Forensic Science International: Digital Investigation 49 (2024) 301759

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA

Was the clock correct? Exploring timestamp interpretation through time anchors for digital forensic event reconstruction

Céline Vanini [a,*], Christopher J. Hargreaves [b], Harm van Beek [c], Frank Breitinger [a,**]

[a] School of Criminal Justice, University of Lausanne, 1015, Lausanne, Switzerland
[b] Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom
[c] Netherlands Forensic Institute (NFI), Laan van Ypenburg 6, 2497 GB, Den Haag, the Netherlands

ARTICLE INFO

Keywords:
Event reconstruction
Time anchor
Timestamp interpretation
Timeline
Formalization
Digital forensic
Dataset

ABSTRACT

Timestamps and their correct interpretation play a crucial role in digital forensic investigations, particularly when the objective is to establish a timeline of events a.k.a. event reconstruction. However, the way these timestamps are generated heavily depends on an internal clock, or 'system time', from which many are derived. Consequently, when this system time is skewed due to tampering, natural clock drift, or system malfunctions, recorded timestamps will not reflect the actual times the (real-world) events occurred. This raises the question of how to validate the correctness of the system clock when recording timestamps and, if found incorrect, how to determine system clock skew. To address this problem, this paper defines several important concepts such as *time anchors*, *anchoring events*, *non-anchoring events* and *time anomalies* which can be used to determine if the system time was correct. Using two examples - a Google search and a file creation - and comparing correct and skewed versions of the same set of performed actions, we illustrate the use and potential benefits of time anchors to demonstrate the correctness of the system clock for event reconstruction.

1. Introduction

In criminal investigations, event reconstruction plays a central role in uncovering the truth, solving crimes, and administering justice. It is used to understand complex phenomena and historical events by piecing together fragmented information into a coherent narrative. A cornerstone of event reconstruction within digital forensics is timestamps which when extracted, normalized, and sorted allow inference of events that occurred.

The origins of timestamps can vary, but many are derived from the internal clock of the computer hardware, i.e., operating system components and applications retrieve the time via an API such as GetSystemTime on Windows (Microsoft Corp., 2024). The origin of this 'system time' is the device hardware frequently referred to as a Real Time Clock (RTC). However, this time does not need to relate to the time in the real world, and often does not (Marouani and Dagenais, 2008; Acer et al., 2017). These inconsistencies may occur due to active tampering or arbitrary changes, natural variations in clocks, or malfunctioning of the system, e.g., a failing battery (Sandvik and Årnes, 2018). In other words, "[s]tored timestamps may not accurately reflect the times that the events occurred" (Willassen, 2008b). This phenomenon is described as clock skew (Kaart and Laraghy, 2014).

An undetected clock skew can have severe consequences on event reconstruction and thus ultimately for the outcome of a case. For instance, let us assume a fictive case where a person was murdered using Potassium cyanide. The police seized the computer of a primary suspect and found the search query 'how does Potassium cyanide react to the body'. If this search query was sent after the body was found and reported, the person may have been interested in details of a crime that they have read about or been accused of. In contrast, if the search query occurred before the incident, it would be incriminating evidence. This example underlines the importance of aligning the times of events in the real world with events reconstructed from digital devices. A real-world example is given in Appendix A.

This leads to the question:

* Corresponding author.
** Corresponding author.
E-mail addresses: celine.vanini@unil.ch (C. Vanini), christopher.hargreaves@cs.ox.ac.uk (C.J. Hargreaves), harm.van.beek@nfi.nl (H. van Beek), frank.breitinger@unil.ch (F. Breitinger).
URL: https://FBreitinger.de (F. Breitinger).

https://doi.org/10.1016/j.fsidi.2024.301759

Available online 5 July 2024

For a given reconstructed event that is inferred using stored timestamps, how can the correctness of the clock from which those timestamps originated, and at the time they were recorded, be demonstrated?

In case the outcome is that the system time is false, a subsequent question arises:

How can we determine the system clock skew?

To answer these questions, we formalize the concept of time anchors, i.e. characteristic elements that categorize and define components that are essential for assessing the accuracy and correctness of system time. In this context, a time anchor is a combination of local and external time references. We demonstrate the validity of these concepts using two multi-part controlled experiments, respectively focusing on scenarios where the correctness of the clock is questioned: during a Google search (Example 1) and a file creation (Example 2). Note that these examples are intended to be illustrative rather than exhaustive. Although these experiments were conducted on Windows, we note that our concepts extend beyond this platform.

In summary, this article provides the following contributions:

- We define and explain the concept of *time anchors* which allow investigators to assess the correctness of the system clock for a certain event.
- We introduce and define the concept of *time anomalies*, the differences between *anchoring* and *non-anchoring events* or *local* and *remote time anchors*, which complement the general definition of time anchors and help in the evaluation.
- We present experiments showcasing these concepts and share the corresponding datasets.

These terms provide formalization that is currently not present in this area of digital forensics and aims to not just improve the process, and potentially make event reconstruction more reliable, but also to provide a published, peer-reviewed reference that practitioners can use to support their use of such approaches.

1.1. Placement

The focus of this work is detecting clock skew, such as modifications to the system time either through the OS or BIOS/EFI or clock drift. We focus exclusively on aspects that influence the outcome of a timestamp generated by the system clock. Therefore, this paper discounts the possibility of timestamps being manipulated after their generation, e.g., an active adversary modifies a timestamp in a database.

This work also discounts the possibility that some timestamps may have initially originated from a different system (e.g., through file transfers) and disregards any timestamp transference issues, such as a file system incorrectly interpreting a timestamp from another file system (Nordvik and Axelsson, 2022).

Future work will investigate these possible manipulations in more detail.

1.2. Outline

The rest of the paper is structured as follows. Sec. 2 provides an overview and discussion of the current state of the field. In Sec. 3, we describe the research problem and introduce four key concepts to the paper: *time anchors*, *anchoring* and *non-anchoring events*, *time anomalies*, and *local time anchors*. Sec. 5 details the experimental setup. We illustrate the practical application of these concepts in two examples where the correctness of the clock is in question: a Google Search (Sec. 6) and a File Creation (Sec. 7), which highlight the alternative approaches needed when different types of events are being reconstructed. In Sec. 8, we highlight the limitations of this paper and explore possible future work, and conclude this work in Sec. 9.

2. Related work

The complexity of date and time has been discussed in early digital forensic research and includes approaches to making sure the clock is correct.

2.1. Using clock models

The behavior of clocks over time can be described using clock models. For instance, Buchholz and Tjaden (2007) employed a graphical approach in a long-term study involving 8000 hosts on the Internet to characterize the clock behaviors of synchronized hosts, revealing that a significant number were not aligned with UTC. Given the importance of external time sources for this work, Buchholz and Tjaden findings underscore the necessity for updated research in this area.

Other research has expanded the application of clock models to simulate historical clock behaviors and remove errors on timestamps. For example, Stevens (2004) used clock models to align timestamps originating from various system clocks to a reference time. However, constructing these models necessitates knowledge of the adjustments a clock has undergone. As these may not be known to an examiner, Willassen (2008a) suggested the formulation of clock hypotheses, which consistency can be assessed using timestamps stored on a device. The work, which was later refined by Willassen (2008b) lies on causal connections between events using Lamport's happened before-relation (Lamport, 1978). For example, under a correct clock hypothesis, if an event a happened before an event b, timestamp a must indicate an earlier time than timestamp b. Similarly, Gladyshev and Patel (2005) and Marrington et al. (2011) respectively use causal connections between events to define event time bounding around an event whose time is unknown, and automate the detection of inconsistencies in timelines.

