Chapter 14

DETECTING DATA CONCEALMENT PROGRAMS USING PASSIVE FILE SYSTEM ANALYSIS

M. Davis, R. Kennedy, K. Pyles, A. Strickler and S. Shenoi

Abstract Individuals who wish to avoid leaving evidence on computers and networks often use programs that conceal data from conventional digital forensic tools. This paper discusses the application of passive file system analysis techniques to detect trace evidence left by data concealment programs. In addition, it describes the design and operation of Seraph, a tool that determines whether certain encryption, steganography and erasing programs were used to hide or destroy data.

Keywords: Data concealment programs, trace evidence, program detection

1. Introduction

Encryption, steganography and erasing tools are increasingly used by malicious individuals to hinder forensic investigations [14]. The 2005 indictment of former *Newsday* CEO Robert Johnson in U.S. District Court [22, 23] exemplifies the importance of detecting the presence of data concealment programs on seized media. Johnson was charged with the receipt and possession of child pornography and obstruction of justice for destroying digital evidence. According to the indictment, he erased thousands of illegal pornographic images using Evidence Eliminator, an erasing program that was found on his computer.

Current methods for detecting data concealment programs are resource intensive and time consuming. In particular, they attempt to discover hidden data without first verifying whether or not data is actually concealed [5, 11, 16]. Forensic tools, e.g., FTK, employ known file filters to alert forensic investigators to the presence of certain utilities on a seized computer [16]. However, they do not provide information if a specific utility was installed, executed and subsequently uninstalled. Obviously, such evidence could be very valuable in legal proceedings.

Forensic investigators need tools to identify the specific programs and techniques used to hide or erase data on a seized computer. Having made this determination, attempts can be made to recover the data by exploiting known vulnerabilities in the programs and their data concealment techniques. This paper discusses the use of passive file system analysis techniques to detect trace evidence left by encryption, steganography and erasing programs. In addition, it describes the design and operation of Seraph, a tool that assists forensic investigators in identifying the programs used to hide or destroy data.

2. Data Concealment Programs

This section discusses data concealment involving the use of encryption, steganography and erasing tools.

2.1 Encryption Tools

Encryption tools pose unique challenges for law enforcement [19]. Such tools are increasingly used to hide legal – and illegal – information [3]. Some encryption programs, e.g., BestCrypt [21] and Cryptainer [6], create a secure container or vault on an allocated area of a hard drive. Files stored in this container are accessed via a key that is usually a password known to the user.

Another approach is to encrypt each file individually. Tools such as Folder Lock [17] and Kryptel [13] allow users to choose from a list of popular encryption algorithms and enable them to hide the encrypted files using advanced file manipulation techniques. Other software suites, e.g., Microsoft Office, provide application-level encryption, allowing users to password-protect sensitive information using various techniques, e.g., RC4-based encryption.

AccessData's Password Recovery Toolkit (PRTK) is often used by law enforcement agents to recover passwords and access evidence stored in encrypted files [1, 14, 16]. PRTK employs three methods for recovering passwords: algorithm attacks, key space attacks and dictionary attacks. Files encrypted with application-based encryption are usually easy to break with PRTK, but those using file system or vault encryption are more difficult. Using PRTK is very time consuming. To address this problem, AccessData has created a companion utility, Distributed Network Attack (DNA), that engages multiple computers on a network to break encryption. However, smaller law enforcement agencies do not have the network resources to use DNA effectively.

The process of breaking encryption may be expedited by first identifying the specific encryption program that was used. Software utilities such as Find Protected [2] and Encrypted File Search [8] find files encrypted with specific protocols and programs. However, these utilities examine every file on a system to detect the presence of encrypted data; this is a very time-consuming process. A better approach is to identify the specific software tool used for encryption, and exploit its vulnerabilities to decrypt data.

2.2 Steganography Tools

Steganography is the process by which a message is hidden so that only the sender and the intended recipient know of its existence. Steganography tools typically employ a graphic file (e.g., bitmap or jpeg file) as a carrier for hidden data. A variety of algorithms and techniques can be used to hide data within a carrier file without making noticeable changes to the file [5, 24].

Programs that implement steganography are widely available and are increasingly used to conceal evidence of illegal activities. Several software tools are available for analyzing graphic files for the presence of steganography [5, 11, 12, 25]. WetStone's Stego Suite [25], one of the premier steganalysis tools, analyzes hard drives for evidence of known steganography algorithms. However, Stego Suite (and other tools) analyze all the files on a suspect hard drive, which is a resource-intensive operation.

