DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

I, Daniel David Rigmaiden, declare[1] the following:

1. The N.D.Cal. 08-70460-HRL/PVT and 08-70502-PVT warrants used by the government to search and seize digital data from my computer system state that the government may only expose and search for files "[f]or the period January 1, 2005, through the present[.]"[2] This declaration details my efforts to determine the most accurate and reliable process to "period" a file contained on an NTFS formatted drive. As a result of the tests detailed in this declaration, I determined (1) an NTFS file "last access date" property is *(i)* an inapplicable/irrelevant record to "period" a file, but *(ii)* a reliable record of the last date/time a file was opened and read (albeit within an approximate "less than or equal to" 60 minute "forward looking" margin of error),[3] (2) an NTFS file "creation date" property is an unreliable record to "period" a file,[4] and (3) an NTFS file "last modified date" property is the absolute file creation date/time of a file and, therefore, a reliable record to "period" a file.[5]

---

1. This declaration is being submitted under the protections of *Simmons*. *See* Simmons v. United States, 390 U.S. 377, 394 (1968) (Holding that "when a defendant testifies in support of a motion to suppress evidence on Fourth Amendment grounds, his testimony may not thereafter be admitted against him at trial on the issue of guilt unless he makes no objection."). I object to the government attempting to introduce this declaration as evidence at trial.

2. *E.g.*, *Submission Of Documents Related To Original Northern District Of California 08-70460-HRL Search Warrant Used To Physically Search Apartment No. 1122*, *Warrant* (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #566-2, p. 5).

3. *See* ¶ No. 17, *infra* (discussing results of tests pertaining to the "last access date" file property), ¶ No. 5, *infra* (discussing the "less than or equal to" 60 minute "forward looking" margin of error applicable to file "last access dates"), and ¶ Nos. 23-26, *infra* (discussing the general consensus regarding the unreliability of using a "last access date" and/or "creation date" to "period" a file and the general consensus that using a file's "last modified date" is the most reliable means to "period" a file).

4. *See* ¶ Nos. 18-22, *infra* (discussing results of tests pertaining to the "creation date" file property) and ¶ Nos. 23-26, *infra* (discussing the general consensus regarding the unreliability of using a "last access date" and/or "creation date" to "period" a file and the general consensus that using a file's "last modified date" is the most reliable means to "period" a file).

5. *See* ¶ Nos. 18-22, *infra* (discussing results of tests pertaining to the "last modified date" file property) and ¶ Nos. 23-26, *infra* (discussing the general consensus regarding the unreliability of using a "last access date" and/or "creation date"

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

---

2. The tests explained in this declaration were conducted using my defense laptop computer (*i.e.*, IBM ThinkPad T43), which has an NTFS formatted system drive running the Microsoft Windows XP SP3 operating system,[6] while logged in with Administrator permissions. Unless otherwise noted, all tests were conducted from the command line using a command-prompt window.[7] In order to conduct my tests, I gathered all technical information from the Microsoft Windows "Help and Support Center," which ships standard with Windows XP and is targeted towards beginner level computer users.[8][9] Because there was a lack of necessity, no expert level computer knowledge or expert level computer forensic knowledge was referenced.

3. The government claims that during execution of the N.D.Cal. 08-70460-HRL/PVT and 08-70502-PVT warrants, "Agent Daun found a file labeled 'filesalot.dcv' that contains the bulk of the incriminating information in this case."[10] Put in accurate technical

---

to "period" a file and the general consensus that using a file's "last modified date" is the most reliable means to "period" a file).

6. Attached to this declaration as ATTACHMENT 01 is a screenshot of relevant system information and disk partition information as displayed by my defense laptop computer using the command line commands "Systeminfo" and "Diskpart." The precise command entered at the command-prompt was as follows:

```
echo. & date /t & time /t & echo. & systeminfo | findstr /b /c:"OS Name" /c:"OS
Version" /c:"System Up Time" & echo. & echo list volume | diskpart | findstr
/c:"Volume #" /c:"Volume 1" & echo.
```

7. In order to open command-prompt windows for my tests, I executed "%SystemRoot%\system32\cmd.exe" on my defense laptop computer. Attached to this declaration as ATTACHMENT 02 is a PDF print of the Microsoft Windows "Help and Support Center"reference page for "Command shell overview," which explains various aspects of the Microsoft Windows command-prompt.

8. "Microsoft Help and Support Center is a comprehensive resource for practical advice, tutorials, and demonstrations to help you learn to use Microsoft Windows XP. Use the Search feature, Index, or table of contents to view all Windows Help resources, including those that are on the Internet." *See* ATTACHMENT 75.

9. Because I do not have access to the Internet, no Internet based features of the "Help and Support Center" were used to gather information.

10. *Government's Response To Defendant's Motion To Suppress* (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz.,

terms, the bulk of the evidence was found within the NTFS formatted logical hard drive created by mounting "filesalot.dcv" via the DriveCrypt software,[11] which assigns "filesalot.dcv" a drive letter (in this case, the "T" drive) within the host Microsoft Windows operating system.[12] As a result of mounting "filesalot.dcv," the "T" drive is indistinguishable from a full-blown physical hard drive when accessed by a user of the host Microsoft Windows operating system.[13]

4. In order to determine the most accurate method to "period" a file contained on an NTFS formatted drive such as the "T" drive, I began by conducting research using the Microsoft Windows "Help and Support Center." Using my defense laptop computer, I clicked the "Start" button on the "Task Bar" and selected "Help and Support," which caused the "Help and Support Center" to open. I then ran a search on "ntfs file system" and clicked on the resulting "NTFS" link, which opened a page about the NTFS file system. Attached to this declaration as ATTACHMENT 04 is a PDF print of the noted "NTFS" page. I then reran the search on "ntfs file system" and clicked on the resulting "Glossary" link, which opened an alphabetical index of various terms. I then clicked the "NTFS" link which opened the

Dkt. #873, p. 65).

11. *See* SecurStar [website], *SecurStar, Encryption Software Solutions - Products - DriveCrypt*, http://www.securstar.com/products_drivecrypt.php (last accessed: Dec. 16, 2010) (explaining the DriveCrypt product and encrypted virtual drives utilizing *.dcv container files).

12. Attached to this declaration as ATTACHMENT 03 is a screenshot of the Microsoft Windows File Explorer drive properties for "filesalot.dcv" showing it to be accessible as the "T" drive, an NTFS formatted drive.

13. IRS-CI Agent Daun explained in her "Computer Forensic Report" that she imaged "filesalot.dcv" as if it were a physical hard drive. *See Third Submission Of Consolidated Exhibits Relating To Discovery And Suppression Issues*, EXHIBIT 01 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #863-1, p. 16) ("Using FTK Imager v.2.5.2, I imaged the virtual container file. The image is identified as 1000220515_S1_TSB100G_1_DriveCrypt01."); *see also* ATTACHMENT 03.

reference section for NTFS. Attached to this declaration as ATTACHMENT 05 is a PDF print of the "NTFS" reference section of the Glossary page. I then ran a search on "file properties" and clicked on the resulting "File properties overview" link, which opened a page about Microsoft Windows file properties. Attached to this declaration as ATTACHMENT 06 is a PDF print of the noted "File properties overview" page. I then ran a search on "file access time" and clicked on the resulting "Fsutil: behavior" link, which opened the reference page for the "fsutil" command relevant to the command-prompt. Attached to this declaration as ATTACHMENT 07 is a PDF print of the noted "Fsutil: behavior" page.

5. After reading the documents noted in ¶ No. 4 above, I learned that NTFS files have three separate date/time file properties: (1) the date and time the file was created (hereafter, "creation date"), (2) the date and time the file was last written to (hereafter, "last modified date"), and (3) the date and time the file was last opened and read (hereafter, "last access date"). Additionally, I learned from the "Fsutil: behavior" reference page that the time values for file "last access dates" are only accurate to a "less than or equal to" 60 minute "forward looking" margin of error: "NTFS typically updates a file's attribute on disk if the current Last Access Time in memory differs by more than an hour from the Last Access Time stored on disk, or when all in-memory references to that file are gone, whichever is more recent. For example, if a file's current Last Access Time is 1:00 P.M., and you read the file at 1:30 P.M., NTFS does not update the Last Access Time. If you read the file again at 2:00 P.M., NTFS updates the Last Access Time in the file's attribute to reflect 2:00 P.M. because the file's attribute shows 1:00 P.M. and the in-memory Last Access Time shows 2:00 P.M."[14]

14. *See* ATTACHMENT 07.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

---

6. Next, I set out to determine a reliable way to read the three noted file date properties for any given NTFS file. Using my defense laptop computer, I ran a search on "command line reference a-z" within the Microsoft Windows "Help and Support Center" and clicked on the resulting "Command-line reference A-Z" link, which opened an alphabetical index of commands that can be executed from the command-prompt. I then clicked the "dir" link which opened the reference page for the "dir" command. Attached to this declaration as ATTACHMENT 08 is a PDF print of the reference page for the "dir" command.

7. After reading the reference page for the "dir" command, I learned that using the command in the following manner would make a list of the "creation dates" for all files on a given drive—for example, the "C" drive as specified by "c:\"—and display the results to the command-prompt window:

```
dir c:\ /T:C /S /O:N /a:-d
```

8. After reading the reference page for the "dir" command, I learned that using the command in the following manner would make a list of the "last modified dates" for all files on a given drive—for example, the "C" drive as specified by "c:\"—and display the results to the command-prompt window:

```
dir c:\ /T:W /S /O:N /a:-d
```

9. After reading the reference page for the "dir" command, I learned that using the command in the following manner would make a list of the "last access dates" for all files on a given drive—for example, the "C" drive as specified by "c:\"—and display the results to the

command-prompt window:

```
dir c:\ /T:A /S /O:N /a:-d
```

10. Before relying upon the "dir" command to read file date/time properties, I first needed to confirm that doing so would not forcibly update any of the file date/time properties for a given file. If use of the "dir" command results in a queried file's "creation date," "last access date," or "last modified date" to update to the date/time the "dir" command was executed, the very data that is the subject of my analysis will be contaminated. By conducting the following test via the command-prompt, I was able to confirm that the "dir" command does not rewrite file date/time properties for any file that is queried by the command: (1) I used the "echo" command to create the file "test28.txt" on my "C" drive while filling the file with the text, "test28," and a carriage return,[15] (2) I used the "dir" command to list the "creation date," "last modified date," and "last access date," of "c:\test28.txt," as explained in ¶ Nos. 7, 8, and 9 above,[16] (3) I advanced the system date/time of my computer one day forward so that

15. Attached to this declaration as ATTACHMENT 09 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test28.txt to show does not exist: & echo. & type c:\test28.txt & echo. &
echo Create c:\test28.txt with the text "test28" followed by a carriage return, i.e.,
0x0D 0x0A: & echo test28> c:\test28.txt & echo. & echo verify that c:\test28.txt
created by reading contents of file: & echo. & type c:\test28.txt
```

16. Attached to this declaration as ATTACHMENT 10 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
file date properties for c:\test28.txt... & echo. & echo File creation date: & dir
c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. &
echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir
c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

---

immediate-future NTFS file access occurrences will be more than 60 minutes past the file "last access date" logged in step No. 2 above[17]—this will force the NTFS file system to update the "last access date" for "c:\test28.txt" if a file access occurs during step No. 4 of my test,[18] (4) with the date/time now more than 60 minutes in the future, I used the "dir" command to list the "creation date," "last modified date," and "last access date," of "c:\test28.txt," as explained in ¶ Nos. 7, 8, and 9 above,[19] (5) I again advanced the system date/time of my computer one day forward so that immediate-future NTFS file access occurrences will be more than 60 minutes past the file "last access date" logged in step No. 4 above[20]—this will force the NTFS file system to update the "last access date" for "c:\test28.txt" if a file access occurs during step No.

---

17. Attached to this declaration as ATTACHMENT 11 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to force NTFS "last 60 minutes" last access check... & date 01-04-
2013 & echo. & echo Date is now: & date /t & time /t
```

18. *See* ATTACHMENT 07 (explaining the NTFS "less than or equal to" 60 minute "forward looking" margin of error issue).

19. Attached to this declaration as ATTACHMENT 12 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
file date properties for c:\test28.txt... & echo. & echo File creation date: & dir
c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. &
echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir
c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

20. Attached to this declaration as ATTACHMENT 13 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to force NTFS "last 60 minutes" last access check... & date 01-05-
2013 & echo. & echo Date is now: & date /t & time /t
```

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

---

6 of my test,[21] (6) with the date/time now more than 60 minutes in the future, I used the "dir" command to list the "creation date," "last modified date," and "last access date," of "c:\test28.txt," as explained in ¶ Nos. 7, 8, and 9 above,[22] (7) I reset the system date/time of my computer back two days to the actual date,[23] and (8) I compared the results of the "dir" command used in step Nos. 2, 4, and 6 and confirmed that the file date/time properties for "C:\test28.txt" had not changed after using the "dir" command on the pseudo-different date/times.[24] Because the dates had not changed, it was confirm that the "dir" command does not rewrite file date/time properties for any file that is queried by the command.

11. Because my planned file date/time properties tests involved determining the "period" of files based on data content, I first looked for a way to take a "fingerprint" of a file so that it could be determine if a given file is later overwritten with a new file having the same filename. According to IRS-CI Agent Daun, there exists a cryptographic hashing process to

---

21. *See* ATTACHMENT 07 (explaining the NTFS "less than or equal to" 60 minute "forward looking" margin of error issue).

22. Attached to this declaration as ATTACHMENT 14 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
file date properties for c:\test28.txt... & echo. & echo File creation date: & dir
c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. &
echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir
c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

23. Attached to this declaration as ATTACHMENT 15 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by two days... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

24. *Compare* listed file property date/times listed at ATTACHMENT 10, ATTACHMENT 12, and ATTACHMENT 14.

take hash value "fingerprints" of files: "A hash value is a mathematical algorithm that generates a hexidecimal output value (alpha-numeric text string). The text string is generated by a formula in such a way that it is extremely unlikely that another file or image will produce the same hash value. The tiniest change in the original data produces huge changes in the resulting text string. Essentially, the hash value of a file or image can be considered as the 'fingerprint' of the file or image."[25] I decided to use the same "fingerprint" process explained by IRS-CI Agent Daun during my file date/time properties tests.

12. Having chosen a process to take "fingerprints" of files, I next researched the freely available and widely used command line program called GnuPG (*i.e.*, GPG), which simplifies various cryptographic operations including the process of taking hash value "fingerprints" of files. My defense previously provided me with an installation file for GPG called "gnupg-w32cli-1.4.11.exe." This installation file was downloaded from http://www.gnupg.org. Having previously installed GPG to "c:\gpg" on my defense laptop computer, I used Microsoft Windows Notepad to open the GPG manual located at "c:\gpg\Doc\gpg.man." Attached to this declaration as ATTACHMENT 16 is a PDF print of the "gpg.man" for GPG.

13. After reading "gpg.man" for GPG, I learned that the following command would produce a SHA-256 hash digest (*i.e.*, "fingerprint") of a specified file:

```
c:\gpg\gpg.exe --print-md sha256 [filename]
```

The above command will take a "fingerprint" of a file using the SHA-256 cryptographic

25. *Third Submission Of Consolidated Exhibits Relating To Discovery And Suppression Issues*, EXHIBIT 01 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #863-1, p. 6).

hashing function[26] but will only "fingerprint" the file's data content while ignoring all file properties and the filename, which are not considered part of a file, and without writing data to the file.

14. Having identified the three available file date/time properties for NTFS files (*see* ¶ Nos. 4-5); and having determined a reliable method to access and list file date/time properties for files on an NTFS formatted drive (*see* ¶ Nos. 6-9) without contaminating the properties themselves (*see* ¶ No. 10); and having determined a reliable method to take a file's "fingerprint" so as to know if a prior file is overwritten with a new file (*see* ¶ Nos. 11-13) – I next set out to determine which of the three file date/time properties most accurately "periods" a file.

15. In order to make my determination, I conducted the following tests via the command-prompt so as to gather sufficient data for my analysis: [PART No. 1] (1) I created the file "test29.txt" on my "C" drive and filled the file with the text, "test29," without the quotation marks and comma,[27] (2) I used the "dir" command to list the "creation date," "last modified

26. SHA-256, as well as SHA-1, SHA-224, SHA-384, SHA512, SHA-512/224 and SHA-512/256, are all "iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. These algorithms enable the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest." National Institute of Standards and Technology, FIPS PUB 180-4, *Federal Information Processing Standards Publication: Secure Hash Standard (SHS)* (Mar. 2012), p. 3.

27. Attached to this declaration as ATTACHMENT 17 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test29.txt to show does not exist: & echo. & type c:\test29.txt & echo. &
echo Create c:\test29.txt with the text "test29" followed by a carriage return, i.e.,
0x0D 0x0A: & echo test29> c:\test29.txt & echo. & echo verify that c:\test29.txt
created by reading contents of file: & echo. & type c:\test29.txt
```

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

---

date," and "last access date," of "c:\test29.txt," as explained in ¶ Nos. 7, 8, and 9 above,[28] (3) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test29.txt," as explained in ¶ Nos. 11, 12, and 13 above,[29] (4) I advanced the system date/time of my computer one day forward so that immediate-future NTFS file access occurrences will be more than 60 minutes past the file "last access date" logged by my computer during step No. 1 above[30]—this will force the NTFS file system to update the "last access date" for "c:\test29.txt" when the file access occurs during step No. 5 of my test,[31] (5) with the date/time now more than 60 minutes in the future, I accessed "c:\test29.txt" by opening and reading the data content of the file to the command

---

28. Attached to this declaration as ATTACHMENT 18 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
all file date properties for c:\test29.txt... & echo. & echo File creation date: &
dir c:\test29.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" &
echo. & echo File last modified date: & dir c:\test29.txt /T:W /O:N /a:-d | findstr /
v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir
c:\test29.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

29. Attached to this declaration as ATTACHMENT 19 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test29.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test29.txt
```

30. Attached to this declaration as ATTACHMENT 20 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to force NTFS "last 60 minutes" last access check... & date 01-04-
2013 & echo. & echo Date is now: & date /t & time /t
```

31. *See* ATTACHMENT 07 (explaining the NTFS "less than or equal to" 60 minute "forward looking" margin of error issue).

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

---

line,[32] (6) I reset the system date/time of my computer back one day to the actual date,[33] (7) I used the "dir" command to list the "creation date," "last modified date," and "last access date," of "c:\test29.txt," as explained in ¶ Nos. 7, 8, and 9 above,[34] (8) to confirm "c:\test29.txt" was still the same file, I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test29.txt," as explained in ¶ Nos. 11, 12, and 13 above,[35] [PART No. 2] (9) I created the file "test30.txt" on my "C" drive and filled the file with the text, "test30," without the quotation marks and comma,[36] (10) I used the "dir" command to list the "creation date" and "last modified date" of

32. Attached to this declaration as ATTACHMENT 21 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo access
c:\test29.txt by opening and reading the data content to the command line: & echo. &
type c:\test29.txt
```

33. Attached to this declaration as ATTACHMENT 22 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

34. Attached to this declaration as ATTACHMENT 23 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
all file date properties for c:\test29.txt... & echo. & echo File creation date: &
dir c:\test29.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" &
echo. & echo File last modified date: & dir c:\test29.txt /T:W /O:N /a:-d | findstr /
v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir
c:\test29.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

35. Attached to this declaration as ATTACHMENT 24 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test29.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test29.txt
```

36. Attached to this declaration as ATTACHMENT 25 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

"c:\test30.txt," as explained in ¶ Nos. 7 and 8,[37] (11) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test30.txt," as explained in ¶ Nos. 11, 12, and 13 above,[38] (12) I advanced the system date/time of my computer one day forward[39] so that immediate-future changes to the file date/time properties for "c:\test30.txt" will be easier to spot during step No. 15 of my test, (13) I modified "c:\test30.txt" by using the "echo" command to write additional data to "c:\test30.txt,"[40] (14) I reset the system date/time of my computer back one day to the

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test30.txt to show does not exist: & echo. & type c:\test30.txt & echo. &
echo Create c:\test30.txt with the text "test30" followed by a carriage return, i.e.,
0x0D 0x0A: & echo test30> c:\test30.txt & echo. & echo verify that c:\test30.txt
created by reading contents of file: & echo. & type c:\test30.txt
```

37. Attached to this declaration as ATTACHMENT 26 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test30.txt... & echo.
& echo File creation date: & dir c:\test30.txt /T:C /O:N /a:-d | findstr /v /b /r
/c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test30.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

38. Attached to this declaration as ATTACHMENT 27 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test30.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test30.txt
```

39. Attached to this declaration as ATTACHMENT 28 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to make changes of file date/time properties easier to spot... & date
01-04-2013 & echo. & echo Date is now: & date /t & time /t
```

40. Attached to this declaration as ATTACHMENT 29 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Modify
c:\test30.txt by adding the text "test30" to c:\test30.txt followed by a carriage
return, i.e., 0x0D 0x0A: & echo test30>> c:\test30.txt & echo. & echo verify that c:\
test30.txt now contains additional text by reading contents of file: & echo. & type
```

---

actual date,[41] (15) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test30.txt," as explained in ¶ Nos. 7 and 8,[42] (16) to confirm that "c:\test30.txt" was now a different file, I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test30.txt," as explained in ¶ Nos. 11, 12, and 13 above,[43] [PART No. 3] (17) I created the file "test31.txt" on my "C" drive and filled the file with the text, "test31," without the quotation marks and comma,[44] (18) I used the "dir" command to list the "creation date" and "last

---

c:\test30.txt

41. Attached to this declaration as ATTACHMENT 30 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

42. Attached to this declaration as ATTACHMENT 31 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test30.txt... & echo.
& echo File creation date: & dir c:\test30.txt /T:C /O:N /a:-d | findstr /v /b /r
/c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test30.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

43. Attached to this declaration as ATTACHMENT 32 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test30.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test30.txt
```

44. Attached to this declaration as ATTACHMENT 33 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test31.txt to show does not exist: & echo. & type c:\test31.txt & echo. &
echo Create c:\test31.txt with the text "test31" followed by a carriage return, i.e.,
0x0D 0x0A: & echo test31> c:\test31.txt & echo. & echo verify that c:\test31.txt
created by reading contents of file: & echo. & type c:\test31.txt
```

modified date" of "c:\test31.txt," as explained in ¶ Nos. 7 and 8,[45] (19) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test31.txt," as explained in ¶ Nos. 11, 12, and 13 above,[46] (20) I advanced the system date/time of my computer one day forward[47] so that immediate-future changes to the file date/time properties for "c:\test31.txt" will be easier to spot during step No. 23 of my test, (21) I copied "c:\test31.txt" to "c:\backup\test31.txt,"[48] (22) I reset the system date/time of my computer back one day to the actual date,[49] (23) I used

45. Attached to this declaration as ATTACHMENT 34 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test31.txt... & echo. & echo File creation date: & dir c:\test31.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test31.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

46. Attached to this declaration as ATTACHMENT 35 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test31.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test31.txt
```

47. Attached to this declaration as ATTACHMENT 36 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance date by one day to make changes of file date/time properties easier to spot... & date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
```

48. Attached to this declaration as ATTACHMENT 37 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Copy c:\test31.txt to c:\backup\test31.txt... & echo. & xcopy c:\test31.txt c:\backup\
```

49. Attached to this declaration as ATTACHMENT 38 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
```

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

---

the "dir" command to list the "creation date" and "last modified date" of "c:\backup\test31.txt," as explained in ¶ Nos. 7 and 8,[50] (24) to show that "c:\backup\test31.txt" was still the same file as "c:\test31.txt," I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\backup\test31.txt," as explained in ¶ Nos. 11, 12, and 13 above,[51] (25) to show that "c:\test31.txt" was still the same file as "c:\backup\test31.txt," I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test31.txt," as explained in ¶ Nos. 11, 12, and 13 above,[52] [PART No. 4] (26) I created the file "test45.txt" on my "C" drive and filled the file with the text, "test45.1," without the quotation marks and comma,[53] (27) I used the "dir" command to

---

50. Attached to this declaration as ATTACHMENT 39 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\backup\test31.txt...
& echo. & echo File creation date: & dir c:\backup\test31.txt /T:C /O:N /a:-d |
findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: &
dir c:\backup\test31.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)"
```

51. Attached to this declaration as ATTACHMENT 40 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\backup\test31.txt... & echo. &
c:\gpg\gpg.exe --print-md sha256 c:\backup\test31.txt
```

52. Attached to this declaration as ATTACHMENT 41 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test31.txt again to show still the same
file... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test31.txt
```

53. Attached to this declaration as ATTACHMENT 42 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test45.txt to show does not exist: & echo. & type c:\test45.txt & echo. &
echo Create c:\test45.txt with the text "test45.1" followed by a carriage return,
i.e., 0x0D 0x0A: & echo test45.1> c:\test45.txt & echo. & echo verify that
```

list the "creation date" and "last modified date" of "c:\test45.txt," as explained in ¶ Nos. 7 and 8,[54] (28) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test45.txt," as explained in ¶ Nos. 11, 12, and 13 above,[55] (29) I advanced the system date/time of my computer one day forward[56] so that immediate-future changes to the file date/time properties for "c:\test45.txt" will be easier to spot during step No. 33 of my test, (30) I deleted the file "c:\test45.txt,"[57] (31) within fifteen (15) seconds of deleting "c:\test45.txt," I recreated "c:\test45.txt" but filled it with different data content than what was used in the first version of

---

```
c:\test45.txt created by reading contents of file: & echo. & type c:\test45.txt
```

54. Attached to this declaration as ATTACHMENT 43 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test45.txt... & echo.
& echo File creation date: & dir c:\test45.txt /T:C /O:N /a:-d | findstr /v /b /r
/c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test45.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

55. Attached to this declaration as ATTACHMENT 44 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test45.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test45.txt
```

56. Attached to this declaration as ATTACHMENT 45 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to make changes of file date/time properties easier to spot... & date
01-04-2013 & echo. & echo Date is now: & date /t & time /t
```

57. Attached to this declaration as ATTACHMENT 46 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Delete
the file c:\test45.txt... & echo. & del c:\test45.txt & echo. & echo Attempt to open
c:\test45.txt to show does not exist: & echo. & type c:\test45.txt
```

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

"c:\test45.txt,"[58] (32) I reset the system date/time of my computer back one day to the actual date,[59] (33) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test45.txt," as explained in ¶ Nos. 7 and 8,[60] (34) to show that new version of "c:\test45.txt" is different compared to original version of "c:\test45.txt," I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test45.txt," as explained in ¶ Nos. 11, 12, and 13 above,[61] [PART No. 5] (35) I created the file "test46a.txt" on my "C" drive and filled the file

58. Attached to this declaration as ATTACHMENT 47 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within
15 seconds of deleting c:\test45.txt, recreate c:\test45.txt but fill with different
data content than what was used in the first version of c:\test45.txt... & echo. &
echo Create c:\test45.txt with the text "test45.22" followed by a carriage return,
i.e., 0x0D 0x0A: & echo test45.22> c:\test45.txt & echo. & echo verify that
c:\test45.txt created by reading contents of file: & echo. & type c:\test45.txt
```

59. Attached to this declaration as ATTACHMENT 48 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

60. Attached to this declaration as ATTACHMENT 49 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for new version of
c:\test45.txt... & echo. & echo File creation date: & dir c:\test45.txt /T:C /O:N
/a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified
date: & dir c:\test45.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)"
```

61. Attached to this declaration as ATTACHMENT 50 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of new version of c:\test45.txt... & echo. &
c:\gpg\gpg.exe --print-md sha256 c:\test45.txt
```

---

with the text, "test46a," without the quotation marks and comma,[62] (36) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test46a.txt," as explained in ¶ Nos. 7 and 8,[63] (37) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test46a.txt," as explained in ¶ Nos. 11, 12, and 13 above,[64] (38) I advanced the system date/time of my computer one day forward[65] so that immediate-future changes to the file date/time properties for "c:\test46a.txt" will be easier to spot during step No. 41 of my test, (39) I created the file "test46b.txt" on my "C" drive and filled the file with the text, "test46b.1,"

---

62. Attached to this declaration as ATTACHMENT 51 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test46a.txt to show does not exist: & echo. & type c:\test46a.txt & echo.
& echo Create c:\test46a.txt with the text "test46a" followed by a carriage return,
i.e., 0x0D 0x0A: & echo test46a> c:\test46a.txt & echo. & echo verify that
c:\test46a.txt created by reading contents of file: & echo. & type c:\test46a.txt
```

63. Attached to this declaration as ATTACHMENT 52 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test46a.txt... &
echo. & echo File creation date: & dir c:\test46a.txt /T:C /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test46a.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

64. Attached to this declaration as ATTACHMENT 53 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test46a.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test46a.txt
```

65. Attached to this declaration as ATTACHMENT 54 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to make changes of file date/time properties easier to spot... & date
01-04-2013 & echo. & echo Date is now: & date /t & time /t
```

---

without the quotation marks and comma,[66] (40) I reset the system date/time of my computer back one day to the actual date,[67] (41) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test46b.txt," as explained in ¶ Nos. 7 and 8,[68] (42) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test46b.txt," as explained in ¶ Nos. 11, 12, and 13 above,[69] (43) I deleted the file "c:\test46a.txt,"[70] (44) within fifteen (15) seconds of

---

66. Attached to this declaration as ATTACHMENT 55 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test46b.txt to show does not exist: & echo. & type c:\test46b.txt & echo.
& echo Create c:\test46b.txt with the text "test46b.1" followed by a carriage return,
i.e., 0x0D 0x0A: & echo test46b.1> c:\test46b.txt & echo. & echo verify that
c:\test46b.txt created by reading contents of file: & echo. & type c:\test46b.txt
```

67. Attached to this declaration as ATTACHMENT 56 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

68. Attached to this declaration as ATTACHMENT 57 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test46b.txt... &
echo. & echo File creation date: & dir c:\test46b.txt /T:C /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test46b.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

69. Attached to this declaration as ATTACHMENT 58 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test46b.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test46b.txt
```

70. Attached to this declaration as ATTACHMENT 59 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Delete
the file c:\test46a.txt... & echo. & del c:\test46a.txt & echo. & echo Attempt to
```

deleting "c:\test46a.txt," I renamed "c:\test46b.txt" to "c:\test46a.txt,"[71] (45) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test46a.txt," as explained in ¶ Nos. 7 and 8,[72] (46) to show that renamed "c:\test46b.txt" (*i.e.*, now "c:\test46a.txt") is still the same file while in renamed form, I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test46a.txt," as explained in ¶ Nos. 11, 12, and 13 above,[73] [PART No. 6] (47) I created the file "test47.txt" in a folder named "test47_location_a" on my "C" drive and filled the file with the text, "test47_location_a," without the quotation marks and comma,[74] (48) I used the

```
open c:\test46a.txt to show does not exist: & echo. & type c:\test46a.txt
```

71. Attached to this declaration as ATTACHMENT 60 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within
15 seconds of deleting c:\test46a.txt, rename c:\test46b.txt to c:\test46a.txt... &
echo. & rename c:\test46b.txt test46a.txt
```

72. Attached to this declaration as ATTACHMENT 61 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for c:\test46a.txt... &
echo. & echo File creation date: & dir c:\test46a.txt /T:C /O:N /a:-d | findstr /v /b
/r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test46a.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
```

73. Attached to this declaration as ATTACHMENT 62 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test46a.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test46a.txt
```

74. Attached to this declaration as ATTACHMENT 63 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
to open c:\test47_location_a\test47.txt to show does not exist: & echo. & type
c:\test47_location_a\test47.txt & echo. & echo Create c:\test47_location_a\test47.txt
with the text "test47_location_a" followed by a carriage return, i.e., 0x0D 0x0A: &
echo. & mkdir c:\test47_location_a & echo. & echo test47_location_a>
c:\test47_location_a\test47.txt & echo. & echo verify that
```

"dir" command to list the "creation date" and "last modified date" of "c:\test47_location_a\test47.txt," as explained in ¶ Nos. 7 and 8,[75] (49) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test47_location_a\test47.txt," as explained in ¶ Nos. 11, 12, and 13 above,[76] (50) I advanced the system date/time of my computer one day forward[77] so that immediate-future changes to the file date/time properties for "c:\test46a.txt" will be easier to spot during step No. 41 of my test, (51) I created the file "test47.txt" in a folder named "test47_location_b" on my "C" drive and filled the file with the text, "test47_location_b," without the quotation marks and comma,[78] (52) I reset the system

---

```
c:\test47_location_a\test47.txt created by reading contents of file: & echo. & type
c:\test47_location_a\test47.txt
```

75. Attached to this declaration as ATTACHMENT 64 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for
c:\test47_location_a\test47.txt... & echo. & echo File creation date: & dir
c:\test47_location_a\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test47_location_a\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)"
```

76. Attached to this declaration as ATTACHMENT 65 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt... & echo.
& c:\gpg\gpg.exe --print-md sha256 c:\test47_location_a\test47.txt
```

77. Attached to this declaration as ATTACHMENT 66 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance
date by one day to make changes of file date/time properties easier to spot... & date
01-04-2013 & echo. & echo Date is now: & date /t & time /t
```

78. Attached to this declaration as ATTACHMENT 67 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
eecho. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt
```

---

date/time of my computer back one day to the actual date,[79] (53) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test47_location_b\test47.txt," as explained in ¶ Nos. 7 and 8,[80] (54) I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test47_location_b\test47.txt," as explained in ¶ Nos. 11, 12, and 13 above,[81] (55) I deleted the file "c:\test47_location_a\test47.txt,"[82] (56) within fifteen (15) seconds of deleting "c:\test47_location_a\test47.txt," I moved "c:\test47_location_b\test47.txt" to

---

```
to open c:\test47_location_b\test47.txt to show does not exist: & echo. & type
c:\test47_location_b\test47.txt & echo. & echo Create c:\test47_location_b\test47.txt
with the text "test47_location_b" followed by a carriage return, i.e., 0x0D 0x0A: &
echo. & mkdir c:\test47_location_b & echo. & echo test47_location_b>
c:\test47_location_b\test47.txt & echo. & echo verify that
c:\test47_location_b\test47.txt created by reading contents of file: & echo. & type
c:\test47_location_b\test47.txt
```

79. Attached to this declaration as ATTACHMENT 68 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset
date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t &
time /t
```

80. Attached to this declaration as ATTACHMENT 69 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for
c:\test47_location_b\test47.txt... & echo. & echo File creation date: & dir
c:\test47_location_b\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test47_location_b\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)"
```

81. Attached to this declaration as ATTACHMENT 70 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test47_location_b\test47.txt... & echo.
& c:\gpg\gpg.exe --print-md sha256 c:\test47_location_b\test47.txt
```

82. Attached to this declaration as ATTACHMENT 71 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

c:\test47_location_a\"[83] (57) I used the "dir" command to list the "creation date" and "last modified date" of "c:\test47_location_a\test47.txt," as explained in ¶ Nos. 7 and 8,[84] and (58) to show that "c:\test47_location_a\test47.txt" is still the same file prior to it being moved from "c:\test47_location_b\test47.txt," I used GPG to take a SHA-256 hash digest "fingerprint" of "c:\test47_location_a\test47.txt," as explained in ¶ Nos. 11, 12, and 13 above.[85]

16. Using the data collected during my tests explained in ¶ No. 15 above, I was able

---

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Delete
the file c:\test47_location_a\test47.txt... & echo. & del
c:\test47_location_a\test47.txt & echo. & echo Attempt to open
c:\test47_location_a\test47.txt to show does not exist: & echo. & type
c:\test47_location_a\test47.txt
```

83. Attached to this declaration as ATTACHMENT 72 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within
15 seconds of deleting c:\test47_location_a\test47.txt, move
c:\test47_location_b\test47.txt to c:\test47_location_a\ ... & echo. & move
c:\test47_location_b\test47.txt c:\test47_location_a\ & echo. & echo verify that
c:\test47_location_b\test47.txt was moved to c:\test47_location_a\ by reading
contents of file: & echo. & type c:\test47_location_a\test47.txt
```

84. Attached to this declaration as ATTACHMENT 73 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read
"creation date" and "last modified date" file properties for
c:\test47_location_a\test47.txt... & echo. & echo File creation date: & dir
c:\test47_location_a\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)" & echo. & echo File last modified date: & dir
c:\test47_location_a\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume"
/c:".*Dir(s)"
```

85. Attached to this declaration as ATTACHMENT 74 is a screenshot of the command-prompt window showing the output that resulted from use of the command. The precise command entered at the command-prompt was as follows:

```
echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt... & echo.
& c:\gpg\gpg.exe --print-md sha256 c:\test47_location_a\test47.txt
```

to make three overall conclusions: (1) an NTFS file "last access date" property is a reliable record of the last date/time a file was opened and read but an inapplicable/irrelevant record to "period" a file, (2) an NTFS file "creation date" property is an unreliable record to "period" a file, and (3) an NTFS file "last modified date" property is the absolute file creation date/time of a file and, therefore, a reliable record to "period" a file. My three overall conclusions are supported by the evidence collected during my tests explained in ¶ No. 15 above, which is placed into context in ¶ No. 17 (discussing "last access date") and ¶ Nos. 18-22 (discussing "creation date" and "last modified date").

17. Part one of my test, *i.e.*, ¶ No. 15, steps 1-8, consisted of testing the NTFS file "last access date" to see if it can be relied upon to determine the temporal origin of an NTFS file. On January 3, 2013, 12:33pm, I created the NTFS file, "c:\test29.txt,"[86] and noted that NTFS set the "creation date," "last modified date," and "last access date" all to January 3, 2013, 12:33pm.[87] I then took a "fingerprint" of "c:\test29.txt"[88] so that any changes to the file will be detectable. At this point during my test, it appeared as if all three file date properties were sufficient to "period" a file, *i.e.*, they all reflected the absolute temporal origin of the data content of the file. However, after opening and reading "c:\test29.txt" on the pseudo-date of January 4, 2013, 12:37pm,[89] the "last access date" of the file updated to January 4, 2013, 12:37pm[90] while the file "creation date" and "last modified date" continued

86. *See* ATTACHMENT 17.
87. *See* ATTACHMENT 18.
88. *See* ATTACHMENT 19.
89. *See* ATTACHMENT 21.
90. *See* ATTACHMENT 23.

to reflect the absolute temporal origin of the file,[91] *i.e.*, January 3, 2013, 12:33pm. After taking a final "fingerprint" of "c:\test29.txt,"[92] I was able to confirm that the file was still the same file (both post and pre access) based on its data content.[93] Because the "last access date" property of the file changed when the file was accessed, I concluded (1) file "last access date" properties are not reliable to "period" NTFS files, and (2) file "last access date" properties are reliable to determine when NTFS files were last opened and read—albeit within an approximate "less than or equal to" 60 minute "forward looking" margin of error.[94]

18. Part two of my test, *i.e.*, ¶ No. 15, steps 9-16, consisted of testing the NTFS file "creation date" and "last modified date" to see if either can be relied upon to determine the temporal origin of an NTFS file. On January 3, 2013, 12:42pm, I created the NTFS file, "c:\test30.txt,"[95] and noted that NTFS set both the "creation date" and "last modified date" to January 3, 2013, 12:42pm.[96] I then took a "fingerprint" of "c:\test30.txt"[97] so that any changes to the file will be detectable. At this point during my test, it appeared as if both the file "creation date" and "last modified date" were sufficient to "period" an NTFS file, *i.e.*, they both reflected the absolute temporal origin of the data content of the file. However, after opening and writing additional data to "c:\test30.txt" on the pseudo-date of January 4, 2013,

91. *See id.*
92. *See* ATTACHMENT 24.
93. *Compare* "fingerprint" at ATTACHMENT 19 *with* "fingerprint" at ATTACHMENT 24.
94. *See* ¶ No. 5, *supra*, for further explanation.
95. *See* ATTACHMENT 25.
96. *See* ATTACHMENT 26.
97. *See* ATTACHMENT 27.

12:46pm,[98] the "last modified date" of the file updated to January 4, 2013, 12:46pm while the file "creation date" continued to reflect January 3, 2013, 12:42pm.[99] I then took a "fingerprint" of "c:\test30.txt"[100] and determined that it was now an entirely different file (albeit with the same filename) compared to the original version of "c:\test30.txt" having a temporal origin of January 3, 2013, 12:42pm.[101] In other words, after writing additional data to "c:\test30.txt" on January 4, 2013, 12:46pm, the file was thereafter a different file based on its "fingerprint" when compared to the version of "c:\test30.txt" created on January 3, 2013, 12:42pm. At this point, the file "creation date" of "c:\test30.txt" (*i.e.*, January 3, 2013, 12:42pm) no longer reflected the absolute temporal origin of the file based on its data content (*see* "fingerprint") while the file "last modified date" (*i.e.*, January 4, 2013, 12:46pm) did reflect the absolute temporal origin of the file based on its data content (*see* "fingerprint"). In summary, because the "creation date" property of the file failed to change to the date/time the file became a new file based on its "fingerprint," and because the "last modified date" property of the file did change to the actual date/time the file became a new file based on its "fingerprint," I concluded (1) file "creation date" properties are not reliable to "period" NTFS files, and (2) file "last modified date" properties are reliable to "period" NTFS files.

19. Part three of my test, *i.e.*, ¶ No. 15, steps 17-25, consisted of additional testing of NTFS file "creation dates" and "last modified dates" to see if either can be further shown as being a reliable or unreliable means of determining the temporal origin of an NTFS file. On

98. *See* ATTACHMENT 29.

99. *See* ATTACHMENT 31.

100. *See* ATTACHMENT 32.

101. *Compare* "fingerprint" at ATTACHMENT 27 *with* "fingerprint" at ATTACHMENT 32.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

January 3, 2013, 12:50pm, I created the NTFS file, "c:\test31.txt"[102] and noted that NTFS set both the "creation date" and "last modified date" to January 3, 2013, 12:50pm.[103] I then took a "fingerprint" of "c:\test31.txt"[104] so that any changes to the file would be detectable. On the pseudo-date of January 4, 2013, 12:55pm, I created a backup of "c:\test31.txt" by copying it from its location at "c:\test31.txt" to "c:\backup\test31.txt."[105] I then noted that the "creation date" of the copied version of the file (*i.e.*, "c:\backup\test31.txt") updated to January 4, 2013, 12:55pm while its file "last modified date" continued to reflect January 3, 2013, 12:50pm.[106] I then took a "fingerprint" of the copied version of the file (*i.e.*, "c:\backup\test31.txt")[107] and determined that it had not changed its data content on the updated "creation date" (*i.e.*, January 4, 2013, 12:55pm) and instead remained identical to the original version of "test30.txt" (*i.e.*, "c:\test31.txt").[108] At this point, the file "creation date" of "c:\backup\test31.txt" (*i.e.*, January 4, 2013, 12:55pm) no longer reflected the absolute temporal origin of the file based on its data content (*see* "fingerprint") while the file "last modified date" (*i.e.*, January 3, 2013, 12:50pm) did reflect the absolute temporal origin of the file based on its data content (*see* "fingerprint"). In summary, because the simple act of copying a file from one location to another caused the "creation date" property of the copied file to update to a date/time *after* the date/time indicated by the "last modified date" property, and because the "last modified date" property of the

102. *See* ATTACHMENT 33.
103. *See* ATTACHMENT 34.
104. *See* ATTACHMENT 35.
105. *See* ATTACHMENT 37.
106. *See* ATTACHMENT 39.
107. *See* ATTACHMENT 40.
108. *Compare* "fingerprint" at ATTACHMENT 35 *with* "fingerprint" at ATTACHMENT 40.

copied file matched the actual date/time upon which the file's data content came into existence, I concluded (1) file "creation date" properties are not reliable to "period" NTFS files, and (2) file "last modified date" properties are reliable to "period" NTFS files.

20. Part four of my test, *i.e.*, ¶ No. 15, steps 26-34, consisted of additional testing of NTFS file "creation dates" and "last modified dates" to see if either can be further shown as being a reliable or unreliable means of determining the temporal origin of an NTFS file. Specifically, this part of my test provides a scenario where normal use of the Microsoft Windows XP operating system (*i.e.*, the act of creating a file) can result in the overwriting of an NTFS file's "creation date" with a "creation date" belonging to an entirely different NTFS file that no longer exists. This test shows that NTFS file "creation dates" are **corruptible** and, therefore, **wholly unreliable** even if otherwise accepted as being sufficient to "period" an NTFS file. On January 3, 2013, 2:07pm, I created the NTFS file, "c:\test45.txt"[109] and noted that NTFS set both the "creation date" and "last modified date" to January 3, 2013, 2:07pm.[110] I then took a "fingerprint" of "c:\test45.txt"[111] so that any changes to the file would be detectable. On the pseudo-date of January 4, 2013, 2:09pm, I deleted "c:\test45.txt."[112] Immediately after deleting "c:\test45.txt" (*i.e.*, on the pseudo-date of January 4, 2013, 2:09pm), I created a new version of "c:\test45.txt" and filled it with different data content when compared to the January 3, 2013, 2:07pm (deleted) version of "c:\test45.txt."[113] I then noted

109. *See* ATTACHMENT 42.
110. *See* ATTACHMENT 43.
111. *See* ATTACHMENT 44.
112. *See* ATTACHMENT 46.
113. *See* ATTACHMENT 47.

that the new version of "c:\test45.txt," which was created on January 4, 2013, 2:09pm, had its "last modified date" set to January 4, 2013, 2:09pm but its "creation date" was now set to January 3, 2013, 2:07pm,[114] *i.e.*, Microsoft Windows XP gave it the "creation date" belonging to the **old, deleted version** of "c:\test45.txt" that was no longer in existence. I then took a "fingerprint" of the new version of "c:\test45.txt"[115] and determined that it was an entirely different file (albeit with the same filename) compared to the original, deleted version of "c:\test45.txt."[116] At this point, the file "creation date" of "c:\test45.txt" (*i.e.*, January 3, 2013, 2:07pm) did not reflect the absolute temporal origin of the file based on its data content while the file "last modified date" (*i.e.*, January 4, 2013, 2:09pm) did reflect the absolute temporal origin of the file based on its data content. In summary, because the simple act of creating a new file using the filename of a previously deleted file resulted in the "creation date" property of the new file being set to the "creation date" property belonging to the previously deleted file, and because the "last modified date" property of the new file matched the actual date/time upon which the new file's data content came into existence, I concluded (1) file "creation date" properties are not reliable to "period" NTFS files, and (2) file "last modified date" properties are reliable to "period" NTFS files.

21. Part five of my test, *i.e.*, ¶ No. 15, steps 35-46, consisted of additional testing of NTFS file "creation dates" and "last modified dates" to see if either can be further shown as being a reliable or unreliable means of determining the temporal origin of an NTFS file. Specifically, this part of my test provides an additional scenario where normal use of the

114. *See* ATTACHMENT 49.

115. *See* ATTACHMENT 50.

116. *Compare* "fingerprint" at ATTACHMENT 44 *with* "fingerprint" at ATTACHMENT 50.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

Microsoft Windows XP operating system (*i.e.*, the act of renaming a file) can result in the overwriting of an NTFS file's "creation date" with a "creation date" belonging to an entirely different NTFS file that no longer exists. This test shows that NTFS file "creation dates" are **corruptible** and, therefore, **wholly unreliable** even if otherwise accepted as being sufficient to "period" an NTFS file. On January 3, 2013, 2:21pm, I created the NTFS file, "c:\test46a.txt"[117] and noted that NTFS set both the "creation date" and "last modified date" to January 3, 2013, 2:21pm.[118] I then took a "fingerprint" of "c:\test46a.txt"[119] so that any changes to the file would be detectable. On the pseudo-date of January 4, 2013, 2:21pm, I created the NTFS file, "c:\test46b.txt"[120]—but filled it with different data content when compared to "c:\test46a.txt"—and noted that NTFS set both the "creation date" and "last modified date" to January 4, 2013, 2:21pm.[121] I then took a "fingerprint" of "c:\test46b.txt"[122] so that any changes to the file would be detectable. On the January 3, 2013, 2:22pm, I deleted "c:\test46a.txt."[123] Immediately after deleting "c:\test46a.txt" (*i.e.*, on January 3, 2013, 2:22pm), I renamed "c:\test46b.txt" to "c:\test46a.txt."[124] I then noted that "c:\test46a.txt" (previously named "c:\test46b.txt"), which was created on January 4, 2013, 2:21pm, had its "last modified date" set to January 4, 2013, 2:21pm but its "creation date" was

117. *See* ATTACHMENT 51.
118. *See* ATTACHMENT 52.
119. *See* ATTACHMENT 53.
120. *See* ATTACHMENT 55.
121. *See* ATTACHMENT 57.
122. *See* ATTACHMENT 58.
123. *See* ATTACHMENT 59.
124. *See* ATTACHMENT 60.

now set to January 3, 2013, 2:21pm,[125] *i.e.*, Microsoft Windows XP gave it the "creation date" belonging to the **old, deleted** "c:\test46a.txt" that was no longer in existence. I then took a "fingerprint" of "c:\test46a.txt"[126] and determined (1) it was still the same file compared to when it was named "c:\test46b.txt,"[127] and (2) it was an entirely different file (albeit with the same filename) compared to the deleted "c:\test46a.txt."[128] At this point, the file "creation date" of "c:\test46a.txt" (*i.e.*, January 3, 2013, 2:21pm) did not reflect the absolute temporal origin of the file based on its data content while the file "last modified date" (*i.e.*, January 4, 2013, 2:21pm) did reflect the absolute temporal origin of the file based on its data content. In summary, because the simple act of renaming a file to the filename of a previously deleted file resulted in the "creation date" property of the file (in its renamed state) being set to the "creation date" property belonging to the previously deleted file, and because the "last modified date" property of the file (in its renamed state) matched the actual date/time upon which the file's data content came into existence, I concluded (1) file "creation date" properties are not reliable to "period" NTFS files, and (2) file "last modified date" properties are reliable to "period" NTFS files.

22. Part six of my test, *i.e.*, ¶ No. 15, steps 47-58, consisted of additional testing of NTFS file "creation dates" and "last modified dates" to see if either can be further shown as being a reliable or unreliable means of determining the temporal origin of an NTFS file. Specifically, this part of my test provides an additional scenario where normal use of the

125. *See* ATTACHMENT 61.

126. *See* ATTACHMENT 62.

127. *Compare* "fingerprint" at ATTACHMENT 58 *with* "fingerprint" at ATTACHMENT 62.

128. *Compare* "fingerprint" at ATTACHMENT 53 *with* "fingerprint" at ATTACHMENT 62.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

Microsoft Windows XP operating system (*i.e.*, the act of moving a file) can result in the overwriting of an NTFS file's "creation date" with a "creation date" belonging to an entirely different NTFS file that no longer exists. This test shows that NTFS file "creation dates" are **corruptible** and, therefore, **wholly unreliable** even if otherwise accepted as being sufficient to "period" an NTFS file. On January 3, 2013, 2:40pm, I created the NTFS file, "test47.txt" in the folder "c:\test47_location_a\"[129] and noted that NTFS set both the "creation date" and "last modified date" to January 3, 2013, 2:40pm.[130] I then took a "fingerprint" of "c:\test47_location_a\test47.txt"[131] so that any changes to the file would be detectable. On the pseudo-date of January 4, 2013, 2:41pm, I created the NTFS file, "test47.txt" in the folder "c:\test47_location_b\"[132]—but filled it with different data content when compared to the version of "test46a.txt" located in "c:\test47_location_a\"—and noted that NTFS set both the "creation date" and "last modified date" to January 4, 2013, 2:41pm.[133] I then took a "fingerprint" of "c:\test47_location_b\test47.txt"[134] so that any changes to the file would be detectable. On the January 3, 2013, 2:41pm, I deleted "c:\test47_location_a\test47.txt."[135] Immediately after deleting "c:\test47_location_a\test47.txt" (*i.e.*, on January 3, 2013, 2:41pm), I moved "c:\test47_location_b\test47.txt" to "c:\test47_location_a\."[136] I then noted that

129. *See* ATTACHMENT 63.
130. *See* ATTACHMENT 64.
131. *See* ATTACHMENT 65.
132. *See* ATTACHMENT 67.
133. *See* ATTACHMENT 69.
134. *See* ATTACHMENT 70.
135. *See* ATTACHMENT 71.
136. *See* ATTACHMENT 72.

"c:\test47_location_a\test47.txt" (previously located at "c:\test47_location_b\test47.txt"), which was created on January 4, 2013, 2:41pm, had its "last modified date" set to January 4, 2013, 2:41pm but its "creation date" was now set to January 3, 2013, 2:40pm,[137] *i.e.*, Microsoft Windows XP gave it the "creation date" belonging to the **old, deleted** "c:\test47_location_a\test47.txt" that was no longer in existence. I then took a "fingerprint" of "c:\test47_location_a\test47.txt"[138] and determined (1) it was still the same file compared to when it was located at "c:\test47_location_b\test47.txt,"[139] and (2) it was an entirely different file (albeit with the same filename) compared to the deleted "test47.txt" previously located at "c:\test47_location_a\test47.txt."[140] At this point, the file "creation date" of "c:\test47_location_a\test47.txt" (*i.e.*, January 3, 2013, 2:40pm) did not reflect the absolute temporal origin of the file based on its data content while the file "last modified date" (*i.e.*, January 4, 2013, 2:41pm) did reflect the absolute temporal origin of the file based on its data content. In summary, because the simple act of moving a file to a location previously occupied by a file that was deleted resulted in the "creation date" property of the file (while in its new location) being set to the "creation date" property belonging to the previously deleted file, and because the "last modified date" property of the file (while in its new location) matched the actual date/time upon which the file's data content came into existence, I concluded (1) file "creation date" properties are not reliable to "period" NTFS files, and (2) file "last modified date" properties are reliable to "period" NTFS files.

137. *See* ATTACHMENT 73.
138. *See* ATTACHMENT 74.
139. *Compare* "fingerprint" at ATTACHMENT 70 *with* "fingerprint" at ATTACHMENT 74.
140. *Compare* "fingerprint" at ATTACHMENT 65 *with* "fingerprint" at ATTACHMENT 74.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

---

23. Having conducted my tests and concluded that the "last modified date" is the most reliable file date/time property for use to "period" an NTFS file, I next set out to determine whether there is a general consensus regarding my conclusion. By making various observations, I was able to come to two conclusions regarding a general consensus of my prior conclusion regarding the reliability of the NTFS "last modified date" file property compared to the other NTFS file date/time properties: (1) because IRS-CI Agent Daun contaminated the "last access date" file properties for nearly all files on my "T" drive, there must be a general consensus amongst government employee computer forensic investigators that NTFS file "last access dates" are not reliable to "period" NTFS files, and (2) because IRS-CI Agent Daun only preserved the "last modified date" file properties for the files on my "T" drive that were saved into a WinRAR archive during her "live acquisition," there must be a general consensus amongst government employee computer forensic investigators that "last modified dates" are reliable to "period" NTFS files while "creation dates" and "last access dates" are not. My two conclusions are supported by my observations which are explained and placed into context in ¶ Nos. 24-26 below.

24. In a companion declaration titled, *Number of "filesalot.dcv" (i.e., drive "T") files that were opened/read by the government after Rigmaiden's arrest and execution of the N.D.Cal. 08-70460-HRL/PVT warrant*, I explained how the file "last access date" properties for **53,342** of the **53,521** files on my NTFS "T" drive (*i.e.*, mounted "filesalot.dcv") indicate that those files were opened and read on or after August 3, 2008, 6:30pm. *See id.*, ¶ No. 18-19. The government gained access to the "T" drive upon execution of the N.D.Cal. 08-70460-

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

HRL/PVT search warrant at 431 El Camino Real, Apartment No. 1122, Santa Clara, CA 95050 on August 3, 2008, approximately 5:20pm.[141] Considering I was in custody at that time,[142] and considering I have not had access to my computer system since then, any logged file "last access date" having a date/time value after August 3, 2008, 6:30pm[143] can absolutely be said to be a record of the government accessing the corresponding file. More relevant here, any file "last access date" having a date/time value after August 3, 2008, 6:30pm indicates that the previous "last access date" (*i.e.*, corresponding to the last date/time the file was accessed by a non-government actor) was overwritten and otherwise **destroyed** by IRS-CI Agent Daun—the computer forensic investigator responsible for searching my physical computer system. Because, IRS-CI Agent Daun contaminated the "last access date" file properties for nearly all files on my "T" drive, I have come to the conclusion that there must be a general consensus amongst government employee computer forensic investigators that NTFS file "last access dates" are not reliable to "period" NTFS files for the purpose of complying with temporal scope limits contained in a warrant.

25. In order to further establish a general consensus of my prior conclusion regarding

---

141. *See Third Submission Of Consolidated Exhibits Relating To Discovery And Suppression Issues*, EXHIBIT 06 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #863-1) (IRS report of search warrant execution at 431 El Camino Real, Apartment No. 1122, Santa Clara, CA – noting that entry was made at approximately 5:20pm).

142. I was arrested at approximately 4:30pm on August 3, 2008. *See Fourth Submission Of Consolidated Exhibits Relating To Discovery And Suppression Issues*, EXHIBIT 12 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #898-1, p. 71) (Santa Clara police report).

143. When checking for file "last access dates" occurring after my August 3, 2008 arrest, I chose the time of 6:30pm considering a "last access date" for an NTFS file will only update if the file is accessed more than 60 minutes after the previously logged "last access date." *See* ¶ No. 5 above. Therefore, if the government made entry into my residence at 5:20pm, a file "last access date" having a time value greater than or equal to 6:30pm is 10 minutes past the first clock-time value (*i.e.*, 6:20pm) indicative of government personnel file access. In other words, by choosing 6:30pm, the government has no room to argue the technicalities of who may have last accessed any given file.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;

BY: Daniel David Rigmaiden

---

the reliability of the NTFS "last modified date" file property compared to the other NTFS file date/time properties, I set out to check which file date/time properties IRS-CI Agent Daun preserved for the files she saved into the archive file, "T_drive.rar," during her "live acquisition"[144] using the WinRAR[145] software. Using my defense laptop computer on

144. IRS-CI Agent Tracy L. Daun claimed that when she first sat down at my computer she detected that the DriveCrypt encrypted virtual drive, "filesalot.dcv," was mounted as the "T" drive in a decrypted state. *See Third Submission Of Consolidated Exhibits Relating To Discovery And Suppression Issues*, EXHIBIT 01 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #863-1, p. 7). IRS-CI Agent Daun determined that she "needed to attempt to retrieve the data [on the "T" drive] without turning off the computer. [Therefore,] WinRAR was used to archive the files on volume T and was saved to the government hard drive as T_drive.rar." *See id.*, EXHIBIT 01 (U.S. v. Rigmaiden, CR08-814-PHX-DGC, D.Ariz., Dkt. #863-1, p. 9). Via the October 26, 2011 discovery set provided to the defense by the government, I received a copy of the WinRAR file, "T_drive.rar," created by IRS-CI Agent Daun during her "live acquisition."

145. WinRAR is a file archiving program that allows a user to save one or more files into a single archive file having a .zip or .rar file extension. If accessing an archive using WinRAR, a user can extract one or more previously archived files which causes the files to be written to the local file system. WinRAR is available at http://www.rarlabs.com. IRS-CI Agent Daun did not indicate which version of WinRAR she used to create "T_drive.rar," however, the May 9, 2008 web.archive.org page for rarlabs.com indicates that version 3.71 was available as of that date. *See* ATTACHMENT 76. In any event, all newer version of WinRAR have the same relevant features, *i.e.*, the option to preserve file date/time properties.

When archiving files, the WinRAR software allows the user to check boxes in order to specify whether each archived file's "creation date," "last modified date," and/or "last access date" is preserved with the file as it is saved into the archive. The choice made on which file date/time properties to preserve applies to all files in a given archive, different choices cannot be made for each individual file. Attached to this declaration as ATTACHMENT 78 and ATTACHMENT 79 are PDF prints of the WinRAR program help pages titled, "Extraction path and options dialog: advanced options" and "Archive name and parameters dialog: time options." Both help pages explain the procedure for storing date/time file properties. The noted help pages were printed from the WinRAR "Help topics" available on the "Help" drop-down menu within the WinRAR program. Attached to this declaration as ATTACHMENT 80 is a screenshot of the actual WinRAR check boxes that allow for preserving file "creation dates," "last modified dates," and "last access dates" for archived files.

Similarly, when extracting files, the WinRAR software allows the user to check boxes in order to specify whether each extracted file's "creation date," "last modified date," and/or "last access date" is restored—but only if those file properties were saved during the archiving process. Attached to this declaration as ATTACHMENT 81 is a screenshot of the WinRAR check boxes that allow for restoring file "creation dates," "last modified dates," and "last access dates" for extracted files. However, if the file properties were not saved during the archiving process, checking the boxes during the file extraction process will have no effect and the corresponding file properties will update to the date/time the files were extracted to the local file system.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

January 18, 2013, I used WinRAR 3.71[146] to open "T_drive.rar."[147] I then extracted the file "aim_buddy_list.txt" after checking the boxes specifying that the file's "creation date," "last modified date," and "last access date" should be restored to reflect the date/times associated with the original data prior to the archival pocess.[148] After restoring the file to "c:\test6\aim_buddy_list.txt" on my defense laptop computer, I used the "dir" command—as shown in ¶ Nos. 7-9 above—to check the "creation date," "last modified date," and "last access date" file properties for "c:\test6\aim_buddy_list.txt." The results of the "dir" command show that "c:\test6\aim_buddy_list.txt" was created on January 18, 2013, 11:17am, last modified on November 29, 2006, 1:14am, and last accessed on January 18, 2013, 11:17am.[149] Because the "creation date" and "last access date" were set to the date/time I extracted the file to "c:\test6\," it is clear that IRS-CI Agent Daun chose to **only** check the box specifying that the "last modified date" be preserved for each file saved/archived into "T_drive.rar" during her "live acquisition" using WinRAR. In order to verify my findings, I searched for the file "aim_buddy_list.txt" within IRS-CI Agent Daun's list of seized *in-scope* files and found that her extraction process caused the "creation date" and "last access date" of "aim_buddy_list.txt"—as well as all files for that matter—to be set to the date/time she conducted her file extraction process and forensic analysis.[150] Both the "creation date" and

146. In order to make my observations regarding "T_drive.rar," I installed WinRAR version 3.71 to "c:\winrar371" on my defense laptop computer. *See* ATTACHMENT 77.

