessible to the operating system and applications.)

Windows Encrypted File System (EFS)

The Windows 2000, XP, and 2003 Server operating systems have the capability of encrypting volumes using Microsoft's Encrypted File System (EFS). EFS is a hybrid method that uses a symmetric algorithm for data encryption and a public-key for protecting the symmetric key. Access to the public key is protected via a password. While Windows 2000 defaults to 56-bit Digital Encryption Standard (DES), XP and 2003 Server default to the Advanced Encryption Standard (AES) with a 256-bit key, a much more secure cipher than either 56-bit DES or 3DES.

The Encrypted File Systems supports third-party data recovery through the concept of a Data Recovery Agent (DRA). A DRA is a designated authority -- most often a security or

network administrator, but could be anyone -- whose private key is used to protect the data in addition to the primary's public key, i.e., the original data owner's public key.

Consequently, in a networked environment data recovery of an EFS encrypted partition is very possible as long as someone has been designated as a third-party DRA. If a DRA does not exist, then the only person capable of decrypting the data is the primary owner. Should this be the case, other methods are required to gain access to the public-key's password. Some of these methods are described in a later section.

How to determine whether files are encrypted

How does one determine whether a file is encrypted? Unfortunately, this sometimes requires trial-and-error attempts to decrypt the files, using a variety of encryption applications. This is demonstrated below with GPG and PGP encryption applications.

An investigator should run a forensic application that uses header information against all files on a hard drive first to determine if any files are obviously encrypted. As demonstrated previously, accurately determining the type of file can be hit-or-miss depending upon what type of encryption is used. If an investigator has on good authority that the suspect is known to use encryption, then some trial-and-error involving attempts to decrypt the files with a variety of encryption applications may be necessary.

Below we demonstrate this with GPG and PGP encryption applications. (Some applications append their own three letter extension when encrypting files, such as '.gpg' or '.pgp.' A user can remove the extension to remove that clue.)

Figure 7 demonstrates a trial-and-error attempt to decrypt files using GPG. Figure 8 demonstrates this process using PGP. The total experimental results are presented in Table 1.

Insert Figure 7 Here

Figure 7. Attempting to decrypt files with GPG

Insert Figure 8 Here

Figure 8: Attempting to decrypt files with PGP

File	<i>file results (based on header)</i>	Actual File Type	GPG	PGP
hotel.doc	Data	Compressed file	Y	Y
request.doc	MS Word	MS Word	Y	Y
request2.doc	PGP armored	PGP Public-key armored	Y	Y
WORMPAPER.pdf	PDF file	PDF file	Y	Y
WORMPAPER2.pdf	Data	Symmetric key encrypted	Y	Y
shoppinglist.txt	ASCII text	PGP symmetric key encrypted	Y	Y

football.sch	Data	Public-key encrypted	Y	Y
linux	Binary	Binary	Y	Y

Table 1. Experimental Results

Table 1 indicates how PGP and GPG performed in determining whether or not a file was encrypted. The ‘Y’s under the “GPG” and “PGP” headings indicate that the application correctly determined whether or not the file was encrypted. For this simple example, the applications were 100% accurate. GPG and PGP correctly determined that files encrypted with the other applications were encrypted, as well as determining whether the cipher used for encryption was a symmetric or public key cipher.

Real world situations are much more complex than this example; however, this simple demonstration shows the type of experiments one might have to conduct in order to determine which files, if any, are encrypted.

Breaking Encryption

Several techniques that can be used to recover digital evidence are described below. Each of these methods relies on recovering the password used for encryption.

Asking the Suspect and Social Engineering

The simplest and easiest method of overcoming encryption is to ask the suspect for the password(s). This technique can be very effective, particularly if it is used immediately after law enforcement serves a search warrant on the suspect. This is the time when suspects may be psychologically weak, confused, or frightened, and are therefore more psychologically amenable to answering questions. We know of numerous examples of suspects who openly shared passwords, and other relevant information, upon being served with a search warrant.

Another approach that relies on psychology is “social engineering.” Social engineering is a term commonly used to describe how computer criminals gather information by tricking people (e.g., secretaries, network administrators, help-desk personnel, regular users, etc.), into divulging information necessary to break into a computer. In social engineering, the investigator attempts to use knowledge of the suspect to obtain the password, either by direct inquiry or by educated guessing based on information provided by the suspect. For example, suspects can often be induced to reveal private information about them that can be used to guess passwords. For example, often words such as pet’s names, children’s names, football team names, etc. are used as passwords. The effectiveness of social engineering is highly dependent on the interpersonal skills and insight of the investigator. Examination of the suspect’s home and other details about their life can also be useful in searching for information relevant to the suspect’s passwords.

If a suspect refuses to divulge his or her passwords, three other options are available. One is to use “password cracking” software. A second method is to search the suspect’s computer, in the hope that the suspect’s password is located somewhere on the computer.

Finally, investigators may attempt to recover passwords from other accounts used by the suspect, in the hope that the suspect has used the same password on more than one account. Each of these methods is described below.

Automated Tools: Password Crackers

Password cracking is a term that implies an automated method of guessing passwords. There are three commonly employed modes of password guessing: a) heuristic or rule-based attacks, b) dictionary attacks, and c) brute force attacks. These are described below in order of the complexity and amount of time it typically takes to perform the attack.

Rule-based Attacks

Rule-based attacks make use of numerous rules-of-thumb that users often follow when creating passwords. Rule-based attacks can be based on information retrieved through social engineering, as mentioned earlier. The reason these attacks are so effective is that human behavior is often predictable: Users often create passwords that are meaningful to themselves personally because these passwords are easy to remember. Examples include birthdays, anniversaries, names of children, simple keyboard combinations ('qwerty,' 'abc123,' etc.), names of pets, and commonly-used passwords such as the word "password" or the username.

Organizational password policies often mandate that users create passwords composed of upper- and lower-case letters, numbers, and special characters in order to increase the difficulty of guessing passwords. Despite these policies, some users create password combinations that follow a formula that is guessable. For instance, a username followed by a number or special character, such as 'lmcbride1,' or 'lmcbride!' are common.

A rule-based attack generates passwords using a defined set of characters. Thus, guesses for a username 'lmcbride' might include: lmcbride1, lmcbride2, lmcbride3 ... lmcbride*, lmcbride/, and so on. Rule-based attacks work remarkably well because passwords that are created from a truly random set of characters are very difficult to remember. When passwords are difficult to remember, users often write their passwords down, and these passwords may be located through other procedures.

Dictionary Attack

A *dictionary attack* uses a list of words from a dictionary as the basis for guessing passwords. If the suspect uses a word contained in the dictionary, it will be guessed fairly quickly, no matter how long the word: 'Antidisestablishmentarianism' will be guessed more quickly than 'cat' because it comes prior to it in a dictionary. Depending upon the size of the dictionary and the speed of the computer, in many cases, a dictionary attack may run through all the words in the dictionary in less than a minute.

Dozens of dictionaries can be downloaded from the Internet, including specialized dictionaries containing names of: cartoon characters, sports teams, or mythical or fictional characters from TV shows, movies, or literature. Dictionaries are also available in multiple languages. We suggest that law enforcement agencies download several sets

of dictionaries, including some of the largest dictionaries available, and use all of these dictionaries before moving to the more time-consuming brute force attack.

Brute Force Attack

The most time-consuming type of password attack is the brute force attack. A brute force attack looks at combinational possible combinations of letters, numbers, and special characters, and uses them in guessing the password. A password eight characters in length that uses upper and lower case letters (52), numbers (10), and special characters (32), means there are 94^8 or 6,095,689,385,410,816 possible combinations. Brute force attacks on passwords more than eight characters in length are generally unfeasible. Use brute force attacks as a last resort only.

Passwords on Disk

Accessdata's Password Recovery Toolkit (PRTK) is a commercial password cracker (www.accessdata.com) that works for many encryption problems. PRTK has a very interesting password cracking methods that works in concert with Accessdata's Forensic Toolkit (FTK). This method is based on the fact that the user's password may appear somewhere on the suspect's hard drive. For instance, the suspect may have included the password in a document, or an ill-behaved encryption application placed the password in RAM, and was subsequently written to a swap or hibernation file. FTK will extract and index every word on a hard drive and output this list to a text file, which can then be imported into PRTK and used as a dictionary. If the password appears anywhere on the hard drive, in allocated, unallocated, or slack space, then the PTK will break the password.