2.3 Erasing Tools

Erasing tools are designed to destroy data, ideally rendering it unreadable by digital forensic tools. One popular erasing tool is Internet Eraser Pro [9], which eliminates all traces of Internet use; it deletes cookies, clears the history and wipes temporary Internet files. Such tools are often used on shared computers, for example, to hide evidence of visits to pornographic websites.

Erasing tools, e.g., Evidence Eliminator [18], may be used to wipe selected files or entire drives. Indeed, it is difficult, if not impossible, to recover evidence after such tools have been used.

Techniques employed by erasing tools vary widely [4]. Some utilities overwrite the targeted files with random data and then delete them in the usual fashion. Others write random data and then zero the disk space. Still others zero the drive in multiple passes. Many tools only make one or two passes when wiping a drive while others surpass even the U.S. Department of Defense's erasing standard [7]. Since some tools

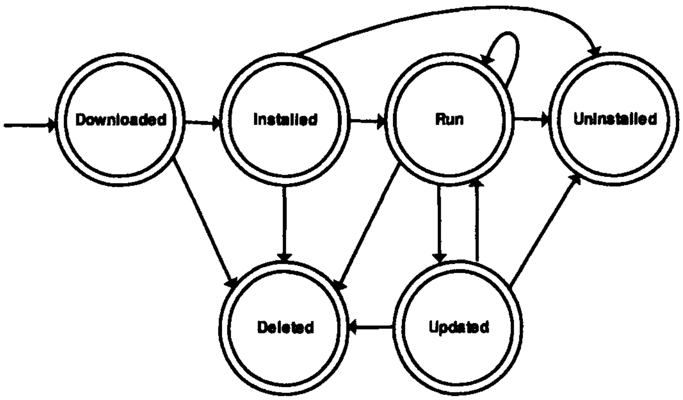


Figure 1. Program lifecycle.

employ weaker data destruction protocols than others, it is important to know the program or at least the protocol used to erase data [4].

The discovery and identification of erasing tools used on a computer can be extremely useful in criminal investigations. As in the Johnson case [22, 23], the presence of an erasing tool could indicate that incriminating evidence was destroyed and possibly lead to an obstruction of justice charge. Many erasing tools leave trace evidence of their activities [4]. Therefore, it may be possible to determine if an erasing tool was installed, and whether or not it was executed.

3. Program Lifecycle

The lifecycle of a software program follows a deterministic path (Figure 1). Information obtained by analyzing the lifecycles of data concealment tools can be extremely useful in forensic investigations. This is because the tools often leave trace evidence during various phases of their lifecycles, e.g., when they are installed, executed or uninstalled.

For our purposes, a program's lifecycle begins when it is downloaded from a network or a physical medium (Figure 1). Many programs include installation packages that copy the files to the system and change registry settings and configuration files. However, some programs must be copied manually by the user. Both these situations cause a program to move to the installed state.

After a program is installed, it may move to the executed state (i.e., when it is executed). However, a program can also be uninstalled without being executed.

After being executed, a program may be updated, in which case it moves to the updated state. The updated state is similar to the installed state, with the exception that the previous version of the program was executed. A program in the updated state may move to the executed state or the uninstalled state.

A program moves to the uninstalled state when a utility is used to remove it from the system. The lifecycle of a program ends when it moves to the uninstalled state.

A program in the downloaded, installed, executed or updated states may move to the deleted state. A program is in the deleted state when it is manually removed from the system, whether or not an uninstallation package exists.

Other scenarios are possible that might alter the program lifecycle. For example, a corrupted program may require the reinstallation of a new version of the program. Most of these scenarios can be treated as the start of the lifecycle of a new program.

In general, the lifecycle of a program could halt in any of the aforementioned states. The state of a program can be determined by examining the evidence left on the system.

4. Potential Evidence

Practically every program generates information and leaves trace evidence during each stage of its lifecycle. Understanding the subtle changes to a file system produced by a specific program (e.g., an erasing tool) during its lifecycle and locating this trace evidence on a seized computer can enable an investigator to determine that the program was installed, executed or uninstalled on the computer.

Downloaded State: The primary source of evidence in this state is the actual downloaded program. The downloaded program is typically a compressed file (e.g., ZIP), or an executable installation package file. Also, the downloading process itself may leave trace evidence in the Internet history and log files.

