What is essential data in digital forensic analysis?

Felix Freiling, Michael Gruhn
Department Computer Science
Friedrich-Alexander University Erlangen-Nürnberg (FAU)
Erlangen, Germany
{*firstname.lastname*}*@cs.fau.de*

Abstract

In his seminal work on file system forensic analysis, Carrier defined the notion of *essential data* as "those that are needed to save and retrieve files." He argues that essential data is therefore more trustworthy since it has to be correct in order for the user to use the file system. In many practical settings, however, it is unclear whether a specific piece of data is essential because either file system specifications are ambiguous or the importance of a specific data field depends on the operating system that processes the file system data. We therefore revisit Carrier's definition and show that there are two types of essential data: strong and weak. While *strongly essential* corresponds to Carrier's definition, *weakly essential* refers to application specific interpretations. We empirically show the amount of strongly and weakly essential data in DOS/MBR and GPT partition systems, thereby complementing and extending Carrier's findings.

Keywords

file system, forensic investigations, operating systems

I. Introduction

A. Motivation

A central role of forensic science is to support the juridical system in the analysis of evidence. In forensic computing, evidence often comes in the form of persistent data stored on hard disks. The interpretation of such *digital* evidence is non-trivial because (at least theoretically) digital data can be manipulated perfectly. But the mere fact that evidence is digital does not mean that it cannot be trusted at all.

Consider for example a simplified partition table entry within the boot sector of a hard disk (see Fig. 1). It consists of a reference to the first sector of the partition and the length of the partition in sectors. Furthermore, it provides an identifier for the "type" of the partition and a bootable flag to indicate whether the partition holds a bootable operating system. While it is possible to manipulate all parts of the entry using a disk editor, changes will have different consequences. For example, changing the bootable flag may only affect the behavior of the computer if it wants to boot from the partition and so changing the bootable flag from false to true may not affect the behavior of the computer at all. However, if the reference to the first sector of the partition is changed, the boot loader/BIOS cannot locate the beginning of the partition. In consequence this means that booting from that partition or mounting it will necessarily fail. A user of that partition must set the field to its original value, which is rather cumbersome. It can therefore be argued that the value of the starting LBA field is more trustworthy than the bootable flag.

The observation that some data fields can be trusted more than others was made by Carrier [1]. Carrier [1, p. 176] defines the term "essential file system data" as follows (emphasis maintained):

> *Essential file system data* are those that are needed to save and retrieve files. Examples of this type of data include the addresses where the file content is stored, the name of a file, and the pointer from a name to a metadata structure. *Non-essential file system data* are those that are there for convenience but not needed for the basic functionality of saving and retrieving files. Access times and permissions are examples of this type of data.

Carrier argues that it is important to differentiate between these two types of data because essential data is more trustworthy than non-essential data: essential data needs to be correct because

> [...] otherwise the person who used the system would not have been able to read the data. [1, p. 12]

Consequently, Carrier classified every data field in file system data structures as being either essential or non-essential.

Start LBA	Length	Type	Bootable

Figure 1. Typical partition entry in partition table.

B. Problem statement and contribution

While the definition of essential data may be intuitively clear, there are many cases where the meaning of a field remains unclear despite the definition. One main problem is that it depends on the operating system how the data is interpreted. While one operating system may respect the bootable flag and only boot into those that officially have their bootable flag set, another operating system may offer to boot any one of all partitions from within the boot loader. Being this so, is then the bootable flag essential or non-essential?

In this paper, we revisit Carrier's notion of essential data and show how to fit application/operating system specific data into Carrier's terminology. We conducted a sequence of experiments regarding the data fields stored in the standard DOS/MBR and GPT partition systems: We systematically changed the value of the field and observed the behavior of different operating systems thereafter on that data. The resulting behavior patterns were not always the same over different operating systems, confirming the fact that essential data depends on the operating system. We argue that there are two types of essential data: strong and weak. While *strongly essential* corresponds to Carrier's definition, *weakly essential* refers to application/OS specific interpretations. We empirically show the amount of strongly and weakly essential data in DOS/MBR and GPT partition systems, thereby complementing and extending Carrier's findings.