2.2. Using timestamp correlation

Within a 'checklist' for date/time evidence by Boyd and Forster (2004), aside from handling timezone issues, it is recommended to "Record the CMOS time on seized or examined system units in relation to actual time" and also to correlate timestamps with "additional times, dates and activities both on the computer and away from it".

Such additional dates and times can take many forms and correlation can also be used to assess the correctness of the clock generating these timestamps at the time of their generation. For example, Weil (2002) investigated the use of locally stored external timestamps inserted by web servers in dynamic web pages (e.g., a news site). When stored in the web browser's cache, these timestamps can be correlated with the creation time of the local cache file that contains the web page, and clock skew can be estimated by calculating the average difference between external and file system timestamps. However, this work is limited to a single data source and does not attempt to generalize the approach. Later, Kaart and Laraghy (2014) exemplified the practicality of employing similar external timestamps from mobile operators stored within messages, like missed call notifications and voicemail alerts, through a case study conducted on an Android smartphone. These timestamps were compared in that particular context with file system timestamps in the message database, to determine time zone and clock settings within a period of interest.

Schatz et al. (2006) continued the work of Weil (2002) and proposed an automated approach that compares timestamps in cache and history records to external timestamps stored remotely within squid logs (proxy logs provided by the Internet Service Provider). Similarly, Kaart and Laraghy (2014) also suggested correlating call detail records (CDR) contents with call logs or SMS messages stored within smartphone devices but lacked an experiment. The practical applicability of such external sources of time is intrinsically tied to their availability and accessibility. These data sources may require legal access for retention and may also be subject to deletion after a specific time frame.

2.3. Summary

While the literature has looked into clock skew and the methods for detecting it, many approaches have relied on sources stored remotely that might not always be available to the investigator (e.g., phone operator/ISP), or they have focused on a single artifact (e.g., cache files), which alone cannot be used to assess the correctness of clocks at *any* point in time. It should also be noted that these discussions are primarily found in older studies, with no recent evident developments. Although the reason remains unclear, one explanation could be the default synchronization, which may foster the assumption that today most system clocks are correct. However, this is not always the case as stated by Acer et al. (2017) who demonstrated that incorrect client clocks are one of the major causes of Chrome HTTPS certificate errors. To address these problems, this work builds upon previous findings and introduces the concept of time anchors, evaluating their strengths, and discussing alternative approaches such as time anomalies that can be used to complement the time anchoring approach.

3. Definitions

This work uses various *times* which are defined in this section. Specifically, we differentiate between the system clock time C_S which is time according to the operating system, and the real-world time C_W; We also define an external clock time C_E which originates from an external source, e.g., a server on the Internet. Generally, this time cannot be controlled by us/an attacker. In an ideal world, $C_S = C_E = C_W$. If this is not the case, i.e., the clocks differ by a non-trivial amount, we denote this as *skew S*.

Ideally $S = 0$ throughout the lifetime of a system. However, if this is not the case, we can only compare clocks at a certain point in time (timestamp) which we denote by T_S, T_E, and T_W, respectively.

Consequently,

Definition 1. Clock skew S refers to the difference (positive or negative) between the T_S and T_W and can be calculated by $S = T_W - T_S$.

where T_S is derived from the system clock and T_W is derived from the real-world clock.

Note, for this work, we do not need a perfect alignment and we accept $S = 0 \pm$ 'a few seconds' and $T_S \approx T_E \approx T_W$ which may occur due to network delays. While it is acceptable for this work as we primarily focus on user events, we acknowledge that a few seconds may matter in some cases, e.g., as discussed by Bi et al. (2006).

A brief discussion on limitations can be found in Sec. 8.1. Furthermore, we do not consider timezones, and compare T_S to T_W independent of any timezone settings applied on top of the basic UTC, but we recognize their importance when dealing with time-based digital evidence (Boyd and Forster, 2004).

3.1. Time anchor

To reliably reconstruct events from digital data, we need to ascertain if stored timestamps are reliable. As the system clock C_S may not be correct, one approach is to identify artifacts that include both T_S and T_E, i.e., the recorded system time, and a recorded timestamp from an external source that we consider to be correct. An example can be a single record that contains data derived from both time sources. If such an artifact exists, we consider this to be a *time anchor* with the following formal definition:

Definition 2. A time anchor *TA* refers to any digital artifact that is associated with both T_S (a timestamp generated by the system clock) and T_E (a corresponding timestamp originating from a reliable external source of time).

If such an artifact exists, there are two possibilities:

1. If $T_S \approx T_E$, the system time (C_S) was correct at that time.
2. If $T_S \neq T_E$, then C_S was subject to clock skew S at that time; specifically $S = T_E - T_S$.

3.2. Anchoring events

There are important differences in time anchoring that relate to the type of event that is being reconstructed. To discuss these differences, we separate events in low-level events (derived from artifacts with associated timestamps) and high-level events (abstracted sets of low-level events used to reconstruct higher-level more 'human understandable' events) as proposed by Hargreaves and Patterson (2012).

Based on this event inference idea, in the context of evaluating the correctness of the system clock, there are two different types of event reconstruction. As examples, first consider the high-level event of 'a Google search was conducted' which can be inferred from a URL in the browser history, along with entries in a cache data structure, along with file creation times for associated cached objects if they are stored as external files on disk. Since one of the low-level events (cache entry record) contains a timestamp derived from the system clock, and a corresponding external timestamp (see Sec. 6.1), it is a time anchor, and therefore the reconstruction of the Google search event is 'self-anchoring'. This is an important distinction because it means that statements can be made about the correctness of C_S *at* the *precise* time of the reconstructed event.

Definition 3. An anchoring event e^{TA} is one that is inferred from one or more artifacts that are themselves time anchors.

Here, the term "anchoring event" must be distinguished from similar and related concepts described by previous research. Cohen (2013) first introduced the concept of an *anchor event* as "some event that can be asserted by the examiner based on personal experience or other similar authority". The term "anchor event" is also used by Trenwith and Venter (2013) as an event that "can draw clear lines between the physical world and the digital world". The term is used to describe an event that can be proved had taken place, e.g., that the user did indeed perform an action on the device and that it was not the result of malicious software.

3.3. Non-anchoring events

Not all events are anchoring events, i.e., conclusive statements cannot be made about the correctness of the clock at the precise time of the event. For instance, let us consider the high-level event 'a file was accessed', which can be inferred from the access times of a file (depending on the operating system), potentially the creation or modification of a link file, entries in most recently used (MRU) lists in the Registry, and other operating system artifacts. Here, none of the artifacts contain external timestamps, and thus they are not time anchors.

Naturally, non-anchoring events (e) and anchoring events (e^{TA}) are intertwined. That is, artifacts from which non-anchoring events are inferred can be surrounded by a *lower* bound anchor TA_l and an *upper*

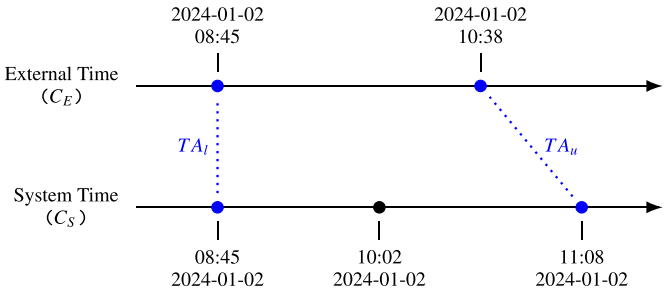

Fig. 1. Two time anchors TA_l and TA_u (expressed as blue dotted lines) surrounding an arbitrary artifact that is not a time anchor, e.g., the 'last accessed time' of a file. Here, TA_l indicates $C_S \approx C_E$ whereas TA_u indicates $C_S \neq C_E$.

bound anchor TA_u as depicted in Fig. 1.