147. Attached to this declaration as ATTACHMENT 82 is a screenshot of "T_drive.rar" opened with WinRAR.

148. Attached to this declaration as ATTACHMENT 83 , ATTACHMENT 84, and ATTACHMENT 85 is a screenshot of "aim_buddy_list.txt" being extracted to "c:\test6" on my defense laptop computer.

149. Attached to this declaration as ATTACHMENT 86 is a screenshot of the results of using the "dir" command to check the "creation date," "last modified date," and "last access date" file properties for "c:\test6\aim_buddy_list.txt."

150. Attached to this declaration as ATTACHMENT 87 is redacted excerpt from IRS-CI Agent Daun's report (*i.e.*,

---

"last access date" of "aim_buddy_list.txt" are "period" the year 20**09**, *i.e.*, dates well after the temporal origin of the file.[151] Additionally, the "last modified date" of IRS-CI Agent Daun's extracted copy of "aim_buddy_list.txt" *was* preserved and set to the absolute file creation date/time of November 29, 2006, 1:14am.

26. When IRS-CI Agent Daun created "T_drive.rar," she had the option of saving the "creation date," "last modified date" and "last access date" for each archived file. However, IRS-CI Agent Daun chose to only save each file's "last modified date" and chose not to save "creation dates" and "last access dates." Because IRS-CI Agent Daun decided to only preserve the "last modified date" file properties for the files she archived from the "T" drive into "T_drive.rar," I have come to the conclusion that there must be a general consensus amongst government employee computer forensic investigators that an NTFS file "last modified date" reflects the absolute file creation date of a file and is, therefore, the only reliable file date/time property to "period" NTFS files for the purpose of complying with temporal scope limits contained in a warrant.

* * * * *

27. I declare, certify, verify, and state under penalty of perjury under the laws of the United States of America that the foregoing is true and correct to the best of my knowledge, except as to those matters which are therein stated on information and belief, and, as to those matters, I believe it to be true. *See* 28 U.S.C. § 1746 ("Wherever... any matter is required or

---

"T_drive.rar.pdf") cataloging all in-scope files seized from "T_drive.rar" with the file "aim_buddy_list.txt" (bottom of page No. 2) listed as having a "creation date" of May 20, 2009, 2:34pm, a "last modified date" of November 29, 2006, 1:14am, and a "last access date" of August 14, 2009, 12:56pm.

151. As previously discussed, only the government has had access to my computer system (which includes the file, "aim_buddy_list.txt") since the date/time of my arrest on August 3, 20**08**.

DECLARATION UNDER PENALTY OF PERJURY

RE: File modified dates are the most accurate Microsoft Windows NTFS file property to period a file;
BY: Daniel David Rigmaiden

---

permitted to be supported, evidenced, established, or proved by the sworn... affidavit, in writing of the person making the same [], such matter may, with like force and effect, be supported, evidenced, established, or proved by the unsworn declaration..., in writing of such person which is subscribed by him, as true under penalty of perjury, and dated..."); 18 U.S.C. § 1621 ("Whoever... in any declaration... under penalty of perjury as permitted under section 1746 of title 28, United States Code, willfully subscribes as true any material matter which he does not believe to be true... is guilty of perjury and shall, except as otherwise expressly provided by law, be fined under this title or imprisoned not more than five years, or both....").

Executed on January 30, 2013, in Florence, Arizona, United States of America.

Daniel David Rigmaiden

Daniel Rigmaiden

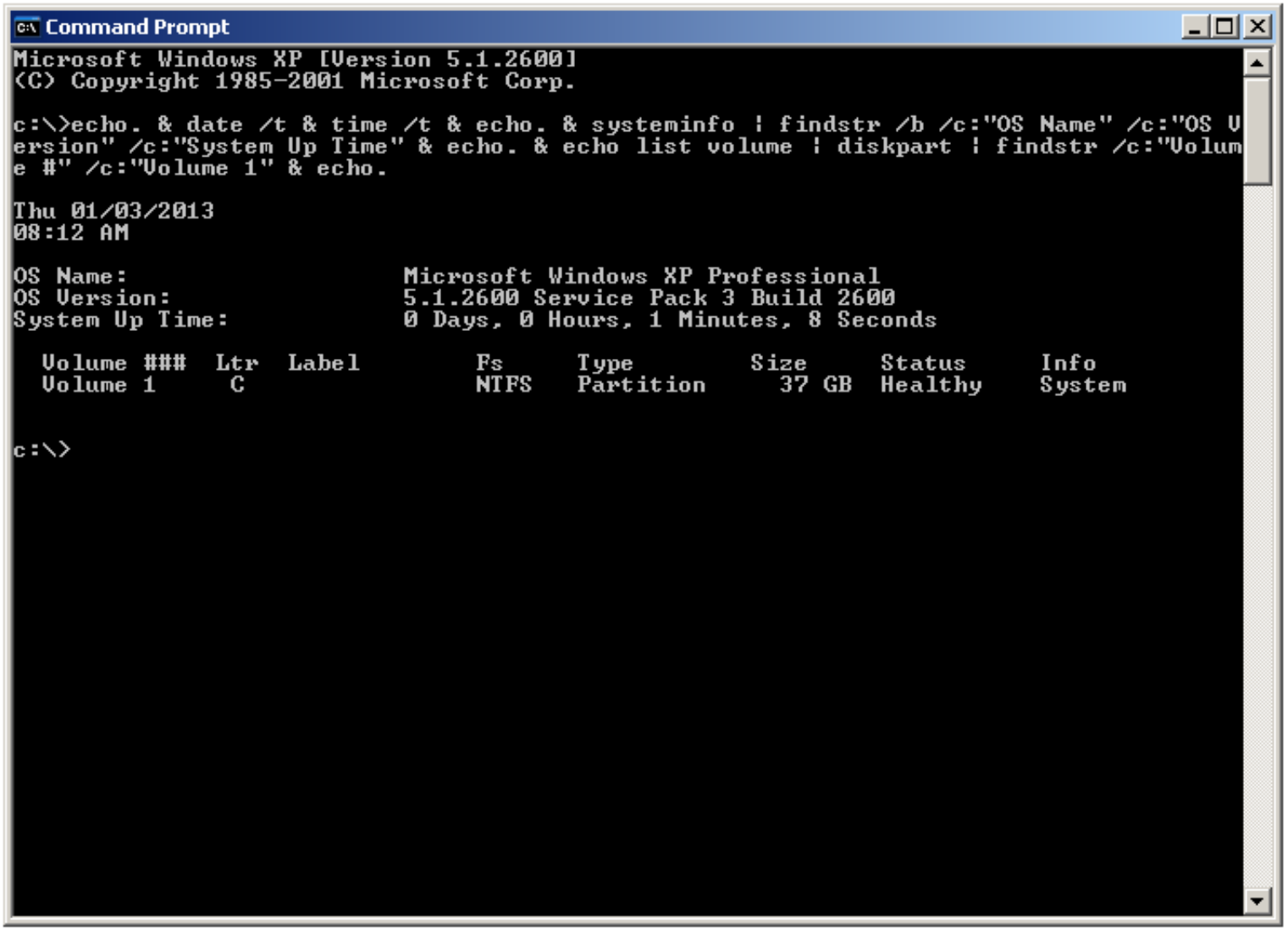
Daniel Rigmaiden
Agency # 10966111
CCA-CADC
PO Box 6300
Florence, AZ 85132

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 01


Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & date /t & time /t & echo. & systeminfo | findstr /b /c:"OS Name" /c:"OS Version" /c:"System Up Time" & echo. & echo list volume | diskpart | findstr /c:"Volume #" /c:"Volume 1" & echo.
Thu 01/03/2013
08:12 AM
OS Name: Microsoft Windows XP Professional
OS Version: 5.1.2600 Service Pack 3 Build 2600
System Up Time: 0 Days, 0 Hours, 1 Minutes, 8 Seconds
Volume ### Ltr Label Fs Type Size Status Info
Volume 1 C NTFS Partition 37 GB Healthy System
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 02

Command shell overview

The command shell is a separate software program that provides direct communication between the user and the operating system. The non-graphical command shell user interface provides the environment in which you run character-based applications and utilities. The command shell executes programs and displays their output on the screen by using individual characters similar to the MS-DOS command interpreter Command.com. The Windows XP command shell uses the command interpreter Cmd.exe, which loads applications and directs the flow of information between applications, to translate user input into a form that the operating system understands.

You can use the command shell to create and edit batch files (also called scripts) to automate routine tasks. For example, you can use scripts to automate the management of user accounts or nightly backups. You can also use the Windows Script Host, CScript.exe, to run more sophisticated scripts in the command shell. You can perform operations more efficiently by using batch files than you can by using the user interface. Batch files accept all commands that are available at the command line. For more information about batch files and scripting, see Using batch files.

You can customize the command prompt window for easier viewing and to increase control over how you run programs. For more information about customizing the command prompt window, see To configure the command prompt.

Using command syntax

Syntax appears in the order in which you must type a command and any parameters that follow it. The following example of the **xcopy** command illustrates a variety of syntax text formats:

xcopy *Source* [*Destination*] [**/w**] [**/p**] [**/c**] [**/v**] [**/q**] [**/f**] [**/l**] [**/g**] [**/d**[**:***mm-dd-yyyy*]] [**/u**] [**/i**] [**/s** [**/e**]] [**/t**] [**/k**] [**/r**] [**/h**] [{**/a**|**/m**}] [**/n**] [**/o**] [**/x**] [**/exclude:***file1*[**+**[*file2*]][**+**[*file3*]] [{**/y**|**/-y**}] [**/z**]

The following table explains how to interpret the different text formats.

Formatting legend

Format	Meaning
Italic	Information that the user must supply
Bold	Elements that the user must type exactly as shown
Ellipsis (...)	Parameter that can be repeated several times in a command line
Between brackets ([])	Optional items
Between braces ({}); choices separated by pipe (|). Example: {even|odd}	Set of choices from which the user must choose only one
`Courier font`	Code or program output

Using multiple commands and conditional processing symbols

You can run multiple commands from a single command line or script using conditional processing symbols. When you run multiple commands with conditional processing symbols, the commands to the right of the conditional processing symbol act based upon the results of the command to the left of the conditional processing symbol. For example, you might want to run a command only if the previous command fails. Or, you might want to run a command only if the previous command is successful.

You can use the special characters listed in the following table to pass multiple commands.

Character	Syntax	Definition
& [...]	*command1* **&** *command2*	Use to separate multiple commands on one command line. Cmd.exe runs the first command, and then the second command.
&& [...]	*command1* **&&** *command2*	Use to run the command following **&&** only if the command preceding the symbol is successful. Cmd.exe runs the first command, and then runs the second command only if the first command completed successfully.
|| [...]	*command1* **||** *command2*	Use to run the command following **||** only if the command preceding **||** fails. Cmd.exe runs the first command, and then runs the second command only if the first command did not complete

		successfully (receives an error code greater than zero).
( ) […]	**(***command1* **&** *command2***)**	Use to group or nest multiple commands.
; or **,**	*command1 parameter1***;***parameter2*	Use to separate command parameters.

Notes

- The ampersand (&), pipe (|), and parentheses ( ) are special characters that must be preceded by the escape character (^) or quotation marks when you pass them as arguments.
- If a command completes an operation successfully, it returns an exit code of zero (0) or no exit code. For more information about exit codes, see Microsoft Windows Resource Kits.

Nesting command shells

You can nest command shells within Cmd.exe by opening a new instance of Cmd.exe at the command prompt. By default, each instance of Cmd.exe inherits the environment of its parent Cmd.exe application. By nesting instances of Cmd.exe, you can make changes to the local environment without affecting the parent application of Cmd.exe. This allows you to preserve the original environment of Cmd.exe and return to it after you terminate the nested command shell. The changes you make in the nested command shell are not saved.

To nest a command shell, at the command prompt, type:

cmd

A message similar to the following appears:

```
Microsoft (R) Windows XP (TM)
(C) Copyright 1985-2001 Microsoft Corp.
```

To close the nested command shell, type **exit**.

You can localize changes even further in an instance of Cmd.exe (or in a script) by using the **setlocal** and **endlocal** commands. **Setlocal** creates a local scope and **endlocal** terminates the local scope. Any changes made within the **setlocal** and **endlocal** scope are discarded, thereby leaving the original environment unchanged. You can nest these two commands to a maximum of 32 levels. For more information about the **setlocal** and **endlocal** commands, see Setlocal and Endlocal.

Using environment variables with Cmd.exe

The Cmd.exe command-shell environment is defined by variables that determine the behavior of the command shell and the operating system. You can define the behavior of the command-shell environment or the entire operating system environment by using two types of environment variables, system and local. System environment variables define the behavior of the global operating system environment. Local environment variables define the behavior of the environment of the current instance of Cmd.exe.

System environment variables are preset in the operating system and available to all Windows XP processes. Only users with administrative privileges can change system variables. These variables are most commonly used in logon scripts.

Local environment variables are only available when the user for whom they were created is logged on to the computer. Local variables set in the **HKEY_CURRENT_USER** hive are valid only for the current user, but define the behavior of the global operating system environment.

The following list describes the various types of variables in descending order of precedence:

1. Built-in system variables
2. System variables found in the **HKEY_LOCAL_MACHINE** hive
3. Local variables found in the **HKEY_CURRENT_USER** hive
4. All environment variables and paths set in the Autoexec.bat file
5. All environment variables and paths set in a logon script (if present)
6. Variables used interactively in a script or batch file

In the command shell, each instance of Cmd.exe inherits the environment of its parent application. Therefore, you can change the variables in the new Cmd.exe environment without affecting the environment of the parent application.

The following table lists the system and local environment variables for Windows XP.

Variable	Type	Description
%ALLUSERSPROFILE%	Local	Returns the location of the All Users Profile.

%APPDATA%	Local	Returns the location where applications store data by default.
%CD%	Local	Returns the current directory string.
%CMDCMDLINE%	Local	Returns the exact command line used to start the current Cmd.exe.
%CMDEXTVERSION%	System	Returns the version number of the current Command Processor Extensions.
%COMPUTERNAME%	System	Returns the name of the computer.
%COMSPEC%	System	Returns the exact path to the command shell executable.
%DATE%	System	Returns the current date. Uses the same format as the **date /t** command. Generated by Cmd.exe. For more information about the **date** command, see Date.
%ERRORLEVEL%	System	Returns the error code of the most recently used command. A non zero value usually indicates an error.
%HOMEDRIVE%	System	Returns which local workstation drive letter is connected to the user's home directory. Set based on the value of the home directory. The user's home directory is specified in Local Users and Groups.
%HOMEPATH%	System	Returns the full path of the user's home directory. Set based on the value of the home directory. The user's home directory is specified in Local Users and Groups.
%HOMESHARE%	System	Returns the network path to the user's shared home directory. Set based on the value of the home directory. The user's home directory is specified in Local Users and Groups.
%LOGONSEVER%	Local	Returns the name of the domain controller that validated the current logon session.
%NUMBER_OF_PROCESSORS%	System	Specifies the number of processors installed on the computer.
%OS%	System	Returns the operating system name. Windows 2000 displays the operating system as Windows_NT.
%PATH%	System	Specifies the search path for executable files.
%PATHEXT%	System	Returns a list of the file extensions that the operating system considers to be executable.
%PROCESSOR_ARCHITECTURE%	System	Returns the chip architecture of the processor. Values: x86, IA64.
%PROCESSOR_IDENTFIER%	System	Returns a description of the processor.
%PROCESSOR_LEVEL%	System	Returns the model number of the processor installed on the computer.
%PROCESSOR_REVISION%	System	Returns the revision number of the processor.
%PROMPT%	Local	Returns the command prompt settings for the current interpreter. Generated by Cmd.exe.
%RANDOM%	System	Returns a random decimal number between 0 and 32767. Generated by Cmd.exe.
%SYSTEMDRIVE%	System	Returns the drive containing the Windows XP root directory (that is, the system root).
%SYSTEMROOT%	System	Returns the location of the Windows XP root directory.
%TEMP% and %TMP%	System and User	Returns the default temporary directories that are used by applications available to users who are currently logged on. Some applications require TEMP and others require TMP.
%TIME%	System	Returns the current time. Uses the same format as the **time /t** command. Generated by Cmd.exe. For more information about the **time** command, see Time.
%USERDOMAIN%	Local	Returns the name of the domain that contains the user's account.
%USERNAME%	Local	Returns the name of the user who is currently logged on.
%USERPROFILE%	Local	Returns the location of the profile for the current user.
%WINDIR%	System	Returns the location of the operating system directory.

Setting environment variables

Use the **set** command to create, change, delete, or display environment variables. The **set** command alters variables in the current shell environment only.

To view a variable, at a command prompt, type:

set *VariableName*

To add a variable, at a command prompt, type:

set variablename=*value*

To delete a variable, at a command prompt, type:

set *VariableName***=**

You can use most characters as variable values, including white space. If you use the special characters <, >, |, &, or ^, you must precede them with the escape character (^) or quotation marks. If you use quotation marks, they are included as part of the value because everything following the equal sign is taken as the value. Consider the following examples:

- To create the variable value **new&name**, type:

  set varname=new^&name

- To create the variable value **"new&name"**, type:

  set varname="new&name"

- If you type **set varname=new&name** at the command prompt, an error message similar to the following appears:

  ```
  "'name' is not recognized as an internal or external command, operable program or batch file."
  ```

Variable names are not case-sensitive. However, **set** displays the variable exactly as you typed it. You can combine uppercase and lowercase letters in your variable names to make your code more readable (for example, UserName).

Notes

- The maximum individual environment variable size is 8192bytes.
- The maximum total environment variable size for all variables, which includes variable names and the equal sign, is 65,536KB.

Substituting environment variable values

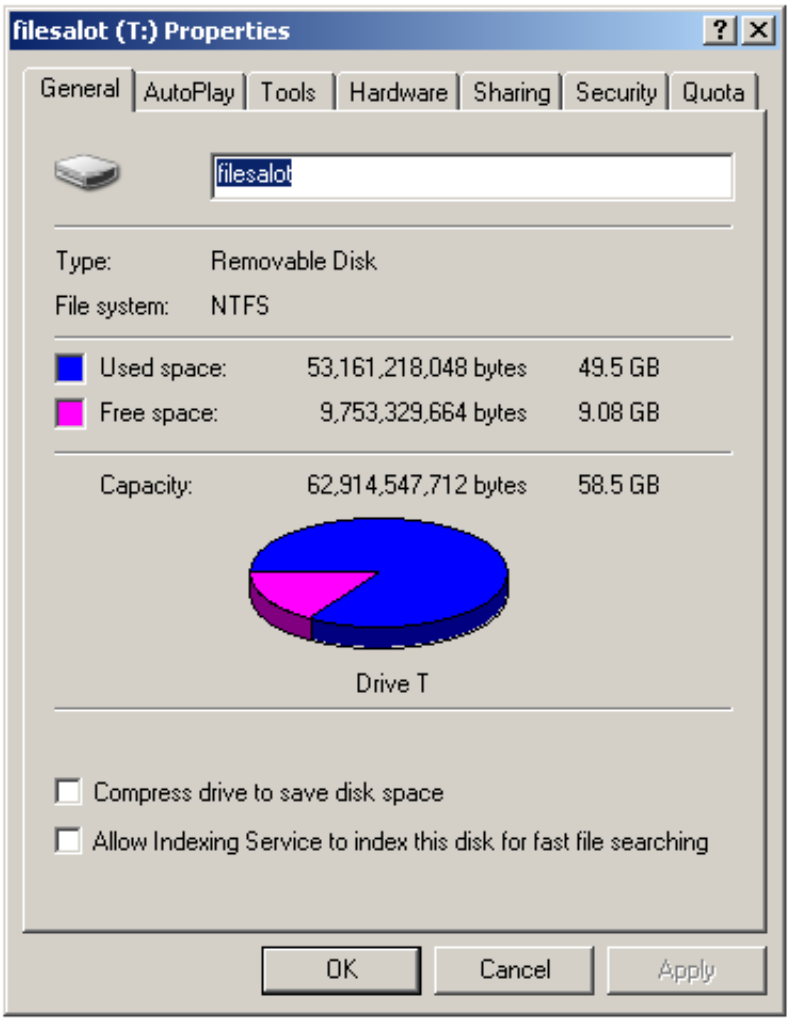
To enable the substitution of variable values at the command line or in scripts, enclose the variable name in percent signs (that is, **%***variablename***%**). By using percent signs, you ensure that Cmd.exe references the variable values instead of making a literal comparison. After you define variable values for a variable name, enclose the variable name in percent signs. Cmd.exe searches for all instances of the variable name and replaces it with the defined variable value. For example, if you create a script that contains different values (for example, user names) and you want to define the USERNAME environment variable for each user with these values, you can write one script using the variable USERNAME enclosed in percent signs. When you run this script, Cmd.exe replaces %USERNAME% with the variable values, which eliminates the need to perform this task manually for each user. Variable substitution is not recursive. Cmd.exe checks variables once. For more information about variable substitution, see For and Call.

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 03


filesalot (T:) Properties
General
AutoPlay
Tools
Hardware
Sharing
Security
Quota
filesalot
Type: Removable Disk
File system: NTFS
Used space: 53,161,218,048 bytes 49.5 GB
Free space: 9,753,329,664 bytes 9.08 GB
Capacity: 62,914,547,712 bytes 58.5 GB
Drive T
Compress drive to save disk space
Allow Indexing Service to index this disk for fast file searching
OK
Cancel
Apply

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 04

NTFS

Some of the features you can use when you choose NTFS are:

- Domains, which are part of Active Directory, and which you can use to fine-tune security options while keeping administration simple. Domain controllers require NTFS.
- File encryption, which greatly enhances security.
- Permissions that can be set on individual files and on folders.
- Sparse files. These are very large files created by applications in such a way that only limited disk space is needed. That is, NTFS allocates disk space only to the portions of a file that are written to.
- Remote Storage, which provides an extension to your disk space by making removable media (such as tapes) more accessible.
- Recovery logging of NTFS meta data, which helps you restore information quickly in the event of power failure or other system problem. This allows access to the volume immediately after restarting the computer without waiting for chkdsk.exe to run.
- Disk quotas, which you can use to monitor and control the amount of disk space used by individual users.
- Better scalability to large drives. The maximum drive size for NTFS is much greater than that for FAT, and as drive size increases, performance with NTFS doesn't degrade as it does with FAT.

The Setup program makes it easy to convert your partition to the new version of NTFS, even if it used FAT or FAT32 before. This kind of conversion keeps your files intact (unlike formatting a partition).

Setup begins by checking the existing file system. If it is NTFS, conversion is not necessary. If it is FAT or FAT32, Setup gives you the choice of converting to NTFS. If you don't need to keep your files intact and you have a FAT or FAT32 partition, it is recommended that you *format* the partition with NTFS rather than *converting* from FAT or FAT32. Formatting a partition erases all data on the partition and allows you to start fresh with a clean drive.

However, it is still advantageous to use NTFS, regardless of whether the partition was formatted with NTFS or converted. A partition can also be converted after Setup by using Convert.exe. For more information about Convert.exe, after completing Setup, click **Start**, click **Run**, type **cmd**, and then press ENTER. In the command window, type **help convert**, and then press ENTER.

Related Topics

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 05

NTFS file system

An advanced file system that provides performance, security, reliability, and advanced features that are not found in any version of FAT. For example, NTFS guarantees volume consistency by using standard transaction logging and recovery techniques. If a system fails, NTFS uses its log file and checkpoint information to restore the consistency of the file system. In Windows 2000 and Windows XP, NTFS also provides advanced features such as file and folder permissions, encryption, disk quotas, and compression.

See also: FAT32; file allocation table (FAT); file system

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 06

File properties overview

Files and folders have property sheets, which display information such as size, location, and the date the file or folder was created. When you view the properties of a file or folder, you can also get information about the:

- File or folder attributes.
- Type of file.
- Name of the program that opens the file.
- Number of files and subfolders contained in the folder.
- Last time the file was modified or accessed.

Depending on the type of file or folder, you can also view additional information, displayed on the following tabs:

General

Identifies the file type; the program associated with the file; its location and size; and the date it was created, last modified, and last opened.

AutoPlay

Allows you to change the way Windows handles the media files it detects on a device with removable storage, such as a digital camera or a CD-ROM. For example, when Windows detects music tracks on your CD-ROM drive, it can automatically play the tracks or it can allow you to view them in a folder.

Security

Lists other users who can modify, read and execute, view folder contents, write to the file or folder, or have read-only access.

Sharing

Allows you to share the folder with other users and set access permissions for the files it contains.

Custom

Allows you to create new properties for a file that provide additional information about the file. To use the **Custom** tab to create a new property for a file, in the **Name** box, select or type a property name. In the **Type** box, select a property type, such as **Text** or **Date**. In the **Value** box, type a value for the property. For example, you might select **Department** in the **Name** box, **Text** in the **Type** box, and type "Furniture" in the **Value** box.

Customize

Allows you to change the picture that appears on a folder in Thumbnails view, change the folder icon, and choose a new template for a folder. Folder templates contain task and information links for specialized files, such as pictures or music files.

Summary

Lists information about the file, including the title, subject, category, and author.

Shortcut

Lists the shortcut name, target information, and shortcut key. Allows you to choose the way the item is displayed when you open the shortcut: in a standard window, a full screen (maximized), or as a button on the taskbar (minimized). Also allows you to view the shortcut's target, change the icon for the shortcut, and open a shortcut as a different user.

Related Topics

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 07

Fsutil: behavior

Queries the current settings for generating 8.3 character-length file names, allowing extended characters in 8.3 character-length file names on NTFS volumes, updating the last access timestamp on NTFS volumes, how often quota events are written to the system log, and the size of the master file table (MFT) Zone. Enables or disables the use of 8.3 character-length file names, allowing extended characters in 8.3 character-length file names on NTFS volumes, and updating the last access timestamp on NTFS volumes. Enables you to change how often quota events are written to the system log and to change the amount of disk space reserved for the MFT Zone.

Syntax

fsutil behavior query {**disable8dot3**|**allowextchar**|**disablelastaccess**|**quotanotify**|**mftzone**}

fsutil behavior set [{**disable8dot3** {**1**|**0**}|**allowextchar** {**1**|**0**}|**disablelastaccess** {**1**|**0**}|**quotanotify** *frequency*|**mftzone** *value*}]

Parameters

query
Queries the file system behavior parameters.

set
Changes the file system behavior parameters.

disable8dot3 {**1**|**0**}
Disables creation of 8.3 character-length file names on FAT- and NTFS-formatted volumes.

allowextchar {**1**|**0**}
Determines whether characters from the extended character set, including diacritic characters, can be used in 8.3 short file names on NTFS volumes.

disablelastaccess {**1**|**0**}
Determines whether NTFS updates the last access timestamp on each directory when it lists the directories on an NTFS volume.

quotanotify *frequency*
Configures how frequently NTFS quota violations are reported in the system log. Enter a frequency 0 through 4294967295 seconds for how often quota violations are written to the system log. Default is 1 hour (3600 seconds).

mftzone *value*
The master file table (MFT) Zone is a reserved area that enables the MFT to expand as needed, in order to prevent MFT fragmentation. Set the *value* from 1 (default) to 4 (maximum). The *value* is in 8ths of the disk.

Remarks

- The **behavior** subcommand writes changes to the registry, so you must restart the computer for changes to take effect.
- Using **disable8dot3** {**1**|**0**}

  When **disable8dot3** is set to **0**, every time you create a file with a long file name, NTFS creates a second file entry that has an 8.3 character-length file name. When NTFS creates files in a folder, it must look up the 8.3 character-length file names associated with the long file names.

  This parameter updates the **HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisable8dot3NameCreation** registry key.
- Using **allowextchar** {**1**|**0**}

  This parameter updates the **HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsAllowExtendedCharacterIn8dot3Name** registry key.
- Using **disablelastaccess** {**1**|**0**}

  The **disablelastaccess** parameter is designed to reduce the logging impact of updating the last access timestamp on folders and directories. Disabling the Last Access Time improves the speed of folder and file access.

  Each file and folder on an NTFS volume contains an attribute called Last Access Time. This attribute defines when the file or folder was last accessed, such as when a user lists folders, adds files to a folder, reads a file, or makes changes to a file. The most up-to-date Last Access Time is stored in memory and is eventually written to the disk in two different locations. One is within the file's attribute, which is part of its MFT record. The second is in the index of the directory that contains the file.

  The Last Access Time on disk is not always current. This lag occurs because NTFS delays writing the Last Access Time to disk when users or programs perform read-only operations on a file or folder, such as listing the folder's contents or reading (but not changing) a file in the folder. If the Last Access Time is kept current on disk for read operations, all read operations become write operations, which impacts NTFS performance.

  Note that file-based queries of Last Access Time are accurate even if all on-disk values are not current. NTFS returns the correct value on queries because the accurate value is stored in memory.

  NTFS typically updates a file's attribute on disk if the current Last Access Time in memory differs by more than an hour from the Last Access Time stored on disk, or when all in-memory references to that file are gone, whichever is more recent. For example, if a file's current Last Access Time is 1:00 P.M., and you read the file at 1:30 P.M., NTFS does not update the Last Access Time. If you read the file again at 2:00 P.M., NTFS updates the Last Access Time in the file's attribute to reflect 2:00 P.M. because the file's attribute shows 1:00 P.M. and the in-memory Last Access Time shows 2:00 P.M.

NTFS updates the index of the directory that contains the file when NTFS updates the file's Last Access Time and detects that the Last Access Time for the file differs by more than an hour from the Last Access Time stored in the directory's index. This update typically occurs after a program closes the handle used to access a file within the directory. If the user holds the handle open for an extended time, a lag occurs before the change appears in the index entry of the directory.

Note that one hour is the maximum time that NTFS defers updating the Last Access Time on disk. If NTFS updates other file attributes such as Last Modify Time, and a Last Access Time update is pending, NTFS updates the Last Access Time along with the other updates without additional performance impact.

Note that using the **disablelastaccess** parameter can affect programs such as backup and Remote Storage that rely on this feature.

This parameter updates the **HKLM\SYSTEM\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate** registry key.

- Using **mftzone** *value*

  The *value* is an approximation of the initial size of the MFT plus the MFT Zone on a new volume and is set at mount time for each file system. As space on the volume is used, NTFS adjusts the space reserved for future MFT growth. If the MFT Zone is already large, the full MFT Zone size is not reserved again. Since the MFT Zone is based on the contiguous range past the end of the MFT, it shrinks as the space is used.

  The file system does not redetermine the MFT Zone location until the current MFT Zone is completely used. This never occurs on a typical system.

Related Topics

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 08

Dir

Displays a list of a directory's files and subdirectories. Used without parameters, **dir** displays the disk's volume label and serial number, followed by a list of directories and files on the disk, including their names and the date and time each was last modified. For files, **dir** displays the name extension and the size in bytes. **Dir** also displays the total number of files and directories listed, their cumulative size, and the free space (in bytes) remaining on the disk.

Syntax

dir [*Drive***:**][*Path*][*FileName*] [...] [**/p**] [**/q**] [**/w**] [**/d**] [**/a**[[**:**]*attributes*]][**/o**[[**:**]*SortOrder*]] [**/t**[[**:**]*TimeField*]] [**/s**] [**/b**] [**/l**] [**/n**] [**/x**] [**/c**] [**/4**]

Parameters

[*Drive***:**][*Path*]
Specifies the drive and directory for which you want to see a listing.

[*FileName*]
Specifies a particular file or group of files for which you want to see a listing.

/p
Displays one screen of the listing at a time. To see the next screen, press any key on the keyboard.

/q
Displays file ownership information.

/w
Displays the listing in wide format, with as many as five file names or directory names on each line.

/d
Same as **/w** but files are sorted by column.

/a [[**:**] *attributes*]
Displays only the names of those directories and files with the attributes you specify. If you omit **/a**, **dir** displays the names of all files except hidden and system files. If you use **/a** without specifying *attributes*, **dir** displays the names of all files, including hidden and system files. The following list describes each of the values you can use for *attributes*. The colon (:) is optional. Use any combination of these values, and do not separate the values with spaces.

Value	Description
h	Hidden files
s	System files
d	Directories
a	Files ready for archiving
r	Read-only files
-h	Files that are not hidden
-s	Files other than system files
-d	Files only (not directories)
-a	Files that have not changed since the last backup
-r	Files that are not read-only

/o [[**:**]*SortOrder*]
Controls the order in which **dir** sorts and displays directory names and file names. If you omit **/o**, **dir** displays the names in the order in which they occur in the directory. If you use **/o** without specifying *SortOrder*, **dir** displays the names of the directories, sorted in alphabetic order, and then displays the names of files, sorted in alphabetic order. The colon (:) is optional. The following list describes each of the values you can use for *SortOrder*. Use any combination of the values, and do not separate these values with white spaces.

Value	Description
n	In alphabetic order by name
e	In alphabetic order by extension
d	By date and time, earliest first
s	By size, smallest first
g	With directories grouped before files
-n	In reverse alphabetic order by name (Z through A)

-e	In reverse alphabetic order by extension (.ZZZ through .AAA)
-d	By date and time, latest first
-s	By size, largest first
-g	With directories grouped after files

/t [[**:**]*TimeField*]
Specifies which time field to display or use for sorting. The following list describes each of the values you can use for *TimeField*.

Value	Description
c	Creation
a	Last access
w	Last written

/s
Lists every occurrence, in the specified directory and all subdirectories, of the specified file name.

/b
Lists each directory name or file name, one per line, including the file name extension. **/b** does not display heading information or a summary. **/b** overrides **/w**.

/l
Displays unsorted directory names and file names in lowercase. **/l** does not convert extended characters to lowercase.

/n
Displays a long list format with file names on the far right of the screen.

/x
Displays the short names generated for files on NTFS and FAT volumes. The display is the same as the display for **/n**, but short names are displayed after the long name.

/c
Displays the thousand separator in file sizes.

/4
Displays four-digit year format.

/?
Displays help at the command prompt.

Remarks

- Using multiple *filenames*

  You can use multiple *filenames*. Separate file names with spaces, commas, or semicolons. You can use wildcard characters (that is, ? and *) in *FileName* to display a group of files.

- Using wildcards

  You can use wildcards (that is, ? and *) to display a list of a subset of files and subdirectories.

- Specifying file display attributes

  If you use **/a** with more than one value in *attributes*, **dir** displays the names of only those files with all the specified attributes. For example, if you use **/a** with **r** and **-h** for attributes by using either **/a:r-h** or **/ar-h**, **dir** displays only the names of read-only files that are not hidden.

- Specifying file name sorting

  If you specify more than one *SortOrder* value, **dir** sorts the file names by the first criterion first, then by the second criterion, and so on. For example, if you use **/o** with the **e** and **-s** values for *SortOrder* by using either **/o:e-s** or **/oe-s**, **dir** sorts the names of directories and files by extension, with the largest first, and then displays the final result. The alphabetic sorting by extension causes file names with no extensions to appear first, then directory names, and then file names with extensions.

- Using redirection symbols and pipes

  When you use a redirection symbol (>) to send **dir** output to a file or a pipe (|) to send **dir** output to another command, use **/a:-d** and **/b** to list the file names only. You can use *FileName* with **/b** and **/s** to specify that **dir** is to search the current directory and its subdirectories for all file names that match *FileName*. **Dir** lists only the drive letter, directory name, file name, and file name extension, one path per line, for each file name it finds. Before you use a pipe for redirection, you should set the TEMP environment variable in your Autoexec.nt file.

- Presetting **dir** parameters

  You can preset **dir** parameters by including **set** with the DIRCMD environment variable in your Autoexec.nt file. You can use any valid combination of **dir** parameters with **set dircmd**, including the location and name of a file.

  For example, to use the DIRCMD environment variable to set the wide display format (that is, **/w**) as the default format, type the following command in your Autoexec.nt file:

  set dircmd=/w

  For a single use of the **dir** command, you can override a parameter by using the DIRCMD environment variable. To do so, type the parameter that you want to override at the **dir** command prompt, preceding the parameter with a minus sign. For example:

  dir /-w

  To change the DIRCMD default settings, type:

  set=*NewParameter*

  The new default settings are effective for all subsequent **dir** commands until you use **set dircmd** again or until you restart your computer.

  To clear all default settings, type:

  set dircmd=

  To view the current settings of the DIRCMD environment variable, type:

  set

  Set displays a list of environment variables and their settings. For more information about setting environment variables, see Related Topics.

- The **dir** command, with different parameters, is available from the Recovery Console.

Examples

To display all directories, one after the other, in alphabetical order, in wide format and pausing after each screen, make sure that the root directory is the current directory, and then type:

dir /s/w/o/p

Dir lists the name of the root directory, the names of the subdirectories of the root directory, and the names of the files in the root directory, including extensions. Then, **dir** lists the subdirectory names and file names in each subdirectory in the tree.

To alter the preceding example so that **dir** displays the file names and extensions, but omits the directory names, type:

dir /s/w/o/p/a:-d

To print a directory listing, type:

ir > prn

When you specify **prn**, the directory list is sent to the printer attached to the LPT1 port. If your printer is attached to a different port, you must replace **prn** with the name of the correct port.

You can also redirect output of the **dir** command to a file by replacing **prn** with a file name. You can also type a path. For example, to direct **dir** output to the file Dir.doc in the Records directory, type:

dir > \records\dir.doc

If Dir.doc does not exist, **dir** creates it, unless the Records directory does not exist. In that case, the following message appears:

```
File creation error
```

To display a list of all the file names with the .txt extension in all directories on drive C, type:

dir c:*.txt /w/o/s/p

Dir displays, in wide format, an alphabetized list of the matching file names in each directory and pauses each time the screen fills up, until you press a key to continue.

Formatting legend

Format	Meaning
Italic	Information that the user must supply
Bold	Elements that the user must type exactly as shown
Ellipsis (...)	Parameter that can be repeated several times in a command line
Between brackets ([])	Optional items
Between braces ({}); choices separated by pipe (|). Example: {even|odd}	Set of choices from which the user must choose only one
`Courier font`	Code or program output

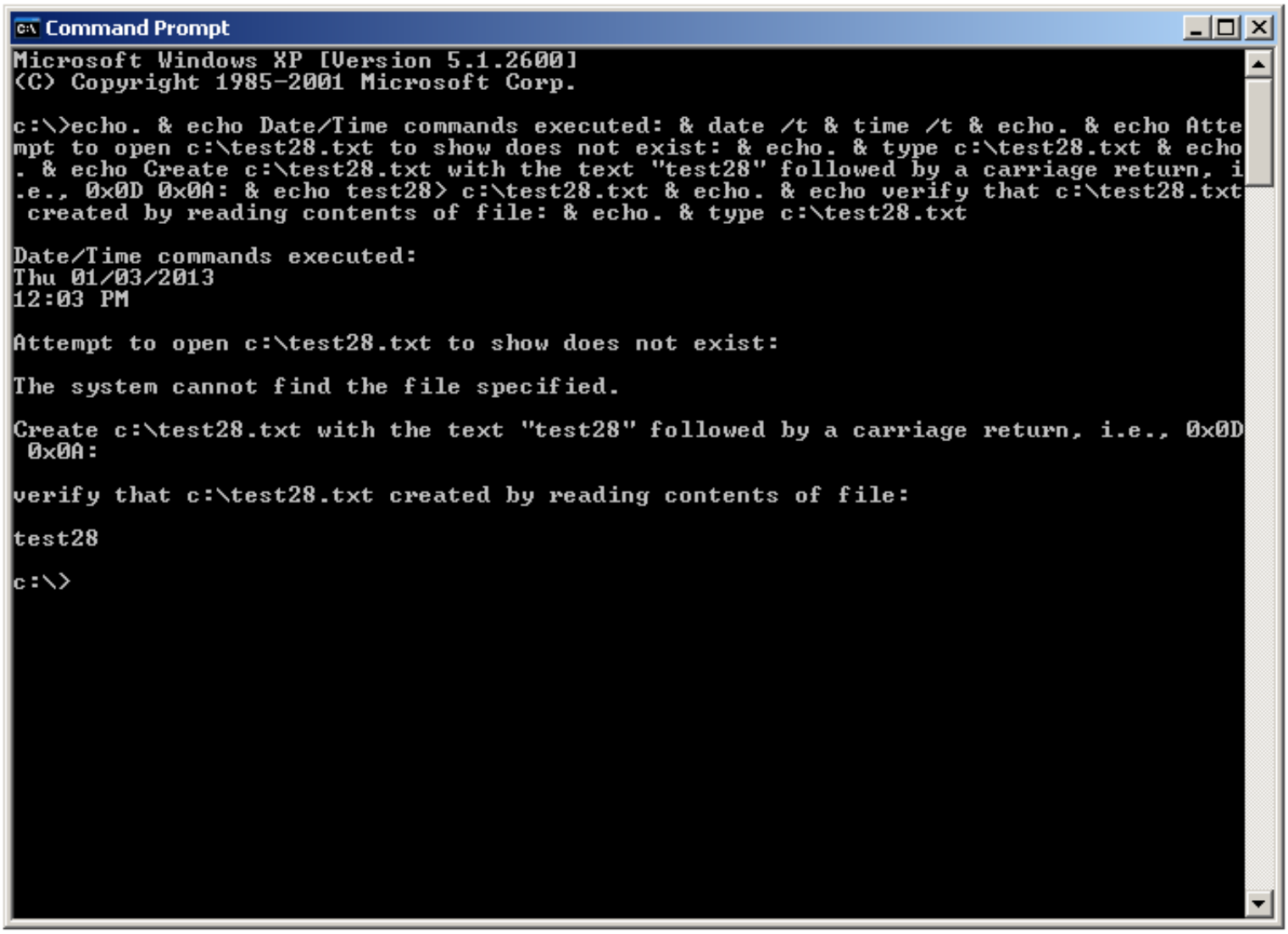
Related Topics

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 09

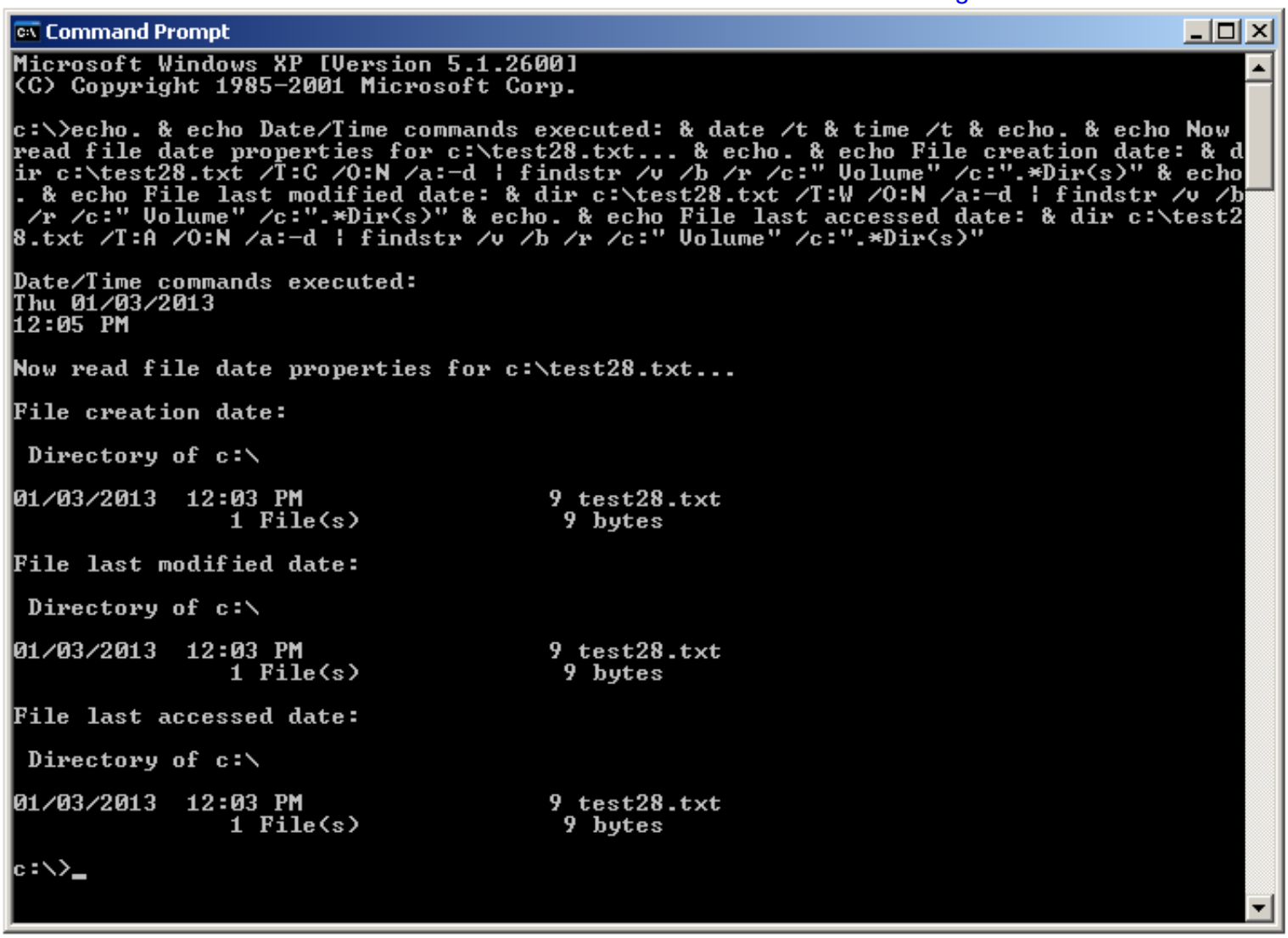

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test28.txt to show does not exist: & echo. & type c:\test28.txt & echo. & echo Create c:\test28.txt with the text "test28" followed by a carriage return, i.e., 0x0D 0x0A: & echo test28> c:\test28.txt & echo. & echo verify that c:\test28.txt created by reading contents of file: & echo. & type c:\test28.txt
Date/Time commands executed:
Thu 01/03/2013
12:03 PM
Attempt to open c:\test28.txt to show does not exist:
The system cannot find the file specified.
Create c:\test28.txt with the text "test28" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test28.txt created by reading contents of file:
test28
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 10

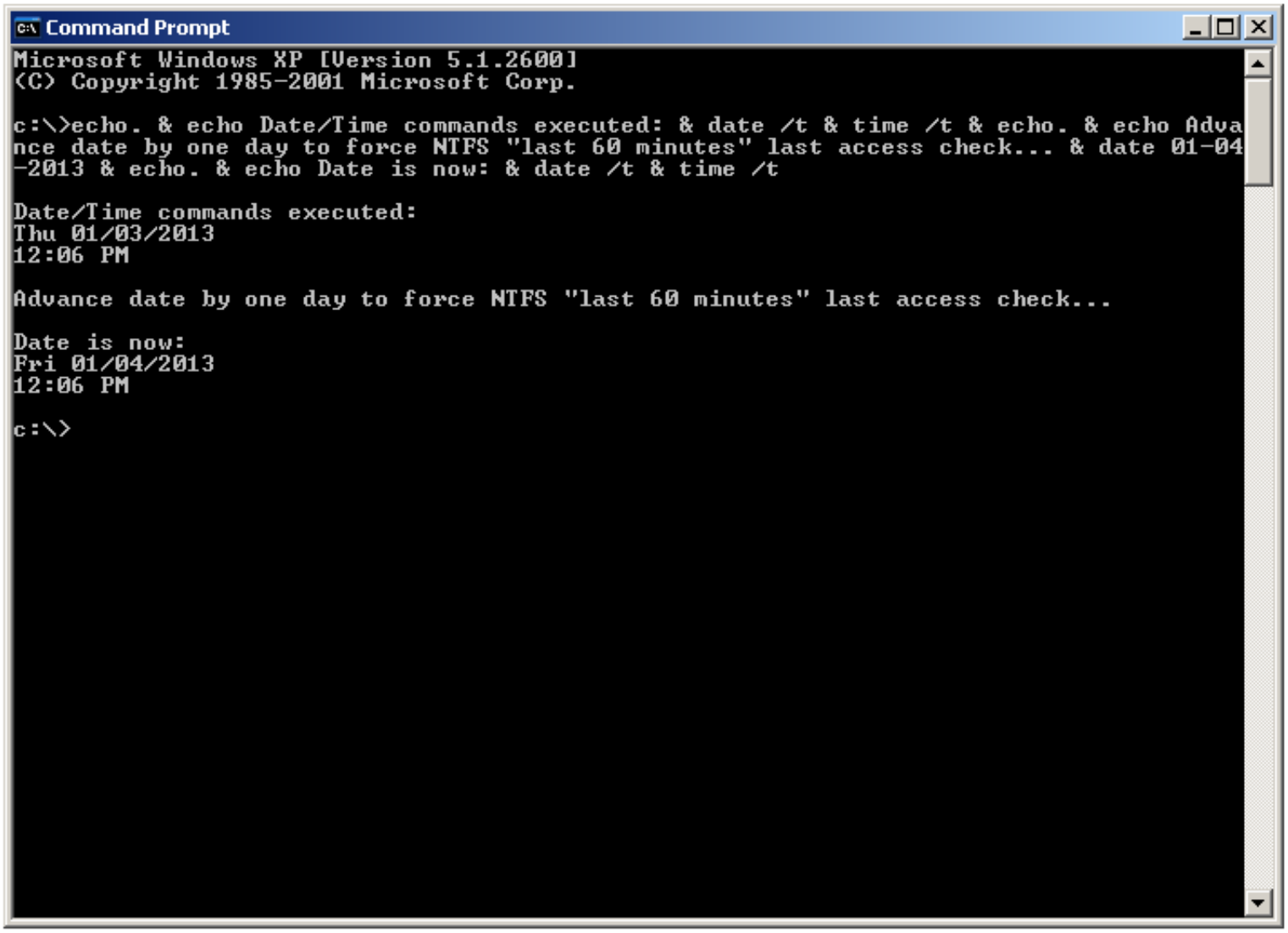

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read file date properties for c:\test28.txt... & echo. & echo File creation date: & dir c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:05 PM
Now read file date properties for c:\test28.txt...
File creation date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last accessed date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 11

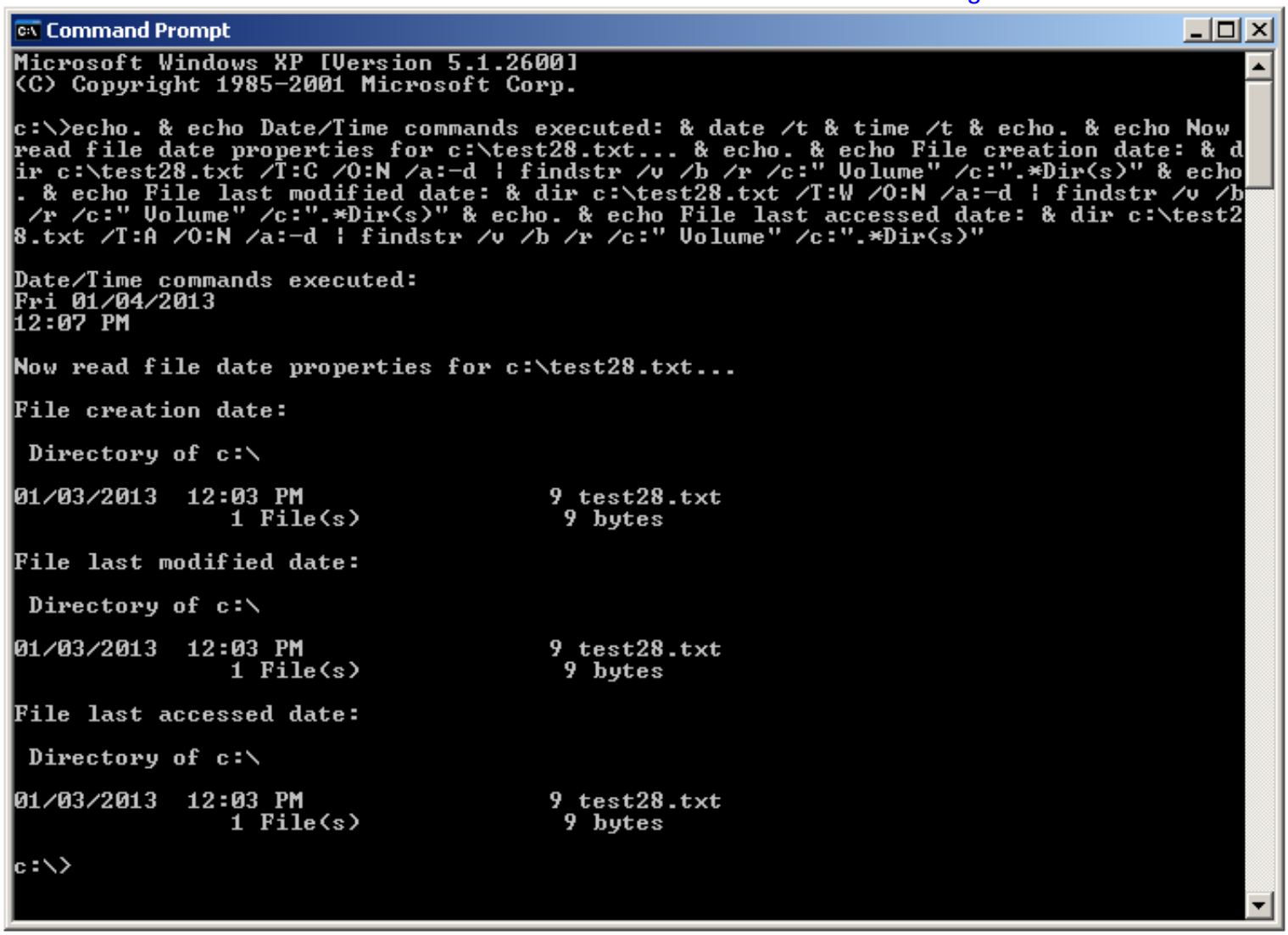

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Adva
nce date by one day to force NTFS "last 60 minutes" last access check... & date 01-04
-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
12:06 PM
Advance date by one day to force NTFS "last 60 minutes" last access check...
Date is now:
Fri 01/04/2013
12:06 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 12

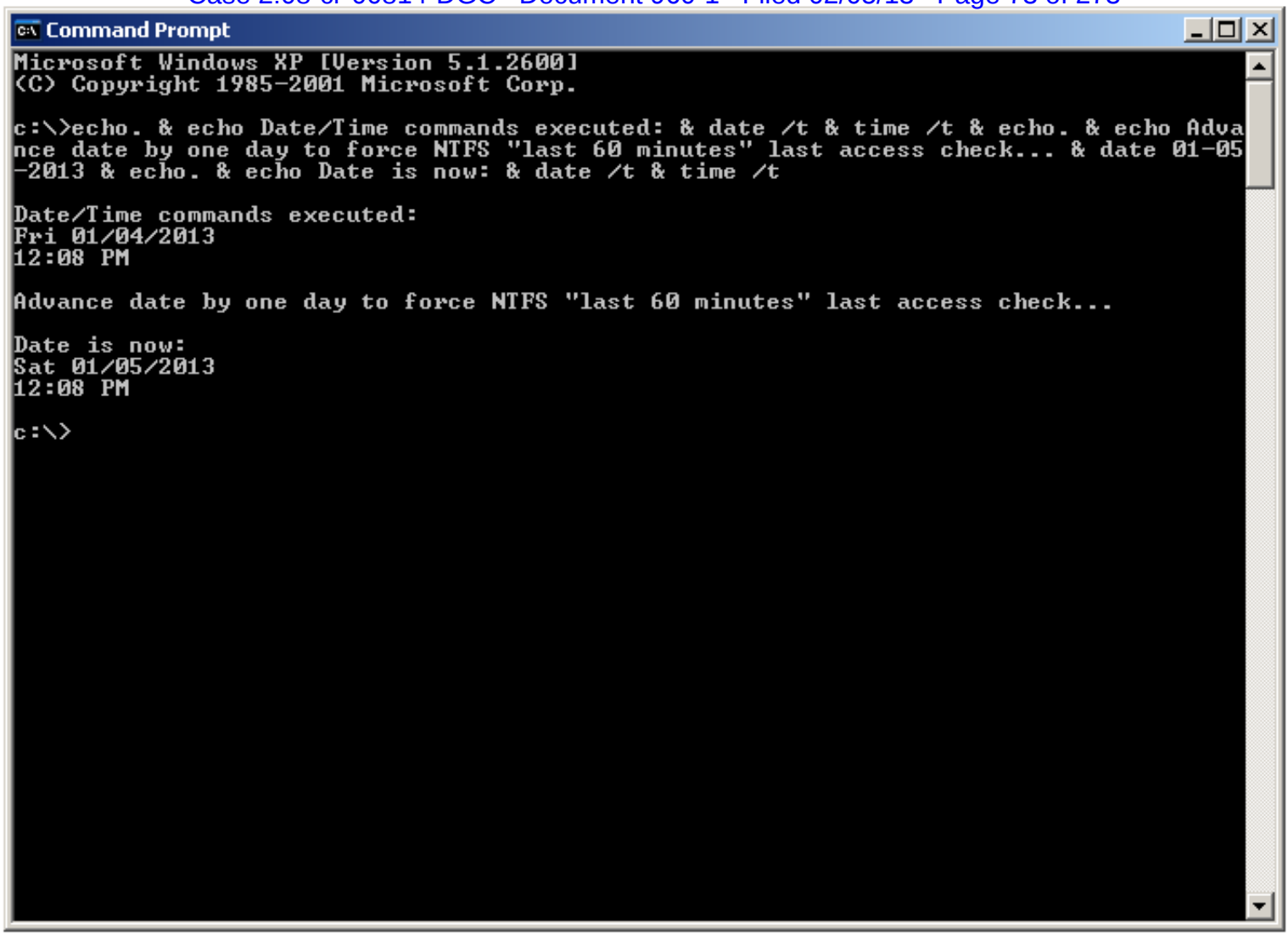

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read file date properties for c:\test28.txt... & echo. & echo File creation date: & dir c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Fri 01/04/2013
12:07 PM
Now read file date properties for c:\test28.txt...
File creation date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last accessed date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 13

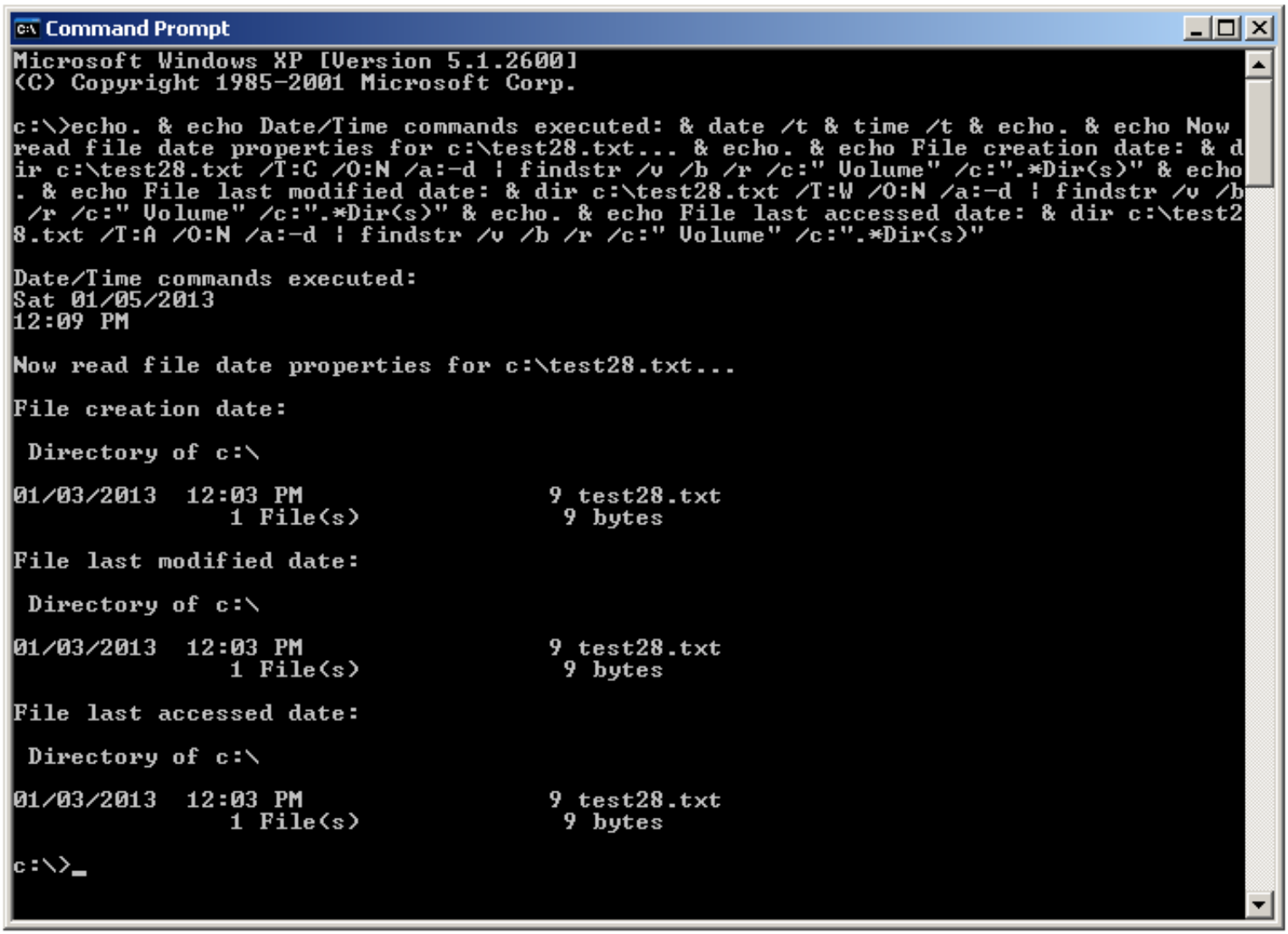

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance date by one day to force NTFS "last 60 minutes" last access check... & date 01-05-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
12:08 PM
Advance date by one day to force NTFS "last 60 minutes" last access check...
Date is now:
Sat 01/05/2013
12:08 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 14

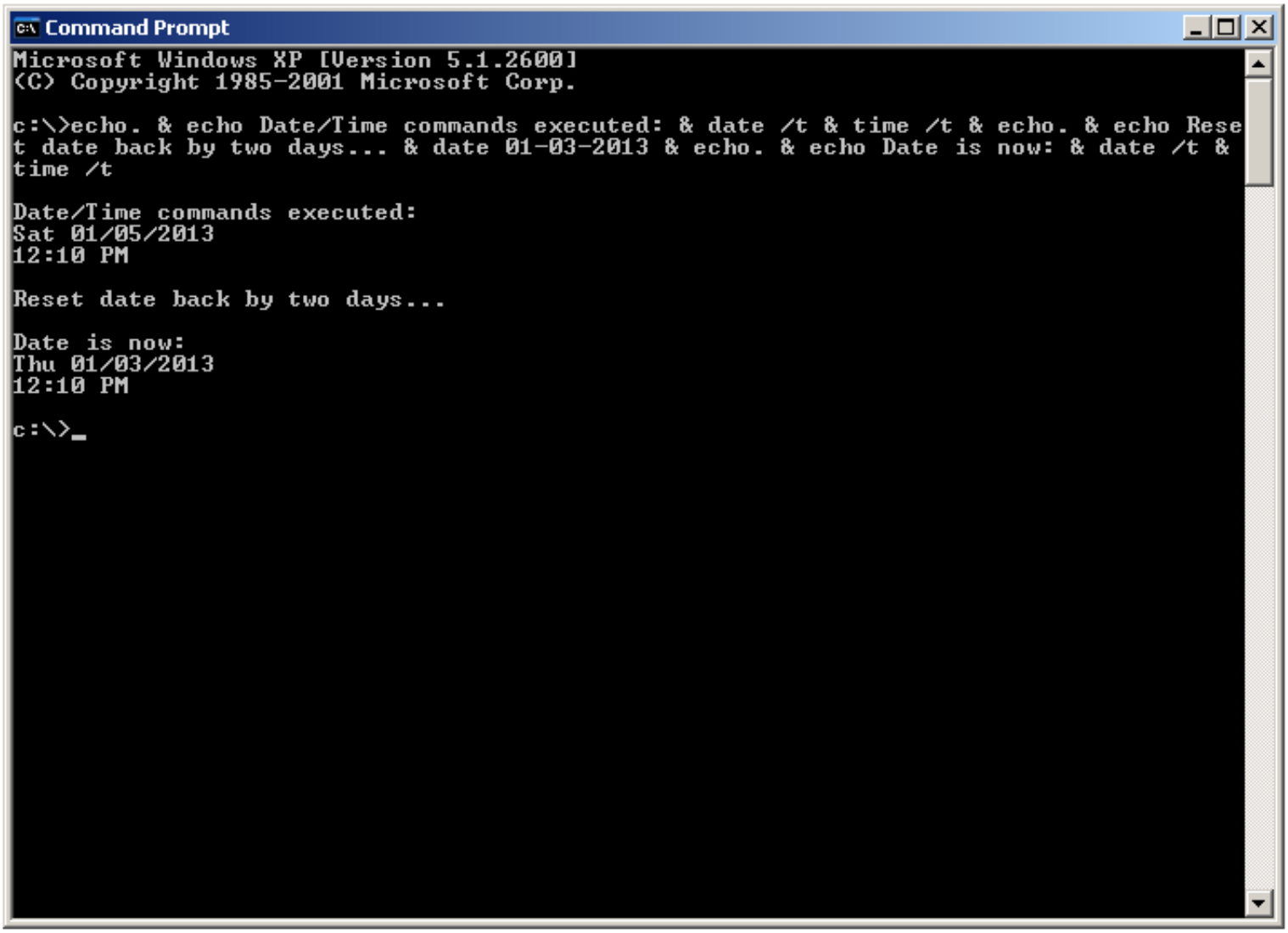

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read file date properties for c:\test28.txt... & echo. & echo File creation date: & dir c:\test28.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test28.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir c:\test28.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Sat 01/05/2013
12:09 PM
Now read file date properties for c:\test28.txt...
File creation date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
File last accessed date:
Directory of c:\
01/03/2013 12:03 PM 9 test28.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 15


Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by two days... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Sat 01/05/2013
12:10 PM
Reset date back by two days...
Date is now:
Thu 01/03/2013
12:10 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 16

NAME

gpg - OpenPGP encryption and signing tool

SYNOPSIS

gpg [--homedir dir] [--options file] [options] command [args]

DESCRIPTION

gpg is the OpenPGP part of the GNU Privacy Guard (GnuPG). It is a tool to provide digital encryption and signing services using the OpenPGP standard. gpg features complete key management and all bells and whistles you can expect from a decent OpenPGP implementation.