C. Related work

In digital forensic science there exist various classifications of evidence. Many of the categorizations of digital forensics are borrowed from physical evidence. For example, Lee and Harris divided evidence into two distinct groups, namely "transient evidence" and "pattern evidence" [2]. They further defined seven criteria by which evidence can be categorized: the kind of crime the evidence indicates (murder, theft, etc.), the kind of material the evidence is made of (metal, plastic, etc.), the aggregation state of the evidence (solid, liquid, gaseous), the kind of forensic question answered by the evidence (contact, event reconstruction, exculpatory of a defendant), how the evidence was generated (blood drips vs. whipped blood), and the specific form of the evidence (fingerprint, color, dust, dirt, blood, etc.). Kirk and Thornton defined the class of microscopic evidence [3]. Gross and Geerds were the first to mention the perishableness of evidence in 1985 [4]. These categorizations can, with some modifications, also be applied to digital evidence as well.

As has been mentioned sufficiently often before, Carrier 2005 defined essential and non-essential data, but he did this without recurring to traditional classifications. Dewald and Freiling later extended this definition to more general settings that include any type of digital evidence, even volatile data in RAM. Like for Carrier, their definitions remain informal but correspond to Carriers terms more or less directly. They distinguish two distinct groups: *technically avoidable* and *technically unavoidable* evidence [5]. They further suggest categorizing evidence by distance from the crime scene and volatility [5]. In this work we further extend and refine the definition of Carrier.

D. Outline

In this paper we revisit Carrier's definition of essential data (in Sect. II) and define the two new types of essential data in Sect. III. We evaluate our definition on DOS/MBR and GPT partition systems in Sect. IV and discuss the new terminology in Sect. V. We conclude in Sect. VI.

II. Definition of Essential Data by Carrier and its Problems

Brian Carrier coined the term *essential data* in his seminal work on file system forensic analysis [1]. Intuitively, essential data is data that is "needed to save and retrieve files." [1, p. 176] While Carrier does not provide a more formal definition, his text contains multiple examples and statements that clarify the concept.

The standard example for essential data is a pointer value from the metadata of a file to its content (see Fig. 2): This pointer needs to be true, otherwise the user of the system would not be able to read the file. As an example for non-essential data Carrier mentions the last-accessed timestamp of the file. Carrier argues that

> [...] [i]f the time value is never updated, it will not affect the user when she tries to read from or write to the file. Therefore, we should trust the essential data more than the non-essential data because it is required and it is needed for the file system to save and restore files. [1, p. 176]

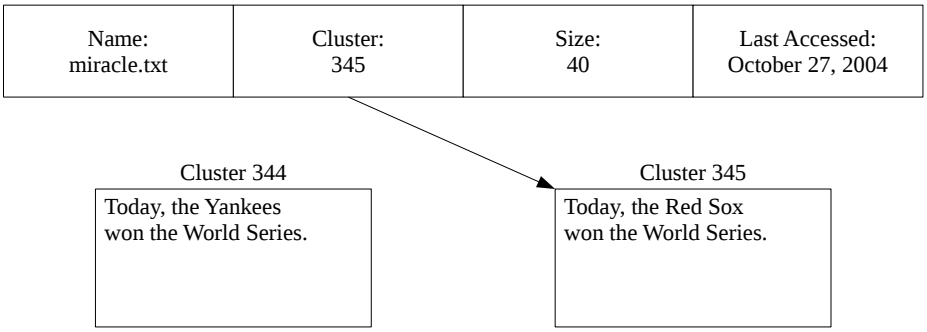

Figure 2. Example of essential and non-essential data [1, Fig. 1.4, p. 13]: the name of the file and the pointer to its content are essential, the last access time is non-essential.

A. Problem 1: Definition depends on assumed basic functionality

In a sense, essential data defines the "core" of the data structures necessary to use the "basic functionality" of a file system. The basic functionality due to Carrier apparently is to manage a set of files on a disk. The introductory quotation at the beginning of this article gives examples for what this basic functionality is [1, p. 176]:

- saving (i.e., writing) a file,
- retrieving (i.e., reading) a file, and
- identifying a file with a name.

