Connecting File Timestamps: A Formal Approach

Lorenz Hornung
lorenz.hornung@ou.nl
Open Universiteit
Heerlen, the Netherlands

Hugo Jonker
hugo.jonker@ou.nl
Open Universiteit
Heerlen, the Netherlands

Harm van Beek
harm.van.beek@nfi.nl
Netherlands Forensic Institute
The Hague, the Netherlands
Open Universiteit
Heerlen, the Netherlands

Abstract

Digital entities can be assigned multiple timestamps that relate to each other, forming what we call a *timestamp structure*. These structures have been used to address several challenges, including the detection of tampered timestamps and the reconstruction of timelines of events. However, a formal understanding of timestamps, their induced timestamp structures, and the operations that affect them is missing. This paper closes this gap and presents a novel formal framework providing the foundations to reason about how timestamps relate to one another, independently of the operating system or file system considered.

CCS Concepts

• **Applied computing → Evidence collection, storage and analysis**.

Keywords

Digital Forensics, Formal Model, Timestamps, NTFS

ACM Reference Format:
Lorenz Hornung, Hugo Jonker, and Harm van Beek. 2026. Connecting File Timestamps: A Formal Approach. In *Proceedings of the Digital Forensics Doctoral Symposium (DFDS '26), March 24, 2026, Linköping, Sweden.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3785318.3785319

1 Introduction

Digital investigations often involve analysing timestamps, as these provide valuable information and clues that can be used to reconstruct timelines of events. However, timestamps are also subject to tampering. The timestamps themselves offer some indication as to whether they have been tampered with. More specifically, this can be established by the internal relations between a file's timestamps, which we call *timestamp structures*. For instance, assume that a file's metadata contains timestamps indicating the file's creation time (c) and last access time (a), where the a cannot naturally be before c. Hence, if we encounter a file with $a < c$, this is a strong indication that the file's timestamps have been tampered with.

Even though timestamp structures have been used to reason about a file's temporal information [1, 3, 4, 7], as of yet there is no formal model characterizing timestamps, their induced timestamp structures and operations affecting timestamp structures. Such a formal model would clarify these concepts and provide the foundation for future research on formal reasoning about a file's timestamps and its timestamp structures.

Research Question. To establish the foundations of formal reasoning about the internal relation of a file's timestamps, we focus on the following research questions.

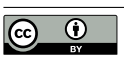
RQ1 **Formal representation of structures of timestamps**: How can we formally represent timestamps and the structure these timestamps induce?
RQ2 **Formal representation of operations affecting structures of timestamps**: How can we uniformly represent the operations that affect structures of timestamps?

Contribution. The main contribution is a novel formalism capable of uniformly capturing timestamps, the structures induced by timestamps, and operations that alter these structures. Importantly, this formalism provides a characterisation of these concepts, independent of the used operating system or file system. For this purpose, we introduce so-called smax $(\cdot)$ operations and demonstrate that these operations are sufficient to capture any modification of structures induced by timestamps. Moreover, we characterise operations that affect timestamps of the NTFS file system.

Related Work. This paper connects to research focusing on reconstructing timelines of events and detecting tampered timestamps. Breitinger et al. [2] and Neale [14] provide an overview of established methods in these areas. In particular, we extend methods that are based on examining a file's timestamp structure within NTFS. For instance, Bouma et al. [1] and Cho [4] examine a file's timestamp structure in NTFS to draw conclusions about previously performed operations on the file and therefore reconstruct a timeline of events. Furthermore, Ding and Zou [5], and Galhuber and Luh [7] illustrate how timestamp structures induced by timestamps can be used to identify tampered timestamps in NTFS.

Additionally, this work is related to the use of formal methods within the field of (digital) forensics. Formal methods can be used to reconstruct timelines of events [6, 8, 10], and to characterize traces [12], and the expressiveness and relevance of evidence [9].

Outline. We will proceed as follows. In Section 2, we introduce the order-theory foundations that underpin our formal analysis. Section 3 introduces the formal framework of file timestamps. We define file timestamps (Section 3.1), timestamp structures (Section 3.2), and operations affecting these structures (Section 3.3). Further, we introduce so-called smax $(\cdot)$ operations and provide some characteristics of these operations in Section 3.4. Lastly, in Section 4, we show how these operations model NTFS file operations.



This work is licensed under a Creative Commons Attribution 4.0 International License.
DFDS '26, Linköping, Sweden
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2120-5/2026/03
https://doi.org/10.1145/3785318.3785319

	SI_1	SI_2	SI_3	SI_4	FN_1	FN_2	FN_3	FN_4
File (F)	2025-01-10 09:30:15 UTC	2025-01-10 09:30:15 UTC	2025-01-15 11:30:45 UTC	2025-01-10 09:30:15 UTC	2025-03-07 18:15:00 UTC	2025-03-07 18:15:00 UTC	2025-03-07 18:15:00 UTC	2025-03-07 18:15:00 UTC

Figure 1: Example of File Timestamps in NTFS

2 Background

To reason about timestamps, we study the structures they induce. For this, we introduce some basic order-theory foundations.

A binary relation $\preceq$ on a set S defines how the elements of S are related. Formally, we define the relation as $\preceq \subseteq S \times S$. A relation on a set can satisfy several properties.

