Timestamp evidence correlation by model based clock hypothesis testing

Svein Yngvar Willassen
Department of Telematics, Norwegian University of Science and Technology
O.S. Bragstads plass 2B
7491 Trondheim, Norway
+47 92449678
svein@willassen.no

ABSTRACT
Timestamps play an important role in digital investigations, since they are necessary for the correlation of evidence from different sources, including network tracing. Use of timestamps as evidence can be questionable due to the reference to a clock with unknown adjustment. This work addresses this problem by taking a hypothesis based approach to timestamp investigation. Historical clock values can be formulated as a clock hypothesis. This hypothesis can be tested for consistency with timestamp evidence by constructing a model of actions affecting timestamps in the investigated system. Acceptance of a clock hypothesis with timestamp evidence can justify the hypothesis, and thereby establish when events occurred in civil time. The results can be used to correlate timestamp evidence from different sources, including identifying correct originators during network trace.

Categories and Subject Descriptors
F.4.1 [**Mathematical Logic and formal languages**]: Mathematical Logic – *model theory, temporal logic*.

General Terms
Theory, Legal Aspects, Verification.

Keywords
Digital investigation, event logic, clock hypothesis

1. INTRODUCTION
Investigations are inquiries into past events. The purpose of an investigation is to find evidence of previous events. Investigation of digital media with the purpose of finding evidence is commonly referred to as *digital investigation*. In recent works, efforts have been made to make the digital investigation based on scientific principles, by using a hypothesis-based approach. [1] In this approach, the investigator formulates his hypothesis about the occurred events, and tests them using the available evidence.

A timestamp is a recorded representation of a specific moment in time Timestamps play an important role in digital investigations. Traditionally, they are used to place the event generating the timestamp at a specific moment in time. The identification that a certain event on a computer took place at a specific time makes it possible to correlate the event with events occurring outside the computer system. These may be events occurring in another digital system, or in the physical world. A particularly important application of timestamps in digital investigation is attribution; the ability to attribute events to a specific person. This is important, because most investigations aim at placing the responsibility for occurred events on one or more individuals. If evidence of the investigated events is digital, it may be necessary to place the event at a specific point in time in order to be able to attribute it to the correct person. If the time of the event inferred from the evidence is incorrect, it may not be possible to attribute it to anyone, or the event may be attributed to the wrong person. The prevalence of dynamic network addresses on the Internet makes timing important in all types of investigations of events that occurred on the Internet. In many such investigations, attribution relies on the identification of which computer were using an IP-address at a particular time. If the IP-address is dynamically assigned, the originating computer can only be identified if a log of the usage of the address exists, and the time of the event can be established with sufficient certainty and accuracy. Only in this case can the originating computer be identified from the usage log by selecting the correct IP-address and time entry.

e-Forensics 2008, January 21-23, 2008, Adelaide, Australia.

A timestamp always refer to the clock from which it is generated. Since the timestamp is a function of the clock, it is always relative to the adjustment of the clock. Unfortunately, clocks are not fully reliable. Clocks may drift, thereby generating timestamps gradually more different from those generated from other clocks. Clocks may also fail, and produce completely incorrect timestamps. [2, 3] Further, clocks on most systems may be adjusted at any time by the user of the system to show a different date and time than civil time. The uncertainty associated with digitally stored timestamps implies that timestamps should not be relied upon as evidence without justification of these factors. In particular, it should not be blindly assumed that timestamps are based on a clock that is adjusted to civil time. These uncertainties are worrying for investigators. If timestamps cannot be relied upon, then it is in many cases not possible to trace the use of an IP-address, since identification of the time of the event is necessary to find the correct originator.

This work takes the approach that time stamps can be tested in the hypothesis based investigation model. The investigator can

formulate a hypothesis about historical values of the clock. By defining a model of the investigated system and observing the timestamp values on the investigated computer, the clock hypothesis can be tested for consistency with the available evidence. Such testing can provide justification for a particular clock hypothesis. When a clock hypothesis is justified, the time of the events on the computer can be interpreted accordingly, and can then be used for correlation with other sources. Previously, a formalism for clock hypotheses and consistency testing with causality between timestamped events has been defined. [4] In this work, a system with actions and timestamps will be defined. This can be used to develop additional consistency tests for clock hypotheses.