Break Other Accounts

Humans are creatures of habit, as they saying goes. Most users employ the same password for many different accounts: Why remember 10 passwords for 10 accounts when I only have to remember one! A 'careful' user may use multiple passwords, but to reduce cognitive load, select passwords that fall into a category that is meaningful to the user, for instance, characters from Star Trek, a pet's name, etc. 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 pattern, e.g., ckirk!, mspock!, msulu! (i.e., Star Trek characters.)

Commercial and Freeware Tools

There are commercial and freeware tools that perform password cracking. There are several free password crackers that are very good and widely used, including John the Ripper (www.openwall.com) and Crack ftp.cerias.purdue.edu/pub/tools/unix/pwdutils/crack. One of the best known and efficient applications is L0phtCrack, a commercial tool from @stake (atstake.com). Password dictionaries may be found at several locations on the Internet, and are easily located by using Google to search for "password cracking dictionary."

Steganography

Steganography is a term that means ‘covered writing.’ Steganographic algorithms take the bits that comprise a message and embed these bits within another file. The most common example is hiding text within a graphical image, which is demonstrated below.

There are several steganographic algorithms. One of the more common uses the least significant bit of a byte from the file to be hidden; exchanging that bit with the least significant bit from a byte within the cover medium, or file the data is to be hidden in. On average, no more than 50 percent of the bits from the cover medium (also called the receptacle image) are changed in this process. Because they are the least significant bits, very little of the receptacle will display an obvious change.

For example, in order to hide the letter ‘A,’ we first convert it to the ASCII character 65 in decimal, or ‘1000001’ in binary form. We need eight bytes of data to hide the ASCII value of “A.” If we have the following eight bytes of data, inserting the ASCII character “A” will yield:

```
1110101 → 1110101
1101101 → 1101100
1001100 → 1001100
0110110 → 0110110
0010111 → 0010110
0101000 → 0101000
0000001 → 0000001
```

Here we’ve hidden the character A, which in this case only required changing 2 of the least significant bits in the 8 bytes of data we selected. We can of course recover the hidden text by stripping the least significant bit from each byte and piece back together.

Steganographic demonstration

Various steganographic algorithms and programs will hide digital data within almost any other form of digital data. Evidence can be hidden in audio files, various types of graphical images, text files, and other files. Below we illustrate hiding the text from the Declaration of Independence (9629 characters, a lengthy document) within a NASA graphic image.

Insert Figure 9 Here

Figure 9. Original Image – Buzz.jpg

This is a large image, 1280 x 1024 pixels. We are hiding a large document; therefore, we need a large number of bits in which to exchange the bits to be hidden. The example below uses the freeware steganographic program Outguess to hide the text of the Declaration of Independence and save the new file as ‘hidden.jpg.’ A password was also

used to make it more difficult to extract the contents of the file.

```
# outguess -k 'password' -d dofi.txt buzz.jpg hidden.jpg
Reading buzz.jpg....
JPEG compression quality set to 75
Extracting usable bits: 266613 bits
Correctable message size: -5457 bits, 1610934.89%
Encoded 'dofi.txt': 74152 bits, 9269 bytes
Finding best embedding...
  0: 37202(50.1%)[50.2%], bias 34881(0.94), saved: -15, total: 13.95%
  1: 37126(50.0%)[50.1%], bias 34887(0.94), saved: -6, total: 13.93%
  2: 37143(50.1%)[50.1%], bias 34822(0.94), saved: -8, total: 13.93%
  4: 37277(50.2%)[50.3%], bias 34644(0.93), saved: -25, total: 13.98%
  5: 37059(50.0%)[50.0%], bias 34530(0.93), saved: 2, total: 13.90%
  8: 37043(49.9%)[50.0%], bias 34440(0.93), saved: 4, total: 13.89%
 45: 36987(49.9%)[49.9%], bias 34448(0.93), saved: 11, total: 13.87%
122: 36796(49.6%)[49.6%], bias 34630(0.94), saved: 35, total: 13.80%
135: 36922(49.8%)[49.8%], bias 34458(0.93), saved: 19, total: 13.85%
164: 36678(49.4%)[49.5%], bias 34136(0.93), saved: 49, total: 13.76%
164, 70814: Embedding data: 74152 in 266613
Bits embedded: 74184, changed: 36678(49.4%)[49.5%], bias: 34136, tot:
266499, skip: 192315
Foiling statistics: corrections: 12996, failed: 494, offset: 118.798470
+- 250.581476
Total bits changed: 70814 (change 36678 + bias 34136)
Storing bitmap into data...
Writing hidden.jpg....
```

Note that there is some loss of information because of the transposition of bits, which causes degradation in the quality of the image. However, this is typically not discernable to the human eye. [Figure 10](#) shows the resulting image, which contains the hidden text.

Insert Figure 10 Here

Figure 10. Stegoed Image: hidden.jpg

To demonstrate that the files are distinct, we calculate the MD5 cryptographic hash of each file:

```
# shasum buzz.jpg hidden.jpg
f131e22ca580183d9767f91379074d9d1ec43029  buzz.jpg
75a6fed4bf9e929b096b6ac65e830e902c53bec4  hidden.jpg
```

Note that the two cryptographic hashes of the original and hidden file are different, indicating that the two files have distinct content.

In order to recover the hidden text from the graphic file it is necessary to know the application that was used to hide the image, because different applications may use different algorithms to hide data.

If someone attempts to extract the hidden content but doesn't know the password, the extraction process essentially fails. Below is an example of results obtained when an incorrect password is used.

```
# outguess -k 'mypassword' -r hidden.jpg declaration.txt
Reading hidden.jpg....
Extracting usable bits: 266613 bits
Steg retrieve: seed: 51622, len: 63321
Extracted datalen is too long: 63321 > 33327
```

File Recovery

We recover the steganographic content by supplying the correct password.

```
# outguess -k 'password' -r hidden.jpg declaration.txt
Reading hidden.jpg....
Extracting usable bits: 266613 bits
Steg retrieve: seed: 164, len: 9269
```

We successfully recovered our text. The length of the text is 9269 bytes (characters), which is the same number of bytes that was contained within the original document. We can determine whether anything has changed in the recovered file by calculating a cryptographic hash of the file and comparing it to the hash of the original file.

```
# shasum declaration.txt dofi.txt
ca2c56c043d7da0fea8ae31891bcff0d289034ca declaration.txt
ca2c56c043d7da0fea8ae31891bcff0d289034ca dofi.txt
```

As is obvious from the hashes, the recovered file is identical to the original file.

Coping with Steganography

A major problem for law enforcement with respect to steganography is that the information is “hidden in plain sight.” A document can be embedded in an MP3 file or a graphical image, and if the MP3 is played or the graphical image is viewed by an investigator, it will not be obvious that information is hidden within the medium. A web page may contain dozens or hundreds of graphic images, any of which may or may not contain an embedded message. It is impossible to determine through visual examination whether a graphic file contains vital evidence embedded as hidden data.

The best clue that a suspect has used steganography is often provided by a search for steganography tools on a suspect's computer. If steganographic tools or applications are located, the next task is to determine which files contain embedded information. The first and best option is to ask the suspect at the outset which files contain hidden information. If the suspect refuses to divulge that information, there are two options. First is to use automated tools that can detect, albeit imperfectly, hidden information. Second is to conduct trial-and-error experimentation as described in the prior section on encryption.

Several automated tools are available to evaluate the frequency of bits within a file to determine whether the file has embedded steganographic information. Stegdetect

(www.outguess.com) is a freeware application that calculates statistics on a graphic as a means of determining whether an image contains hidden information. The listing below demonstrates the use of Stegdetect to determine if steganographic content is present in several files:

```
# Stegdetect *.jpg
testimg.jpg : jphide(***)
testimgp.jpg : jphide(***)
testorig.jpg : jphide(***)
testprog.jpg : jphide(***)
travel.jpg : negative
```

In the listing above we used Stegdetect to determine that four of the five files contain steganographic contents, and it includes the name of the application that was most likely used to hide the content. In this example, JPhide, a popular Windows-based steganography program, was identified. Stegdetect correctly identified the files that contain steganographic content, as well correctly determining that travel.jpg did not contain any embedded information.

Various other tools are available that use similar calculations to determine whether a file has hidden information. We have found that these tools are not foolproof; they can miss files with embedded information.

Manual methods of determining steganographic content

A suspect may leave clues as to files with steganographic content. For instance, in the presence of steganography programs, one should conduct a visual search for all graphical files. Two graphical files that appear to be the same visually but have different cryptographic hashes may be evidence of embedded information. The graphic file with the latest creation or modified date and time is more likely to be the file with embedded content.

