Digital Investigation xxx (2017) 1–11

Contents lists available at ScienceDirect

Digital Investigation

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

A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics

Kai Shi [a], Ming Xu [a, b, *], Haoxia Jin [a], Tong Qiao [b], Xue Yang [a], Ning Zheng [a], Jian Xu [a], Kim-Kwang Raymond Choo [c]

[a] School of Computer Science and Technology, Hangzhou Dianzi University, HangZhou, China
[b] School of Cyberspace, Hangzhou Dianzi University, HangZhou, China
[c] Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA

ARTICLE INFO

Article history:
Received 10 March 2017
Received in revised form
14 August 2017
Accepted 29 August 2017
Available online xxx

Keywords:
GPS forensics
NMEA
Metadata-based recovery
File carving
Trajectory reconstruction

ABSTRACT

Globe positioning system (GPS) devices are an increasing importance source of evidence, as more of our devices have built-in GPS capabilities. In this paper, we propose a novel framework to efficiently recover National Marine Electronics Association (NMEA) logs and reconstruct GPS trajectories. Unlike existing approaches that require file system metadata, our proposed algorithm is designed based on the file carving technique without relying on system metadata. By understanding the characteristics and intrinsic structure of trajectory data in NMEA logs, we demonstrate how to pinpoint all data blocks belonging to the NMEA logs from the acquired forensic image of GPS device. Then, a discriminator is presented to determine whether two data blocks can be merged. And based on the discriminator, we design a reassembly algorithm to re-order and merge the obtained data blocks into new logs. In this context, deleted trajectories can be reconstructed by analyzing the recovered logs. Empirical experiments demonstrate that our proposed algorithm performs well when the system metadata is available/unavailable, log files are heavily fragmented, one or more parts of the log files are overwritten, and for different file systems of variable cluster sizes.

Introduction and contributions

In our daily life, Global Navigation Satellite System (GNSS) indeed plays an important role for providing autonomous geo-spatial positioning information. To our best knowledge, GNSS is mainly referred to as Global Positioning System (GPS), Glonass or the future-used Galileo. In this practical context, portable and widely-used GPS devices are mainly studied. For instance, drivers use GPS devices when driving in an unfamiliar area (Zheng et al., 2009). Besides, data stored on GPS devices are rich sources of information in a forensic investigation. For example, those data such as timestamp, position (latitude and longitude) and velocity from GPS devices could be used to reconstruct an event and link a device (and vehicle) to a specific place at a specific time, as well as determining the velocity of a vehicle at a certain place (Van Eijk and Roeloffs, 2010). Moreover, the analysis of trajectory data from GPS devices semantically will allow us to have a better understanding of the driver's daily activities and identify unusual driver's behaviors (e.g. a route deviation) (Liao et al., 2015). Similar to other areas of digital forensics, in GPS forensics we seek to identify, preserve and analyze data from the use of GPS devices (e.g. deleted data). For example, after a hit-and-run incident, the guilty driver could attempt to delete GPS data from the vehicle's navigation system or tamper with the GPS data to provide a false alibi at the time of the incident. Thus, it is important for forensic investigators to be able to forensically recover evidence from these devices on which this paper mainly focuses.

State of the art

The National Marine Electronics Association (NMEA) 0183 protocol (Wikipedia and Nmea 0183) is generally used in the design of GPS devices. Since the GPS data received by GPS receivers (referred to as NMEA data) comply with the NMEA 0183 standard, those data should also exist in the Random Access Memory (RAM) of GPS devices. For example, the authors of (Van Eijk and Roeloffs,

* Corresponding author. School of Computer Science and Technology, Hangzhou Dianzi University, HangZhou, China
E-mail address: 575913857@qq.com (M. Xu).

http://dx.doi.org/10.1016/j.diin.2017.08.004
1742-2876/

2010) proposed an approach referred to as TomCopy to dump the RAM of TomTom devices and extracted some NMEA data from the acquired RAM image. Besides, the authors of (de Sousa and Gondim, 2016) focused on the RAM of Android devices and found these devices also utilized NMEA 0183. Then, they both attempted to reconstruct trajectories by analyzing the NMEA data found in the device's RAM.

In addition to the volatile memory (i.e. RAM), NMEA data could also be stored as logs in the non-volatile memory of GPS devices. For example, the authors of (Cusack and Simms, 2011) examined the non-volatile storage of Navman devices and recovered a number of NMEA logs using EnCase.[1] However, it should be noted that the recovery algorithm in Encase relies on the existence of file system metadata. Since file deletion only marks the physical location of a file as being available for storing new information, the metadata-based recovery algorithm can recover deleted files according to the information (still present in file allocation table) linking to the clusters of deleted file (Carrier, 2005). However, in the event that file system metadata is no longer present, corrupt, or has been deliberately removed (e.g. a damaged disk, an incomplete forensic image, or a partially overwritten file), then the data even though has not been overwritten will not be able to be recovered using the metadata-based recovery algorithm. Thus, there is a need for a novel technique to recover NMEA logs without relying on system metadata that can be used to reconstruct trajectories from GPS devices.

In this context, the authors of (Pal and Memon, 2009) explained the importance of file carving technique to recover files, particularly in the absence of file system metadata. File carving can be generally divided into three categories, namely:

- **Contiguous files carving:** One key focus of file carving has been on recovering contiguous files (Iii and Roussev, 2005). Files with distinct headers (start of file) and footers (end of file) can be easily detected when scanning disk images. Once detected, the file content between the header and footer could be extracted as a new file if the file is contiguous. However, when files are added, deleted and modified, the distribution of files stored in disk becomes complex and they are likely going to be fragmented. Thus, contiguous files carving is likely to result in many false positives that a recovered file is considered as intact but in fact contains invalid data.
- **Fragmented files carving:** In file-signature based carving algorithms such as those reported in (Na et al., 2014; Karresand and Shahmehri, 2008), the special characteristics and intrinsic structures of a specific file are used to reassembly file fragments into intact files. The author of (Cohen, 2007) in his submission to the DFRWS 2007 challenge (DFRWS, 2007) described the carving problem as a mathematical construction of a mapping function between file bytes and disk image bytes. In other words, it is required to design a mapping function generator and extract files from the disk image according to the generated mapping functions.
- **Hash-based carving:** Hash-based carving, proposed by the authors of (Garfinkel) and (Garfinkel and Mccarrin, 2015), is designed to detect the presence of specific "target files" on digital media by evaluating the hashes of individual data blocks. Hash-based carving is also designed to identify and recover files that are fragmented, files that are incomplete, or files that have been partially modified.