In contrast, Carrier explicitly states which data fields are solely "for convenience" and are thus non-essential data. For example, a timestamp or a user ID "is not essential because it does not need to be true" [1, p. 176] to read or write the file since

> [...] the OS may not have enforced the access control settings on a file; therefore, an investigator cannot conclude that a user did or did not have read access to a file. [1, p. 186]

Other examples of non-essential data are file attributes such as *read-only* in FAT. They are non-essential because

> [...] [t]he impact of these being set depends on how the OS wants to enforce them. The read only attribute should prevent a file from being written to, but I found that directories in Windows XP and 98 can have new files created in them when they are set to read only. [1, p. 227]

In other words, if the operating system can ignore the data field and still provide "basic functionality", then the data field is non-essential. If this functionality is merely reading and writing to files, then we are close to Carrier's definition used throughout most parts of his book. However, if it comes to partition systems, basic functionality is different: Carrier states that the "purpose of a partition system is to organize the layout of a volume" [1, p. 72]. This means that the only essential data are those that identify the start and end of each partition:

> A partition system cannot serve its purpose if those values are corrupt or non-existent. All other fields, such as a type and description, are nonessential and could be false. [1, p. 72]

Interestingly, Carrier at one point switches to a totally different "basic functionality" while analyzing data from the *Application category* (such as file system journals):

> In this section, we will refer to the data being essential if they are required for the application-level goal of the feature and not if they are required for storing data. [1, p. 339]

The application level goal is not simply reading and writing to files, but rather they "are essential for the goal of providing a log of file changes" [1, p. 392]. In several other places, application-level features are used to vary the notion of essential data, e.g., in context of the `$STANDARD_INFORMATION` attribute in NTFS [1, p. 282, p. 316].

So overall, Carrier's definition is relative to what basic functionality one assumes.

B. Problem 2: Definition depends on application

Carrier also observes that the basic functionality (and therefore the definition of essential data) may be dependent on the "application". The term application refers to any software accessing the data on disk, but most often it refers to an operating system (OS):

> Some OSes may require a certain value to be set, but that does not mean it is essential. For example, a very strict (and fictitious) OS might not mount a file system that has any files with a last access time that is set in the future. Another OS may not have a problem with the times and will mount the file system and save data to it. Microsoft Windows requires that all FAT file systems start with a certain set of values even though they are used only when the file system is bootable. Linux, on the other hand, has no requirements for those values. [1, p.176]

So the fact that a specific value needs to be set for a particular operating system to boot the system does not mean it is essential. Carrier argues as follows:

> [...] knowing the OS that wrote to the file system is just as important as knowing the type of file system. When discussing file recovery, it is not enough to ask how to recover a file from a FAT file system. Instead, ask how to recover a file that was deleted in Windows 98 on a FAT file system. Many OSes implement FAT file systems, and each can delete a file using different techniques. For example, most OSes do the minimal amount of work needed to delete a file, but there could be others that erase all data associated with the file. In both cases, the end result is a valid FAT file system. [1, p. 240]

So a data field is essential only if *all* operating systems *have to* use it in order to provide basic functionality.

C. Problem 3: Definition cannot deal with redundant information

When analyzing the flags in the `$STANDARD_INFORMATION` attribute in NTFS, Carrier makes an interesting observation that points to a third ambiguity of his notion of essential data:

> Many of these flags [e.g., encrypted, hidden, read-only, etc.] are the same as were seen with FAT, and a description of them can be found there. The flags for encrypted and sparse attributes are also given in the attribute headers, so I consider them to not be essential in this location. This is debatable, though, because another person could claim that this flag is essential and the MFT entry header values are not essential. [1, p. 361]

In other words, if an important data field appears redundantly in multiple places and both fields can be used to provide basic functionality, then it is "debatable" which one is essential and which one is non-essential.

III. What is essential data?

Given the ambiguities of Carrier's informal definition, we now revisit the concept of essential data and try to approach its different meanings by using a simple formal notation.

As observed in Section II, the notion of essential data depends on the "basic functionality" and the "application". Different basic functionalities can be characterized as a set of operations that are "basic" and assumed vital for the task at hand. For example, the "standard" basic functionality throughout most parts of Carrier's book is the set

$$\{\text{store a file}, \text{retrieve a file}\}$$

but there may be different sets in different contexts. We define a set B of basic functionalities as

$$B = \{b_1, b_2, b_3, \ldots\}.$$

Next to a basic functionality, the definition of essential data is relative to the set of applications A which we denote as

$$A = \{a_1, a_2, a_3, \ldots\}.$$

The applications can be operating systems, database systems, or in general every computer system that accesses a data structure on disk that makes up a partition table, a file system or a database. The data structure consists of several *data fields*. Examples for data fields have been mentioned above, e.g., the file name, pointers from metadata to file content, access timestamps, the start address of a partition in a partition table, or a magic value in a file header. It is on the level of data fields that we want to determine essential data. For simplicity, we abstract from the semantics of these data fields and simply denote by F the set of all data fields of interest.