Steganography programs often use a password to increase the difficulty of extracting the hidden content. As discussed in the section on encryption, there are several avenues one can pursue for recovering passwords. The first and easiest is simply to ask the suspect for any passwords he or she uses. It may come as a surprise how often suspects freely provides passwords when asked.

Digital Forensic Tool Validation

Digital forensic techniques and tools, as with all other forensic disciplines, must meet basic evidentiary and scientific standards to be allowed as evidence in legal proceedings. The requirements for the admissibility of scientific evidence and expert opinion were outlined in the precedent setting U.S. Supreme Court decision in the case of *Daubert vs. Merrell Dow Pharmaceuticals, Inc.*, 509 U.S. 579 (1993). In order to be admissible, evidence or opinion derived from scientific or technical activities must come from methods that are proven to be “scientifically valid.” Scientifically valid techniques are capable of being proven correct through empirical testing. In practice, this means that the tools and techniques used in digital forensics must be validated, and that crime laboratories, including digital forensic labs, should be accredited or otherwise proven to meet such scientific standards. Obviously strict and accurate validation testing of new forensic tools is required if the results from such applications are to be acceptable as evidence in criminal cases.

In the United States, the American Society for Crime Lab Directors/Lab Accreditation Board (ASCLD/LAB: www.asclld-lab.org) is the official body that accredits crime labs. The board has developed numerous standards relating to establishing the validity and acceptability of forensic techniques, tools, and accreditation of individual crime labs. ASCLD/LABs criteria for accreditation consist of standards covering crime lab management and operations, personnel, and physical plant. Each standard is labeled as desirable, important, or essential, depending upon its importance and requirement for meeting ASCLD/LAB specifications. Labs seeking accreditation must meet 100% of the essential criteria, 75% of the important criteria, and 50% of desirable criteria.

With respect to forensic tools and techniques, ASCLD/LAB standard 1.4.2.6 addresses the required scientific validation of procedures used in crime labs:

1.4.2.6 ARE NEW TECHNICAL PROCEDURES SCIENTIFICALLY VALIDATED BEFORE BEING USED IN CASEWORK AND IS THE VALIDATION DOCUMENTATION AVAILABLE FOR REVIEW?

Standard 1.4.2.6 is an essential standard, indicating the importance of scientific validation of tools and techniques. In the context of digital forensics labs this standard requires that software (and hardware) must be validated prior to its use in examinations.

In the context of digital evidence, the Scientific Working Group for Digital Evidence (SWGDE) defines the term validation as: “An evaluation to determine if a tool, technique or procedure functions correctly and as intended” (SWGDE, p. 2). Software validation has long been an important component of software design and development. However, it plays a crucial role in digital forensics because the potential consequence of denying a defendant’s constitutional rights to life and liberty:

“Validation testing is critical to the outcome of the entire examination process. Validation, based on sound scientific principles, is required to demonstrate that examination tools (hardware and software), techniques and procedures are

suitable for their intended purpose. Tools, techniques and procedures should be validated prior to initial use in digital forensic processes. Failure to implement a validation program can have detrimental effects. (SWGDE, p. 2)

Validation Testing Challenges

The National Institute for Standards and Technology's (NIST) Computer Forensics Tool Testing (CFTT: (www.cftt.nist.gov) division is one government entity that formally tests computer forensics software. CFTT performs extremely rigorous scientific tests to validate software tools used in digital forensic examinations. CFTT has and continues to perform testing on numerous computer forensic software applications, and has identified various problems that have been addressed by the software vendors. Unfortunately, the ability of one organization to examine all forensic software products and their variations is limited due to the sheer magnitude of the task. For example, software should be revalidated when, either a major new release occurs or when a patch or update is released to add features or correct previously existing problems. Such retesting is called regression testing. In addition, revalidation should occur when significant changes are made to the operating system on which the software will run. Updates to operating systems have the potential to change the operation of software that was previously found to work correctly on an earlier version of the operating system. Most people have experiences from attempting to run software under an operating system change that was guaranteed to be "backwards compatible" that will make this requirement self-evident.

As mentioned previously, the forensic community cannot rely on a single certifying body to test and evaluate all forensic software, as the sheer pace of change and number of software products is overwhelming. In addition, software tools written by forensic examiners, that are not commercial products, often provide additional functionality examiners find useful. Such software, unless it is widely distributed and used, will not rise to the attention of major validation organizations. Consequently, the onus of validation testing is placed on individual examiners who often have no training in software validation testing. Validation is thus often an ad hoc measure that suffices for the moment. Care must be taken in such testing to insure that sufficient scientific rigor is applied to prevent the invalidation of the tool and the evidence it produces.

Validation Testing Approaches

There are various ways to validate a forensic software tool, some of which are more rigorous than others. The most rigorous involves both testing and a detailed examination of the source code, called a code walkthrough. As its name implies, code walkthroughs require that the actual source code is available, and also requires programmers, software engineers, and managers with technical expertise. In addition to knowledge of software coding techniques and languages, expertise must also include detailed understanding of the task being performed, in this case forensic examination of computer files and hardware (e.g. hard drives). Code walkthroughs may take months or even years to conduct on large software applications. Obviously, such testing will not be possible for all applications of interest.

From the standpoint of a forensic investigator, there is a more practical and informal approach, black box testing. This is where much validation of digital forensic software occurs due to both the complexity and time required for code reviews and due to the fact that commercial companies are usually unwilling to share source code (which is rightly considered valuable proprietary intellectual property). Black box testing is an attractive alternative because it does not require programming and software engineering expertise or source code, while allowing the testing of software in a less expensive and quicker fashion. It does, however, require that the testing be designed by personnel with domain knowledge of digital forensics and computer hardware/software standards (i.e. understanding of the processes to be tested). In black box testing, the software is treated without regard for its internals (code), only the ability of the tool to function correctly and as intended is evaluated. The software code is treated as a “black box” the internal functioning of which is unknown.

Black box testing of forensic tools can be accomplished in several ways. One way is to exercise a tool's capability against a known standard. For instance, say that we wish to test the keyword search capability of hypothetical Tool A. To do this we need a known sample (e.g., a hard drive) that contains known and defined instances of the text for which we will search. Tool A's search function is tested by searching our known sample for instances of the keyword. In this example, we search for the term “coke_buddy” on the hard drive as in previous examples. For completeness and fidelity to the real world, our known instances must include the keyword in common formats such as 16-bit UNICODE and 7-bit ASCII, as both formats are used by the operating system to store data on the hard drive. We then use Tool A to perform a search for the keyword and record the results.

Another accepted method of black box testing is to compare the validation results between more than one tool that performs the same function, i.e. Tools A, B, and C. This method is useful when we do not have a known sample to test, that is, we do not have a validated reference data source. This approach can provide particularly strong evidence of validation if one or more of the tools (A, B, or C) have been validated previously.

This approach allows us to perform validation testing in the absence of a known sample, unlike the testing procedure above that requires the contents of the test data to be known *a priori*. If we run the search and all three tools return the same results, then we have supporting evidence that the software functions as intended. This result is strongest when Tools B and C have both been validated, and Tool A returned the same results. The claim is weaker if the only one of the other tools has been validated. If none of the other tools have been validated, the confidence is the weakest, although the fact three tools created by three separate programming teams returned the same results can be interpreted as triangulating on the results. Care must be taken, however, to exercise the tools over their full range of user selectable parameters and against a number of different data sets or test samples. While one or two tests may show excellent results, there can always be situations where the tool will fail that are unusual enough to have not been tested or addressed by the designers. Some peculiar combination of set-up parameters, operating criteria, or type of files being searched may reveal a hidden error that, while rarely occurring, is sufficient to invalidate the tool.

Test Samples

Selecting or creating test samples is one of the most challenging aspects of validation testing. The test sample should consist of a number of heterogeneous examples that replicate conditions that will be found in the real-world. In addition to common types of data, the test samples must include boundary cases. These boundary cases are conditions or examples of things the tool must be capable of detecting even if they are very uncommonly found in most situations. If the tool correctly reports the results from real-world examples as well as boundary cases, then we can say with some authority that the software functions as expected.