This is the standalone version of gpg. For desktop use you should consider using gpg2 ([On some platforms gpg2 is installed under the name gpg]).

RETURN VALUE

The program returns 0 if everything was fine, 1 if at least a signature was bad, and other error codes for fatal errors.

WARNINGS

Use a *good* password for your user account and a *good* passphrase to protect your secret key. This passphrase is the weakest part of the whole system. Programs to do dictionary attacks on your secret keyring are very easy to write and so you should protect your "~/.gnupg/" directory very well.

Keep in mind that, if this program is used over a network (telnet), it is *very* easy to spy out your passphrase!

If you are going to verify detached signatures, make sure that the program knows about it; either give both filenames on the command line or use '-'' to specify STDIN.

INTEROPERABILITY

GnuPG tries to be a very flexible implementation of the OpenPGP standard. In particular, GnuPG implements many of the optional parts of the standard, such as the SHA-512 hash, and the ZLIB and BZIP2 compression algorithms. It is important to be aware that not all OpenPGP programs implement these optional algorithms and that by forcing their use via the --cipher-algo, --digest-algo, --cert-digest-algo, or --compress-algo options in GnuPG, it is possible to create a perfectly valid OpenPGP message, but one that cannot be read by the intended recipient.

There are dozens of variations of OpenPGP programs available, and each supports a slightly different subset of these optional algorithms. For example, until recently, no (unhacked) version of PGP supported the BLOWFISH cipher algorithm. A message using BLOWFISH simply could not be

read by a PGP user. By default, GnuPG uses the standard OpenPGP preferences system that will always do the right thing and create messages that are usable by all recipients, regardless of which OpenPGP program they use. Only override this safe default if you really know what you are doing.

If you absolutely must override the safe default, or if the preferences on a given key are invalid for some reason, you are far better off using the --pgp6, --pgp7, or --pgp8 options. These options are safe as they do not force any particular algorithms in violation of OpenPGP, but rather reduce the available algorithms to a "PGP-safe" list.

COMMANDS

Commands are not distinguished from options except for the fact that only one command is allowed.

gpg may be run with no commands, in which case it will perform a reasonable action depending on the type of file it is given as input (an encrypted message is decrypted, a signature is verified, a file containing keys is listed).

Please remember that option as well as command parsing stops as soon as a non-option is encountered, you can explicitly stop parsing by using the special option --.

Commands not specific to the function

--version
: Print the program version and licensing information. Note that you cannot abbreviate this command.

--help

-h
: Print a usage message summarizing the most useful command line options. Note that you cannot abbreviate this command.

--warranty
: Print warranty information.

--dump-options
: Print a list of all available options and commands. Note that you cannot abbreviate this command.

Commands to select the type of operation

--sign

-s Make a signature. This command may be combined with --encrypt (for a signed and encrypted message), --symmetric (for a signed and symmetrically encrypted message), or --encrypt and --symmetric together (for a signed message that may be decrypted via a secret key or a passphrase). The key to be used for signing is chosen by default or can be set with the --local-user and --default-key options.

--clearsign

Make a clear text signature. The content in a clear text signature is readable without any special software. OpenPGP software is only needed to verify the signature. Clear text signatures may modify end-of-line whitespace for platform independence and are not intended to be reversible. The key to be used for signing is chosen by default or can be set with the --local-user and --default-key options.

--detach-sign

-b Make a detached signature.

--encrypt

-e Encrypt data. This option may be combined with --sign (for a signed and encrypted message), --symmetric (for a message that may be decrypted via a secret key or a passphrase), or --sign and --symmetric together (for a signed message that may be decrypted via a secret key or a passphrase).

--symmetric

-c Encrypt with a symmetric cipher using a passphrase. The default symmetric cipher used is CAST5, but may be chosen with the --cipher-algo option. This option may be combined with --sign (for a signed and symmetrically encrypted message), --encrypt (for a message that may be decrypted via a secret key or a passphrase), or --sign and --encrypt together (for a signed message that may be decrypted via a secret key or a passphrase).

--store

Store only (make a simple RFC1991 literal data packet).

--decrypt

-d Decrypt the file given on the command line (or STDIN if no file is specified) and write it to STDOUT (or the file specified with --output). If the decrypted file is signed, the signature is also verified. This command differs from the default operation, as it never writes to the filename which is included in the file and it rejects files which don't begin with an encrypted message.

--verify

Assume that the first argument is a signed file or a detached signature and verify it without generating any output. With no arguments, the signature packet is read from STDIN. If only a sigfile is given, it may be a complete signature or a detached signature, in which case the signed stuff is expected in a file without the ".sig" or ".asc" extension. With more than 1 argument, the first should be a detached signature and the remaining files are the signed stuff. To read the signed stuff from STDIN, use '-'' as the second filename. For security reasons a detached signature cannot read the signed material from STDIN without denoting it in the above way.

--multifile
This modifies certain other commands to accept multiple files for processing on the command line or read from STDIN with each filename on a separate line. This allows for many files to be processed at once. --multifile may currently be used along with --verify, --encrypt, and --decrypt. Note that --multifile --verify may not be used with detached signatures.

--verify-files
Identical to --multifile --verify.

--encrypt-files
Identical to --multifile --encrypt.

--decrypt-files
Identical to --multifile --decrypt.

--list-keys

-k

--list-public-keys
List all keys from the public keyrings, or just the keys given on the command line. -k is slightly different from --list-keys in that it allows only for one argument and takes the second argument as the keyring to search. This is for command line compatibility with PGP 2 and has been removed in gpg2.

Avoid using the output of this command in scripts or other programs as it is likely to change as GnuPG changes. See --with-colons for a machine-parseable key listing command that is appropriate for use in scripts and other programs.

--list-secret-keys

-K List all keys from the secret keyrings, or just the ones given on the command line. A # after the letters sec means that the secret key is not usable (for example, if it was created via --export-secret-subkeys).

--list-sigs
Same as --list-keys, but the signatures are listed too.

For each signature listed, there are several flags in between

the "sig" tag and keyid. These flags give additional information about each signature. From left to right, they are the numbers 1-3 for certificate check level (see --ask-cert-level), "L" for a local or non-exportable signature (see --lsign-key), "R" for a nonRevocable signature (see the --edit-key command "nrsign"), "P" for a signature that contains a policy URL (see --cert-policy-url), "N" for a signature that contains a notation (see --cert-notation), "X" for an eXpired signature (see --ask-cert-expire), and the numbers 1-9 or "T" for 10 and above to indicate trust signature levels (see the --edit-key command "tsign").

--check-sigs
Same as --list-sigs, but the signatures are verified. Note that for performance reasons the revocation status of a signing key is not shown.

The status of the verification is indicated by a flag directly following the "sig" tag (and thus before the flags described above for --list-sigs). A "!" indicates that the signature has been successfully verified, a "-" denotes a bad signature and a "%" is used if an error occurred while checking the signature (e.g. a non supported algorithm).

--fingerprint
List all keys (or the specified ones) along with their fingerprints. This is the same output as --list-keys but with the additional output of a line with the fingerprint. May also be combined with --list-sigs or --check-sigs. If this command is given twice, the fingerprints of all secondary keys are listed too.

--list-packets
List only the sequence of packets. This is mainly useful for debugging.

--card-edit
Present a menu to work with a smartcard. The subcommand "help" provides an overview on available commands. For a detailed description, please see the Card HOWTO at http://www.gnupg.org/documentation/howtos.html#GnuPG-cardHOWTO .

--card-status
Show the content of the smart card.

--change-pin
Present a menu to allow changing the PIN of a smartcard. This functionality is also available as the subcommand "passwd" with the --card-edit command.

--delete-key name
Remove key from the public keyring. In batch mode either --yes is required or the key must be specified by fingerprint. This is a safeguard against accidental deletion of multiple keys.

--delete-secret-key name
Remove key from the secret and public keyring. In batch mode the key must be specified by fingerprint.

--delete-secret-and-public-key name
Same as --delete-key, but if a secret key exists, it will be removed first. In batch mode the key must be specified by fingerprint.

--export
Either export all keys from all keyrings (default keyrings and those registered via option --keyring), or if at least one name is given, those of the given name. The new keyring is written to STDOUT or to the file given with option --output. Use together with --armor to mail those keys.

--send-keys key IDs
Similar to --export but sends the keys to a keyserver. Fingerprints may be used instead of key IDs. Option --keyserver must be used to give the name of this keyserver. Don't send your complete keyring to a keyserver --- select only those keys which are new or changed by you. If no key IDs are given, gpg does nothing.

--export-secret-keys

--export-secret-subkeys
Same as --export, but exports the secret keys instead. This is normally not very useful and a security risk. The second form of the command has the special property to render the secret part of the primary key useless; this is a GNU extension to OpenPGP and other implementations can not be expected to successfully import such a key. See the option --simple-sk-checksum if you want to import such an exported key with an older OpenPGP implementation.

--import

--fast-import
Import/merge keys. This adds the given keys to the keyring. The fast version is currently just a synonym.

There are a few other options which control how this command works. Most notable here is the --import-options merge-only option which does not insert new keys but does only the merging of new signatures, user-IDs and subkeys.

--recv-keys key IDs
Import the keys with the given key IDs from a keyserver. Option --keyserver must be used to give the name of this keyserver.

--refresh-keys
Request updates from a keyserver for keys that already exist on the local keyring. This is useful for updating a key with the

latest signatures, user IDs, etc. Calling this with no arguments will refresh the entire keyring. Option --keyserver must be used to give the name of the keyserver for all keys that do not have preferred keyservers set (see --keyserver-options honor-keyserver-url).

--search-keys names
Search the keyserver for the given names. Multiple names given here will be joined together to create the search string for the keyserver. Option --keyserver must be used to give the name of this keyserver. Keyservers that support different search methods allow using the syntax specified in "How to specify a user ID" below. Note that different keyserver types support different search methods. Currently only LDAP supports them all.

--fetch-keys URIs
Retrieve keys located at the specified URIs. Note that different installations of GnuPG may support different protocols (HTTP, FTP, LDAP, etc.)

--update-trustdb
Do trust database maintenance. This command iterates over all keys and builds the Web of Trust. This is an interactive command because it may have to ask for the "ownertrust" values for keys. The user has to give an estimation of how far she trusts the owner of the displayed key to correctly certify (sign) other keys. GnuPG only asks for the ownertrust value if it has not yet been assigned to a key. Using the --edit-key menu, the assigned value can be changed at any time.

--check-trustdb
Do trust database maintenance without user interaction. From time to time the trust database must be updated so that expired keys or signatures and the resulting changes in the Web of Trust can be tracked. Normally, GnuPG will calculate when this is required and do it automatically unless --no-auto-check-trustdb is set. This command can be used to force a trust database check at any time. The processing is identical to that of --update-trustdb but it skips keys with a not yet defined "ownertrust".

For use with cron jobs, this command can be used together with --batch in which case the trust database check is done only if a check is needed. To force a run even in batch mode add the option --yes.

--export-ownertrust
Send the ownertrust values to STDOUT. This is useful for backup purposes as these values are the only ones which can't be recreated from a corrupted trustdb. Example:

```
gpg --export-ownertrust > otrust.txt
```

--import-ownertrust
Update the trustdb with the ownertrust values stored in files (or STDIN if not given); existing values will be overwritten. In case of a severely damaged trustdb and if you have a recent

backup of the ownertrust values (e.g. in the file `otrust.txt', you may re-create the trustdb using these commands:

```
cd ~/.gnupg
rm trustdb.gpg
gpg --import-ownertrust < otrust.txt
```

--rebuild-keydb-caches
When updating from version 1.0.6 to 1.0.7 this command should be used to create signature caches in the keyring. It might be handy in other situations too.

--print-md algo

--print-mds
Print message digest of algorithm ALGO for all given files or STDIN. With the second form (or a deprecated "*" as algo) digests for all available algorithms are printed.

--gen-random 0|1|2 count
Emit count random bytes of the given quality level 0, 1 or 2. If count is not given or zero, an endless sequence of random bytes will be emitted. If used with --armor the output will be base64 encoded. PLEASE, don't use this command unless you know what you are doing; it may remove precious entropy from the system!

--gen-prime mode bits
Use the source, Luke :-). The output format is still subject to change.

--enarmor

--dearmor
Pack or unpack an arbitrary input into/from an OpenPGP ASCII armor. This is a GnuPG extension to OpenPGP and in general not very useful.

How to manage your keys

This section explains the main commands for key management

--gen-key
Generate a new key pair. This command is normally only used interactively.

There is an experimental feature which allows you to create keys in batch mode. See the file `doc/DETAILS' in the source distribution on how to use this.

--gen-revoke name
Generate a revocation certificate for the complete key. To revoke a subkey or a signature, use the --edit command.

--desig-revoke name
Generate a designated revocation certificate for a key. This allows a user (with the permission of the keyholder) to revoke someone else's key.

--edit-key
Present a menu which enables you to do most of the key management related tasks. It expects the specification of a key on the command line.

uid n Toggle selection of user ID or photographic user ID with index n. Use * to select all and 0 to deselect all.

key n Toggle selection of subkey with index n. Use * to select all and 0 to deselect all.

sign Make a signature on key of user name If the key is not yet signed by the default user (or the users given with -u), the program displays the information of the key again, together with its fingerprint and asks whether it should be signed. This question is repeated for all users specified with -u.

lsign Same as "sign" but the signature is marked as non-exportable and will therefore never be used by others. This may be used to make keys valid only in the local environment.

nrsign Same as "sign" but the signature is marked as non-revocable and can therefore never be revoked.

tsign Make a trust signature. This is a signature that combines the notions of certification (like a regular signature), and trust (like the "trust" command). It is generally only useful in distinct communities or groups.

Note that "l" (for local / non-exportable), "nr" (for non-revocable, and "t" (for trust) may be freely mixed and prefixed to "sign" to create a signature of any type desired.

delsig Delete a signature. Note that it is not possible to retract a signature, once it has been send to the public (i.e. to a keyserver). In that case you better use revsig.

revsig Revoke a signature. For every signature which has been

generated by one of the secret keys, GnuPG asks whether a revocation certificate should be generated.

check Check the signatures on all selected user IDs.

adduid Create an additional user ID.

addphoto
Create a photographic user ID. This will prompt for a JPEG file that will be embedded into the user ID. Note that a very large JPEG will make for a very large key. Also note that some programs will display your JPEG unchanged (GnuPG), and some programs will scale it to fit in a dialog box (PGP).

showphoto
Display the selected photographic user ID.

deluid Delete a user ID or photographic user ID. Note that it is not possible to retract a user id, once it has been send to the public (i.e. to a keyserver). In that case you better use revuid.

revuid Revoke a user ID or photographic user ID.

primary
Flag the current user id as the primary one, removes the primary user id flag from all other user ids and sets the timestamp of all affected self-signatures one second ahead. Note that setting a photo user ID as primary makes it primary over other photo user IDs, and setting a regular user ID as primary makes it primary over other regular user IDs.

keyserver
Set a preferred keyserver for the specified user ID(s). This allows other users to know where you prefer they get your key from. See --keyserver-options honor-keyserver-url for more on how this works. Setting a value of "none" removes an existing preferred keyserver.

notation
Set a name=value notation for the specified user ID(s). See --cert-notation for more on how this works. Setting a value of "none" removes all notations, setting a notation prefixed with a minus sign (-) removes that notation, and setting a notation name (without the =value) prefixed with a minus sign removes all notations with that name.

pref List preferences from the selected user ID. This shows the actual preferences, without including any implied preferences.

showpref
More verbose preferences listing for the selected user ID. This shows the preferences in effect by including the implied preferences of 3DES (cipher), SHA-1 (digest), and Uncompressed (compression) if they are not already included in the preference list. In addition, the preferred keyserver and signature notations (if any) are shown.

setpref string
Set the list of user ID preferences to string for all (or just the selected) user IDs. Calling setpref with no arguments sets the preference list to the default (either built-in or set via --default-preference-list), and calling setpref with "none" as the argument sets an empty preference list. Use gpg --version to get a list of available algorithms. Note that while you can change the preferences on an attribute user ID (aka "photo ID"), GnuPG does not select keys via attribute user IDs so these preferences will not be used by GnuPG.

When setting preferences, you should list the algorithms in the order which you'd like to see them used by someone else when encrypting a message to your key. If you don't include 3DES, it will be automatically added at the end. Note that there are many factors that go into choosing an algorithm (for example, your key may not be the only recipient), and so the remote OpenPGP application being used to send to you may or may not follow your exact chosen order for a given message. It will, however, only choose an algorithm that is present on the preference list of every recipient key. See also the INTEROPERABILITY WITH OTHER OPENPGP PROGRAMS section below.

addkey Add a subkey to this key.

addcardkey
Generate a subkey on a card and add it to this key.

keytocard
Transfer the selected secret subkey (or the primary key if no subkey has been selected) to a smartcard. The secret key in the keyring will be replaced by a stub if the key could be stored successfully on the card and you use the save command later. Only certain key types may be transferred to the card. A sub menu allows you to select on what card to store the key. Note that it is not possible to get that key back from the card - if the card gets broken your secret key will be lost unless you have a backup somewhere.

bkuptocard file
Restore the given file to a card. This command may be used to restore a backup key (as generated during card initialization) to a new card. In almost all cases this will be the encryption key. You should use this command only with the corresponding public key and make sure that

the file given as argument is indeed the backup to restore. You should then select 2 to restore as encryption key. You will first be asked to enter the passphrase of the backup key and then for the Admin PIN of the card.

delkey Remove a subkey (secondart key). Note that it is not possible to retract a subkey, once it has been send to the public (i.e. to a keyserver). In that case you better use revkey.

revkey Revoke a subkey.

expire Change the key or subkey expiration time. If a subkey is selected, the expiration time of this subkey will be changed. With no selection, the key expiration of the primary key is changed.

trust Change the owner trust value for the key. This updates the trust-db immediately and no save is required.

disable

enable Disable or enable an entire key. A disabled key can not normally be used for encryption.

addrevoker
Add a designated revoker to the key. This takes one optional argument: "sensitive". If a designated revoker is marked as sensitive, it will not be exported by default (see export-options).

passwd Change the passphrase of the secret key.

toggle Toggle between public and secret key listing.

clean Compact (by removing all signatures except the selfsig) any user ID that is no longer usable (e.g. revoked, or expired). Then, remove any signatures that are not usable by the trust calculations. Specifically, this removes any signature that does not validate, any signature that is superseded by a later signature, revoked signatures, and signatures issued by keys that are not present on the keyring.

minimize
Make the key as small as possible. This removes all signatures from each user ID except for the most recent self-signature.

cross-certify
Add cross-certification signatures to signing subkeys

that may not currently have them. Cross-certification signatures protect against a subtle attack against signing subkeys. See --require-cross-certification. All new keys generated have this signature by default, so this option is only useful to bring older keys up to date.

save Save all changes to the key rings and quit.

quit Quit the program without updating the key rings.

The listing shows you the key with its secondary keys and all user ids. The primary user id is indicated by a dot, and selected keys or user ids are indicated by an asterisk. The trust value is displayed with the primary key: the first is the assigned owner trust and the second is the calculated trust value. Letters are used for the values:

- No ownertrust assigned / not yet calculated.

e Trust calculation has failed; probably due to an expired key.

q Not enough information for calculation.

n Never trust this key.

m Marginally trusted.

f Fully trusted.

u Ultimately trusted.

--sign-key name
Signs a public key with your secret key. This is a shortcut version of the subcommand "sign" from --edit.

--lsign-key name
Signs a public key with your secret key but marks it as non-exportable. This is a shortcut version of the subcommand "lsign" from --edit-key.

OPTIONS
gpg features a bunch of options to control the exact behaviour and to change the default configuration.

Long options can be put in an options file (default "~/.gnupg/gpg.conf"). Short option names will not work - for example, "armor" is a valid option for the options file, while "a" is not. Do not write the 2 dashes, but simply the name of the option and any required arguments. Lines with a hash ('#') as the first non-white-space character are ignored. Commands may be put in this file too, but that is not generally useful as the command will execute automatically with every execution of gpg.

Please remember that option parsing stops as soon as a non-option is encountered, you can explicitly stop parsing by using the special option --.

How to change the configuration

These options are used to change the configuration and are usually found in the option file.

--default-key name
: Use name as the default key to sign with. If this option is not used, the default key is the first key found in the secret keyring. Note that -u or --local-user overrides this option.

--default-recipient name
: Use name as default recipient if option --recipient is not used and don't ask if this is a valid one. name must be non-empty.

--default-recipient-self
: Use the default key as default recipient if option --recipient is not used and don't ask if this is a valid one. The default key is the first one from the secret keyring or the one set with --default-key.

--no-default-recipient
: Reset --default-recipient and --default-recipient-self.

-v, --verbose
: Give more information during processing. If used twice, the input data is listed in detail.

--no-verbose
: Reset verbose level to 0.

-q, --quiet
: Try to be as quiet as possible.

--batch

--no-batch
: Use batch mode. Never ask, do not allow interactive commands.

--no-batch disables this option. Note that even with a filename given on the command line, gpg might still need to read from STDIN (in particular if gpg figures that the input is a detached signature and no data file has been specified). Thus if you do not want to feed data via STDIN, you should connect STDIN to `/dev/null'.

--no-tty
Make sure that the TTY (terminal) is never used for any output. This option is needed in some cases because GnuPG sometimes prints warnings to the TTY even if --batch is used.

--yes Assume "yes" on most questions.

--no Assume "no" on most questions.

--list-options parameters
This is a space or comma delimited string that gives options used when listing keys and signatures (that is, --list-keys, --list-sigs, --list-public-keys, --list-secret-keys, and the --edit-key functions). Options can be prepended with a no- (after the two dashes) to give the opposite meaning. The options are:

show-photos
Causes --list-keys, --list-sigs, --list-public-keys, and --list-secret-keys to display any photo IDs attached to the key. Defaults to no. See also --photo-viewer.

show-policy-urls
Show policy URLs in the --list-sigs or --check-sigs listings. Defaults to no.

show-notations

show-std-notations

show-user-notations
Show all, IETF standard, or user-defined signature notations in the --list-sigs or --check-sigs listings. Defaults to no.

show-keyserver-urls

Show any preferred keyserver URL in the --list-sigs or --check-sigs listings. Defaults to no.

show-uid-validity
Display the calculated validity of user IDs during key listings. Defaults to no.

show-unusable-uids
Show revoked and expired user IDs in key listings. Defaults to no.

show-unusable-subkeys
Show revoked and expired subkeys in key listings. Defaults to no.

show-keyring
Display the keyring name at the head of key listings to show which keyring a given key resides on. Defaults to no.

show-sig-expire
Show signature expiration dates (if any) during --list-sigs or --check-sigs listings. Defaults to no.

show-sig-subpackets
Include signature subpackets in the key listing. This option can take an optional argument list of the subpackets to list. If no argument is passed, list all subpackets. Defaults to no. This option is only meaningful when using --with-colons along with --list-sigs or --check-sigs.

--verify-options parameters
This is a space or comma delimited string that gives options used when verifying signatures. Options can be prepended with a `no-' to give the opposite meaning. The options are:

show-photos
Display any photo IDs present on the key that issued the signature. Defaults to no. See also --photo-viewer.

show-policy-urls
Show policy URLs in the signature being verified. Defaults to no.

show-notations

show-std-notations

show-user-notations
Show all, IETF standard, or user-defined signature notations in the signature being verified. Defaults to IETF standard.

show-keyserver-urls
Show any preferred keyserver URL in the signature being verified. Defaults to no.

show-uid-validity

Display the calculated validity of the user IDs on the key that issued the signature. Defaults to no.

show-unusable-uids
Show revoked and expired user IDs during signature verification. Defaults to no.

show-primary-uid-only
Show only the primary user ID during signature verification. That is all the AKA lines as well as photo Ids are not shown with the signature verification status.

pka-lookups
Enable PKA lookups to verify sender addresses. Note that PKA is based on DNS, and so enabling this option may disclose information on when and what signatures are verified or to whom data is encrypted. This is similar to the "web bug" described for the auto-key-retrieve feature.

pka-trust-increase
Raise the trust in a signature to full if the signature passes PKA validation. This option is only meaningful if pka-lookups is set.

--enable-dsa2

--disable-dsa2
Enable hash truncation for all DSA keys even for old DSA Keys up to 1024 bit. This is also the default with --openpgp. Note that older versions of GnuPG also required this flag to allow the generation of DSA larger than 1024 bit.

--photo-viewer string
This is the command line that should be run to view a photo ID. "%i" will be expanded to a filename containing the photo. "%I" does the same, except the file will not be deleted once the viewer exits. Other flags are "%k" for the key ID, "%K" for the long key ID, "%f" for the key fingerprint, "%t" for the extension of the image type (e.g. "jpg"), "%T" for the MIME type of the image (e.g. "image/jpeg"), "%v" for the single-character calculated validity of the image being viewed (e.g. "f"), "%V" for the calculated validity as a string (e.g. "full"), and "%%" for an actual percent sign. If neither %i or %I are present, then the photo will be supplied to the viewer on standard input.

The default viewer is "xloadimage -fork -quiet -title 'KeyID 0x%k' STDIN". Note that if your image viewer program is not secure, then executing it from GnuPG does not make it secure.

--exec-path string
Sets a list of directories to search for photo viewers and keyserver helpers. If not provided, keyserver helpers use the compiled-in default directory, and photo viewers use the $PATH environment variable. Note, that on W32 system this value is ignored when searching for keyserver helpers.

--keyring file
Add file to the current list of keyrings. If file begins with a tilde and a slash, these are replaced by the $HOME directory. If the filename does not contain a slash, it is assumed to be in the GnuPG home directory ("~/.gnupg" if --homedir or $GNUPGHOME is not used).

Note that this adds a keyring to the current list. If the intent is to use the specified keyring alone, use --keyring along with --no-default-keyring.

--secret-keyring file
Same as --keyring but for the secret keyrings.

--primary-keyring file
Designate file as the primary public keyring. This means that newly imported keys (via --import or keyserver --recv-from) will go to this keyring.

--trustdb-name file
Use file instead of the default trustdb. If file begins with a tilde and a slash, these are replaced by the $HOME directory. If the filename does not contain a slash, it is assumed to be in the GnuPG home directory (`~/.gnupg' if --homedir or $GNUPGHOME is not used).

--homedir dir
Set the name of the home directory to dir. If this option is not used, the home directory defaults to `~/.gnupg'. It is only recognized when given on the command line. It also overrides any home directory stated through the environment variable `GNUPGHOME' or (on W32 systems) by means of the Registry entry HKCU\Software\GNU\GnuPG:HomeDir.

--pcsc-driver file
Use file to access the smartcard reader. The current default is `libpcsclite.so.1' for GLIBC based systems, `/System/Library/Frameworks/PCSC.framework/PCSC' for MAC OS X, `winscard.dll' for Windows and `libpcsclite.so' for other systems.

--disable-ccid
Disable the integrated support for CCID compliant readers. This allows to fall back to one of the other drivers even if the internal CCID driver can handle the reader. Note, that CCID support is only available if libusb was available at build time.

--reader-port number_or_string
This option may be used to specify the port of the card terminal. A value of 0 refers to the first serial device; add 32768 to access USB devices. The default is 32768 (first USB device). PC/SC or CCID readers might need a string here; run the program in verbose mode to get a list of available readers. The default

is then the first reader found.

--display-charset name
Set the name of the native character set. This is used to convert some informational strings like user IDs to the proper UTF-8 encoding. Note that this has nothing to do with the character set of data to be encrypted or signed; GnuPG does not recode user-supplied data. If this option is not used, the default character set is determined from the current locale. A verbosity level of 3 shows the chosen set. Valid values for name are:

iso-8859-1
This is the Latin 1 set.

iso-8859-2
The Latin 2 set.

iso-8859-15
This is currently an alias for the Latin 1 set.

koi8-r The usual Russian set (rfc1489).

utf-8 Bypass all translations and assume that the OS uses native UTF-8 encoding.

--utf8-strings

--no-utf8-strings
Assume that command line arguments are given as UTF8 strings. The default (--no-utf8-strings) is to assume that arguments are encoded in the character set as specified by --display-charset. These options affect all following arguments. Both options may be used multiple times.

--options file
Read options from file and do not try to read them from the default options file in the homedir (see --homedir). This option is ignored if used in an options file.

--no-options
Shortcut for --options /dev/null. This option is detected before an attempt to open an option file. Using this option will also prevent the creation of a `~/.gnupg' homedir.

-z n

--compress-level n

--bzip2-compress-level n
Set compression level to n for the ZIP and ZLIB compression algorithms. The default is to use the default compression level of zlib (normally 6). --bzip2-compress-level sets the compression level for the BZIP2 compression algorithm (defaulting to 6 as well). This is a different option from --compress-level since BZIP2 uses a significant amount of memory for each additional compression level. -z sets both. A value of 0 for n disables compression.

--bzip2-decompress-lowmem
Use a different decompression method for BZIP2 compressed files. This alternate method uses a bit more than half the memory, but also runs at half the speed. This is useful under extreme low memory circumstances when the file was originally compressed at a high --bzip2-compress-level.

--mangle-dos-filenames

--no-mangle-dos-filenames
Older version of Windows cannot handle filenames with more than one dot. --mangle-dos-filenames causes GnuPG to replace (rather than add to) the extension of an output filename to avoid this problem. This option is off by default and has no effect on non-Windows platforms.

--ask-cert-level

--no-ask-cert-level
When making a key signature, prompt for a certification level. If this option is not specified, the certification level used is set via --default-cert-level. See --default-cert-level for information on the specific levels and how they are used. --no-ask-cert-level disables this option. This option defaults to no.

--default-cert-level n
The default to use for the check level when signing a key.

0 means you make no particular claim as to how carefully you verified the key.

1 means you believe the key is owned by the person who claims to own it but you could not, or did not verify the key at all. This is useful for a "persona" verification, where you sign the key of a pseudonymous user.

2 means you did casual verification of the key. For example, this could mean that you verified that the key fingerprint and checked the user ID on the key against a photo ID.

3 means you did extensive verification of the key. For example, this could mean that you verified the key fingerprint with the owner of the key in person, and that you checked, by means of a hard to forge document with a photo ID (such as a passport) that the name of the key owner matches the name in the user ID on the key, and finally that you verified (by exchange of email) that the email address on the key belongs to the key owner.

Note that the examples given above for levels 2 and 3 are just that: examples. In the end, it is up to you to decide just what "casual" and "extensive" mean to you.

This option defaults to 0 (no particular claim).

--min-cert-level
When building the trust database, treat any signatures with a certification level below this as invalid. Defaults to 2, which disregards level 1 signatures. Note that level 0 "no particular claim" signatures are always accepted.

--trusted-key long key ID
Assume that the specified key (which must be given as a full 8 byte key ID) is as trustworthy as one of your own secret keys. This option is useful if you don't want to keep your secret keys (or one of them) online but still want to be able to check the validity of a given recipient's or signator's key.

--trust-model pgp|classic|direct|always|auto
Set what trust model GnuPG should follow. The models are:

pgp This is the Web of Trust combined with trust signatures as used in PGP 5.x and later. This is the default trust model when creating a new trust database.

classic
This is the standard Web of Trust as used in PGP 2.x and earlier.

direct Key validity is set directly by the user and not calculated via the Web of Trust.

always Skip key validation and assume that used keys are always fully trusted. You generally won't use this unless you are using some external validation scheme. This option also suppresses the "[uncertain]" tag printed with signature checks when there is no evidence that the user ID is bound to the key.

auto Select the trust model depending on whatever the internal trust database says. This is the default model if such a database already exists.

--auto-key-locate parameters

--no-auto-key-locate
GnuPG can automatically locate and retrieve keys as needed using this option. This happens when encrypting to an email address (in the "user@example.com" form), and there are no user@example.com keys on the local keyring. This option takes any number of the following mechanisms, in the order they are to be tried:

cert Locate a key using DNS CERT, as specified in rfc4398.

pka Locate a key using DNS PKA.

ldap Using DNS Service Discovery, check the domain in question for any LDAP keyservers to use. If this fails, attempt to locate the key using the PGP Universal method of checking 'ldap://keys.(thedomain)''.

keyserver
Locate a key using whatever keyserver is defined using the --keyserver option.

keyserver-URL
In addition, a keyserver URL as used in the --keyserver option may be used here to query that particular keyserver.

local Locate the key using the local keyrings. This mechanism allows to select the order a local key lookup is done. Thus using '--auto-key-locate local'' is identical to --no-auto-key-locate.

nodefault
This flag disables the standard local key lookup, done before any of the mechanisms defined by the --auto-key-locate are tried. The position of this mechanism in the list does not matter. It is not required if local is also used.

--keyid-format short|0xshort|long|0xlong
Select how to display key IDs. "short" is the traditional 8-character key ID. "long" is the more accurate (but less convenient) 16-character key ID. Add an "0x" to either to include an "0x" at the beginning of the key ID, as in 0x99242560.

--keyserver name
Use name as your keyserver. This is the server that --recv-keys, --send-keys, and --search-keys will communicate with to receive keys from, send keys to, and search for keys on. The format of the name is a URI: `scheme:[//]keyservername[:port]' The scheme is the type of keyserver: "hkp" for the HTTP (or compatible) keyservers, "ldap" for the LDAP keyservers, or "mailto" for the Graff email keyserver. Note that your particular installation of GnuPG may have other keyserver types available as well. Keyserver schemes are case-insensitive. After the keyserver name, optional keyserver configuration options may be provided. These are the same as the global --keyserver-options from below, but apply only to this particular keyserver.

Most keyservers synchronize with each other, so there is generally no need to send keys to more than one server. The keyserver

hkp://keys.gnupg.net uses round robin DNS to give a different keyserver each time you use it.

--keyserver-options name=value1

This is a space or comma delimited string that gives options for the keyserver. Options can be prefixed with a `no-' to give the opposite meaning. Valid import-options or export-options may be used here as well to apply to importing (--recv-key) or exporting (--send-key) a key from a keyserver. While not all options are available for all keyserver types, some common options are:

include-revoked

When searching for a key with --search-keys, include keys that are marked on the keyserver as revoked. Note that not all keyservers differentiate between revoked and unrevoked keys, and for such keyservers this option is meaningless. Note also that most keyservers do not have cryptographic verification of key revocations, and so turning this option off may result in skipping keys that are incorrectly marked as revoked.

include-disabled

When searching for a key with --search-keys, include keys that are marked on the keyserver as disabled. Note that this option is not used with HKP keyservers.

auto-key-retrieve

This option enables the automatic retrieving of keys from a keyserver when verifying signatures made by keys that are not on the local keyring.

Note that this option makes a "web bug" like behavior possible. Keyserver operators can see which keys you request, so by sending you a message signed by a brand new key (which you naturally will not have on your local keyring), the operator can tell both your IP address and the time when you verified the signature.

honor-keyserver-url

When using --refresh-keys, if the key in question has a preferred keyserver URL, then use that preferred keyserver to refresh the key from. In addition, if auto-key-retrieve is set, and the signature being verified has a preferred keyserver URL, then use that preferred keyserver to fetch the key from. Defaults to yes.

honor-pka-record

If auto-key-retrieve is set, and the signature being verified has a PKA record, then use the PKA information to fetch the key. Defaults to yes.

include-subkeys

When receiving a key, include subkeys as potential targets. Note that this option is not used with HKP keyservers, as they do not support retrieving keys by subkey

id.

use-temp-files
On most Unix-like platforms, GnuPG communicates with the keyserver helper program via pipes, which is the most efficient method. This option forces GnuPG to use temporary files to communicate. On some platforms (such as Win32 and RISC OS), this option is always enabled.

keep-temp-files
If using `use-temp-files', do not delete the temp files after using them. This option is useful to learn the keyserver communication protocol by reading the temporary files.

verbose
Tell the keyserver helper program to be more verbose. This option can be repeated multiple times to increase the verbosity level.

timeout
Tell the keyserver helper program how long (in seconds) to try and perform a keyserver action before giving up. Note that performing multiple actions at the same time uses this timeout value per action. For example, when retrieving multiple keys via --recv-keys, the timeout applies separately to each key retrieval, and not to the --recv-keys command as a whole. Defaults to 30 seconds.

http-proxy=value
Set the proxy to use for HTTP and HKP keyservers. This overrides the "http_proxy" environment variable, if any.

max-cert-size
When retrieving a key via DNS CERT, only accept keys up to this size. Defaults to 16384 bytes.

debug Turn on debug output in the keyserver helper program. Note that the details of debug output depends on which keyserver helper program is being used, and in turn, on any libraries that the keyserver helper program uses internally (libcurl, openldap, etc).

check-cert
Enable certificate checking if the keyserver presents one (for hkps or ldaps). Defaults to on.

ca-cert-file
Provide a certificate store to override the system default. Only necessary if check-cert is enabled, and the keyserver is using a certificate that is not present in a system default certificate list.

Note that depending on the SSL library that the keyserver

helper is built with, this may actually be a directory or a file.

--completes-needed n
Number of completely trusted users to introduce a new key signer (defaults to 1).

--marginals-needed n
Number of marginally trusted users to introduce a new key signer (defaults to 3)

--max-cert-depth n
Maximum depth of a certification chain (default is 5).

--simple-sk-checksum
Secret keys are integrity protected by using a SHA-1 checksum. This method is part of the upcoming enhanced OpenPGP specification but GnuPG already uses it as a countermeasure against certain attacks. Old applications don't understand this new format, so this option may be used to switch back to the old behaviour. Using this option bears a security risk. Note that using this option only takes effect when the secret key is encrypted - the simplest way to make this happen is to change the passphrase on the key (even changing it to the same value is acceptable).

--no-sig-cache
Do not cache the verification status of key signatures. Caching gives a much better performance in key listings. However, if you suspect that your public keyring is not save against write modifications, you can use this option to disable the caching. It probably does not make sense to disable it because all kind of damage can be done if someone else has write access to your public keyring.

--no-sig-create-check
GnuPG normally verifies each signature right after creation to protect against bugs and hardware malfunctions which could leak out bits from the secret key. This extra verification needs some time (about 115% for DSA keys), and so this option can be used to disable it. However, due to the fact that the signature creation needs manual interaction, this performance penalty does not matter in most settings.

--auto-check-trustdb

--no-auto-check-trustdb
If GnuPG feels that its information about the Web of Trust has to be updated, it automatically runs the --check-trustdb command internally. This may be a time consuming process. --no-auto-check-trustdb disables this option.

--use-agent

--no-use-agent

Try to use the GnuPG-Agent. With this option, GnuPG first tries to connect to the agent before it asks for a passphrase. --no-use-agent disables this option.

--gpg-agent-info
Override the value of the environment variable 'GPG_AGENT_INFO''. This is only used when --use-agent has been given. Given that this option is not anymore used by gpg2, it should be avoided if possible.

--lock-once
Lock the databases the first time a lock is requested and do not release the lock until the process terminates.

--lock-multiple
Release the locks every time a lock is no longer needed. Use this to override a previous --lock-once from a config file.

--lock-never
Disable locking entirely. This option should be used only in very special environments, where it can be assured that only one process is accessing those files. A bootable floppy with a stand-alone encryption system will probably use this. Improper usage of this option may lead to data and key corruption.

--exit-on-status-write-error
This option will cause write errors on the status FD to immediately terminate the process. That should in fact be the default but it never worked this way and thus we need an option to enable this, so that the change won't break applications which close their end of a status fd connected pipe too early. Using this option along with --enable-progress-filter may be used to cleanly cancel long running gpg operations.

--limit-card-insert-tries n
With n greater than 0 the number of prompts asking to insert a smartcard gets limited to N-1. Thus with a value of 1 gpg won't at all ask to insert a card if none has been inserted at startup. This option is useful in the configuration file in case an application does not know about the smartcard support and waits ad infinitum for an inserted card.

--no-random-seed-file
GnuPG uses a file to store its internal random pool over invocations. This makes random generation faster; however sometimes write operations are not desired. This option can be used to achieve that with the cost of slower random generation.

--no-greeting
Suppress the initial copyright message.

--no-secmem-warning
Suppress the warning about "using insecure memory".

--no-permission-warning
Suppress the warning about unsafe file and home directory (--homedir) permissions. Note that the permission checks that GnuPG performs are not intended to be authoritative, but rather they simply warn about certain common permission problems. Do not assume that the lack of a warning means that your system is secure.

Note that the warning for unsafe --homedir permissions cannot be suppressed in the gpg.conf file, as this would allow an attacker to place an unsafe gpg.conf file in place, and use this file to suppress warnings about itself. The --homedir permissions warning may only be suppressed on the command line.

--no-mdc-warning
Suppress the warning about missing MDC integrity protection.

--require-secmem

--no-require-secmem
Refuse to run if GnuPG cannot get secure memory. Defaults to no (i.e. run, but give a warning).

--require-cross-certification

--no-require-cross-certification
When verifying a signature made from a subkey, ensure that the cross certification "back signature" on the subkey is present and valid. This protects against a subtle attack against subkeys that can sign. Defaults to --require-cross-certification for gpg.

--expert

--no-expert
Allow the user to do certain nonsensical or "silly" things like signing an expired or revoked key, or certain potentially incompatible things like generating unusual key types. This also disables certain warning messages about potentially incompatible actions. As the name implies, this option is for experts only. If you don't fully understand the implications of what it allows you to do, leave this off. --no-expert disables this option.

Key related options

--recipient name

-r Encrypt for user id name. If this option or --hidden-recipient is not specified, GnuPG asks for the user-id unless --default-recipient is given.

--hidden-recipient name

-R Encrypt for user ID name, but hide the key ID of this user's key. This option helps to hide the receiver of the message and is a limited countermeasure against traffic analysis. If this option or --recipient is not specified, GnuPG asks for the user ID unless --default-recipient is given.

--encrypt-to name
Same as --recipient but this one is intended for use in the options file and may be used with your own user-id as an "encrypt-to-self". These keys are only used when there are other recipients given either by use of --recipient or by the asked user id. No trust checking is performed for these user ids and even disabled keys can be used.

--hidden-encrypt-to name
Same as --hidden-recipient but this one is intended for use in the options file and may be used with your own user-id as a hidden "encrypt-to-self". These keys are only used when there are other recipients given either by use of --recipient or by the asked user id. No trust checking is performed for these user ids and even disabled keys can be used.

--no-encrypt-to
Disable the use of all --encrypt-to and --hidden-encrypt-to keys.

--group name=value1
Sets up a named group, which is similar to aliases in email programs. Any time the group name is a recipient (-r or --recipient), it will be expanded to the values specified. Multiple groups with the same name are automatically merged into a single group.

The values are key IDs or fingerprints, but any key description is accepted. Note that a value with spaces in it will be treated as two different values. Note also there is only one level of expansion --- you cannot make an group that points to another group. When used from the command line, it may be necessary to quote the argument to this option to prevent the shell from treating it as multiple arguments.

--ungroup name
Remove a given entry from the --group list.

--no-groups
Remove all entries from the --group list.

--local-user name

-u Use name as the key to sign with. Note that this option overrides --default-key.

--try-secret-key name
For hidden recipients GPG needs to know the keys to use for trial decryption. The key set with --default-key is always tried first, but this is often not sufficient. This option allows to set more keys to be used for trial decryption. Although any valid user-id specification may be used for name it makes sense to use at least the long keyid to avoid ambiguities. Note that gpg-agent might pop up a pinentry for a lot keys to do the trial decryption. If you want to stop all further trial decryption you may use close-window button instead of the cancel button.

--try-all-secrets
Don't look at the key ID as stored in the message but try all secret keys in turn to find the right decryption key. This option forces the behaviour as used by anonymous recipients (created by using --throw-keyids or --hidden-recipient) and might come handy in case where an encrypted message contains a bogus key ID.

--skip-hidden-recipients

--no-skip-hidden-recipients
During decryption skip all anonymous recipients. This option helps in the case that people use the hidden recipients feature to hide there own encrypt-to key from others. If oneself has many secret keys this may lead to a major annoyance because all keys are tried in turn to decrypt soemthing which was not really intended for it. The drawback of this option is that it is currently not possible to decrypt a message which includes real anonymous recipients.

Input and Output

--armor

-a Create ASCII armored output. The default is to create the binary OpenPGP format.

--no-armor
Assume the input data is not in ASCII armored format.

--output file

-o file
Write output to file.

--max-output n
This option sets a limit on the number of bytes that will be generated when processing a file. Since OpenPGP supports various levels of compression, it is possible that the plaintext of a given message may be significantly larger than the original OpenPGP message. While GnuPG works properly with such messages, there is often a desire to set a maximum file size that will be generated before processing is forced to stop by the OS limits. Defaults to 0, which means "no limit".

--import-options parameters
This is a space or comma delimited string that gives options for importing keys. Options can be prepended with a `no-' to give the opposite meaning. The options are:

import-local-sigs
Allow importing key signatures marked as "local". This is not generally useful unless a shared keyring scheme is being used. Defaults to no.

repair-pks-subkey-bug
During import, attempt to repair the damage caused by the PKS keyserver bug (pre version 0.9.6) that mangles keys with multiple subkeys. Note that this cannot completely repair the damaged key as some crucial data is removed by the keyserver, but it does at least give you back one subkey. Defaults to no for regular --import and to yes for keyserver --recv-keys.

merge-only
During import, allow key updates to existing keys, but do not allow any new keys to be imported. Defaults to no.

import-clean
After import, compact (remove all signatures except the self-signature) any user IDs from the new key that are not usable. Then, remove any signatures from the new key that are not usable. This includes signatures that were issued by keys that are not present on the keyring. This option is the same as running the --edit-key command "clean" after import. Defaults to no.

import-minimal
Import the smallest key possible. This removes all signatures except the most recent self-signature on each user ID. This option is the same as running the --edit-key command "minimize" after import. Defaults to no.

--export-options parameters
This is a space or comma delimited string that gives options for exporting keys. Options can be prepended with a `no-' to give the opposite meaning. The options are:

export-local-sigs
Allow exporting key signatures marked as "local". This is not generally useful unless a shared keyring scheme is being used. Defaults to no.

export-attributes
Include attribute user IDs (photo IDs) while exporting. This is useful to export keys if they are going to be used by an OpenPGP program that does not accept attribute user IDs. Defaults to yes.

export-sensitive-revkeys
Include designated revoker information that was marked as "sensitive". Defaults to no.

export-reset-subkey-passwd
When using the --export-secret-subkeys command, this option resets the passphrases for all exported subkeys to empty. This is useful when the exported subkey is to be used on an unattended machine where a passphrase doesn't necessarily make sense. Defaults to no.

export-clean
Compact (remove all signatures from) user IDs on the key being exported if the user IDs are not usable. Also, do not export any signatures that are not usable. This includes signatures that were issued by keys that are not present on the keyring. This option is the same as running the --edit-key command "clean" before export except that the local copy of the key is not modified. Defaults to no.

export-minimal
Export the smallest key possible. This removes all signatures except the most recent self-signature on each user ID. This option is the same as running the --edit-key command "minimize" before export except that the local copy of the key is not modified. Defaults to no.

--with-colons
Print key listings delimited by colons. Note that the output will be encoded in UTF-8 regardless of any --display-charset setting. This format is useful when GnuPG is called from scripts and other programs as it is easily machine parsed. The details of this format are documented in the file `doc/DETAILS', which is included in the GnuPG source distribution.

--fixed-list-mode
Do not merge primary user ID and primary key in --with-colon listing mode and print all timestamps as seconds since 1970-01-01.

--with-fingerprint
Same as the command --fingerprint but changes only the format of the output and may be used together with another command.

--with-keygrip
Include the keygrip in the key listings.

OpenPGP protocol specific options.

-t, --textmode

--no-textmode
Treat input files as text and store them in the OpenPGP canonical text form with standard "CRLF" line endings. This also sets the necessary flags to inform the recipient that the encrypted or signed data is text and may need its line endings converted back to whatever the local system uses. This option is useful when communicating between two platforms that have different line ending conventions (UNIX-like to Mac, Mac to Windows, etc). --no-textmode disables this option, and is the default.

If -t (but not --textmode) is used together with armoring and signing, this enables clearsigned messages. This kludge is needed for command-line compatibility with command-line versions of PGP; normally you would use --sign or --clearsign to select the type of the signature.

--force-v3-sigs

--no-force-v3-sigs
OpenPGP states that an implementation should generate v4 signatures but PGP versions 5 through 7 only recognize v4 signatures on key material. This option forces v3 signatures for signatures on data. Note that this option implies --no-ask-sig-expire, and unsets --sig-policy-url, --sig-notation, and --sig-keyserver-url, as these features cannot be used with v3 signatures. --no-force-v3-sigs disables this option. Defaults to no.

--force-v4-certs

--no-force-v4-certs
Always use v4 key signatures even on v3 keys. This option also changes the default hash algorithm for v3 RSA keys from MD5 to SHA-1. --no-force-v4-certs disables this option.

--force-mdc
Force the use of encryption with a modification detection code. This is always used with the newer ciphers (those with a block-size greater than 64 bits), or if all of the recipient keys indicate MDC support in their feature flags.

--disable-mdc
Disable the use of the modification detection code. Note that by using this option, the encrypted message becomes vulnerable to a message modification attack.

--personal-cipher-preferences string
Set the list of personal cipher preferences to string. Use gpg --version to get a list of available algorithms, and use none to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked cipher in this list is also used for the --symmetric encryption command.

--personal-digest-preferences string
Set the list of personal digest preferences to string. Use gpg --version to get a list of available algorithms, and use none to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked digest algorithm in this list is also used when signing without encryption (e.g. --clearsign or --sign). The default value is SHA-1.

--personal-compress-preferences string
Set the list of personal compression preferences to string. Use gpg --version to get a list of available algorithms, and use none to set no preference at all. This allows the user to safely override the algorithm chosen by the recipient key preferences, as GPG will only select an algorithm that is usable by all recipients. The most highly ranked compression algorithm in this list is also used when there are no recipient keys to consider (e.g. --symmetric).

--s2k-cipher-algo name
Use name as the cipher algorithm used to protect secret keys. The default cipher is CAST5. This cipher is also used for conventional encryption if --personal-cipher-preferences and --cipher-algo is not given.

--s2k-digest-algo name
Use name as the digest algorithm used to mangle the passphrases. The default algorithm is SHA-1.

--s2k-mode n
Selects how passphrases are mangled. If n is 0 a plain passphrase (which is not recommended) will be used, a 1 adds a salt to the passphrase and a 3 (the default) iterates the whole process a number of times (see --s2k-count). Unless --rfc1991 is used, this mode is also used for conventional encryption.

--s2k-count n
Specify how many times the passphrase mangling is repeated. This value may range between 1024 and 65011712 inclusive, and the default is 65536. Note that not all values in the 1024-65011712 range are legal and if an illegal value is selected, GnuPG will round up to the nearest legal value. This option is only meaningful if --s2k-mode is 3.

Compliance options

These options control what GnuPG is compliant to. Only one of these options may be active at a time. Note that the default setting of this is nearly always the correct one. See the INTEROPERABILITY WITH OTHER OPENPGP PROGRAMS section below before using one of these options.

--gnupg
Use standard GnuPG behavior. This is essentially OpenPGP behavior (see --openpgp), but with some additional workarounds for common compatibility problems in different versions of PGP. This is the default option, so it is not generally needed, but it may be useful to override a different compliance option in the gpg.conf file.

--openpgp
Reset all packet, cipher and digest options to strict OpenPGP behavior. Use this option to reset all previous options like --s2k-*, --cipher-algo, --digest-algo and --compress-algo to OpenPGP compliant values. All PGP workarounds are disabled.

--rfc4880
Reset all packet, cipher and digest options to strict RFC-4880 behavior. Note that this is currently the same thing as --openpgp.

--rfc2440
Reset all packet, cipher and digest options to strict RFC-2440 behavior.

--rfc1991
Try to be more RFC-1991 (PGP 2.x) compliant.

--pgp2 Set up all options to be as PGP 2.x compliant as possible, and warn if an action is taken (e.g. encrypting to a non-RSA key) that will create a message that PGP 2.x will not be able to handle. Note that `PGP 2.x' here means `MIT PGP 2.6.2'. There are other versions of PGP 2.x available, but the MIT release is a good common baseline.

This option implies --rfc1991 --disable-mdc --no-force-v4-certs --escape-from-lines --force-v3-sigs --cipher-algo IDEA --digest-algo MD5 --compress-algo ZIP. It also disables --textmode when encrypting.

--pgp6 Set up all options to be as PGP 6 compliant as possible. This restricts you to the ciphers IDEA (if the IDEA plugin is installed), 3DES, and CAST5, the hashes MD5, SHA1 and RIPEMD160, and the compression algorithms none and ZIP. This also disables --throw-keyids, and making signatures with signing subkeys as PGP 6 does not understand signatures made by signing subkeys.

This option implies --disable-mdc --escape-from-lines --force-v3-sigs.

--pgp7 Set up all options to be as PGP 7 compliant as possible. This is identical to --pgp6 except that MDCs are not disabled, and the list of allowable ciphers is expanded to add AES128, AES192, AES256, and TWOFISH.

--pgp8 Set up all options to be as PGP 8 compliant as possible. PGP 8 is a lot closer to the OpenPGP standard than previous versions of PGP, so all this does is disable --throw-keyids and set --escape-from-lines. All algorithms are allowed except for the SHA224, SHA384, and SHA512 digests.

Doing things one usually doesn't want to do.

-n

--dry-run
Don't make any changes (this is not completely implemented).

--list-only
Changes the behaviour of some commands. This is like --dry-run but different in some cases. The semantic of this command may be extended in the future. Currently it only skips the actual decryption pass and therefore enables a fast listing of the encryption keys.

-i

--interactive
Prompt before overwriting any files.

--debug-level level
Select the debug level for investigating problems. level may be a numeric value or by a keyword:

none No debugging at all. A value of less than 1 may be used instead of the keyword.

basic Some basic debug messages. A value between 1 and 2 may be used instead of the keyword.

advanced
More verbose debug messages. A value between 3 and 5 may be used instead of the keyword.

expert Even more detailed messages. A value between 6 and 8 may be used instead of the keyword.

guru All of the debug messages you can get. A value greater than 8 may be used instead of the keyword. The creation of hash tracing files is only enabled if the keyword is used.

How these messages are mapped to the actual debugging flags is not specified and may change with newer releases of this program. They are however carefully selected to best aid in debugging.

--debug flags
Set debugging flags. All flags are or-ed and flags may be given in C syntax (e.g. 0x0042).

--debug-all
Set all useful debugging flags.

--debug-ccid-driver
Enable debug output from the included CCID driver for smart-cards. Note that this option is only available on some system.

--faked-system-time epoch
This option is only useful for testing; it sets the system time back or forth to epoch which is the number of seconds elapsed since the year 1970. Alternatively epoch may be given as a full ISO time string (e.g. "20070924T154812").

--enable-progress-filter
Enable certain PROGRESS status outputs. This option allows front-tends to display a progress indicator while gpg is processing larger files. There is a slight performance overhead using it.

--status-fd n
Write special status strings to the file descriptor n. See the file DETAILS in the documentation for a listing of them.

--status-file file
Same as --status-fd, except the status data is written to file file.

--logger-fd n
Write log output to file descriptor n and not to STDERR.

--log-file file

--logger-file file
Same as --logger-fd, except the logger data is written to file file. Note that --log-file is only implemented for GnuPG-2.

--attribute-fd n
Write attribute subpackets to the file descriptor n. This is most useful for use with --status-fd, since the status messages are needed to separate out the various subpackets from the

stream delivered to the file descriptor.

--attribute-file file
Same as --attribute-fd, except the attribute data is written to file file.

--comment string

--no-comments
Use string as a comment string in clear text signatures and ASCII armored messages or keys (see --armor). The default behavior is not to use a comment string. --comment may be repeated multiple times to get multiple comment strings. --no-comments removes all comments. It is a good idea to keep the length of a single comment below 60 characters to avoid problems with mail programs wrapping such lines. Note that comment lines, like all other header lines, are not protected by the signature.

--emit-version

--no-emit-version
Force inclusion of the version string in ASCII armored output. --no-emit-version disables this option.