Definition 1. *A binary relation $\preceq$ on a set S is called*

- *reflexive if $x \preceq x$ for all $x \in S$*
- *transitive if $x \preceq y$ and $y \preceq z$ implies $x \preceq z$ for all $x, y, z \in S$*
- *anti-symmetric if $x \preceq y$ and $y \preceq x$ implies $x = y$ for all $x, y \in S$*
- *total if $x \preceq y$ or $y \preceq x$ for all $x, y \in S$*

A binary relation $\preceq$ on a set S is *preorder* if it is reflexive and transitive. Further, it is a *total preorder* if it is reflexive, transitive and total. It is a *partial order* if it is a preorder and anti-symmetric and a *total order* if it is a partial order and total. Further, we call a tuple $(S, \preceq)$ a totally (pre)ordered set if $\preceq$ is a total (pre)order on S.

Example 1. *Let us consider the set $S = \{a, b, c\}$ and the binary relations $\rightarrow_1$ and $\rightarrow_2$ on S.*

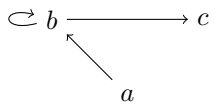


Figure 2: $\rightarrow_1$ on S is *not* a total (pre-)order

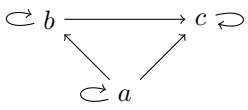


Figure 3: $\rightarrow_2$ on S is a total (pre-)order

$(S, \rightarrow_1)$ is not a totally preordered set, as $\rightarrow_1$ is neither reflexive, transitive, nor total. It is not reflexive because, for example, $a \rightarrow_1 a$ does not hold. It fails to be transitive since $a \rightarrow_1 b$ and $b \rightarrow_1 c$, yet $a \rightarrow_1 c$ does not hold. Finally, the relation is not total because neither $a \rightarrow_1 c$ nor $c \rightarrow_1 a$ holds. In contrast, it is easy to verify that $(S_2, \rightarrow_2)$ is a totally preordered and totally ordered set.

3 A Formal Model for File Timestamps

We are concerned with modeling timestamps of file systems and operations that modify timestamps. For this purpose, we want to characterize timestamps as abstract entities. That is, we do not want to characterize timestamps of actual files formatted in line with a file system; e.g. the file metadata of the pdf version of this paper downloaded on your laptop. We aim to specify timestamps in terms of how these timestamps are used within a file system and what we can conclude based on this information. For this purpose, we understand timestamps as follows.

Notion 1 (Timestamps and file systems). *Timestamps provide temporal information about a digital entity, such as a file or a folder. A file system has a uniform representation of time; i.e., a numerical value that refers to a fixed point in time. Second, a file system sets the number of timestamps to assign to any digital entity.*

To exemplify these aspects, let us consider the NTFS file system.

Example 2. *The NTFS file system stores the metadata of an entry in a Master File Table (MFT) using multiple attributes, including the* `$STANDARD_INFORMATION` *(SI) and* `$FILE_NAME` *(FN) where the timestamps of an entry are stored. The NTFS file system provides eight timestamps, four of which are stored SI, and four in FN. The timestamps in both SI and FN are labeled as last update time, last access time, last metadata change time, and creation time. Interestingly, timestamps in FN do not need to align with their counterparts in SI. Each timestamp is a numeric value which represents a point in time measured as units of 100 nanoseconds that happened after 1601-01-01 00:00:00 UTC [1, 7]. Figure 1 provides an example of such file metadata.*

3.1 From Temporal Features to Timestamps

To model timestamps, we can reason alongside the characteristics given in Notion 1 that any file system provides a fixed number of timestamps and these timestamps are given as numeric values representing some moment in time. The former yields *temporal features* and the latter yields *temporal values*.

Temporal Feature. Since any entry of a file system is assigned a fixed number of timestamps, we view timestamps as values assigned to *temporal features*. A temporal feature x_i is a variable that acts as a placeholder for a timestamp, and the file system must provide a numeric value representing a point in time for this temporal feature. We denote temporal features by $x_i, \ldots, x_j$ and X_{TF} denotes an (infinite) set of temporal features. Since every file in a file system has a fixed number of timestamps, we associate a file system with a set of temporal features. That is, for any file system FS, we denote the set of temporal features associated to FS by $X_{\mathsf{FS}} \subseteq X_{\mathsf{TF}}$.

Example 3. *We characterize the temporal features of NTFS by the set $X^\star = \{SI_1, SI_2, SI_3, SI_4, FN_1, FN_2, FN_3, FN_4\}$. For each entry, these temporal features are assigned to a value representing some moment in time. The temporal features $SI_1, \ldots, SI_4$ are the temporal features underlying the timestamps in SI and the temporal features $FN_1, \ldots, FN_4$ are those underlying the timestamps given in FN.*

Temporal Values. Timestamps of a file refer to specific moments of times. For instance, in Figure 1, file `F` is associated with the timestamp 2025-01-10 09:30:15 UTC for the temporal features SI_1, SI_2, and SI_4. To model these points in time, we introduce *temporal values*. We define the set of temporal values *Val* as $Val := \mathbb{N}^+$. The set $\mathbb{N}^+$ denotes the set of the positive natural numbers $1, 2, 3, \ldots$. In our setting, a natural number represents an assigned temporal value like 2025-01-10 09:30:15 UTC.

The relation $<$ on $\mathbb{N}^+$ suffices to express that a temporal value succeeds another temporal value. Given two temporal values $v_i, v_j \in Val$, the temporal value v_i *precedes* the temporal value v_j or the temporal value v_j *follows* the temporal value v_i if $v_i < v_j$.

