

# Finding Evidence of Antedating in Digital Investigations

Svein Yngvar Willassen

**Abstract**— Finding evidence of antedating is an important goal in many digital investigations. This paper explores how causality can expose antedating by investigating storage systems for causality and correlate causality with stored timestamps. Causality is determined in two different system types; storage systems using sequence numbers and storage systems using the first-fit allocation strategy. Causality found in these systems was used to implement a timestamp consistency checker for the NTFS file system. The implementation was then tested in an experiment, in which four subjects were asked to antedate a document on a given computer in such a way that the antedating could not be determined by an investigator. The results from this experiment show that the implemented consistency checker can be used to expose antedating. Investigators can use this method to find evidence of antedating to be presented to fact-finders in real cases.

**Index Terms**— digital investigation, antedating, timestamps, causality

## I. INTRODUCTION

Antedating is the creation of files, documents and other material with date- or timestamps set to another date than the date the material was created. Exposing antedating is a common goal in digital investigation, either because the matters under investigation involves documents produced with digital computers, or because the timing of the production of information stored on a computer is otherwise important. The typical digital investigation involving antedating is investigations of financial crimes or other matters where the date of production of a document has legal implications. In these matters, the goal of the digital investigation is often to find out if the document was really produced at the date printed on the document, or if it could have been produced at a later date.

When typing a document in a word processor, it is easy to change the written date. This would not change the timestamps stored on the file system when the document is stored. It is however possible to antedate these timestamps too, by altering the computer system clock to represent a different date than

the current. If the system clock is altered before the document is produced, the timestamps associated with the produced document will be set to the date the system clock was adjusted to. With this procedure, it will not be possible to determine that the document was antedated with previous digital investigation methods.

In this work, causality in digital systems will be used to determine if particular timestamps have been antedated or not. Causality defines which events are necessary for others to occur. In a digital investigation, the medium to be investigated is a storage system storing digital data. In such a system, the events of storage of specific items can be causally related to the events of storage of other items. Since the stored data may contain timestamps, the causal relation can be used to test the consistency of the timestamps, and thereby expose antedating.

## II. RELATED WORK

Being recognized as a research challenge, the problem of timestamp interpretation in digital investigation has been studied by a few researchers during recent years. Schatz et al suggests mitigating the problem by correlating the timestamps in web cache stored on the computer with records obtained from the web servers. [1] Weil and Boyd et al suggest similar correlation methods, by using timestamps stored on the investigated computer coming from other clocks, such as timestamps in dynamically generated web pages. [2, 3] Such methods would provide correlation for the period for which cached data exist on the investigated computer only. Correlation with server records is only possible when such records actually exist, and the investigator has legal access to them.

Stevens studied clocks and described a clock model where each clock is described as the clock it was originally derived from plus the sum of all adjustments, errors and drift. [4] The clock model described by Stevens was refined by Buchholz, in the formalization of a clock model as the sum of clock drift and adjustments. [5] These models are versatile and provide good tools for event reconstruction in cases where clock adjustments, error and drift are known or measurable. They do not however by themselves assist in the identification of clock adjustment, error or drift.

Manuscript received October 11, 2007.

S. Y. Willassen is with the Department of Telematics, Norwegian University of Science and Technology, O.S. Bragstads plass 2B, 7491 Trondheim, Norway (phone: +47 92449678; email: svein@willassen.no).

In previous works, we have defined a formalism for clock hypotheses and consistency testing with causality between timestamped events. [6] This formalism has been utilized to develop models for the updating of timestamp values in digital systems. [7] In this work, we use the happened-before relation defined in [6] to analyze specific properties of file systems. These properties are then used to find evidence of antedating.