The combination of basic functionality and application uniquely defines the context in which we can precisely decide whether a data field is essential or not. The idea of our definition is to construct general notions of essential data from individual observations of how an application behaves when trying to provide some basic functionality using a specific data field. More precisely, we define a boolean *evaluation function* E with the following signature

$$E : B \times A \times F \mapsto \{\text{true}, \text{false}\}$$

which for a given behavior b, application a and field f returns whether the field f is necessary by a to correctly provide b.

It is important to note that E can be evaluated by experiment, i.e., by choosing a concrete application a and looking how it behaves after manipulating some data field f. So, for example, if application a (e.g., Windows XP) can provide b (e.g., access to file content) even if field f (e.g., the pointer to file content) is not present or malformed, then $E(b, a, f) = \text{false}$. In this case we say that a evaluates negatively on f, i.e. the field f is non-essential to a to provide b. If, however, the application a cannot correctly access the data, not access the data at all, or even stops working or crashes, then $E(b, a, f) = \text{true}$. In this case we say that a evaluates positively on f, meaning that the field f is essential for application a. to provide behavior b.

We now define what it means for a data field to be essential data. We distinguish between application dependence and basic functionality dependence and define two different notions of essential data. Let us fix some basic functionality $b \in B$ and some data field $f \in F$ and consider a fixed set A of applications.

Definition 1. *We say that* f is strongly essential w.r.t. A and b *iff all applications in* A *evaluate positively on* f*, i.e., iff* $\forall a \in A : E(b, a, f)$.

The term strongly essential tries to precisely capture Carrier's definition of essential data. Because all applications require the data field, it seems logical that this data field is absolutely necessary for the data structure to function. However, it may also be the case that some application evaluates positively on the data field and some negatively. This captures an application-dependent form of essential data: Roughly spoken, the field is essential for some applications but not essential for others. This results in the notion of *weakly essential data*.

Definition 2. *We say that* f is weakly essential w.r.t. A and b *iff some applications in* A *evaluate positively on* f *and some evaluate negatively. Formally, the following condition has to be satisfied:*

$$\exists a \in A : E(b, a, f) \wedge \exists a \in A : \neg E(b, a, f)$$

Intuitively, the term weakly essential comprises data that only some applications require to correctly process the data structure. Often these are fields that are specified to be of a specific form by the data structure specification, but besides the specification stating so, there is no reason for these fields to contain the specified data. Examples of such weakly essential data only required by the specification are checksums and informational fields.

Finally, we define what it means for a data field to be neither strongly nor weakly essential.

Definition 3. *We say that* f is non-essential w.r.t. A and b *iff all applications in* A *evaluate negatively on* f*, i.e., iff* $\forall a \in A : \neg E(b, a, f)$.

The term non-essential includes all data fields that no application requires to function correctly. Examples for this are meta data, file contents, slack space or padding. This is usually content that the user of the data structure is free to decide on (like file content), or parts which are not considered part of the data structure at all (such as slack space or padding).

Note that a data field can either be strongly essential or weakly essential but not both. This corresponds to Carrier's concept of essential data since in his view, data was either essential or non-essential, only that we now allow to distinguish application-specific types of essential data.

Also note that our definition is relative to the basic functionality. We believe that this is a defining aspect of essential data: As long as two basic functionalities are incomparable, so must be the notions of essential data for these functionalities. This is also reflected by Carrier's statements.

IV. Evaluation

We now evaluate our definitions of essential data by fixing basic functionality and a set of applications and then empirically running the application on a file system for which different data fields were changed or manipulated. We therefore tried to systematically reproduce the findings of Carrier [1], which we were not always able to. The

set of applications consisted of Windows XP, Windows 7 and Ubuntu 12.04, and we systematically evaluated the data fields of the DOS/MBR and GPT partition systems. For these parameters, we attempted to reliably reproduce deterministic behavior of operating systems when certain data fields were changed. In a sense, we tried to recreate the evaluation function E upon which the general notion of strongly/weakly essential data is based. The analysis was performed by 19 students taking part in a course on digital forensics for computing professionals.