--sig-notation name=value

--cert-notation name=value

-N, --set-notation name=value
Put the name value pair into the signature as notation data. name must consist only of printable characters or spaces, and must contain a '@' character in the form keyname@domain.example.com (substituting the appropriate keyname and domain name, of course). This is to help prevent pollution of the IETF reserved notation namespace. The --expert flag overrides the '@' check. value may be any printable string; it will be encoded in UTF8, so you should check that your --display-charset is set correctly. If you prefix name with an exclamation mark (!), the notation data will be flagged as critical (rfc2440:5.2.3.15). --sig-notation sets a notation for data signatures. --cert-notation sets a notation for key signatures (certifications). --set-notation sets both.

There are special codes that may be used in notation names. "%k" will be expanded into the key ID of the key being signed, "%K" into the long key ID of the key being signed, "%f" into the fingerprint of the key being signed, "%s" into the key ID of the key making the signature, "%S" into the long key ID of the key making the signature, "%g" into the fingerprint of the key making the signature (which might be a subkey), "%p" into the fingerprint of the primary key of the key making the signature, "%c" into the signature count from the OpenPGP smartcard, and "%%" results in a single "%". %k, %K, and %f are only meaningful when making a key signature (certification), and %c is only meaningful when using the OpenPGP smartcard.

--sig-policy-url string

--cert-policy-url string

--set-policy-url string
Use string as a Policy URL for signatures (rfc2440:5.2.3.19). If you prefix it with an exclamation mark (!), the policy URL packet will be flagged as critical. --sig-policy-url sets a policy url for data signatures. --cert-policy-url sets a policy url for key signatures (certifications). --set-policy-url sets both.

The same %-expandos used for notation data are available here as well.

--sig-keyserver-url string
Use string as a preferred keyserver URL for data signatures. If you prefix it with an exclamation mark (!), the keyserver URL packet will be flagged as critical.

The same %-expandos used for notation data are available here as well.

--set-filename string
Use string as the filename which is stored inside messages. This overrides the default, which is to use the actual filename of the file being encrypted.

--for-your-eyes-only

--no-for-your-eyes-only
Set the 'for your eyes only' flag in the message. This causes GnuPG to refuse to save the file unless the --output option is given, and PGP to use a "secure viewer" with a claimed Tempest-resistant font to display the message. This option overrides --set-filename. --no-for-your-eyes-only disables this option.

--use-embedded-filename

--no-use-embedded-filename
Try to create a file with a name as embedded in the data. This can be a dangerous option as it allows to overwrite files. Defaults to no.

--cipher-algo name
Use name as cipher algorithm. Running the program with the command --version yields a list of supported algorithms. If this is not used the cipher algorithm is selected from the preferences stored with the key. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. --personal-cipher-preferences is the safe way to accomplish the same thing.

--digest-algo name
Use name as the message digest algorithm. Running the program with the command --version yields a list of supported algorithms. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. --personal-digest-preferences is the safe way to accomplish the same thing.

--compress-algo name
Use compression algorithm name. "zlib" is RFC-1950 ZLIB compression. "zip" is RFC-1951 ZIP compression which is used by PGP. "bzip2" is a more modern compression scheme that can compress some things better than zip or zlib, but at the cost of more memory used during compression and decompression. "uncompressed" or "none" disables compression. If this option is not used, the default behavior is to examine the recipient key preferences to see which algorithms the recipient supports. If all else fails, ZIP is used for maximum compatibility.

ZLIB may give better compression results than ZIP, as the compression window size is not limited to 8k. BZIP2 may give even better compression results than that, but will use a significantly larger amount of memory while compressing and decompressing. This may be significant in low memory situations. Note, however, that PGP (all versions) only supports ZIP compression. Using any algorithm other than ZIP or "none" will make the message unreadable with PGP. In general, you do not want to use this option as it allows you to violate the OpenPGP standard. --personal-compress-preferences is the safe way to accomplish the same thing.

--cert-digest-algo name
Use name as the message digest algorithm used when signing a key. Running the program with the command --version yields a list of supported algorithms. Be aware that if you choose an algorithm that GnuPG supports but other OpenPGP implementations do not, then some users will not be able to use the key signatures you make, or quite possibly your entire key.

--disable-cipher-algo name
Never allow the use of name as cipher algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--disable-pubkey-algo name
Never allow the use of name as public key algorithm. The given name will not be checked so that a later loaded algorithm will still get disabled.

--throw-keyids

--no-throw-keyids
Do not put the recipient key IDs into encrypted messages. This helps to hide the receivers of the message and is a limited countermeasure against traffic analysis. ([Using a little social engineering anyone who is able to decrypt the message can check whether one of the other recipients is the one he suspects.]) On the receiving side, it may slow down the decryption process because all available secret keys must be tried. --no-throw-keyids disables this option. This option is essentially the same as using --hidden-recipient for all recipients.

--not-dash-escaped
This option changes the behavior of cleartext signatures so that they can be used for patch files. You should not send such an

armored file via email because all spaces and line endings are hashed too. You can not use this option for data which has 5 dashes at the beginning of a line, patch files don't have this. A special armor header line tells GnuPG about this cleartext signature option.

--escape-from-lines

--no-escape-from-lines
Because some mailers change lines starting with "From " to ">From " it is good to handle such lines in a special way when creating cleartext signatures to prevent the mail system from breaking the signature. Note that all other PGP versions do it this way too. Enabled by default. --no-escape-from-lines disables this option.

--passphrase-repeat n
Specify how many times gpg will request a new passphrase be repeated. This is useful for helping memorize a passphrase. Defaults to 1 repetition.

--passphrase-fd n
Read the passphrase from file descriptor n. Only the first line will be read from file descriptor n. If you use 0 for n, the passphrase will be read from STDIN. This can only be used if only one passphrase is supplied.

--passphrase-file file
Read the passphrase from file file. Only the first line will be read from file file. This can only be used if only one passphrase is supplied. Obviously, a passphrase stored in a file is of questionable security if other users can read this file. Don't use this option if you can avoid it.

--passphrase string
Use string as the passphrase. This can only be used if only one passphrase is supplied. Obviously, this is of very questionable security on a multi-user system. Don't use this option if you can avoid it.

--command-fd n
This is a replacement for the deprecated shared-memory IPC mode. If this option is enabled, user input on questions is not expected from the TTY but from the given file descriptor. It should be used together with --status-fd. See the file doc/DETAILS in the source distribution for details on how to use it.

--command-file file
Same as --command-fd, except the commands are read out of file file

--allow-non-selfsigned-uid

--no-allow-non-selfsigned-uid

Allow the import and use of keys with user IDs which are not self-signed. This is not recommended, as a non self-signed user ID is trivial to forge. --no-allow-non-selfsigned-uid disables.

--allow-freeform-uid
Disable all checks on the form of the user ID while generating a new one. This option should only be used in very special environments as it does not ensure the de-facto standard format of user IDs.

--ignore-time-conflict
GnuPG normally checks that the timestamps associated with keys and signatures have plausible values. However, sometimes a signature seems to be older than the key due to clock problems. This option makes these checks just a warning. See also --ignore-valid-from for timestamp issues on subkeys.

--ignore-valid-from
GnuPG normally does not select and use subkeys created in the future. This option allows the use of such keys and thus exhibits the pre-1.0.7 behaviour. You should not use this option unless you there is some clock problem. See also --ignore-time-conflict for timestamp issues with signatures.

--ignore-crc-error
The ASCII armor used by OpenPGP is protected by a CRC checksum against transmission errors. Occasionally the CRC gets mangled somewhere on the transmission channel but the actual content (which is protected by the OpenPGP protocol anyway) is still okay. This option allows GnuPG to ignore CRC errors.

--ignore-mdc-error
This option changes a MDC integrity protection failure into a warning. This can be useful if a message is partially corrupt, but it is necessary to get as much data as possible out of the corrupt message. However, be aware that a MDC protection failure may also mean that the message was tampered with intentionally by an attacker.

--no-default-keyring
Do not add the default keyrings to the list of keyrings. Note that GnuPG will not operate without any keyrings, so if you use this option and do not provide alternate keyrings via --keyring or --secret-keyring, then GnuPG will still use the default public or secret keyrings.

--skip-verify
Skip the signature verification step. This may be used to make the decryption faster if the signature verification is not needed.

--with-key-data
Print key listings delimited by colons (like --with-colons) and print the public key data.

--fast-list-mode
Changes the output of the list commands to work faster; this is achieved by leaving some parts empty. Some applications don't need the user ID and the trust information given in the listings. By using this options they can get a faster listing. The exact behaviour of this option may change in future versions. If you are missing some information, don't use this option.

--no-literal
This is not for normal use. Use the source to see for what it might be useful.

--set-filesize
This is not for normal use. Use the source to see for what it might be useful.

--show-session-key
Display the session key used for one message. See --override-session-key for the counterpart of this option.

We think that Key Escrow is a Bad Thing; however the user should have the freedom to decide whether to go to prison or to reveal the content of one specific message without compromising all messages ever encrypted for one secret key. DON'T USE IT UNLESS YOU ARE REALLY FORCED TO DO SO.

--override-session-key string
Don't use the public key but the session key string. The format of this string is the same as the one printed by --show-session-key. This option is normally not used but comes handy in case someone forces you to reveal the content of an encrypted message; using this option you can do this without handing out the secret key.

--ask-sig-expire

--no-ask-sig-expire
When making a data signature, prompt for an expiration time. If this option is not specified, the expiration time set via --default-sig-expire is used. --no-ask-sig-expire disables this option.

--default-sig-expire
The default expiration time to use for signature expiration. Valid values are "0" for no expiration, a number followed by the letter d (for days), w (for weeks), m (for months), or y (for years) (for example "2m" for two months, or "5y" for five years), or an absolute date in the form YYYY-MM-DD. Defaults to "0".

--ask-cert-expire

--no-ask-cert-expire
When making a key signature, prompt for an expiration time. If this option is not specified, the expiration time set via

--default-cert-expire is used. --no-ask-cert-expire disables this option.

--default-cert-expire
The default expiration time to use for key signature expiration. Valid values are "0" for no expiration, a number followed by the letter d (for days), w (for weeks), m (for months), or y (for years) (for example "2m" for two months, or "5y" for five years), or an absolute date in the form YYYY-MM-DD. Defaults to "0".

--allow-secret-key-import
This is an obsolete option and is not used anywhere.

--allow-multiple-messages

--no-allow-multiple-messages
Allow processing of multiple OpenPGP messages contained in a single file or stream. Some programs that call GPG are not prepared to deal with multiple messages being processed together, so this option defaults to no. Note that versions of GPG prior to 1.4.7 always allowed multiple messages.

Warning: Do not use this option unless you need it as a temporary workaround!

--enable-special-filenames
This options enables a mode in which filenames of the form `-&n', where n is a non-negative decimal number, refer to the file descriptor n and not to a file with that name.

--no-expensive-trust-checks
Experimental use only.

--preserve-permissions
Don't change the permissions of a secret keyring back to user read/write only. Use this option only if you really know what you are doing.

--default-preference-list string
Set the list of default preferences to string. This preference list is used for new keys and becomes the default for "setpref" in the edit menu.

--default-keyserver-url name
Set the default keyserver URL to name. This keyserver will be used as the keyserver URL when writing a new self-signature on a key, which includes key generation and changing preferences.

--list-config
Display various internal configuration parameters of GnuPG. This option is intended for external programs that call GnuPG to perform tasks, and is thus not generally useful. See the file

`doc/DETAILS' in the source distribution for the details of which configuration items may be listed. --list-config is only usable with --with-colons set.

--gpgconf-list
This command is similar to --list-config but in general only internally used by the gpgconf tool.

--gpgconf-test
This is more or less dummy action. However it parses the configuration file and returns with failure if the configuration file would prevent gpg from startup. Thus it may be used to run a syntax check on the configuration file.

Deprecated options

--load-extension name
Load an extension module. If name does not contain a slash it is searched for in the directory configured when GnuPG was built (generally "/usr/local/lib/gnupg"). Extensions are not generally useful anymore, and the use of this option is deprecated.

--show-photos

--no-show-photos
Causes --list-keys, --list-sigs, --list-public-keys, --list-secret-keys, and verifying a signature to also display the photo ID attached to the key, if any. See also --photo-viewer. These options are deprecated. Use --list-options [no-]show-photos and/or --verify-options [no-]show-photos instead.

--show-keyring
Display the keyring name at the head of key listings to show which keyring a given key resides on. This option is deprecated: use --list-options [no-]show-keyring instead.

--ctapi-driver file
Use file to access the smartcard reader. The current default is `libtowitoko.so'. Note that the use of this interface is deprecated; it may be removed in future releases.

--always-trust
Identical to --trust-model always. This option is deprecated.

--show-notation

--no-show-notation
Show signature notations in the --list-sigs or --check-sigs listings as well as when verifying a signature with a notation in it. These options are deprecated. Use --list-options

[no-]show-notation and/or --verify-options [no-]show-notation instead.

--show-policy-url

--no-show-policy-url
Show policy URLs in the --list-sigs or --check-sigs listings as well as when verifying a signature with a policy URL in it. These options are deprecated. Use --list-options [no-]show-policy-url and/or --verify-options [no-]show-policy-url instead.

EXAMPLES

gpg -se -r Bob file
sign and encrypt for user Bob

gpg --clearsign file
make a clear text signature

gpg -sb file
make a detached signature

gpg -u 0x12345678 -sb file
make a detached signature with the key 0x12345678

gpg --list-keys user_ID
show keys

gpg --fingerprint user_ID
show fingerprint

gpg --verify pgpfile

gpg --verify sigfile
Verify the signature of the file but do not output the data. The second form is used for detached signatures, where sigfile is the detached signature (either ASCII armored or binary) and are the signed data; if this is not given, the name of the file holding the signed data is constructed by cutting off the extension (".asc" or ".sig") of sigfile or by asking the user for the filename.

HOW TO SPECIFY A USER ID

There are different ways to specify a user ID to GnuPG. Some of them are only valid for gpg others are only good for gpgsm. Here is the entire list of ways to specify a key:

By key Id.
This format is deduced from the length of the string and its content or 0x prefix. The key Id of an X.509 certificate are the low 64 bits of its SHA-1 fingerprint. The use of key Ids is just a shortcut, for all automated processing the fingerprint should be used.

When using gpg an exclamation mark (!) may be appended to force using the specified primary or secondary key and not to try and calculate which primary or secondary key to use.

The last four lines of the example give the key ID in their long form as internally used by the OpenPGP protocol. You can see the long key ID using the option --with-colons.

234567C4
0F34E556E
01347A56A
0xAB123456

234AABBCC34567C4
0F323456784E56EAB
01AB3FED1347A5612
0x234AABBCC34567C4

By fingerprint.
This format is deduced from the length of the string and its content or the 0x prefix. Note, that only the 20 byte version fingerprint is available with gpgsm (i.e. the SHA-1 hash of the certificate).

When using gpg an exclamation mark (!) may be appended to force using the specified primary or secondary key and not to try and calculate which primary or secondary key to use.

The best way to specify a key Id is by using the fingerprint. This avoids any ambiguities in case that there are duplicated key IDs.

1234343434343434C434343434343434
123434343434343C34343434343434734349A3434
0E1234343434343434343434EAB3484343434343434
0xE1234343434343434343434EAB3484343434343434

(gpgsm also accepts colons between each pair of hexadecimal digits because this is the de-facto standard on how to present X.509 fingerprints.)

By exact match on OpenPGP user ID.
This is denoted by a leading equal sign. It does not make sense for X.509 certificates.

=Heinrich Heine <heinrichh@uni-duesseldorf.de>

By exact match on an email address.
This is indicated by enclosing the email address in the usual way with left and right angles.

<heinrichh@uni-duesseldorf.de>

By word match.
All words must match exactly (not case sensitive) but can appear in any order in the user ID or a subjects name. Words are any sequences of letters, digits, the underscore and all characters with bit 7 set.

+Heinrich Heine duesseldorf

By exact match on the subject's DN.
This is indicated by a leading slash, directly followed by the RFC-2253 encoded DN of the subject. Note that you can't use the string printed by "gpgsm --list-keys" because that one as been reordered and modified for better readability; use --with-colons to print the raw (but standard escaped) RFC-2253 string

/CN=Heinrich Heine,O=Poets,L=Paris,C=FR

By exact match on the issuer's DN.
This is indicated by a leading hash mark, directly followed by a slash and then directly followed by the rfc2253 encoded DN of the issuer. This should return the Root cert of the issuer. See note above.

#/CN=Root Cert,O=Poets,L=Paris,C=FR

By exact match on serial number and issuer's DN.
This is indicated by a hash mark, followed by the hexadecimal representation of the serial number, then followed by a slash and the RFC-2253 encoded DN of the issuer. See note above.

#4F03/CN=Root Cert,O=Poets,L=Paris,C=FR

By keygrip
This is indicated by an ampersand followed by the 40 hex digits of a keygrip. gpgsm prints the keygrip when using the command --dump-cert. It does not yet work for OpenPGP keys.

&D75F22C3F86E355877348498CDC92BD21010A480

By substring match.
This is the default mode but applications may want to explicitly indicate this by putting the asterisk in front. Match is not case sensitive.

Heine
*Heine

Please note that we have reused the hash mark identifier which was used in old GnuPG versions to indicate the so called local-id. It is not

anymore used and there should be no conflict when used with X.509 stuff.

Using the RFC-2253 format of DNs has the drawback that it is not possible to map them back to the original encoding, however we don't have to do this because our key database stores this encoding as meta data.

FILES

There are a few configuration files to control certain aspects of gpg's operation. Unless noted, they are expected in the current home directory (see: [option --homedir]).

gpg.conf
: This is the standard configuration file read by gpg on startup. It may contain any valid long option; the leading two dashes may not be entered and the option may not be abbreviated. This default name may be changed on the command line (see: [option --options]). You should backup this file.

Note that on larger installations, it is useful to put predefined files into the directory `/etc/skel/.gnupg/' so that newly created users start up with a working configuration.

For internal purposes gpg creates and maintains a few other files; They all live in in the current home directory (see: [option --homedir]). Only the gpg may modify these files.

~/.gnupg/secring.gpg
: The secret keyring. You should backup this file.

~/.gnupg/secring.gpg.lock
: The lock file for the secret keyring.

~/.gnupg/pubring.gpg
: The public keyring. You should backup this file.

~/.gnupg/pubring.gpg.lock
: The lock file for the public keyring.

~/.gnupg/trustdb.gpg
: The trust database. There is no need to backup this file; it is better to backup the ownertrust values (see: [option --export-ownertrust]).

~/.gnupg/trustdb.gpg.lock
: The lock file for the trust database.

~/.gnupg/random_seed
A file used to preserve the state of the internal random pool.

/usr[/local]/share/gnupg/options.skel
The skeleton options file.

/usr[/local]/lib/gnupg/
Default location for extensions.

Operation is further controlled by a few environment variables:

HOME Used to locate the default home directory.

GNUPGHOME
If set directory used instead of "~/.gnupg".

GPG_AGENT_INFO
Used to locate the gpg-agent. This is only honored when --use-agent is set. The value consists of 3 colon delimited fields: The first is the path to the Unix Domain Socket, the second the PID of the gpg-agent and the protocol version which should be set to 1. When starting the gpg-agent as described in its documentation, this variable is set to the correct value. The option --gpg-agent-info can be used to override it.

PINENTRY_USER_DATA
This value is passed via gpg-agent to pinentry. It is useful to convey extra information to a custom pinentry.

COLUMNS

LINES Used to size some displays to the full size of the screen.

LANGUAGE
Apart from its use by GNU, it is used in the W32 version to override the language selection done through the Registry. If used and set to a valid and available language name (langid), the file with the translation is loaded from gpgdir/gnupg.nls/langid.mo. Here gpgdir is the directory out of which the gpg binary has been loaded. If it can't be loaded the Registry is tried and as last resort the native Windows locale system is used.

BUGS

On older systems this program should be installed as setuid(root). This is necessary to lock memory pages. Locking memory pages prevents the operating system from writing memory pages (which may contain passphrases or other sensitive material) to disk. If you get no warning

message about insecure memory your operating system supports locking without being root. The program drops root privileges as soon as locked memory is allocated.

Note also that some systems (especially laptops) have the ability to ``suspend to disk'' (also known as ``safe sleep'' or ``hibernate''). This writes all memory to disk before going into a low power or even powered off mode. Unless measures are taken in the operating system to protect the saved memory, passphrases or other sensitive material may be recoverable from it later.

Before you report a bug you should first search the mailing list archives for similar problems and second check whether such a bug has already been reported to our bug tracker at http://bugs.gnupg.org .

SEE ALSO

gpgv(1),

The full documentation for this tool is maintained as a Texinfo manual. If GnuPG and the info program are properly installed at your site, the command

```
info gnupg
```

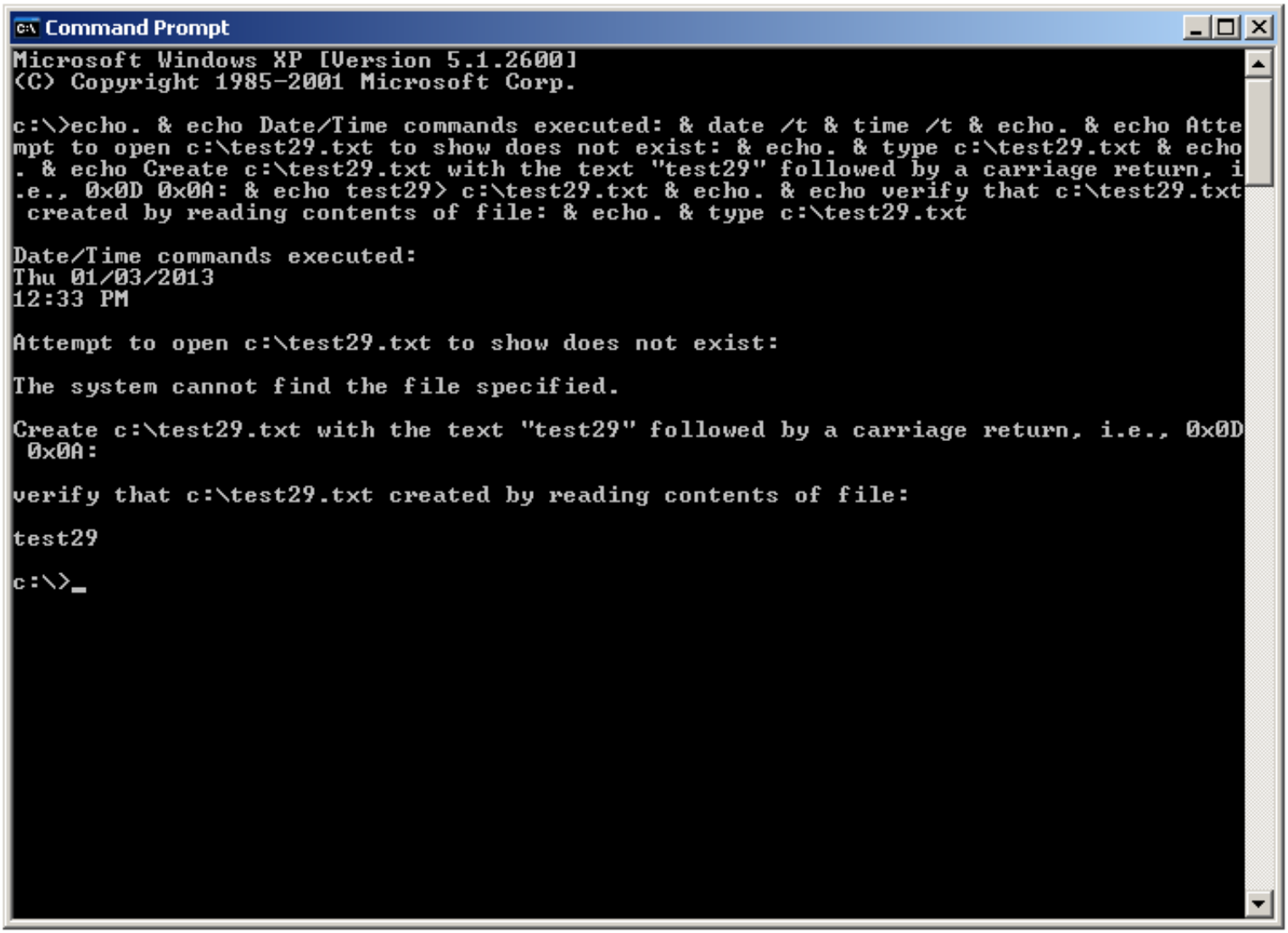
should give you access to the complete manual including a menu structure and an index.

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 17

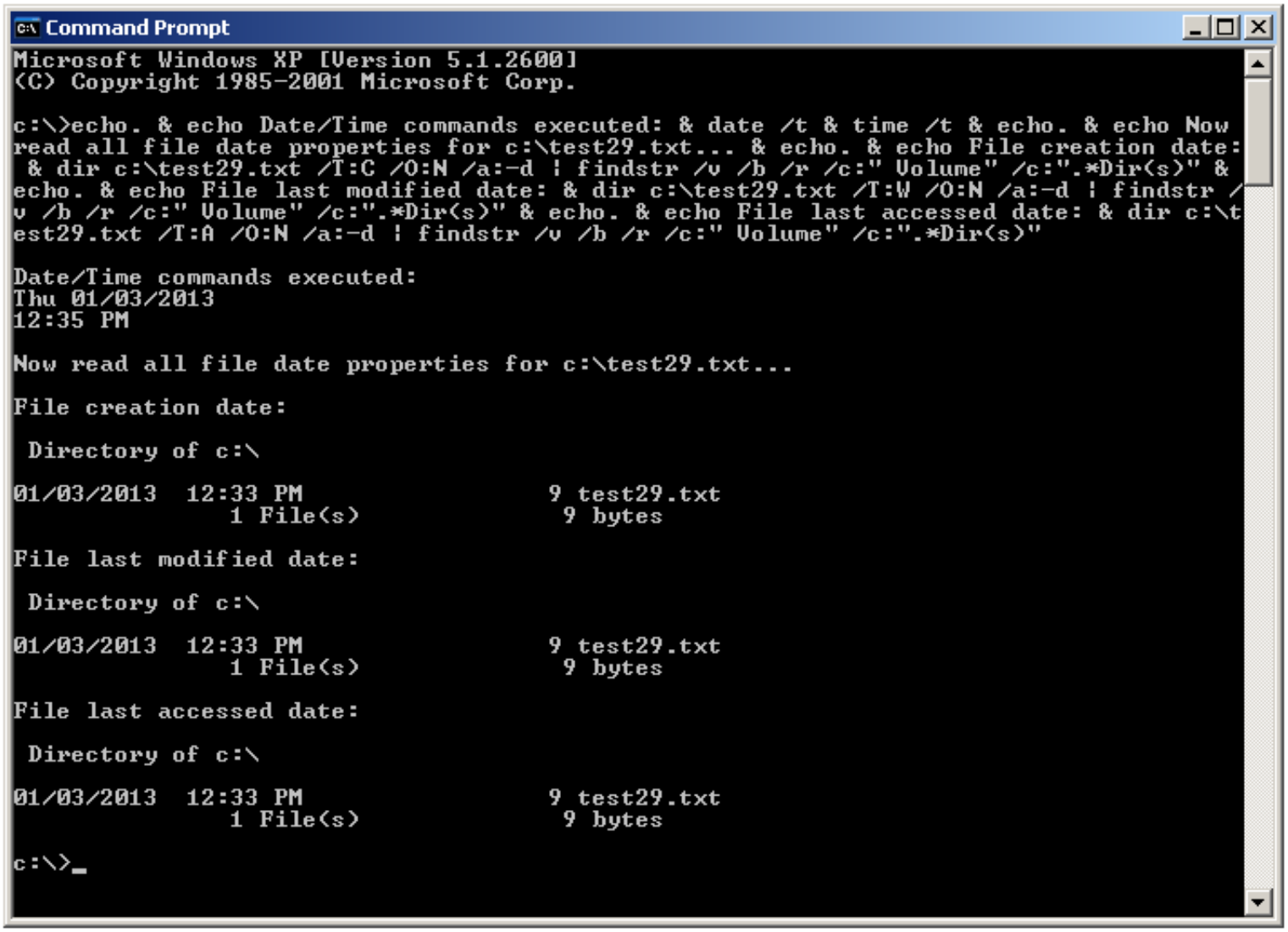

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test29.txt to show does not exist: & echo. & type c:\test29.txt & echo. & echo Create c:\test29.txt with the text "test29" followed by a carriage return, i.e., 0x0D 0x0A: & echo test29> c:\test29.txt & echo. & echo verify that c:\test29.txt created by reading contents of file: & echo. & type c:\test29.txt
Date/Time commands executed:
Thu 01/03/2013
12:33 PM
Attempt to open c:\test29.txt to show does not exist:
The system cannot find the file specified.
Create c:\test29.txt with the text "test29" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test29.txt created by reading contents of file:
test29
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 18

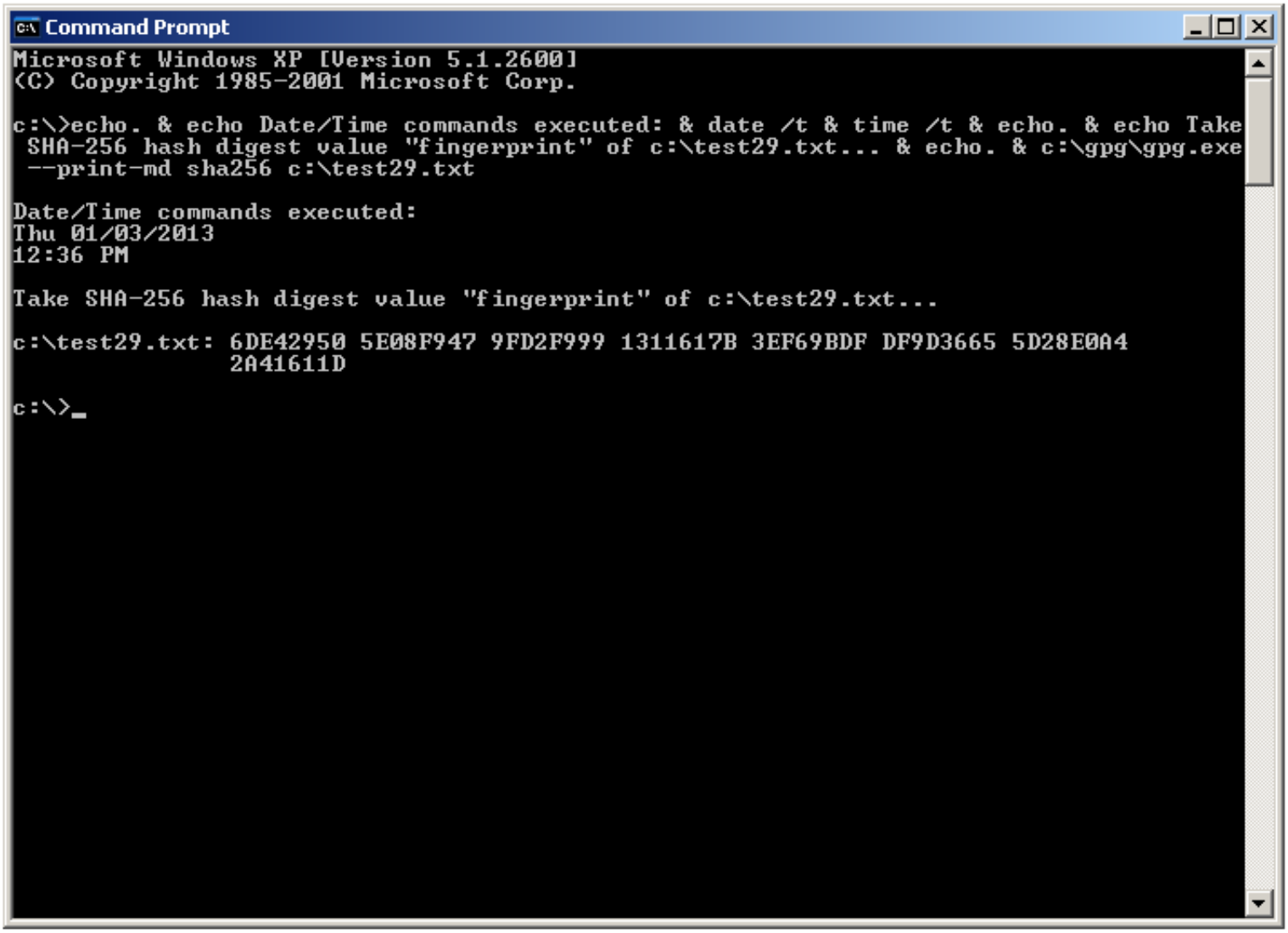

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read all file date properties for c:\test29.txt... & echo. & echo File creation date: & dir c:\test29.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test29.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir c:\test29.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:35 PM
Now read all file date properties for c:\test29.txt...
File creation date:
Directory of c:\
01/03/2013 12:33 PM 9 test29.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:33 PM 9 test29.txt
1 File(s) 9 bytes
File last accessed date:
Directory of c:\
01/03/2013 12:33 PM 9 test29.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 19

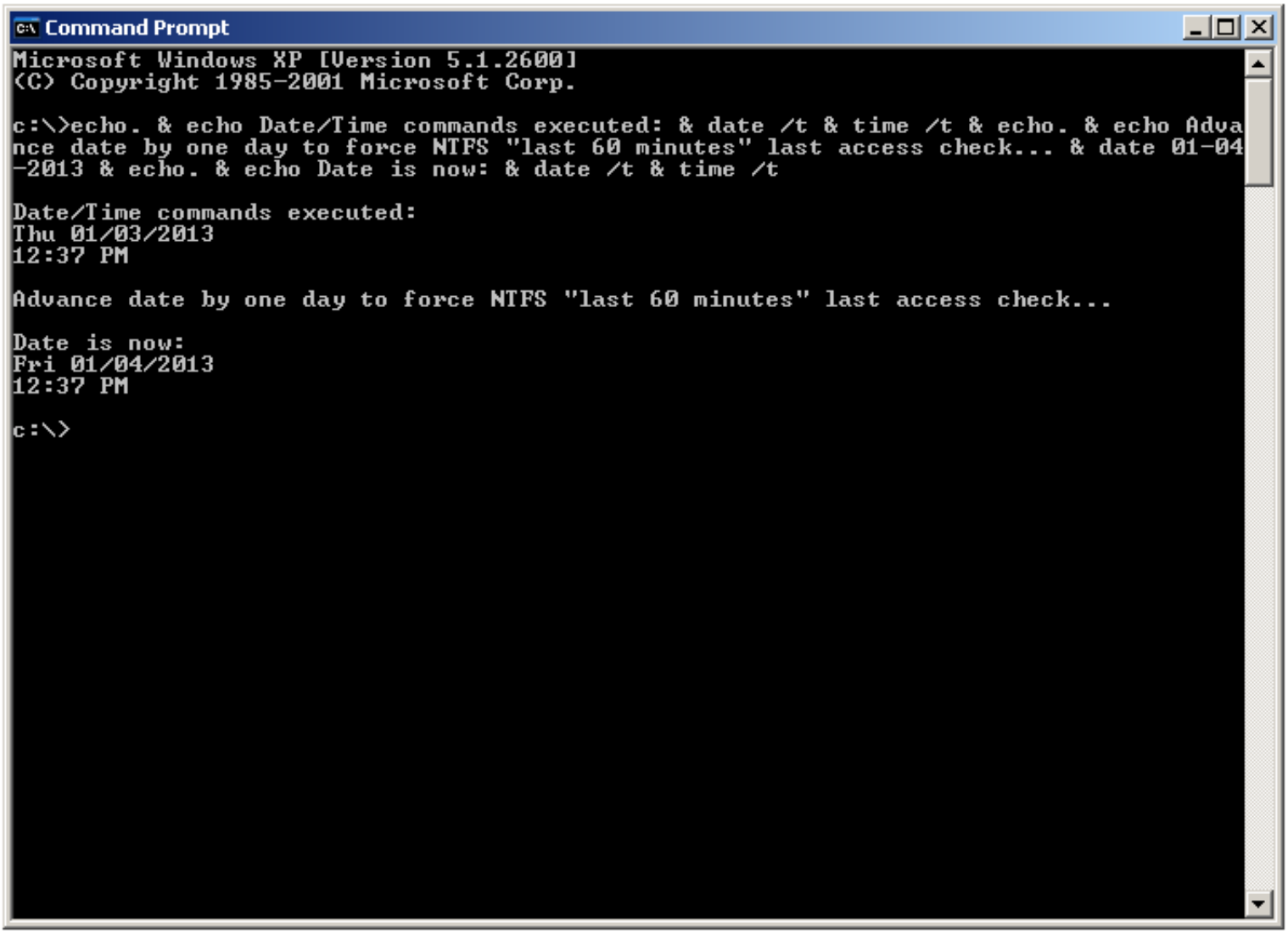

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test29.txt... & echo. & c:\gpg\gpg.exe
--print-md sha256 c:\test29.txt
Date/Time commands executed:
Thu 01/03/2013
12:36 PM
Take SHA-256 hash digest value "fingerprint" of c:\test29.txt...
c:\test29.txt: 6DE42950 5E08F947 9FD2F999 1311617B 3EF69BDF DF9D3665 5D28E0A4
2A41611D
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 20

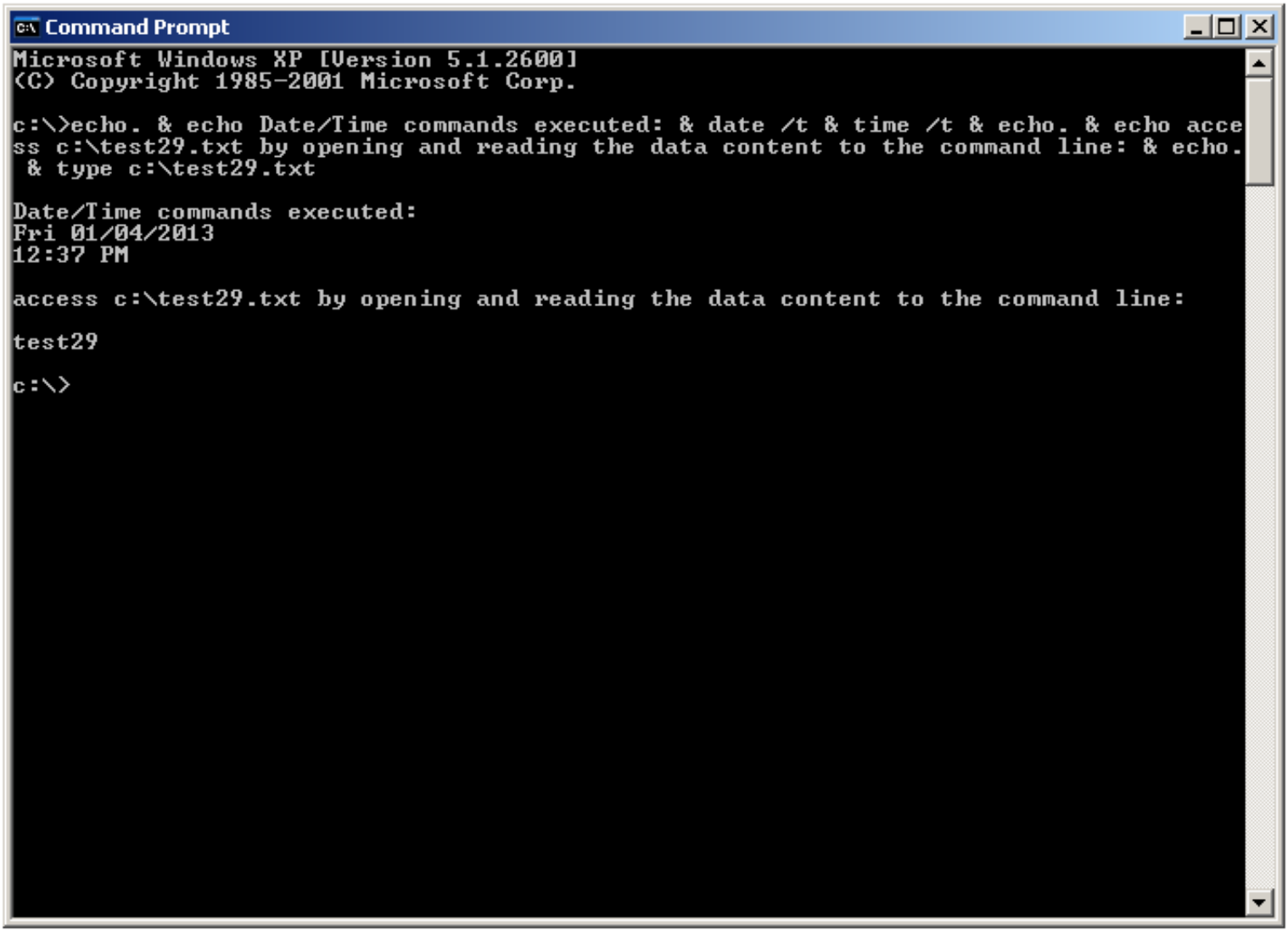

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Adva
nce date by one day to force NTFS "last 60 minutes" last access check... & date 01-04
-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
12:37 PM
Advance date by one day to force NTFS "last 60 minutes" last access check...
Date is now:
Fri 01/04/2013
12:37 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 21

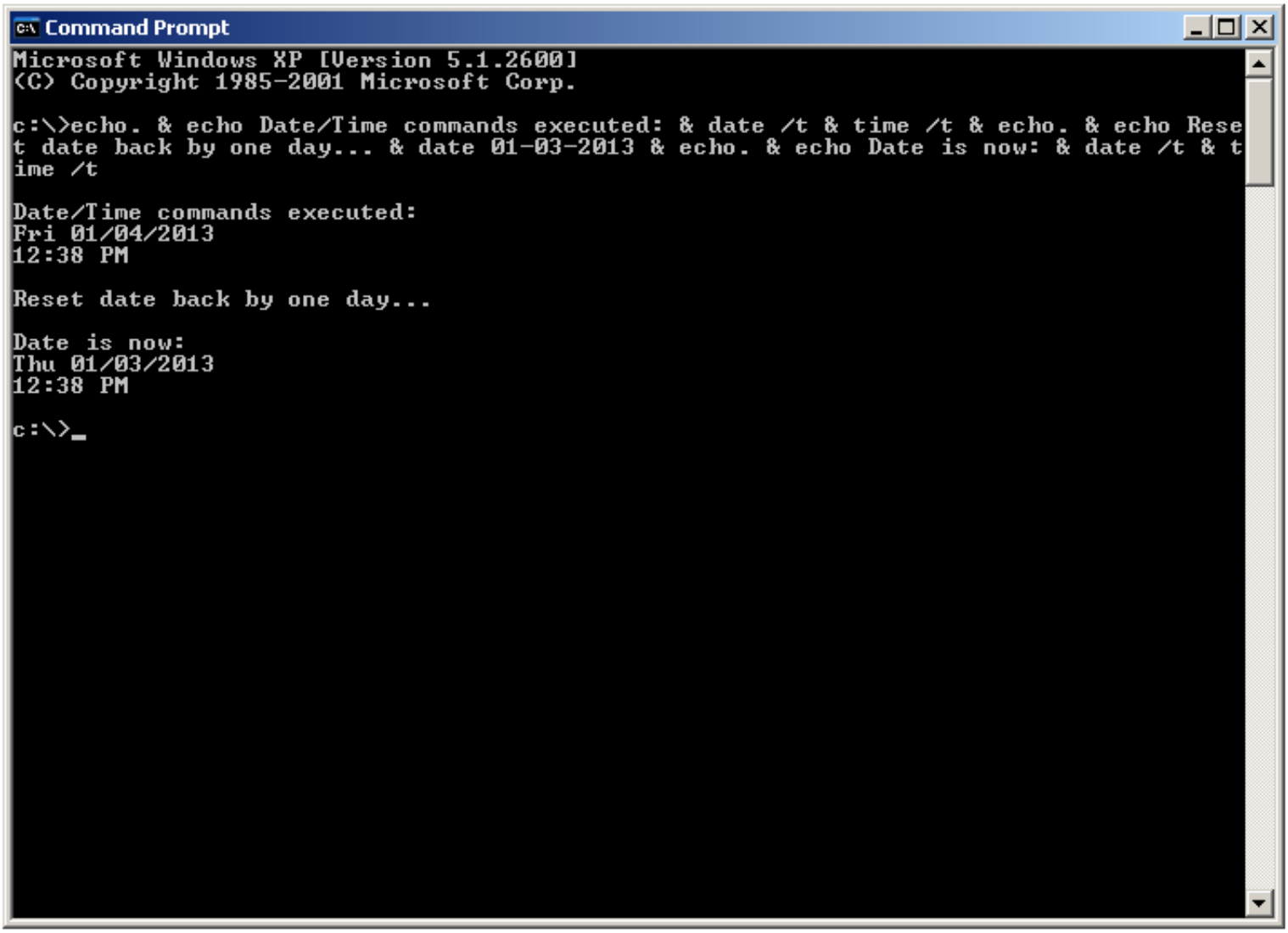

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo acce
ss c:\test29.txt by opening and reading the data content to the command line: & echo.
& type c:\test29.txt
Date/Time commands executed:
Fri 01/04/2013
12:37 PM
access c:\test29.txt by opening and reading the data content to the command line:
test29
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 22

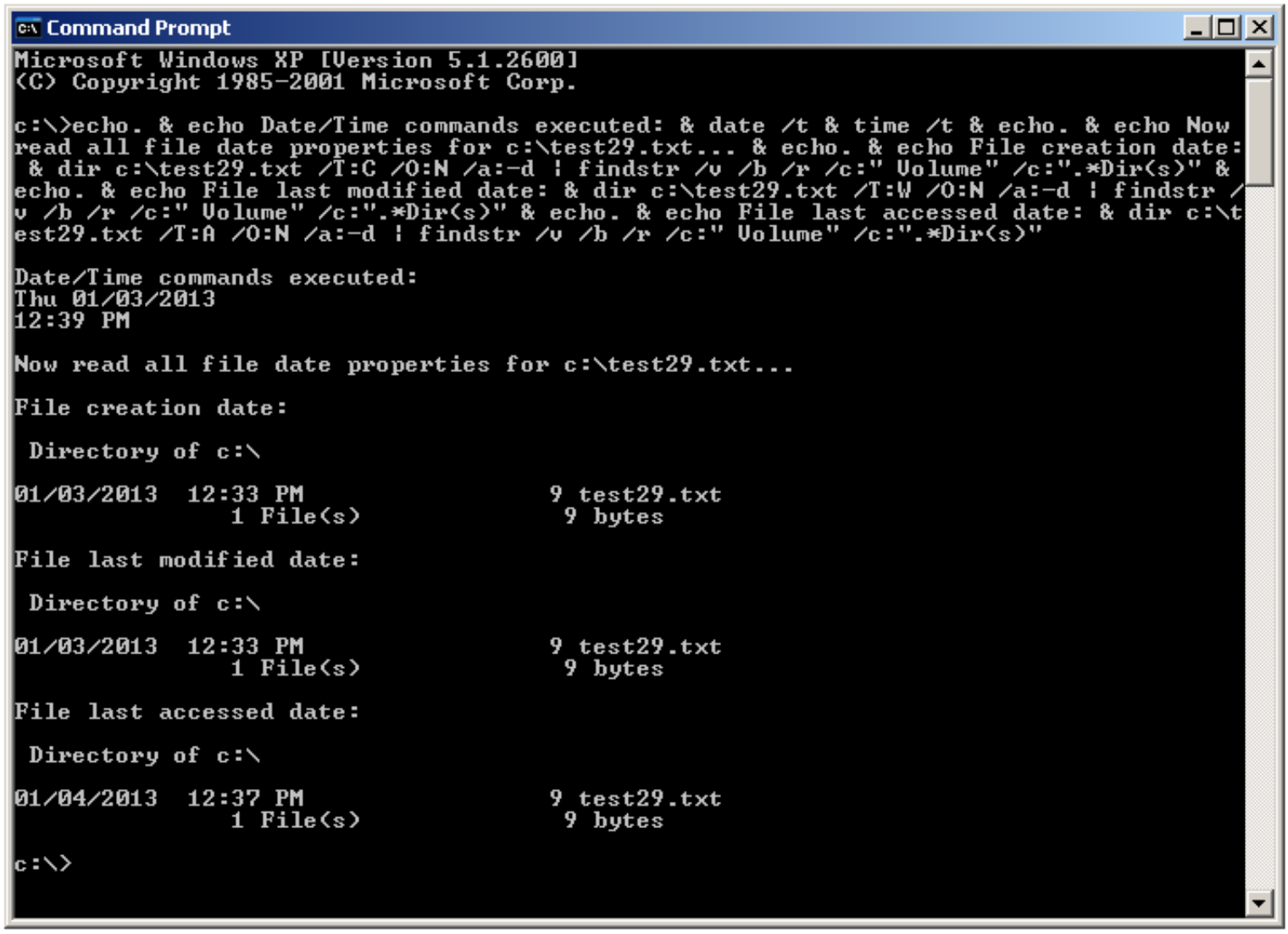

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Rese
t date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & t
ime /t
Date/Time commands executed:
Fri 01/04/2013
12:38 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
12:38 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 23

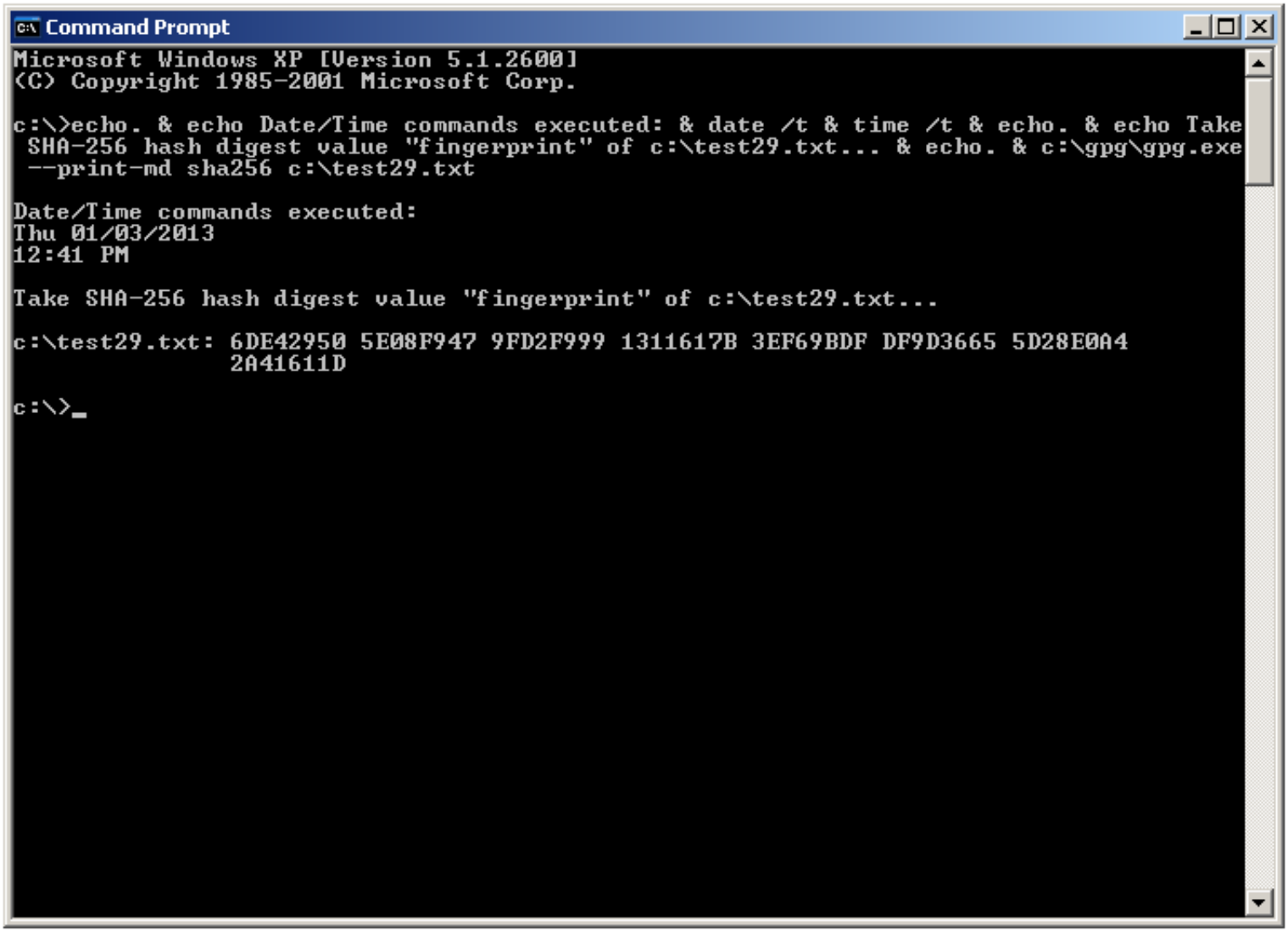

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read all file date properties for c:\test29.txt... & echo. & echo File creation date: & dir c:\test29.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test29.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last accessed date: & dir c:\test29.txt /T:A /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:39 PM
Now read all file date properties for c:\test29.txt...
File creation date:
Directory of c:\
01/03/2013 12:33 PM 9 test29.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:33 PM 9 test29.txt
1 File(s) 9 bytes
File last accessed date:
Directory of c:\
01/04/2013 12:37 PM 9 test29.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 24

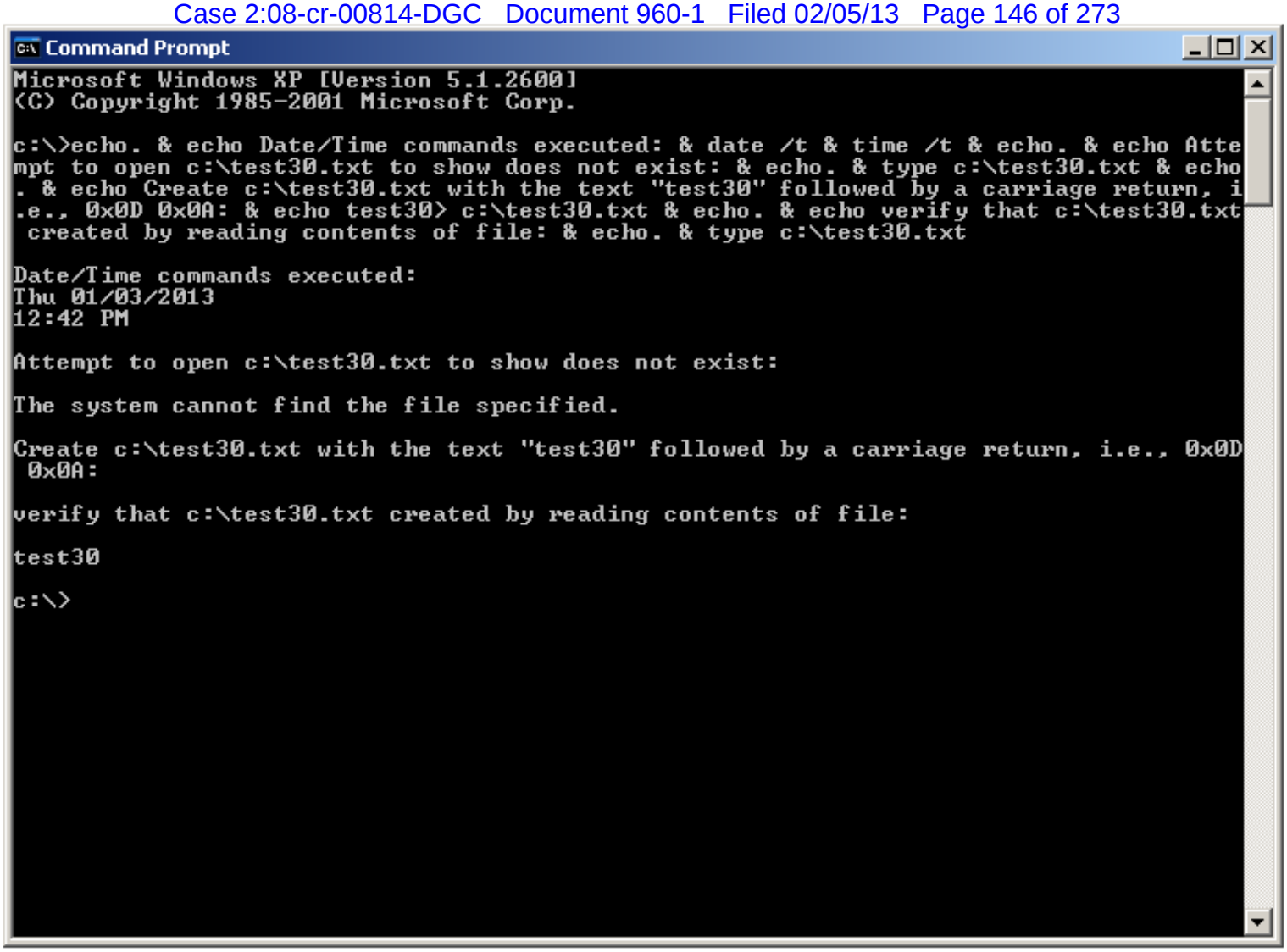

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test29.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test29.txt
Date/Time commands executed:
Thu 01/03/2013
12:41 PM
Take SHA-256 hash digest value "fingerprint" of c:\test29.txt...
c:\test29.txt: 6DE42950 5E08F947 9FD2F999 1311617B 3EF69BDF DF9D3665 5D28E0A4 2A41611D
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 25

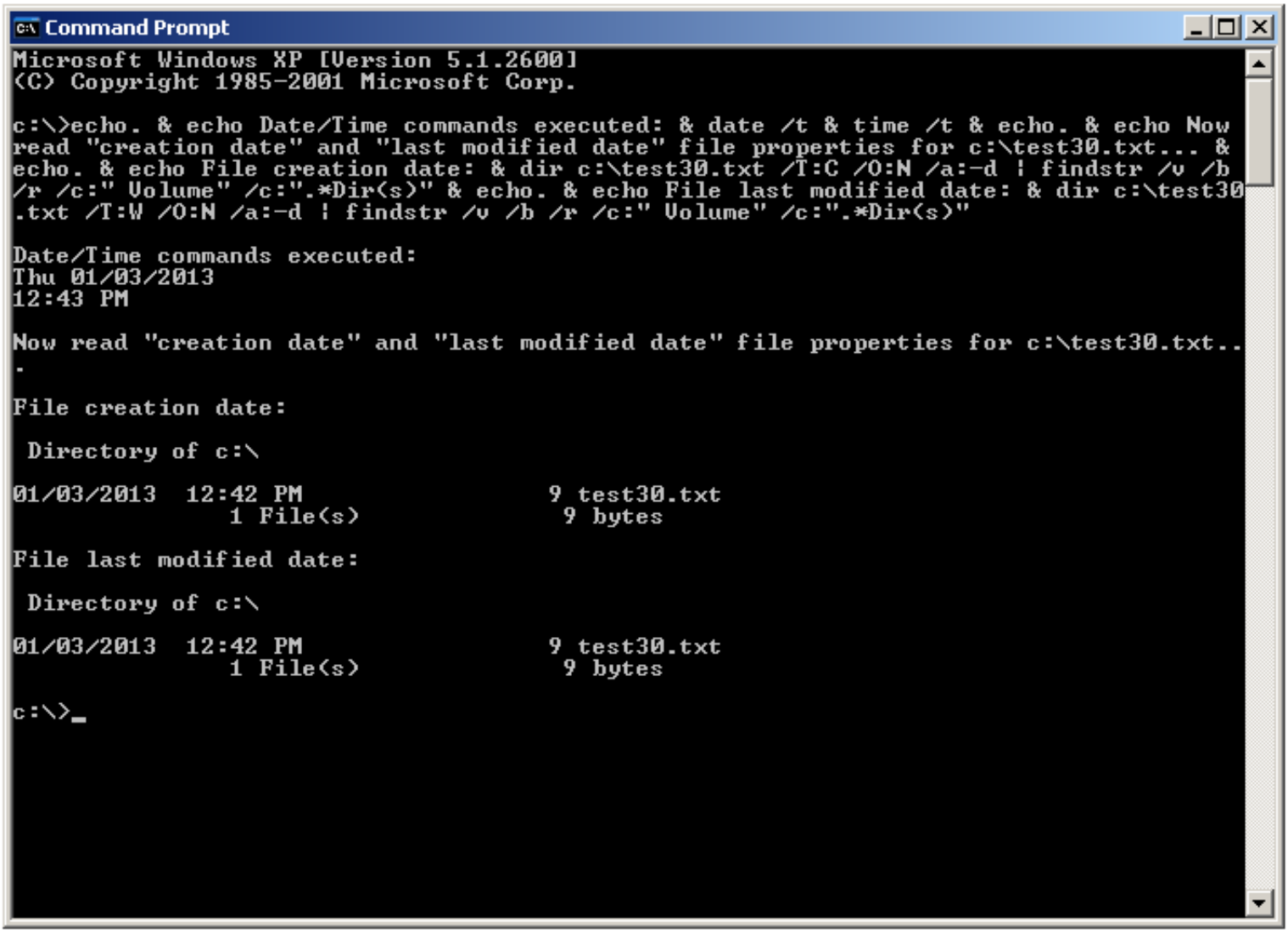

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test30.txt to show does not exist: & echo. & type c:\test30.txt & echo. & echo Create c:\test30.txt with the text "test30" followed by a carriage return, i.e., 0x0D 0x0A: & echo test30> c:\test30.txt & echo. & echo verify that c:\test30.txt created by reading contents of file: & echo. & type c:\test30.txt
Date/Time commands executed:
Thu 01/03/2013
12:42 PM
Attempt to open c:\test30.txt to show does not exist:
The system cannot find the file specified.
Create c:\test30.txt with the text "test30" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test30.txt created by reading contents of file:
test30
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 26