### III. SEQUENCE NUMBER CAUSALITY

Sequence numbers is a feature occurring in many digital systems, such as file systems and networks. By using a sequence number, the systems designer ensures that sequence numbered entities can be ordered in the correct order and be distinguished from each other. Sequence numbers are usually implemented by a counter increasing whenever a new sequence numbered entity is produced and associating a copy of the value of the counter (the sequence number) with that entity. It is useful to distinguish between *wrapping sequence numbers* and *strictly increasing sequence numbers*. Wrapping sequence numbers have a limited span of values. When the highest value is reached, the counter wraps and starts at the lowest value. A strictly increasing sequence number on the other hand is a sequence number that does not wrap. In theory a strictly increasing sequence number would have to be able to represent infinite numbers. In practice however, a sequence number can be viewed as strictly increasing as long as the number of values that can be represented is large enough to produce strictly increasing numbers over a significant time span, for example the lifetime of a computer.

When investigating a system with sequence numbered entities, the distinction between wrapping sequence numbers and strictly increasing sequence numbers is important. With a wrapping sequence number, one would not be able to know how many times the counter had wrapped when the sequence number was generated. When correlating two entities with sequence numbers, one would therefore not be able to determine if one of the entities was produced before the other. In a system with strictly increasing sequence numbers on the other hand, one can be sure that the entity with the highest sequence number has been produced after the entity with the lowest sequence number. In such a system, each production of a sequence numbered entity is causally dependant on the production of every other sequence numbered entity with lower sequence number.

Many file systems contain File System Journals with sequence numbered entities. For example, in NTFS, journal file transactions are labelled with a 64-bit number (so called Logical Sequence Number - LSN) that increases throughout the lifetime of the file system. The proper functioning of the journaling feature in NTFS depends on this number being strictly increasing. [8]

In these systems, it is possible to find causal connections by analyzing journal files. The amount of information that can be derived from the journal file itself is however limited. Since every write to a file produces a journal file entry and the journal file has limited size, old entries will quickly be overwritten. Some file systems, such as NTFS, store the journal file sequence number (the LSN in NTFS) in the file metadata entry. If the journal file sequence number is strictly increasing, the generating events are causally connected. Causal connections then exist between the events of the last change of the file entry on all files stored on the file system.

Example 1. Consider the following set of allocated file entries from an NTFS Master File Table. Let  $e_i$  be the last update of the current data in entry  $i$ .

Entry 45 log file sequence number 432627  
 Entry 46 log file sequence number 186345  
 Entry 47 log file sequence number 735294  
 Entry 48 log file sequence number 165093  
 Entry 49 log file sequence number 878121  
 Entry 50 log file sequence number 782427  
 Entry 51 log file sequence number 561987

Since logical sequence numbers in the journal file (log file) are allocated sequentially, we can obtain the causal ordering of the last update events by sorting the file entries by their log file sequence number:  $e_{48} \rightarrow e_{46} \rightarrow e_{45} \rightarrow e_{51} \rightarrow e_{50} \rightarrow e_{47} \rightarrow e_{49}$ .

### IV. ALLOCATION SEQUENCE CAUSALITY

A first-fit allocation storage is a system in which each new data item is stored in the first available storage location. Deleting data items is allowed and can be done at any time after the data item has been stored in a storage location. After a data item has been deleted, it may be overwritten by new data at any time. It may be possible to recover deleted data, but it is not possible to recover data that has been overwritten.

A special form of first-fit storage is first-fit storage *with generation-markers*. In this storage, it is possible to identify which generation the data in each storage location belongs to. The generation of a storage location is an identification of how many times that data in that storage location has been overwritten. Fig. 1 shows a possible allocation sequence with generation markers.