[1] EnCase is a famous forensic software with maintaining its reputation as the gold standard in criminal investigations.

Contributions

Due to the limitations of existing recovery algorithms and the lack of file carving designed for NMEA logs, we propose a novel recovery approach utilizing the characteristics and intrinsic structure of NMEA data in this paper. The proposed recovery algorithm is vendor-independent (i.e. GPS device brand), as long as the device generates NMEA logs. We regard the contributions of this paper to be three-fold, as follows:

- We first describe the characteristics of NMEA 0183-based log data. This provides a baseline for (new) forensic investigators in analyzing devices generating NMEA logs to extract relevant information (e.g. time and position).
- Based on the file carving technique, we present a novel system (hereafter referred to as GPSDroid) designed to recover and reconstruct trajectories even in the absence of system metadata.
- Since data stored on GPS devices can be subject to modification (e.g. drivers seeking to inject data to provide an alibi), we describe relevant GPS anti-forensic techniques and present counter-measures.

Organization

In the next section, we will briefly describe the NMEA 0183 protocol and its forensic relevance. In Sections The proposed GPSDroid and GPS anti-forensics and counter-measures, we present the system design of GPSDroid and discuss various GPS anti-forensic approaches. We then evaluate the utility of the proposed system in Section Evaluations, before concluding the paper in the last section.

NMEA 0183 protocol

By investigating the characteristics and intrinsic structure of NMEA data, we can smoothly establish our proposed system for realizing GPS forensics. Therefore, let us introduce the detailed description of NMEA 0183 protocol first.

Structure and characteristics

The NMEA 0183 protocol is a set of communication standards developed, maintained and released by the National Marine Electronics Association (NMEA). According to the protocol specification, GPS receivers communicate once per second and their data are presented in simple ASCII text as "sentences". Fig. 1(b) is an example of two consecutive records excerpted from a NMEA log. Each record consists of several sentences observing a specific format (see Fig. 1(a)). In each sentence, it starts with the "$" character and ends with a carriage return and line feed. Then, five letters defining the sentence type are followed by the "$" character. This is followed by the data area, where fields are separated by a comma. The checksum is the last field, and is the bitwise exclusive OR of ASCII codes of all characters between "$" and "*".

While NMEA defines several kinds of sentences, the most widely used by GPS receivers are GGA, GSA, GSV, RMC, VTG and GLL. It should be noted that RMC sentence is extremely important for forensic investigators. As Fig. 1(c) illustrates, the information relating to time, date, latitude, longitude and speed can be extracted according to the RMC sentence's structure.

Forensic relevance

As illustrated in Fig. 2 illustrates, each NMEA log contains pieces of fragments with several records and each record consists of some

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

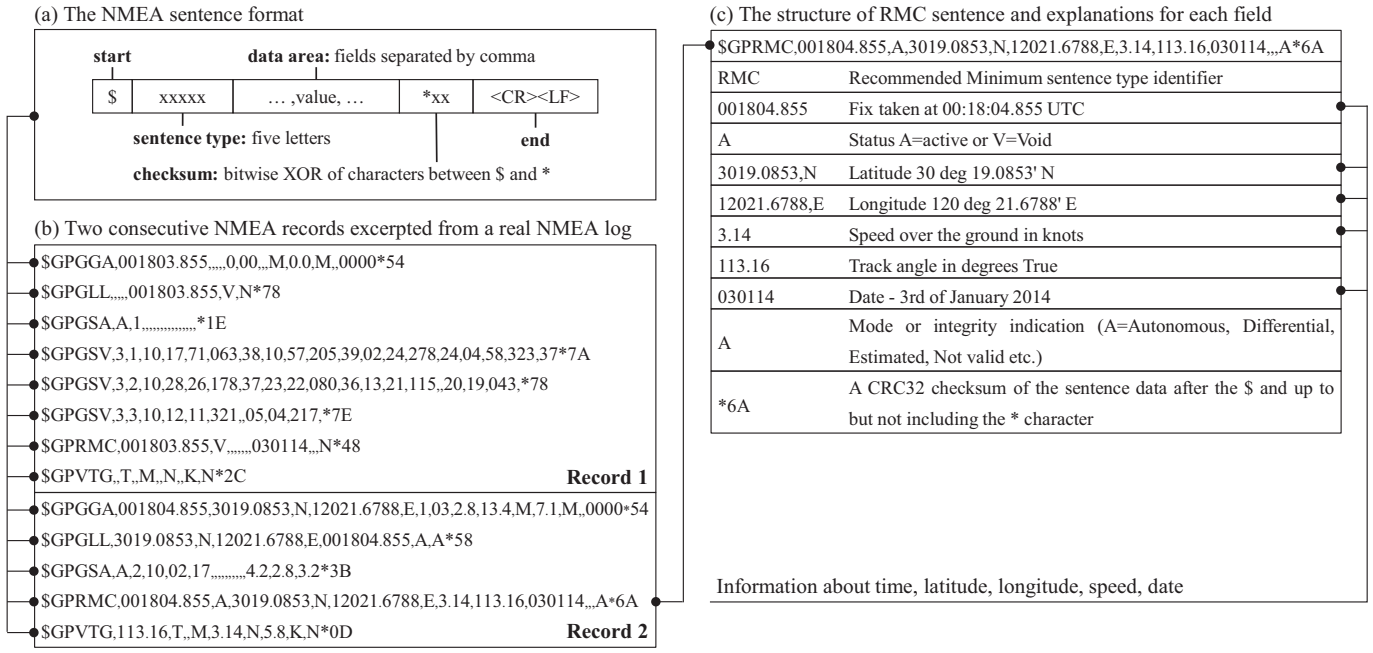


Fig. 1. A detailed illustration of NMEA data.

sentences (e.g. GGA, RMC and GLL). For each RMC sentence, it includes information such as time, position and speed. Thus, this can be expressed as:

$$S = \{r_1, r_2, r_3, \ldots, r_n\} \tag{1}$$

$$r_i = (date_i, time_i, lat_i, lng_i), i = 1, 2, 3, \ldots, n, \tag{2}$$

where a NMEA record is an item in S, and each item r_i is denoted as a chronological track point in the map based on its position. S is then a set of track points, and deleted trajectories can be reconstructed by linking the track points together chronologically. Thus, if we can first recover NMEA logs, the deleted trajectories can be reconstructed by analyzing the recovered log files.