2. CLOCKS

A clock is a device designed to give the user an approximation of time that is sufficiently coherent to allow him to measure and compare time periods and sufficiently consistent with other clocks to allow him to perform actions concurrent with other clock users without continuous coordination. Clocks are in other words designed to give an approximation of time.

Definition 1. Let T be the domain of time. Let V be the domain of time values produced by a clock. $c(t)$ is a clock function $T \mapsto V$.

The definition of a clock function does not impose any restrictions on the clock values as a function of time. For example, if $t_1 < t_2$ it may well be the case that $c(t_1) > c(t_2)$. And even if $t_1 < t_2 < t_3$, it may be the case that $c(t_1) = c(t_2) = c(t_3)$. The latter situation may for example occur if the events occurring at t_1, t_2, t_3 are so close together in time that the clock is unable to differentiate between them.

A clock hypothesis is a hypothesis about historical values of a clock. In an investigation, the investigator can formulate clock hypotheses and test them for consistency with the available evidence. [4]

Definition 2. A clock hypothesis $c_h(t)$ is a clock function $T \mapsto V$ that is hypothesized to represent the real clock on $c(t)$ on a system.

In an investigation, the investigator formulates a clock hypothesis, which is then the working theory about the historical values of the clock on the investigated computer. The investigator may for example hypothesize that the clock has always been adjusted to UTC+10.

The clock hypothesis may also include previous adjustments to the clock. It is for example possible for the investigator to formulate a hypothesis in which the owner of the computer adjusted the clock one year back, then created some (antedated) documents, and then adjusted the clock forward again. In such a hypothesis, the clock function $c_h(t)$ will be a discontinuous function.

3. ACTIONS AFFECT TIMESTAMPS

In order to test if a clock hypothesis is consistent with the timestamps in a system; we can build a model of the investigated system, by representing the operations in the system that can possibly change the timestamps as *actions*. A model of a system with timestamps can then be described as a table listing the timestamps and the actions that affect them. We call this an affects table.

Definition 3. An action *affects* a timestamp if and only if an occurrence of that action sets a new value for the timestamp and removes the previous value for the timestamp. An *affects table* is a table listing all possible combinations of timestamps in a system, and all actions in the system and timestamps they affect. An affects table for a system with n timestamps has 2^n entries.

Example 1. Create an affects table for the following simple file system: A file system contains files, and each file has a Created timestamp, an Accessed timestamp and a Modified timestamp. Files can be Created, Read or Written. Reading a file causes the Accessed timestamp to be updated. Writing a file causes both the Accessed timestamp and the Modified timestamp to be updated, and Creating a file causes all three timestamps to be updated. There is only one timestamp of each type for each file, so whenever a timestamp is changed, the previous value is lost.

The affects table for this file system is given in the following:

Table 1. Affects table for the file system in Example 1

	Created	Modified	Accessed	Actions
0				
1	X			
2		X		
3	X	X		
4			X	Read
5	X		X	
6		X	X	Write
7	X	X	X	Create

The affects table states clearly how timestamps are affected by actions. The affects table also shows which timestamp affect combination does not occur with any action. This information can be utilized to derive invariants on timestamp, by reasoning on sequences of timestamp updating and corresponding sequences of actions.

4. TIMESTAMPING ORDERS

In an investigation, the investigator observes values of timestamps on each investigated file. Each file has n different timestamps $\theta_1, \theta_2, ..., \theta_n$. The observed values of these timestamps were set at moments in time $t_{\theta_1}, t_{\theta_2}, ..., t_{\theta_n}$, where the values observed by the investigator are $c(t_{\theta_1}), c(t_{\theta_2}), ..., c(t_{\theta_n})$, set by the clock of the investigated system. Since the clock function $c(t)$ of the investigated system is unknown, the investigator cannot map these values directly to the moments in time $t_{\theta_1}, t_{\theta_2}, ..., t_{\theta_n}$ when timestamping occurred. But the investigator can list possible sequences of timestamping, and determine if the observed result is consistent with a specific clock hypothesis, given the affects table for the system.

Definition 4. In a system with n timestamps, the *stamping time set* Θ is the set of moments in time $t_{\theta_1}, t_{\theta_2}, ..., t_{\theta_n}$ when each

observed timestamp value $c(t_{\theta_1}), c(t_{\theta_2}), ..., c(t_{\theta_n})$ for the observed timestamps $\theta_1, \theta_2, ..., \theta_n$ was set.