A test sample for validation testing would ideally be a sample prepared by the tester containing a complete set of variables and data to thoroughly exercise the application under test. The advantage of running the tool against a known sample is that the results are known a priori given that the examiner knows what exists on the source media. The disadvantage is that time and effort to create the known sample, which can be extensive, and the potential lack of knowledge about the range of variables that can exist. When preparing a hard disk for testing a tool for text searching, for example, both common formats (16-bit UNICODE and 7-bit ASCII) must be included. If the preparer of the test sample does not know about or forgets to include 16-bit UNICODE examples in the sample, the sample will not exercise the tool sufficiently to evaluate its performance. In the creation of test samples for computer forensic applications, this requires that the sample be developed by experts with extensive knowledge of both computer hardware and operating system standards. Such expertise is required to ensure that the test sample does not overlook important conditions or data which would hamper the validation tests thoroughness.

As mentioned previously, it is not always possible to generate a complete and thorough test sample, and in such a case a variety of real world data sets may be used. A sufficient number of diverse data sets must be used to adequately test the application. Selection should be made from a number of possible operating systems and including a variety of different data types produced by a number of different applications. Careful selection of such data sets can produce a test sample sufficiently varied to give confidence in the validation testing performed. One option is to use test samples consisting of data from previous forensic examinations where the results of the examination are already known. This is preferable to an unknown test sample, and is much less time consuming and difficult than deliberately generating a new test sample. Test samples derived from previous, extensive forensic examinations have the advantage of being known samples that is the examiner has a good idea of what exists on the disk, higher confidence coming from the most thorough of previous investigations. Regardless of how the test sample is selected, care must be taken to ensure the samples exercise the boundary conditions.

Analysis of Results

In the Daubert decision, the court found that in evaluating a scientific technique, known or potential rates of error, and error type should be considered. Two error types are of particular interest, Type I and Type II errors. Type I errors occur when a tool or process falsely identifies a positive result when none is present, i.e. a false positive. Type II errors

occur when a tool or process does not identify results that are actually there, i.e. a false negative. Identification of what type of error is present is of particular importance due to the legal implications of using such results in a court proceeding. Therefore, if the forensic tool does not function as intended and returns erroneous results, it is important to identify the type of error that occurred.

Returning to our example involving the search for the keyword “coke_buddy,” we know that our sample hard drive contains instances of our keyword as we prepared the test drive. Tool A’s performance is evaluated by comparing the tools search results with our expected results. The tool should find all known instances of the keyword, if Tool A’s search function works as expected. Table 2 shows the result of a test where Tool A found all instances of the keyword.

Format	Location (byte offset)	Found
UNICODE	14432	Y
ASCII	212178	Y
ASCII	242966	Y
UNICODE	7663911	Y

Table 2. Search results for Tool A

In this example, Tool A has passed the test by detecting all instances of the keyword and not returning any errors. If, for example, the test results had turned out as in Table X below:

Format	Location (byte offset)	Found
UNICODE	14432	N
ASCII	212178	Y
ASCII	242966	Y
UNICODE	7663911	N

Table 3. Search Results for Tool B

In this instance, Tool A did not find the keyword when present in UNICODE format. This is a Type II error, the return of a false negative. If, for example, the test sample hard drive only contained the keyword in ASCII format, and we recorded the following results:

Format	Location (byte offset)	Found
UNICODE	N/A	Y
ASCII	212178	Y
ASCII	242966	Y
UNICODE	N/A	Y

Table 4. Search Results for Tool C.

In this example, Tool A falsely identified the keyword as being present in a case where it was not, and is an example of a Type I error, a false positive.

Of course there are other combinations of results that can occur, such as if the tool misidentifies the contents of the drive, identifying a UNICODE instance as an ASCII instance. The larger the number of different types of data to be identified, the more convoluted the results can become. The larger the number of test samples, or the larger the number of relevant data in the test sample, the higher the confidence in the results of the test.

In an actual tool validation we would have included several more stringent tests including keywords that overlapped contiguous sectors and keywords that overlapped non contiguous (fragmented) sectors. These two circumstances are typically considered two of the most strenuous tests for a search tool.

In the case of a test failure, it is important to determine, if possible, why the tool did not function as intended. In the example where the tool failed to identify the UNICODE keyword, it could be that the keyword crossed non-contiguous cluster boundaries, i.e., part of the keyword was situated in one cluster, and the remaining part was in another cluster that was not contiguous with the first (which occurs due to file fragmentation). In this instance one would continue testing to determine if any other situations caused a failure to report the correct results. A single failure does not necessarily discredit the use of the software in its entirety. The failure needs to be interpreted in light of the remaining test results. Regardless of the outcome, validation testing requires extensive and thorough documentation, identifying all test conditions, variables used, hardware and associated software used in the test, and all test results. In the case of test failures, the results and conditions that resulted in the error should be especially well documented, and the examiner may wish to consider contacting the company or individual that created the software, and informs them of the anomaly.

A tool doesn't have to meet 100% success to be validated. There are cases where it may be impossible, for various reasons, to achieve perfect results. For instance, the ability of a password cracker to break encryption is determined by many factors, one of which is time (every password is crackable, given enough time). Therefore, in some cases a more realistic goal of less than 100% results may be used.

Summary

As can be seen, while there are numerous methods by which validation testing may be performed, all of them require careful thought, preparation, and documentation to test the application as completely as possible. Documentation is one of the most important activities in validation testing, as it allows others to both understand and to repeat the examiners experiments. Replication of experimental results by other examiners is one of the foremost requirements for acceptance of scientific testing.

Forensic Countermeasures

Our population as a whole is becoming more computer literate, and so are criminals. Criminals who use computers to conduct illegal endeavors are learning techniques that

allow them to cover traces of these criminal acts. The term ‘forensic countermeasures’ and ‘anti-forensics’ have been used to describe the techniques performed on digital media in order to reduce the likelihood that digital evidence will be recovered. Forensic countermeasures include a collection of disparate techniques, such as encryption and steganography (covered previously), file compression, media formatting, file wiping, and even wholesale destruction of media. Criminals may combine these techniques for added effect, for instance, encrypting a document and then wiping the original file from the media. In this section, we will concentrate on the topic of file wiping as forensic countermeasure primarily because several commercially available applications advertise the ability to remove all traces of evidence from magnetic media.

Although forensic countermeasures can be used in criminal endeavors, there are legitimate and legal uses of these techniques. For example, disposing of old computer hardware and media can be problematic because residual sensitive information can remain on media. We are not making a value judgment regarding the uses of these techniques, but rather are merely describing what we know to be true with regard to computer criminal behavior.

File Wiping

A number of commercially available tools promise to remove all traces of “evidence” on a computer. These tools, and their freeware cousins, write a series of characters, 1s, 0s, or random characters over a file several times. This overwriting technique eliminates law enforcement’s opportunity to recover the file’s contents because the contents have been overwritten, destroying all previously written data. This file-overwriting technique is commonly referred to as ‘wiping’ a file. Although these wiped files may be recovered using special techniques such as magnetic force microscopy, the tools required are expensive enough that only the best-funded agencies have access to such equipment.

The following Linux command line will write a series of zeros over every sector on a floppy disk.

```
# dd if=/dev/zero of=/dev/fd0
```

This command will wipe not only the file contents, but also the file system areas (master boot record, FAT, and root directory entry), as well as the unaddressable space at the end of the disk (i.e., disk space that cannot be used by the file system).

Several companies advertise their products’ ability to purge unwanted files (evidence) from magnetic media (floppy disks, hard drive, etc.) We have tested several of these applications in our lab, and none appear to work perfectly without some investigation, and trial-and-error experimentation. Some of these tools are simple command line tools that do a respectable job of overwriting files but fail to wipe the directory entries that contain information such as file names, attributes, date and times, and file size.

Many GUI-based tools are extensively configurable. When we ran these tools with the default settings we were able to recover some information that we surmised would have been wiped. This may be a serious problem for those using GUI-based tools: Users must

understand something about where ‘trace’ evidence is located on a computer, and understand how to correctly configure the tool in order to completely remove trace evidence. Learning how to configure software correctly usually entails reading a manual that is written from a technical point of view that may be difficult to understand for the average user. It also entails technical computer knowledge about where such information is likely located on the disk. Most criminal suspects are like other computer users in that they are averse to reading technical manuals, relatively uneducated about the intricate technical details of computers, and quite happy to work with a tool’s default settings.

A second reason that wiping a file may not remove all traces of the file is that the operating system or many applications may save a file to several different locations on media. From the user’s viewpoint, a single copy of the file seems to exist on the computer; however, there may be several complete copies of the file and several dozen parts of the file scattered throughout the hard drive. Users are unaware of the existence of these ‘extra’ copies. In order to remove all remnants of a file users must know where these remnants may remain on a computer, as well as how to configure a tool correctly to wipe these areas. Below we describe several places from which evidence may be recovered. These include: a) RAM, b) swap file, c) hibernation file, and d) unallocated space..