The proposed GPSDroid

In this section, we present the proposed recovery system, GPSDroid. As shown in Fig. 3, GPSDroid consists of four stages, namely:

1) **Image acquisition (of GPS devices):** As NMEA logs are stored in non-volatile storage, there are two potential sources of data, namely: internal flash memory and removable memory card. Then we need the appropriate approaches (e.g. JTAG, Chip-off and forensic software) to dump the memory copy.

2) **Data pre-processing:** Data blocks belonging to NMEA logs should be distinguished from other invalid blocks. Thus, according to the features of NMEA data, the pre-processing operation includes pinpointing all valid data blocks while scanning the acquired image.

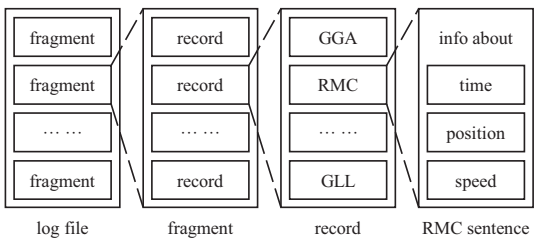


Fig. 2. Structure of NMEA log file.

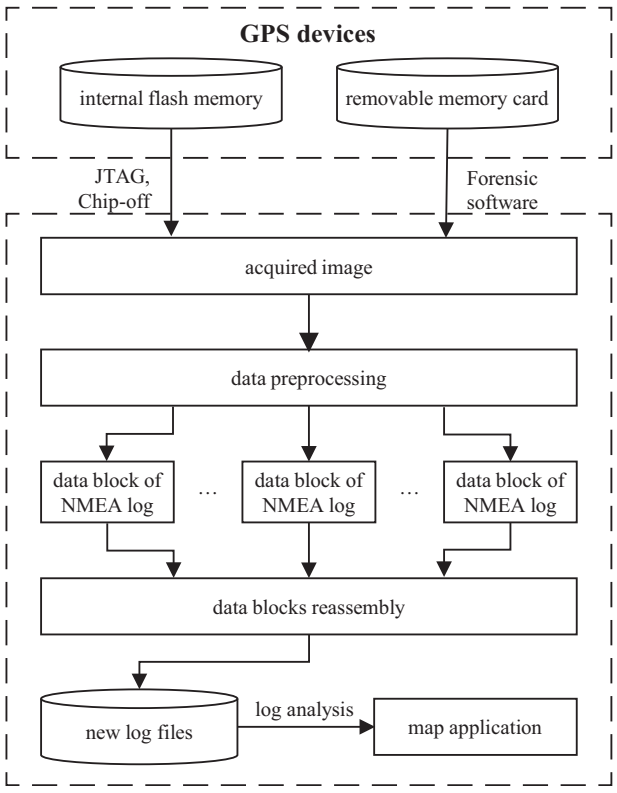


Fig. 3. GPSDroid architecture.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

3) **Data blocks reassembly:** In this stage, we process the pinpointed data blocks and re-assemble them into new files. For this purpose, a discriminator that can determine whether two data blocks can be merged is proposed. And based on the discriminator, we design an algorithm to re-order and re-assemble the obtained data blocks to generate new logs.
4) **Log analysis:** In this phase, we analyze and seek to recover deleted trajectories based on the understanding of NMEA log format.

Image acquisition

Image acquisition of GPS devices is a complex but extremely important step for investigators as it needs to ensure the reliability and integrity of raw data (Boddington et al., 2008). As NMEA logs are stored in non-volatile storage, we need to examine the internal flash memory and removable memory card (e.g. SD or TF card).

For internal flash memory, some GPS devices (e.g. the first generation of TomTom (Elstner and Roeloffs, 2016)) can be connected to a personal computer (PC) using USB port. In this context, we can use forensic imaging software (e.g. Encase and WinHex) to perform the memory dump of GPS devices. While not all GPS devices support mass storage mode, we can utilize JTAG (see (Breeuwsma, 2006)) to debug the processor and dump the flash memory if this device has JTAG interface. In the worst case, if the device does not have a JTAG interface, then we can use the Chip-off method (see (Breeuwsma et al., 2009)) by first desoldering the flash chip from Printed Circuit Board (PCB), and using a chip reader to gain access to the content.

It is relatively easier to acquire a forensic image from removable memory cards, since the latter can be mounted as a mass storage device on PC with a write-protected card reader. In other words, we can utilize forensic software to extract its memory dump.

Data pre-processing

In this phase, we seek to identify all data blocks of NMEA logs using the unique features of NMEA 0183. However, due to the nature of the underlying file system, log data might be scattered/fragmented. Thus, two key points need to be considered as follows:

(1) How to choose the scanning step size?
(2) What features can be utilized?

Choosing the scanning step size

As scanning step size is relevant to the data distribution in disks, forensic investigators need to be familiar with physical layout of a disk used to store the data. Most disks are generally described in terms of data blocks referring to as sectors and clusters, where cluster size = sector size × sectors, and disk size = cluster size × clusters.

In other words, a cluster rather than a sector is the smallest logical amount of disk space allocated to store a file (Poisel et al., 2011). Therefore, when the file system is not able to provide sufficient contiguous clusters to store a file, this file will be fragmented and stored in different physical addresses of the disk (Garfinkel, 2007), which is referred to as file fragmentation in the literature. However, due to the fact that files are typically stored in clusters, fragmentation points can only appear at the boundary between two clusters. Therefore, in order to avoid the errors resulting from fragmentation during the scanning stage, we will choose the cluster size of the acquired image as the scanning step size.

However, since our recovery algorithm does not rely on the existence of system metadata, we will not know the acquired image's cluster size. In the meanwhile, we also remark that using the "wrong" scanning step size may lead to misreading of data blocks. For example, as Fig. 4 illustrates, when we choose 4KB as the scanning step size while the cluster size is 1KB, it will result in the inclusion of both valid data blocks (area A) and dirty data (area B) as blocks of NMEA logs. Therefore, in order to ensure the compatibility of our proposed recovery algorithm, we use 512B as the scanning step size. This is because cluster size is typically in multiples of 512B and each larger cluster can be seen as consisting of smaller data blocks. It is demonstrated in Fig. 4 that while valid data blocks (area A) are pinpointed, the dirty data (area B) are also not included.