Example 2. For the file system described in Example 1, the stamping time set is $\Theta = \{t_c, t_m, t_a\}$, where t_c is the time of production of the observed Created timestamp, t_m is the time of production of the observed Modified timestamp and t_a is the time of production of the observed Accessed timestamp.

To determine which (if any) sequence of actions in the system could have resulted in the observed timestamps, it is interesting to determine the different sequences in which timestamping could have occurred. Each pair of values in Θ, (t_i, t_j), may be related by either $t_i < t_j$, $t_i = t_j$ or $t_i > t_j$.

Definition 5. A *timestamping order* is a sequence of all elements in the stamping time set Θ, where each element is related to the next element in the sequence with the equals-relation = or the less-than relation <. The equals relation imply that the stamping times are equal; the two timestamps were set at the same time. The less-than relation imply that the first stamping time is earlier than the second stamping time; the production of the first timestamp occurred at an earlier time than the production of the second timestamp. Each different stamping time in a timestamping order constitutes a *step* in the timestamping order. When two or more stamping times are equal, they constitute a step in the timestamping order together.

An example timestamping order for the simple file system is ($t_c = t_m < t_a$). With this timestamping order, the Created and Modified timestamps were set at the same time, and the Accessed timestamp was set at a later time than the Created and Modified timestamps.

A list of all timestamping orders can be constructed where each stamping of a specific timestamp may have occurred before, after or at the same time as the stamping of the other timestamps. The list of possible sequences for $n = 3$ can be found in Table 2.

Table 2. All timestamping orders, n = 3

Number	Sequence	
1	$(t_1 < t_2 < t_3)$	
2	$(t_1 < t_3 < t_2)$	
3	$(t_2 < t_1 < t_3)$	
4	$(t_2 < t_3 < t_1)$	
5	$(t_3 < t_1 < t_2)$	
6	$(t_3 < t_2 < t_1)$	
7	$(t_1 = t_2 < t_3)$	
8	$(t_3 < t_1 = t_2)$	
9	$(t_2 = t_3 < t_1)$	
10	$(t_1 < t_2 = t_3)$	
11	$(t_1 = t_3 < t_2)$	
12	$(t_2 < t_1 = t_3)$	
13	$(t_1 = t_2 = t_3)$	

5. POSSIBLE ACTION SEQUENCES

When all timestamp updating is represented by actions, the cause of timestamping having occurred in a specific sequence must have been actions that have occurred in a specific sequence. An action sequence is a sequence of actions of arbitrary length.

Definition 6. An *action sequence* is a sequence of one or more actions, where each element is related to the next element in the sequence with the equals-relation = or the less-than relation <. The equals relation imply that the actions occurred at the same time. The less-than relation imply that the first action occurred earlier than the second action.

The relationship between an action sequence and a timestamping order is that every observed timestamping order must have been created by an action sequence. When considering all timestamping orders, there may be many action sequences that may cause a particular timestamping order. There may however also be timestamping orders, which cannot be created by any action sequence. These timestamping orders cannot occur in the system. The relationship between action sequences and timestamping orders can be deducted from the affects table.

Definition 7. A timestamping order is *possible in a system* if there is at least one action sequence in the system that may cause the timestamping order. If there is no action sequence that can cause the timestamping order, then the timestamping order is *impossible in the system.*

By using the affects table, it is possible to find all action sequences that may have caused a specific timestamping order, by the following procedure:

1. Find all actions or combination of actions affecting all timestamps in the first step in the timestamping order.
2. For each following step in the timestamping order, find all actions or combination of actions affecting all timestamps in that step, and not affecting any timestamps listed in previous steps. If there is no such action, then this timestamping order is not possible in the system.

The task of finding all actions or combination of actions can be implemented as follows:

1. For every timestamp θ_i find all actions affecting it, and add them to a set A_i.
2. For every action $a \in A_i$, check if a affects any timestamp θ_j listed in previous steps in the timestamping order. If so, remove it from A_i.
3. Actions $a \in (A_1 \cap A_2 \cap ... \cap A_n)$ affect all timestamps in that step. Remove them from A_i.
4. If all sets A_i are still non-empty, the remaining actions represent combinations of actions affecting all timestamps for that step. The combinations can be found with the Cartesian product $A_1 \times A_2 \times ... \times A_n$

Example 3. Find all action sequences for the timestamping order ($t_c < t_m < t_a$) for a file in the file system in Example 1.