Data Point. Connecting the notions of temporal feature and temporal value, we can formalize timestamps. A timestamp is the result of assigning a temporal value to a temporal feature. This assignment is what we call a *data point*. We define data points as functions from a set of temporal features to the set of temporal values.[1]

Definition 2 (Data Point). *Given a set of temporal features X, a data point d on X is a function $d : X \to Val$.*

We call a data point d on the set of NTFS temporal features $X^\star$ a *NTFS data point*. The file timestamps given in Figure 1, can be for instance modeled by the following NTFS data point d_F on $X^\star$:

$$d_\mathsf{F}(x) = \begin{cases} 1 & \text{if } x = SI_1, x = SI_2, \text{ or } x = SI_4 \\ 2 & \text{if } x = SI_3 \\ 3 & \text{if } x = FN_1, x = FN_2, x = FN_3, \text{ or } x = FN_4 \end{cases}$$

where the natural number 1 refers to 2025-01-10 09:30:15 UTC, 2 refers to 2025-01-15 11:30:45 UTC, and 3 refers to 2025-03-07 18:15:00 UTC. In general, a data point models the stored (temporal) data yielding the following formal definition of timestamp.

Definition 3 (Formal Definition of Timestamp). *Given a set of temporal features X, and a data point d on X, $d(x)$ is the timestamp for temporal feature x of data point d.*

A data point d on a set of features X does not necessarily provide a unique timestamp. If X contains multiple temporal features, the data point d results in several timestamps. For instance, the previously defined d_F on $X^\star$ results in the timestamps $d_\mathsf{F}(SI_1), \ldots,$ $d_\mathsf{F}(SI_4)$ and $d_\mathsf{F}(FN_1), \ldots, d_\mathsf{F}(FN_4)$.

This formal characterization of timestamps captures the characteristics of timestamps provided in Notion 1. Each file provides metadata where the timestamps are stored. The temporal metadata is modeled as a data point. Analogously to the timestamp stored in the metadata, the data point provides the temporal information as the assigned temporal value to a temporal feature. The set of temporal features reflects the timestamps considered in the metadata for any file. That is, the set of these timestamps stored in the metadata refers to the set of temporal values assigned to the set of temporal features constituting the domain of the data point.

Example 4. *To show that the framework works with file systems other than NTFS, we examine ext4, which is a common Linux file system. Ext4 records four timestamps for non-deleted files[2] with nanosecond precision: last modification time (m), last access time (a), last metadata change time (c), and creation time (cr). Each timestamp is given as a numeric value representing the number of nanoseconds that have elapsed since the Unix epoch; i.e., 1970-01-01 00:00:00 GMT [11]. Formally, the set of temporal features of ext4 is $X^{ext4} = \{m, a, c, cr\}$ and for a file F', the assignment of its timestamps is definable by $d_{\mathsf{F}'} : X^{ext4} \to Val$.*

[1]To account the absence of timestamps for certain temporal features, data points can be modeled as partial functions. If a dense domain of temporal values is needed for the sake of granularity, one can use $\mathbb{R}^+$. Future research needs to evaluate the impact of these on the proposed framework, particularly with regard to modeling operations.

[2]Ext4 also provides a timestamp for deletion time (d); for non-deleted files 0 is assigned to d. Incorporating d into the formal model hence requires treating ext4 data points as partial functions to capture missing temporal value for d for non-deleted files.

3.2 Timestamp Structures

Given some data points on the same set of temporal features, we want to compare these data points. Particularly, we want to compare the structures these data points induce on the temporal features.

Timestamp Structure. To analyse this structure, we need to define a binary relation on the temporal features considered by a data point. Intuitively, this introduces some sort of ordering on temporal features, more specifically, a ranking with ties. A temporal feature is ranked at least as high as another temporal feature if the data point assigns a temporal value to the former that is at least as great as the the temporal value assigned to the latter.

Definition 4. *Given a set of temporal features X and a data point d on X, the timestamp structure of d on X is a tuple $(X, \preceq_d)$ where $\preceq_d \subseteq X \times X$ is defined as: for all $x_i, x_j \in X$, $x_i \preceq_d x_j$ if and only if $d(x_i) \leq d(x_j)$.*

Note that $x_j \not\preceq_d x_i$ denotes that it does *not* hold that $x_j \preceq_d x_i$. Further, $x_i =_d x_j$ denotes that $x_i \preceq_d x_j$ and $x_j \preceq_d x_i$, while $x_i \prec_d x_j$ denotes $x_i \preceq_d x_j$ and $x_j \not\preceq_d x_i$.

For instance, the following defines the timestamp structure induced by the NTFS data point d_F on $X^\star$ (omitting indicating reflexivity and transitivity):

$$SI_1 =_{d_\mathsf{F}} SI_2 =_{d_\mathsf{F}} SI_4 \prec_{d_\mathsf{F}} SI_3 \prec_{d_\mathsf{F}} FN_1 =_{d_\mathsf{F}} FN_2 =_{d_\mathsf{F}} FN_3 =_{d_\mathsf{F}} FN_4$$

Any data point induces timestamps, which in turn provide the timestamp structure of this data point. This structure is a total preorder on the temporal features that form the basis of the timestamps, which is given by the following proposition.

Proposition 1. *For any set of temporal features X and data point d on X, the timestamp structure of d on X, $(X, \preceq_d)$, is a totally preordered set.*

An immediate consequence of this result is that the number of distinct[3] timestamp structures for a finite set of temporal features X is finite as well. This is due to the fact that there are only finitely many ways in which the temporal features of X can be ordered by a total preorder, and this number depends on the size of X. For instance, for NTFS with $|X^\star| = 8$, there are 545,835 distinct timestamp structures.[4]

Quotient Timestamp Structure. Quotient timestamp structures are deducible from timestamp structures and are needed for modeling file system operations, like the NTFS file operations (Section 4).