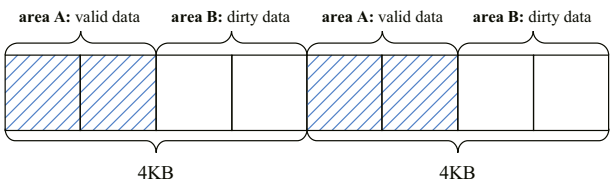
Choosing the features

After choosing 512B as the scanning step size, we will attempt to find a distinct feature in order to determine whether a data block belongs to the NMEA log. Since sentences defined by NMEA 0183 always begin with special characters such as $GPGGA, $GPGSA, $GPGSV, $GPRMC, $GPVTG, and $GPGLL, data blocks including these special characters can be easily identified. Thus, this "special characters" feature can be utilized to effectively identify NMEA data blocks. Besides, when compared with some statistical file fragment classifying methods (Li et al., 2005; Karresand and Shahmehri, 2006), this search approach will be relatively more effective and simpler.

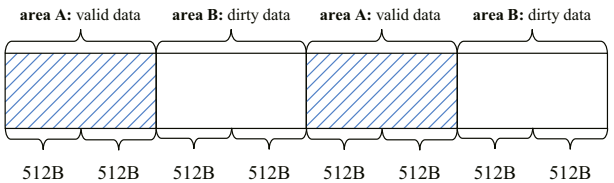
Based on our experimental results and statistical analysis, the size of a NMEA record does not exceed 512 bytes. Therefore, as Fig. 5 illustrates, a record might be stored in a single data block (case 1), across two data blocks (case 2), or fragmented (case 3).

All cases can be identified using the special characters. However, only case (1) contains a complete record where temporal information can be extracted. For cases (2) and (3), relevant temporal information can be extracted after merging two corresponding data blocks. However, it should be noted that the adjacent logical clusters in a file tend to occupy adjacent physical clusters of the disk, and many clusters are probably still contiguous with their neighbors in the event of fragmentation. Thus, it is proposed to assume that the probability of case (2) happening is significantly larger than case (3). And for case (2), if we can guarantee that two contiguous data blocks are consecutive logically, then temporal information can be extracted after merging them.

Based on our investigation of the structure of NMEA 0183, we proposed using the "checksum" feature (the last field of NMEA sentence, see Fig. 1) to determine whether case (2) or case (3)



(a) Choosing 4KB as scanning step size while cluster size is 1KB.



(b) Choosing 512B as scanning step size while cluster size is 1KB.

Fig. 4. Results of different scanning step size choice.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

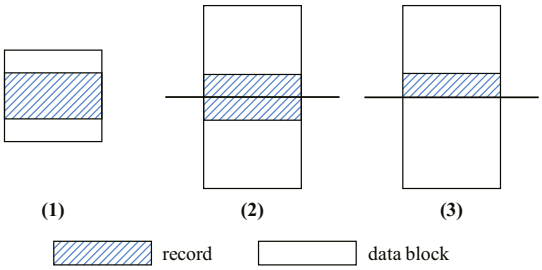


Fig. 5. Three cases of records distribution.

happens. As checksum value is the bitwise exclusive OR of ASCII codes of all characters between $ and *, we can compare it with the manually calculated value to determine whether contiguous data blocks are consecutive logically. For example, for two physically contiguous data blocks, we can extract the last few bytes of the previous data block and the first few bytes of the next data block and join them as a new sentence. Then, we compute the XOR value and compare it with the last field of the joined sentence. If both values are equal, then it means that two data blocks are consecutive logically and temporal information can be extracted after merging them. For the unequal values, it means fragmentation happens at the boundary of this block, i.e. case (3).

To sum up, we present the data pre-processing workflow in Fig. 6. When scanning the acquired image of GPS device, it reads 512 bytes as a data block each time. If the data block contains keywords (i.e. special characters), then we can attempt to extract relevant information including time and position based on the RMC sentence's format. However, if the information cannot be extracted, then we can attempt to merge it with its next neighbor. And if they can match the checksum test, relevant information can be extracted after merging them. On the contrary, if the checksum values are not equal, then the data block will be stored in $List_2$. After scanning the whole image, we can successfully pinpoint all NMEA data blocks.

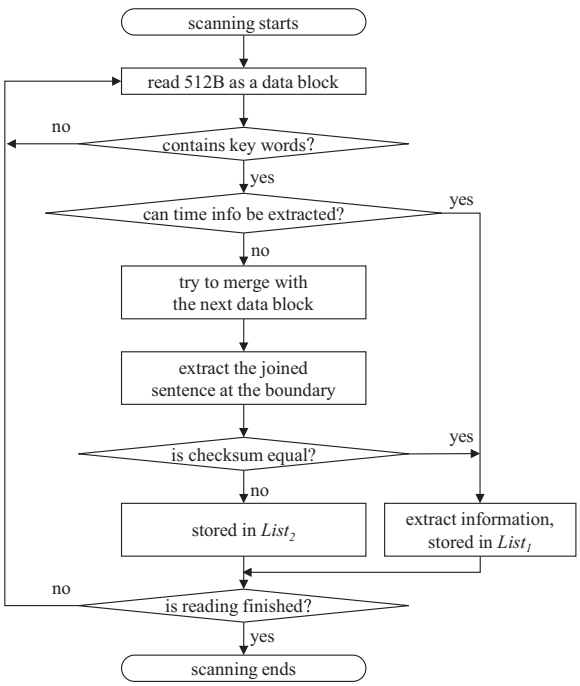


Fig. 6. Flowchart of data pre-processing.

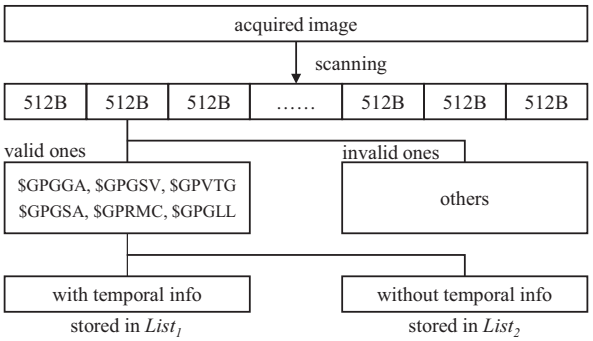


Fig. 7. Resultant data blocks.