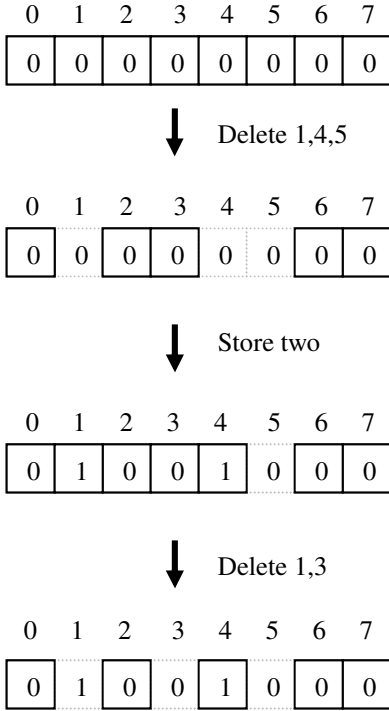


Fig. 1 Allocation sequence with generation markers

In a system with generation markers, there is a causal connection between every pair of consecutive generations at each storage location. The storage of data in the  $i$ -th storage location generation  $g$  can only commence if the data present in the  $i$ -th storage location generation  $(g-1)$  has already been stored and deleted. Therefore, for every storage location  $i$ , the storage of data in generation  $(g-1)$  happened-before the storage of data in generation  $g$ . Generally; due to the transitivity property of the happened-before relation, the event of storing data in a storage location is causally dependant on the storing of all previous generations in that storage location.

Let  $s_{i,g}$  be the  $i$ -th storage location generation  $g$ . Let  $e_{s_{i,g}}$  be the event of storing data in the  $i$ -th storage location at the  $g$ -th generation. Then for all  $i$  and  $g$ ;  $e_{s_{i,g-1}} \rightarrow e_{s_{i,g}}$ . Due to the transitivity of  $\rightarrow$ , for all generations  $g$ ,

$$\forall h < g (e_{s_{i,h}} \rightarrow e_{s_{i,g}}) \tag{1}$$

We now consider the storage of data in storage locations with generation  $g = 0$ . When  $g = 0$ , there cannot exist any storage location which has been deleted and then overwritten with another data item, because this would have increased the generation number above 0. The only place where new storage locations can be allocated with generation number 0 is at the end of the storage.

Let  $s_{i,0}$  be the  $i$ -th storage location in a first-fit storage, generation 0. Let  $e_{s_{i,0}}$  be the event of storing data at generation

0 in the  $i$ -th storage location. Then, for all  $i$ ,

$$\forall j < i (e_{s_{j,0}} \rightarrow e_{s_{i,0}}) \tag{2}$$

Two different types of causal event sets have now been defined from the first available storage with generation markers; the causality between storage of storage locations with  $g = 0$ , and causality between storage of increasing generations at each storage location. These sets intersect. Each causality set for increasing generations start at  $g = 0$ . Each such element is also part of the  $g = 0$  causal set. With these two types of causal connections in the first available storage with generation markers, a causal connection relating all storage locations in the set has been found.

Example 2. Consider the storage location set in Fig. 2. In the figure, the storage locations are shown horizontally, and generations vertically. Deleted data are shown in lighter colour. For each storage location, the topmost item is always the current data stored in the location. There are now causal connections, where each generation within a storage location happened-before the next generation, and each storage location at generation 0 happened-before the next location at generation 0. The resulting Direct Acyclic Graph of the creation events of the existing storage locations is shown in Fig. 3.

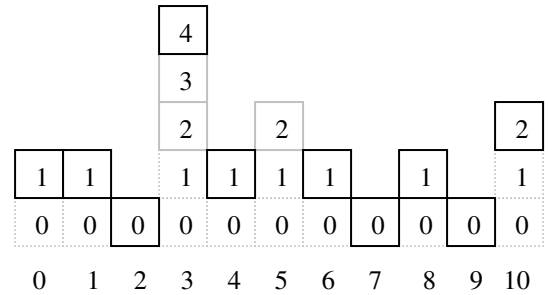


Fig. 2 Storage locations with generations

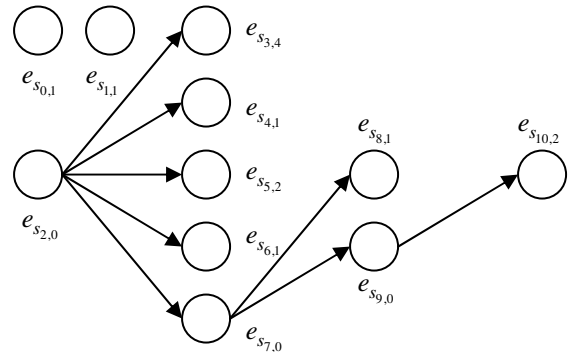


Fig. 3 Causality of existing (non-overwritten) storage locations

Example 2 shows how the happened-before relation imposes a strict partial order on the set of storing events. The partial order follows from the properties of the happened-before relation; it is irreflexive, transitive and asymmetric. [6] The relation does not however relate all elements of the set of

storing events. Elements with  $g > 0$  are not related to other elements with  $g > 0$ , as shown in Fig. 3. A total order is therefore not imposed on the set of storing events.