Two temporal features $x_i, x_j \in X$ are *equivalent* relative $\preceq_d$ on X (denoted by $x_i \sim_d x_j$) if and only if $x_i =_d x_j$. Thus, $x_i \sim_d x_j$ implies $d(x_i) = d(x_j)$. For any $x_i \in X$, $[x_i]_{\sim_d} = \{x_j \in X \mid x_i \sim_d x_j\}$ is the equivalence class of x_i relative to $(X, \preceq_d)$. For instance, $[SI_1]_{\sim_{d_\mathsf{F}}} = \{SI_1, SI_2, SI_4\}$ is the equivalence class of SI_1 relative to the timestamp structure induced by d_F. Further, $X/{\sim_d} := \{[x_i]_{\sim_d} \mid x_i \in X\}$ denotes

[3]Two timestamp structures $(X, \preceq_{d_i})$ and $(X, \preceq_{d_j})$ are identical, if $\preceq_{d_i} = \preceq_{d_j}$; i.e. if for any $x_i, x_j \in X$, $x_i \preceq_{d_i} x_j \iff x_i \preceq_{d_j} x_j$. They are distinct, if there exist $x_i, x_j \in X_j$ such that $x_i \preceq_{d_i} x_i$ and $x_i \not\preceq_{d_j} x_j$ or vice versa.

[4]The number of distinct timestamp structures is given by a Fubini number [15] and they are also referred to as ordered Bell numbers. For more information, see [13, 15].

the set of equivalence classes of X by $\sim_d$. Each element of $X/\sim_d$ is a set of temporal features that are assigned identical temporal values by the data point under consideration. In our example, $X^\star/\sim_{d_F} = \{\{SI_1, SI_2, SI_4\}, \{SI_3\}, \{FN_1, FN_2, FN_3, FN_4\}\}$.

Given the set of equivalence classes, we define *quotient timestamps structures* as follows.

Definition 5. *Given a set of temporal features X and a data point d on X, the quotient timestamp structure of d on X is a tuple $(X/\sim_d, \preceq_d)$ where $\preceq_d \subseteq (X/\sim_d) \times (X/\sim_d)$ is defined as: for all $[x_i]_{\sim_d}, [x_j]_{\sim_d} \in X/\sim_d$, $[x_i]_{\sim_d} \preceq_d [x_j]_{\sim_d}$ if and only if $d(x_i) \leq d(x_j)$.*

Given some quotient timestamp structure $(X/\sim_d, \preceq_d)$, $[x_i]_{\sim_d} \not\preceq_d [x_j]_{\sim_d}$ denotes that it does *not* hold that $[x_i]_{\sim_d} \preceq_d [x_j]_{\sim_d}$.

As an example, the following defines the quotient timestamp structures induced by the NTFS data point d_F on $X^\star$ (omitting indicating reflexivity and transitivity):

$$\{SI_1, SI_2, SI_4\} \preceq_{d_F} \{SI_3\} \preceq_{d_F} \{FN_1, FN_2, FN_3, FN_4\}$$

One can show that $\preceq_d$ on $X/\sim_d$, in contrast to $\preceq_d$ on X, is antisymmetric. As a consequence, quotient timestamp structures are totally ordered sets, as given by Proposition 2.

Proposition 2. *For any set of temporal features X and data point d on X, the quotient timestamp structure of d on X, $(X/\sim_d, \preceq_d)$, is a totally ordered set.*

3.3 Operations Altering Timestamp Structures

File system entries may be affected by certain operations. For instance, a file might be accessed and modified. Such operation on the entries might modify the timestamps stored in the metadata. This means that after the operation is applied that the temporal values assigned to some temporal features might be distinct from the temporal value assigned to this temporal feature before performing the operation. We aim to formalize how these operations affect timestamps and hence how they affect a specific data point. Importantly, certain operations on timestamps do not result in a unique set of resulting timestamps.

Example 5. *The operation* `Overwriting Copy` *in the NTFS file system indicates that a file f_1 overwrites another file f_2. This operation leads to a modification of the metadata of f_1. The second timestamp stored in SI of f_1 is replaced by the second timestamp stored in SI of f_2. Further, the third timestamp stored in SI of f_1 is replaced by starting time of the operation. The remaining timestamps of f_1 stay unchanged [1].*

Given $X^\star = \{SI_1, \ldots, FN_4\}$ and a data point d on $X^\star$ that represents the metadata of file f_1, the operation `OVERWRITING_COPY` results in a modification of SI_2 and SI_3. The operation leads to a new temporal value assigned to SI_2; i.e. the corresponding timestamp of file f_2. To model such an operation performed on the timestamps f_1 without depending on any further information (and hence not depending on the metadata of f_2), we suppose that performing such an operation to a multiple outputs given as a set of data points with distinct values for SI_2. This set of data points refers to the set of possible data points that result from taking the temporal value assigned to SI_2 in the metadata of f_2.

Definition 6. *Let X be a set of features, an operation on X is a function $g^X : Val^X \to (\mathcal{P}(Val^X) \setminus \emptyset)$.*

Note that Val^X represents the set of all functions that map temporal features of X to a temporal value of Val and hence it is the set of all data points on X. Further, $\mathcal{P}(Val^X) \setminus \emptyset$ denotes the power set of all data points, excluding the empty set. Therefore, Definition 6 states that an operation on data points takes data points as an input and returns a non-empty set of data points.