Data blocks reassembly

After data pre-processing, all data blocks can be classified as either valid and invalid. Valid data blocks can then be further classified into two categories depending on whether temporal information can be extracted. Fig. 7 presents the classified results of data pre-processing. The obtained data blocks are stored in separate lists: $List_1$ (with temporal information) and $List_2$ (without temporal information).

Since NMEA logs are recorded chronologically, the time sequence should be consistent with the ordering of the data blocks. Thus, all valid data blocks can be reassembled into their correct ordering by using such time information. However, according to the classified results, the data blocks from $List_2$ will not be included because they do not have temporal information. Therefore, a discriminator and a reassembly algorithm are proposed to deal with how to insert data blocks without including any temporal information (from $List_2$) into the chronological data blocks (from $List_1$).

Discriminator to merge two data blocks

In this section, we present a discriminator designed to determine whether two data blocks can be merged. As shown in Fig. 8, four constraints are proposed to exclude the mismatch cases. We assume the following criteria: when both two data blocks match all proposed constraints, these small blocks can be merged into a large

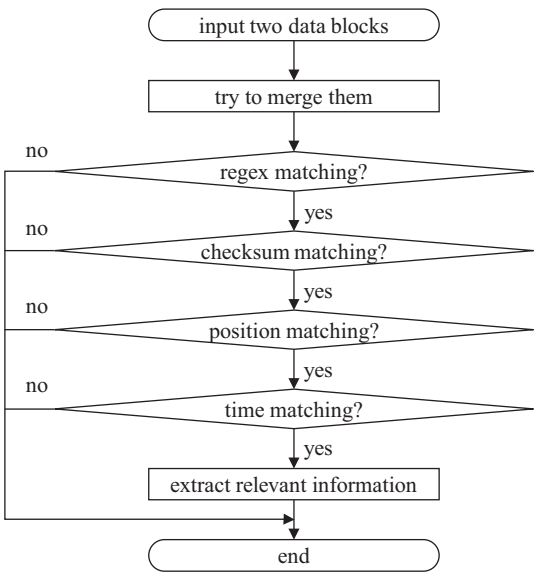


Fig. 8. Design of discriminator.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

block. And the four constrains utilized in the discriminator are as follows:

- **Regex:** If two data blocks are logically consecutive, then the joined sentence consisting of two pieces from two data blocks have to comply with the structure defined in NMEA. Thus, some regexes can be summarized according to the NMEA sentence's structure to exclude those mismatch cases.
- **Checksum:** Checksum is the last field of each sentence. According to the checksum's calculation method defined by NMEA, the calculated value should be equal to the joined sentence's last field.
- **Geographic position information:** Since the moving distance of a vehicle is limited in a short amount of time, the geographic position information of two contiguous clusters are assumed to be consecutive.
- **Temporal information:** Since GPS receivers write file data into NMEA logs continuously, we assume that the time of two contiguous clusters is consecutive.

Reassembly algorithm

Based on the presented discriminator, we design a reassembly algorithm to merge data blocks from $List_2$ into $List_1$. The detailed process is described in Algorithm 1.

Algorithm 1. Reassembly of Fragments.

Input: $List_1$, $List_2$;
Output: $List_1$;
1: sort $List_1$ chronologically;
2: $count = List_1.count + List_2.count + 1$;
3: **while** $count > List_1.count + List_2.count$ **do**;
4: merge blocks of $List_1$ based on discriminator;
5: merge blocks from $List_1$ and $List_2$ based on discriminator;
6: $count = List_1.count + List_2.count$;
7: drop $List_2$;
8: **return** $List_1$;

In our algorithm, as we consider the worst scenario that all data blocks are disordered, there is no spatial relation between data blocks from $List_1$ and $List_2$. In this context, if we utilize a brute force approach to determine each possible merging case, the computation complexity of the brute force is $O(n^2)$. Clearly, it is inefficient when dealing with a large number of data blocks.

Therefore, we present an optimized resolution to reduce the possible merging cases. As data blocks stored in $List_1$ have temporal information, we can first sort them chronologically. Then, in each iteration, we will merge each pair of data blocks in $List_1$ with its neighbor. Since data blocks of $List_1$ are ordered and we attempt to merge only the sorted data block with its neighbor, the computation complexity of this merging process becomes O(n). This is a significant reduction in the number of elements in $List_1$, which reduces the possible merging cases from $List_1$ and $List_2$. Finally, as the data blocks from $List_2$ are gradually merged into $List_1$, all scattered data blocks will be merged into file blocks step-by-step.

Logs reconstruction

After the reassembly algorithm, all valid data blocks are merged into some chunks of NMEA data stored in $List_1$. Since logs are generated based on the relevant date, we can classify the data from $List_1$ and save them in new files. Then, we can successfully recover NMEA logs from GPS devices.

Logs analysis

After recovering NMEA logs from GPS devices, we attempt to reconstruct trajectories by analyzing logs. As we have previously discussed in Section Data pre-processing, each RMC sentence can be transformed into a track point in the map. Thus, we can sort those track points chronologically and utilize mapping applications, such as Baidu Map (Wikipedia and Baidua) to visualize the sorted points. Then, we can successfully reconstruct the deleted trajectories from GPS devices.

GPS anti-forensics and counter-measures

As anti-forensic techniques can be utilized by malicious individuals with knowledge of digital forensics, it will complicate forensic investigation (D'Orazio et al., 2014; Eterovicsoric et al., 2017; Leom et al., 2016; Stamm and Liu, 2011). For example, in the context of this paper, anti-forensic efforts including deleting trajectory data from GPS devices, and tampering of trajectory data to insert a fabricated trajectory or replace the original data with fabricated data. In this scenario, if we ignore the reliability and authenticity of recovered information, we are likely to be cheated by malicious individuals.

In this section, we discuss some possible GPS anti-forensic techniques and design relevant counter-measures to resist against them. In practice, by considering the realistic motives in GPS forensic cases, it is proposed to generally classify GPS anti-forensic techniques into two categories: *data removing* and *data tampering*.