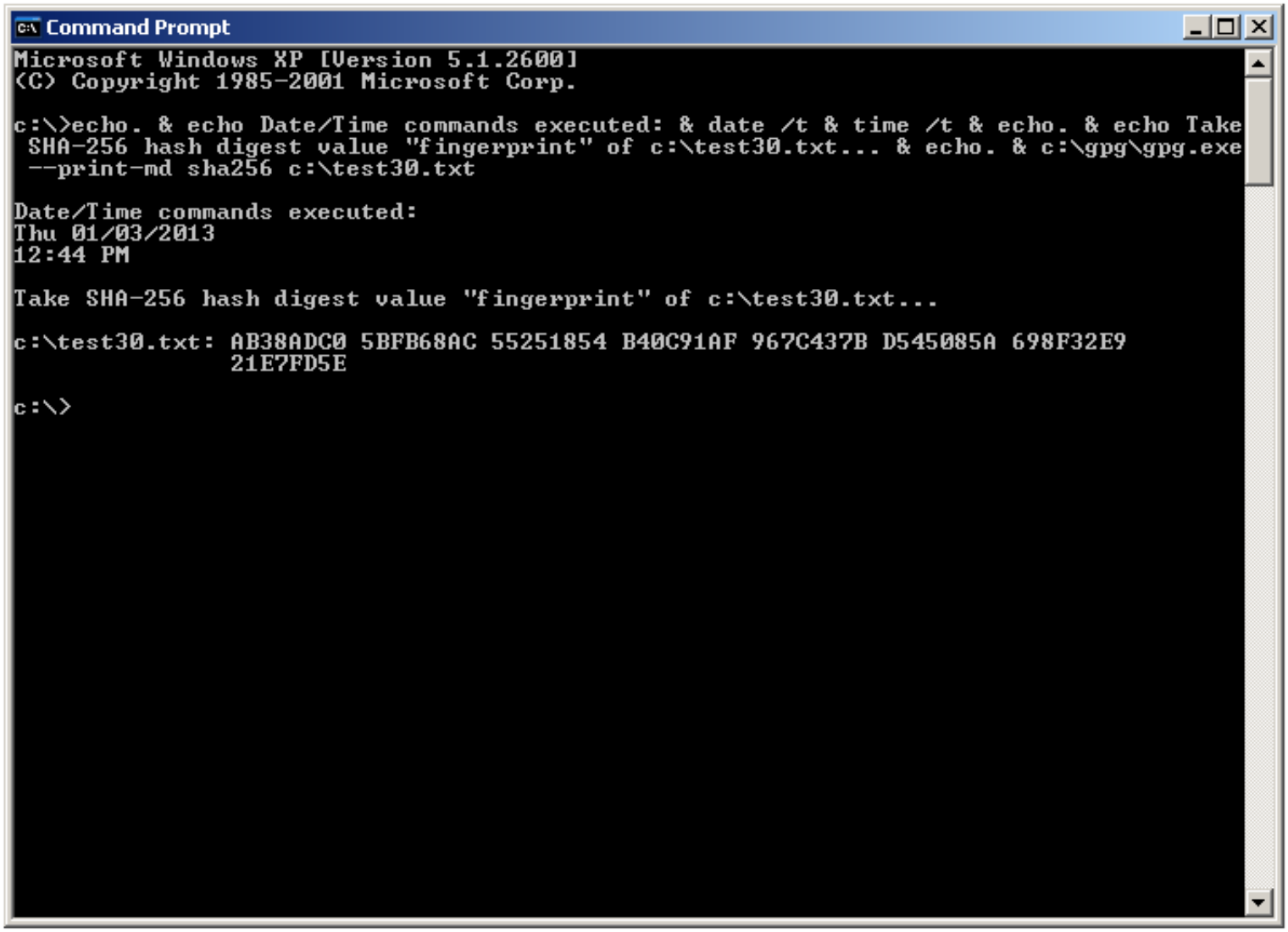

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test30.txt... & echo. & echo File creation date: & dir c:\test30.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test30.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:43 PM
Now read "creation date" and "last modified date" file properties for c:\test30.txt...
File creation date:
Directory of c:\
01/03/2013 12:42 PM 9 test30.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:42 PM 9 test30.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 27

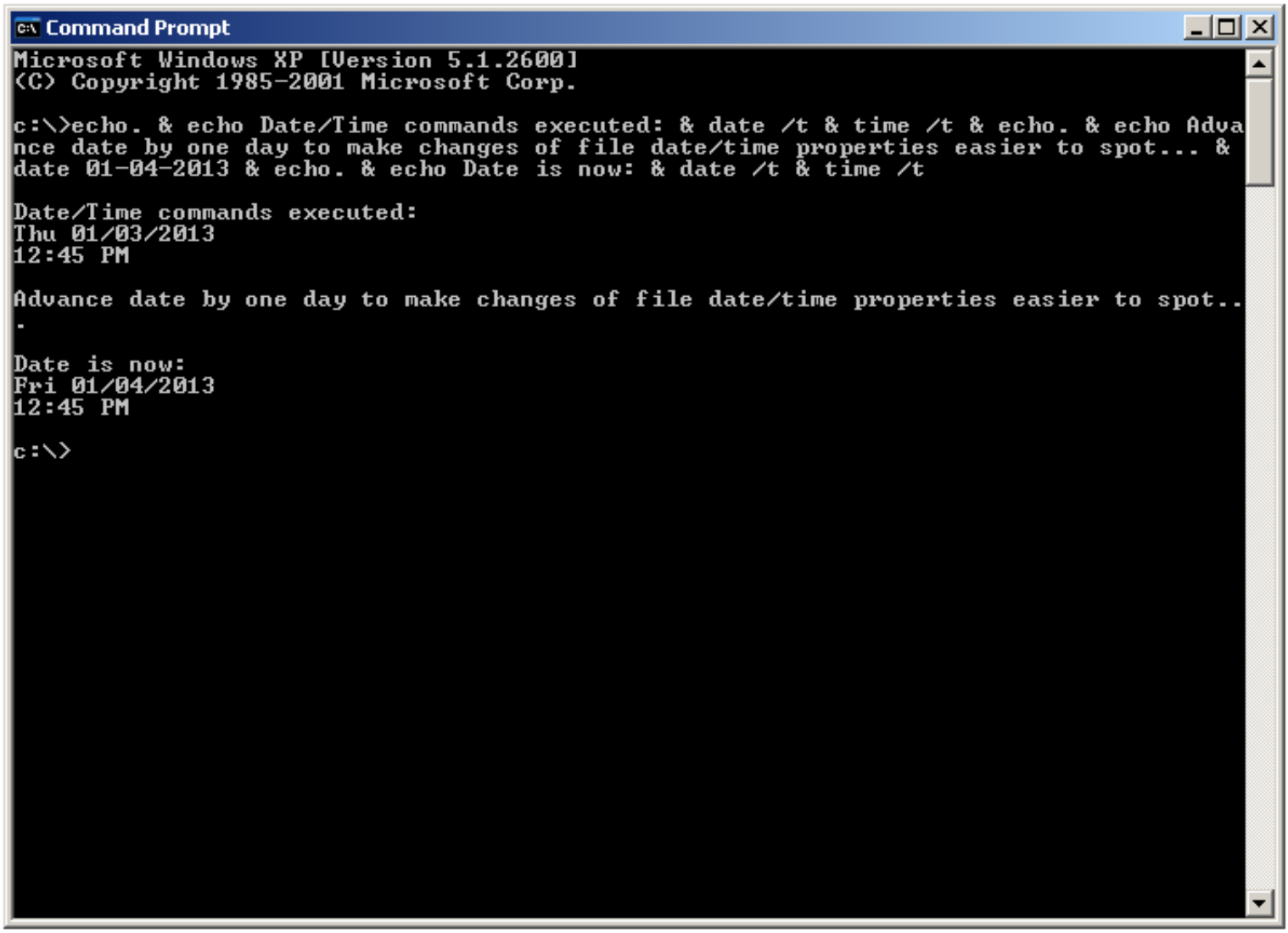

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test30.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test30.txt
Date/Time commands executed:
Thu 01/03/2013
12:44 PM
Take SHA-256 hash digest value "fingerprint" of c:\test30.txt...
c:\test30.txt: AB38ADC0 5BFB68AC 55251854 B40C91AF 967C437B D5450B5A 698F32E9 21E7FD5E
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 28

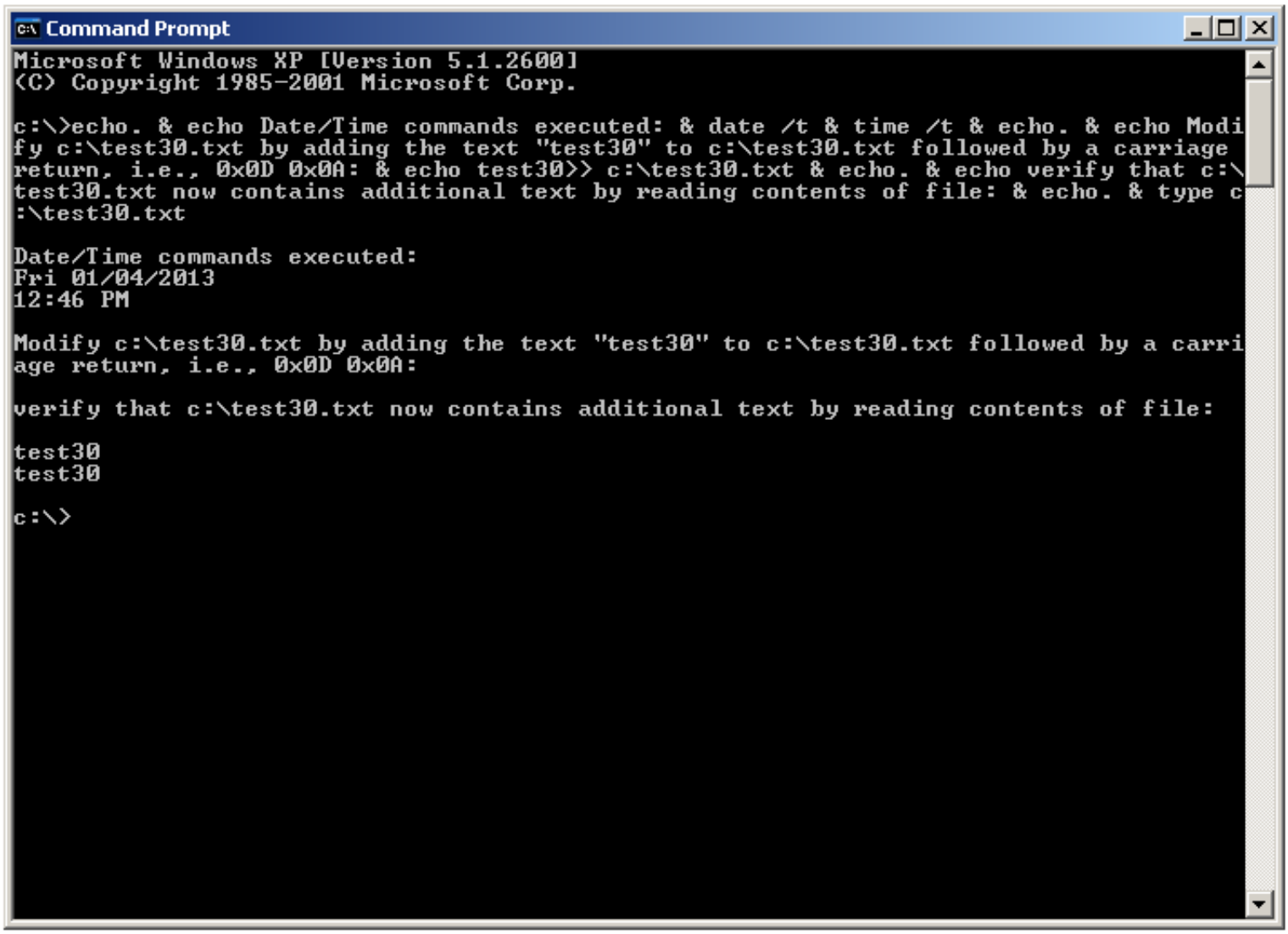

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Adva
nce date by one day to make changes of file date/time properties easier to spot... &
date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
12:45 PM
Advance date by one day to make changes of file date/time properties easier to spot..
.
Date is now:
Fri 01/04/2013
12:45 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 29

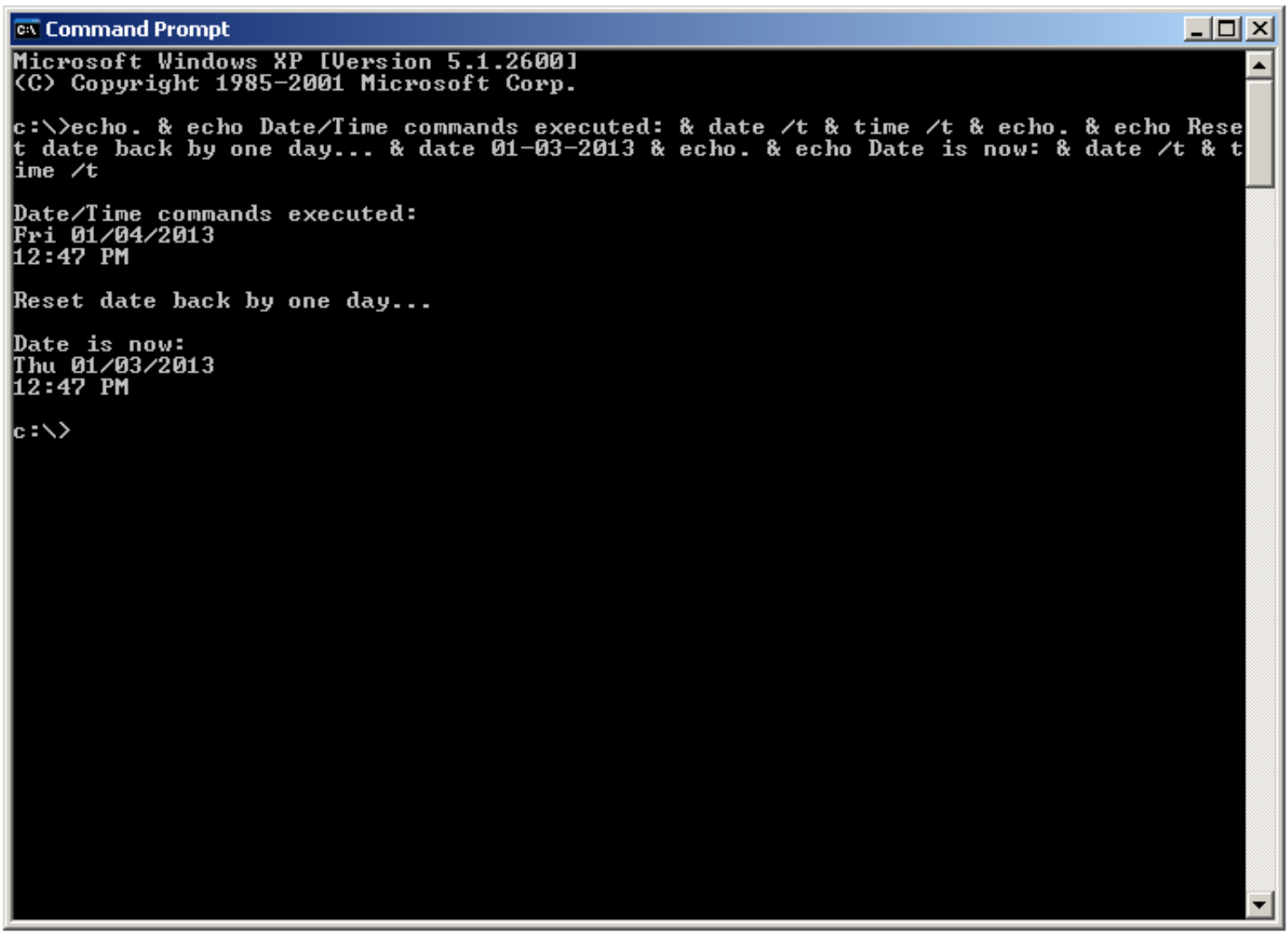

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Modify c:\test30.txt by adding the text "test30" to c:\test30.txt followed by a carriage return, i.e., 0x0D 0x0A: & echo test30>> c:\test30.txt & echo. & echo verify that c:\test30.txt now contains additional text by reading contents of file: & echo. & type c:\test30.txt
Date/Time commands executed:
Fri 01/04/2013
12:46 PM
Modify c:\test30.txt by adding the text "test30" to c:\test30.txt followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test30.txt now contains additional text by reading contents of file:
test30
test30
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 30

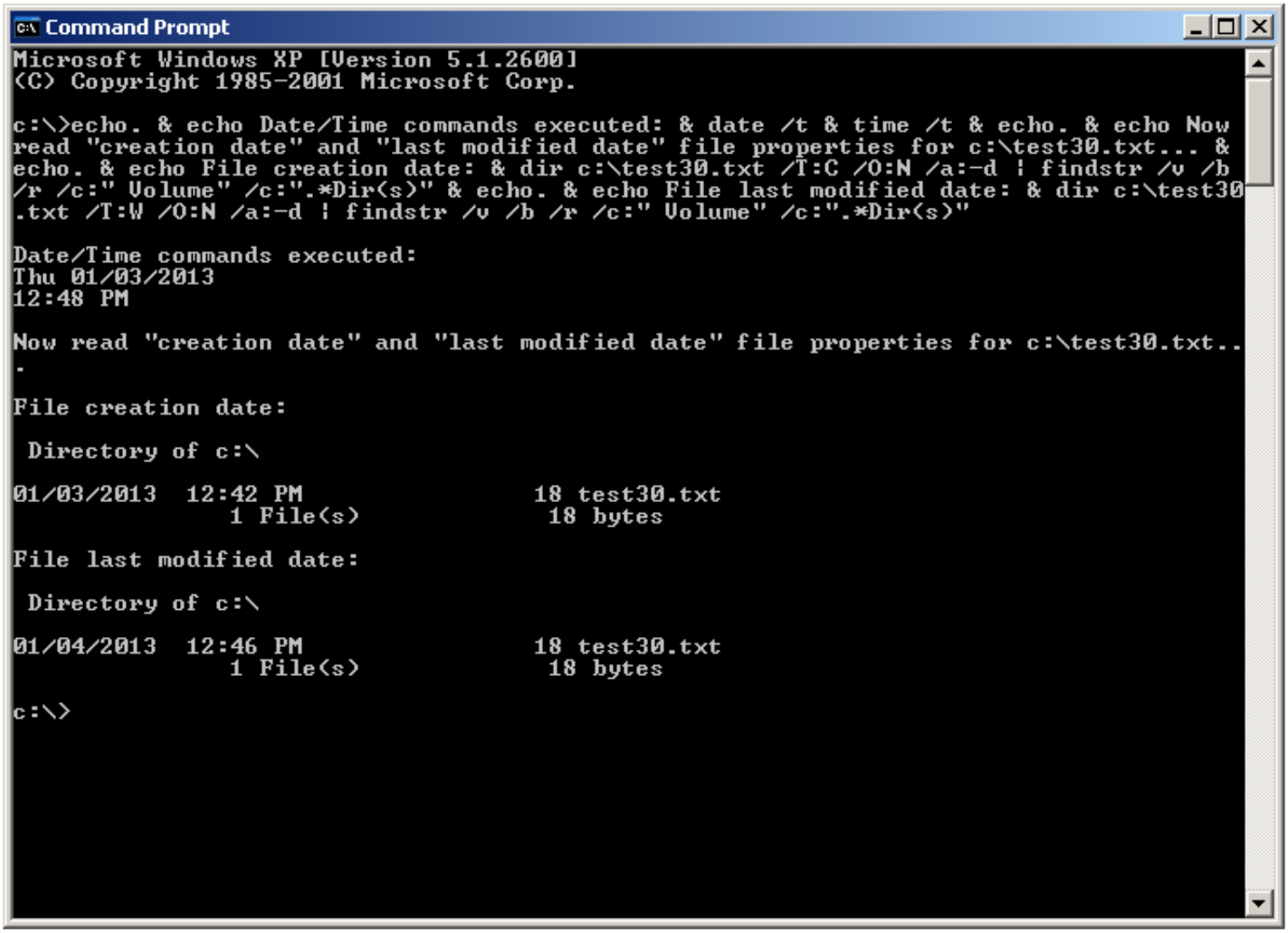

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
12:47 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
12:47 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 31

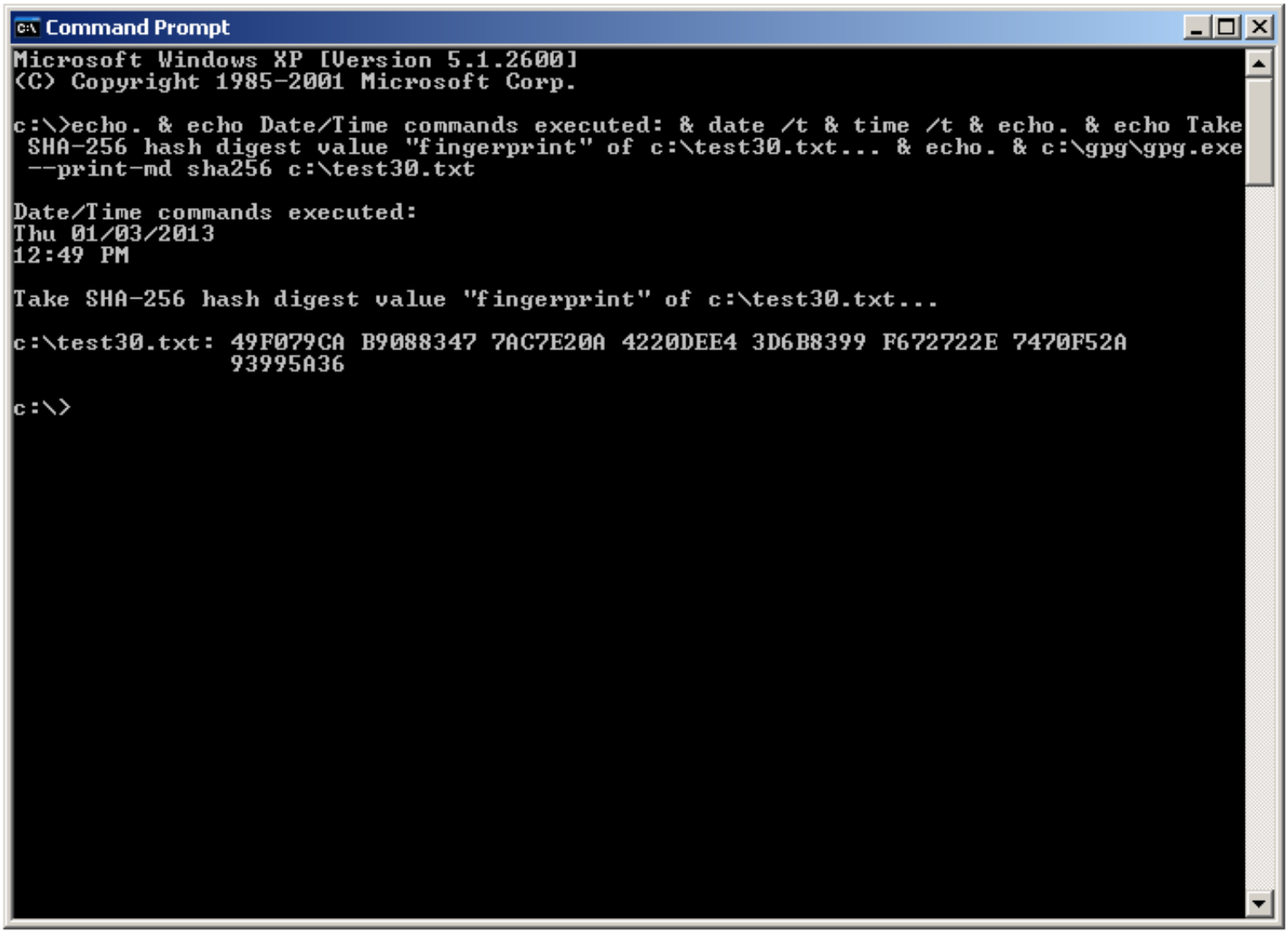

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test30.txt... & echo. & echo File creation date: & dir c:\test30.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test30.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:48 PM
Now read "creation date" and "last modified date" file properties for c:\test30.txt...
File creation date:
Directory of c:\
01/03/2013 12:42 PM 18 test30.txt
1 File(s) 18 bytes
File last modified date:
Directory of c:\
01/04/2013 12:46 PM 18 test30.txt
1 File(s) 18 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 32

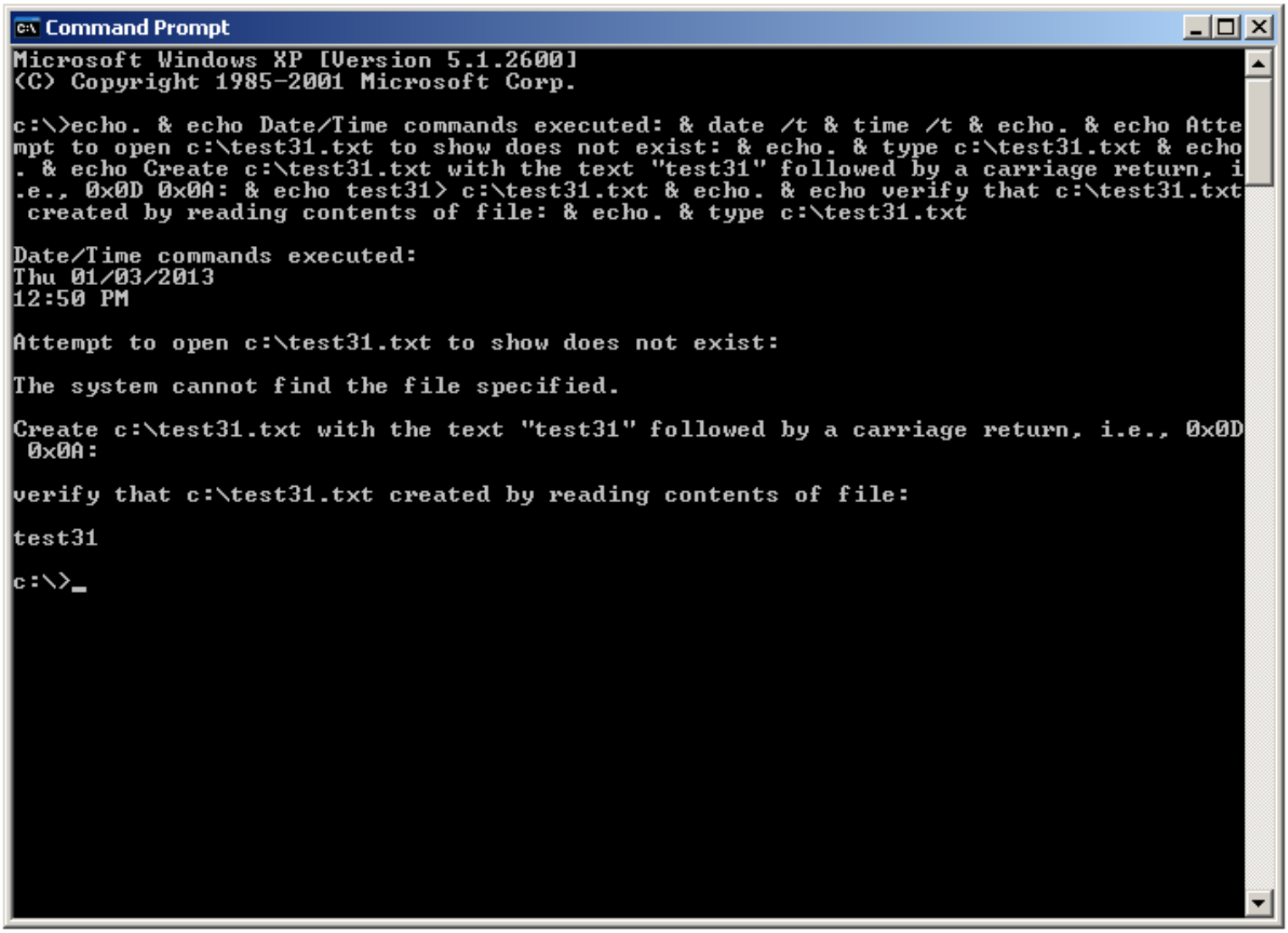

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test30.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test30.txt
Date/Time commands executed:
Thu 01/03/2013
12:49 PM
Take SHA-256 hash digest value "fingerprint" of c:\test30.txt...
c:\test30.txt: 49F079CA B9088347 7AC7E20A 4220DEE4 3D6B8399 F672722E 7470F52A
93995A36
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 33

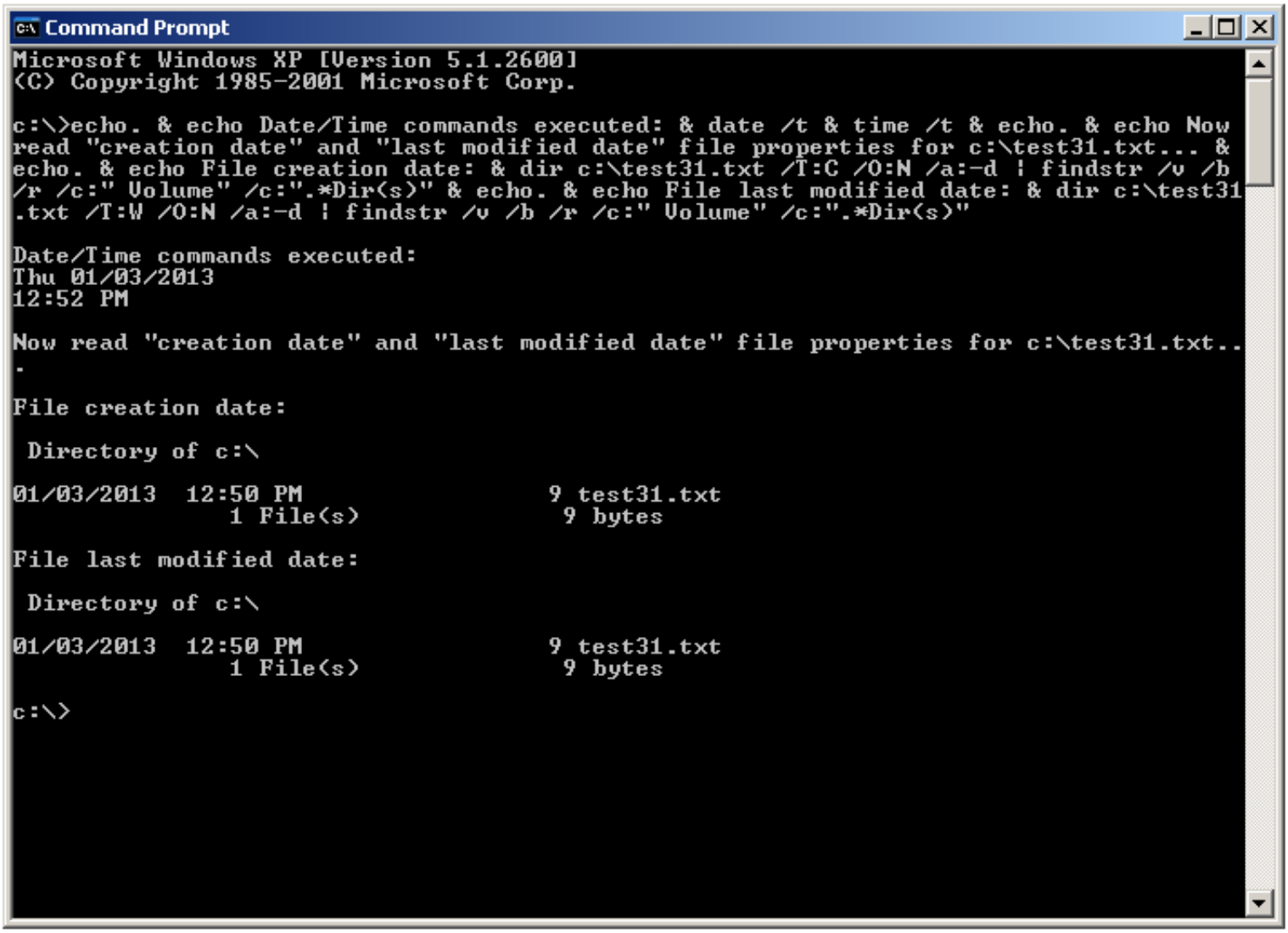

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test31.txt to show does not exist: & echo. & type c:\test31.txt & echo. & echo Create c:\test31.txt with the text "test31" followed by a carriage return, i.e., 0x0D 0x0A: & echo test31> c:\test31.txt & echo. & echo verify that c:\test31.txt created by reading contents of file: & echo. & type c:\test31.txt
Date/Time commands executed:
Thu 01/03/2013
12:50 PM
Attempt to open c:\test31.txt to show does not exist:
The system cannot find the file specified.
Create c:\test31.txt with the text "test31" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test31.txt created by reading contents of file:
test31
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 34

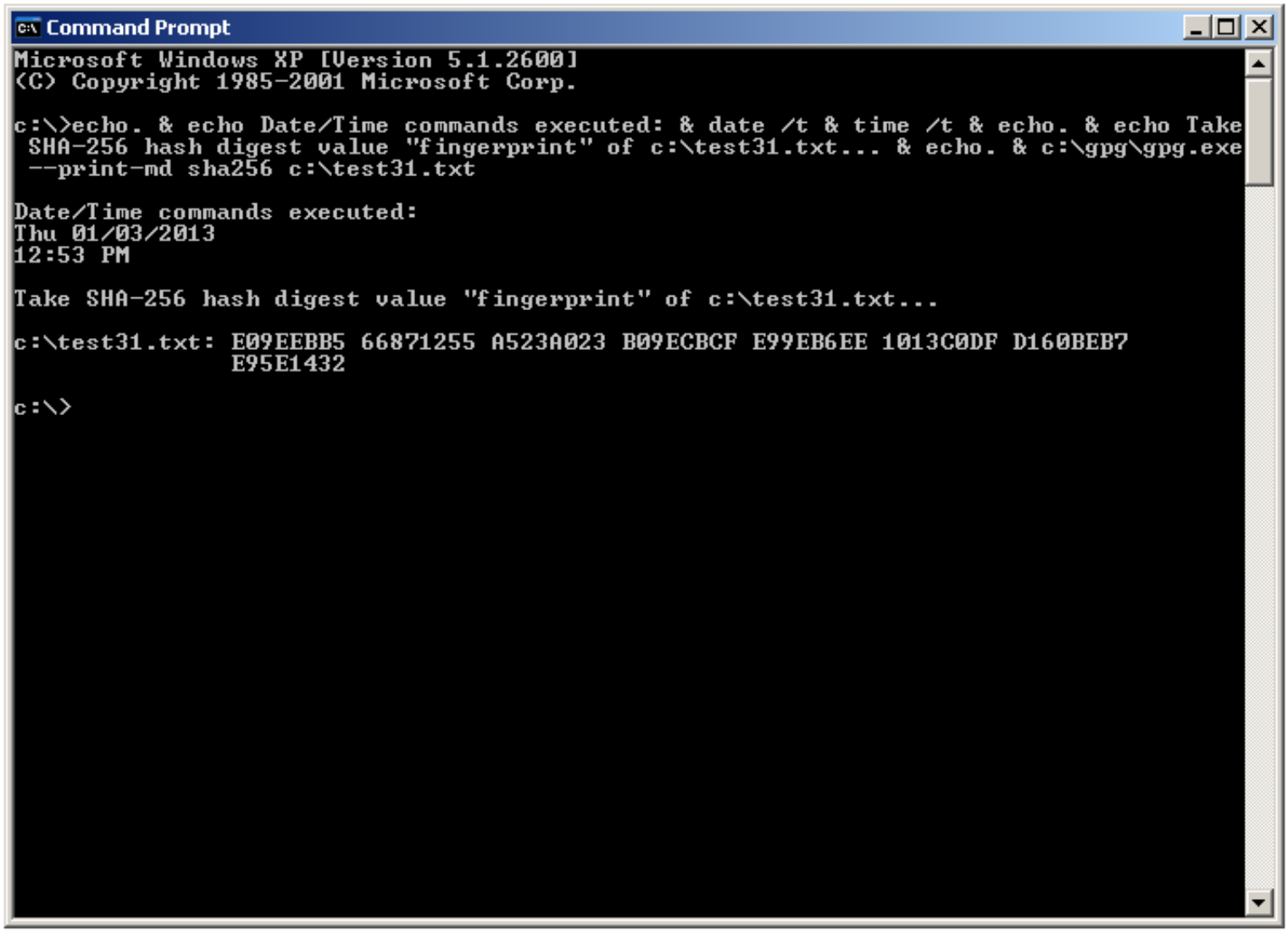

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test31.txt... & echo. & echo File creation date: & dir c:\test31.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test31.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:52 PM
Now read "creation date" and "last modified date" file properties for c:\test31.txt...
File creation date:
Directory of c:\
01/03/2013 12:50 PM 9 test31.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\
01/03/2013 12:50 PM 9 test31.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 35

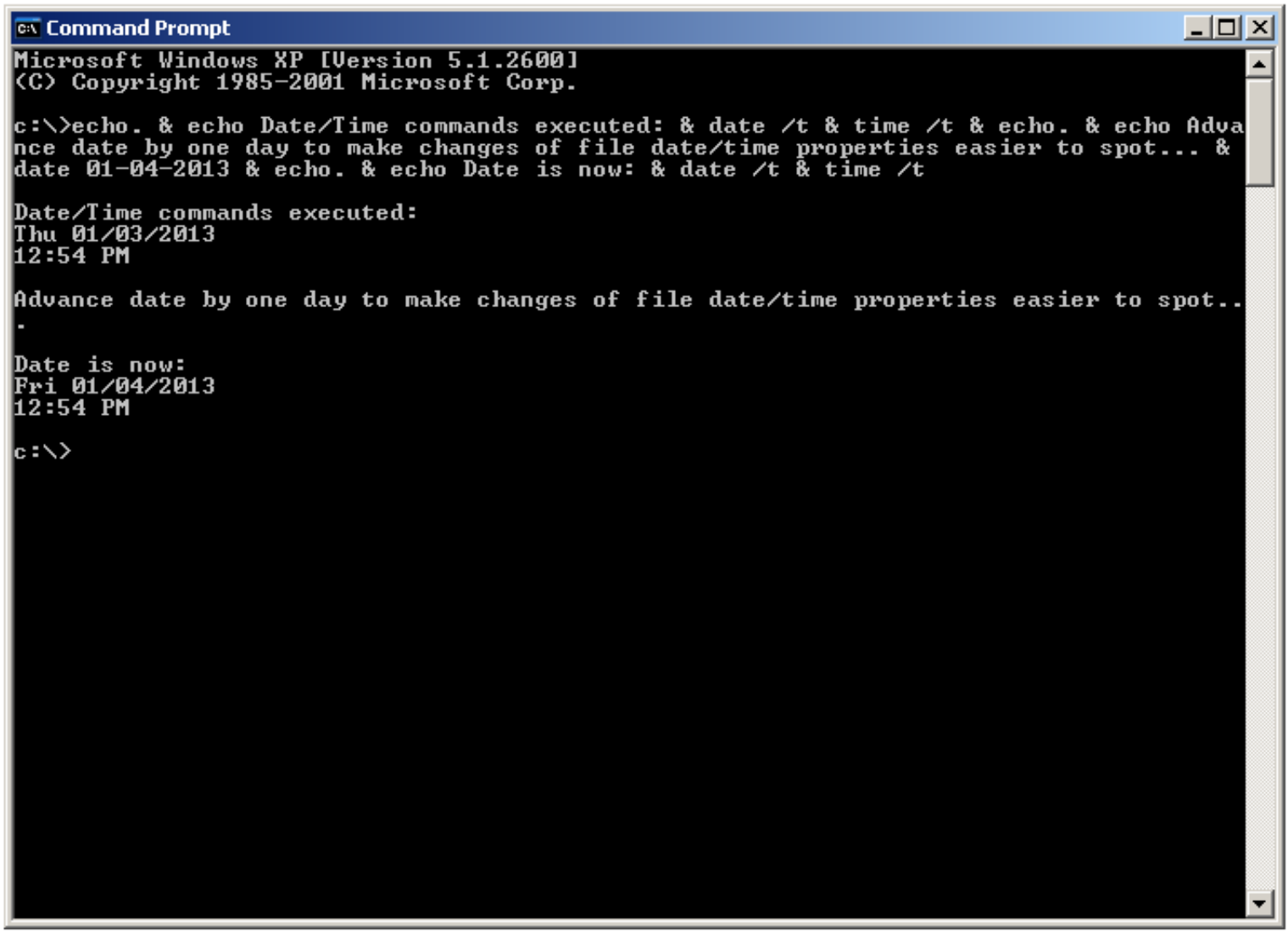

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test31.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test31.txt
Date/Time commands executed:
Thu 01/03/2013
12:53 PM
Take SHA-256 hash digest value "fingerprint" of c:\test31.txt...
c:\test31.txt: E09EEBB5 66871255 A523A023 B09ECBCF E99EB6EE 1013C0DF D160BEB7
E95E1432
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 36

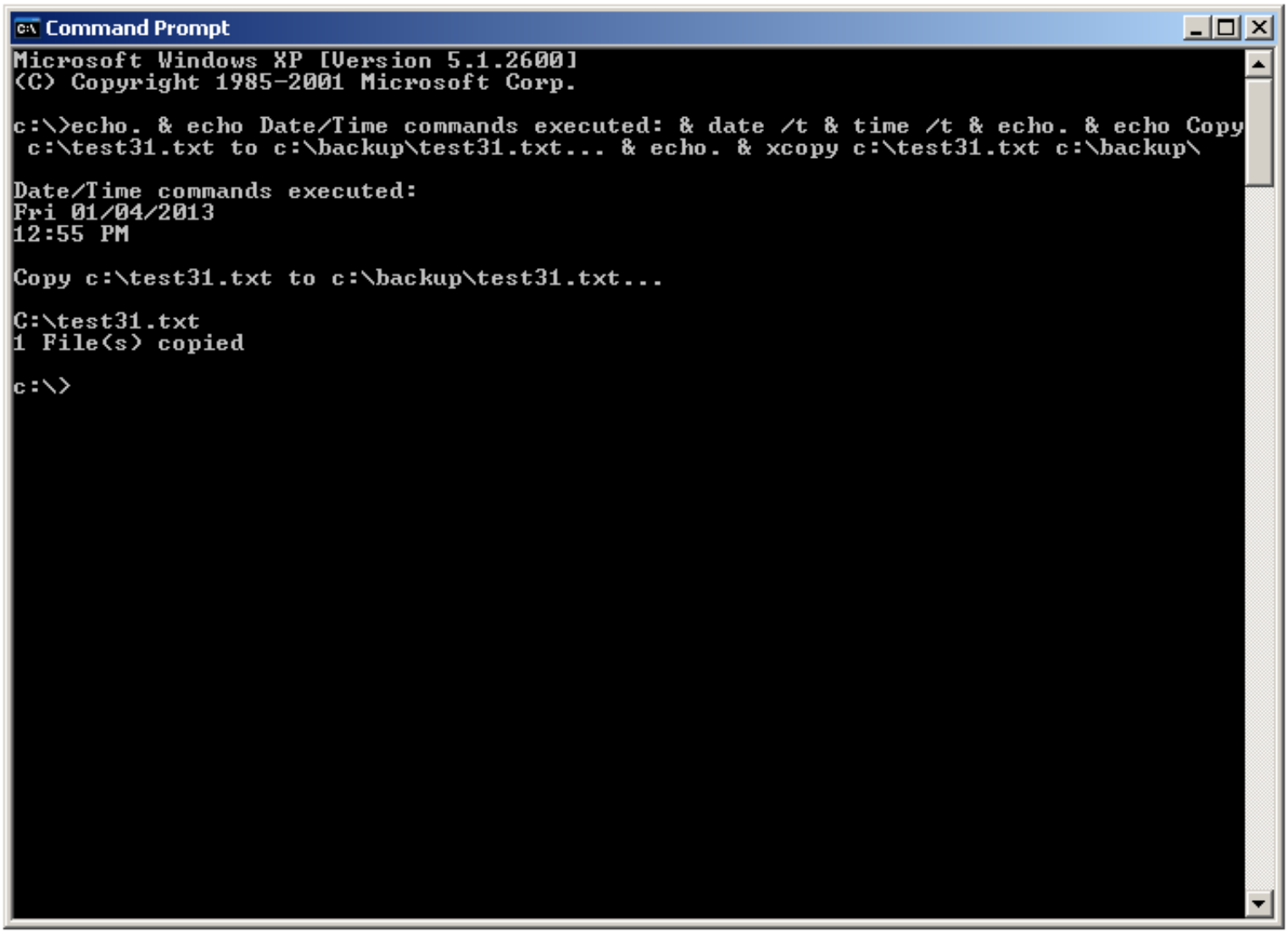

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance date by one day to make changes of file date/time properties easier to spot... & date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
12:54 PM
Advance date by one day to make changes of file date/time properties easier to spot...
Date is now:
Fri 01/04/2013
12:54 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 37

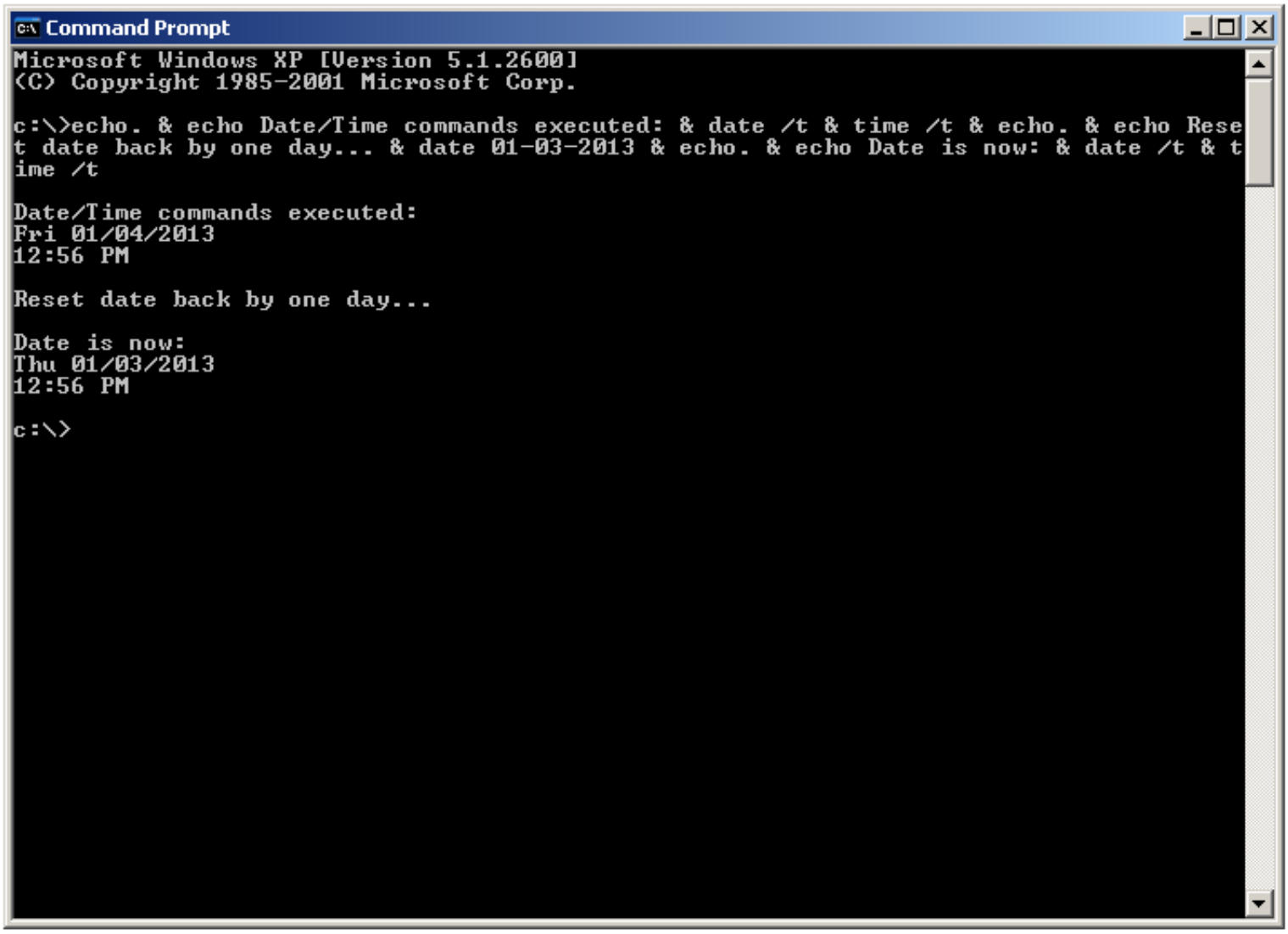

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Copy
c:\test31.txt to c:\backup\test31.txt... & echo. & xcopy c:\test31.txt c:\backup\
Date/Time commands executed:
Fri 01/04/2013
12:55 PM
Copy c:\test31.txt to c:\backup\test31.txt...
C:\test31.txt
1 File(s) copied
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 38

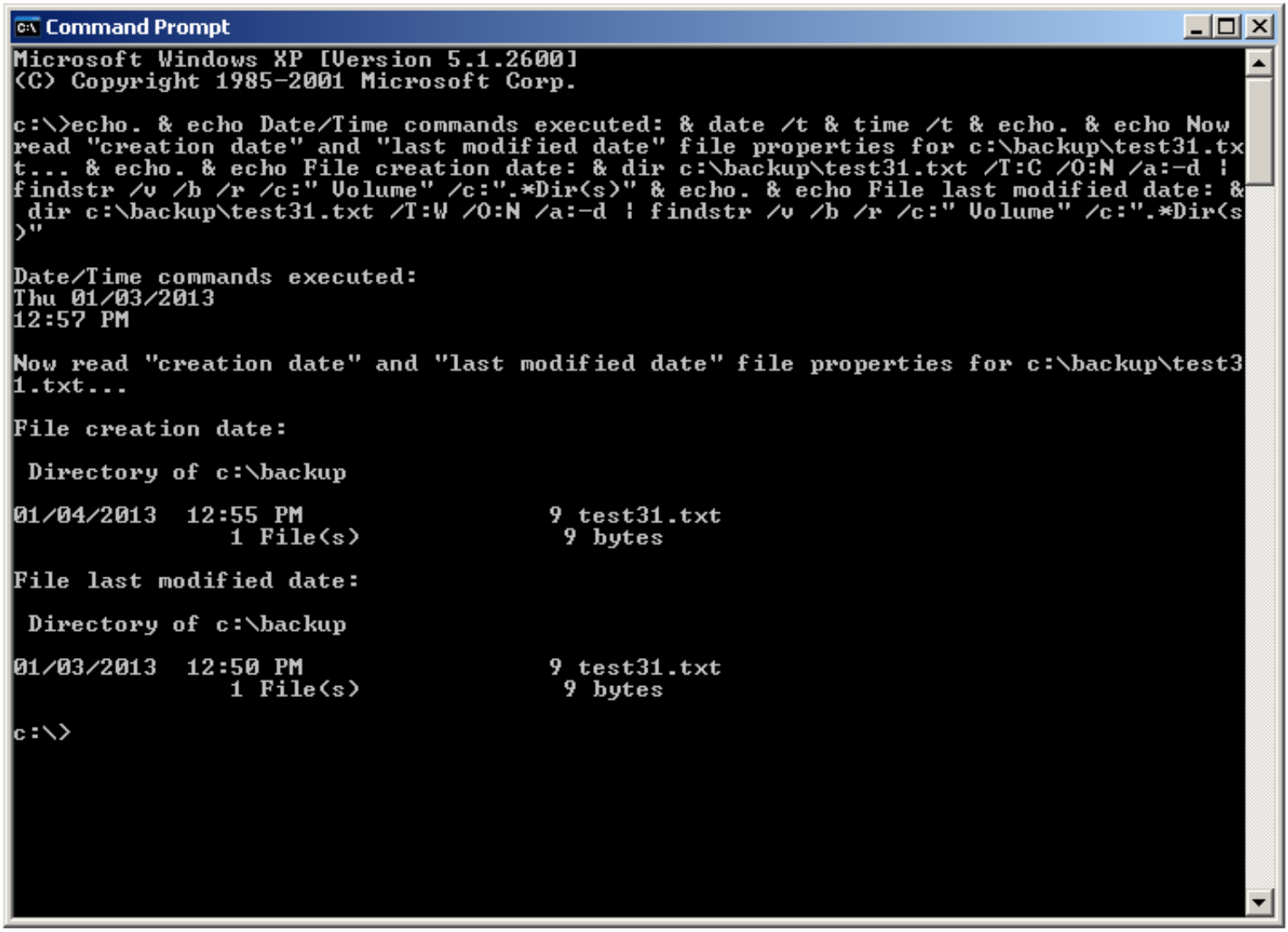

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
12:56 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
12:56 PM
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 39

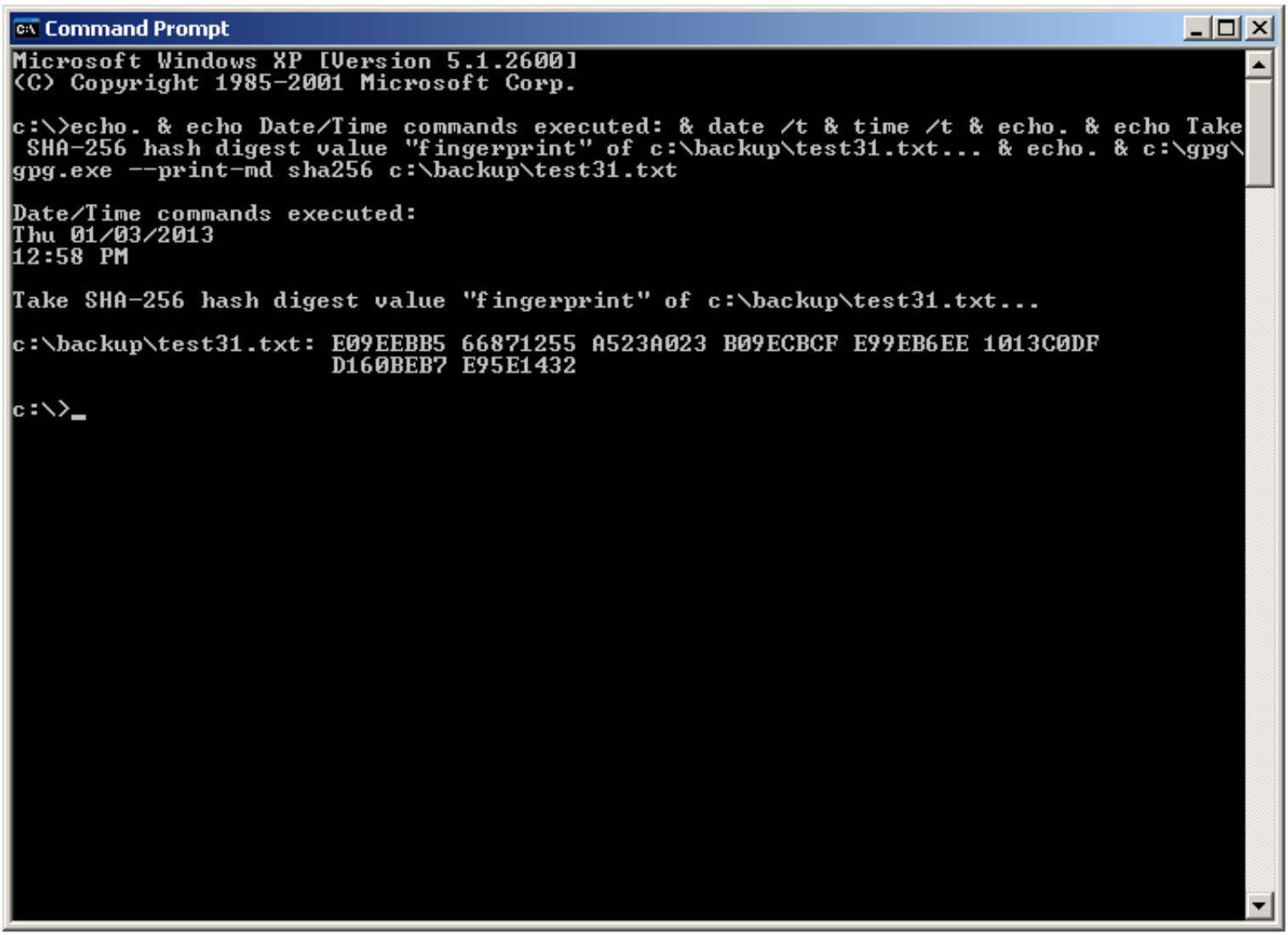

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\backup\test31.txt... & echo. & echo File creation date: & dir c:\backup\test31.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\backup\test31.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
12:57 PM
Now read "creation date" and "last modified date" file properties for c:\backup\test31.txt...
File creation date:
Directory of c:\backup
01/04/2013 12:55 PM 9 test31.txt
1 File(s) 9 bytes
File last modified date:
Directory of c:\backup
01/03/2013 12:50 PM 9 test31.txt
1 File(s) 9 bytes
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 40

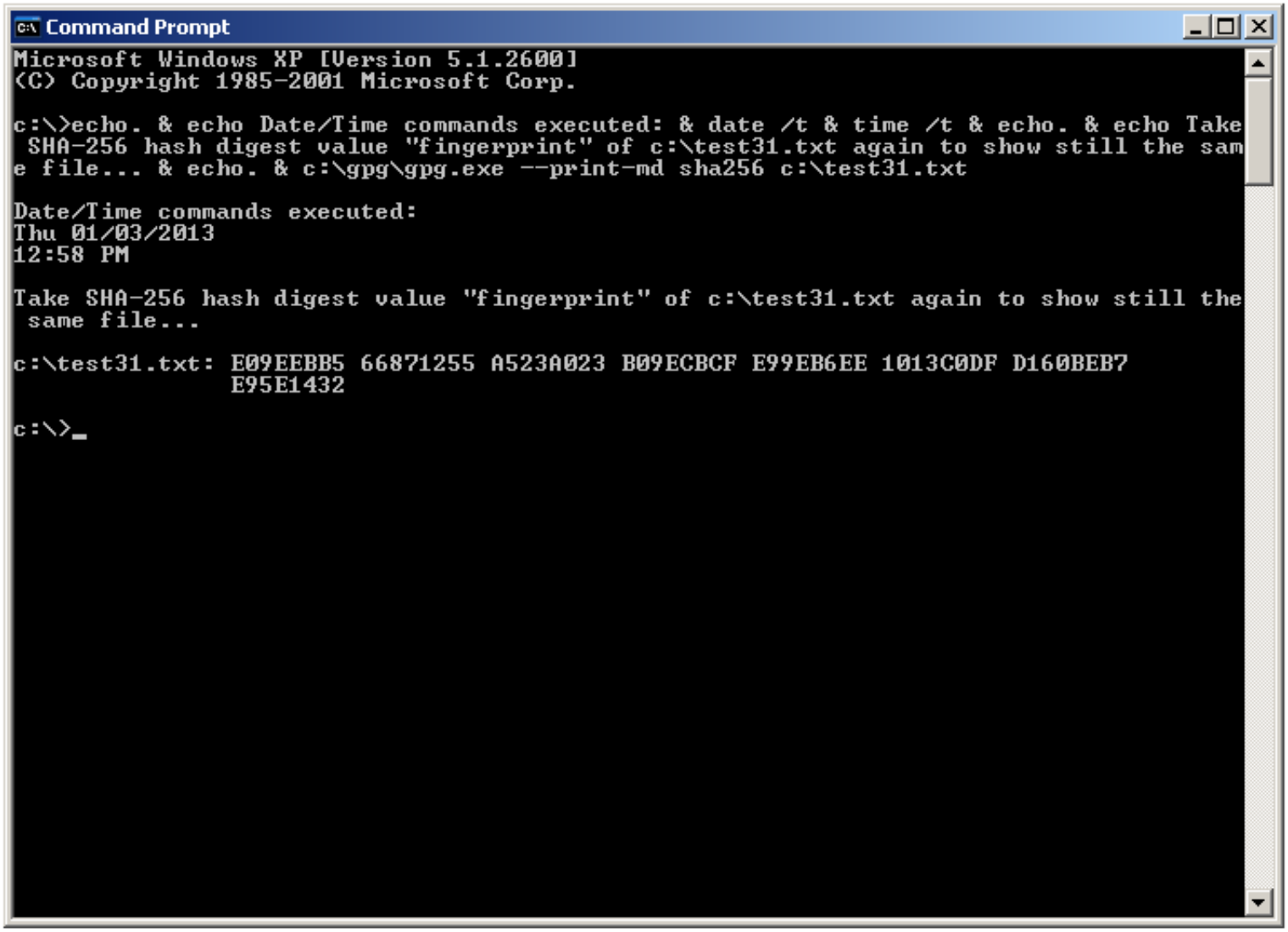

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\backup\test31.txt... & echo. & c:\gpg\
gpg.exe --print-md sha256 c:\backup\test31.txt
Date/Time commands executed:
Thu 01/03/2013
12:58 PM
Take SHA-256 hash digest value "fingerprint" of c:\backup\test31.txt...
c:\backup\test31.txt: E09EEBB5 66871255 A523A023 B09ECBCF E99EB6EE 1013C0DF
D160BEB7 E95E1432
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 41