The set of basic functionality in our experiment was the ability to access or mount a partition (in order to store and retrieve files). This is the default functionality B in the following. However, in some cases we observed that the operating systems behaved differently regarding other functionalities. In this case we qualify our results with a footnote specifying the specific (non-default) functionality in question.

We present our results as tables listing the data fields in the MBR and GPT headers and partition entries, with each data field being marked as essential or not for each tested operating system. More formally, the tables show the boolean values of the evaluation function for the specific combination of functionality and application. For comparison, we also always note the classification by Carrier [1]. Based on the results of the evaluation, the last column denotes whether the data field is strong (SE), weakly (WE) or non-essential (NE).

A. DOS/MBR

Bytes			Data Field f	Essential [1]	Essential Linux	Essential Windows XP and 7	Type
0	-	424...446	Boot Code	No	No*	No*	NE
440	-	443	Unique MBR Disk Signature	No§	No	No	NE
444	-	445	UEFI Unknown	No§	No	No	NE
446	-	461	1st Partition Table Entry	Yes	Yes†	Yes†	SE
462	-	477	2nd Partition Table Entry	Yes	Yes†	Yes†	SE
478	-	494	3rd Partition Table Entry	Yes	Yes†	Yes†	SE
494	-	461	4th Partition Table Entry	Yes	Yes†	Yes†	SE
510	-	511	Boot Signature	No	Yes	Yes‡	WE
512	-	Logical Blocksize	Reserved	No	No	No	NE

*Only essential for booting from the device.
§Part of boot code and not explicitly defined by Carrier.
†Otherwise the partition is not recognized.
‡Otherwise the Microsoft Windows wants to repair the drive containing the partition.

Table I
MBR DATA FIELDS [6, TABLE 13. LEGACY MBR] AND THEIR TYPE.

Table I gives an overview of entries in the Master Boot Record (MBR) and their categorization. As can be seen, only the partition table entries are strongly essential, while the boot code is non-essential, unless the application is supposed to boot from the device. But even in that case the boot code can be whatever the application decides and any other application should not be hindered to access the partitions by different boot code. The same is true for the disk signature and the "UEFI unknown value", which are also user controllable.

The boot code is non-essential for accessing/mounting the partition, but for another basic functionality, namely booting from the volume, it is even strongly essential. Since our default functionality is accessing the partition, we refer to the field as non-essential.

The interesting field is the boot signature field. Indeed, an OS can access the partition table and from there the partitions without this value, as shown by Carrier. However, in our evaluation, all three tested OSes required this value to be set. Otherwise they refused to mount any partition, because they considered the MBR to be invalid, which according to the UEFI Specification [6] is correct. Windows XP and 7 even attempted to repair the volume. So given the evidence of Carrier and our own findings, we argue that it is weakly essential.

We now turn to the data fields of an MBR partition entry. As can be seen from Table II only the LBA Addresses are strongly essential, because the CHS Addresses are not used by the OSes. As Carrier already indicated the CHS Addresses are non-essential if the OS uses the LBA Addresses and those are present and correct. With our new levels of essential, we are able to adequately represent those instances where it depends on the application whether a field is essential or not.

To provide backwards compatibility a storage medium partitioned via the GPT system contains a so-called *protective MBR* (PMBR) at LBA 0. This PMBR protects the GPT partitions in the sense that legacy applications

Bytes			Data Field f	Essential [1]	Essential Linux	Essential Windows	Type
0	-	0	Boot-Indicator	No	No*	No*	NE
1	-	3	CHS Start-Address	Yes§	No	No	WE
4	-	4	Partition Type	No	Yes†	Yes†	WE
5	-	7	CHS End-Address	Yes§	No	No	WE
8	-	11	LBA Start-Address	Yes§	Yes	Yes	SE
12	-	15	Size in LBA-Blocks	Yes§	Yes	Yes	SE

*Only essential for booting from Partition
§Either CHS Addresses or LBA Addresses must be provided.
†Essential for automatically mounting the partition.