Any operation g^X constitutes a relation R_g^X defined as $R_g^X := \{(d_i, d_j) \in Val^X \times Val^X \mid d_j \in g^X(d_i)\}$. An operation g^X is called *simple* if and only if R_g^X is a function. This means that for any $d_i \in Val^X$, it returns a singleton set, i.e., $|g^X(d_i)| = 1$. For simplicity, for any simple operation $g^X(d_i) = d_j$ denotes $g^X(d_i) = \{d_j\}$.

Example 6. *An example of a simple operation is the identity operation ID defined as $ID(d) = \{d\}$ for any $d \in Val^X$.*

Operations might be applied subsequently. For instance, the operation g_1^X is first applied on d, then g_2^X is applied to the outcome of the previous step, and so on. We refer to such a subsequent application of operations as a composition of operations. Given some operations $g_1^X, \ldots, g_n^X$ on a set of temporal features X, we denote the composition of operations by $g_n^X \odot \ldots \odot g_1^X$. Further, a composition of operations $g_n^X \odot \ldots \odot g_1^X : Val^X \to (\mathcal{P}(Val^X) \setminus \emptyset)$ is defined as for all $d_i \in Val^X$, $d_j \in (g_n^X \odot \ldots \odot g_1^X)(d_i)$ if and only if $(d_i, d_j) \in R_{g_n}^X \circ \cdots \circ R_{g_1}^X$ where $R_{g_n}^X \circ \cdots \circ R_{g_1}^X$ denotes the composition of relations.[5]

3.4 smax$(\cdot)$ Operations

Since we are concerned with operations that affect the structure of timestamps, we will introduce what we call smax $(\cdot)$ operations. The name smax $(\cdot)$ operation is an abbreviation for "to set a set of temporal features to be the maximum temporal features". Hence, applying such an operation to a data point d_i on X yields an altered induced timestamp structure $(X, \preceq_{d_j})$ in which some set of temporal features $Y \subseteq X$ are now the maximal temporal features relative to $\preceq_{d_j}$. We will demonstrate that they suffice to reason about timestamp structures so that we will use them to define operations performed on timestamp structures.

Definition 7. *Given a set of temporal features X, a* smax $(\cdot)$ *operation on X is a function* smax $(Y) : Val^X \to (\mathcal{P}(Val^X) \setminus \emptyset)$ *with $Y \subseteq X$. For any $d_i \in Val^X$,* smax (Y) $(d_i) :=$

$$\left\{ d_j \in Val^X \;\middle|\; \begin{array}{l} (x_i \in X \setminus Y \implies d_j(x_i) = d_i(x_i)) \;\wedge \\ (x_i \in Y \implies d_j(x_i) = \max(\{d_i(x_j) \mid x_j \in X\}) + 1) \end{array} \right\}$$

Any smax $(\cdot)$ operation is a simple operation. That is, for any data point, any smax $(\cdot)$ operation returns a single data point.

Proposition 3. *Given a set of temporal features X, any* smax $(\cdot)$ *operation on X is a simple operation.*

Applying subsequently multiple smax $(\cdot)$ operations results in a composition of smax $(\cdot)$ operations. This composition of smax $(\cdot)$ operations is a simple operation as well.

Proposition 4. *Given a set of temporal features X, and some* smax $(\cdot)$ *operations on X denoted by $g_1^X, \ldots, g_n^X$, the composition $g_n^X \odot \ldots \odot g_1^X$ is a simple operation.*

[5] Given two binary relations $R, S \subseteq Y \times Y$ on a set Y, the composition $R \circ S$ is defined as $R \circ S := \{(x, y) \in Y \times Y \mid \exists z : (x, z) \in S \wedge (z, y) \in R\}$.

We can demonstrate that smax $(\cdot)$ operations suffice to reason about timestamp structures. Given a data point on a set of temporal features with an induced timestamp structure, any other timestamp structure on the same set of temporal features can be obtained by applying only smax $(\cdot)$ operations to that data point.

Proposition 5. *Let X be a finite set of temporal features, d_i and d_j be data points on X such that $\preceq_{d_i} \neq \preceq_{d_j}$. There exists a finite composition* $\mathsf{smax}(Y_1) \odot \ldots \odot \mathsf{smax}(Y_n)$ *with $Y_1, \ldots, Y_n \subseteq X$ such that* $(\mathsf{smax}(Y_1) \odot \ldots \odot \mathsf{smax}(Y_n))(d_i) = d_k$ *and $\preceq_{d_k} = \preceq_{d_j}$.*

Further, smax $(\cdot)$ affects the relative position of two temporal features within a timestamp structure only if one of those features is set to be a maximal temporal feature.

Proposition 6. *Given a set of temporal features X, data points d_i and d_j on X such that $d_j = \mathsf{smax}(Y)(d_i)$ with $Y \subseteq X$. For any $x_i, x_j \in X \setminus Y$, $x_i \preceq_{d_i} x_j$ if and only if $x_i \preceq_{d_j} x_j$.*

Importantly, for any finite composition of smax $(\cdot)$ operations, there exists a composition in which each smax $(\cdot)$ operation occurs at most once, and the timestamp structures induced by the original and the simplified compositions are identical.