From the affects table for the simple file system in Table 1, the steps in the sequence yields:

Step 1 (t_c): Create (t_c is only affected by Create)

Step 2 (t_m): Write (t_m is affected by Create and Write, only Write does not affect t_c)

Step 3 (t_a): Read (t_a is affected by Read/Write/Create, only Read does not affect t_c, t_m)

Thus, the only possible action sequence for timestamping order ($t_c < t_m < t_a$) is (Create < Write < Read).

Example 4. Find all action sequences for the timestamping order ($t_m = t_a < t_c$) for a file in the file system in Example 1.

From the affects table for the simple file system in Table 1, the steps in the sequence yields:

Step 1 (t_m=t_a): Create, Write (t_m and t_a are both affected by Create and Write)

Step 2 (t_c): *none* (t_c is only affected by Create, but Create also affects t_m and t_a)

Thus, the timestamping order ($t_m = t_a < t_c$) is not possible in the system.

By using this procedure for all timestamping orders for a given number of timestamps, one can now complete the reasoning in a system with known actions. The result of this exercise will be a list of timestamping orders impossible in the system and a table of possible action sequences of each timestamping order possible in the system.

Example 5. Find all action sequences for the simple file system.

This file system has three timestamps for each file (n = 3). All timestamping orders for such a system are given in Table 2. Assigning $t_1 = t_c$, $t_2 = t_m$ and $t_3 = t_a$ produces all timestamping orders for this system, shown in column "Timestamping order" in Table 3. Following the action sequence procedure for each timestamping order listed in the table by using the affects table for the simple file system given in Table 1, gives the possible action sequences for that timestamping order, shown in the column "Action Sequence":

Table 3. Action sequences for the simple file system

Number	Timestamping order	Action Sequence
1	($t_c < t_m < t_a$)	(Create, Write, Read)
2	($t_c < t_a < t_m$)	None
3	($t_m < t_c < t_a$)	None
4	($t_m < t_a < t_c$)	None
5	($t_a < t_c < t_m$)	None
6	($t_a < t_m < t_c$)	None
7	($t_c = t_m < t_a$)	(Create, Read)
8	($t_a < t_c = t_m$)	None
9	($t_m = t_a < t_c$)	None
10	($t_c < t_m = t_a$)	(Create, Write)
11	($t_c = t_a < t_m$)	None
12	($t_m < t_c = t_a$)	None
13	($t_c = t_m = t_a$)	(Create)

The only timestamping orders in Table 3 possible in the system are sequences where $t_c \le t_m \le t_a$. Thus, $t_c \le t_m \le t_a$ is a property that always holds for this system, an invariant.

Invariants for a system that has been found using the reasoning above can be used to test a clock hypothesis. In the example file system, it is now known that $t_c \le t_m \le t_a$. If for example $c(t_c) > c(t_a)$, a hypothesis that the clock has always been adjusted to UTC+10 would be rejected, since UTC is never adjusted backwards.

6. MODELLING A REAL FILE SYSTEM

The procedure described in the previous sections can be used to create a model of a real file system, determine which timestamping orders are possible in the system and derive invariants of the file system for use with a clock hypothesis checker. To illustrate this procedure, this section performs it on the semantics in Windows XP for file timestamps stored in the NTFS $STANDARD_INFORMATION attribute. The basis for the model described here is the semantics determined by Carrier. [5] The model assumes that the files in question exist, are larger than the file cache size, and that updating of the last accessed timestamp is enabled.

In a system with three timestamps, the affects table contains $2^3 = 8$ entries. The actions are:

Read: reading a file

Create: creating a new file

Write: modifying an existing file

CopySrc: copying a file (source file)

CopyDest: copying a file (destination file)

MoveIntra: moving a file internal to a file system

MoveInterSrc: moving a file across file systems (source file)

MoveInterDest: moving a file across file systems (destination file)

The following affects table can then be constructed:

Table 4. Affects table for Windows XP / NTFS

	Created	Modified	Accessed	Actions
0				
1	X			
2		X		
3	X	X		
4			X	Read,

				CopySrc, MoveIntra, MoveInterSrc, MoveInterDest (ReadGroup)
5	X		X	CopyDest
6		X	X	Write
7	X	X	X	Create

The actions in row 4 of the affects table all have the same effect on timestamps. In the following, they will be grouped together as ReadGroup, meaning that where this action occurs, any of the actions Read, CopySrc, MoveIntra, MoveInterSrc or MoveInter-Dest may have occurred.