Trace Evidence in RAM

When a user opens the file in an application, the contents of a file are transferred into RAM. The file’s contents may remain in RAM until the file has been closed and memory used by the file is required by the operating system. This is an important reason not to simply pull the plug on a suspect’s computer, as doing so may destroy critical evidence.

To demonstrate how evidence may remain in RAM even after a file has been wiped from a hard drive, we performed a simple experiment. First we opened an old version of this chapter in a word processor, then closed the file, and used a wiping tool to remove the copy of the chapter from the hard drive. Next we used the utility *dd* to write the contents of RAM to a hard drive (see listing_below). “*dd*” is a general-purpose UNIX utility that can copy files and is useful for creating forensic images. (We used a copy of *dd* that was modified to be able to access RAM and compiled to run under Windows operating system.)

```
C:\>dd if=\\.\PhysicalMemory bs=4k conv=noerror of=e:\RAM.img
Forensic Acquisition Utilities, 1, 0, 0, 1035
dd, 3, 16, 2, 1035
Copyright (C) 2002-2004 George M. Garner Jr.
```

```
Command Line: dd if=\\.\PhysicalMemory bs=4k conv=noerror of=e:\RAM.img
Based on original version developed by Paul Rubin, David MacKenzie, and
Stuart Kemp
Microsoft Windows: Version 5.1 (Build 2600.Professional Service Pack 2)
```

```
19/01/2005 16:53:52 (UTC)
19/01/2005 11:53:52 (local time)
```

```
Current User: TIGER\pc
```

```
Total physical memory reported: 1047472 KB
Copying physical memory...
Physical memory in the range 0x00002000-0x00002000 could not be read.
Physical memory in the range 0x00124000-0x00163000 could not be read.
C:\WINDOWS\system32\dd.exe:
    Stopped reading physical memory:

The parameter is incorrect.
The parameter is incorrect.

Output e:\ram.img (1073082368 bytes)
261983+0 records in
261983+0 records out
```

The “if=\\.\PhysicalMemory” specifies dd to read from RAM, with a block size of 4096 bytes (bs=4096), and to write 0s to the file if part of RAM cannot be read (conv=noerror). We are writing the contents of RAM to a file called RAM.img on a shared drive on a different computer so as not to taint our suspect’s computer.

After creating an image of RAM (RAM.img) we can perform a keyword search on the contents to determine if any of portions of our chapter remained in RAM even after we closed our file. We first extracted the human readable text from RAM.img, and then performed our search for the keyword “**coke_buddy**” which appears several times in this chapter. The listing below demonstrates that we found a large part of our chapter in RAM, even after wiping the file from the hard drive. Our keyword is in bold.

A forensics examiner can access the contents of unallocated space by either: a) making a forensic image of the media and performing a physical analysis on the image, b) or booting the suspect’s computer with a bootable forensic disk and performing a physical analysis at the device level. The latter is typically performed with a Linux bootable disk. Figure 13 shows a search of the physical hard drive for the term **coke_buddy**. Because we are accessing the physical hard drive, we are searching both allocated and unallocated space. This means that our search will normally find instances of the keywords in deleted files as well as in active files; however, because we previously wiped our copy of the chapter from our hard drive, our keyword search should only return results from unallocated space.

Because RAM is volatile, it is important to create an image of the contents of RAM post haste if it is suspected that evidence resides therein. The likelihood of the operating system overwriting a file in RAM increases as time passes. First responders should process the most volatile evidence first, including RAM, followed by less volatile evidence such as floppy disks, CD and DVDs.

Creating a forensic image of RAM involves working with a live system. The very fact that investigators work with a live system means that he or she is changing the suspect media, which is contradictory to good computer forensics practice. However, in some circumstances, e.g., evidence in RAM, the investigator must work outside the scope of these general rules in order to pursue the evidence. Forensic procedures for working with live systems are not as “hard and fast” as those for working on media that has been

powered down and may depend heavily on the individual circumstances of the investigation/case.)

Trace Evidence in the Swap and Hibernation Files

When a computer's RAM is full and the operating system must allocate memory for an application, the operating system makes room in RAM by writing out part of its contents to a temporary scratchpad called a 'swap file.' A swap file is simply dedicated space on a hard drive whose contents are temporary and overwritten as needed. Under Windows operating systems, swap files are usually located in the root directory. These files are named pagefile.sys in the most current versions of Windows operating systems (Windows NT, 2000, XP, and 2003) and win386.swp in older versions of Windows (3/3.1, 95, 98, and ME). Under Linux and UNIX systems, a single partition is devoted to the swap file, and is usually labeled /swap. The swap file is essentially a scratchpad to temporarily hold data, and its contents are overwritten frequently.

Hibernation files are usually found on laptop computers, as opposed to on workstations or servers. The hibernation file holds the contents of RAM when the laptop is placed in hibernation mode. When a laptop is put into hibernation mode, the operating system writes out the contents of RAM to the hard drive and then puts the computer into a suspended state. When the laptop is 'awakened,' the previous contents of RAM are transferred back into RAM from the hibernation file, thus restoring the state of the computer.

Below we illustrate the procedure for searching for a file with the keyword 'coke_buddy' in the hibernation file. The swap and hibernation files are locked while the operating system is running and therefore cannot be easily accessed under a live system. The safest way to access these files is by powering off the computer and rebooting using a forensic-capable boot disk such as Knoppix (<http://www.knoppix.com>) or FIRE (Forensic Incident Response Environment, (<http://www.sourceforge.net/projects/biatchux>)).

```
# strings /mnt/hda1/hiberfil.sys | grep -ian -C 5 'coke_buddy'
```

Hibernation files contain both binary and human-readable text. We run the *hiberfil.sys* file through the UNIX *strings* command to recover the human readable text, and pipe the results to the UNIX *grep* command, a powerful search utility. The flags "-ian" indicate that we want a case insensitive search (-i), to treat the input as text (-a), and display line number associated with the results (-n). The flag '-C 5' indicates that we want five lines of 'context' both before and after each hit. The listing below shows that our keyword was found in the hibernation file.

```
# strings /mnt/hda1/hiberfil.sys | grep -ian -C 3 'coke_buddy'
1048197-x
1048198-knowX
1048199-ard dri
1048200:coke_buddy
1048201- UNICoDE fo
1048202-rmat; howeverJ
1048203-to deter
```

An examiner can search swap and hibernation files for a particular type of file, such as a graphics file, by searching for the headers and footers associated with the particular type of file. For example, a JPG graphic file begins with the hex expression “D8 FF D8” in the header, and ends with “FF D9.” An examiner could recover a JPEG file by extracting the content between the header and footer. (See Craiger (2005) in this volume to see how to accomplish this with the UNIX *dd* command.)

Trace Evidence in Unallocated Space

Hard drive space can be partitioned into two primary types of space: allocated and unallocated space. Allocated space holds active files, i.e., files that can be accessed by the file system and operating system. These files contain a pointer in the file system’s “table of contents” (for example, a FAT root directory, an NTFS master file table, or UNIX EXT2 superblock). When a user deletes a file, the operating system marks the file’s entry in the “table of contents” as reusable. Thereafter, the contents of the deleted file are in unallocated space, i.e., space not reserved for a file. Although the information is still on the disk, the contents of unallocated space are not directly accessible by the normal computer user.

Removal software writes a series of characters over a file, making it difficult to recover its contents; however, many software applications will leave trace evidence; that is, they may leave all or part of the contents of the file that are separate from the location of the original file with which a user (suspect) worked. For example, when a suspect modifies a file, many applications write copies of the file to the hard drive. These copies, called temporary files, are intended to help applications recover the file’s contents when there is an abnormal termination of the application or operating system.

Within Microsoft Word, each time a document is saved through the key combination [CTRL-S], the application writes a temporary file to the hard drive. Figure 11 shows temporary files that were written to the hard drive while we were modifying this chapter. The fact that an application writes a temporary copy of the file to the hard drive is not obvious to the average computer user. For many applications, temporary files are written to the /TEMP or /TMP directory under Windows, although the location where the temporary files are written is customizable by a technically knowledgeable computer user.

When the user closes a file, the application silently removes the temporary files from the hard drive. The temporary files’ contents, however, will remain in unallocated space until overwritten.