Data removing

It is important to know whether a removing method deletes data completely. Data removing methods can be generally categorized as follows:

- **normal/professional deletion:** It is known that a normal deletion will not remove the content of a NMEA log file, in the sense that it is still possible to recover the file before its content is overwritten. However, some wiping software such as Eraser and CCleaner are designed to remove the file data permanently. However in practice, it has been shown that in the event that wiping software had been used, it could still be possible to recover remnants (Quick et al., 2013).
- **system format:** In Windows operation system, there are two kinds of format, namely: regular format and quick format. Malicious individuals could attempt to perform a regular format to remove NMEA logs. In this scenario, the deleted trajectories might not be able to be reconstructed, as in a regular format, bad sectors will be checked and content of the files removed from the volume. However, it should be noted that quick format is recommended by default in Windows operation system. Moreover, the time taken by quick format is much shorter than regular format because during quick format, the system only removes files from the partition but does not scan the disk for bad sectors. Therefore, most users will prefer quick format in practice. As after a quick format has been performed, the content of deleted files that has not been overwritten could be recovered.

Data tampering

In addition to data removing, malicious individuals may attempt to mislead a forensic investigation by tampering the files, using the following methods:

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

- **typical modification:** For a typical modification, one may open a NMEA log and save his/her modifications into the particular file. By doing so, the last-written timestamp of this log will be updated to the current time of the underlying operating system. In addition to the last-written timestamp, each file has a creation timestamp, which is the time when the file is created. It should be noted that the trajectory data in NMEA logs also contain temporal information. Thus, if the creation timestamp of a NMEA log differs from the first track point's time or the last-written timestamp differs from the last track point's time, the log file is probably tampered. Therefore, a forensic investigator should check whether the timestamps of modified file are consistent.
- **non-intrusive modification:** We define non-intrusive modification as a way of modifying the content of a file without affecting the timestamps of the file. For example, one could use existing software (e.g. WinHex and SleuthKit) to analyze the system metadata of the file to obtain the physical address of the file's content, which would then facilitate the data manipulation. In addition, one could attempt to change the timestamps after a file has been modified. For example, in Windows operating system, one could use the SetFileTime Application Programming Interface (API) to modify a file's timestamp. In this scenario, forensic investigators might be unable to find any traces left by malicious individuals. However, it is still not easy for these malicious individuals because they have to prepare the fabricated NMEA data in advance. In other words, this is not an opportunistic attempt.
- **file splicing:** Another way of data tampering is file splicing, where one prepares a fabricated NMEA log to replace the original log, and modifies the associated timestamps. In other words, an individual seeks to insert a fabricated trajectory (see Trajectory AC in Fig. 9) in order to conceal the real trajectory, say by modifying the logs to change the destination of the original trajectory (Trajectory AB in Fig. 9). If this individual removes the log by simply deleting it, then the fabricated log could be easily detected (see abnormality in Fig. 9). As contents of both deleted file and fabricated log exist on the GPS device's storage, the mixed track points from two logs alternate in time. As the GPSDroid's reconstruction algorithm sorts track points chronologically, the recovered trajectory appears to be impossible jumping.

Evaluations

In this section, we will present the evaluation of our proposed approach. Due to the lack of publicly available GPS dataset, we generates our own GPS dataset using an Eroda (HD-X10) portable navigation device, which runs a Microsoft Windows CE 6.0 operation system. Besides, since EnCase is a widely used commercial forensic software, we will "benchmark" our approach with EnCase.

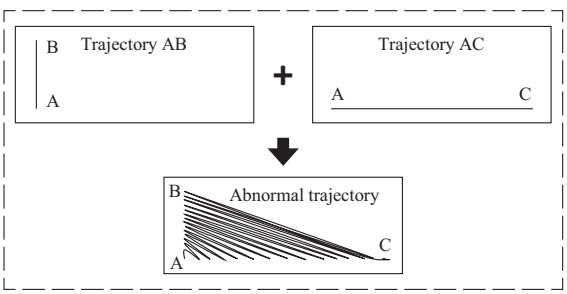


Fig. 9. Abnormal trajectory after file splicing.

Experiment I: file recovery with metadata

This experiment simulated a typical forensic scenario where the file system metadata was available. Since our GPS device supported removable SD card, we could use a write-protected card reader and forensic software to copy the memory dump of the SD card. Prior to deleting any NMEA logs, we copied the image of SD card (approximately 14.8 GB) as a backup, with known data (i.e. 359 NMEA logs which occupy 339 MB). It should be noted that this forensic image could be used to restore the SD card to the original state. We then performed a simple deletion of the NMEA logs and made a forensic image of SD card again. Then, we used both EnCase and GPSDroid to recover the deleted logs – see the recovered trajectory in Fig. 10.

$$P(Precision) = \frac{A}{A+B} * 100\% \tag{3}$$

$$R(Recall) = \frac{A}{A+C} * 100\% \tag{4}$$

$$F - Measure = \frac{2*P*R}{P+R} \tag{5}$$

In order to evaluate the quality of recovered results, three criteria were considered, namely: precision, recall and F-Measure. In both Eqs. (3) and (4), A represents the number of recovered data blocks belonging to the original logs, B is the number of recovered data blocks that do not belong to the original logs, and C is the number of data blocks belonging to the original logs but not the recovered logs. In addition, we used F-measure (Wikipedia and F-measure), weighted average of precision and recall, to evaluate the recovered results.

As presented in Table 1, the recovered results of both methods were comparable except for the time taken. As the recovery mechanism in EnCase relies on system metadata, it only needs to analyze the area of metadata in the image. While for our GPSDroid, it has to scan the entire image to pinpoint all data blocks associated with the NMEA logs. Thus, GPSDroid requires additional time to process a 14.8 GB image. In addition, we observed that two additional logs were recovered by using GPSDroid. Upon manual examination, we found that two errors occurred in the 'date' field of RMC sentence in the original logs. Both incorrect dates were not included in any of the logs recovered. Since in the logs reconstruction stage we generated new logs based on their dates, the two incorrect ones were stored in the additional logs. We speculated that the errors occurred when the software wrote the data to logs or during data transmission. However, it should be noted that these errors are noise track points and need to be excluded in the trajectories reconstruction stage.

Experiment II: file recovery without metadata

This experiment simulated the scenario where the file system metadata is missing. Similar to experiment I, we first copied a new image from the image (after deleting the logs). Then, we fragmented the new image into pieces of 512 bytes and shuffled them. In this scenario, the new image could be seen as a set of shuffled data blocks (512B). Since the shuffle operation destroyed the relations among original data blocks, the system metadata became unavailable. Besides, it should be noted that all NMEA logs were heavily fragmented due to the shuffle operation. Finally, both EnCase and GPSDroid were used to recover logs.