With $n = 3$, the timestamping order table in Table 2 can be used. Applying the action sequence procedure for each timestamping order yields the table of action sequences listed in Table 5.

Table 5. Timestamping orders in Windows XP/NTFS.

No	Timestamping order	Action Sequence
1	($t_c < t_m < t_a$)	(Create / CopyDest < Write < ReadGroup)
2	($t_c < t_a < t_m$)	None
3	($t_m < t_c < t_a$)	(Create / Write < CopyDest < ReadGroup)
4	($t_m < t_a < t_c$)	None
5	($t_a < t_c < t_m$)	None
6	($t_a < t_m < t_c$)	None
7	($t_c = t_m < t_a$)	(Create / CopyDest=Write < ReadGroup)
8	($t_a < t_c = t_m$)	None
9	($t_m = t_a < t_c$)	None
10	($t_c < t_m = t_a$)	(Create / CopyDest, Write)
11	($t_c = t_a < t_m$)	None
12	($t_m < t_c = t_a$)	(Create / Write, CopyDest)
13	($t_c = t_m = t_a$)	(Create / CopyDest=Write)

From the table, it is evident that there are no possible action sequences where t_a does not occur in the last step. Consequently, in this system, $t_m \leq t_a$ and $t_c \leq t_a$. These invariants can be used to check clock hypotheses for Windows XP systems with NTFS.

7. RESULTS

This work studied how a system model can be created used to test a clock hypothesis for consistency with timestamp evidence. A system model can be created by listing the actions in the system and their effect on timestamps in an affects table. By listing all possible timestamping orders, it can be determined which timestamping orders are possible in the system and which action sequences that may cause them. A procedure for deriving possible action sequences from the list of possible timestamping orders is given. From the list of possible and impossible timestamping orders, invariants for a system can be derived. These invariants can be used to test a clock hypothesis for consistency with evidence in the form of timestamps stored on an investigated system.

On the systems examined in real digital investigations, there will exist tens- or even hundreds of thousands of timestamps. By modelling the system using the techniques described in this paper, it is then possible to test a clock hypothesis against a large number of timestamps. This will put a clock hypothesis under close scrutiny, and will lead to its justification if there is no evidence to refute it. Justification of a clock hypothesis is important in digital investigations, because it will provide a possibility to translate the timestamps observed on a system to an independent clock. Thus, the real time of stamping can be established, which can be used to correlate the time of the events on a digital system with events occurring elsewhere.

8. CONCLUDING REMARKS

The testing of clock hypotheses provided in this work requires a model of the investigated system to be constructed. In order to provide a complete model of a real system one must clearly understand the system completely, something that can probably only be accomplished by studying the implementation details of the system. It might however be reasonable to construct a partial model only by studying the effects of operations on the real system, if it can be justified that the only actions taken on the system were those that were included in the model. In a real operating system, this could for example be accomplished by testing the different operations in the system and how they affect timestamps. If one could not be sure that all possible operations in the operating system had been included, one would not know for certain if the rejection of a clock hypothesis was caused by a wrong clock hypothesis or by missing actions in the model. This does not have to be a serious problem in digital investigations, where timestamp operations must be manifested in software, which can be found during the investigation.

The method provided in this work can be applied during investigations of digital media, such as seized computers. Since most systems use common operating systems, the construction of a model does not have to be repeated in every investigation. It is enough that the model has been built for a specific system type once, it can thereafter be used in all digital investigations concerning that system type. The method presented here are therefore well suited for implementation in integrated software packages for digital investigation.

9. REFERENCES

[1] 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.

[2] 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.

[3] F. Buchholz and B. Tjaden, "A brief study of time," *Digital Investigation,* vol. 2007:4S, pp. 31-42, 2007.

[4] S. Y. Willassen, "Hypothesis based investigation of digital timestamps," in *IFIP WG 11.9 Workshop*, Kyoto, Japan, 2008.

[5] B. Carrier, *File system forensic analysis*. Upper Saddle River, N.J.: Addison-Wesley, 2005.