Intuitively, one approach is to use these *TA*s as temporal boundaries to determine the closest anchoring times around artifacts that are not time anchors themselves. While one may think that this is 'good enough' as long as the anchors (TA_l, TA_u) are close to this artifact (e.g., in the range of seconds or minutes), this is more complex. Returning to the 'file was accessed' example, let us consider two anchors as illustrated in Fig. 1 surrounding the 'last accessed time' of a file at time $T_S = 10{:}02$. In this scenario, the artifact lacks external timestamps, and the real-world time the file was last accessed remains unknown. Therefore, if the file was accessed with a system clock negatively skewed by 5 h (i.e., $T_W = 15{:}02$), the 'last accessed time' would fall between the two anchors, TA_l and TA_u at 10:02. In such a situation, inferring that the clock was correct would be a false interpretation.

3.4. Time anomalies

To validate the correctness of non-anchoring events, the concept of anchors should be complemented by *time anomalies*:

Definition 4. A time anomaly is an indicator from within the system suggesting that the system clock did not follow a temporal progression aligned with the real-world time.

Identifying anomalies comprehensively is difficult. Intuitively, one has to holistically search for everything that may indicate that the clock was skewed. Some examples that can be seen as time anomalies are the following:

Explicit artifact: For example, a record in a Windows event log may indicate 'the system time was changed - previous T_S is 13:05:21 and new T_S is 08:05:21'.
Incorrect relative sequence: Inconsistencies in sequential information or a relative position may indicate a time anomaly. This could be an incorrect relative sequence in filename incrementation, order of (raw) entries in a database, or logical positions on a hard drive. Example: A filename IMG_003.JPG with an earlier creation time than IMG_002.JPG.
Differences in measurable changes: For example, should w32time synchronize every 9 h, any deviation in the regular time frame between these synchronizations could be indicative of an anomaly.

3.5. Local vs. remote time anchors

Depending on where the reference timestamp T_E is stored, we differentiate between local and remote anchors. If the timestamp used to validate the local system timestamp is stored locally along with the system timestamp, then the time anchor can always be constructed which is denoted as *local*. In contrast, if the external timestamp does not exist as an artifact on the local system and requires access to an external data source to create the time anchor, we denote this as *remote*.

For instance, Weil (2002) discusses external timestamps stored within dynamically generated web content that are cached locally. This scenario is referred to as *local* as the data point can be found on the device under investigation; the artifact itself contains a system-derived timestamp T_S, and a corresponding external one T_E. In contrast, Kaart and Laraghy (2014) suggest comparing call detail records (accessible via a *remote* service provider) with local artifacts of call logs or messages. Consequently, the evidence is *remote*.

The boundaries of available data sources will change between investigations, organizations, and investigators, for example, not all will have access to Call Data Records, or not all cases will have a cache proxy to correlate with. However, in the first example, where artifacts of the local and corresponding external timestamps are stored on the same device that is under investigation, this represents a more generic scenario and is defined below as a *local time anchor*. However, the potential for the collection of data from additional sources for correlation is acknowledged and included in the earlier *time anchor* definition, but this work focuses on the more generally applicable situation of local time anchors.

Definition 5. A local time anchor is a specific instance of a time anchor where both the timestamp generated by the system clock C_S and the corresponding external timestamp C_E are available on the same device.

In summary, this section defined *time anchor*, *(anchoring and non-anchoring events*, *time anomalies*, and *local time anchors*. The next section provides discussions as well as experiments demonstrating the validity of these terms and their usefulness in practice.

4. Time anchors in investigations

Time anchors and anchoring events allow one to make statements about the correctness of the system clock. Specifically, an examiner can say that the clock was likely correct when $T_S \approx T_E$. We say likely as this is not the case for active tampering.

4.1. Time-anchoring artifacts

By definition, time anchors require external time sources. Consequently, only those artifacts that trigger remote events can produce anchoring events. Examples of such artifacts are:

Browsers: Events related to browser activity may generate anchoring artifacts. This could be cache files, temporary files, history (i.e., databases), cookies, or downloads. Example: the creation time of cookies is local and the expiry time is external (Whitfield, 2011).
Messaging applications: Installed software used for communication may include anchors within messages or data that have been transmitted. Example: A message received via the Signal Desktop application records the times 'sent' (time of sending from external) and 'received' (local time).
Email software: Programs such as Thunderbird or MS Outlook may include time anchors. Example: Emails sent via Thunderbird have local times stored within the Global Database global-messages-db.sqlite or the popstate.dat file (time of sending the email) and in the email headers (server times) located in the Inbox file.
Online storage providers: Applications such as Dropbox, OneDrive, or Google Drive may include time anchors when files are synchronized and activities are logged. Example: Files synchronized on a Google Drive have modified times stored in a database called metadata_sqlite_db (Joun et al., 2023), and from observations, the modified time is based on server time, which can be compared with the (local) file system modification times of these files.

This is not an exhaustive list. Many apps are web-based and can provide external timestamps. In addition, network logs that contain peer/server timestamps may be considered. This article uses three different time anchors within the Chrome cache, history files, and Windows event logs. Additional time anchors would enhance the technical analysis in the case studies, but the concept would remain the same.

4.2. Time anchor identification

The automatic identification of anchors is non-trivial and beyond the scope of this paper. For this article, we employed a manual approach that consists of comparing artifact generation under both skewed and correct system time conditions.

Initially, the idea was to use timeline generation and visualization tools such as *Plaso/Log2timeline* (Plaso documentation) and *Timesketch* (Berggren, 2018) to automate the analysis. However, we found that *Plaso* does not extract all required information, specifically, timestamps generated by an external time source (e.g., received timestamps in email

headers).

This manual approach is sufficient for our example but is not suitable for comprehensive analyses. Therefore, it is necessary to advance the automation of the analysis of time anchors and time anomalies. This could be done by creating a software tool or extending existing frameworks such as Plaso by creating plugins. The software would need to fill in the timestamps not extracted by timeline generation tools, and either identify discrepancies or automatically tag time anchors in timelines.

5. Experimental setup

This section describes the setup before the details of the specific experiments are provided. To demonstrate and validate the concept of time anchors, two multi-part experiments were performed: one exploring anchoring events (Example 1: Google Search), and the other one exploring non-anchoring events (Example 2: File Creation). The experiments are based on Microsoft Windows 10 Professional and VMware Workstation 17 Player.

5.1. Baseline system preparation

In preparation for these experiments, we created a playbook (further detailed in the upcoming sections). This included setting up all necessary accounts, such as email accounts. Next, we built a baseline virtual machine (VM) and installed the corresponding programs. Lastly, we conducted a series of preliminary user actions, which included file creation, sending emails, and navigating websites.

5.2. Data set creation

For the purpose of this work, we established a pair of identical VMs for each experiment: one with C_S synchronized and the other with C_S deliberately skewed. To simulate clock skew within the virtual machines, we chose to adjust the system time backward through the user interface.

As VMs have unusual properties when it comes to time management, modifications were made to ensure they behaved as close as possible to regular devices (details can be found in Appendix B).