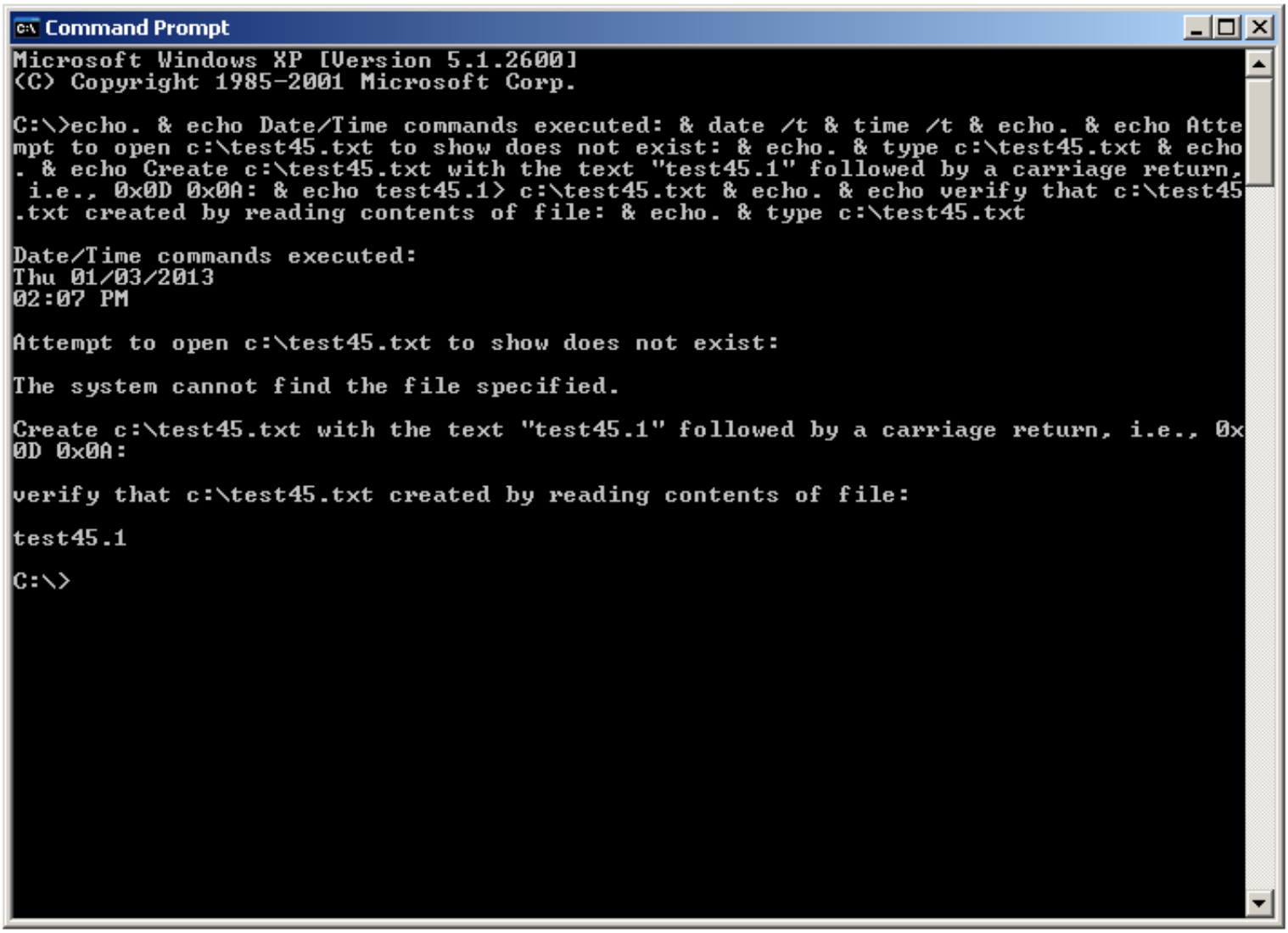

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test31.txt again to show still the same file... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test31.txt
Date/Time commands executed:
Thu 01/03/2013
12:58 PM
Take SHA-256 hash digest value "fingerprint" of c:\test31.txt again to show still the same file...
c:\test31.txt: E09EEBB5 66871255 A523A023 B09ECBCF E99EB6EE 1013C0DF D160BEB7
E95E1432
c:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 42

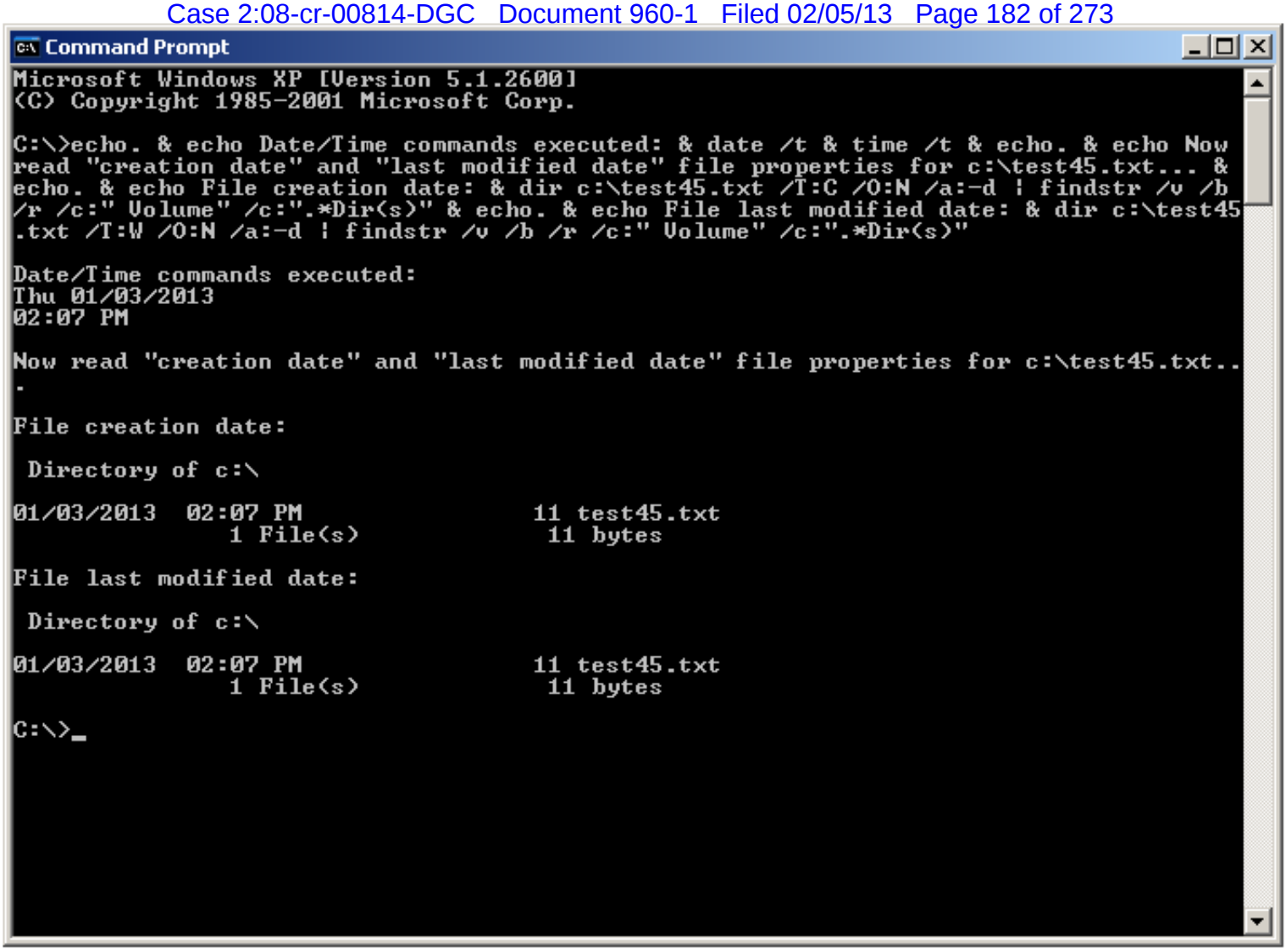

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test45.txt to show does not exist: & echo. & type c:\test45.txt & echo. & echo Create c:\test45.txt with the text "test45.1" followed by a carriage return, i.e., 0x0D 0x0A: & echo test45.1> c:\test45.txt & echo. & echo verify that c:\test45.txt created by reading contents of file: & echo. & type c:\test45.txt
Date/Time commands executed:
Thu 01/03/2013
02:07 PM
Attempt to open c:\test45.txt to show does not exist:
The system cannot find the file specified.
Create c:\test45.txt with the text "test45.1" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test45.txt created by reading contents of file:
test45.1
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 43

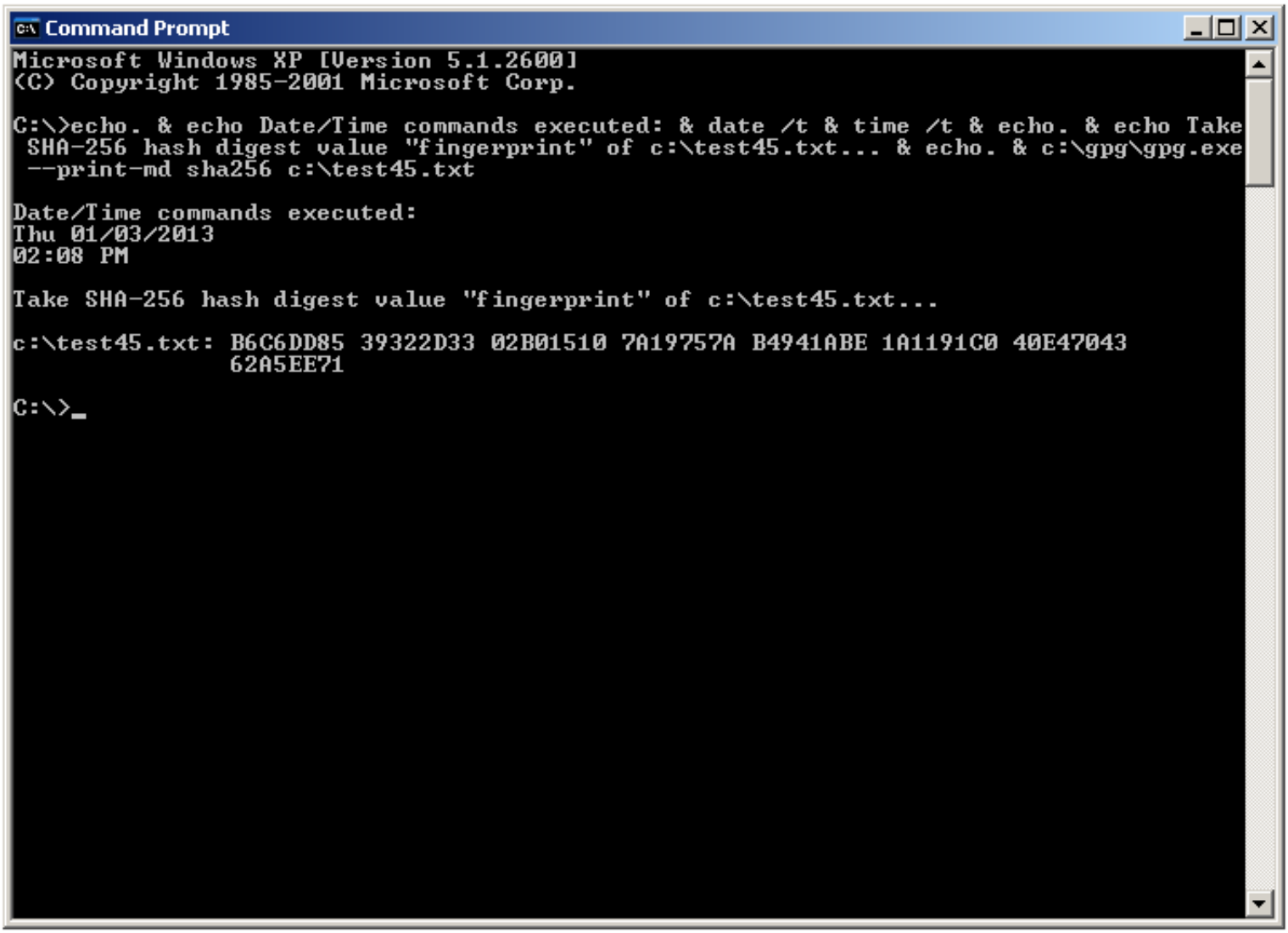

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test45.txt... & echo. & echo File creation date: & dir c:\test45.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test45.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:07 PM
Now read "creation date" and "last modified date" file properties for c:\test45.txt...
File creation date:
Directory of c:\
01/03/2013 02:07 PM 11 test45.txt
1 File(s) 11 bytes
File last modified date:
Directory of c:\
01/03/2013 02:07 PM 11 test45.txt
1 File(s) 11 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 44

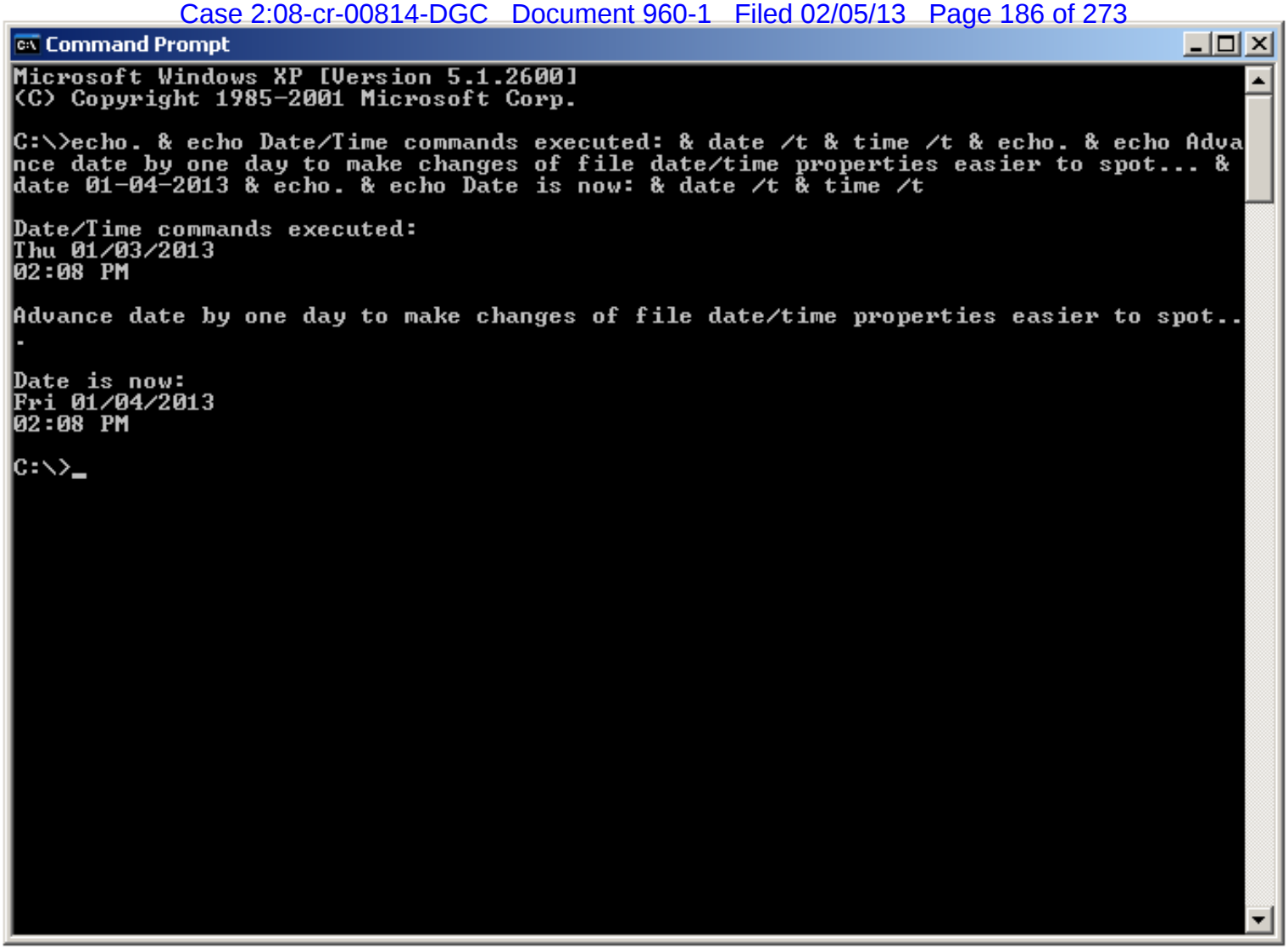

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test45.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test45.txt
Date/Time commands executed:
Thu 01/03/2013
02:08 PM
Take SHA-256 hash digest value "fingerprint" of c:\test45.txt...
c:\test45.txt: B6C6DD85 39322D33 02B01510 7A19757A B4941ABE 1A1191C0 40E47043
62A5EE71
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 45

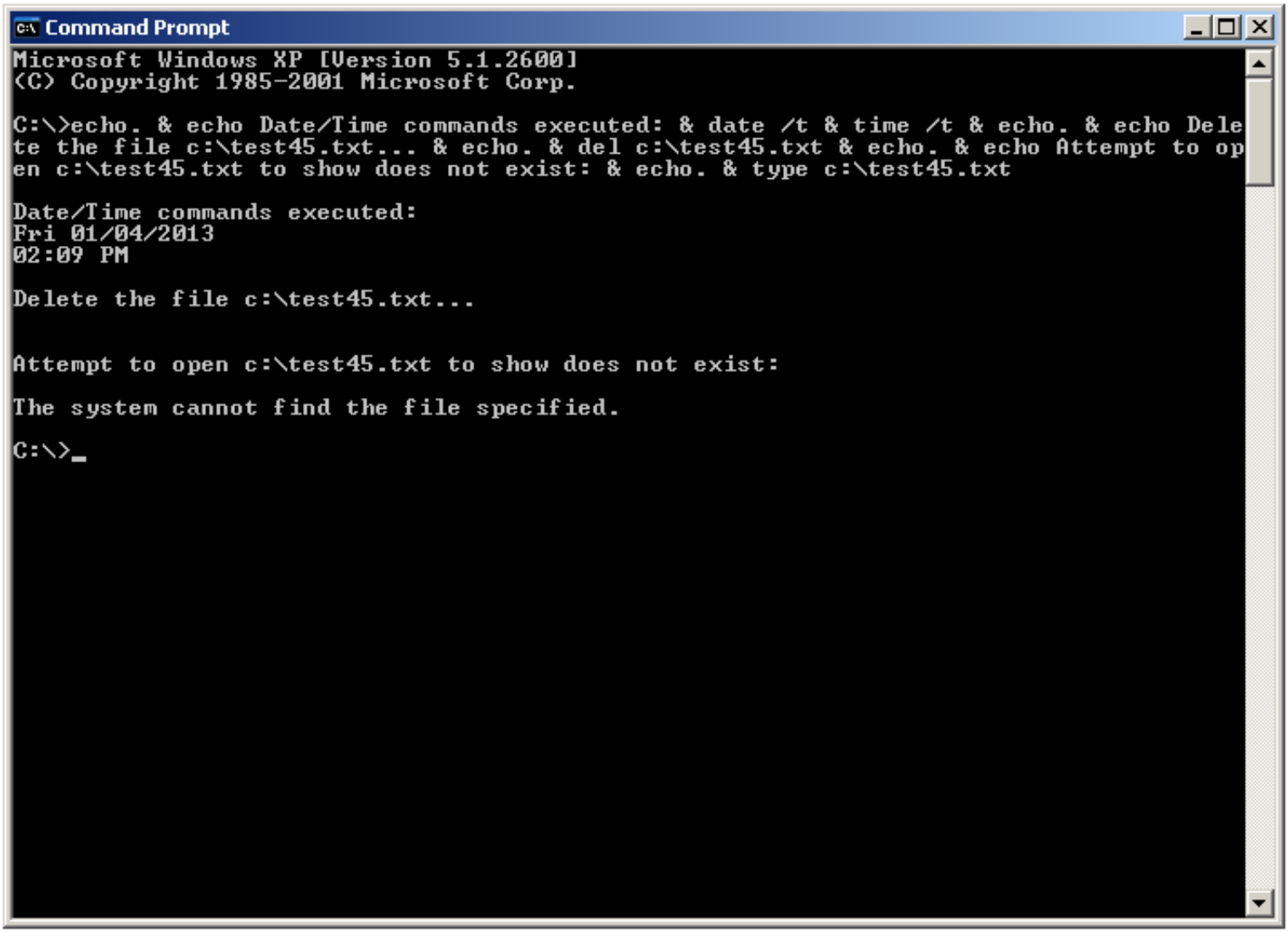

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Adva
nce date by one day to make changes of file date/time properties easier to spot... &
date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
02:08 PM
Advance date by one day to make changes of file date/time properties easier to spot..
.
Date is now:
Fri 01/04/2013
02:08 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 46

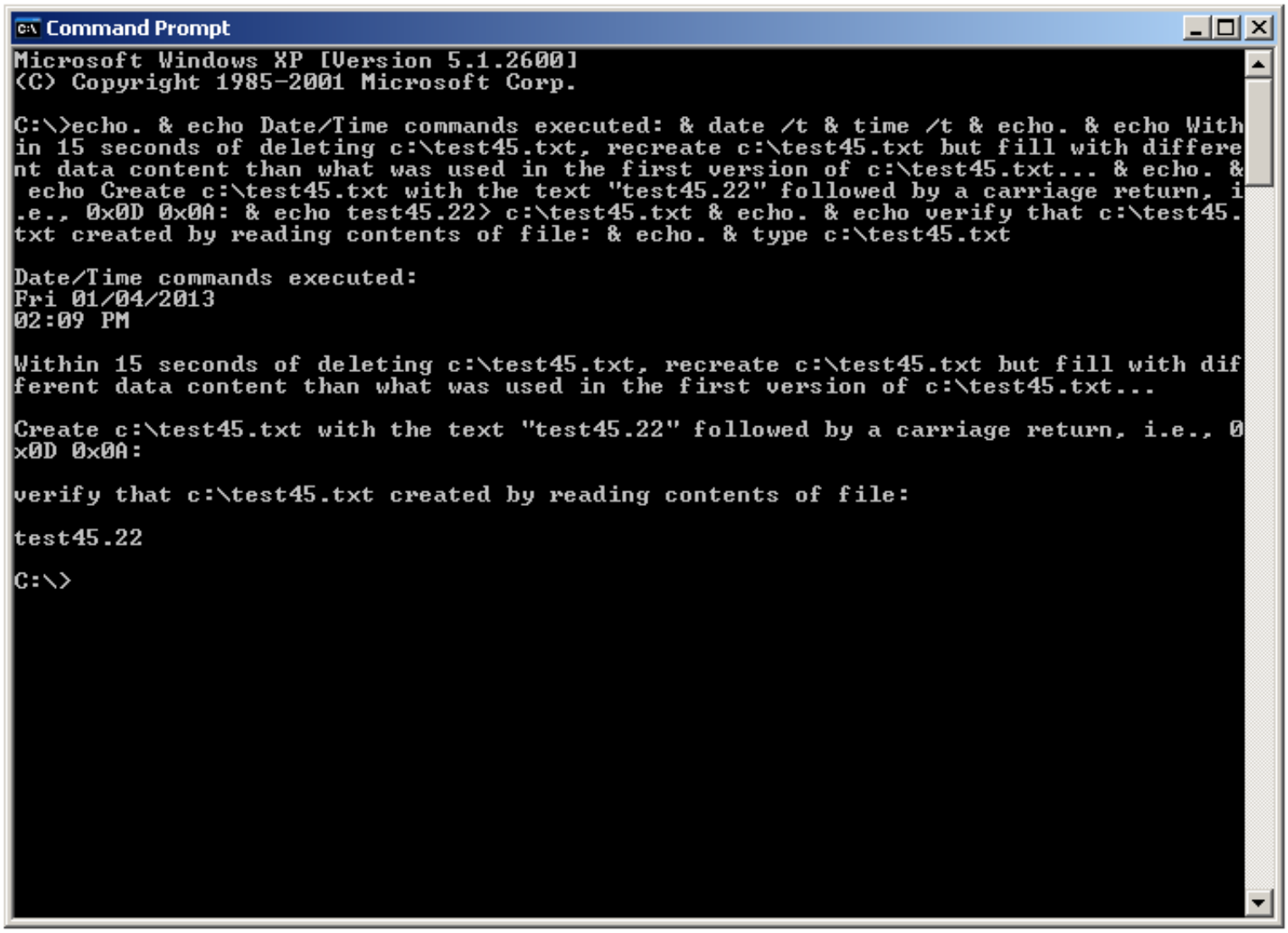

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Dele
te the file c:\test45.txt... & echo. & del c:\test45.txt & echo. & echo Attempt to op
en c:\test45.txt to show does not exist: & echo. & type c:\test45.txt
Date/Time commands executed:
Fri 01/04/2013
02:09 PM
Delete the file c:\test45.txt...
Attempt to open c:\test45.txt to show does not exist:
The system cannot find the file specified.
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 47

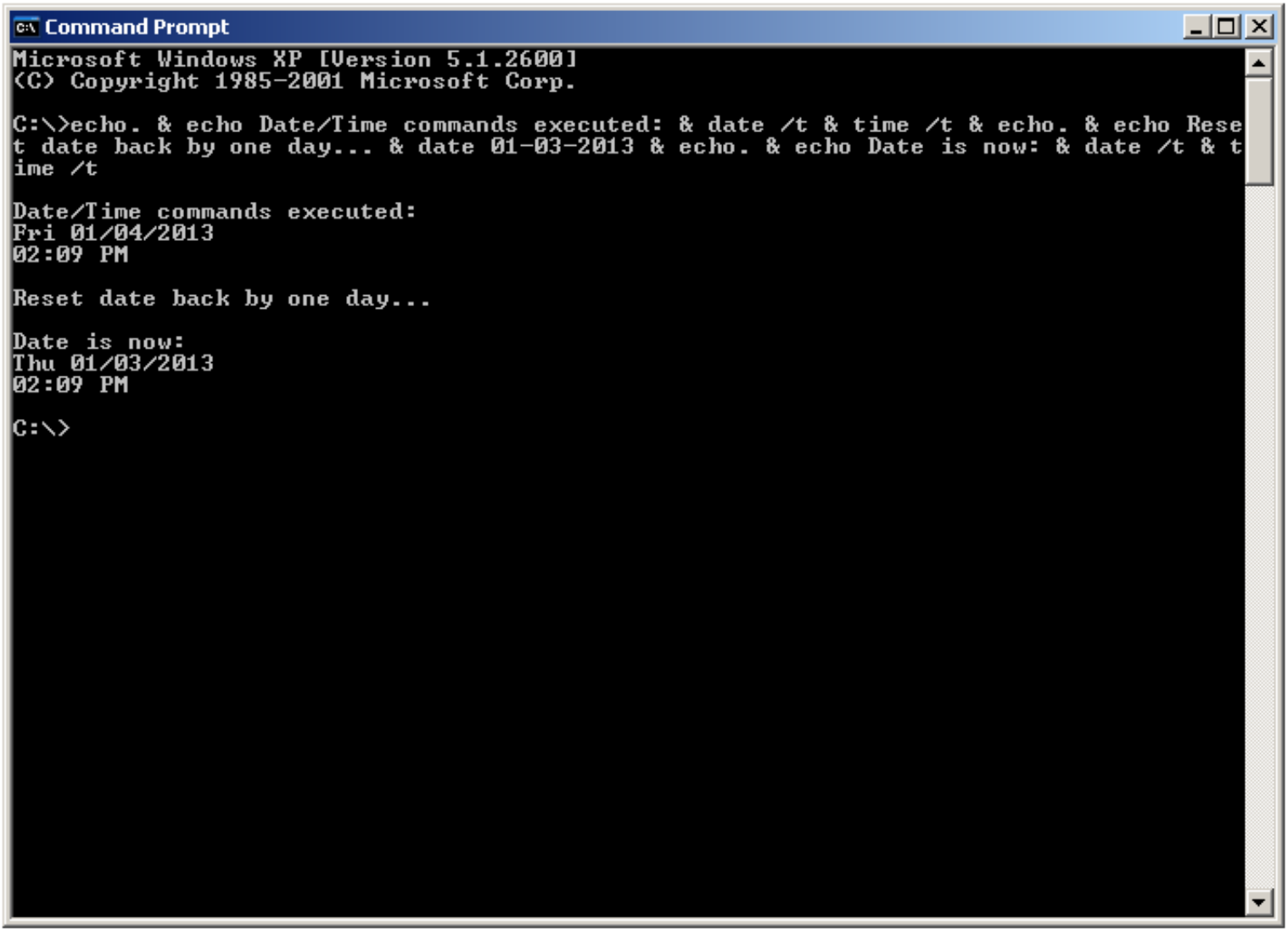

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within 15 seconds of deleting c:\test45.txt, recreate c:\test45.txt but fill with different data content than what was used in the first version of c:\test45.txt... & echo. & echo Create c:\test45.txt with the text "test45.22" followed by a carriage return, i.e., 0x0D 0x0A: & echo test45.22> c:\test45.txt & echo. & echo verify that c:\test45.txt created by reading contents of file: & echo. & type c:\test45.txt
Date/Time commands executed:
Fri 01/04/2013
02:09 PM
Within 15 seconds of deleting c:\test45.txt, recreate c:\test45.txt but fill with different data content than what was used in the first version of c:\test45.txt...
Create c:\test45.txt with the text "test45.22" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test45.txt created by reading contents of file:
test45.22
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 48

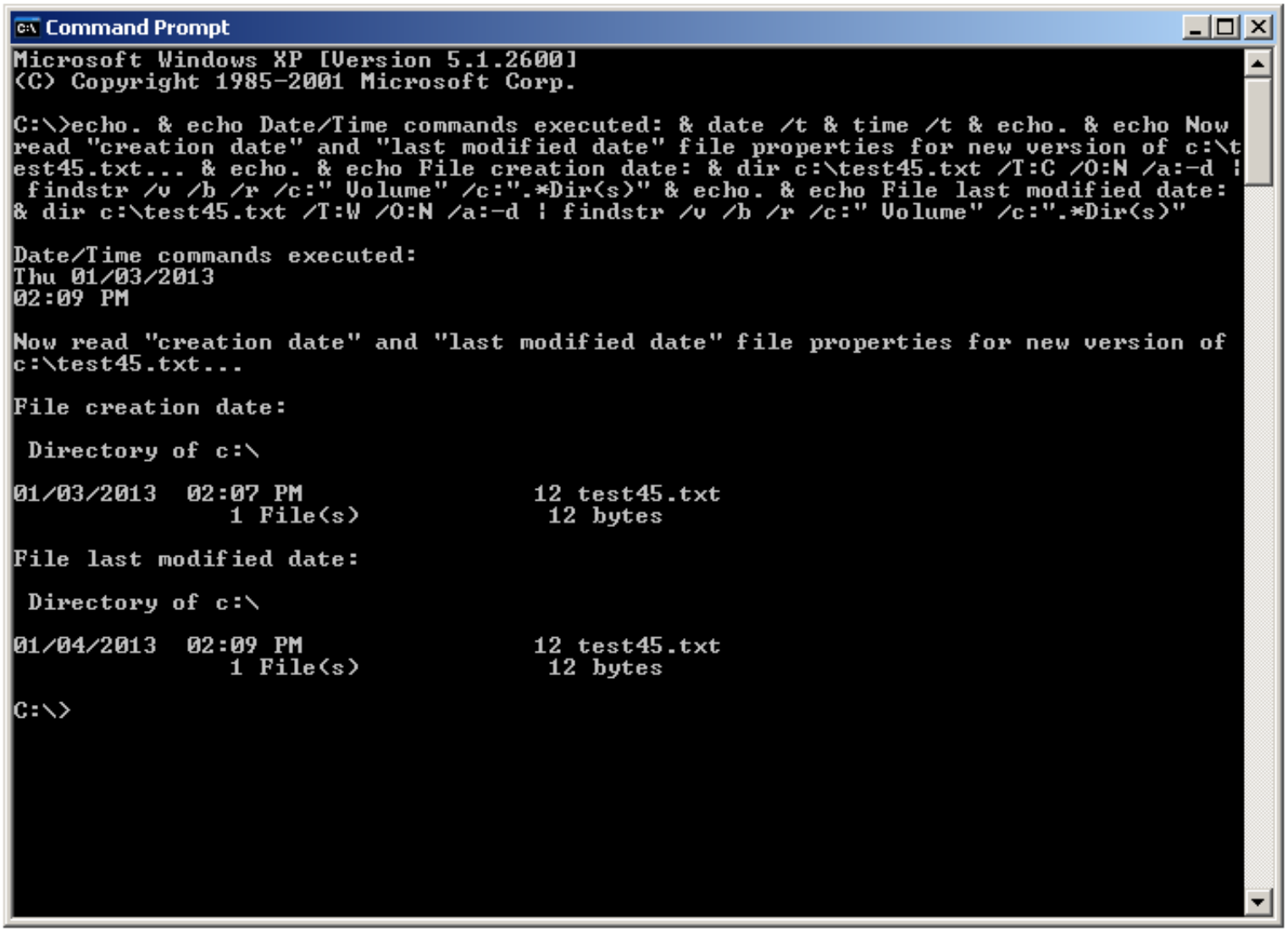

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
02:09 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
02:09 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 49

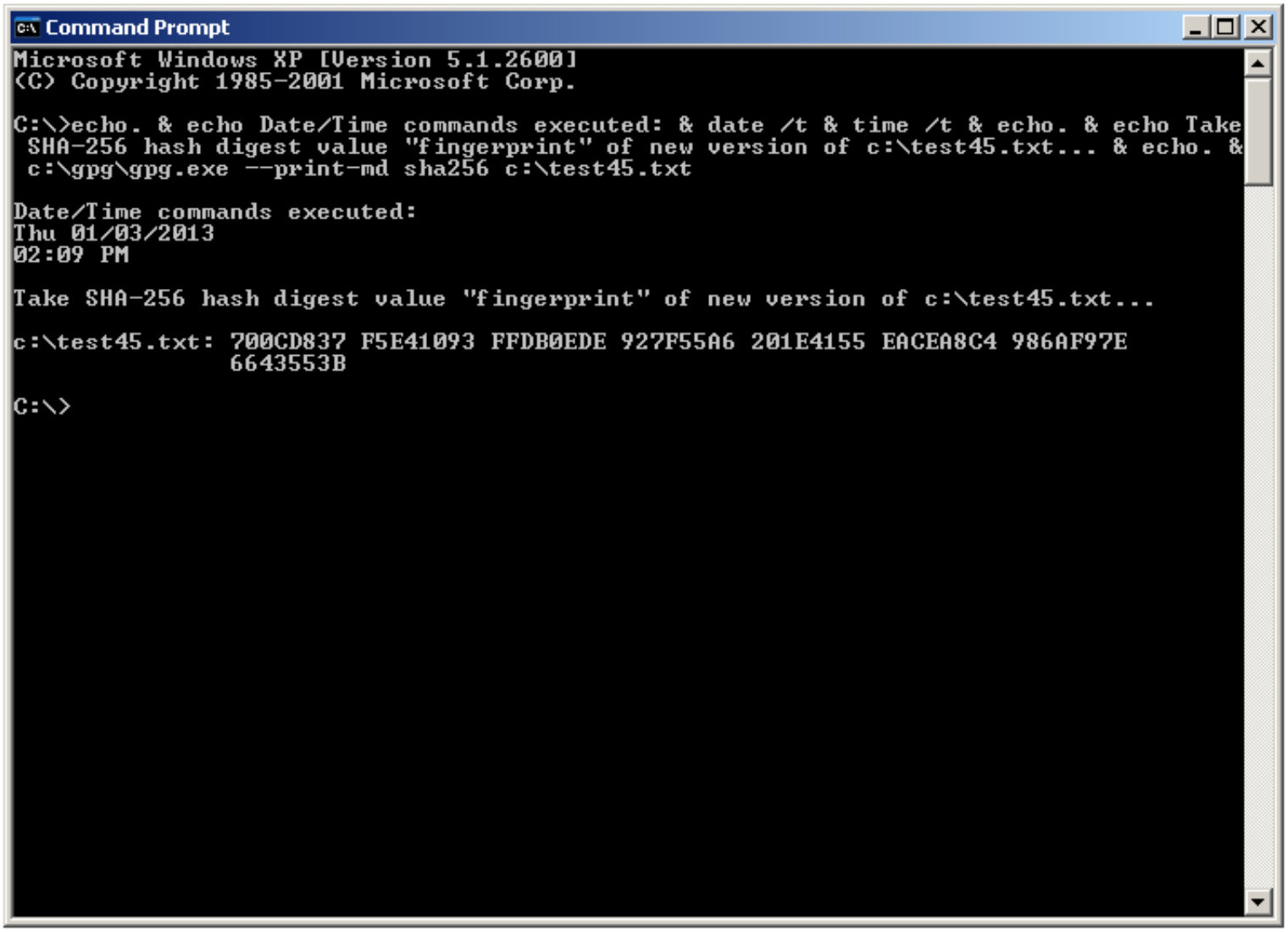

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for new version of c:\test45.txt... & echo. & echo File creation date: & dir c:\test45.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test45.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:09 PM
Now read "creation date" and "last modified date" file properties for new version of c:\test45.txt...
File creation date:
Directory of c:\
01/03/2013 02:07 PM 12 test45.txt
1 File(s) 12 bytes
File last modified date:
Directory of c:\
01/04/2013 02:09 PM 12 test45.txt
1 File(s) 12 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 50

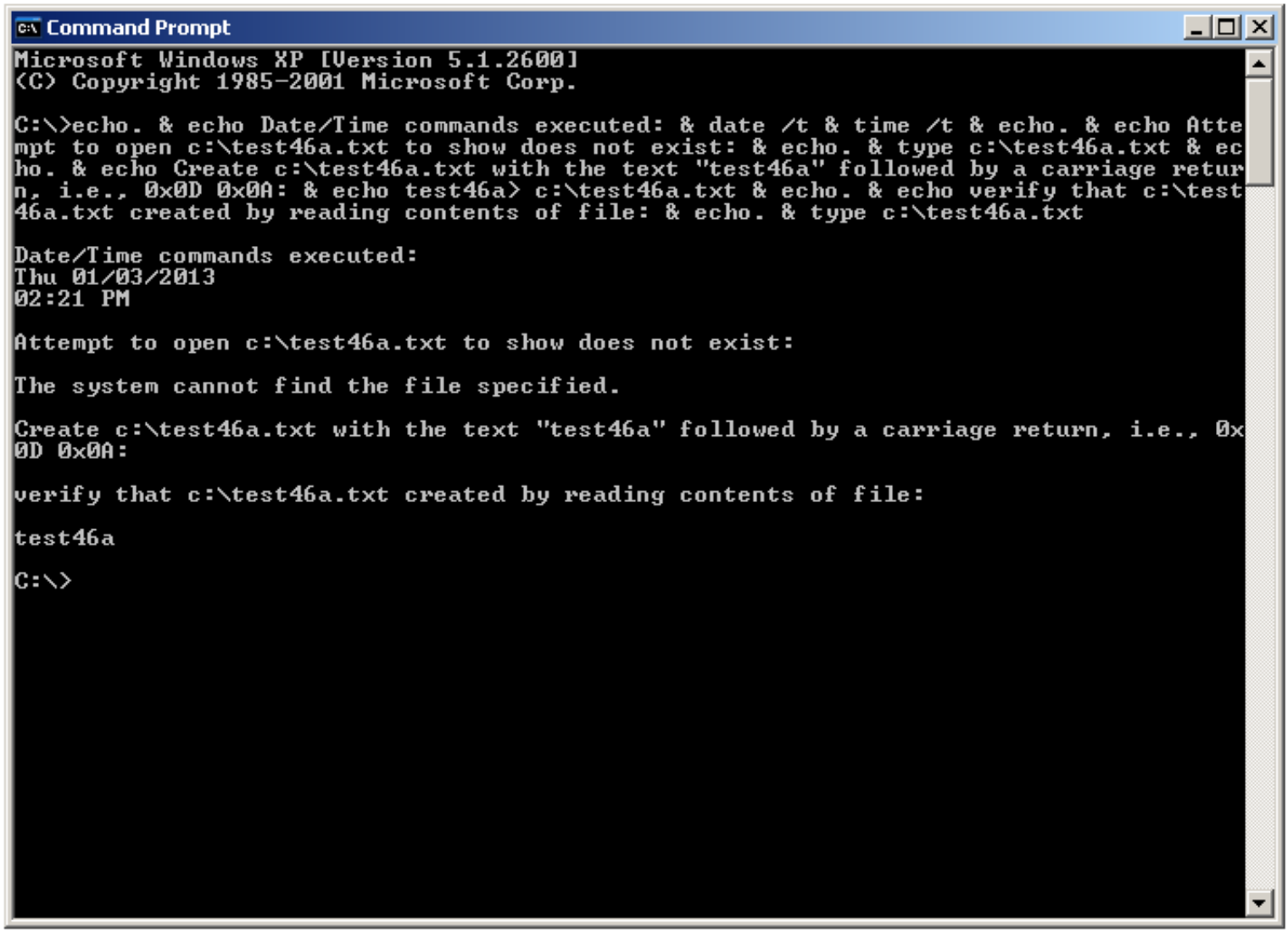

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of new version of c:\test45.txt... & echo. &
c:\gpg\gpg.exe --print-md sha256 c:\test45.txt
Date/Time commands executed:
Thu 01/03/2013
02:09 PM
Take SHA-256 hash digest value "fingerprint" of new version of c:\test45.txt...
c:\test45.txt: 700CD837 F5E41093 FFDB0EDE 927F55A6 201E4155 EACEA8C4 986AF97E
6643553B
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 51

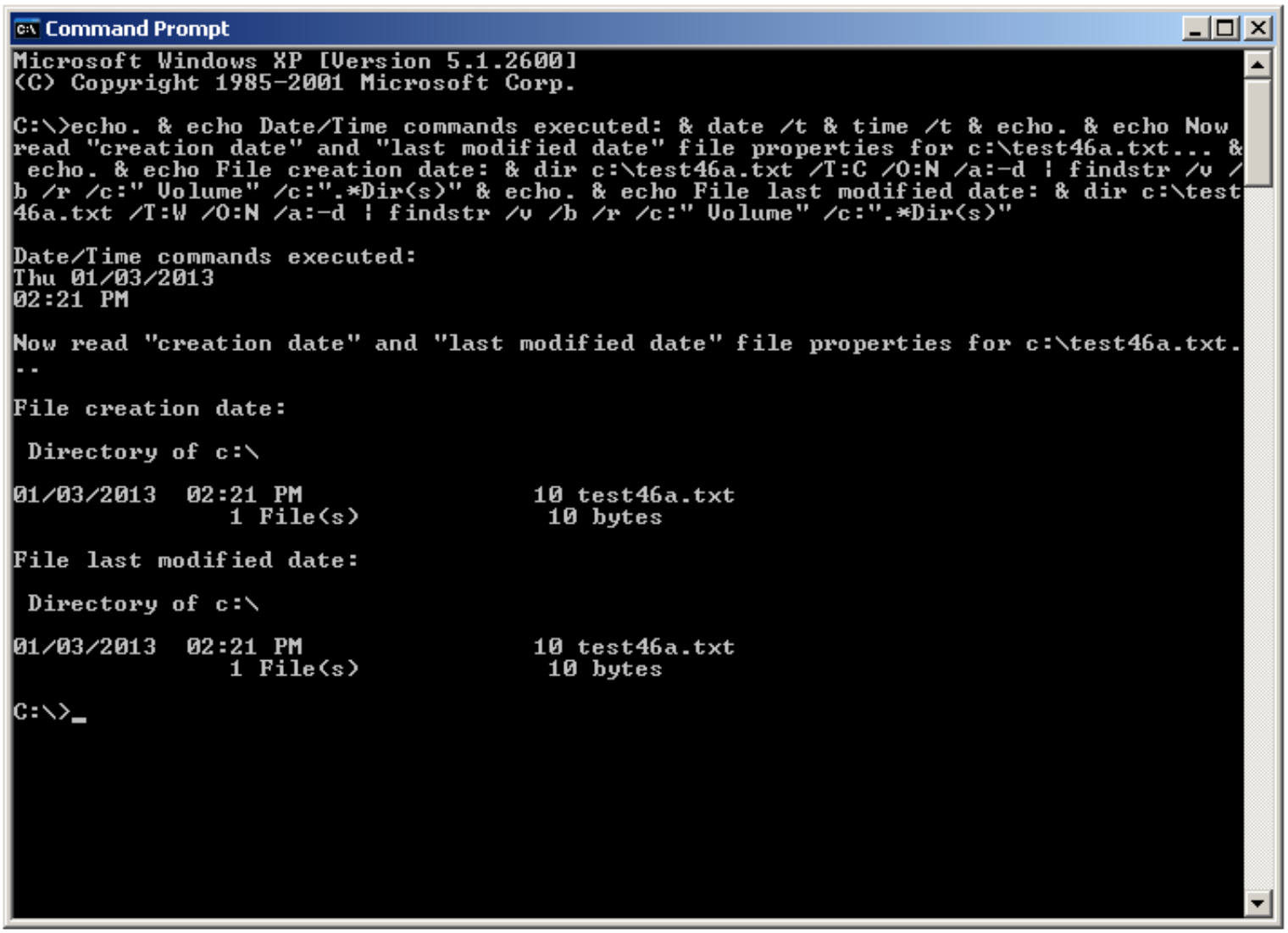

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test46a.txt to show does not exist: & echo. & type c:\test46a.txt & echo. & echo Create c:\test46a.txt with the text "test46a" followed by a carriage return, i.e., 0x0D 0x0A: & echo test46a> c:\test46a.txt & echo. & echo verify that c:\test46a.txt created by reading contents of file: & echo. & type c:\test46a.txt
Date/Time commands executed:
Thu 01/03/2013
02:21 PM
Attempt to open c:\test46a.txt to show does not exist:
The system cannot find the file specified.
Create c:\test46a.txt with the text "test46a" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test46a.txt created by reading contents of file:
test46a
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 52

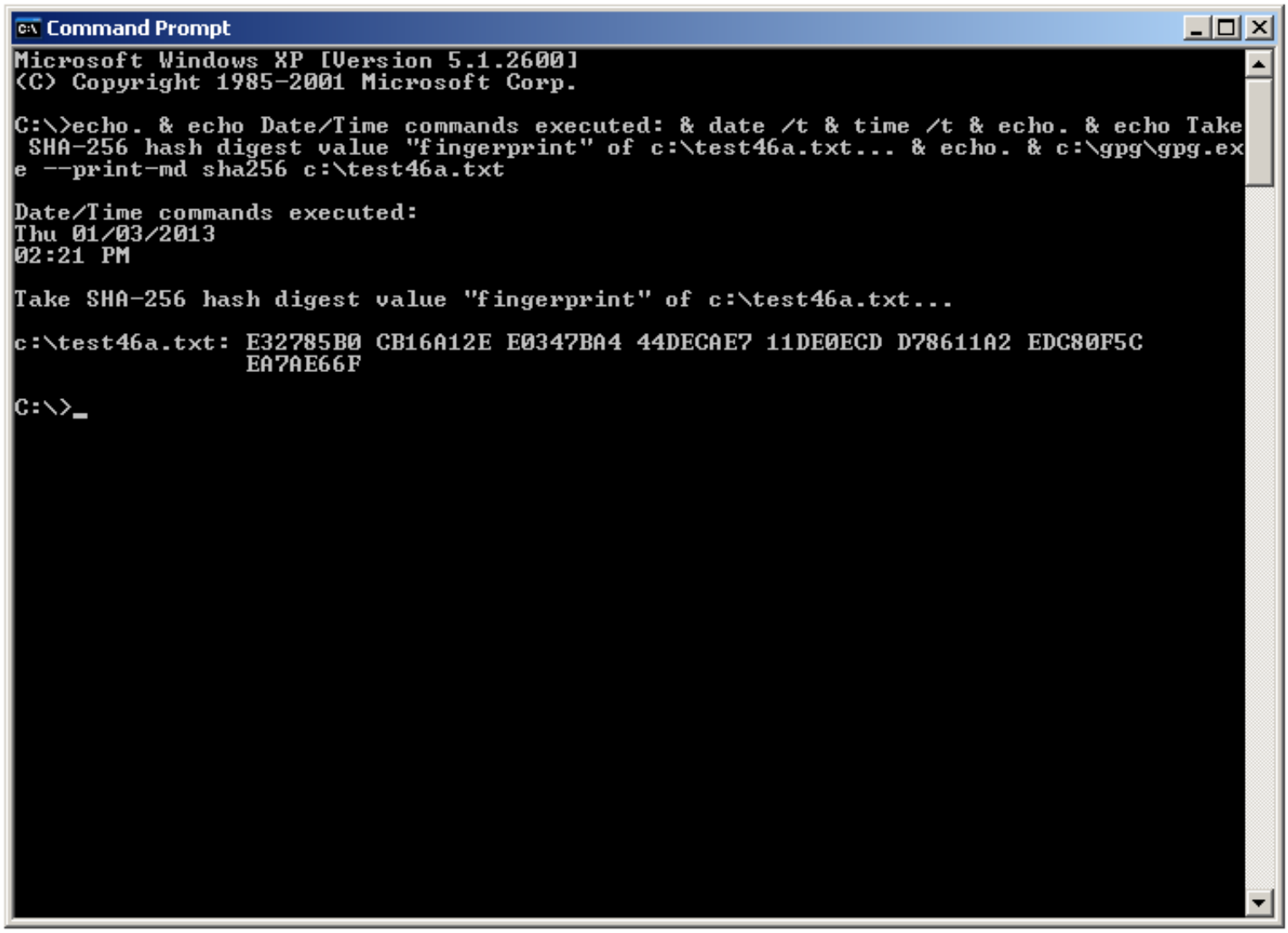

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test46a.txt... & echo. & echo File creation date: & dir c:\test46a.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test46a.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:21 PM
Now read "creation date" and "last modified date" file properties for c:\test46a.txt...
File creation date:
Directory of c:\
01/03/2013 02:21 PM 10 test46a.txt
1 File(s) 10 bytes
File last modified date:
Directory of c:\
01/03/2013 02:21 PM 10 test46a.txt
1 File(s) 10 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 53

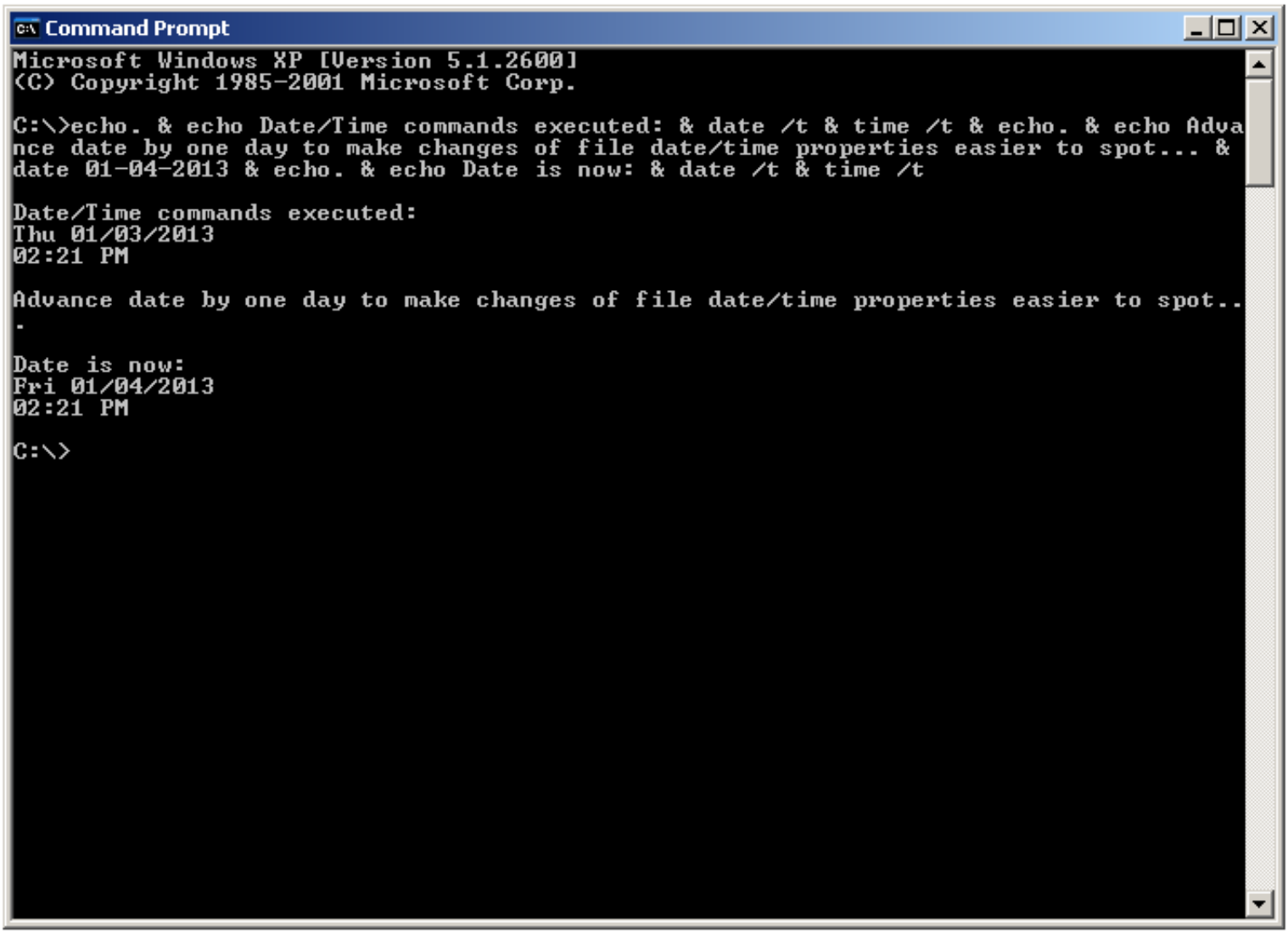

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test46a.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test46a.txt
Date/Time commands executed:
Thu 01/03/2013
02:21 PM
Take SHA-256 hash digest value "fingerprint" of c:\test46a.txt...
c:\test46a.txt: E32785B0 CB16A12E E0347BA4 44DECAE7 11DE0ECD D78611A2 EDC80F5C
EA7AE66F
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 54

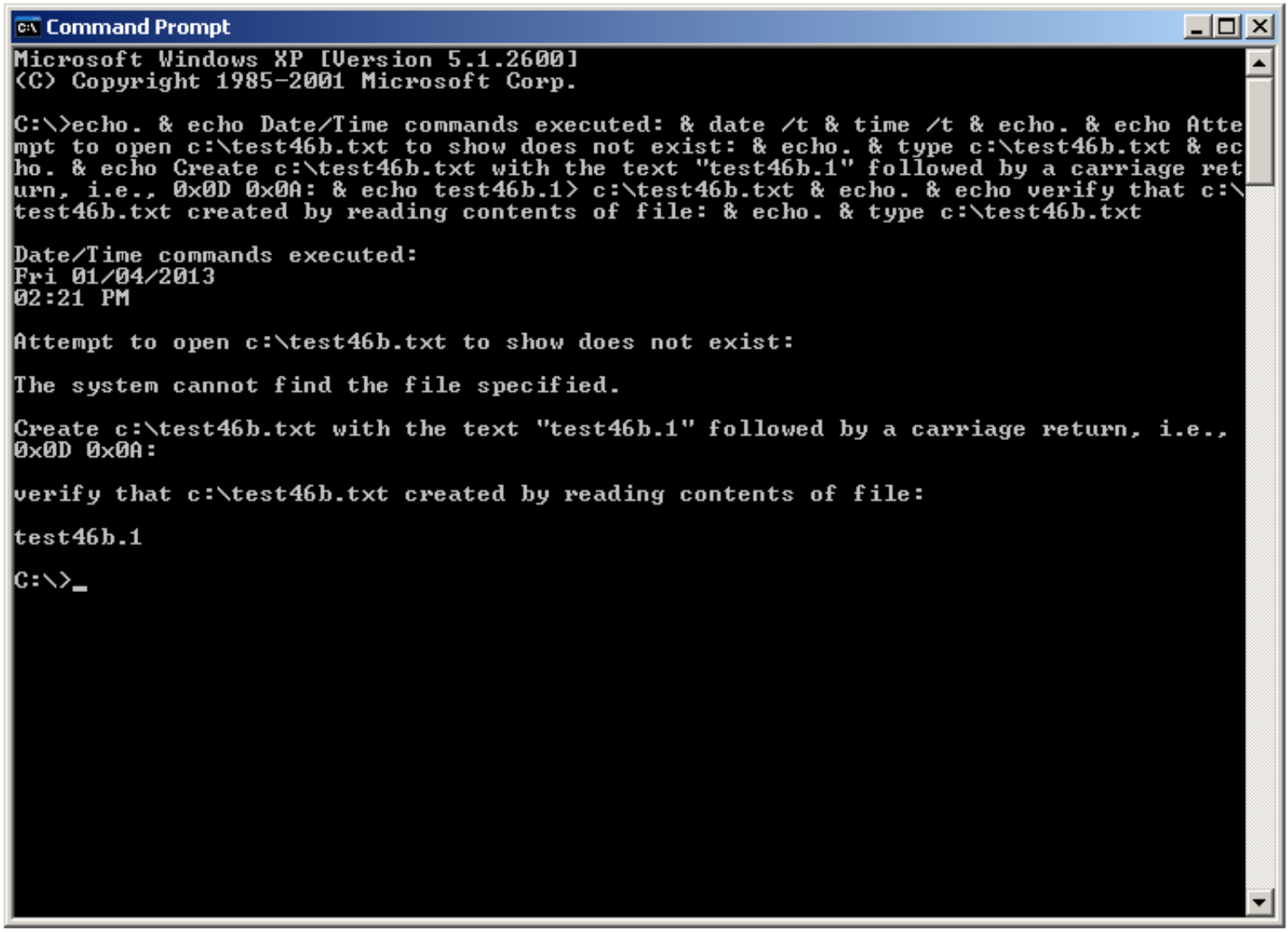

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Adva
nce date by one day to make changes of file date/time properties easier to spot... &
date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
02:21 PM
Advance date by one day to make changes of file date/time properties easier to spot..
.
Date is now:
Fri 01/04/2013
02:21 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 55

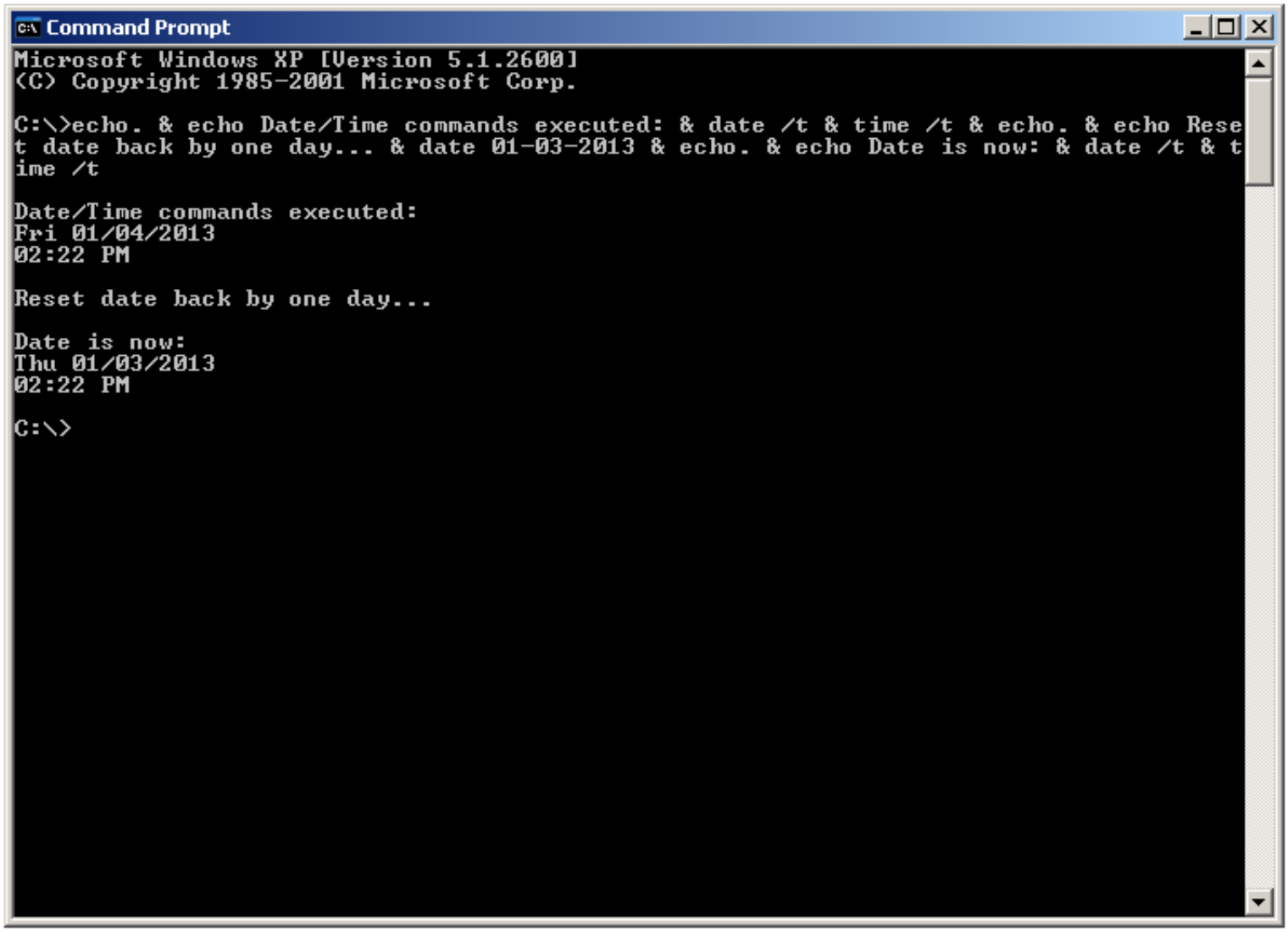

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test46b.txt to show does not exist: & echo. & type c:\test46b.txt & echo. & echo Create c:\test46b.txt with the text "test46b.1" followed by a carriage return, i.e., 0x0D 0x0A: & echo test46b.1> c:\test46b.txt & echo. & echo verify that c:\test46b.txt created by reading contents of file: & echo. & type c:\test46b.txt
Date/Time commands executed:
Fri 01/04/2013
02:21 PM
Attempt to open c:\test46b.txt to show does not exist:
The system cannot find the file specified.
Create c:\test46b.txt with the text "test46b.1" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test46b.txt created by reading contents of file:
test46b.1
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 56