Installed State: After a program has entered the installed state, files may be added to the hard drive. If the program was copied to a directory from a compressed file or downloaded directly, the only evidence is the program itself. If an installation package was used, evidence of the installation often exists in the registry settings and entries in shared configuration files [10].

Executed State: Once a program has entered the executed state, several traces of its activity can be found. During its first execution the program may ask the user for configuration information and create a file

```
11/06/2005  12:04 PM            95,028 VISIO.EXE-1F9B2047.pf
11/01/2005  01:49 PM            36,210 VPC32.EXE-29593AFF.pf
11/01/2005  01:49 PM            21,570 VPDN_LU.EXE-1D1611C8.pf
10/28/2005  02:15 PM            30,922 WCESMGR.EXE-2FB86E92.pf
11/02/2005  06:23 PM            53,510 WINLOGON.EXE-32C57D49.pf
11/03/2005  03:28 PM            90,390 WINRAR.EXE-39C6DAD9.pf
11/03/2005  03:57 PM            67,426 WINWORD.EXE-37F6AE09.pf
```

Figure 2. Windows XP Prefetch entries.

or add entries to the Windows registry. Some encryption programs create a container for the encrypted data. The presence of such a container would indicate that an encryption program was executed. A program could also create user files, e.g., Microsoft Word (`.doc`) files.

Another example is the Microsoft Windows XP Prefetch directory. This directory, located at `%Windir%\Prefetch`, contains information about executables and where they are stored on disk (Figure 2). The Prefetch capability, which is turned on by default, enables Windows XP to access files more efficiently. A program has an entry in the Prefetch directory only after it has been executed [20].

Trace evidence added to a file system when a program enters the executed state helps determine that a program was actually executed. This is more useful than knowing that the program was installed (and possibly never used).

Uninstalled State: A program in the uninstalled state has been removed from the system using an installation program or a separate utility. Little direct evidence remains as program files and associated folders are removed from the disk. However, shared Windows configuration files, e.g., `win.ini` and `system.ini`, as well as the Windows registry may still contain traces of the program. Also, the Windows Prefetch entry is typically retained.

Deleted State: Since the deleted state requires that the user manually delete the program and associated files, it is likely that most, if not all, the program files are removed. However, as with the uninstalled state, it is possible to find traces of the program in shared configuration files, registry entries and the Windows Prefetch directory. Users who manually delete files also tend to miss shared files, e.g., dynamic link libraries (DLLs) in the `%Windir%\System` directory.

Updated State: Evidence retained in the updated state is similar to that left in the installed and executed states. If a drive is seized when a program is in the updated state, evidence that the program was executed

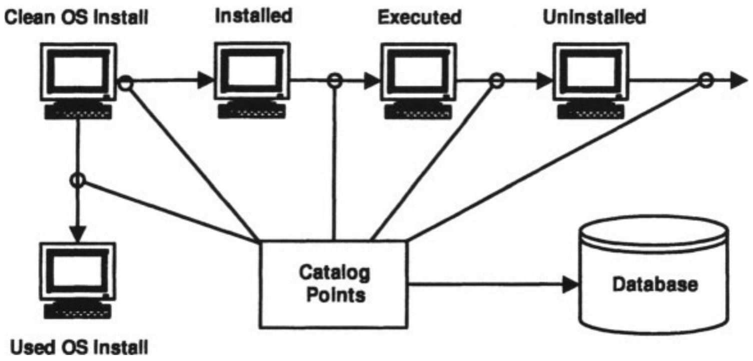


Figure 3. Fingerprint generation system.

prior to update should exist. However, this evidence is not from the execution of the current version of the program. When a program is updated, a completely new version may be installed in other folders in the file system; this is treated as a separate program. If the program itself or its data are updated, the behavior of the program is affected, and new evidence may be left in the executed state.

5. Generating Program Fingerprints

A laboratory configuration for generating program fingerprints during the various stages of its lifecycle is presented in Figure 3. One computer was configured with a base install of Windows XP without any other software packages. This computer was imaged and the image copied to other identical computers to create a baseline configuration for analysis. Several encryption, steganography and erasing programs were downloaded and burned on a single CD-ROM. Each program was analyzed independently.