The Master File Table of the NTFS file system is a first-fit storage with generation markers. Each file stored in NTFS has its own entry in the Master File Table. Data stored in the MFT entry include the file name, the list of data runs where the file data is stored, timestamps and other data such as information on whether the data is compressed or encrypted using the compression and encryption features in the file system. Allocation of file entries within the MFT occurs on a first-fit basis. [9] Each file entry contains a generation marker, termed *entry sequence number*, which identifies the generation of that entry. This number is increased whenever the file entry is reused. Thus, causal connections exist between file entries in the Master File Table of the NTFS file system, following the reasoning above.

## V. IMPLEMENTATION

An implementation of the above reasoning above was made in a program named TimeStampLogic. The program uses the utilities in a modified version of the Sleuthkit [10] to find file instances in an NTFS file system image. It then parses the output of these utilities and produces internal representations of file instances, which can then be analyzed. The analysis consists of two modules; SequenceChecker and LogSequenceChecker. SequenceChecker tests the generation causality of the file entries, by the reasoning in Section IV. All entries are compared with the last preceding entry with the base generation sequence number. Since the allocation of Master File Table entries are done in a first-fit fashion, all entries have been stored at a later time than the last preceding entry with the base generation sequence number. LogSequenceChecker tests the causality of the updating of file entries based on the log file sequence numbers stored in the Master File Table file entries, by the reasoning in Section III. The instances are sorted by the associated Log File Sequence Number (LSN) and then printed in sorted order.

By inspecting the timestamps associated with each file entry in comparison with the two different sequences produced by SequenceChecker and LogSequenceChecker, it can now be determined if the computer clock has been changed.

## VI. DOCUMENT ANTEDATING EXPERIMENT

In order to test the theory and implementation, a document antedating experiment was performed, in which four subjects were asked to antedate a document. A laptop computer was prepared for the experiment. The hard drive of the computer was wiped. The computer was then started and the system clock was adjusted to approximately two and a half years before the time of the experiment with the BIOS setup

program. Then, the Windows XP operating system was installed. After installation, a series of shutdowns, clock forward adjustments and reboots were performed, until the clock was adjusted to civil time in October 2006, when the experiment was performed. The goal of this procedure was to produce data on the hard drive similar to data that would have been produced by real usage of the computer. For each step, the computer was shut down, then started in BIOS setup, where the clock was adjusted forward. The computer was then booted into Windows XP and used for web surfing, downloading files or other normal user activity. After this procedure, the hard drive was copied to an image file on another hard drive using the disk dump utility dd, producing a reference image of the experiment computer.

The experiment computer was then handed to the participating subjects with the following task: "Store a document on this computer in such a way that a person investigating the computer will conclude that the document was produced on May 17<sup>th</sup>, 2006." When each subject returned the computer, the hard drive was copied to an image file on another hard drive for analysis. Then, the experiment image was copied back to the computer before it was handed to the next subject. The subjects participating in the experiment were:

Subject no	Computer experience level
1	Average computer user, using computer every day
2	Law Enforcement Computer Forensic Investigator
3	Inexperienced office user, mostly used to websurfing
4	Advanced computer user with programming experience

Each image was then analyzed using the TimeStampLogic program. Each subject was also interviewed to determine how they had chosen to perform the task.

## VII. RESULTS

In the following, each of the images resulting from imaging the experiment computer after each subject had completed the task is analyzed. The purpose of the analysis is to determine if the document in question has been antedated or not. In a hypothesis based approach to digital investigation (defined by Carrier [11]), this can be formulated as two different hypotheses:

$H_0$ : The document was produced on 17<sup>th</sup> of May.

$H_1$ : The document was produced later than 17<sup>th</sup> of May, but has been antedated to 17<sup>th</sup> of May.