Table II
MBR partition table entry data fields [6, Table 14. Legacy MBR Partition Record] and their type.

are prevented from accidentally accessing the disk areas containing the GPT Partitions as "used". To this end the PMBR contains one partition of type EFI (0xee). The rest of the partition entries is zeroed. The essential data of this PMBR with regard to the tested operating systems is the same as the classical MBR, hence not explicitly listed.

B. GPT Header

Bytes			Data Field f	Essential [1]	Essential Linux	Essential Windows 7	Type
0	-	7	EFI Signature "EFI PART"	No	Yes*	Yes*	WE
8	-	11	Revision	Yes	Yes	Yes	SE
12	-	15	GPT Header Size in Bytes	Yes	Yes	Yes	SE
16	-	19	Header CRC32-Checksum	No	Yes	Yes	WE
20	-	23	Reserved (zeroed)	No	No	No	NE
24	-	31	LBA of Header (self-reference)	No	No	Yes	WE
32	-	39	LBA of GPT Backup Header	No	No	No	NE
40	-	47	LBA of 1st Block of Partition Area	Yes	Yes	Yes	SE
48	-	55	LBA of last Block of Partition Area	No	No	No	NE
56	-	71	Disk GUID	No	No	No	NE
72	-	79	LBA of 1st Block of Partition Table Entries	Yes	Yes	Yes	SE
80	-	83	Number of Partition Table Entries	Yes	Yes†‡	Yes‡	SE
84	-	87	Size of Partition Table Entries	Yes	Yes	Yes	SE
88	-	91	CRC32-Checksum of Partition Table Entries	No	Yes	Yes	WE
92	-	Blocksize	Reserved (zeroed)	No	No	No	NE

*Without the EFI Signature the PMBR is used.
†Setting the Number of Partition Table Entries to zero leads to a Blue Screen of Death in Windows 7.
‡Must be greater than the partition number to be used.

Table III
GPT Header [6, Table 17. GPT Header]

Table III shows our empirical results for the GPT header. Because Windows XP does not support EFI and GPT it was not tested and hence is not listed. As can be seen from the table, again all three types of essential data are present. The main reason for the amount of weakly essential data is again due to a specification requiring a field but applications non necessarily relying in it. According to the UEFI Specification [6, 5.3.2 GPT Header] the EFI Signature, Header CRC32-Checksum, LBA of Header and CRC32-Checksum of partition table entries must all be valid. But as Carrier already indicates, a GPT partitioned drive can be used without these fields being valid.

An interesting fact is that the LBA of GPT backup header is non-essential, even though the UEFI Specification [6, 5.3.2 GPT Header] clearly states that the GPT Backup Header must also be checked for validity. Obviously this is not done by any tested operating systems.

V. Discussion

We now discuss the benefits and shortcomings of our definition.

A. Usefulness of new definitions

Strongly essential data must be relied upon, because by definition it is the minimum amount of data needed to correctly access a data structure according to its specification. If less data is required to access the data structure this would violate our definition of strongly essential.

In theory with an infinite number of applications, this definition would exactly correspond to Carrier's definition, because if a field is essential according to Carrier there cannot exist an application that is able to correctly interpret the data structure without this field. However, if such an application a_w would exist then the original premise of the field being absolutely necessary to access the data structure could be dismissed because the a_w would be a witness against the necessity of the data field. Hence, strongly essential data w.r.t. the set of all (thinkable) applications is exactly the minimal data necessary for a data structure to function correctly (with respect to a certain basic functionality, or course).

As stated earlier weakly essential data, is often data that is required by the specification, but not necessary at all to access the data structure. One example is the MBR Boot Signature. Even though in our tests all OSes required the signature, it is obviously possible to construct an OS that does not require this signature. If weakly essential data is encountered during an analysis, care must be taken to consider the circumstances correctly, e.g., as can be seen from Table I neither Windows nor the tested Linux version recognized the partition table as valid, hence, this fact could be exculpatory.