After installing a program on one of the test machines, the machine was immediately switched off. This was done to limit file system activity performed by Windows upon shutdown. The hard drive was moved from the machine and connected to an analysis machine via a hardware write blocker. The drive was processed and cataloged using customized software. After this process was completed, the drive was replaced in the test machine and the program was executed, following which the drive was removed and re-analyzed. The drive was again returned to the test machine and the program uninstalled, upon which the drive was removed and analyzed one last time.

This procedure created the base set of data pertaining to the program during various stages in its lifecycle. Note that the updated and deleted

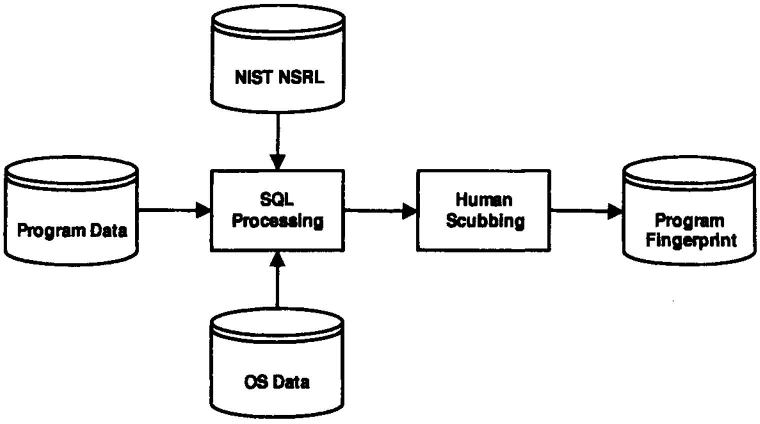


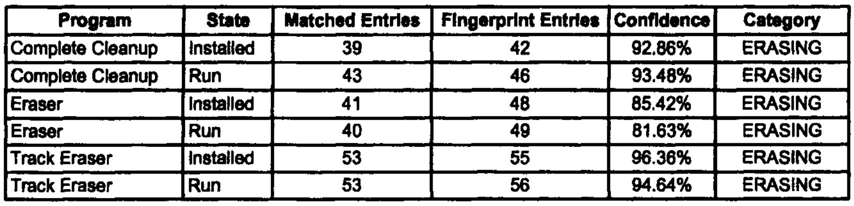
Figure 4. Generating program fingerprints.

states were not analyzed because programs typically generate little new evidence during these stages of their lifecycles.

Next, the test machines were re-imaged with the original base image and placed in a shared laboratory for general use for a period of two weeks. This allowed the base image to gather user and operating system changes that were used to reduce the program lifecycle data set. After the two-week period, the hard drives were removed from the machines and analyzed as before.

The images from the computers in the shared laboratory provided a large data set for programs in various stages of their lifecycles. These images were processed using a fingerprint generation engine to remove erroneous and redundant data, and to create fingerprints for each program during its lifecycle stages (Figure 4). The NIST National Software Reference Library [15], the clean base image, and the used base images were all used to process the data and create fingerprints. Each fingerprint was manually verified before producing the final program fingerprint.

Upon analyzing the program fingerprints, it was determined that many of the encryption, steganography and erasing programs shared files (e.g., DLLs and installation package files). This created a significant challenge to matching program fingerprints accurately. For example, in the uninstalled state, a program typically leaves a copy of the uninstall tool, which is shared by other programs. However, it was possible to determine which programs were present on the system using information such as full paths and registry settings.



Program	State	Matched Entries	Fingerprint Entries	Confidence	Category
Complete Cleanup	Installed	39	42	92.86%	ERASING
Complete Cleanup	Run	43	46	93.48%	ERASING
Eraser	Installed	41	48	85.42%	ERASING
Eraser	Run	40	49	81.63%	ERASING
Track Eraser	Installed	53	55	96.36%	ERASING
Track Eraser	Run	53	56	94.64%	ERASING

Figure 5. Results of Seraph fingerprint matching.

6. Seraph Program Detection Tool

A software tool named Seraph was created to assist forensic investigators in cataloging hard drives and detecting programs of interest based on their fingerprints. Seraph helps determine if any of the fingerprinted programs are/were present on the catalogued drive. Program fingerprints may be matched by filename, full path or MD5 hash of the file. Each of these options produces slightly different results. Filename matching produces a significant number of false positives. For example, many programs include a file named `readme.txt`. The number of false positives is reduced greatly when the full path information of files is considered, especially when the programs are installed in default directories.