The task for the investigator is then to find evidence supporting or rejecting  $H_0$  and  $H_1$  using TimeStampLogic and other investigative tools.

### A. Subject 1

The subject gave the following information about how the task was completed: *I adjusted the clock on my Mac to May 17<sup>th</sup>. I then produced the document in Microsoft Word on the Mac. When saved on the Mac, I copied the document to my USB stick and inserted it into the PC. I then copied the document from the USB stick to the PC. I believe producing the document on the Mac may have prevented the creation of timestamps inside the Word document.*

When analyzed with TimeStampLogic, the results of this operation did not produce a result significantly different from analysis of the reference image. The introduction of new files when the computer was booted and a new document was copied to it, did not produce any new inconsistencies reported by TimeStampLogic. The document has Modified timestamp on the 17<sup>th</sup> of May, and Created and Accessed timestamps on the date of the experiment. This is consistent with timestamps produced when files are copied to a medium. Other evidence suggesting that the file had been copied to the medium on the date of the experiment was also found, for example a link-file to an external drive, showing that an external drive had been connected to the computer. If the file had been copied from another computer, the Modified timestamp would then be related to the clock of that computer and not the investigated computer. Since no evidence is available to test clock hypotheses for the other computer, there is no evidence to either support or reject a hypothesis that the production of the document actually occurred on 17<sup>th</sup> of May civil time. The analysis is therefore inconclusive in this case. The reasonable investigative response in cases like this is to try to get hold of the computer on which the document was produced and do the same type of analysis on that computer.

In response to the subject's claim that timestamps had not been created inside the Word document, it was examined for timestamps in the metadata. Such timestamps were found, identifying that the document had been created and last changed on May 17<sup>th</sup>. These timestamps would also refer to the clock on the other computer, which will have to be analyzed for evidence.

### B. Subject 2

The subject gave the following information about how the task was completed: *I started the PC and connected it to the Internet. I then downloaded and installed OpenOffice on the PC. I then restarted the computer, went into BIOS and adjusted the date back to May 17<sup>th</sup>. After booting the computer again, I used OpenOffice to create and store the document. I then booted again and adjusted the clock back to current time. I used OpenOffice because I think it doesn't have the same amount of metadata as Microsoft Word. I also think downloading and installing OpenOffice would prevent a proper investigation, since it probably overwrote a lot of deleted data.*

When analyzed with TimeStampLogic, a significant higher number of inconsistencies were reported with both SequenceChecker and LogSequenceChecker. Listing all files on the medium ordered by both the MFT Entry number (SequenceChecker) and Log Sequence Number (LogSequenceChecker), showed several hundred files with Created, Modified and Accessed timestamps on Oct 11<sup>th</sup> in the time period 07:28-07:40 AM. After these (in terms of entry number and log sequence number), approximately 50 files with Created, Modified and Accessed timestamps on May 17<sup>th</sup> time period 07:42-07:48 were listed. All timestamps of the document in question were set to May 17<sup>th</sup> in the period 07:42-07:48.

The timestamps on the document itself were in this case set to May 17<sup>th</sup>, in contrast to Subject 1. There is however evidence in this case supporting H<sub>1</sub>:

- Storing of a significant number of files causally dependant on storing of files occurring on Oct 11<sup>th</sup>, were timestamped May 17<sup>th</sup>, something that is not possible unless the clock has been adjusted, or the timestamps changed.
- When the date changes from Oct 11<sup>th</sup> to May 17<sup>th</sup>, the time of day only moves approximately 2 minutes forward. This indicates that the subject changed the date but did not bother to change the time of day. If the clock adjustment had occurred by an error or some other mysterious event, it is not very likely that it would have ended up at this exact time of day.

The subject's claim that he made the investigation more difficult by installing OpenOffice, does not seem to be correct in the context of using TimeStampLogic to check timestamp consistency. It may be the case that installing a new program would overwrite previously overwritten material, but this does not help, since TimeStampLogic does not rely on the investigator's ability to recover deleted material.