For each experiment, we performed the following steps: (1) duplicate the baseline system twice; (2) launch one copy of the VM, (3) execute the sequence of predefined actions, (4) shut down the VM, (5) launch the other copy, (6) modify the settings of the system time backward in the user interface, (7) execute the *same* sequence of predefined actions, (8) reset system time using the "automatic synchronization" setting in the user interface, and (9) shut down the VM. All actions were performed manually to simulate normal user activity. The timing of each user action (in C_W) was also manually documented using as a reference the host clock, synchronized with the native Windows Time service (Microsoft Corp, 2022).

5.3. Data set analysis

As mentioned in Sec. 4.2, timelining tools mostly ignore external timestamps. Hence, disk images were manually analyzed using *X-Ways Forensics*.[1] For some artifacts requiring further data to be parsed, we also used specialized forensic tools, which will be mentioned throughout the text. A comparative analysis of artifacts resulting from each action on both virtual machines (C_S correct vs. C_S skewed) allowed us to identify external times and time anchors. These artifacts are publicly available Vanini et al. (2024).

[1] https://www.x-ways.net/forensics/(v19.8).

6. Example 1: Google Search

This section illustrates the use of time anchors in instances where the event of interest is an anchoring event. Specifically, we chose to focus on Google searches as previously explored by Weil (2002). An example of an investigative question may be:

Was C_S correct when conducting the search?

To answer this question, we followed the procedure described in Sec. 5.2. The base VM was duplicated and two identical copies (VM3 and VM4) were created. For VM1, C_S remained unaltered and reflects C_W. For VM2, C_S was backdated by approximately 3h using the Graphical User Interface (16:04 to 13:09 on Sept. 25, 2023), after we had disabled automatic synchronization. Later, time synchronization was re-enabled and the time of VM2 was corrected to 16:27. For this example, a range of web browsing activities were conducted on both virtual machines; expanding the list of potential browsing activities to the Google search to identify any edge cases. Sample activities included visiting different websites, sending/receiving emails (via Outlook on the web and Gmail), reopening tabs, refreshing a web page, downloading/uploading files, and synchronizing files with local applications. The web browser that was used throughout these experiments was Google Chrome.

6.1. Time anchors

Based on the investigative question, our analysis focused on two data sources: Chrome cache and Chrome history, which are used to (temporarily) store data about visited websites. Chrome history records are stored in an SQLite database called History and cache entries are dispatched in different files in the user's Chrome data. The tool *ChromeCacheView*[2] was used to simplify the parsing of cache entries and the History database was opened in *DB Browser for SQLite*.[3]

6.1.1. Chrome cache

Each cache file stores a copy of the web resource along with its corresponding HTTP header response returned by the server that issued this resource. As previously discussed by Weil (2002), these cached resources (although not explicitly labeled as such in his work) function as time anchors: The HTTP header response contains a timestamp issued by the server (T_E) that can be compared with the last accessed timestamp included in the associated cache entry (T_S).

In VM1, where the system clock remained unaltered, our observations generally revealed that for most artifacts $T_S \approx T_E$ (plus or minus a few seconds). However, this was not always the case, as we noted instances where $T_S > T_E$. This phenomenon can be partially explained since the last accessed timestamp reflects the most recent date and time a cached resource has been accessed. Therefore, encountering situations where $T_S > T_E$ is not surprising, especially in our data set where we revisited previously accessed web pages multiple times. Nevertheless, we observed that many of these records had time differences ranging from a few hours to several days. Possible explanations include that some servers might not maintain accurate clocks, or that the server time indicates when a resource was uploaded to the web server rather than reflecting the 'actual time'.

Furthermore, we also discovered that while the activities for VM1 spanned from 10:40 to 11:45, cache records were only found starting from 11:35 onward. This phenomenon may be because around that time the virtual machine was restarted and all tabs previously opened were reopened using the shortcut CTRL+SHIFT+T. This underlines the fact that cached data may not always be retrieved.

In VM2, C_S was backdated by approximately 3 h. Thus, we primarily observed artifacts with $T_S < T_E$, as depicted in Fig. 2 (records highlighted

[2] https://www.nirsoft.net/utils/chrome_cache_view.html (v2.46).
[3] https://sqlitebrowser.org/(v3.12.2).

URL	Last Accessed	Server Time	Web Site
https://www.youtube.com/ptracking?html5=1&video_id=DcCSTs9UHhY&cpn=X7DZo2nShSVOMGNt&ei	25.09.2023 16:29:15	25.09.2023 16:29:15	https://youtube.com
https://www.google.com/client_204?atyp=i&biw=1920&bih=960&dpr=1&ei=NJkRZf31CZr_7_UPodO62	25.09.2023 16:29:09	25.09.2023 16:29:09	https://google.com
https://www.google.com/search?q=youtube+radiohead&rlz=1C1VDKB_deCH1076CH1076&oq=you&gs	25.09.2023 16:29:08	25.09.2023 16:29:08	https://google.com
https://repo.zotero.org/repo/metadata?version=5.0.112&last=0	25.09.2023 16:27:47	25.09.2023 16:27:47	chrome-extension://ekhagklcjbdpajgpjgmbionohlpdbjgc
https://ogs.google.com/u/0/widget/app?awwd=1&gm3=1&origin=chrome-untrusted%3A%2F%2Fnew-t	25.09.2023 16:27:47	25.09.2023 16:27:47	chrome://new-tab-page
https://ogs.google.com/u/0/widget/app?awwd=1&gm3=1&origin=https%3A%2F%2Fmail.google.com&(	25.09.2023 13:31:15	25.09.2023 16:26:27	https://google.com
https://www.google.com/setgmail?use_corp=on&no_1pjar=1&zx=vcqf5crvg5kn	25.09.2023 13:31:13	25.09.2023 16:26:26	https://google.com
https://mail.google.com/dynamic-email/relay/v5/index.html	25.09.2023 13:31:13	22.09.2023 11:29:57	https://google.com
https://www.msn.com/staticsb/statics/latest/auth/auth-redirect-blank.html	25.09.2023 13:30:00	25.09.2023 16:25:12	https://msn.com
https://login.live.com/Me.htm?v=3	25.09.2023 13:30:00	25.09.2023 16:25:12	https://msn.com

Fig. 2. Results from VM2. Sample of Chrome cache records (text/html body type) around the time of resynchronizing C_S to C_W.

in blue). Here, the skew S can be estimated by comparing the results of the subtraction between T_E and T_S from multiple records: $S \approx$ 2:55h. As shown in the figure, after resynchronizing C_S to C_W around 16:27, $C_S \approx C_E$.

6.1.2. Chrome history

The History database contains a visits table which, when combined (sql: join) with the url table, outputs when web pages were visited. Each of these records is associated with a timestamp (T_S) that is based on the system time, indicating when a specific address was accessed ('visit_time' field). During our comparative analysis, we identified that *some* records also included an external timestamp (T_E) in UTC embedded within particular URLs ('url' field). These specific records are thus by definition time anchors. Several of these external timestamps were found in URLs associated with Google searches or logins to online platforms such as Gmail and Outlook on the web. Note that their format was different across web servers and services. As an example, an outlook URL from the url table of VM2 is considered: https://login.live.com/logout.srf?ct=1695651904&rver=7.0.6738.0&id=292841&ru=https:%2F%2Foutlook.live.com%2Fowa%2Fcsignout.aspx%3F%3f%3fumkt%3Dfr-FR%26exch%3D1%26RpsCsrfState%3D09cad5ee-c6cd-d359-61b1-c9e18b3df45c.