The findings were summarized in Table 2. Using GPSDroid, we were able to recover NMEA logs with/without metadata but this was not the case when we used EnCase. In addition, when

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

Fig. 10. One of the recovered trajectories using GPSDroid.

Table 1
Recovered results with file system metadata.

Method	Recovered/original logs count	A	B	C	Precision rate	Recall rate	F-Measure	Time taken (hh:mm:ss)
EnCase	359/359	695744	0	0	100%	100%	1	00:00:05
GPSDroid	361/359	695577	7	167	99.999%	99.976%	0.99987	00:08:18

Table 2
Recovered results without file system metadata.

Method	Recovered/original logs count	A	B	C	Precision rate	Recall rate	F-Measure	Time taken (hh:mm:ss)
EnCase	0/359	0	0	695744	0	0	0	00:00:05
GPSDroid	361/359	695542	49	202	99.993%	99.971%	0.99982	00:13:11

comparing the recovered results between experiments I and II, we demonstrated that GPSDroid was able to deal with fragmentation although time taken in experiment II was slightly longer. This was because in this experiment, the content of NMEA logs was heavily fragmented due to the shuffle operation. Since many logically consecutive NMEA data blocks lost their original neighbors, temporal information could not be extracted. And for those data blocks, they had to be stored in $List_2$. Due to the large number of data blocks stored in $List_2$, there is a corresponding increase in the size of possible merging from $List_1$ and $List_2$.

Experiment III: file recovery with partial data overwritten

In this experiment, we sought to evaluate the effectiveness of GPSDroid in dealing with data that have been partially overwritten, as one would find in a typical real world scenario. We first constructed several synthetic datasets to simulate the different possible scenarios. Specifically, after making a copy of the image from Experiment I, we scanned the data blocks (512B) in the image to determine whether it belonged to a NMEA log. If yes, then we generated a random number (from 0 to 1). And if this number is less than a pre-determined threshold (i.e. overwritten rate), then we would fill the data blocks with random data. Thus, by choosing each value from the sequences (i.e. 0.01, 0.02, 0.03 … 0.3) as the overwritten rate, we could construct the associated image respectively. Besides, similar to the setting in Experiment II, we fragmented the image. In this scenario, each image was not only heavily fragmented and partially overwritten, the system metadata was

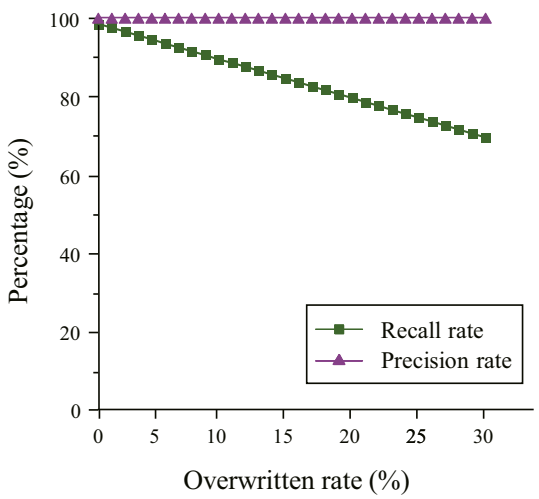


Fig. 11. Recovered results after partially overwritten.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

Table 3
Recovered results after partially overwritten.

Overwritten rate	Recall rate	Precision rate
1%	98.951%	99.973%
2%	97.964%	99.972%
3%	96.970%	99.971%
4%	95.934%	99.973%
5%	94.990%	99.974%
6%	93.874%	99.972%
7%	92.942%	99.971%
8%	91.937%	99.971%
9%	90.957%	99.970%
10%	89.829%	99.972%
11%	88.942%	99.974%
12%	87.961%	99.971%
13%	86.943%	99.973%
14%	85.917%	99.972%
15%	84.957%	99.973%
16%	83.964%	99.971%
17%	82.928%	99.972%
18%	81.962%	99.975%
19%	80.947%	99.971%
20%	79.964%	99.969%
21%	78.899%	99.972%
22%	77.914%	99.973%
23%	76.902%	99.975%
24%	75.925%	99.970%
25%	74.911%	99.969%
26%	73.934%	99.971%
27%	72.890%	99.969%
28%	71.876%	99.968%
29%	70.869%	99.969%
30%	69.865%	99.967%

also unavailable. Then, we utilized GPSDroid to recover logs from each constructed image. It is general knowledge that once a file had been deleted and overwritten, the system metadata would be reallocated for new files. Thus, metadata-based recovery approaches such as EnCase would not work.

As presented in Fig. 11 and Table 3, GPSDroid could pinpoint data blocks belonging to NMEA logs and not include other dirty data even though some data blocks were overwritten with random data. As the overwritten rate increases, the precision rate remains unchanged. However, we observed that the recall rate decreases with an increase in the overwritten rate. It was because the overwritten data blocks belonging to the original logs could not be recognized by GPSDroid, resulting in the increase of C (see also Equation (4)). It should also be noted that the sum of overwritten rate and its

Table 6
Recovered results after using system format.

System format	Recovered/original logs count	Time taken by format
Regular format	0/359	11 min10 s
Quick format	361/359	7 s

Table 7
Different programming languages' APIs for changing timestamps.

Programming language	API
C/C++	SetFileTime
JAVA	SetLastmodified
C#	SetLastWriteTime
Python	SetSystemTime

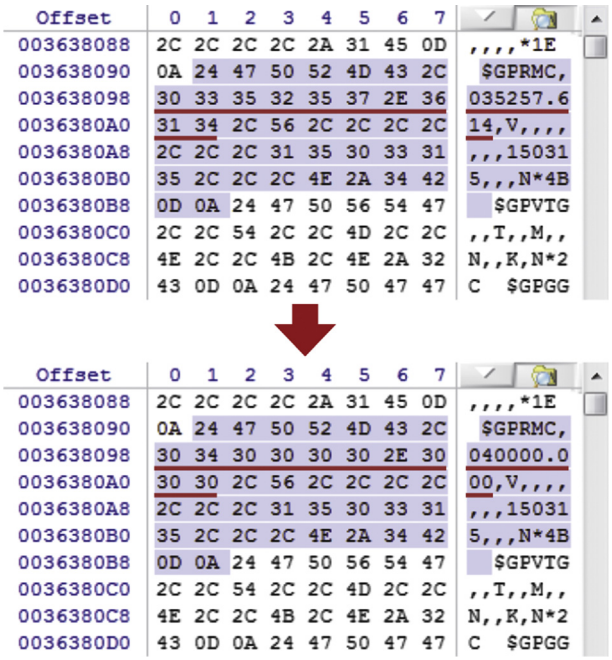


Fig. 12. Using WinHex to change file's content.