Matching MD5 hash values can be effective, but it can actually reduce the number of features that a program will match against its fingerprint. This is because different versions of the same program (due to changes to executables and associated files) yield different MD5 hash values. However, programs typically use the same default installation folder and filenames, allowing filename and full path matching techniques to identify different versions of a program.

Seraph allows investigators to set a "confidence level" threshold for fingerprint matching. Many of the fingerprinted programs share DLLs or have files with identical names (e.g., `readme.txt`). By setting the threshold, an investigator can adjust the number of features used for fingerprint matching. When searching for uninstalled or deleted programs, a lower threshold should be used as many of the files associated with the programs have been removed. Since this can produce many false positives, the investigator must examine the results very carefully.

Once all the options have been selected, the investigator can use Seraph's "detect" function to match programs on the image with the stored fingerprints (Figure 5). Within seconds, Seraph displays a list of matched fingerprints at the specified confidence level. The investigator can then examine the details of the fingerprint matches or create a text report (Figure 6).

```
Seraph Report
Created on: 11-06-2005
====================================================================

Image Details
====================================================================
Image Name: Test Image 3
Image Taken By: Amanda Strickler
Image Date: 9/26/2005 at 12:57:58
====================================================================

Options
====================================================================
Filename: Yes
Fullpath: Yes
Hash: No
Confidence Level: 30%
====================================================================

Matched Programs
====================================================================

Program: Complete Cleanup
State: Installed
Matched Entries: 39
Fingerprint Entries: 42
Confidence: 92.86%
Category: ERASING

Fullpath: \Documents and Settings\All Users\Start Menu\Programs\
Complete Cleanup Trial\Complete Cleanup Trial.lnk
Filename: Complete Cleanup Trial.lnk
Fingerprint File Size:  700
Fingerprint MD5 Hash:   e898dbb32d5c7e79145f1b4a17321273
Image Size:             700
Image Hash:             e6fca3fb2f5a8476e188bbbc4629dda7

Fullpath: \Documents and Settings\User\Desktop\Complete Cleanup
Trial.lnk
Filename: Complete Cleanup Trial.lnk
Fingerprint File Size:  688
Fingerprint MD5 Hash:   28be74c11e765b967d5661dc78f41134
Image Size:             688
Image Hash:             37d19c438019dbea9bde43e9bbc4b5c3

Fullpath: \Program Files\Complete Cleanup Trial\activex_t.htm
Filename: activex_t.htm
Fingerprint File Size:  1064
Fingerprint MD5 Hash:   7fa6f58a735db46d215c683bfc4b262d
Image Size:             1064
Image Hash:             7fa6f58a735db46d215c683bfc4b262d
```

Figure 6. Seraph report.

7. Results and Discussion

Seraph provides an investigator with results based on program name, state and confidence level threshold. In the example output in Figure 5, it is possible to conclude that the program Complete Cleanup may have been executed on the system. Since the fingerprint catalog for the executed state contains more entries than the installed state, it can be ascertained that some files are added during program execution. Furthermore, the investigated drive exhibits additional matches with the executed fingerprint. Seraph offers a "detailed analysis" option that displays all matched features. This can be used to determine if a Windows Prefetch entry or certain configuration files were added to the system, indicating that the program had entered the executed state.

Seraph was blind tested for accuracy. Three test hard drives were loaded with several data concealment programs at various stages of their lifecycles. The test drives were then imaged by Seraph. Using the default settings to search for filenames and full paths with a confidence threshold of 30%, Seraph correctly detected 7 out of 7 random test programs in the installed and executed states.

Upon decreasing the confidence threshold to 15%, Seraph correctly detected 3 of 3 programs in the uninstalled state. However, Seraph did return 8 false positives at the lower confidence level. Nevertheless, using Seraph's "detailed analysis" option, it was possible to determine which matches were false positives. Figure 5 shows the relevant portion of a Seraph report that can aid in the analysis of these results.

Seraph is being tested by detectives from the Tulsa Police Department's Cyber Crimes Unit and agents from the Oklahoma State Bureau of Investigation. Currently, Seraph is able to detect data concealment programs based on the existence of files. But Seraph's fingerprint entries also contain information about changes to shared configuration files and registry settings; the next version of Seraph will use this information to improve the accuracy of program detection. Furthermore, the new version will incorporate several automated features to reduce the amount of manual analysis performed by forensic investigators.