While not immediately obvious, the URL key ct represents a timestamp: 1695651904 which equals 25.09.2023 at 16:25:04 (local time).

Table 1 shows the results for several actions that were performed on both virtual machines (VM1 and VM2), in which timestamps T_S stored within the 'visit_time' field and timestamps T_E embedded within URLs are compared with the documented timing of events T_W. Note that for comparison purposes, all timestamps were normalized to the timezone set in the virtual machines. This table shows that $T_E \approx T_W$ for each experiment. For VM2, S can be approximated by subtracting T_E with T_S ($\approx$ 2:54h).

It is worth mentioning that we identified some edge cases. For instance, when "refreshing a web page" or "reloading tabs from the history", timestamps T_E may refer to the first date and time a URL was accessed (i.e., requests contain timestamps with the 'old' timestamp as a parameter). An example of the Google search is depicted in Table 2 (where visit (1) corresponds to the first time the specific resource was accessed). On both VMs, we observe that T_E is equivalent for each action. While this may be obvious (the browser reloads an existing URL that already includes the timestamp), this shows that care is needed when comparing T_S and T_E (false positives are possible).

6.2. Was T_S correct when conducting the search?

Going back to the investigative question raised earlier, we show how time anchors can be used to answer it: Consider an example where an investigator examines a suspect's computer (embodied by our two virtual machines VM1 and VM2). The investigator discovers a 'peculiar' search query in the keyword_search_terms table of the Google Chrome history database: 'do digital forensic investigators dream in hexadecimal?' (for illustration, we selected a query that we genuinely performed on both VMs, albeit one that does not appear particularly inquisitive). The examiner wants to determine if C_S was correct when conducting the search query.

To express an opinion, an examiner may use the C-Scale as proposed by Casey (2020) (also referred to as the 'Strength of Evidence scale'). The scale aims at helping practitioners to express their evaluative opinion in a more understandable and refined manner, at the final stages of the investigation. It includes two core elements: the number of sources that agree and their resistance to tampering. According to the C-Scale, the strength of evidence is higher when multiple and independent sources agree and these sources are tamper proof/more difficult to tamper with.

6.2.1. Findings (VM1)

After consideration of the keyword_search _terms table, the examiner discovers that the query of interest is linked to three distinct URL IDs. These IDs, also present in the urls and visits tables, link to three URL strings. Two of these include external timestamps and are (by definition) time anchors. These time anchors indicate $T_E \approx T_S \approx$ 10:45. When looking at cached data on the computer, the examiner finds that records only cover a period from 11:35 to 11:45. Hence, no cached data linked to the search query can be found. Considering these observations, the examiner may conclude that the strength of evidence is very strong (C5) under the hypothesis that the clock was correct when conducting the Google search. The examiner assigns C1 to the observed digital evidence under the hypothesis that the clock was skewed as the time anchors contradict the hypothesis, but future observations might necessitate a reevaluation.

6.2.2. Findings (VM2)

Performing the identical analysis steps to a skewed machine leads to a different conclusion. When analyzing the urls and visits tables, the examiner finds two time anchors indicating that $C_S \neq C_E$ ($T_S \approx$ 13:10 and $T_E \approx$ 16:05). The examiner also finds multiple cached resources linked to the search query (web pages, pictures, etc) similarly indicating that $T_S \approx$ 13:10 and $T_E \approx$ 16:05. The examiner may conclude then that the strength of evidence is very strong (C5.5) under the hypothesis that the clock was skewed by approximately 2:55h behind when conducting the Google search, and erroneous/extremely weak (C0.5) under the alternate hypothesis (contradictive evidence).

7. Example 2: file creation

As discussed previously, there are situations in which an event relevant to an investigation is not an anchoring event, i.e., it does generate artifacts that are themselves time anchors. The second example therefore focuses on file creation, which in many instances, is a non-anchoring event. Here, an example of a question of interest is:

Was C_S correct when creating the file?

To answer this question, we followed the same procedure for Example 1. The base VM was duplicated and two identical copies (VM3 and VM4) were created. For VM3, C_S remained unaltered and reflected C_W. For this case, we performed a variety of actions (subject to the creation of time anchors as described in Sec. 4.1) including browsing activities (again on Google Chrome), creating and modifying files of different formats (texts, PDFs, spreadsheets, etc.), sending emails using both the native Windows mail client and Outlook for Windows. For

Table 1
Comparison of timestamps as a result of several events in the Chrome history database of VM1 and VM2. We can see on VM1 that $T_S \approx T_E$ while $T_S < T_E$ on VM2.

n°	Action	Stored T_S (VM1)	Stored T_E (VM1)	T_W	Stored T_S (VM2)	Stored T_E (VM2)	T_W
(1)	Google search using keywords	10:45:20	10:44:59	10:45	13:10:52	16:05:55	16:05
(2)	Navigating through the results of (1)	10:45:40	10:44:59	10:45	13:10:53	16:05:55	16:05
(3)	Accessing Outlook	11:17:26	11:17:26	11:17	13:19:57	16:15:09	16:15
(4)	Log out from Outlook	11:42:23	11:42:54	11:42	13:29:51	16:25:04	16:25

Table 2
Edge cases events in the Chrome history database of VM1 and VM2. We can see for both experiments that for each event the same T_E is stored.

n°	Action	Stored T_S (VM1)	Stored T_E (VM1)	T_W	Stored T_S (VM2)	Stored T_E (VM2)	T_W
(1)	Google search using keywords	10:45:20	10:44:59	10:45	13:10:52	16:05:55	16:05
(2)	Closing and reopening tab (1)	11:34:18	10:44:59	11:34	13:24:14	16:05:55	16:19
(3)	Reloading page (2)	11:36:01	10:44:59	11:36	13:26:39	16:05:55	16:21

VM4, C_S was backdated and set to C_W-3:53h (i.e., 17:00 to 13:07 on Sept. 25, 2023), again using the Graphical User Interface. After performing the same series of actions, synchronization was enabled and C_S was adjusted to 17:23.

7.1. Time anchors

In contrast to Example 1, the investigative question necessitates broadening the range of artifacts to analyze. Based on the usage simulated in this case, we extracted and analyzed the following artifacts that we knew or suspected to contain T_E timestamps: again Chrome history and cache files, and Windows Time Service event logs.

Windows event log files are used by the operating system to record a variety of system events. These files are located in the Windows/System32/winevt directory. We discovered while investigating the functioning of the Windows Time Service that a set of events related to time synchronization are stored within the event log file Microsoft-Windows-Time-Service%4Operational.evtx. According to the Microsoft documentation, the event IDs range from 257 to 266 (Microsoft Corp., 2021b). A description of the most relevant event IDs is provided in the following section. In addition, we also extracted and analyzed file internal and external metadata, along other Windows event logs. Files' internal and file system metadata were examined using *X-Ways Forensics* and *Exiftool*,[4] whereas event logs were examined using the *EvtxECmd* command line tool.[5] Lastly, the History and cache files were analyzed using the same process and tools as described in Sec. 6.