Proposition 7. *Let X be a set of temporal features, d be a data point on X, and $g^X_{k_n}, \ldots, g^X_{k_1}$ be* smax $(\cdot)$ *operations. Further, let $g^X_{k_n} \odot \ldots \odot g^X_{k_1}(d) = d_i$. There exists a finite composition of operations*

$$g^X_{l_m} \odot \ldots \odot g^X_{l_1}(d) = d_j \text{ with } g^X_{l_1}, \ldots, g^X_{l_m} \in \{g^X_{k_1}, \ldots, g^X_{k_n}\}$$

satisfying the following constraints

1. *any $g^X_{l_i}$ occurs at most once in the composition $g^X_{l_m} \odot \ldots \odot g^X_{l_1}$,*
2. *$\preceq_{d_i} = \preceq_{d_j}$; i.e. $\preceq_{d_i}$ and $\preceq_{d_j}$ encode the same structure on X.*

4 Modeling NTFS operations

To provide an intuition how smax $(\cdot)$ model operations that affect timestamps and timestamp structures, we will provide characterizations of some NTFS file operations using smax $(\cdot)$. For this purpose, we will build upon the described timestamp affect by NTFS file operations by Bouma et al. [1, Table 1].[6]

Since we are concerned with the timestamp structure, we are only concerned with modeling the possible change in timestamp structure due to these NTFS file operations. For this purpose, we model some NTFS file operations as operations (in the sense of Definition 6), and some as data points. The latter is needed, because these NTFS file operations generate a completely new NTFS data point instead of affecting an existing NTFS data point.

Simplifications. To model NTFS file operations, we propose the following simplifications. First, the operation start time and end time are distinct, so we assume no operation can occur without a detected elapsed time. Second, we assume the start and end times are always assigned the highest and second-highest temporal values among all temporal features. This encodes that an NTFS operation's time points always occur after any previously considered times, including the temporal values assigned to temporal features by the affected data point.

[6] Due to space constraints, we will not provide the formal characterization of the NTFS file operation `Overwriting from other volume`. Note that the formal characterization can be similarly constructed as that of the NTFS file operation `Overwriting Copy`.

4.1 NTFS File Operations as Data Points

Since we are interested in the timestamp structure, we can select any temporal values, and hence any natural numbers (see Section 3.1), that maintain this structure.

Create (d_{CR}). This NTFS file operation yields a data point d_{CR} where each temporal feature is assigned the starting time. This is expressible by $d_{CR} \in Val^{X^\star}$ defined as $d_{CR}(x) = 1$ for all $x \in X^\star$.

Copy (d_{CO}). This NTFS file operation keeps the temporal value assigned to SI_2 of the copied file, assigns the remaining temporal features, besides SI_3, the starting time, and assigns SI_3 the end time. This is expressible by $d_{CO} \in Val^{X^\star}$ defined as $d_{CO}(SI_2) = 1$, $d_{CO}(SI_3) = 3$, and $d_{CO}(x) = 2$ for all $x \in X^\star \setminus \{SI_2, SI_3\}$.

Move from other volume (d_{MV}). This NTFS file operation keeps the temporal value assigned to SI_1 and SI_2 of the moved file, assigns the remaining temporal features, besides SI_3, the starting time, and assigns SI_3 the end time. Since we do not know which temporal value assigned to SI_1 and SI_2 is higher, multiple possibilities arise. This is expressible by $d_{MV1}, d_{MV2}, d_{MV3} \in Val^{X^\star}$ defined as for all $1 \leq i \leq 3$, $d_{MVi}(SI_3) = 4$, $d_{MVi}(x) = 3$ for all $x \in X^\star \setminus \{SI_1, SI_2, SI_3\}$, and $d_{MV1}(SI_1) = 1, d_{MV1}(SI_2) = 2, d_{MV2}(SI_1) = 2, d_{MV2}(SI_2) = 1, d_{MV3}(SI_1) = d_{MV3}(SI_2) = 1$.

4.2 NTFS File Operations as Operations

Access (AC) and Delete (DE). Applying this operation to a NTFS data point does not affect the temporal structure. We model these NTFS file operations by the identity operation (see Example 6). For any NTFS data point d, they are defined as $\mathsf{AC}(d), \mathsf{DE}(d) := ID(d)$.

Attribute Change (AT). This operation only affects the ranking of the temporal feature SI_3 and results in a novel data point where SI_3 is assigned the starting time and hence the maximum temporal value. Thus, for any NTFS data point d, we define this operation as $\mathsf{AT}(d) := \mathsf{smax}(\{SI_3\})(d)$.

Update (UP). Applying this operation to a NTFS data point alters the rank of SI_2 and SI_3 by putting the former as highest-ranked and the latter as second-highest-ranked feature because the former is assigned the starting time and the latter the end time. It does not affect the relative rankings of the remaining features. For any NTFS data point d, the operation is hence defined as $\mathsf{UP}(d) := (\mathsf{smax}(\{SI_2\}) \odot \mathsf{smax}(\{SI_3\}))(d)$.

Rank of Temporal Features. To formalise the remaining NTFS file operations, we need to define ranks of temporal features. The rank of temporal feature indicates the relative position of a feature among the other features (relative to a timestamp structure).