### C. Subject 3

The subject gave the following information about how the task was completed: *I don't know how to manipulate timestamps, so I just went into the control panel and set the date to May 17<sup>th</sup>. Then I used Microsoft Word to produce the document. Then I set the current date again in the control panel.*

On this image, TimeStampLogic produced the same type of results as on the image from Subject 2. Approximately 10 files were listed with Created, Modified and Accessed timestamps on Oct 12<sup>th</sup> from 9:17-9:44 PM. After this (in terms of MFT entry sequence and LSN sequence), approximately 10 files were listed with timestamps at May 17<sup>th</sup> 9:46-9:52 PM. This gives evidence for H<sub>1</sub>, for the same reasons as for Subject 2.

In this case, as opposed to the case of Subject 2, the clock change was done in the operating system. Therefore, the event logs of the system were searched to determine if the clock change had logged a system event. No such event was found. Windows XP has a system logging feature that allow logging of clock change events. This particular event is however logged only if Privileged Use logging is enabled, something it is not by default. [12]

It is interesting to note that both Subject 2 and 3 changed the date without changing the time of day. Both in the BIOS of the experiment computer and in the Windows XP control panel, changing date is done by a separate control than changing time of day, even if they are both related to the same underlying clock. A plausible rationale for not changing the time of day could be that it would then be easier to adjust the clock back to the current time, because one would then not have to resynchronize with an external clock. When asked about this, subject 3 said: *I didn't think about that. I just wanted the correct date on the document. The time of day didn't matter to me. I might have thought about it if the time of day were of any importance, for example if it mattered if I were at work at the time or not.*

#### D. Subject 4

The subject gave the following information about how the task was completed: *I used my own pc for the antedating. I adjusted its clock back to May 17<sup>th</sup>, and produced the document using Microsoft Word. I then copied the document over to the experiment PC using my USB-stick.*

The story of Subject 4 matches the story of Subject 1, and the results of TimeStampLogic were similar. No additional inconsistencies were found, and the results were inconclusive on the question of whether the document was antedated or not. Also in this case, link files pointing to an external medium identified another computer as the likely source for the document.

#### E. Summary

In the document antedating experiment, four subjects were asked to antedate a document in such a way that it could not be determined that the document file was antedated. Two of the subjects performed the antedating in such a way that the methods described in this work could produce evidence supporting the hypothesis that the document was antedated and not produced on the date it was timestamped to. Two of the subjects did the antedating itself on another computer and copied the resulting document to the investigated computer. In this case, it could not be determined that the document was antedated, but it could be determined that the document had been copied from another computer, thus another possible item of evidence was found. It is known from the explanation from

the subjects, that they produced the antedated document on the other computer by adjusting the clock back to May 17<sup>th</sup>, which is the same method used by Subject 2 and 3 on the investigated computer. Investigation of the other computer with the methods described in this work would therefore most likely have produced evidence supporting the hypothesis that the document was antedated.

### VIII. CONCLUDING REMARKS

Causality reasoning can be used to check timestamp evidence for consistency with causal ordering of events. Such reasoning can be used to determine if digital information has been antedated or not. The document antedating experiment has shown that causality reasoning can provide evidence of antedating of computer files in practical situations, where subjects have antedated a file. The described methods can be implemented in existing tools for digital investigation such as the Sleuthkit or EnCase. This would provide investigators with the possibility for time and causality reasoning in real cases.

The antedating methods used by the subjects in the experiment are certainly not the only way to antedate a document. Other possible methods can be conceived:

1. Produce the document at current time, then changing its timestamps by special software. This can be done without introducing the software in question on the investigated computer by removing the medium and perform the change on another computer.
2. Finding another file matching the desired timestamps, then replacing the contents of that file with specialized software.
3. Using the same method as used by the subjects in the experiment. Then use special software that adjusts all timestamps on the medium to match the story. Such software could be called anti-TimeStampLogic.

In the case of conceived method 1, TimeStampLogic would probably report the single file as an inconsistency. In the case of conceived method 2 and 3 however, it is not likely that TimeStampLogic would be able to find any inconsistencies. Producing evidence of antedating in these cases would have to rely on other methods, if possible at all. Thus, clock hypothesis testing methods described here are not perfect methods that cannot be avoided by a crafty antedater.