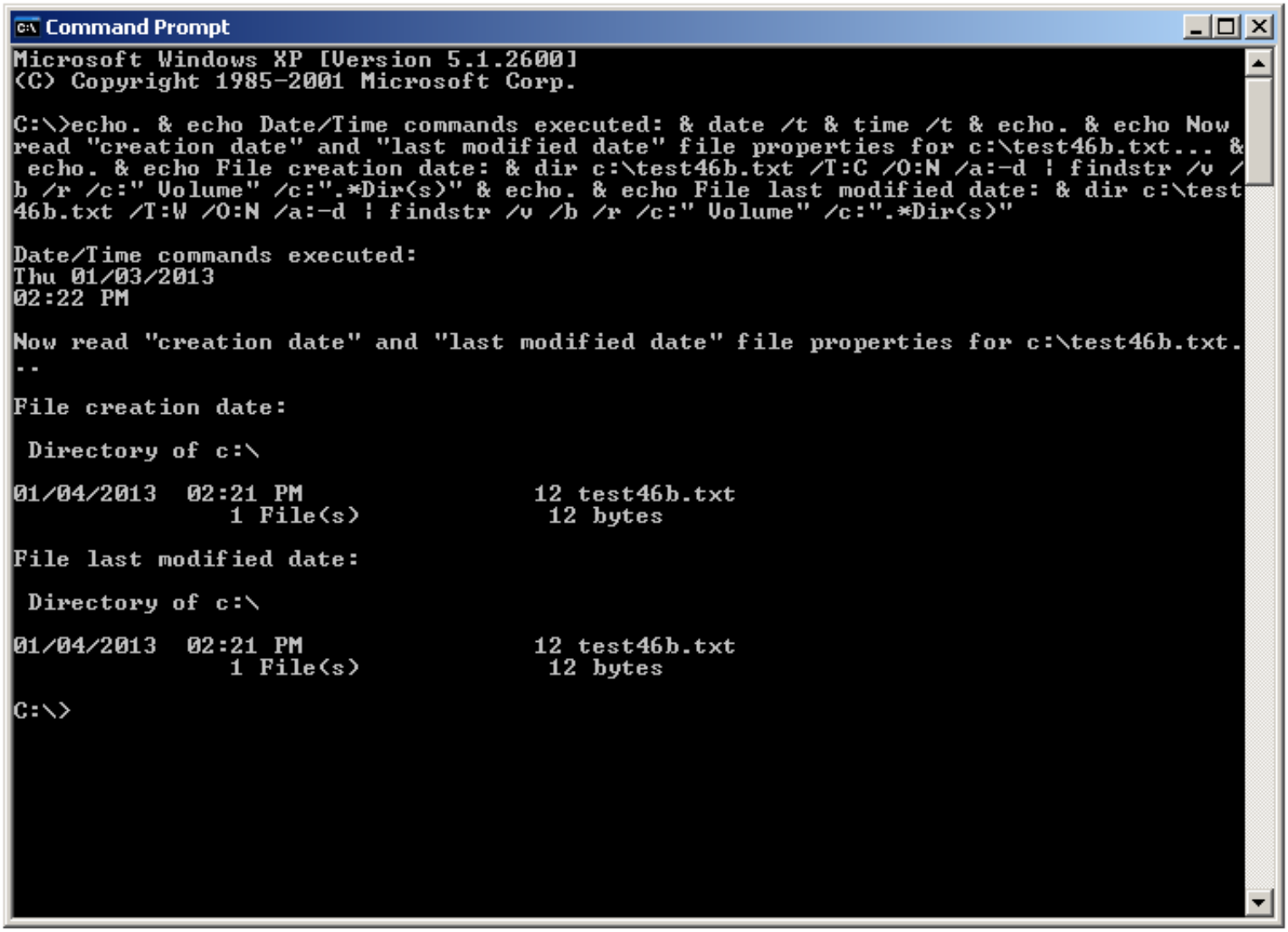

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
02:22 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
02:22 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 57

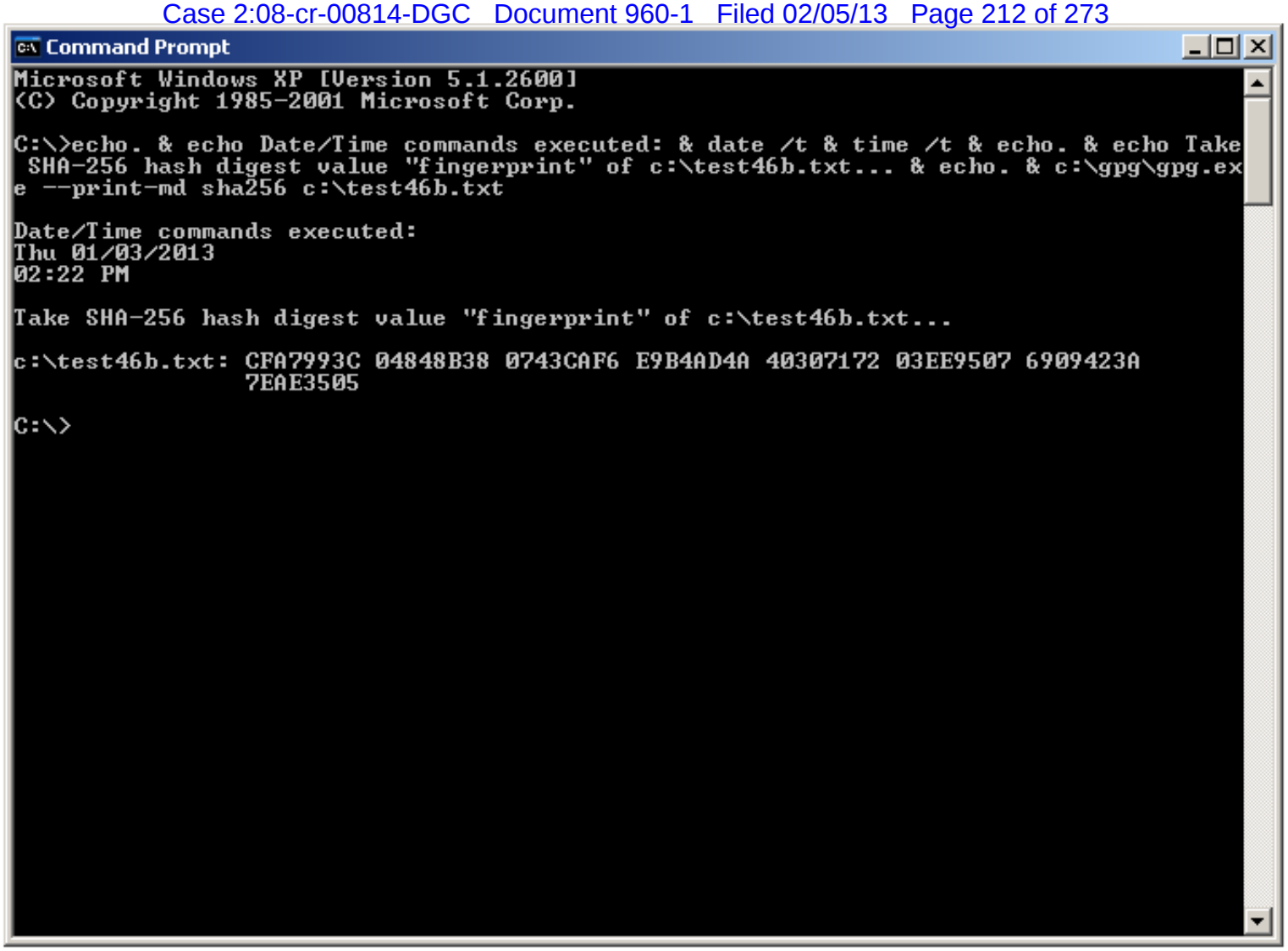

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test46b.txt... & echo. & echo File creation date: & dir c:\test46b.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test46b.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:22 PM
Now read "creation date" and "last modified date" file properties for c:\test46b.txt...
File creation date:
Directory of c:\
01/04/2013 02:21 PM 12 test46b.txt
1 File(s) 12 bytes
File last modified date:
Directory of c:\
01/04/2013 02:21 PM 12 test46b.txt
1 File(s) 12 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 58

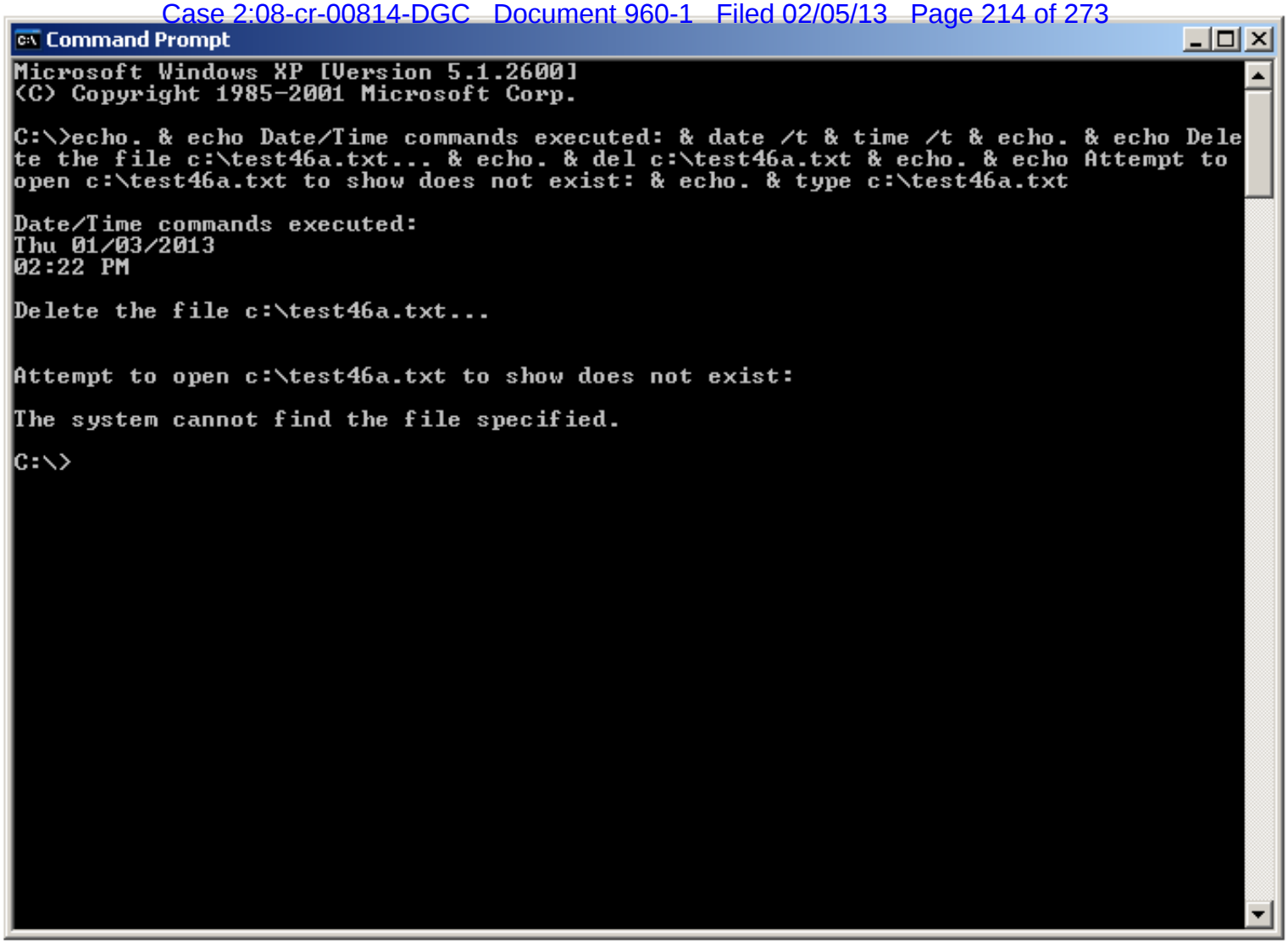

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test46b.txt... & echo. & c:\gpg\gpg.ex
e --print-md sha256 c:\test46b.txt
Date/Time commands executed:
Thu 01/03/2013
02:22 PM
Take SHA-256 hash digest value "fingerprint" of c:\test46b.txt...
c:\test46b.txt: CFA7993C 04848B38 0743CAF6 E9B4AD4A 40307172 03EE9507 6909423A
7EAE3505
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 59

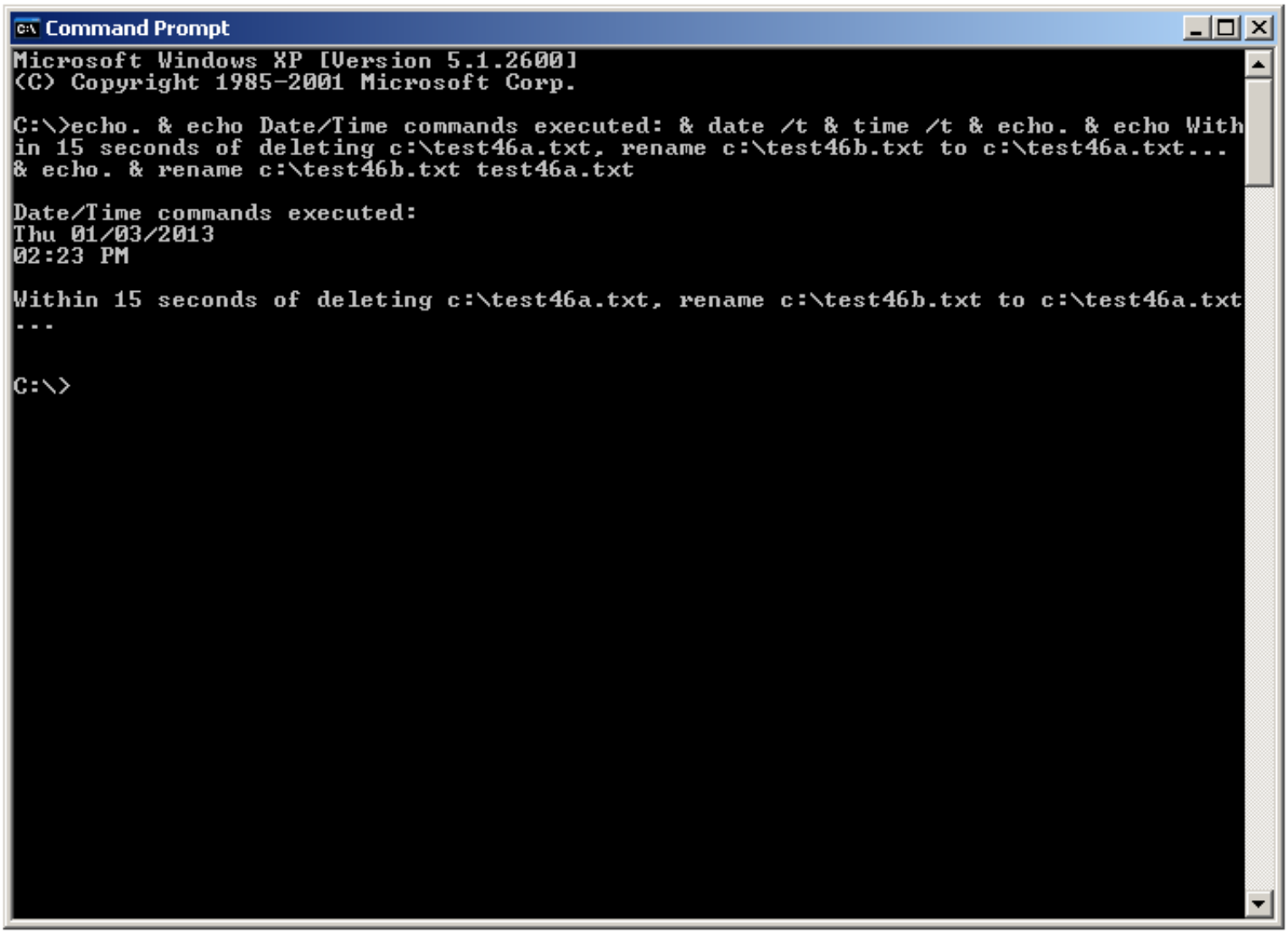

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Dele
te the file c:\test46a.txt... & echo. & del c:\test46a.txt & echo. & echo Attempt to
open c:\test46a.txt to show does not exist: & echo. & type c:\test46a.txt
Date/Time commands executed:
Thu 01/03/2013
02:22 PM
Delete the file c:\test46a.txt...
Attempt to open c:\test46a.txt to show does not exist:
The system cannot find the file specified.
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 60

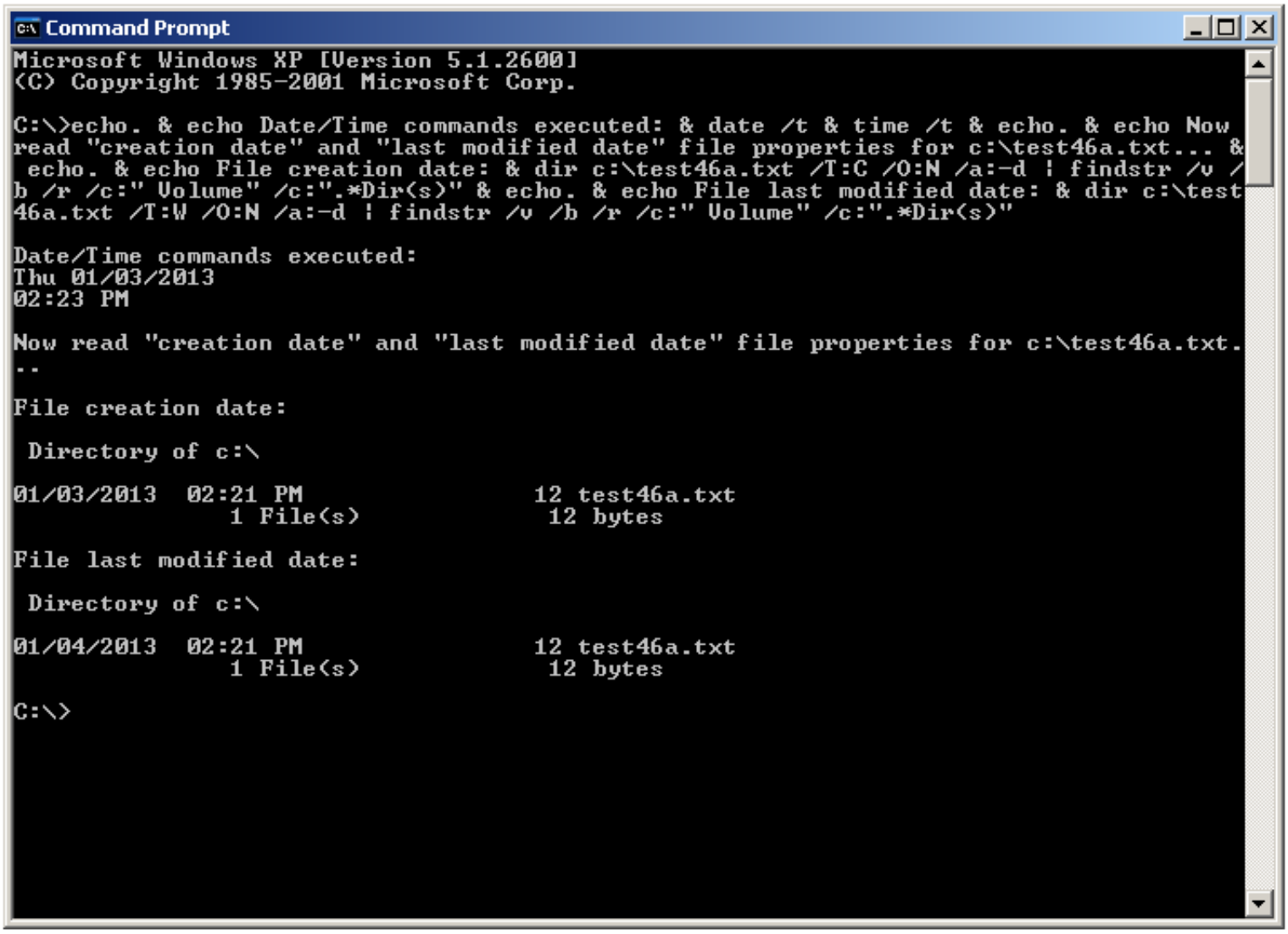

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within 15 seconds of deleting c:\test46a.txt, rename c:\test46b.txt to c:\test46a.txt... & echo. & rename c:\test46b.txt test46a.txt
Date/Time commands executed:
Thu 01/03/2013
02:23 PM
Within 15 seconds of deleting c:\test46a.txt, rename c:\test46b.txt to c:\test46a.txt...
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 61

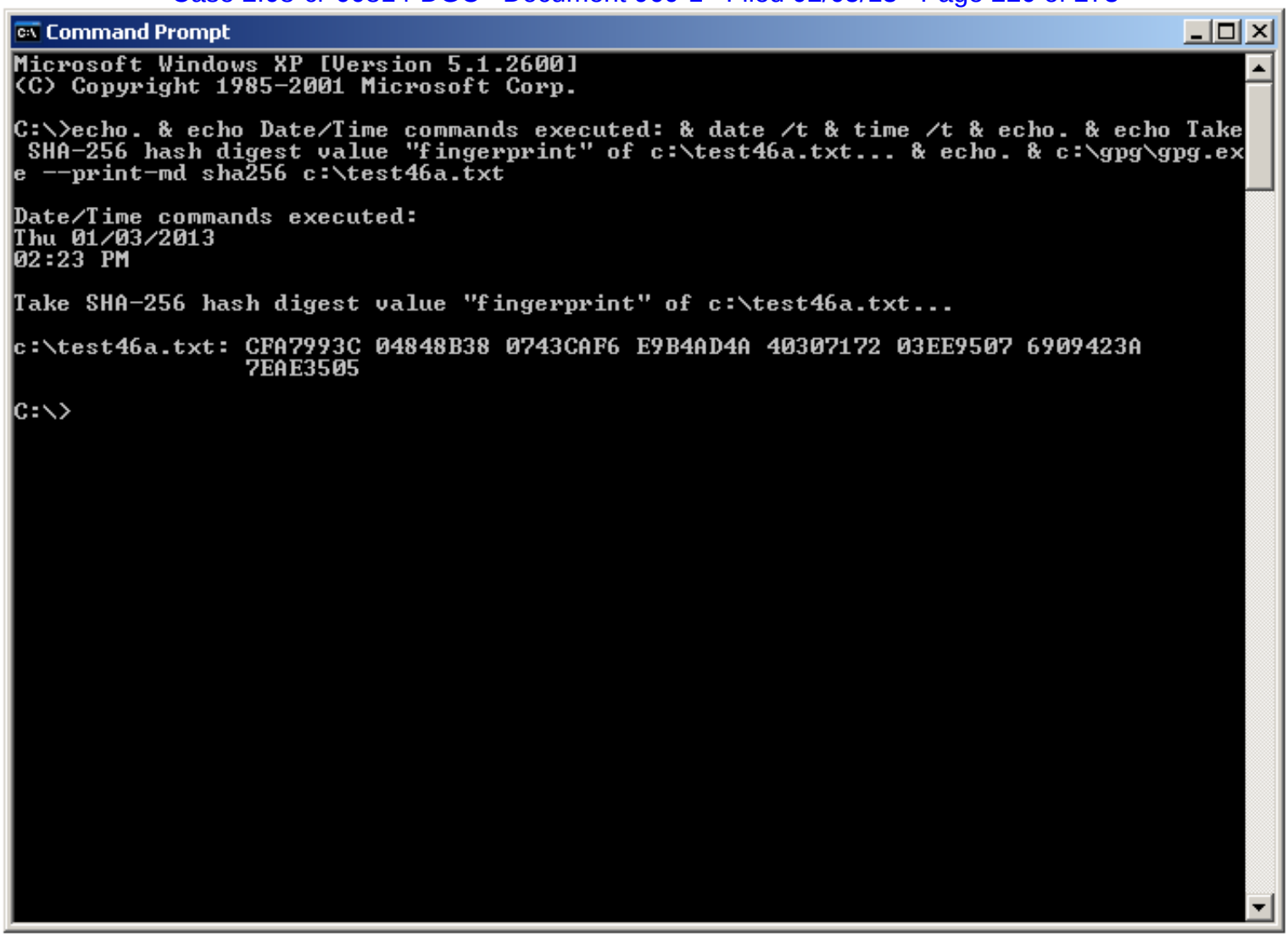

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test46a.txt... & echo. & echo File creation date: & dir c:\test46a.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test46a.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:23 PM
Now read "creation date" and "last modified date" file properties for c:\test46a.txt...
File creation date:
Directory of c:\
01/03/2013 02:21 PM 12 test46a.txt
1 File(s) 12 bytes
File last modified date:
Directory of c:\
01/04/2013 02:21 PM 12 test46a.txt
1 File(s) 12 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 62

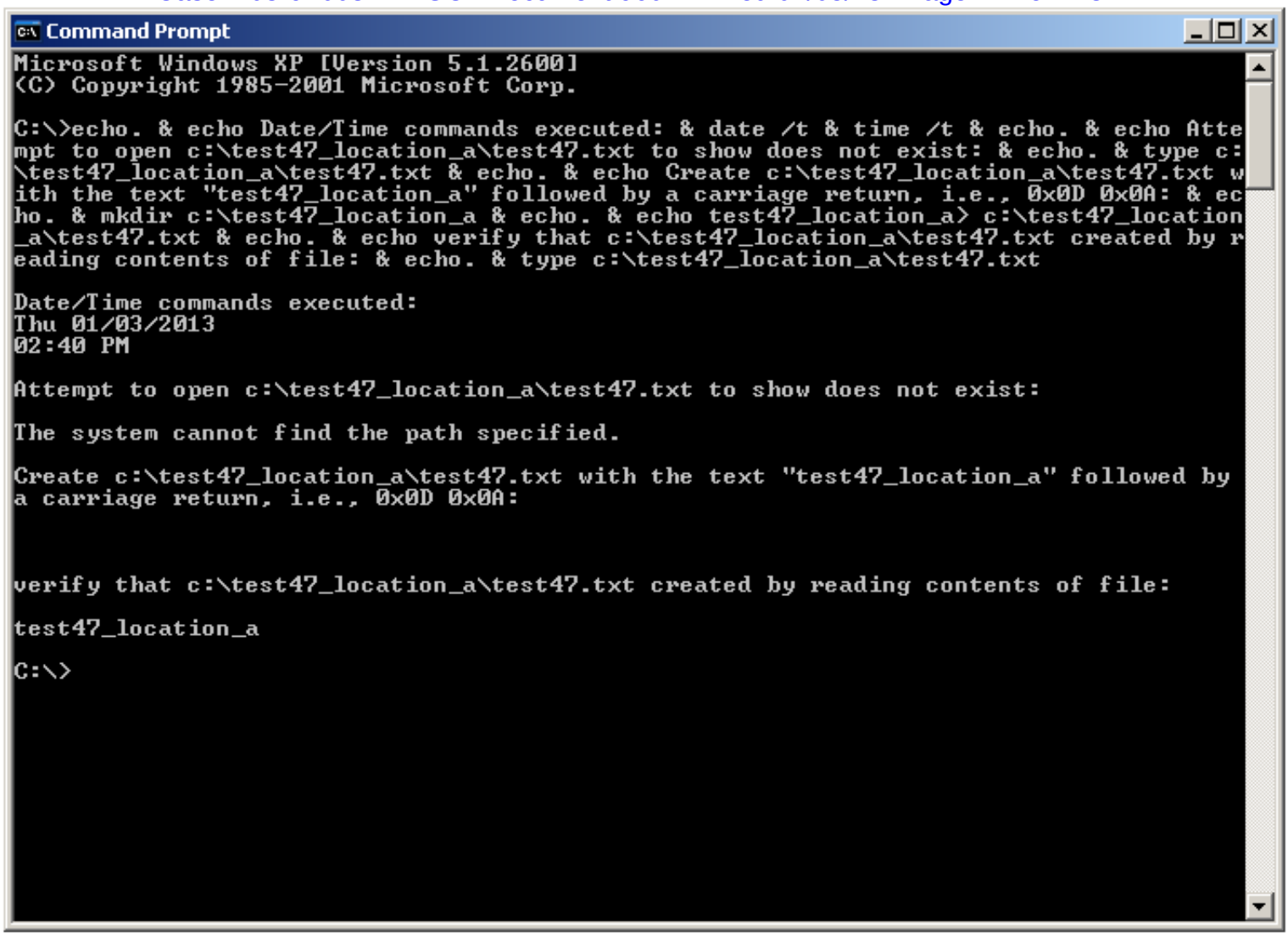

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take
SHA-256 hash digest value "fingerprint" of c:\test46a.txt... & echo. & c:\gpg\gpg.ex
e --print-md sha256 c:\test46a.txt
Date/Time commands executed:
Thu 01/03/2013
02:23 PM
Take SHA-256 hash digest value "fingerprint" of c:\test46a.txt...
c:\test46a.txt: CFA7993C 04848B38 0743CAF6 E9B4AD4A 40307172 03EE9507 6909423A
7EAE3505
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 63

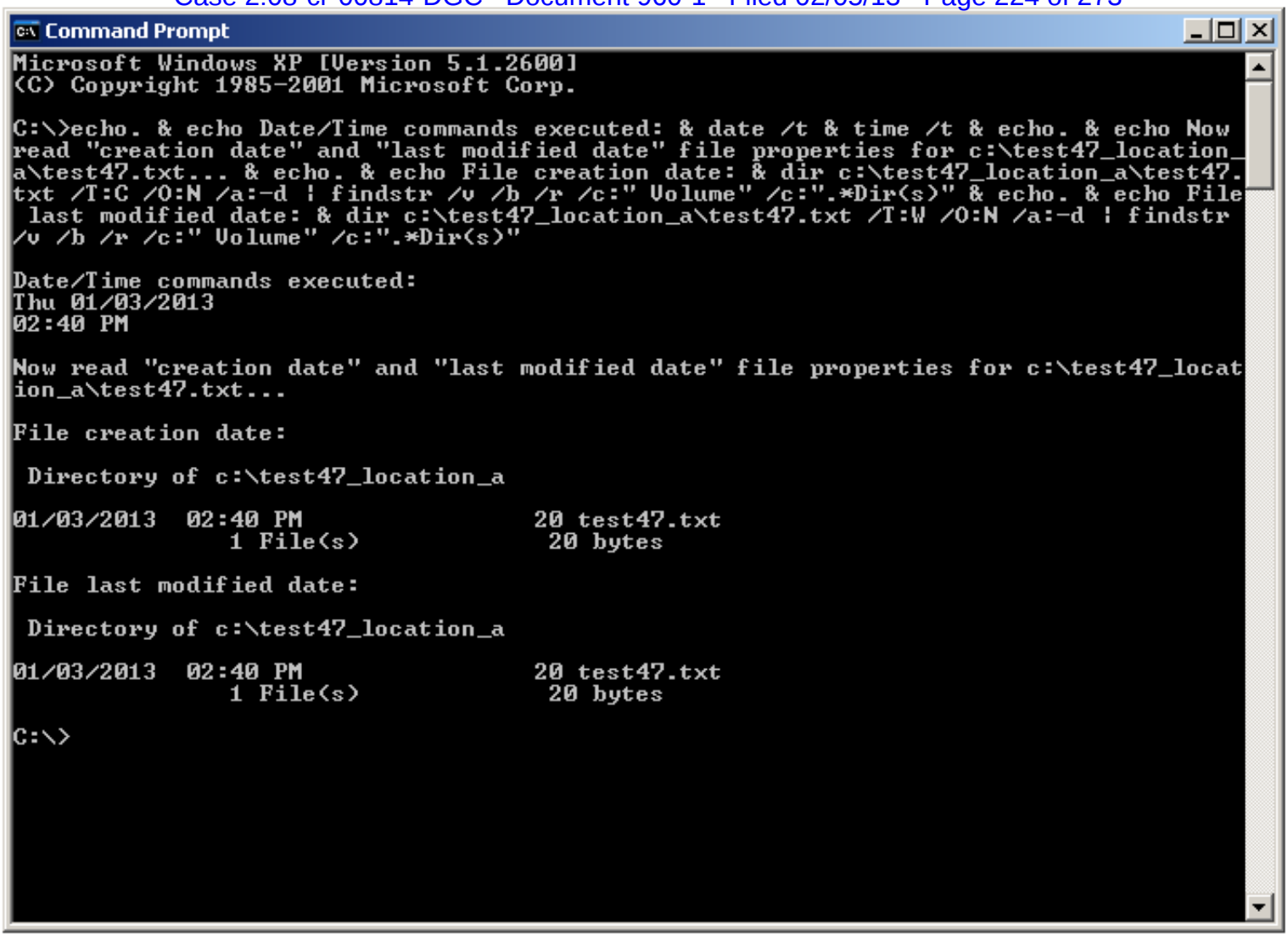

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test47_location_a\test47.txt to show does not exist: & echo. & type c:\test47_location_a\test47.txt & echo. & echo Create c:\test47_location_a\test47.txt with the text "test47_location_a" followed by a carriage return, i.e., 0x0D 0x0A: & echo. & mkdir c:\test47_location_a & echo. & echo test47_location_a> c:\test47_location_a\test47.txt & echo. & echo verify that c:\test47_location_a\test47.txt created by reading contents of file: & echo. & type c:\test47_location_a\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:40 PM
Attempt to open c:\test47_location_a\test47.txt to show does not exist:
The system cannot find the path specified.
Create c:\test47_location_a\test47.txt with the text "test47_location_a" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test47_location_a\test47.txt created by reading contents of file:
test47_location_a
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 64

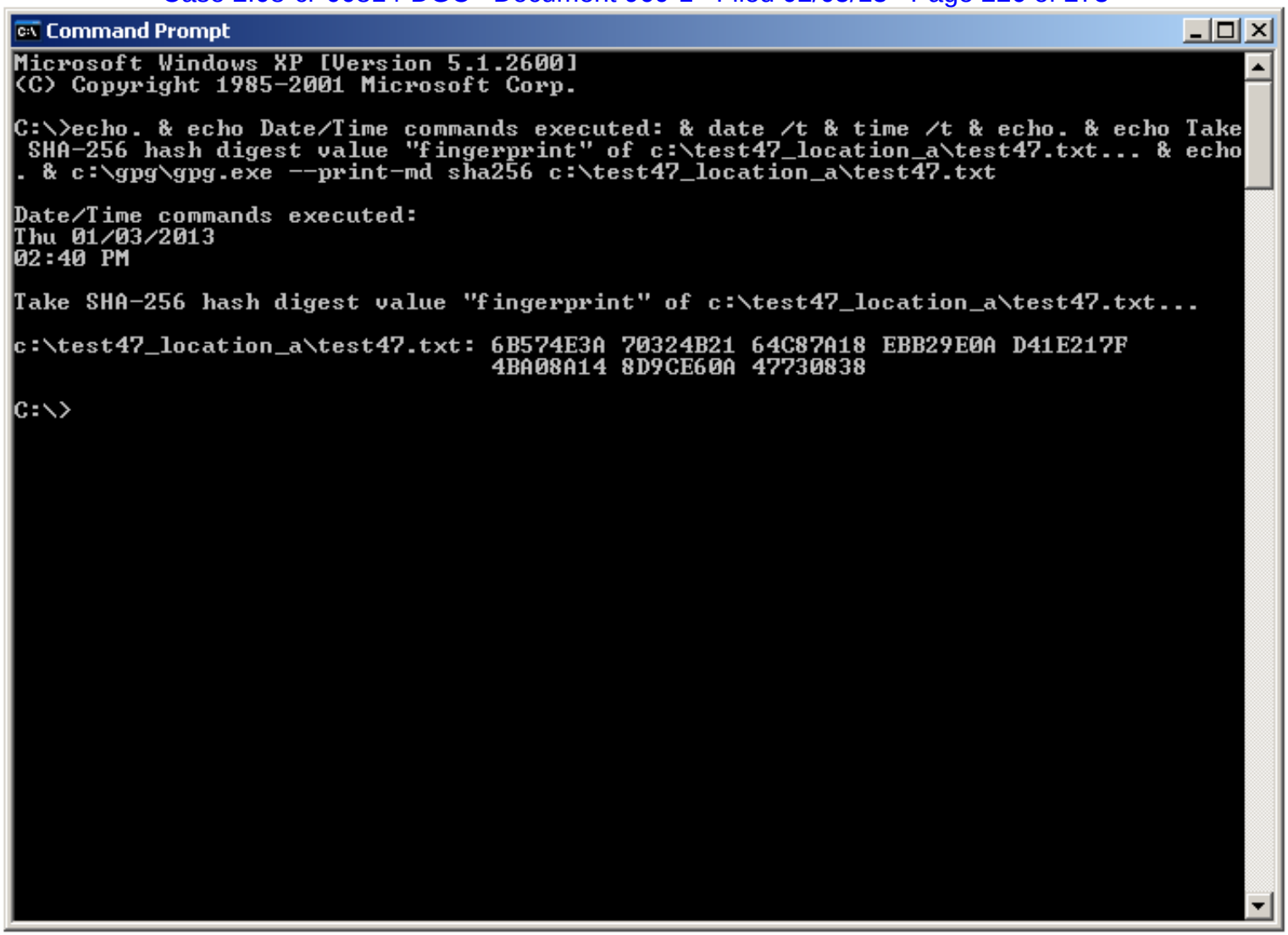

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test47_location_a\test47.txt... & echo. & echo File creation date: & dir c:\test47_location_a\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test47_location_a\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:40 PM
Now read "creation date" and "last modified date" file properties for c:\test47_location_a\test47.txt...
File creation date:
Directory of c:\test47_location_a
01/03/2013 02:40 PM 20 test47.txt
1 File(s) 20 bytes
File last modified date:
Directory of c:\test47_location_a
01/03/2013 02:40 PM 20 test47.txt
1 File(s) 20 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 65

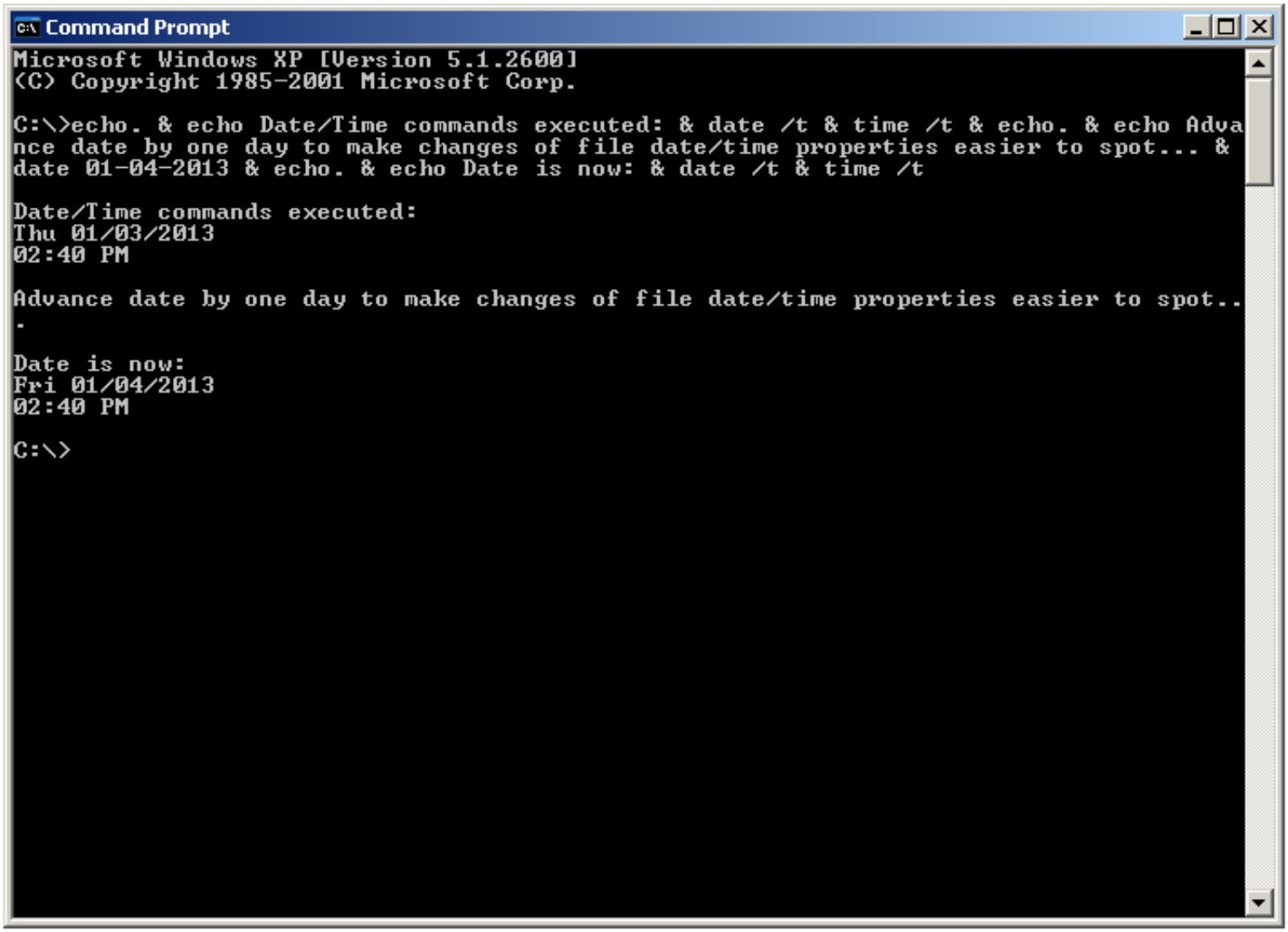

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test47_location_a\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:40 PM
Take SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt...
c:\test47_location_a\test47.txt: 6B574E3A 70324B21 64C87A18 EBB29E0A D41E217F
4BA08A14 8D9CE60A 47730838
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 66

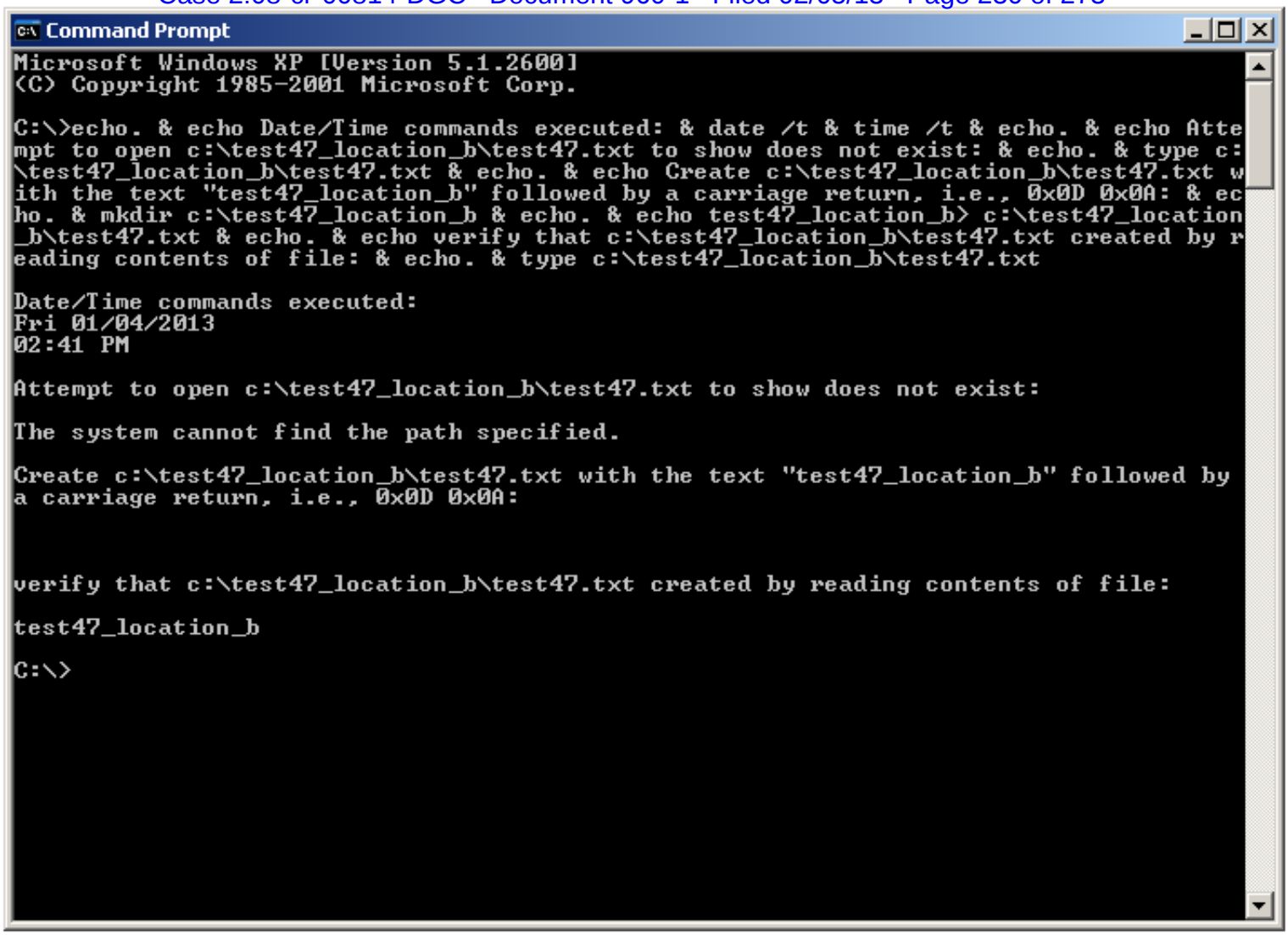

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Advance date by one day to make changes of file date/time properties easier to spot... & date 01-04-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Thu 01/03/2013
02:40 PM
Advance date by one day to make changes of file date/time properties easier to spot...
Date is now:
Fri 01/04/2013
02:40 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 67

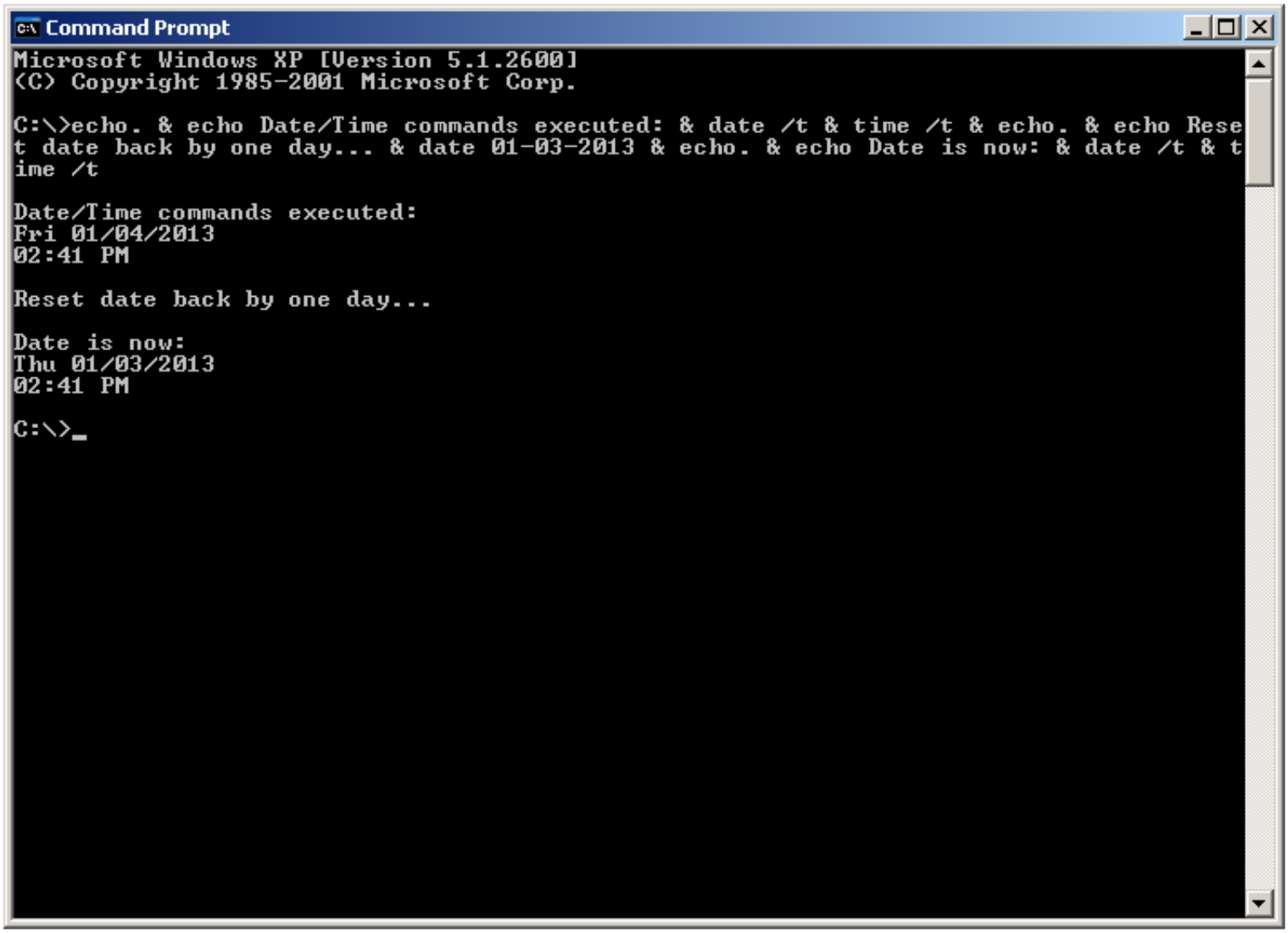

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Attempt to open c:\test47_location_b\test47.txt to show does not exist: & echo. & type c:\test47_location_b\test47.txt & echo. & echo Create c:\test47_location_b\test47.txt with the text "test47_location_b" followed by a carriage return, i.e., 0x0D 0x0A: & echo. & mkdir c:\test47_location_b & echo. & echo test47_location_b> c:\test47_location_b\test47.txt & echo. & echo verify that c:\test47_location_b\test47.txt created by reading contents of file: & echo. & type c:\test47_location_b\test47.txt
Date/Time commands executed:
Fri 01/04/2013
02:41 PM
Attempt to open c:\test47_location_b\test47.txt to show does not exist:
The system cannot find the path specified.
Create c:\test47_location_b\test47.txt with the text "test47_location_b" followed by a carriage return, i.e., 0x0D 0x0A:
verify that c:\test47_location_b\test47.txt created by reading contents of file:
test47_location_b
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 68

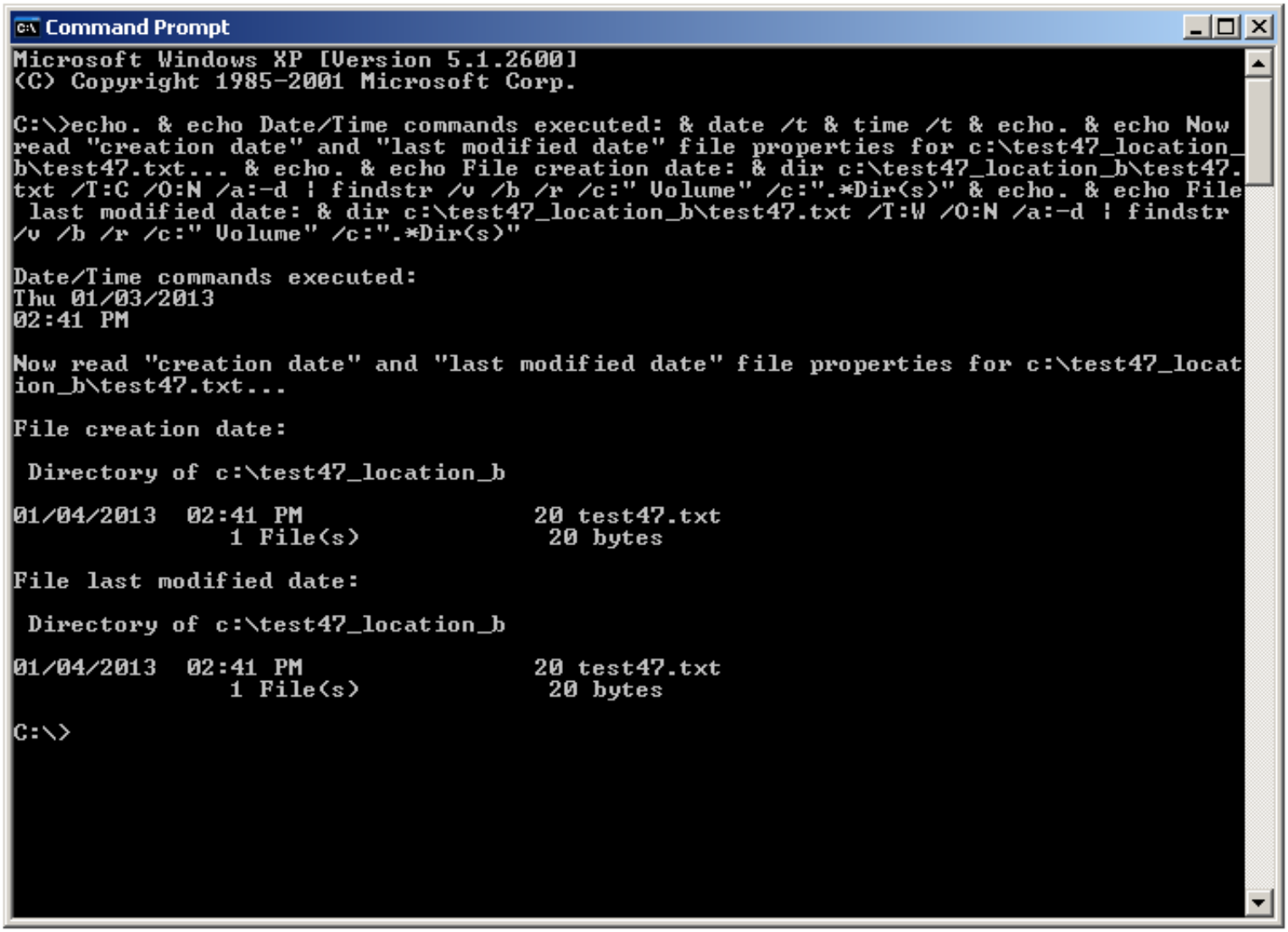

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Reset date back by one day... & date 01-03-2013 & echo. & echo Date is now: & date /t & time /t
Date/Time commands executed:
Fri 01/04/2013
02:41 PM
Reset date back by one day...
Date is now:
Thu 01/03/2013
02:41 PM
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 69

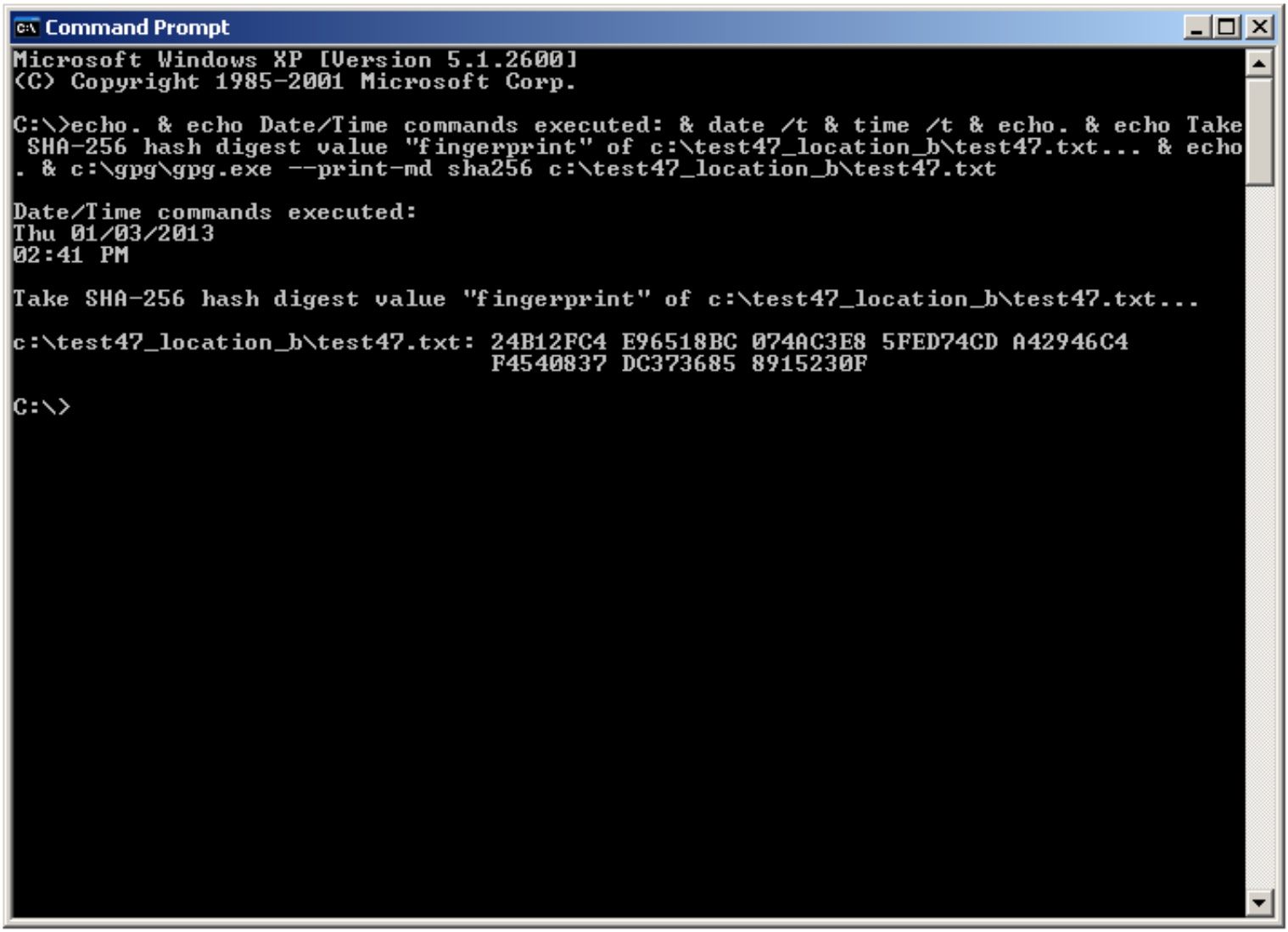

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test47_location_b\test47.txt... & echo. & echo File creation date: & dir c:\test47_location_b\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test47_location_b\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:41 PM
Now read "creation date" and "last modified date" file properties for c:\test47_location_b\test47.txt...
File creation date:
Directory of c:\test47_location_b
01/04/2013 02:41 PM 20 test47.txt
1 File(s) 20 bytes
File last modified date:
Directory of c:\test47_location_b
01/04/2013 02:41 PM 20 test47.txt
1 File(s) 20 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 70