7.1.1. Windows Time Service event logs

The analysis of the Windows Time service events logs on both experiments allowed us to determine the following: events 257 and 258 respectively indicate the times at which the time service starts and stops, according to C_S. Event 261 records any changes in the system time and provides the old T_S and new T_S. As the new T_S is provided by the configured set of time servers during time synchronization, this new T_S reflects T_E ($T_S \approx T_E$). Event 216 is thus by definition a time anchor. Other events provide some contextual information: Event 264 stores the name of these time servers, e.g., 'time.windows.com', and event 266 keeps track of any time synchronization request and indicates the reason code for this request, e.g., 'reason code 0: an explicit request from an administrator'. An example of event 261 extracted from VM4 is provided below:

> W32time service has set the system time to 2023-09-25T15:22:59.610Z(UTC). Previous system time was 2023-09-25T11:29:03.926Z (UTC). System Tick Count: 1174671

7.2. Time anomalies

As C_S on VM3 remained unaltered, this section concerns findings from VM4. Due to the clock being set backward, several examples of time anomalies were observed and are described in the following:

7.2.1. Specific artifacts

Within the Security.evtx file, events with ID 4616 are generated every time C_S is changed (Microsoft Corp, 2021a). Unlike event ID 261, which logs only changes by the Windows Time service, this event ID records all adjustments to the system time, including manual adjustments. For example, within VM4, one event ID created at 25.09.2023 13:07:31 indicates 'The system time was changed (previous C_S 25.09.2023 17:01:26, new C_S: 25.09.2023 13:07:31)'.

7.2.2. Incorrect relative sequence

Several sequential order (time) anomalies were identified during analysis. As both the event log files and history database store records using a specific order, having an incorrect clock for VM4 caused a number of these records to not be ordered chronologically. This is illustrated in Table 3, where an inconsistency in the sequence of record IDs was observed in the *visits* table of the Chrome history database.

7.2.3. Differences in measurable changes

The time between each time synchronization with the Windows time service is stored in the SpecialPollIntervalsetting.[6] On the virtual machines, this setting is set to approximately 9 hours, meaning a synchronization event is triggered every 9 hours on the system. As the time interval during which the activities were performed was relatively short (less than 9 hours), we did not observe any specific changes in these time dynamics. However, it is possible to assume that over a longer period, the time synchronization requests resulting from the correction of the clock within VM4 would have manifested as a typical pattern within

Table 3
Results from VM4. Samples of visit records from the Chrome history database show an inconsistency in the sequence of record IDs (time jumping backwards from 16:55:34 to 13:21:29).

id	Stored T_S
116	25.09.2023 16:55:25
117	25.09.2023 16:55:34
118	25.09.2023 13:21:29
119	25.09.2023 13:21:35

[4] https://exiftool.org/(v12.67).
[5] https://github.com/EricZimmerman/evtx (v1.5.0.0).
[6] SYSTEM/CurrentControlSet/services/W32Time/Time-Providers/NtpClient

these time dynamics.

7.3. Was C_S correct when creating the file?

This section illustrates an example of the potential benefits and limitations of time anchors when the investigative question concerns an event that is not anchoring.

Let us again assume that during an investigation, a suspect's computer is seized and analyzed (embodied here by VM3 and VM4). The examiner is interested in determining if C_S was correct when the file under investigation was created. For illustration purposes, let us further assume that this investigative question concerns the file Ideas.odt created on VM3 (the equivalent file on VM4 was called `PaperIdeas.odt`). The file system timestamps (MACE) stored within the $STANDARD_INFORMATION attribute (SIA) of these files are described in Table 4.

Compared to the first example, the reasoning for non-anchoring events is more delicate: As the file creation does not include an external timestamp, a holistic search of (all) time anchors and time anomalies in the system is required.

7.3.1. Findings (VM3)

After consideration of several time anchors within VM3, including artifacts from the Chrome cache, browsing history, and Windows Time service event logs, the examiner identifies multiple cache records (all containing the same T_S and T_E values) as the closest time anchors to the creation timestamp of Ideas.odt, as illustrated in Fig. 3. In this scenario, no time anomalies are found.

The time anchors identified are 01:36:01 before and 01:11:19 after the event of interest. While this is an indication that the clock may have been correct, the evidence for the correctness of the clock is weak (C3 on the C-scale[7]). Even if there are no apparent discrepancies, this is not enough to conclude about the time of the event, therefore the examiner may assign C3 under the alternate hypothesis too. However, a negative result for a search for *time anomalies* provides an additional data point and can be an additional source of evidence. This increases our confidence that the time may have been correct (C4).

7.3.2. Findings (VM4)

In VM4, the examiner finds similar neighboring time anchors which are illustrated in Fig. 4. Interestingly, the upper time anchors (TA_u) indicate that C_S is different from C_E by approximately 2:55h. When searching for anomalies, the examiner discovers an event 4616 in the Security.evtx log file which suggests C_S was changed from 17:01:26 to 13:07:31. Additionally, incorrect relative sequences, as discussed in Sec. 7.2, are found in the *visits* and *downloads* table. For example, a download file with ID 14 which has a starting time (T_S) at 25.09.2023 16:55:05 is followed by download 15 which has a starting time at 25.09.2023 13:22:06. As numerous time anomalies were found around the creation time of `PaperIdeas.odt` and one of the closest time anchors indicates $C_S \neq C_E$, the examiner concludes that the strength of evidence is strong/very strong (C4.5) under the hypothesis that the clock was skewed at the time of the creation of the file. The examiner assigns C1 to the observed digital evidence under the hypothesis that the clock was correct as the time anchors and anomalies contradict the hypothesis, but future observations might necessitate a reevaluation.

Table 4
File system timestamps (M: Last modified, C: File creation, E: MFT entry changed, and A: Last accessed).

Filename	SIA-MCE	SIA-A
`Ideas.odt`	25.09.2023 12:05:29	25.09.2023 13:22:21
`PaperIdeas.odt`	25.09.2023 13:17:11	25.09.2023 13:17:12

[7] "The source(s) of evidence are more difficult to tamper with but there is not enough evidence to support a firm conclusion […]." (Casey, 2020).

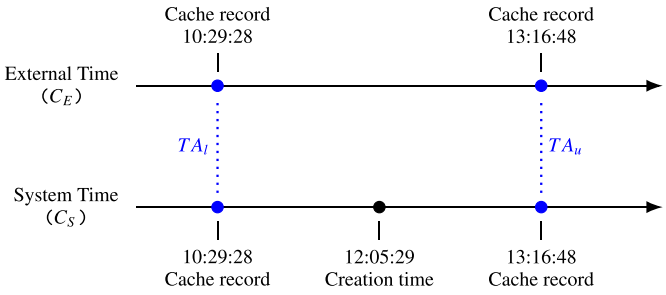

Fig. 3. Illustration of a file creation (non-anchoring event) surrounded by two time anchors. Here, the closest time anchors to the creation timestamp of `Ideas.odt` are two cache records (TA_l, TA_u) indicating that the times were aligned, i.e., $C_S \approx C_E$.

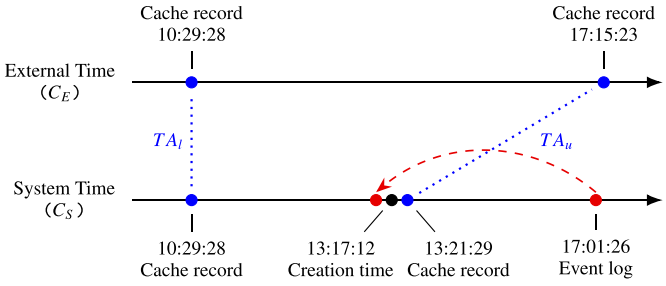

Fig. 4. Illustration of a file creation (non-anchoring event) surrounded by two time anchors. Here, the closest time anchors to the creation timestamp of `PaperIdeas.odt` are two cache records where TA_l: $C_S \approx C_E$ and TA_u: $C_S \neq C_E$. A time anomaly (in red) also indicates C_S was backdated (expressed by an arrow).