Insert Figure 11 Here

Figure 11. Temporary files

A forensics examiner can access the contents of unallocated space by either: a) making a forensic image of the media and performing a physical analysis on the image, b) or

booting the suspect's computer with a bootable forensic disk and performing a physical analysis at the device level. The latter is typically performed with a Linux bootable disk.

Figure 12 shows a search of the physical hard drive for the term 'coke_buddy.' Because we are accessing the physical hard drive, we are searching both allocated and unallocated space. This means that our search will normally find instances of the keywords in deleted files as well as in active files; however, because we previously 'wiped' our copy of the chapter from our hard drive, our keyword search should only return results from unallocated space.

We must access the entire physical partition to search for keywords in unallocated space. If we mount the partition, we will only be able to access files in allocated space. We access the entire physical partition through Linux' device file `/dev/hda1`. (See Craiger, 2005, this volume for more information on Linux file systems.). `/dev/hda1` corresponds to the first partition of the first IDE hard drive, which contains our Windows operating system. We use the `dd` command to read the file, piping it through the `strings` command to extract human-readable strings. We use `grep` to search through all the strings on our physical partition.

```
# dd if=/dev/hda1 | strings | grep -ia -C 2 'coke_buddy'
```

Insert Figure 12 Here

Figure 12. Keyword search results on physical partition

Figure 12 shows the first three instances of our keyword. From this we can conclude that at least one document contained this keyword.

Thus far, we have only searched for the keyword in 7-bit ASCII format, the default for the `grep` utility. Any instance of our keyword in any other format, such as 16-bit UNICODE, will not be found. We must conduct another search and specify that `grep` search for UNICODE (16-bit little endian) instances of our keyword, which we specify with the flags `-e l`. The results appear in the listing below: On our 37 GB hard drive, there were 50 occurrences of the term 'coke_buddy' in UNICODE format.

```
# dd if=/dev/hda1 | strings -e l | grep -ic 'coke_buddy'
71683856+0 records in
50
71683856+0 records in
36702134272 bytes transferred in 1163.48693 seconds (3154494
bytes/sec)
```

Where did all of these occurrences of our keyword come from? The simple answer is from the temporary files. Again, the contents of these files will remain on the media until overwritten, or until the user runs a file wiping software to wipe the contents of unallocated space.

Summary

This section demonstrates that forensic countermeasures such as file wiping may or may not have the intended effect of removing all traces of a file. The less technically sophisticated the user, the less likely it is that they fully understand the file system and operating system characteristics necessary to configure file-wiping software to perform correctly. On the other hand, technically sophisticated users often have the knowledge to remove all traces of a file; thus, it is helpful to obtain background information on the user's technical sophistication before performing a forensic examination.

Many evidence-elimination tools provide the capability to wipe swap files, hibernation files, and unallocated space. In this instance, the techniques described above will not be effective. Nevertheless, the forensic examiner should thoroughly analyze all of the files described above when working with evidence on which evidence removal software may have been used.

Digital Evidence: Growing in Volume and Diversity

The amount of evidence that exists in digital form is growing rapidly. This growth is demonstrated in the following chart, which was presented by the Federal Bureau of Investigation at the 14th INTERPOL Forensic Science Symposium:

<u>FBI CART Examinations</u>	
Caseload:	
	FY '99 - 2084 cases
	FY '00 - 3591 cases
	FY '01 - 5166 cases
	FY '02 - 5924 cases
	FY '03 - 6546 cases
Data Burden:	
	FY '99 - 17 terabytes
	FY '00 - 39 terabytes
	FY '01 - 119 terabytes
	FY '02 - 358 terabytes
	FY '03 - 782 terabytes
Source: FBI Computer Analysis & Response Team (CART)	

Table 6. Digital Evidence Growth

The Computer Analysis Response Team (CART) is the FBI's computer forensic unit and is primarily responsible for conducting forensic examinations of all types of digital hardware and media. The data above represents two metrics of the unit's activity. A "case," as used in this chart, represents a submission of evidence to the unit. A case may be a single floppy disk or several hundred computers and a multi-terabyte database. The

data burden represents the total capacity of digital media examined by CART. While the FBI is somewhat unique as a law enforcement agency, its broad jurisdiction makes it useful as an indicator of investigative activity.

While the number of cases increased threefold in the past 3 years, the volume of data increased by forty-six times during the same period! This is a staggering number, given that it is many times the volume of data in the Library of Congress, the largest library on Earth (Jesdanan, 2004). Given the declining prices of digital storage media and the corresponding increases in sales of storage devices, the volume of digital information that investigators must deal with is likely to continue its meteoric increase.

CNET News published a report on March 5, 2004, which described the over \$12 billion business growing at a rate of more than five percent, while the cost per megabyte of storage fell by 30%. (http://news.com.com/2100-1015_3-5170267.html?tag=fd_nbs_ent). This tremendous increase in data presents a number of problems for law enforcement. Traditionally, law enforcement has seized all storage media, duplicated it, and then conducted their examination of the data on the duplicated copy. One of the first steps in the examination process is to recover latent data such as deleted files, hidden data and fragments from unallocated file space. This process is called *data recovery* and requires processing every byte of any given piece of media. If this methodology continues, the number of pieces of digital media with their increasing size will push budgets, processing capability and physical storage space to their limits. Compounding these problems are the practices of providing the defendant with a copy of the data and retaining the data for the length of the defendant's sentence.

If this methodology continues, the growing amounts and types of digital media will push budgets, processing capability and physical storage space to their limits. Compounding these problems are the practices of providing the defendant with a copy of the data and retaining the data for the length of the defendant's sentence.

Consequences for Law Enforcement

As Table 6 shows, the number of FBI cases has tripled in just five years. This is the result of the increased presence of digital devices at crime scenes combined with a heightened awareness of digital evidence by investigators. The FBI has indicated that digital evidence has spread from a few types of investigations, such as hacking and child pornography, to virtually every investigative classification, including fraud, extortion, homicide, identity theft, and so on.

A simple solution to these problems is not evident, but several factors may reduce the impact of data expansion. The most obvious is that only the specific data needed to prove guilt or innocence need be seized can be further processed to reduce the amount of data which must be examined and analyzed. Forensic examiners call this data reduction.

With this increasing prevalence of digital evidence, it is likely that a majority of all criminal investigations will involve digital evidence, and in fact, we may have already reached this point. The Chief Executive of the British Library estimated in 2002, that each year 250 megabytes of information was created for each person on the planet

(<http://www.researchinformation.info/rispring03data.html>). The growth of digital evidence causes three related problems. First, anyone involved in conducting criminal investigation will need to be able to recognize digital evidence. While this may seem fairly straightforward, it is not. As we will discuss later in this chapter that there is a growing diversity of digital tools and toys that can hold digital evidence, and it will not always be obvious to an investigator that these small digital objects may hold the key to a crime.

The second problem is that investigators must seize and process this evidence. Again, while this may seem straightforward for a single computer with an 80GB hard drive, the problem is much more complicated by what law enforcement now encountering: Crime scenes with several networked computers each of which may contain a 200+GB hard drives, and several hundred DVDs. The average 40 hour forensic examination for an 80GB hard drive will soon require several months (or years) to process all of the digital evidence from new crime scenes. The implications in terms of increased budgets for law enforcement personnel, training, hardware and software are clear.

Finally, the ubiquity of digital evidence will require that every agency will require access to digital forensic examination services. No longer will the small law enforcement agency be able to send their digital evidence to state or federal agencies because these agencies will be overwhelmed with their own case work. Smaller law enforcement agencies must find ways of coping with this requirement.

Solutions for Data Reduction

A simple solution to these problems is not evident, but several factors may reduce the impact of data expansion. The most obvious is that only the specific data needed to prove guilt or innocence need be seized. Seized evidence can be further processed to reduce the amount of data which must be examined and analyzed. Forensic examiners call this data reduction.

There are a number of approaches to data reduction, each of which has advantages and weaknesses. Some of these techniques involve eliminating information based upon factors such as where the data is found in the file system, the type of data, the header information, and whether or not the file matches a database of known digital signatures. Once the data has been reduced on the basis of its intrinsic characteristics, it can more efficiently be searched for the content of the data.