8. Conclusions

Trace evidence left by data concealment programs during various stages of their lifecycles can be analyzed to provide valuable information, including the names and versions of the programs, and whether they were installed, executed or uninstalled. Having established the presence of a specific data concealment program on a seized computer, an investigator can attempt to exploit known vulnerabilities in the program to recover concealed or erased data. The Seraph program detection tool uses passive file system analysis to obtain information pertaining to the use of encryption, steganography and erasing programs. The current version has performed reasonably well in tests and in real investigations by two law enforcement agencies. However, more research and development efforts are needed to expand the coverage of the program detection tool and improve its accuracy.

References

[1] Access Data, Password Recovery Toolkit (PRTK) (www.accessdata .com/products/prtk).

[2] AKS-Labs, Find password protected files (www.aks-labs.com/solu tions/find-password-protected.htm).

[3] C. Brown, Detecting and collecting whole disk encryption media, presented at the *Department of Defense Cyber Crime Conference*, Palm Harbor, Florida, 2006.

[4] P. Burke and P. Craiger, Assessing trace evidence left by secure deletion programs, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, New York, pp. 185-195, 2006.

[5] K. Curran and K. Bailey, An evaluation of image-based steganography methods, *International Journal of Digital Evidence* vol. 2(2), 2003.

[6] Cypherix, Cryptainer LE (www.cypherix.com/cryptainerle).

[7] Defense Security Service, *National Industrial Security Program Operating Manual (NISPOM)*, DoD 5220.22-M, U.S. Department of Defense (www.dss.mil/isec/nispom_0195.pdf), 1995.

[8] Free Downloads Center, Encrypted Files Search 1.2 (www.freedown loadscenter.com/Utilities/Misc_Encryption_Utilities/Encrypted_Fi les_Search.html).

[9] Giant Internet, Internet Eraser Pro (www.interneteraser.net).

[10] InstallShield, Creating registry keys (helpnet.installshield.com/robo /projects/helplibdevstudio9/IHelpRegistryKeys.htm).

[11] J. Jackson, G. Gunsch, G. Lamont and R. Claypoole, Blind steganography detection using a computational immune system: A work in progress, *International Journal of Digital Evidence*, vol. 1(4), 2003.

[12] G. Kessler, An overview of steganography for the computer forensics examiner, *Forensic Science Communications*, vol. 6(3), 2004.

[13] Kryptel, Kryptel Encryption Suite (www.kryptel.com/products/kryptel).

[14] K. Mandia, C. Prosise and M. Pepe, *Incident Response and Computer Forensics*, McGraw-Hill/Osborne, Emeryville, California, 2003.

[15] National Institute of Standards and Technology (NIST), National Software Reference Library (NSRL), Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, Maryland (www.nsrl.nist.gov).

[16] B. Nelson, A. Phillips, F. Enfinger and C. Steuart, *Guide to Computer Forensics and Investigations*, Thompson Course Technology, Boston, Massachusetts, 2004.

[17] NewSoftwares.net, Folder Lock (www.newsoftwares.net/folderlock).

[18] Robin Hood Software, Evidence Eliminator (www.evidence-eliminator.com/product.d2w).

[19] J. Seigfried, C. Siedsma, B. Countryman and C. Hosmer, Examining the encryption threat, *International Journal of Digital Evidence*, vol. 2(3), 2004.

[20] J. Sheesley, Use XP's Prefetch feature to improve system performance, TechRepublic (techrepublic.com.com/5100-1035_11-5165773.html?tag=e064#), 2004.

[21] Softpedia, Bestcrypt 7.20.2 (www.softpedia.com/get/Security/Encrypting/BestCrypt.shtml), 2005.

[22] U.S. District Court (Southern District of New York), United States of America v. Robert Johnson (files.findlaw.com/news.findlaw.com/hdocs/docs/chldprn/usjhnsn62805ind.pdf), June 28, 2005.

[23] U.S. Immigration and Customs Enforcement, U.S. charges ex CEO with using the Internet for child pornography and with obstruction of justice (www.ice.gov/graphics/news/newsreleases/articles/050628newyork.htm), June 28, 2005.

[24] P. Wayner, *Disappearing Cryptography: Information Hiding, Steganography and Watermarking*, Morgan Kauffman, San Francisco, California, 2002.

[25] WetStone Technologies, Stego Suite (www.wetstonetech.com).