In theory, non-essential data should only comprise data fields that are by specification allowed to be *completely controlled by the user* of the data structure. If an application would exist that imposes a restriction on the data field content (e.g., assumes a particular value to be present), that application would violate the specification of the data structure and therefore would have to be considered broken. For Carrier too, the fact that data is fully user-controllable is a good indicator of something being non-essential:

> Technically, any file that an OS or an application creates could be designed as a feature in a file system. For example, Acme Software could decide that its OS would be faster if an area of the file system were reserved for an address book. Instead of saving names and addresses in a file, they would be saved to a special section of the volume. This might cause a performance improvement, but it is not essential for the file system. [1, p. 205]

B. Trust hierarchy

With our new notions of essential data, we are able to adequately represent those instances where it depends on the application whether a field is essential or not, which allows us to define a hierarchy of trustworthiness:

- Strongly essential data must be trusted. They are generally not manipulateable without impacting the correct functionality of the data structure.
- Weakly essential data can be trusted for specific applications. Because not all applications require this data to be specifically formated, it is possible that this data is manipulated.
- Non-essential data cannot be trusted. Non-essential data is in general either user content, that is data which can be freely chosen by the user of the data structure, or "wasted" space within the data structure, that serves no data storage purpose, such as slack space or padding. However, non-essential data can become essential if "higher level" basic functionalities (functionality referring to user applications like browsers or word processors) are considered.

C. Evidence hierarchy

Besides a hierarchy of trustworthiness of data, a hierarchy of where evidence is most likely stored can be specified:

- While strongly essential data can be evidence in itself, it cannot "contain" evidence. This is because the strongly essential data's content is completely defined by the data structure and is not freely user influencable.
- Weakly essential data can be used to store user content, given it is not used by an application which evaluates positively on that data field. So for weakly essential data the case circumstances, i.e., the set of installed applications, must be considered in order to judge whether a data field contains user data that can be used as evidence.

- Non-essential data can be used freely to store any data. Hence, this data must in any case be analyzed. During a forensic investigation, even fields that the data structure specification does not explicitly tag as user storage must be analyzed for hidden data if the field is non-essential.

VI. Conclusions and future work

Carrier's notion of essential data is important to understand the trustworthiness of evidence found in file systems. In this paper, we revisited Carrier's definition and distinguished a "pure" notion (consistent across all possible contexts) and an "application-dependent" notion of his term. With our evaluation, we therefore followed a recommendation by Carrier to verify and test specific behavior of applications as stated in his book:

> [...] it is important for an investigator to verify and test the application-specific behavior in the context of a specific incident. [1, p. 176]

In future work, we need to extend our analysis to other and other more complex data structures. Interesting structures are file systems, such as FAT, NTFS and EXT. However, also memory structures such as kernel process structures could be evaluated. However, we would then need to also take general notions like *technically avoidable* and *technically unavoidable* evidence [5] into account.

Further possible research needs to be done on data fields which have a logical dependency on each other. One example for this would be the GPT Backup Header. It is not used if a valid GPT Header is defined. This means that the GPT Backup Header could be considered non-essential. If no valid GPT Header exists the GPT Backup Header is used. This would make the GPT Header non-essential. However, both cannot be non-essential at the same time. At least one of them needs to be defined, making at least one strongly essential, but not both. Ways to formalize this need to be found.

Acknowledgment

This work was suppored by the German Federal Ministry of Education and Research (BMBF) as part of the project "Open Competence Center for Cyber Security (Open C3S)". The authors would like to thank the 19 students participating in the 2014 Open C3S course "Grundlagen digitaler Forensik", which performed the main bulk of the analysis as part of their course exercises. We would also like to thank Andreas Dewald for helpful input to the formalization of the problem. Further we would like to thank the anonymous reviewers (especially Reviewer 1) for their constructive feedback and suggestions.

References

[1] B. Carrier, *File System Forensic Analysis*. Addison-Wesley Professional, 2005.
[2] H. Lee and H. Harris, *Physical Evidence in Forensic Science*. Lawyers & Judges Publishing Company, 2000.
[3] P. Kirk and J. Thornton, *Crime Investigation*. Wiley, 1974.
[4] H. Gross and F. Geerds, *Handbuch der Kriminalistik: Wiss. u. Praxis d. Verbrechensbekämpfung. Kriminaltaktik. Die Organisation der Verbrechensbekämpfung*. Pawlak, 1985, no. v. 2.
[5] A. Dewald and F. Freiling, *Forensische Informatik*. Books on Demand, 2013.
[6] *Unified Extensible Firmware Interface Specification*, 2nd ed., Unified EFI, Inc., June 2013. [Online]. Available: http://www.uefi.org/specs/download