A method currently used for data reduction involves performing a hash analysis against digital evidence. A cryptographic one-way hash (or “hash” for short) is essentially a digital fingerprint: a very large number that uniquely identifies the content of a digital file. Figure 13 displays the hashes for several files. On the left are the names of the files, and on the right, under the heading “Hash” are the 128-bit (MD5) hashes. A hash is uniquely determined by the contents of a file. Therefore, two files with different name but the exact same contents will produce the same hash. This is demonstrated in Figure 13 as ‘orlando.txt’ is a copy of ‘forensic.pdf,’ but simply renamed. Note that they result in the same hash.

Figure 13. MD5 Cryptographic one-way hashes

After hashing the file 'CET4932.doc' we added a single space and then attempted to verify the file. As Figure 13 shows an error was generated, indicating the contents of the file have changed.

The larger the number of bits produced by a cryptographic hash algorithm, the smaller the likelihood of a collision. A collision occurs when two files with different contents result in the same hash. Even with the 128-bit MD5 hash the likelihood of a collision are astronomically small, although some researchers have been able to demonstrate the possibility. In practice we have never observed a collision. Nevertheless, to further reduce the possibility of collisions NIST developed the SHA-1 (secure hash algorithm revision 1) hash algorithm that comes in 160-, 192, and 256-bit versions. The difference in the amount of time required to calculate an MD5 versus SHA-1 is negligible with fast computers, consequently, we suggest using the 160-bit SHA-1 as a minimum.

NIST produces a set of hash sets called the National Software Reference Library that contains hashes for approximately 7 million files as of 2004 (www.nsrll.nist.gov).

Files in a hash set typically fall into one of two categories. *Known* files are known to be "ok," and can typically be ignored, such as system files such as win.exe, explore.exe, etc. *Notable* files are suspicious files that are flagged for further scrutiny; files that have been identified as illegal or inappropriate, such as hacking tools, pictures of child pornography, and so on. A hash analysis automates the process of distinguishing between files that can be ignored while identifying the files known to be of possible evidentiary value. Once the known files have been identified then these files can be filtered. Filtering out the known files may reduce the number of files the investigator must evaluate by half or more. NSRL contains MD5 as well as SHA-1 hashes for each file.

Once the data reduction process has been completed, individual data objects could be digitally signed to ensure reliability, and only contentious objects retained for court purposes. An additional approach is to make these data objects available virtually to reduce the number of physical copies needed for review and testimony.

Using Technology to Cope

The FBI's Regional Computer Forensic Laboratory in Dallas, Texas, developed a number of innovative techniques in order to deal with the massive amounts of data seized as a result of the September 11, 2001, attacks.

One of the first techniques was to utilize commercially available network attached storage devices (NAS). At first, these devices were used to store file system images from multiple computers that were being imaged simultaneously. It did not take long to realize that the bandwidth of these devices was not up to the task, and imaging took longer than with dedicated devices. Another technique was to use these NAS devices as repository

for logical copies of multiple computers. In this way, a single examination process such as a string search could be performed against multiple evidence items simultaneously. Since the processing was being done by examiner workstations that were connected over a network, bandwidth was still an issue. Another issue was that the NAS's internal operating systems did not always deal with file modification times, access times, and creation (MAC) times in a way consistent with the operating system used in the original evidence. This presented some serious issues with analytical reliability.

The primary problems with this approach can be categorized as either bandwidth issues or data reliability issues.

Despite these issues, the use of mass storage was crucial to the rapid analysis of the 9-11 data and showed tremendous promise for future improvements in forensic examination techniques. To solve the bandwidth problem, the RCFL developed the use of fiber channel architecture to connect the examination and imaging machines to the mass storage device. Utilizing this technology, bandwidth was increased from network speeds to hardware speeds. This went a long way toward speeding up the process, but in order to allow multiple examiners to use this system concurrently, it required very expensive fiber channel switches and huge quantities of storage. To support up to 24 examiners simultaneously while processing terabytes of data, few other options existed.

The mass storage solution developed by the RCFL utilized a commercially available Storage Attached Network solution (SAN). This solution not only allowed for on-the-fly configuration of virtual drives, rapid access and redundancy, but also for central management and backup. The combination of the two technologies has proven to be revolutionary.

An additional benefit to the SAN/fiber channel solution has been the ability to create virtual drives for investigator/attorney review. A drive containing the examination product can be created on the SAN, permissions can be set to prevent write access, and the drive made available over a network, where access is controlled and logged. The review of examination results over a network was pioneered by the Colorado Regional Computer Forensic Laboratory, and was further refined by the Dallas laboratory.

One might well ask why we should be interested in this very high-end process that is far more expensive than many organizations can afford. Experience has shown that the problems faced by very large Federal entities such as the FBI, IRS, and Department of Defense push the envelope of technology. The solutions developed by these organizations often can be adapted, in less expensive ways, to other organizations. Because today's "bleeding edge" technology becomes tomorrow's everyday tools, it is useful to recognize the lessons being learned every day by these large agencies. History has shown that the problems faced by these agencies tend to migrate downward and become problems for smaller organizations.

The Explosion of Diverse Digital Media

Volume of data is not the only growing challenge facing forensic investigators. The increasing diversity of storage objects and data formats can also present formidable

problems to law enforcement. These challenges include recognizing new media, obtaining technology to read new objects and formats, and developing the expertise to forensically examine each new format.

Technology product lifecycles are short, but criminal investigations and prosecutions often take a number of years. Likewise, law enforcement training cycles are usually measured in years. As a result, investigators and forensic examiners must have the training and equipment to deal with several generations of technology.

As this chapter is being written, the epitaph for floppy disks is also being written (Niesse, 2004); however, there are likely millions of floppy disks sitting in evidence rooms around the world that investigators will need to access and examine for years to come. Likewise, large amounts of data storage are being added to an increasing variety of everyday objects. This presents an increase in the number of sites in which probative evidence may be found and often leads to multiple redundant sources of the same data. Because redundant/related evidence may be present in multiple locations such as desktop computers, laptop computers, phones, and PDAs, there are advantages to adopting a “case focused” examination of all collected evidence, as opposed to a “media-centric” examination in which each piece of media is examined separately.

Along with the proliferation of devices incorporating digital storage capacity, the variety of storage formats has increased. An example of this is found in the portable music devices that contain miniature hard drives and flash memory. While sharing a common purpose, each supports a variety of storage formats, many of which are proprietary.

From a forensic perspective, the examiner needs to understand each of these formats and must have hardware and software capable of examining each type. The challenges presented by the growing myriad of digital devices are only exacerbated by the fact that these devices now hold a quantity of information that would have been unimaginable only a few years ago.

While it may be obvious that a desktop computer’s hard drive, laptop, CD, or floppy disk might hold evidence, there may be sources of digital evidence that may be overlooked by the first responder who has not been trained to identify digital devices. Common place devices that now hold digital files are ubiquitous. A new version of the Swiss Army Knife includes a USB data storage that can hold 256MB of digital files. Similarly, there are several watches and pens that hold 256MB of digital files. It is not clear from simple observation that these simple, common devices have a dual purpose.

The simplest and most cost-effective solution for the problem of recognizing digital storage devices is continual, mandatory training. First responders and investigators must be trained to perform a thorough investigation of all elements at a crime scene, no matter how innocuous a device might seem. Our preferred solution to this problem would be a set of courses developed and maintained by, or overseen by, a Federal agency and provided in an online format such as a series of web-based training courses. Law enforcement budgets have always been tight. Online courses would obviate the need for travel which is always costly.

The growth in size and diversity of data storage is a continuing problem that will not likely disappear soon. The challenges associated with this growth are likely to become more even problematic over time, as digital data becomes more ingrained into the fabric of everyday life. These challenges will necessitate organizational, training, financial, and operational evolution if law enforcement is to provide competent and timely service in the coming years.

A Law Enforcement View of the Future of Digital Evidence

The volume and pervasiveness of digital evidence will force law enforcement to adapt in a wide variety of ways. The current volume overwhelms the currently deployed resources, as volume increases substantial new resources will be needed. Additional people with significantly higher levels of education and training will be required to manage and process the growing amounts of digital evidence. New hardware and software tools will be needed to more efficiently process the increased volume of evidence. But the increase in pervasiveness will affect a much broader range of issues in law enforcement.

Today, digital evidence is collected in only a fraction of cases. As more of our lives are spent and recorded online, law enforcement will have to provide for the collection and preservation of digital evidence in virtually every case. As more and more of law enforcement's activities are digitized, from computer aided dispatch systems, digital booking systems and even digital video cameras in patrol cars, policies and procedures will need to be adapted to ensure the integrity of these evidentiary records. In fact, the volume of internally and externally generated information will require agencies to not merely automate, but rather to develop specialized law enforcement enterprise architectures. While law enforcement has been evolving from "high touch" to "high tech" that transition must go from bureaucratic sloth to Internet time.