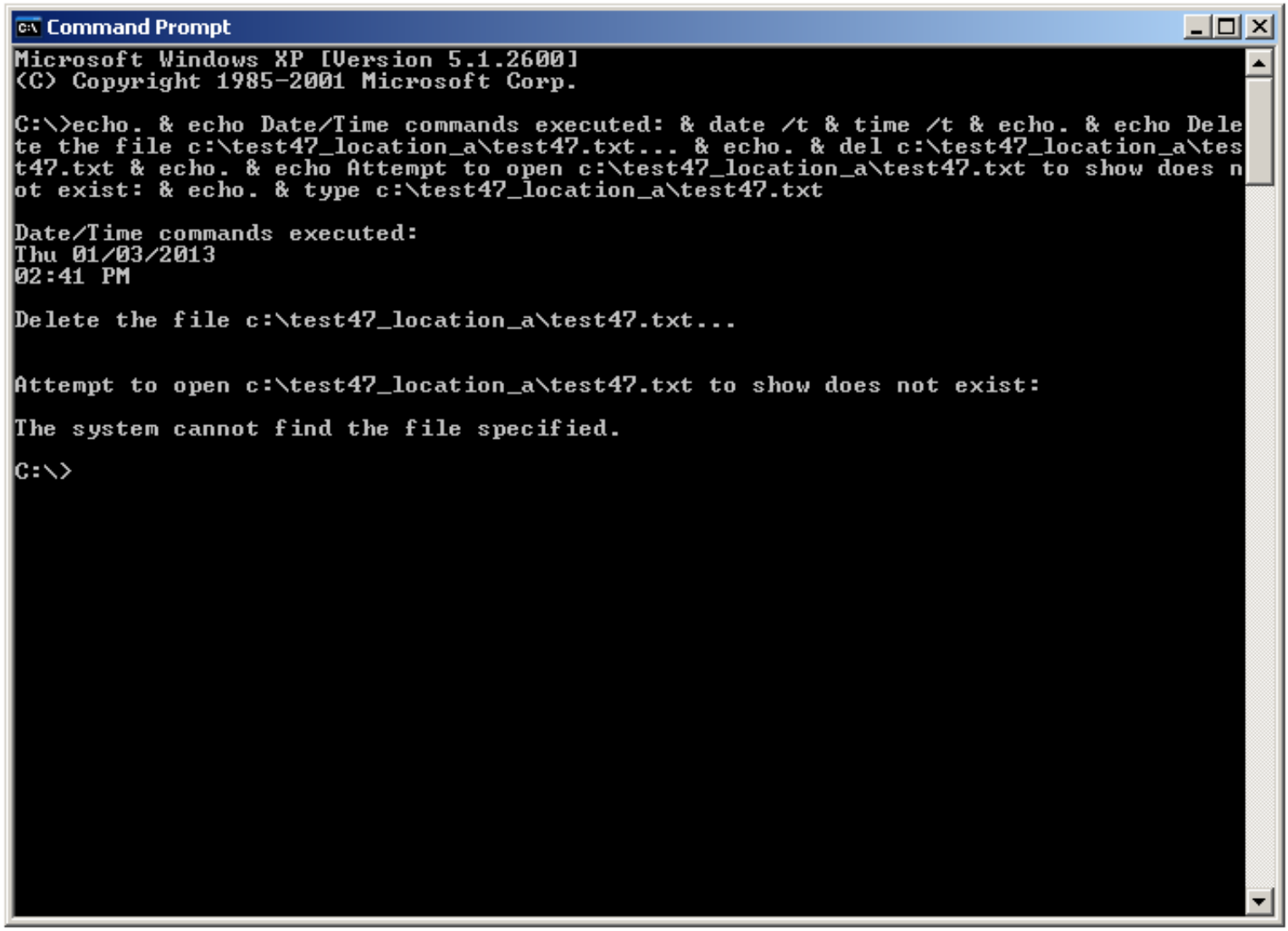

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test47_location_b\test47.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test47_location_b\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:41 PM
Take SHA-256 hash digest value "fingerprint" of c:\test47_location_b\test47.txt...
c:\test47_location_b\test47.txt: 24B12FC4 E96518BC 074AC3E8 5FED74CD A42946C4
F4540837 DC373685 8915230F
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 71

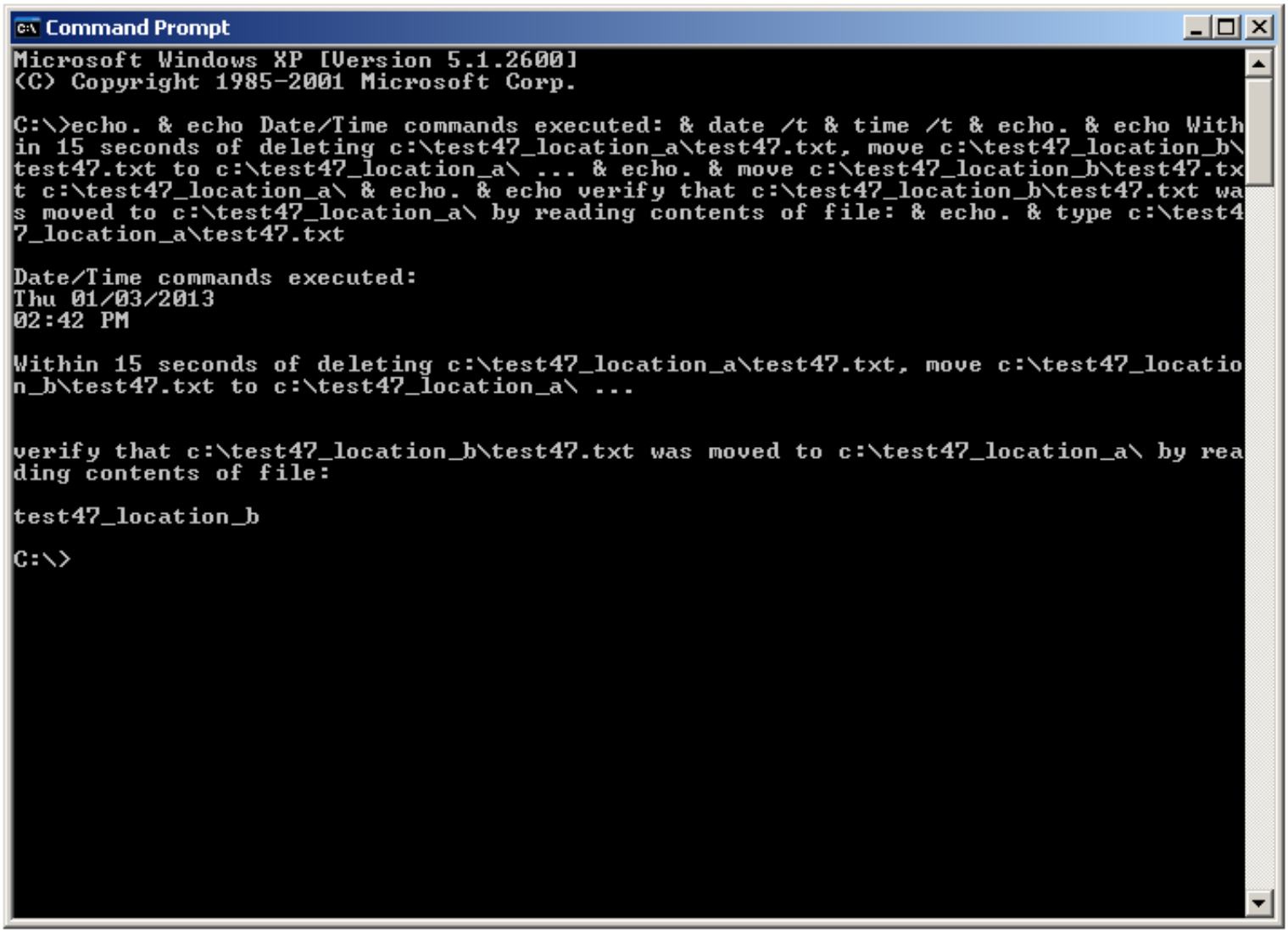

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Dele
te the file c:\test47_location_a\test47.txt... & echo. & del c:\test47_location_a\tes
t47.txt & echo. & echo Attempt to open c:\test47_location_a\test47.txt to show does n
ot exist: & echo. & type c:\test47_location_a\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:41 PM
Delete the file c:\test47_location_a\test47.txt...
Attempt to open c:\test47_location_a\test47.txt to show does not exist:
The system cannot find the file specified.
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 72

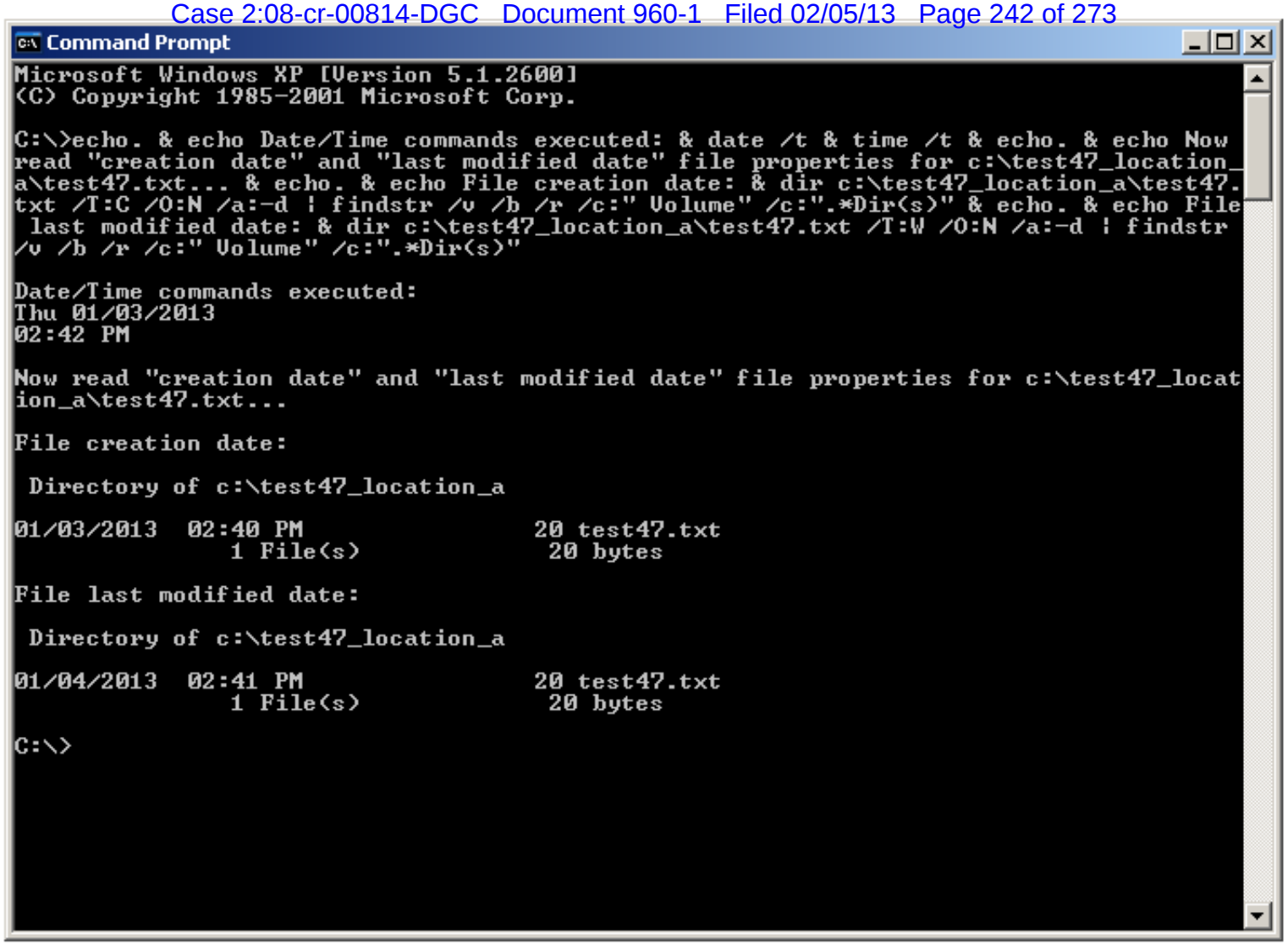

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Within 15 seconds of deleting c:\test47_location_a\test47.txt, move c:\test47_location_b\test47.txt to c:\test47_location_a\ ... & echo. & move c:\test47_location_b\test47.txt c:\test47_location_a\ & echo. & echo verify that c:\test47_location_b\test47.txt was moved to c:\test47_location_a\ by reading contents of file: & echo. & type c:\test47_location_a\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:42 PM
Within 15 seconds of deleting c:\test47_location_a\test47.txt, move c:\test47_location_b\test47.txt to c:\test47_location_a\ ...
verify that c:\test47_location_b\test47.txt was moved to c:\test47_location_a\ by reading contents of file:
test47_location_b
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 73

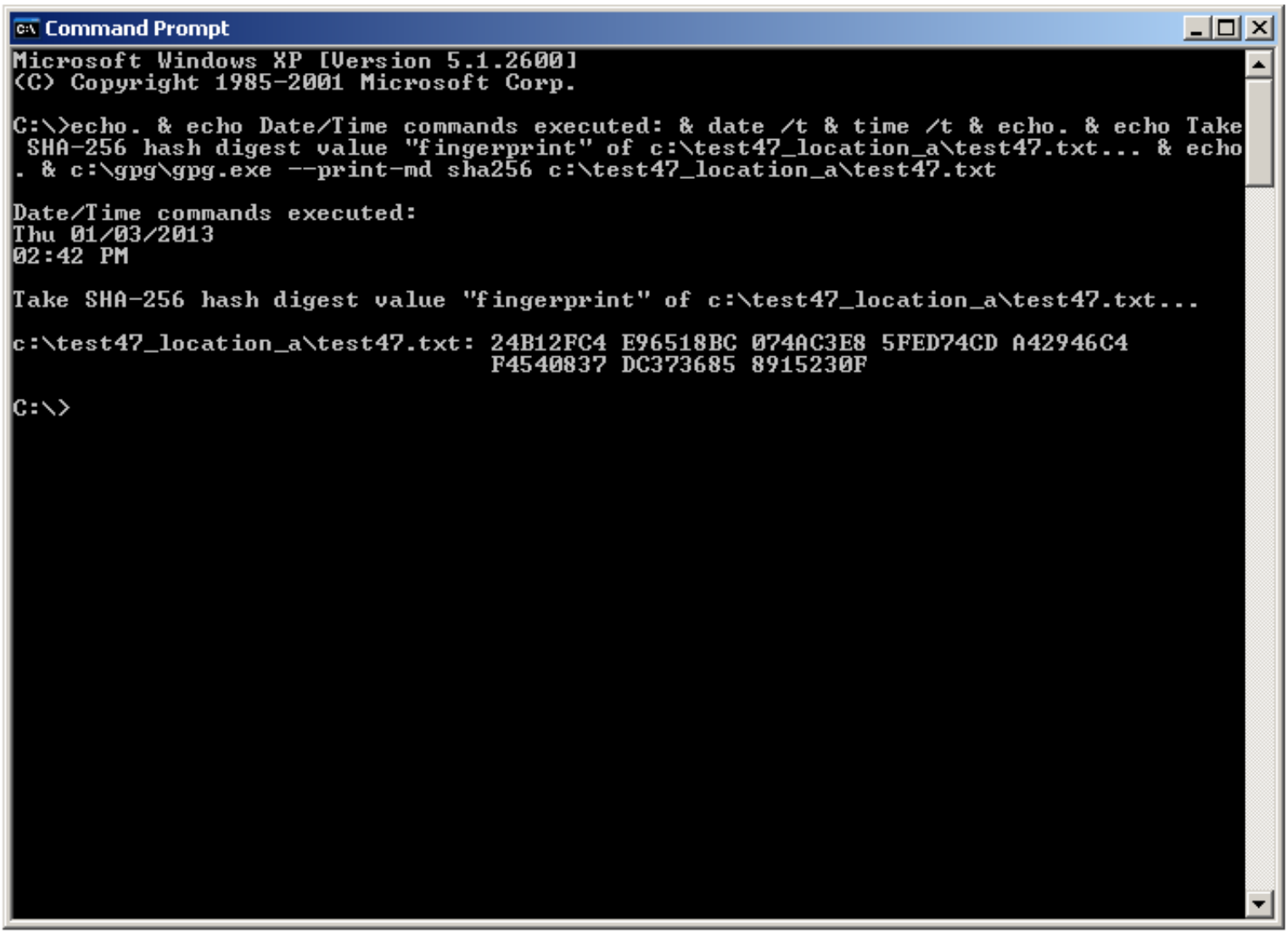

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Now read "creation date" and "last modified date" file properties for c:\test47_location_a\test47.txt... & echo. & echo File creation date: & dir c:\test47_location_a\test47.txt /T:C /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)" & echo. & echo File last modified date: & dir c:\test47_location_a\test47.txt /T:W /O:N /a:-d | findstr /v /b /r /c:" Volume" /c:".*Dir(s)"
Date/Time commands executed:
Thu 01/03/2013
02:42 PM
Now read "creation date" and "last modified date" file properties for c:\test47_location_a\test47.txt...
File creation date:
Directory of c:\test47_location_a
01/03/2013 02:40 PM 20 test47.txt
1 File(s) 20 bytes
File last modified date:
Directory of c:\test47_location_a
01/04/2013 02:41 PM 20 test47.txt
1 File(s) 20 bytes
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 74


Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>echo. & echo Date/Time commands executed: & date /t & time /t & echo. & echo Take SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt... & echo. & c:\gpg\gpg.exe --print-md sha256 c:\test47_location_a\test47.txt
Date/Time commands executed:
Thu 01/03/2013
02:42 PM
Take SHA-256 hash digest value "fingerprint" of c:\test47_location_a\test47.txt...
c:\test47_location_a\test47.txt: 24B12FC4 E96518BC 074AC3E8 5FED74CD A42946C4
F4540837 DC373685 8915230F
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 75

Help and Support Center overview

Microsoft Help and Support Center is a comprehensive resource for practical advice, tutorials, and demonstrations to help you learn to use Microsoft Windows XP. Use the Search feature, Index, or table of contents to view all Windows Help resources, including those that are on the Internet.

In addition to Help resources, you can take advantage of various services and perform important support tasks - all from a single location.

In Help and Support, you can:

- Let a friend help you over the Internet by using Remote Assistance.
- Keep your computer up-to-date with the latest downloads from Windows Update.
- Research which hardware and software are compatible with Windows XP.
- Get help online from a support professional by using Microsoft Online Assisted Support.
- Undo changes to your computer by using System Restore.
- Use tools such as System Information to manage and maintain your computer.
- Stay current with the latest support information and Help news from sources such as Microsoft Product Support Services and your computer manufacturer.

Note

- The Help topics in Help and Support Center are specific to the Windows XP operating system. Other programs that you might have installed on your computer (for example, Microsoft Word or Microsoft Excel) have their own Help topics that you can view from within those particular programs.

Related Topics

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 76

RAR LAB

Welcome to RARLAB, home of WinRAR and RAR archivers

Home
RAR
News
Themes
Extras
FAR
Downloads
Dealers
Feedback
Partnership
Other

WinRAR is a powerful archive manager. It can backup your data and reduce size of email attachments, decompress RAR, ZIP and other files downloaded from Internet and create new archives in RAR and ZIP file format. You may try WinRAR before buy, its trial version is available in downloads.

Our shop

Better compression, easier use, lower price. All extras included, upgrades free. Just click on the link below:

- **Buy WinRAR archiver**
- **Buy WinRAR and shareware CD**
- **Buy FAR 1.70 file manager**
- **Browse all products**

Our online service handles orders by credit cards, fax, bank/wire transfers and cheques.

Recommended Software

-Advertisement-

Spyware Doctor

Spyware Doctor is a top-rated malware & spyware removal utility that detects, removes and protects your PC from thousands of potential spyware, adware, spyware trojans, keyloggers, spybots and tracking threats. Protect your privacy and computing habits from prying eyes and virtual trespassers with the help of Spyware Doctor.

Click here to download More information

Find other fine software titles on our recommended software site

Last updated: 21 September 2007

- **WinRAR and RAR 3.71 release**

  WinRAR 3.71 available in Albanian, Arabic, Azerbaijani, Belarusian, Bosnian, Bulgarian, Catalan, Chinese Simplified, Chinese Traditional, Croatian, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hebrew, Hungarian, Italian, Japanese, Korean, Lithuanian, Macedonian, Norwegian, Persian, Polish, Portuguese, Portuguese Brazilian, Romanian, Russian, Serbian Cyrillic, Serbian Latin, Slovak, Spanish, Spanish (Latin American), Swedish, Thai, Turkish, Ukrainian, Uzbek, Valencian.

  Pocket RAR 3.71 available in Azerbaijani, Chinese Traditional, Danish, English, German, Italian, Korean, Polish, Spanish, Ukrainian.

- **New WinRAR themes**

  Tulliana, Alpha Dista

- **FAR 1.70 release**

  New FAR is out.

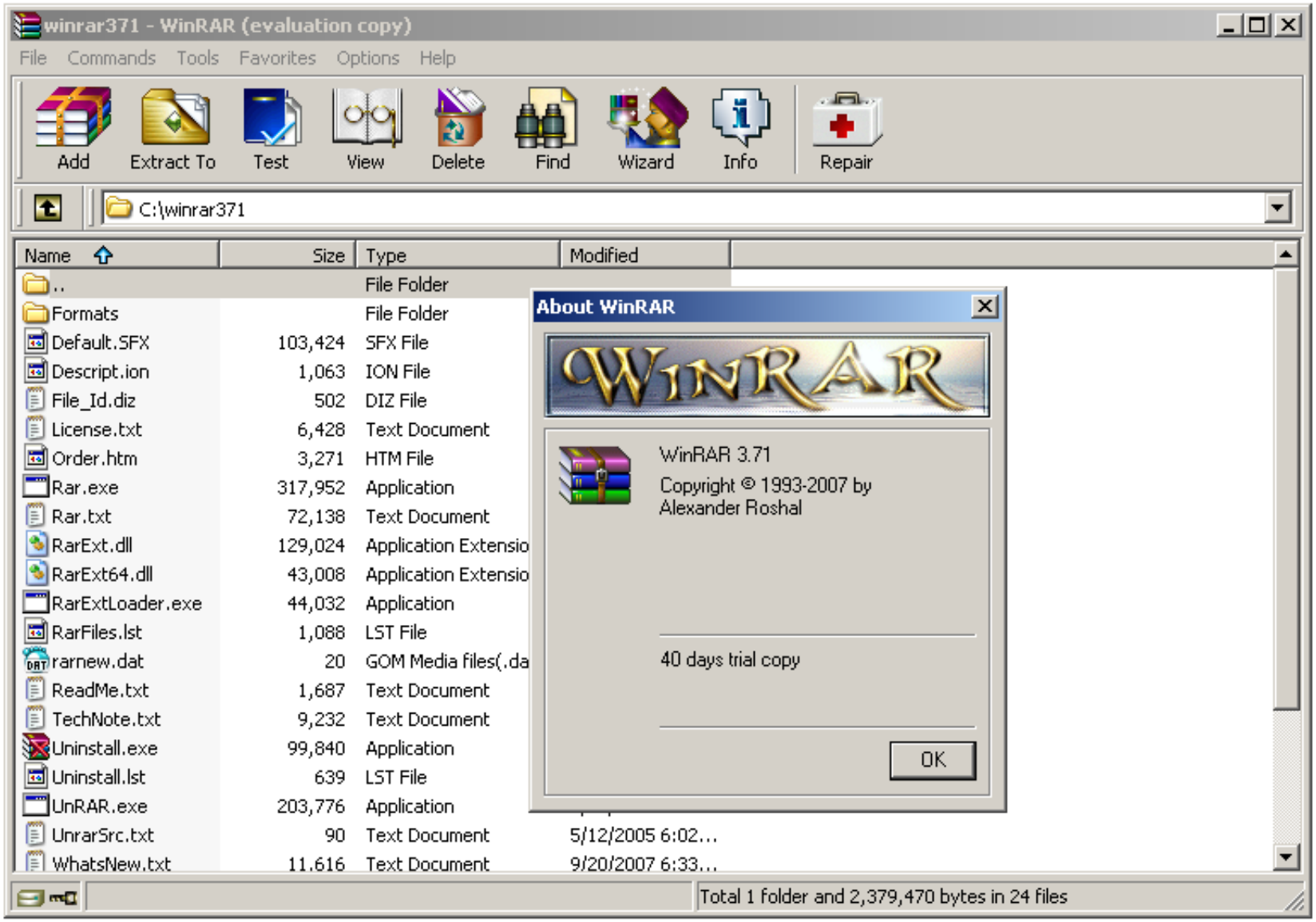

INTERNET ARCHIVE
Wayback Machine
Go
APR
AUG
Close
421 captures
2007
2009
Help

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 77


winrar371 - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info Repair
C:\winrar371
Name Size Type Modified
.. File Folder
Formats File Folder
Default.SFX 103,424 SFX File
Descript.ion 1,063 ION File
File_Id.diz 502 DIZ File
License.txt 6,428 Text Document
Order.htm 3,271 HTM File
Rar.exe 317,952 Application
Rar.txt 72,138 Text Document
RarExt.dll 129,024 Application Extensio
RarExt64.dll 43,008 Application Extensio
RarExtLoader.exe 44,032 Application
RarFiles.lst 1,088 LST File
rarnew.dat 20 GOM Media files(.da
ReadMe.txt 1,687 Text Document
TechNote.txt 9,232 Text Document
Uninstall.exe 99,840 Application
Uninstall.lst 639 LST File
UnRAR.exe 203,776 Application
UnrarSrc.txt 90 Text Document 5/12/2005 6:02...
WhatsNew.txt 11,616 Text Document 9/20/2007 6:33...
About WinRAR
WinRAR
WinRAR 3.71
Copyright © 1993-2007 by
Alexander Roshal
40 days trial copy
OK
Total 1 folder and 2,379,470 bytes in 24 files

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 78

Extraction path and options dialog: advanced options

You may specify the following options in this dialog:

File time

Set modification/creation/last access time

Windows file systems keep three different time fields for each file: last modification, creation and last access. By default WinRAR stores only last modification time, but using file time options in Time part of archiving dialog you may enable storing of creation and last access time. Even if a desired time field is present in archive, it is also necessary to set the corresponding time option in the extraction dialog to restore it on extraction. All these options are supported only by RAR archives, ZIP format supports only "Set modification time", for other formats WinRAR always restores only the modification time.

Attributes

Clear attribute "Archive"

Clear attribute "Archive" on the extracted files. This option is designed for backup purposes.

Set file security

This option has meaning only for NTFS file system and allows to restore file owner, group, access control and audit information if it was previously saved in the archive. It is necessary to specify "Save file security data" option in Advanced part of archiving dialog to preserve security information when creating an archive. You need to have necessary privileges in order to use this facility. Processing of security data may decrease the speed of archiving operation, so set this option only if you understand its meaning and really need it, in most cases security processing is not required for home users.

This feature is supported only by RAR archives.

Set attribute "Compressed"

This option allows to restore NTFS "Compressed" attribute when extracting files. WinRAR saves "Compressed" file attributes when creating an archive, but does not restore them unless this option is specified.

This feature is supported only by RAR archives.

File paths

Extract full paths

Usually this mode is most appropriate. WinRAR unpacks contents of archive including the path information into the destination folder.

Do not extract pathnames

If the option is set, selected files from the root archive folder and from selected subfolders will be extracted into the destination folder. The path information is ignored.

Extract absolute paths

If archive was created using "Store full paths including the drive letter" mode selected in Files part of archiving dialog and you set "Extract absolute paths" option, WinRAR will create unpacked files in their original folders and disks. Be careful, do not set this option unless you are completely sure that archive does not contain malicious files. You may read more about potential benefits and dangers of "Extract absolute paths" mode in the description of -ep3 switch, which is the command line equivalent of this WinRAR option. This feature is supported only by RAR and ZIP archives.

Delete archive

Never

Do not delete an unpacked archive.

Ask for confirmation

Ask for user confirmation before deleting an unpacked archive.

Always

Delete an unpacked archive without a confirmation.

WinRAR deletes an archive only if it had been unpacked without errors and if all archived files were selected to unpack. If you unpack a multivolume archive, all its volumes will be deleted.

Note that you can save the default state of this option with "Save settings" button in General page of extraction dialog and it will also affect command line and context menu extraction commands. Use "Ask for confirmation" and especially "Always" mode with care. Deleting an unpacked archive can cause data loss if used improperly.

Miscellaneous

Check authenticity information

Check the information about creator of RAR archive added by "Put authenticity verification" option in General part of archiving dialog. Turning this option off can save some time when unpacking large RAR archives.

Wait if other WinRAR copies are active

Wait if other WinRAR copies are creating, modifying or unpacking an archive and start the operation only when other WinRAR tasks are complete. If you are going to perform several archiving or decompressing tasks, such queued execution can help to reduce amount of disk seeks and improve overall performance.

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 79

Archive name and parameters dialog: time options

High precision modification time

If this option is disabled, file modification time is stored with two seconds precision. It is enough for FAT file system, but insufficient for NTFS. Enabling this option may slightly increase the archive size, up to 5 bytes per each archived file, but it allows WinRAR to preserve the file time with 0.0000001 second precision. WinRAR automatically turns off the high precision mode if it is not supported by source file system, so it will not increase the archive size when compressing FAT disks. Supported only by RAR archives.

"Store creation time" and "Store last access time"

Windows file systems keep three different time fields for each file: last modification, creation and last access. By default WinRAR stores only last modification time, but using these options you may enable storing of creation and last access time. It may be useful for backups. Supported only by RAR archives.

Note that you need to enable "Set creation time" and "Set last access time" options in the "Advanced" part of extraction dialog to restore these times when extracting files.

Include files: of any time / older than / newer than / modified before / modified after

The default value of this option is "any time", so WinRAR will archive all selected files regardless of their time stamp. But changing it to "older than" or "newer than" you may force WinRAR to archive only those files which are older or newer than a specified number of days, hours and minutes. For example, it can be useful if you wish to archive only files modified in last 3 days. Using "modified before" or "modified after" you may choose only those files which are older or newer that the concrete specified date.

Set archive time to: current system time / original archive time / latest file time

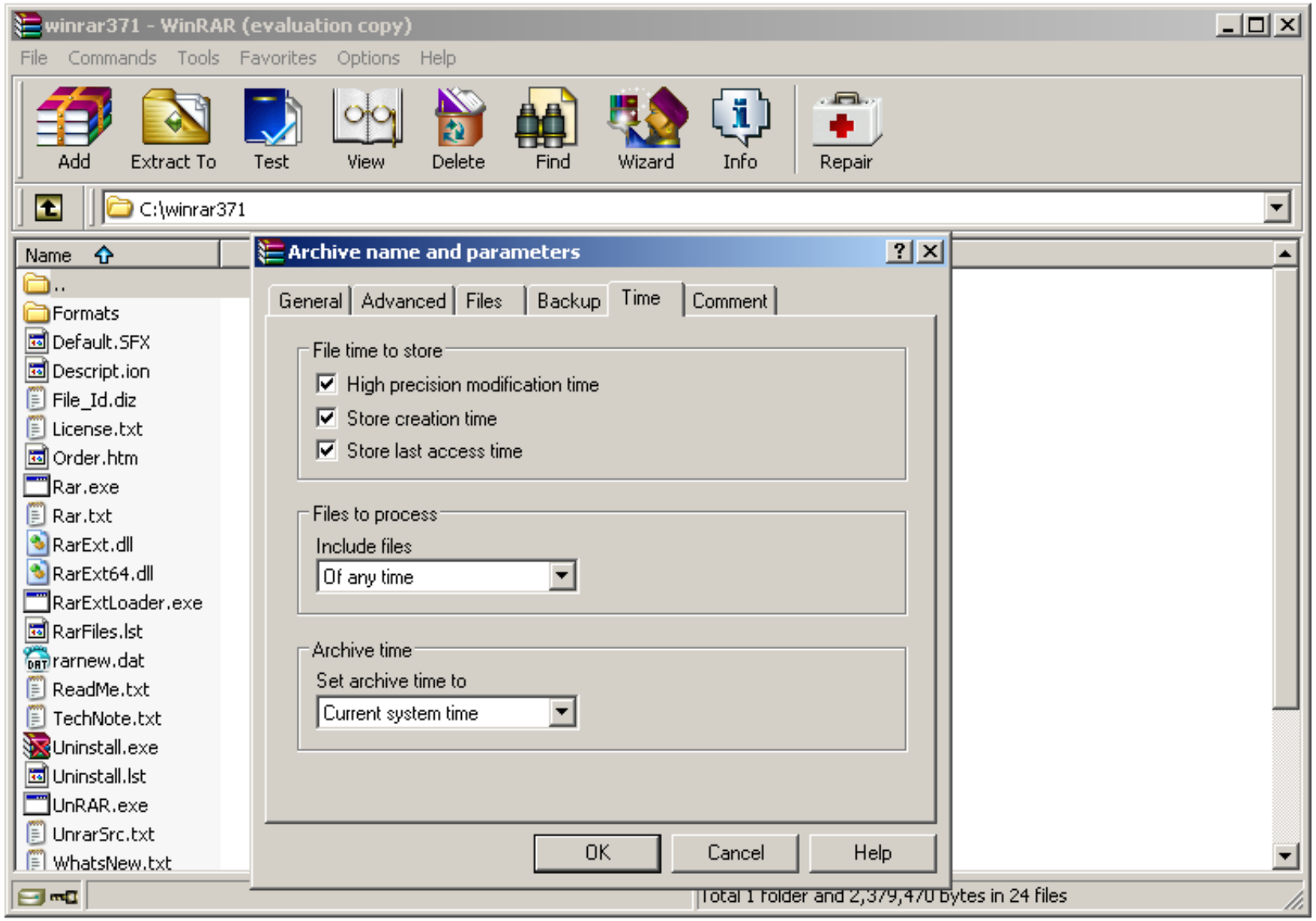
The default value of this option is the "current system time", so every new or modified archive receives the current system time. But you also may set this option to the "original archive time" to prevent the archive time from changing or to the "latest file time" to set the time of a changed archive to the time of the newest file in the archive.

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 80

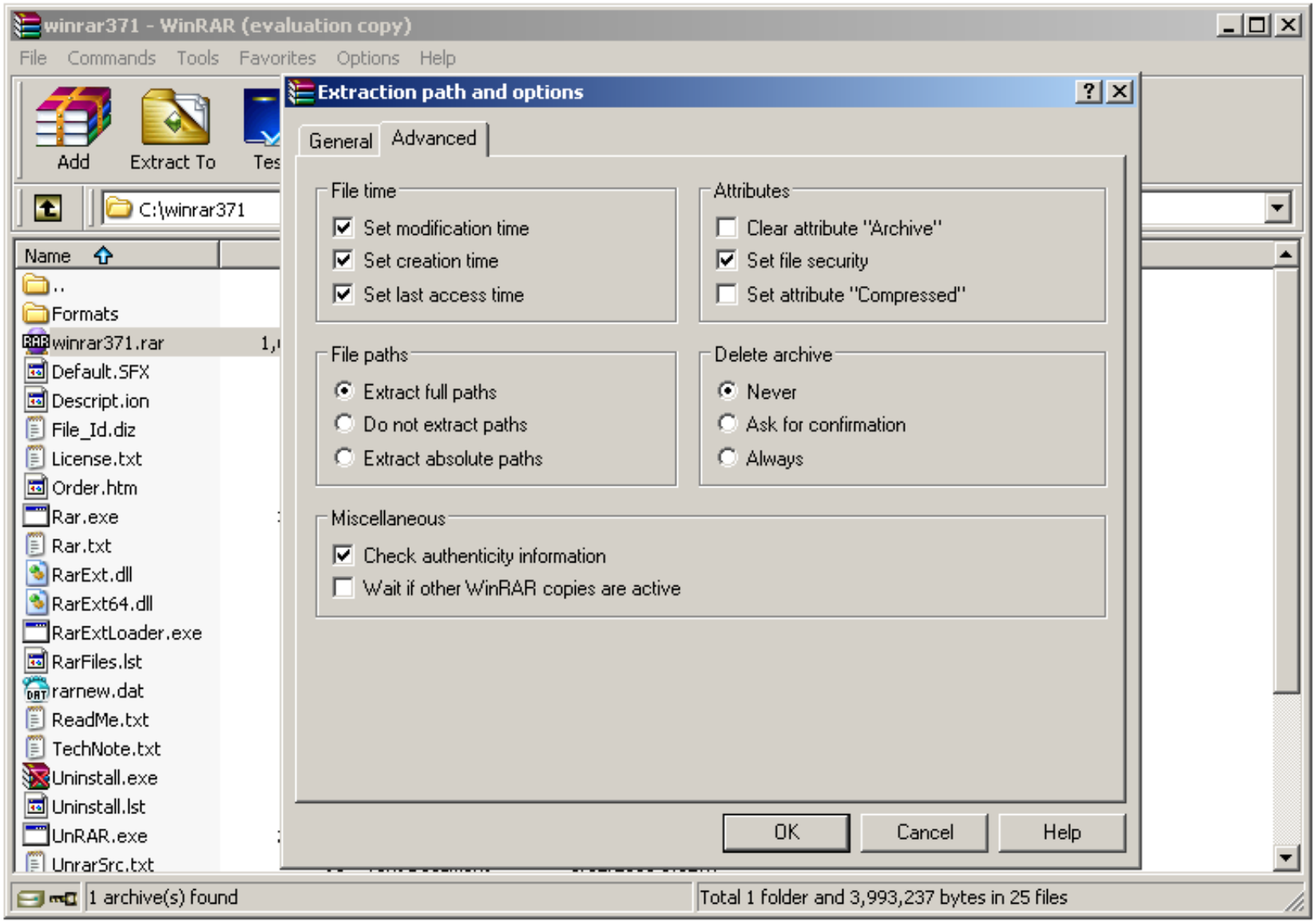

winrar371 - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info Repair
C:\winrar371
Name
..
Formats
Default.SFX
Descript.ion
File_Id.diz
License.txt
Order.htm
Rar.exe
Rar.txt
RarExt.dll
RarExt64.dll
RarExtLoader.exe
RarFiles.lst
rarnew.dat
ReadMe.txt
TechNote.txt
Uninstall.exe
Uninstall.lst
UnRAR.exe
UnrarSrc.txt
WhatsNew.txt
Archive name and parameters
General Advanced Files Backup Time Comment
File time to store
High precision modification time
Store creation time
Store last access time
Files to process
Include files
Of any time
Archive time
Set archive time to
Current system time
OK Cancel Help
Total 1 folder and 2,379,470 bytes in 24 files

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 81

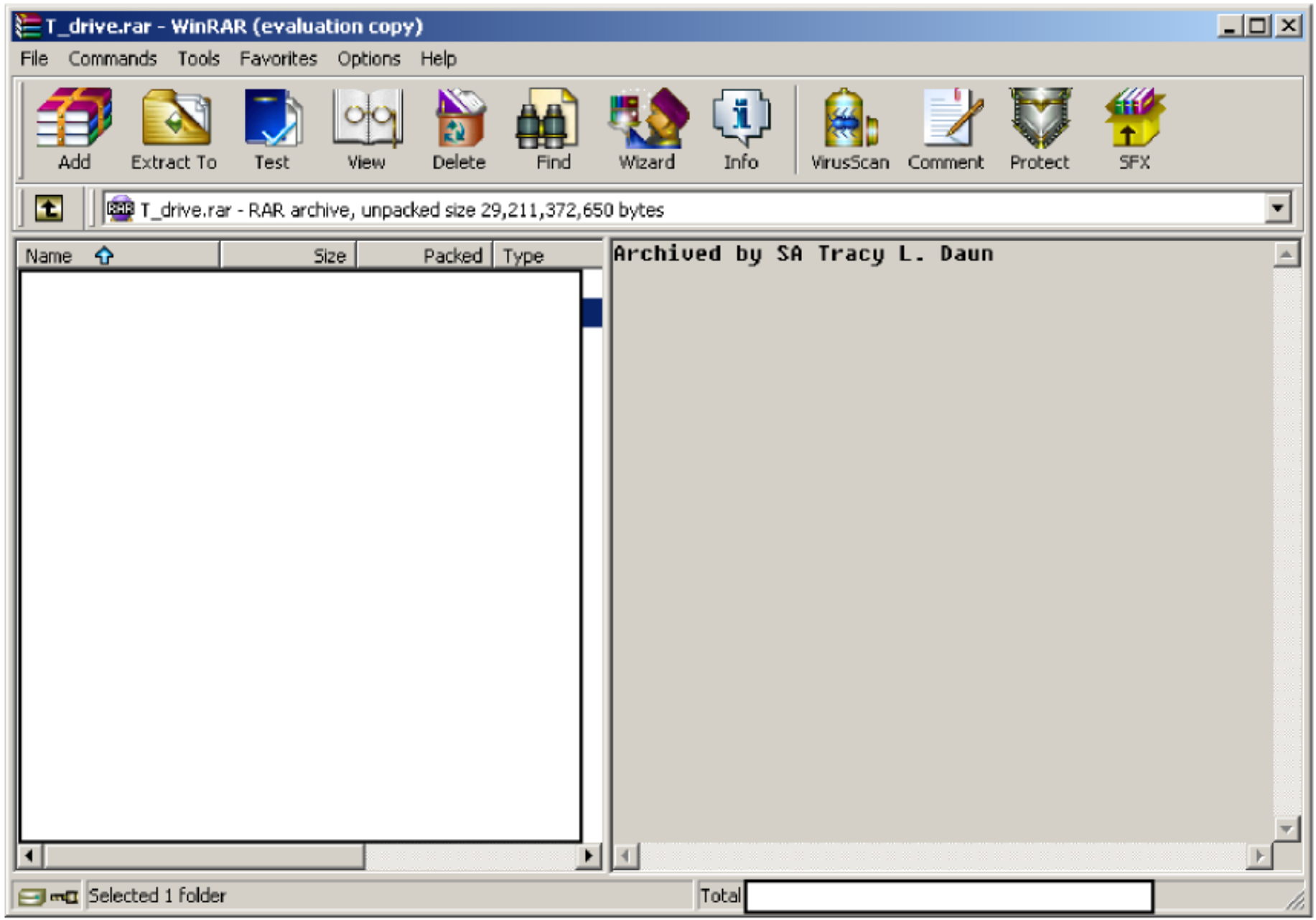

winrar371 - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
C:\winrar371
Name
..
Formats
winrar371.rar
Default.SFX
Descript.ion
File_Id.diz
License.txt
Order.htm
Rar.exe
Rar.txt
RarExt.dll
RarExt64.dll
RarExtLoader.exe
RarFiles.lst
rarnew.dat
ReadMe.txt
TechNote.txt
Uninstall.exe
Uninstall.lst
UnRAR.exe
UnrarSrc.txt
Extraction path and options
General
Advanced
File time
Set modification time
Set creation time
Set last access time
Attributes
Clear attribute "Archive"
Set file security
Set attribute "Compressed"
File paths
Extract full paths
Do not extract paths
Extract absolute paths
Delete archive
Never
Ask for confirmation
Always
Miscellaneous
Check authenticity information
Wait if other WinRAR copies are active
OK
Cancel
Help
1 archive(s) found
Total 1 folder and 3,993,237 bytes in 25 files

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 82

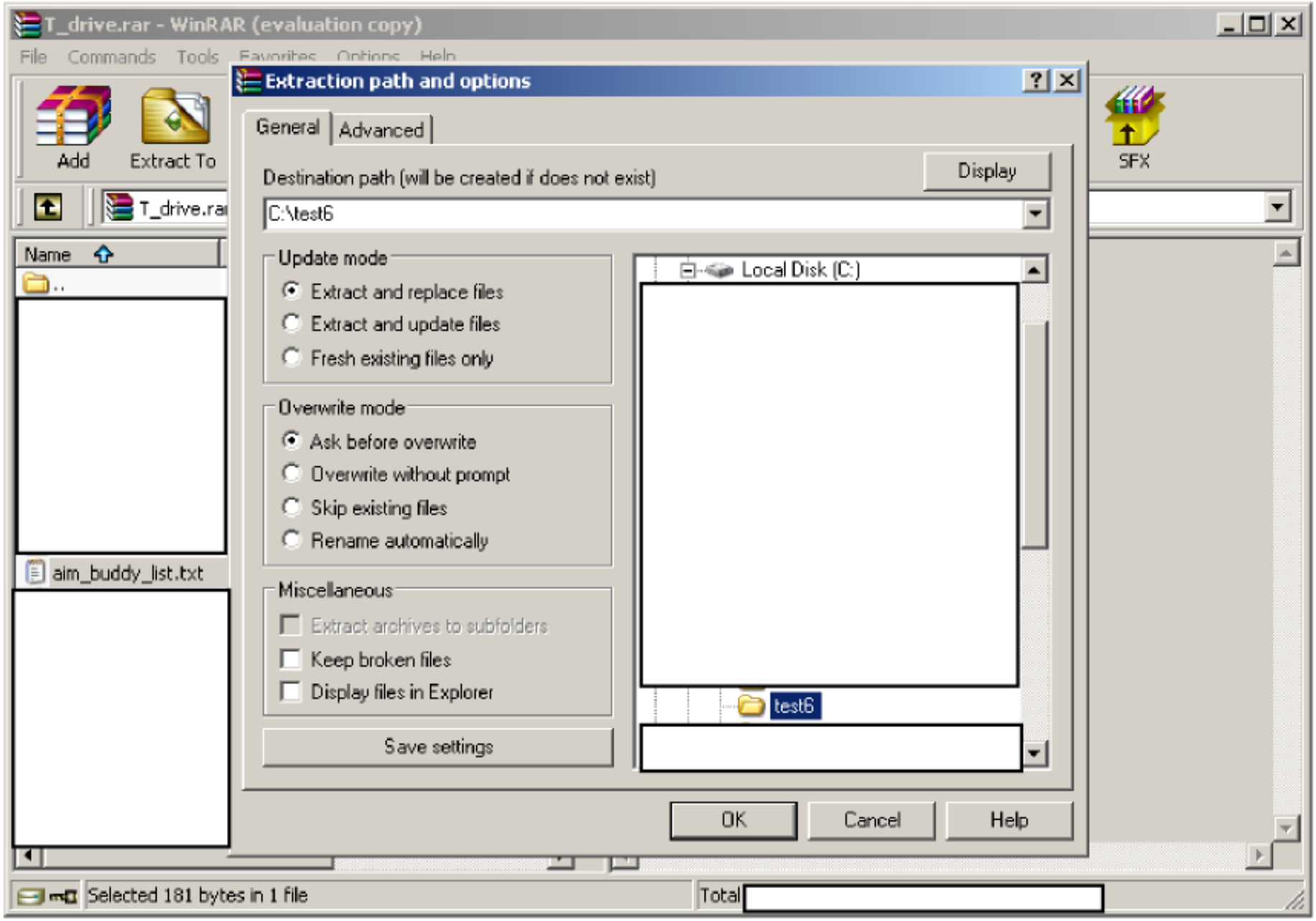

T_drive.rar - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
VirusScan
Comment
Protect
SFX
T_drive.rar - RAR archive, unpacked size 29,211,372,650 bytes
Name
Size
Packed
Type
Archived by SA Tracy L. Daun
Selected 1 folder
Total

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 83

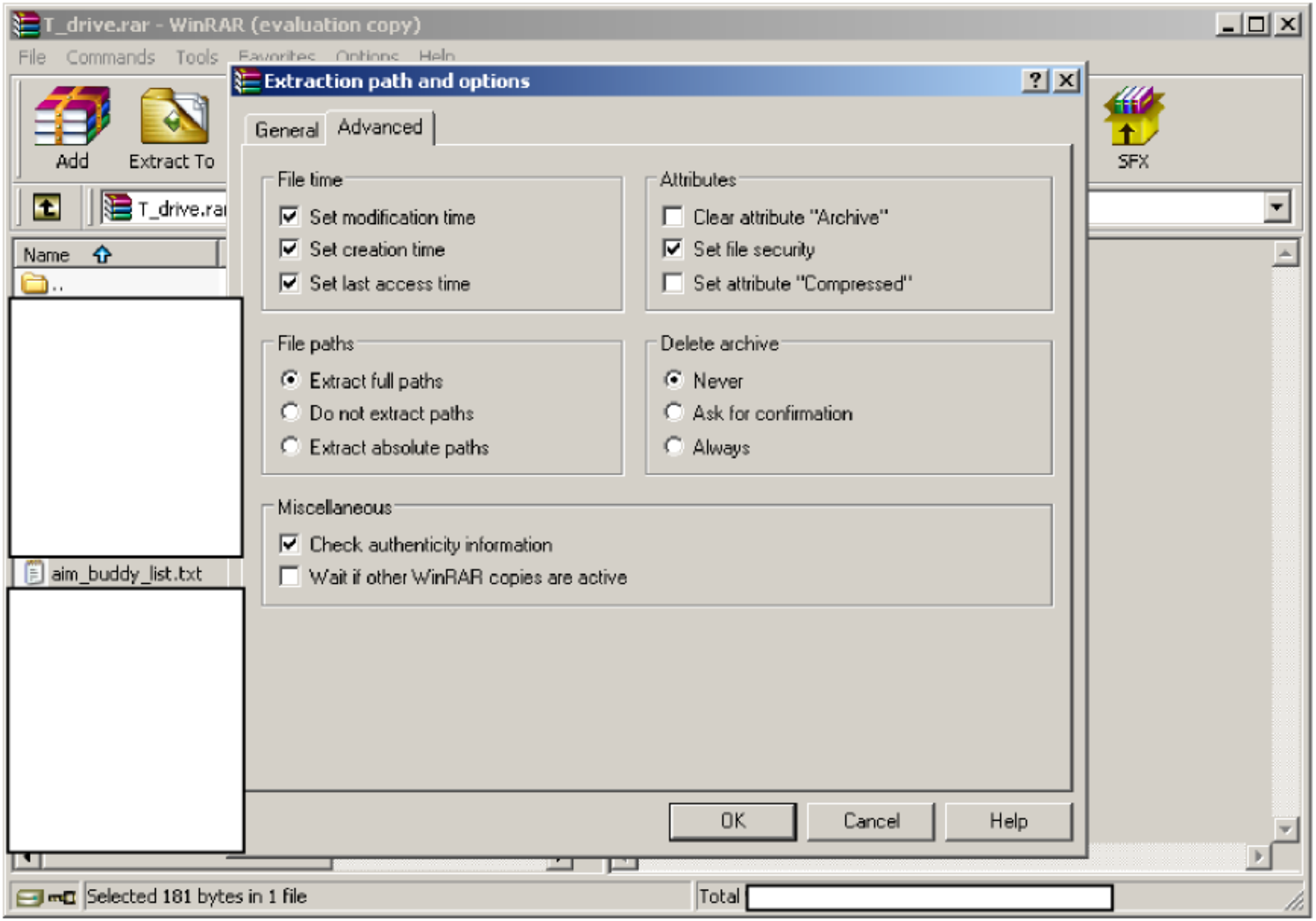

T_drive.rar - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
SFX
T_drive.ra
Name
..
aim_buddy_list.txt
Extraction path and options
General
Advanced
Destination path (will be created if does not exist)
Display
C:\test6
Update mode
Extract and replace files
Extract and update files
Fresh existing files only
Overwrite mode
Ask before overwrite
Overwrite without prompt
Skip existing files
Rename automatically
Miscellaneous
Extract archives to subfolders
Keep broken files
Display files in Explorer
Save settings
Local Disk (C:)
test6
OK
Cancel
Help
Selected 181 bytes in 1 file
Total

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 84

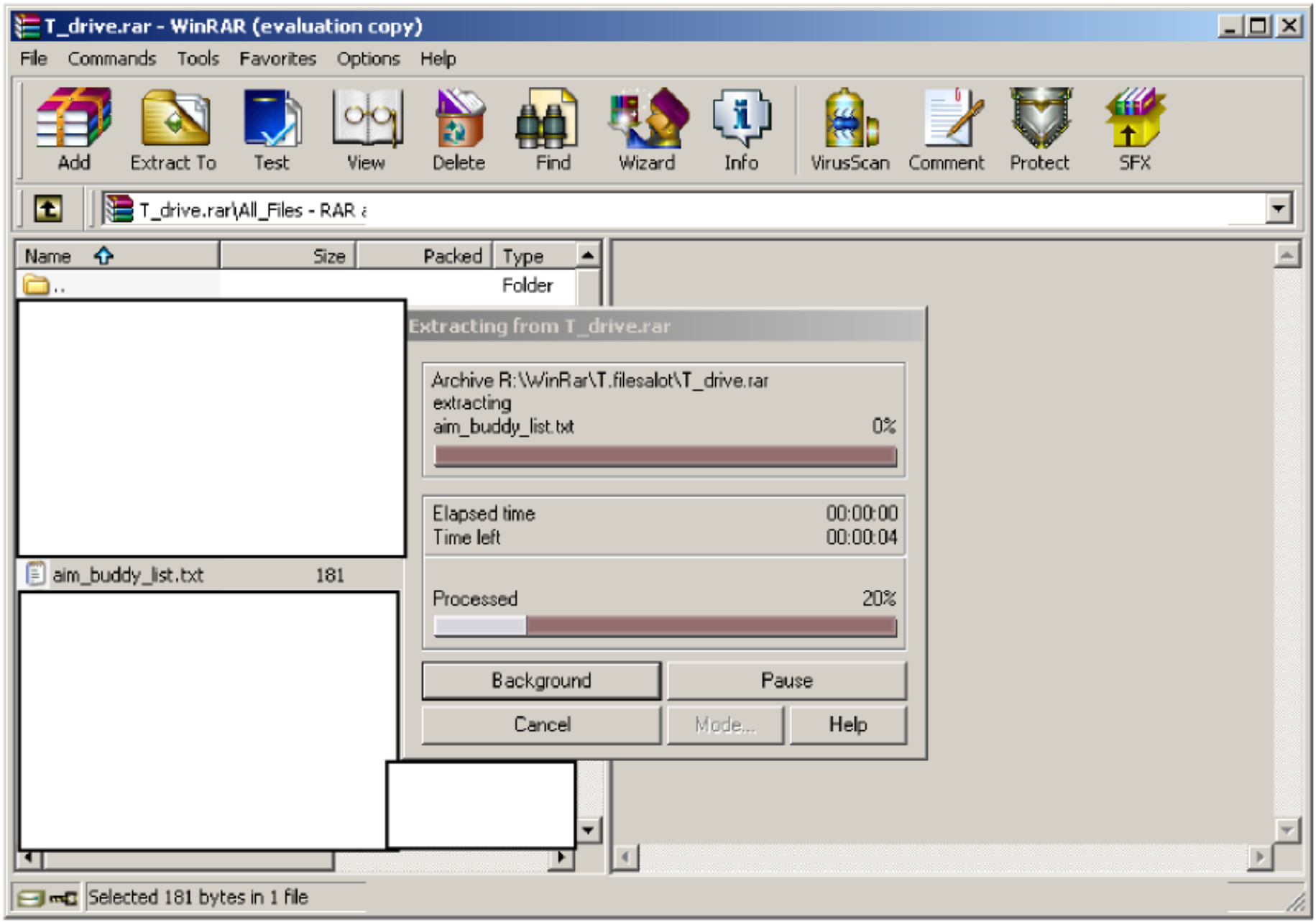

T_drive.rar - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
SFX
T_drive.rar
Name
..
aim_buddy_list.txt
Extraction path and options
General
Advanced
File time
Set modification time
Set creation time
Set last access time
Attributes
Clear attribute "Archive"
Set file security
Set attribute "Compressed"
File paths
Extract full paths
Do not extract paths
Extract absolute paths
Delete archive
Never
Ask for confirmation
Always
Miscellaneous
Check authenticity information
Wait if other WinRAR copies are active
OK
Cancel
Help
Selected 181 bytes in 1 file
Total

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 85

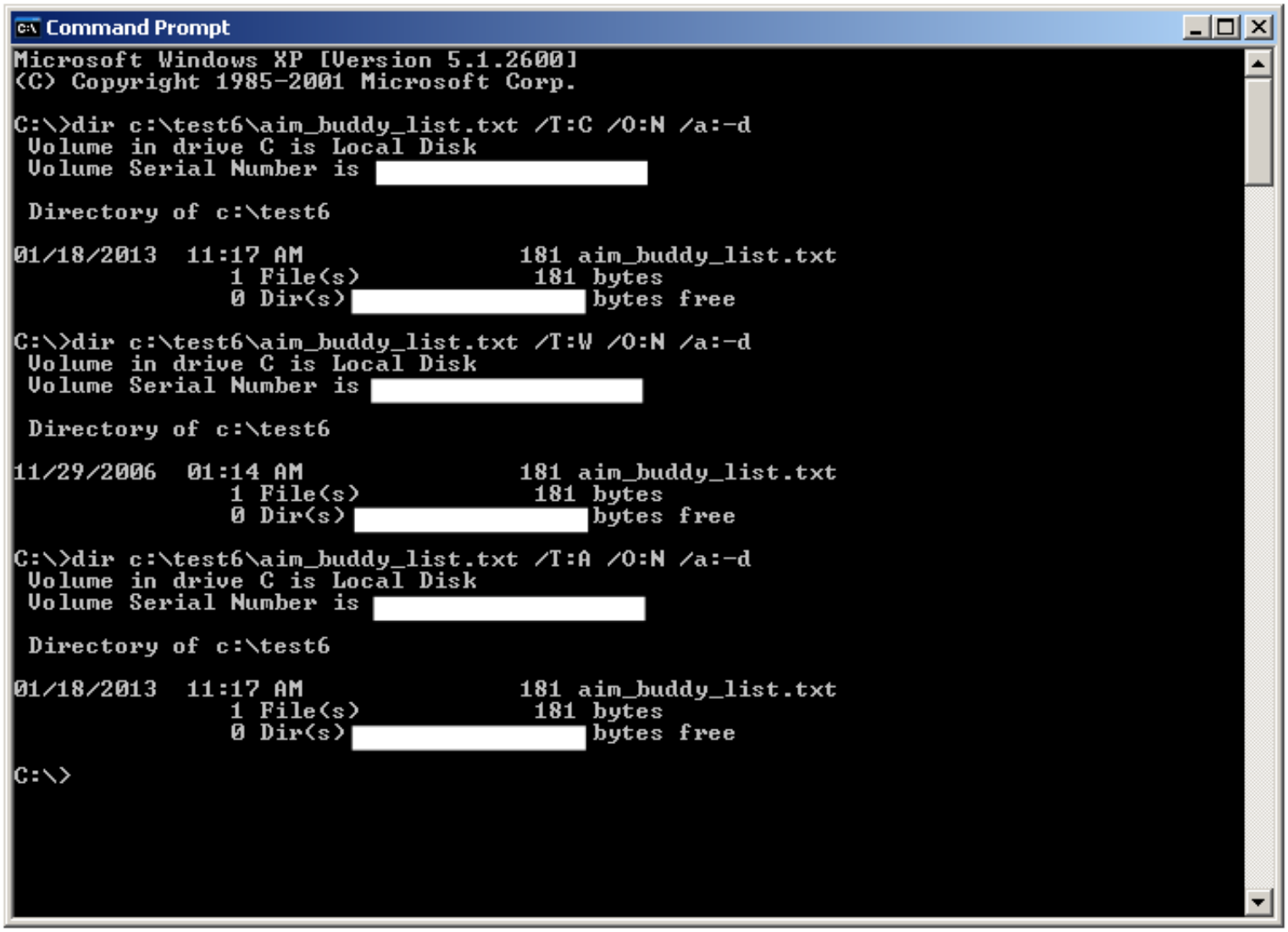

T_drive.rar - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment Protect SFX
T_drive.rar\All_Files - RAR
Name Size Packed Type
Folder
Extracting from T_drive.rar
Archive R:\WinRar\T.filesalot\T_drive.rar
extracting
aim_buddy_list.txt 0%
Elapsed time 00:00:00
Time left 00:00:04
aim_buddy_list.txt 181
Processed 20%
Background Pause
Cancel Mode... Help
Selected 181 bytes in 1 file

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 86


Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\>dir c:\test6\aim_buddy_list.txt /T:C /O:N /a:-d
Volume in drive C is Local Disk
Volume Serial Number is
Directory of c:\test6
01/18/2013 11:17 AM 181 aim_buddy_list.txt
1 File(s) 181 bytes
0 Dir(s) bytes free
C:\>dir c:\test6\aim_buddy_list.txt /T:W /O:N /a:-d
Volume in drive C is Local Disk
Volume Serial Number is
Directory of c:\test6
11/29/2006 01:14 AM 181 aim_buddy_list.txt
1 File(s) 181 bytes
0 Dir(s) bytes free
C:\>dir c:\test6\aim_buddy_list.txt /T:A /O:N /a:-d
Volume in drive C is Local Disk
Volume Serial Number is
Directory of c:\test6
01/18/2013 11:17 AM 181 aim_buddy_list.txt
1 File(s) 181 bytes
0 Dir(s) bytes free
C:\>

DECLARATION UNDER PENALTY OF PERJURY

File modified dates are the most accurate Microsoft Windows NTFS file property to period a file

By Daniel Rigmaiden

ATTACHMENT 87

Analysis Report - T_drive.rar

Analysis Report - T_drive.rar

INVESTIGATION NAME: DANIEL RIGMAIDEN
INVESTIGATION NUMBER: 1000220515

FORENSIC EXAMINER: SA-CIS Tracy L. Daun, IRS-CI

DESCRIPTION OF EVIDENCE: Files archived out using WinRAR
LOCATION OF EVIDENCE: "T" Drive of IBM ThinkPad found on and unsecured at the execution of the search warrant.
ADDRESS: 431 El Camino Real, Apt 1122, Santa Clara CA 95050
IMAGE ID: \T.filesalot\T_drive.rar
IMAGE HASHES:
MD5: \T.filesalot\T_drive.md5

The use of files from this report are subject to the parameters of the Search Warrant and associated Attachment B.

****
****
Examiner Note:

The file path of each file was prefaced with "Daniel Rigmaiden\Single Fingles\Extracted Files\" by the examiner and the forensic software. This portion of the file path did not exist on the original computer.

Files have been extracted with the original file path into its respective folder.

****
****

Analysis Report - T_drive.rar

File Ext:
File Type:
Full Path:
File Created: 05/20/09 02:42:08PM
Last Written: 04/01/08 03:18:21AM
Last Accessed: 08/14/09 12:56:17PM
Is Deleted: No
Hash Value:

Name:
File Ext:
File Type:
Full Path:
File Created: 05/20/09 02:42:24PM
Last Written: 07/17/08 06:29:40PM
Last Accessed: 08/14/09 12:56:17PM
Is Deleted: No
Hash Value:

Name:
File Ext:
File Type:
Full Path:
File Created: 05/20/09 02:43:17PM
Last Written: 04/06/08 05:50:27PM
Last Accessed: 08/14/09 12:56:17PM
Is Deleted: No
Hash Value:

Name:
File Ext:
File Type:
Full Path:
File Created: 05/20/09 02:43:17PM
Last Written: 05/10/08 06:25:46PM
Last Accessed: 08/14/09 12:56:17PM
Is Deleted: No
Hash Value:

****
****

Information Regarding any Online Identity with Online Communications Service (15)

Name: aim_buddy_list.txt
File Ext: txt
File Type: Text
Full Path: \aim_buddy_list.txt
File Created: 05/20/09 02:34:04PM
Last Written: 11/29/06 01:14:12AM
Last Accessed: 08/14/09 12:56:17PM
Is Deleted: No