8. Discussion and future work

[RQ1] For a given reconstructed event that is inferred using stored timestamps, how can the correctness of the clock from which those timestamps originated, and at the time they were recorded, be demonstrated? Our experiments demonstrated the distinct differences in determining the system time between an anchoring event, such as Google searches, and a non-anchoring event like file creation. When the event of interest is an anchoring event, time anchors produced by this event can demonstrate the correctness of the system clock at the time of the event by comparing T_S and T_E. Of course, if the event generates a multitude of time anchors, the confidence is increased (e.g., a search query may produce entries in cache files and the history database). However, the procedure is more complex when the investigative question concerns a non-anchoring event. In these instances, time anchors are employed to establish temporal boundaries surrounding the event. Our experiments highlighted the necessity to complement the approach with a comprehensive search for time anomalies. Nevertheless, the task is non-trivial and, when approached manually, becomes time-consuming. Future work should therefore look into the automatization of time anchor and anomaly detection.

[RQ2] How can we determine the system clock skew? The examples illustrated that when a time anchor indicates $C_S \neq C_E$, the system clock skew S at this point in time can be calculated by $T_S - T_E$. However, in situations where time anchors do not reveal an obvious clock skew, the examiner must rely on and infer the skew based on time anomalies. For instance, when the anomaly is explicit, such as an entry in Windows event logs, the skew can be computed directly.

8.1. Reliability of external time sources

To define and illustrate time anchors, the external time sources (NTP servers, servers hosting cached content, etc.) have been assumed to be correct. However, a study by Buchholz and Tjaden (2007) has shown that not all servers maintain accurate clocks. In case the external source is incorrect, those external time skews manifest in the locally stored data. During our experiments, we were able to determine that timestamps returned by highly-frequented servers all reflected the current time. Performing a detailed study of external time sources is beyond the scope of this paper but should be considered in the future.

Relying on a single time anchor creates some uncertainty. This is why, even for anchoring events, a probabilistic approach should be used. This is consistent with qualitative descriptions on the C-Scale by Casey (2020) which would place this as C3 - 'weak evidence'. However, examining multiple time anchors, which originate from different independent sources, and are in agreement, allows us to move to C4/C5.

It is also important to note that external time sources can be further separated into external-and-trustable or external-and-not-trustable. For instance, an external source is not always physically distant and maintained by a different entity but could be a device linked to the seized device such as a laptop that syncs with a phone. If the suspect can access the external source, we denote this as external-and-not-trustable and vice versa.

8.2. Forward dating

This article explores backdating but does not address forward dating. Although backdating seems more common and is the main timestamp manipulation example used by Casey (2020), forward dating also has its relevance, such as when establishing an alibi. Precisely, if a suspect plans a crime for tomorrow at 14:00 and wants to prove they were using a computer, they can adjust the system time (C_S) today to 13:55 the next day, browse for 30 min, and power off the device. When the device is eventually seized, potentially days later, the logs will confirm the suspect's statement.

While no experiments have been conducted, we believe that our concepts also apply to forward dating and that time anchors can be used to assess the correctness of C_S. However, there may be some new peculiarities that come with forward dating especially if the device is used (1) after reverting the time and tomorrow at 13:55 and/or (2) during the fictive time, i.e., tomorrow between 13:55 and 14:25.

9. Conclusion

Event reconstruction is a core procedure for uncovering the truth and relies on timestamps. Consequently, it is important to ensure the correctness of the system clock T_S when a timestamp is recorded. To address this concern, the paper formalized the concept of *time anchors* which can be used to show the correctness of T_S at a specific time in a device's history. This concept is complemented by different types of events, e.g., *anchoring* and *non-anchoring* events, which have major implications for the uncertainty that remains after the approach is used. With respect to the latter, we also defined *time anomalies* which can be used to increase confidence in clock accuracy in scenarios involving non-anchoring events. The practicality of these concepts has been demonstrated based on two examples. Lastly, this article includes several examples of time anchors and categorizations, making additional types of time anchors easier to identify.

CRediT authorship contribution statement

Céline Vanini: Conceptualization, Methodology, Validation, Investigation, Writing - Original Draft, Writing - Review and Editing. **Christopher J. Hargreaves:** Conceptualization, Methodology, Writing - Original Draft, Writing - Review and Editing, Supervision. **Harm van Beek:** Conceptualization, Methodology, Writing - Review and Editing. **Frank Breitinger:** Conceptualization, Methodology, Writing - Original Draft, Writing - Review and Editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Jan Gruber and Felix Freiling for their feedback during our discussions. Additionally, we express our appreciation to the anonymous reviewers whose constructive critiques contributed to the enhancement of this research.

Appendix A. Case overview

On the morning of the 23rd of December 2015, Richard Dabate reported that his wife, Connie Dabate, had been shot by a masked man who had broken into their house. First responders promptly arrived at the scene and found Richard Dabate tied in the kitchen, while Connie's lifeless body lay in the basement. During questioning, Richard Dabate informed authorities he had driven back home around 9 a.m. that morning because he had forgotten his computer on his way to work. Upon his arrival, he was attacked by the masked intruder. He claimed to have heard Connie Dabate return home earlier than usual, and shortly after, to have heard two shots. Following that, Dabate explained that he managed to free himself enabling him to activate the alarm and call the police.

In May 2022, Richard Dabate was convicted of the murder of his wife. Among the traces used during the trial were data obtained from Connie's Fitbit activity tracker and iPhone. Fitbit data played a crucial role in the prosecution's argumentation, revealing significant discrepancies in the timeline of events provided by the accused. While Dabate claimed that his wife was killed around 9:05 a.m., the data retrieved from her wearable device indicated alternate patterns of activity and inactivity persisting until 10:05 a.m. Similarly, traces extracted from her iPhone revealed Facebook activity around 9:40 a.m.[8]

In response, the defense questioned the reliability of the traces retrieved from the Fitbit, casting doubts about the accuracy of the timestamps used to invalidate Richard Dabate's claims. They argued that the precision of timing data is intricately linked to the device with which the Fitbit is synchronized, and those devices could be inaccurate (Rohrlich, 2022; NBC Connecticut, 2022; Associated Press, 2022).

While the prosecution's expert ensured the court that it would be very unlikely for the device's clock to be off by an hour, it is not entirely unreasonable to consider the defense's argument a valid point.

Appendix B. Virtual machines and their time

The management of time within VMs differs from a standard Windows. In general, VMs are configured to periodically synchronize with the host. When the guest time deviates from the host time, a process of correction comes into action. This is typically done with the help of additional services installed on the guest OS. On VMware, these additional services (*VMware Tools*) provide two types of time synchronization to track host time: (1) *periodic* (disabled by default) where the guest clock is checked by default every 60 s, and (2) *one-off* where the synchronization occurs upon specific events such as taking a snapshot or starting the *VMware tools* daemon when booting the virtual machine

[8] The details presented in this example are based on reported newspaper content and are employed for illustrative purposes.

(VMware, 2023).

To be able to manipulate the guest clock and to use the native synchronization service on Windows (Windows Time service), we made the following changes to the baseline virtual machine: (1) installed *VMware Tools*, (2) checked that periodic time synchronization was off and (3) disabled one-off time synchronization by adding a set of instructions within the *.vmx* file of the machine.[9] It is worth mentioning that these changes propagate to any copy of the virtual machine.