The hardware and software tools used by law enforcement practitioners will have to evolve along with the available technology, or rather beyond it. Keeping up with current technology trends is essentially impossible. Technology diffusion in society can take years, and law enforcement, being conservative in nature, is typically slow to embrace the latest and greatest in technology. Although public agencies may not be able to "get ahead of the curve," the creative use of cutting edge technologies can provide significant leverage. For instance, there is a growing trend, exemplified at the Dallas Regional Computer Forensic Labs, to the use of aggregated storage coupled with very fast input/output channels to create storage area networks (SAN's). This is an example of leveraging human and technical resources in ways that individual agencies cannot.

Forensic software will have to evolve as well. Currently, virtually all of the forensic software focuses on the documentation and data recovery aspects of the examination. What few tools, such as filters for header information or text string search tools, are in use are designed with the notion of answering the question: "Is it there?" rather than "What do we know about the information contained in this digital evidence." Consequently, more research is needed that assists law enforcement deal with the deluge of information they are facing. One of the current research trends is to use data mining techniques to assist law enforcement in identifying evidence from terabytes of data. Data

mining uses statistical, clustering, or artificial intelligence techniques to reduce large volumes of data to a manageable size, by identifying patterns among the individual data elements.

Another change that will significantly impact law enforcement is the increase of network-connected devices. As our phones, computers, entertainment devices and household appliances become connected, we will have many more sources of digital evidence to integrate into a given investigation. Further, networked communications, which has expanded from text e-mail into video, audio, and rich text forms easily and quickly sent and received from a wide variety of devices, will require the largely “static” evidence focus to expand substantially into the dynamic collection of data in transit. This change will have both technical and legal implications. Law enforcement’s technical capability to collect, preserve, process, and analyze intercepted dynamic data is generally primitive and very labor intensive. New tools and processes will have to be developed if the large proportion of data is transient. The current legal schema for collecting communications evidence, despite being updated in 2001 with the PATRIOT Act, essentially is modeled on a 1960’s notion of communication. The PATRIOT Act’s mechanics are so onerous that it is used only in situations which can justify the huge overhead of the application process, the live monitoring and minimization, court reporting and notice requirements. If this process is not streamlined, then a very high proportion of the available evidence will not be collected.

Terms

Code walkthrough – analysis of source code by programmers and managers for quality assurance purposes.

Data recovery -- Finding latent information and restoring its context

Data reduction -- Eliminating data that is not significant

Digital evidence – Information of probative value stored or transmitted in digital form.

Duplicate digital evidence – An accurate digital reproduction of all data objects contained on an original physical item (see forensic image)

Forensic image – An exact, bit-for-bit copy of media.

Hash – Also known as a message digest, cryptographic hash, or one-way hash. A hash is a hex value, typically 128- or 160-bit, that is unique to the contents of a file.

MD5 hash – A 128-bit cryptographic hash algorithm created Ron Rivest of MIT.

Metadata –A file's metadata consists of all information about the file excluding its contents: File name, size, MAC times, starting cluster, permissions, attributes, etc.

Original data evidence – Physical items and the data objects associated with such items at the time of acquisition or seizure.

Physical evidence – Items on which data objects of information may be stored and/or through which data objects are transferred.

Regression testing – Testing performed to find bugs in software applications or modules after new code is added.

SHA-1 hash – the Secure Hash Algorithm version 1, a 160-bit cryptographic hash developed by the National Institute for Science and Technology.

Unallocated space - the clusters not in use by a file. Where deleted files reside.

References

- Craiger, J.P. (2005). Computer forensics procedures and methods. In H. Bidgoli (Ed.), Handbook of Information Security. New York: John Wiley & Sons.
- Grundy, B. (2004). The Law Enforcement and Forensic Examiner Introduction to Linux: A Beginner's Guide. Available at www.linux-forensics.com
- National Institute of Justice. (2002). Electronic Crime Scene Investigation: A Guide for First Responders, NCJ 187736, 2001. <http://www.ncjrs.org/pdffiles1/nij/187736.pdf>
- INTERPOL. (2004). "Proceedings of the 14th INTERPOL Forensic Science Symposium", Lyon, France.
- Jesdanun, Anick. " *Mars Internet downloads exceed volume of Library of Congress* ", Detroit News, January 10, 2004. available at:
<http://www.detnews.com/2004/technology/0401/12/technology-31652.htm>
- National Institute of Justice, Forensic Examination of Digital Evidence: A Guide for Law Enforcement, NCJ 199408, 2004 .
<http://puborder.ncjrs.org/Content/ItemDetails.asp?strItem=NCJ+199408&intCounter=1>
- Federal Guidelines for Searching and Seizing Computers, U.S. Department of Justice.
<http://www.usdoj.gov/criminal/cybercrime/searching.html>
- Niese, Mark, "Floppy Disk Becoming Relic of the Past", Tallahassee Democrat, September 6, 2004. available at
<http://www.tallahassee.com/mld/tallahassee/business/9595612.htm>
- Noblett, M.G., Pollitt, M.M., & Presley, L.A. (October, 2000). Recovering and examining computer forensic evidence. Forensic Science Communications. Volume 2 Number 4.
- Brezinski, D., & Killalea, T. (2001). RFC 3227: Guidelines for Evidence Collection and Archiving. <ftp://ftp.isi.edu/in-notes/rfc3227.txt>.
- Special Working Group on Digital Evidence. (April 2000). Digital Evidence: Standards and Principles. Forensic Science Communications. Volume 2 Number 2.
- Special Working Group on Digital Evidence (SWGDE). www.swgde.org

Further Reading

- Britz, M.T., (2003). Computer Forensics and Cyber Crime: An Introduction. Prentice Hall.
- Caloyannides, M.A. (2001). Computer Forensics and Privacy. Artech House Publishers.
- Carvey, H. (2004). Windows Forensics and Incident Recovery. Addison-Wesley Professional.
- Carrier, B. (2005). File System Forensic Analysis. New York: Addison-Wesley.
- Casey, E. (2001). Handbook of Computer Crime Investigation: Forensic Tools & Technology. Academic Press.
- Casey, E. (2003). Practical approaches to recovering encrypted digital evidence. International Journal of Digital Evidence, 3, 1-26.
- Casey, E. (2004). Digital Evidence and Computer Crime. New York: Academic Press.
- Cole, E. (2003). Hiding in Plain Sight : Steganography and the Art of Covert Communication. New York: John Wiley & Sons.
- Craiger, J.P. (2005). Recovering digital evidence from Linux Systems. To appear in S. Sheno (Ed.), Advances in Digital Forensics. International Federation of Information Professionals.
- Farmer, D., & Venema, W. (2005). Forensic Discovery. New York: Addison Wesley Professional.
- Hinder, D.L., & Tittel, E. (2002). Scene of the Cybercrime : Computer Forensics Handbook. Syngress.
- Johnson, N.F., & Jajodia, S. (1998). Steganalysis of images created using current steganography software. Lecture Notes in Computer Science, 1525, pp. 273-289.
- Jones, K., Shema, M., & Johnson, B. (2002). Anti-Hacker Tool Kit. : McGraw-Hill Osborne Media.
- Kruse, W.G., & Heiser, J.G. (2001). Computer Forensics : Incident Response Essentials. Addison-Wesley Professional
- Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A. (1996) Handbook of Applied Cryptography. New York. CRC Press.
- Mohay, G., Anderson, A., Collie, B. Olivier, R. (2003). Computer and Intrusion Forensics. Artech House.

Network Associates, Inc. (1999). Introduction to Cryptography. Network Associates Press. (Download from: <http://www.pgp.com/resources/whitepapers>)

Parker, D. (1998). Fighting Computer Crime : A New Framework for Protecting Information. Wiley

Proise, C., Mandia, K., & Pepe, M. (2003). Incident Response and Computer Forensics. McGraw-Hill Osborne Media

Sammes, T., & Jenkinson, B. (2000). Forensic Computing: A Practitioner's Guide. Springer-Verlag.

Schneier, B. (1995). Applied Cryptography. New York. John Wiley & Sons.

Vacca, J.R. (2002). Computer Forensics: Computer Crime Scene Investigation. Charles River Media.

Wayner, P. (2002). Disappearing Cryptography Information Hiding: Steganography and Watermarking. San Francisco: Morgan Kaufmann.