This possibility of evidence manipulation does not however imply that the described methods are not useful in real investigations. Consider the adversaries in a digital investigation, the *Investigator* and the *Perpetrator*. The Investigator usually possesses knowledge of digital investigation and tools that can comb a digital medium for evidence, including tools for digital imaging and data recovery. The Perpetrator on the other hand is likely to be an

average computer user, and does not know how to protect himself from the scrutiny of a digital investigation or where he should go to obtain the necessary tools. The Investigator also has time on his side. Once a digital medium has been forensically imaged, he has plenty of time to investigate its contents. The Perpetrator on the other hand never knows when the Investigator will turn up to seize his data, if ever. He therefore has to be prepared at all times and run the anti-forensic procedure again and again after every action that would leave incriminating evidence. There is no room for mistakes by the Perpetrator. If he makes a small mistake in his anti-forensic procedure, the evidence may be there waiting to be discovered by the Investigator. The Investigator on the other hand can make a lot of mistakes, as long as he doesn't mess up the original data. He can always start from a fresh image at a later time, should he feel that there is more to find or that current results rely on misinterpretations. All in all, the Investigator has a tremendous advantage over the Perpetrator in digital investigations.

The above can also be viewed in light of Locard's exchange principle, in which it is stated that every physical contact yields exchange of matter so that subsequent forensic investigations can prove that the contact occurred by analyzing the exchanged matters. By using special software that adjusts all timestamps on the medium to match a predefined story, it is likely that a special timestamp pattern specific to that software would be created. It would then be possible for the investigator to produce evidence of the usage of such software. This would be highly undesirable for the perpetrator, since it would create an impression that he had something to hide.

Applying this reasoning to the methods developed in this work, the conclusion must be that there exist methods by which the investigation methods described in the work can be avoided. This is however difficult to do to in such a way that it cannot be detected. Subjects who want to use antedating are not likely to possess the required knowledge to do this. The described methods are therefore adequate for exposing antedating in most real investigations.

#### REFERENCES

- [1] B. Schatz, G. Mohay, and A. Clark, "A correlation method for establishing provenance of timestamps in digital evidence," *Digital Investigation*, vol. 2006:3S, pp. 98-107, 2006.
- [2] M. C. Weil, "Dynamic Time & Date Stamp Analysis," *International Journal of Digital Evidence*, vol. 1:2, 2002.
- [3] C. Boyd and P. Forster, "Time and date issues in forensic computing - a case study," *Digital Investigation*, vol. 2004:1, pp. 18-23, 2004.
- [4] M. W. Stevens, "Unification of relative time frames for digital forensics," *Digital Investigation*, vol. 2004:1, pp. 225-239, 2004.
- [5] F. Buchholz, "An Improved Clock Model for Translating Timestamps," James Madison University, Department of Computer Science JUM-INFOSEC-TR-2007-001, 2007.
- [6] S. Y. Willassen, "Hypothesis based investigation of digital timestamps," in *IFIP WG 11.9 Workshop*, Kyoto, Japan, 2008.
- [7] S. Y. Willassen, "Timestamp evidence correlation by model based clock hypothesis testing," in *E-Forensics 2008*, Adelaide, Australia, 2008.
- [8] M. E. Russinovich and D. A. Solomon, *Microsoft Windows internals : Microsoft Windows Server 2003, Windows XP, and Windows 2000*, 4th ed. Redmond, Washington: Microsoft Press, 2005.
- [9] B. Carrier, *File system forensic analysis*. Upper Saddle River, N.J.: Addison-Wesley, 2005.
- [10] B. Carrier, "Sleuthkit," Available at: [www.sleuthkit.org](http://www.sleuthkit.org).
- [11] B. Carrier, "A hypothesis-based approach to digital forensic investigations," Center for Education and Research in Information Assurance and Security, Purdue University Tech Report 2006-06, 2006.
- [12] University of Delaware Police Computer Forensics Lab, "Time Change Captured in Event Log," <http://128.175.24.251/forensics/timechange.htm> Accessed: Oct 3, 2007