References

Acer, M.E., Stark, E., Felt, A.P., Fahl, S., Bhargava, R., Dev, B., Braithwaite, M., Sleevi, R., Tabriz, P., 2017. Where the Wild warnings are: root causes of Chrome HTTPS certificate errors. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, Dallas Texas USA, pp. 1407–1420. https://doi.org/10.1145/3133956.3134007.

Associated Press, 2022. Jury Begins Deliberating in Richard Dabate Murder Trial. URL: https://nypost.com/2022/08/18/connecticut-man-richard-dabate-sentenced-in-fitbit-murder-case-of-wife-connie-dabate/.

Berggren, J., 2018. Timesketch. URL: https://timesketch.org/.

Bi, J., Wu, Q., Li, Z., 2006. On estimating clock skew for one-way measurements. Comput. Commun. 29, 1213–1225.

Boyd, C., Forster, P., 2004. Time and date issues in forensic computing—a case study. Digit. Invest. 1, 18–23. https://doi.org/10.1016/j.diin.2004.01.002.

Buchholz, F., Tjaden, B., 2007. A brief study of time. Digit. Invest. 4, 31–42. https://doi.org/10.1016/j.diin.2007.06.004. URL: https://www.sciencedirect.com/science/article/pii/S1742287607000394.

Casey, E., 2020. Standardization of forming and expressing preliminary evaluative opinions on digital evidence. Forensic Sci. Int.: Digit. Invest. 32, 200888 https://doi.org/10.1016/j.fsidi.2019.200888. URL: https://www.sciencedirect.com/science/article/pii/S1742287619303147.

Cohen, F., 2013. Digital Forensic Evidence Examination, fifth ed. 2013 ed. Fred Cohen & Associates, Livermore, CA.

Gladyshev, P., Patel, A., 2005. Formalising event time bounding in digital investigations. Int. J. Digit. Evid. 4, 1–14. https://www.semanticscholar.org/paper/Formalising-Event-Time-Bounding-in-Digital-Gladyshev-Patel/11ab843e473f834c3204c1bcdde1e4b5ceda3192.

Hargreaves, C., Patterson, J., 2012. An automated timeline reconstruction approach for digital forensic investigations. Digit. Invest. 9, S69–S79. https://doi.org/10.1016/j.diin.2012.05.006. URL: https://www.sciencedirect.com/science/article/pii/S174228761200031X.

Joun, J., Lee, S., Park, J., 2023. Discovering spoliation of evidence through identifying traces on deleted files in macos. Forensic Sci. Int.: Digit. Invest. 44, 301502 https://doi.org/10.1016/j.fsidi.2023.301502. https://www.sciencedirect.com/science/article/pii/S2666281723000033. selected papers of the Tenth Annual DFRWS EU Conference.

Kaart, M., Laraghy, S., 2014. Android forensics: interpretation of timestamps. Digit. Invest. 11, 234–248. https://doi.org/10.1016/j.diin.2014.05.001. URL: https://www.sciencedirect.com/science/article/pii/S1742287614000449.

Lamport, L., 1978. Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21, 558–565. https://doi.org/10.1145/359545.359563.

Marouani, H., Dagenais, M.R., 2008. Internal clock drift estimation in computer clusters. Journal of Computer Systems, Networks, and Communications 2008, 583162. https://doi.org/10.1155/2008/583162 publisher: Hindawi Publishing Corporation.

Marrington, A., Baggili, I., Mohay, G., Clark, A., 2011. CAT Detect (Computer Activity Timeline Detection): a tool for detecting inconsistency in computer activity timelines. Digit. Invest. 8, S52–S61. https://doi.org/10.1016/j.diin.2011.05.007. URL: https://www.sciencedirect.com/science/article/pii/S1742287611000314.

Microsoft Corp, 2021a. 4616(s): the System Time Was Changed. URL: https://learn.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4616.

Microsoft Corp., 2021b. Windows Time for Traceability. URL: https://learn.microsoft.com/en-us/windows-server/networking/windows-time-service/windows-time-for-traceability.

Microsoft Corp, 2022. How the Windows Time Service Works. URL: https://learn.microsoft.com/en-us/windows-server/networking/windows-time-service/how-the-windows-time-service-works.

Microsoft Corp., 2024. Getsystemtime Function (sysinfoapi.H). https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getsystemtime.

NBC Connecticut, 2022. Jury Begins Deliberating in Richard Dabate Murder Trial. URL: https://www.nbcconnecticut.com/news/local/closing-arguments-to-begin-in-richard-dabate-murder-trial/2780196/.

Nordvik, R., Axelsson, S., 2022. It is about time–Do exFAT implementations handle timestamps correctly? Forensic Sci. Int.: Digit. Invest. 42–43, 301476 https://doi.org/10.1016/j.fsidi.2022.301476.

Plaso documentation, Parsers and plugins. URL: https://plaso.readthedocs.io/en/latest/sources/user/Parsers-and-plugins.html.

Rohrlich, J., 2022. Guilty! Two-Timing Hubby Is Undone by Murdered Wife's Fitbit. The Daily Beast. https://www.thedailybeast.com/two-timing-husband-richard-dabate-found-guilty-of-wife-connies-murder-in-connecticut-fitbit-case.

Sandvik, J.P., Årnes, A., 2018. The reliability of clocks as digital evidence under low voltage conditions. Digit. Invest. 24, S10–S17. https://doi.org/10.1016/j.diin.2018.01.003. URL: https://www.sciencedirect.com/science/article/pii/S1742287618300355.

Schatz, B., Mohay, G., Clark, A., 2006. A correlation method for establishing provenance of timestamps in digital evidence. Digit. Invest. 3, 98–107. https://doi.org/10.1016/j.diin.2006.06.009.

Stevens, M.W., 2004. Unification of relative time frames for digital forensics. Digit. Invest. 1, 225–239. https://doi.org/10.1016/j.diin.2004.07.003. URL: https://www.sciencedirect.com/science/article/pii/S174228760400057X.

Trenwith, P.M., Venter, H., 2013. Digital forensic readiness in the cloud. In: 2013 Information Security for South Africa, pp. 1–5. https://doi.org/10.1109/ISSA.2013.6641055 iSSN: 2330-9881.

Vanini, C., Hargreaves, C.J., Breitinger, F., van Beek, H., 2024. Time anchoring artifacts for digital forensic event reconstruction (Version 1.0.0) [Data set]. UNIL data service. https://doi.org/10.48657/55mc-5440.

VMware, 2023. Vmware Tools Administration. URL: http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf.

Weil, M.C., 2002. Dynamic time & date stamp analysis. Int. J. Digit. EVid. URL:. https://www.semanticscholar.org/paper/Dynamic-Time-%26-Date-Stamp-Analysis-Weil/77c5edbba3710d9e614012f2615ab6926e5219d0.

Whitfield, L., 2011. Detecting Cmos Clock Changes. https://forensic4cast.com/2011/01/detecting-cmos-clock-changes/.

Willassen, S., 2008a. Hypothesis-based investigation of digital timestamps. In: Ray, I., Shenoi, S. (Eds.), Advances in Digital Forensics IV, vol. 285. Springer US, Boston, MA, pp. 75–86. https://doi.org/10.1007/978-0-387-84927-0_7. http://link.springer.com/10.1007/978-0-387-84927-0_7. iSSN: 1571-5736 Series Title: IFIP — The International Federation for Information Processing.

Willassen, S.Y., 2008b. Timestamp evidence correlation by model based clock hypothesis testing. In: Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, pp. 1–6.

[9] The precise list of instructions can be found here: https://kb.vmware.com/s/article/1189.