The function $\mathsf{rank}(\cdot)$ provides the rank of a temporal feature relative to a data point and the induced temporal structure. Given a set of temporal features X and a data point d on X, $\mathsf{rank}^X_d : X \to \{0, \ldots, |X/{\sim_d}| - 1\}$ is defined as

$$\mathsf{rank}^X_d(x_i) = |\{[x_j]_{\sim_d} \in X/{\sim_d} \mid x_i \prec_d x_j\}|.$$

The rank of an element is identical to the number of equivalence classes of temporal features that are assigned to a strictly higher temporal value. Thus, $\mathsf{rank}^X_d(x) = 0$ expresses that the highest

For any NTFS data point d, $Z^\star = X^\star \setminus \{SI_2, SI_3\}$, $n = \left|Z^\star/\sim_d\right|$ and $\mathsf{ask}_d^{Z^\star} : \{0,\dots,n-1\} \to Z^\star/\sim_d$,

$$\mathsf{OC}(d) := \bigcup_{i=0}^{n} \left\{ \left(\mathsf{smax}\left(\{SI_3\}\right) \odot \bigodot_{j=0}^{i-1} \mathsf{smax}\left(\mathsf{ask}_d^{Z^\star}(j)\right) \odot \mathsf{smax}\left(\{SI_2\}\right) \odot \bigodot_{k=i}^{n-1} \mathsf{smax}\left(\mathsf{ask}_d^{Z^\star}(k)\right) \right)(d) \right\}$$
$$\cup \bigcup_{i=0}^{n-1} \left\{ \left(\mathsf{smax}\left(\{SI_3\}\right) \odot \bigodot_{j=0}^{i-1} \mathsf{smax}\left(\mathsf{ask}_d^{Z^\star}(j)\right) \odot \mathsf{smax}\left(\mathsf{ask}_d^{Z^\star}(i) \cup \{SI_2\}\right) \odot \bigodot_{k=i+1}^{n-1} \mathsf{smax}\left(\mathsf{ask}_d^{Z^\star}(k)\right) \right)(d) \right\}$$

Figure 4: Characterization of Overwriting Copy (OC)

temporal value is assigned to x by d, while $\mathsf{rank}_d^X(x) = |X/\sim_d| - 1$ means that d assigns the lowest temporal value to x.

The function $\mathsf{ask}(\cdot)$ outputs the set of temporal features of a specific rank. Given a set of temporal features X and a data point d on X, the function $\mathsf{ask}_d^X : \{0,\dots,|X/\sim_d| - 1\} \to X/\sim_d$ is defined for $0 \le n \le |X/\sim_d| - 1$ as

$$\mathsf{ask}_d^X(n) = [x_i]_{\sim_d} \iff \mathsf{rank}_d^X(x_i) = n.$$

Rename (RE) and Move within volume (MV). Applying these operations once yields a data point in which the features FN_1, FN_2, FN_3, and FN_4 are assigned the temporal values of SI_1, SI_2, SI_3, and SI_4, respectively. Consequently, their rank is changed by putting them at the same rank as these features. Further, SI_3 is set to be the highest-ranked feature (while FN_3 obtains the previous rank of SI_3), because it is assigned the starting time. For any NTFS data point d, $Z^\star = \{SI_1, SI_2, SI_3, SI_4\}$, $n = \left|Z^\star/\sim_d\right|$ and $\mathsf{ask}_d^{Z^\star} : \{0,\dots,n-1\} \to Z^\star/\sim_d$, we define these operations as

$$\mathsf{RE}(d), \mathsf{MV}(d) := \left(\mathsf{smax}\left(\{SI_3\}\right) \odot \bigodot_{i=0}^{n-1} \mathsf{smax}\left(\left\{ SI_i, FN_i \,\middle|\, SI_i \in \mathsf{ask}_d^{Z^\star}(i) \right\} \setminus \{SI_3\} \right) \right)(d)$$

Overwriting Copy (OC). This operation introduces a novel value for SI_2 representing the SI_2-value of the file overwriting the considered file so that SI_2 might take any possible rank of the original timestamp structure, resulting in multiple possible output timestamp structures where SI_2 can have the same rank as other temporal features or not. Since SI_3 is assigned the starting time and hence becomes the highest ranked feature, we can only exclude the possibility of SI_2 being the highest-ranked feature. The formal characterization of OC is given in Figure 4.

5 Conclusion and Future Work

With the digitization of our societies, digital forensics plays a vital role in our justice systems. This necessitates a formal underpinning of the foundational concepts of digital forensics to ensure explainability of, and trust in established methods of digital investigations.

Summary. With respect to RQ1, we introduce a formal framework that characterizes timestamps as the output of assigning temporal values to temporal features (Section 3.1). These assignments, known as data points, induce timestamp structures that are total preorders on the set of temporal features (Section 3.2). Timestamps and timestamp structures have been previously used to reconstruct timelines of events [1, 4] and to determine tampered timestamps [3, 7], but have lacked a precise formal definition. Our provided formalism closes this gap. Regarding RQ2, we formalize operations as functions that map data points to a set of data points (Section 3.3). In particular, we characterize some NTFS file operations (Section 4).

Future Work. Our framework enables formal reasoning about timestamps and timestamp structures. For instance, it lays the foundation of a formal mechanism to identify whether timestamps of a file were tampered.

Due to finite number of timestamp structures for a finite number of temporal features, we can define a *reachability problem* that allows us to conclude that timestamps of any data point, like the timestamps in Figure 1, are tampered.

Problem 1 (Reachability Problem of a Data Point).
Input: *A finite set of temporal features X; a starting d_i and a target data point d_j on X; and a set of operations $O = \{g_1^X, \dots, g_m^X\}$.*
Question: *Does there exist a composition of operations $g_k^X \odot \dots \odot g_l^X$ with $g_k^X, \dots, g_l^X \in O$ such that $d_k \in g_k^X \odot \dots \odot g_l^X(d_i)$, and the timestamp structures $(X, \preceq_{d_k})$ and $(X, \preceq_{d_j})$ are identical?*

If this question is answered negatively, then such a timestamp structure must be the result of timestamp tampering. Future research therefore needs to provide a complete (or, for the specific use case, sufficiently close to complete) list of operations and starting data points. The latter represent the format for how a file system assigns timestamps to a newly created file.

If NTFS is considered, this list would include the data points and operations defined in Section 4. This allows us to identify whether the timestamp of file F (shown in Figure 1 and modeled by d_F in Section 3.1) has been tampered with by analysing whether the induced timestamp structure $(X^\star, \preceq_{d_\mathsf{F}})$ (given in Section 3.2) can be reached from these starting data points and operations.

To evaluate this reachability problem efficiently, future research needs to establish that any composition of operations results in a timestamp structure that is identical to one produced by a composition of operations of fixed length.

Propositions 5 and 7 suggest that this is indeed possible, because any modification of timestamp structures is expressible as a composition of $\mathsf{smax}(\cdot)$ operations and these compositions can be reduced to a composition of fixed finite length. However, this is not a trivial task because it is not immediately apparent how to construct a composition of operations like OC and RE using a reduced composition of only $\mathsf{smax}(\cdot)$ operations.

References

[1] Jelle Bouma, Hugo Jonker, Vincent van der Meer, and Eddy Van Den Aker. 2023. Reconstructing Timelines: From NTFS Timestamps to File Histories. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ACM, Benevento Italy, 1 – 9. doi:10.1145/3600160.3605027

[2] Frank Breitinger, Hudan Studiawan, and Chris Hargreaves. 2025. SoK: Timeline based event reconstruction for digital forensics: Terminology, methodology, and current challenges. *Forensic Science International: Digital Investigation* 53 (2025), 301932. doi:10.1016/j.fsidi.2025.301932 DFRWS USA 2025 - Selected Papers from the 25th Annual Digital Forensics Research Conference USA.

[3] Gyu-Sang Cho. 2013. A computer forensic method for detecting timestamp forgery in NTFS. *Computers & Security* 34 (2013), 36–46. doi:10.1016/j.cose.2012.11.003

[4] Gyu Sang Cho. 2014. An Intuitive Computer Forensic Method by Timestamp Changing Patterns . In *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. IEEE Computer Society, Los Alamitos, CA, USA, 542–548. doi:10.1109/IMIS.2014.92

[5] Xiaoqin Ding and Hengming Zou. 2011. Time based data forensic and cross-reference analysis. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC '11)*. Association for Computing Machinery, New York, NY, USA, 185–190. doi:10.1145/1982185.1982227

[6] Lisa Marie Dreier, Céline Vanini, Christopher J. Hargreaves, Frank Breitinger, and Felix Freiling. 2024. Beyond timestamps: Integrating implicit timing information into digital forensic timelines. *Forensic Science International: Digital Investigation* 49 (2024), 301755. doi:10.1016/j.fsidi.2024.301755 DFRWS USA 2024 - Selected Papers from the 24th Annual Digital Forensics Research Conference USA.

[7] Michael Galhuber and Robert Luh. 2021. Time for Truth: Forensic Analysis of NTFS Timestamps. In *Proceedings of the 16th International Conference on Availability, Reliability and Security* (Vienna, Austria) *(ARES '21)*. Association for Computing Machinery, New York, NY, USA, Article 44, 10 pages. doi:10.1145/3465481.3470016

[8] Pavel Gladyshev and Ahmed Patel. 2004. Finite state machine approach to digital event reconstruction. *Digital Investigation* 1, 2 (2004), 130–149. doi:10.1016/j.diin.2004.03.001

[9] Jan Gruber and Merlin Humml. 2023. A Formal Treatment of Expressiveness and Relevanceof Digital Evidence. *Digital Threats* 4, 3 (2023), 1 – 16. doi:10.1145/3608485

[10] Jan Gruber, Merlin Humml, Lutz Schröder, and Felix C. Freiling. 2023. Formal Verification of Necessary and Sufficient Evidence in Forensic Event Reconstruction. In *Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU'23)*, Edita Bajramovic and Ricardo J. Rodríguez (Eds.). dfrws.org, Bonn, Germany, 1–11.

[11] Thomas Göbel and Harald Baier. 2018. Anti-forensics in ext4: On secrecy and usability of timestamp-based data hiding. *Digital Investigation* 24 (2018), S111–S120. doi:10.1016/j.diin.2018.01.014

[12] David-Olivier Jaquet-Chiffelle and Eoghan Casey. 2021. A formalized model of the Trace. *Forensic Science International* 327 (2021), 110941. doi:10.1016/j.forsciint.2021.110941

[13] István Mező. 2019. *Combinatorics and Number Theory of Counting Sequences* (1 ed.). Chapman and Hall/CRC, New York, NY, USA. doi:10.1201/9781315122656

[14] Christopher Neale. 2023. Fool me once: A systematic review of techniques to authenticate digital artefacts. *Forensic Science International: Digital Investigation* 45 (2023), 301516. doi:10.1016/j.fsidi.2023.301516

[15] OEIS Foundation Inc. 2025. A000670: Fubini numbers. https://oeis.org/A000670. Accessed: 2025-09-17.