Table 4
Recovered results with various cluster sizes.

Cluster size	Recovered/original logs count	A	B	C	Precision rate	Recall rate	F-Measure	Time taken (hh:mm:ss)
512B	361/359	695542	49	202	99.993%	99.971%	0.99982	00:13:11
1KB	361/359	695542	49	202	99.993%	99.971%	0.99982	00:08:30
2KB	361/359	695535	56	209	99.992%	99.970%	0.99981	00:08:24
4KB	361/359	695549	49	195	99.993%	99.972%	0.99983	00:08:21
8KB	361/359	695542	42	202	99.994%	99.971%	0.99983	00:08:26
16KB	361/359	695549	63	195	99.991%	99.972%	0.99981	00:08:20
32KB	361/359	695528	56	216	99.992%	99.969%	0.9998	00:08:22
64KB	361/359	695549	49	195	99.993%	99.972%	0.99983	00:08:18

Table 5
Recovered results after normal/professional deletion.

Deletion method	Total size	Recovered/original logs count	Time taken by deletion
Normal deletion	111MB	119/119	1.5 s
BCWipe software	115MB	0/120	20.1 s
Eraser software	112MB	0/120	59.4 s

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

corresponding recall rate was close to 100%. In other words, GPSDroid was able to recover the remaining parts of NMEA logs and reduced the impact of overwritten data to a minimal.

Experiment IV: file recovery with various cluster sizes

This experiment was conducted to determine GPSDroid's capability to deal with different cluster sizes. While cluster sizes might vary between different file systems, they were generally in multiples of 512 bytes. Therefore, in order to simulate different cluster sizes, we constructed several datasets. After determining that the cluster size of our used SD card was 4KB, we made 8 copies of the image from experiment I. Then, we fragmented the image into pieces of 512B, 1KB, 2KB, 4KB, 8KB, 16KB, 32KB and 64KB prior to subjecting them to random shuffling. Thus, we obtained 8 different images and GPSDroid was utilized to recover logs for each case.

As reported in Table 4, we achieved a high precision rate and recall rate for the 8 images, and interestingly the time required to process cluster size of 512B was the longest. This was because clusters bigger than 512B were likely to contain a more complete record, in the sense that most valid data blocks had temporal information and they were stored in $List_1$. Thus, the time taken in merging the clusters from $List_1$ and $List_2$ was reduced.

Experiment V: file recovery after professional deletion

In this experiment, we attempted to determine the possibility of recovering NMEA logs after professional deletion. First, we used Win32DiskImager (a tool designed to write a raw disk image to a removable device or backup a removable device to a raw image file) to restore our SD card to the original state (prior to deleting the logs). Then, we divided all NMEA logs into three parts, namely: 119 logs with 111MB, 120 logs with 115MB and 120 logs with 112MB, prior to deleting these logs using normal deletion, BCWipe and Eraser. Finally, GPSDroid was utilized to recover the deleted logs.

As summarized in Table 5, we were only able to recover logs that had been deleted normally. We observed that both BCWipe and Eraser took a long time to delete log files, as these software would re-write the area of file's content many times to ensure that the file was removed permanently.

Experiment VI: file recovery after system format

The aim of this experiment is to determine whether NMEA logs could be recovered after using system format. Similar to Experiment V, we used Win32DiskImager to restore the SD card to the original state, before formatting and copying the corresponding forensic images. GPSDroid was then used to recover logs from these

Fig. 13. The abnormal trajectory recovered by GPSDroid.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004

two images, and the experimental results were summarized in Table 6. We were able to recover the logs after a quick format had been performed.

Experiment VII: tampering detection after non-intrusive modification

In this experiment, we simulated non-intrusive modification. After using Win32DiskImager to restore the SD card to its original state, we conducted the following activities on two separate images.

- We used WinHex to modify a log's content without affecting its last-written timestamps. Specifically, the SD card was first mounted in mass storage mode using a card reader. After this was done, WinHex could analyze the metadata of disk to obtain the log's physical address so that the content of NMEA logs could be accessed. Finally, we changed the time information of a RMC sentence.
- We modified the log's timestamp (see Table 7) to ensure consistency with its trajectory data after modifying the log's content.

As shown in Fig. 12, the temporal information was successfully modified (i.e. replaced 035257.614 with 040000.000). However, we were not able to detect traces suggesting the modifications (see also Table 7).

Experiment VIII: tampering detection after file splicing

This experiment was designed to simulate the scenario of file splicing. We first constructed a fabricated log using a copy of an actual NMEA log, which had two trajectories with the near departure but different destinations. Besides, the temporal information of two trajectories were different:

- 1st trajectory from 08:10:28 to 08:16:36
- 2nd trajectory from 17:01:01 to 17:09:29

Therefore, we wrote a python script to correct this difference so that the 2nd trajectory was from 08:10:00 to 08:18:28. Then, we deleted the other trajectory data in this log and changed the timestamps to maintain consistency with the 2nd trajectory. Finally, we replaced the original log with the fabricated log.

As Fig. 13 illustrates, the reconstructed trajectory was an irregular area rather than a curve. This is due to the lines being too dense. After zooming in, we can see physically impossible jumping for that car, which is an obvious tell-tale sign that the log had been modified.

Conclusion

NMEA logs are a rich source of evidence in a GPS forensic investigation, particularly as modern vehicles are increasingly equipped with built-in GPS navigation systems. Existing forensic approaches in the literature generally rely on the existence of file system metadata. Therefore, once the system metadata is unavailable, these approaches will not be effective.

In this paper, we presented a file carving based algorithm to facilitate forensic investigation of NMEA compliant GPS devices, including recovering of deleted NMEA logs from GPS devices without system metadata. We then demonstrated the utility of the proposed algorithm using a set of experiments.

Future research includes evaluating the proposed algorithm on a wider range of NMEA compliant navigation systems, with the aims of fine-tuning the algorithm if necessary.

Acknowledgment

This work is supported by the cyberspace security Major Program in National Key Research and Development Plan of China under grant No.2016YFB0800201, the Natural Science Foundation of China under grant No.61070212, 61572165 and 61702150, the State Key Program of Zhejiang Province Natural Science Foundation of China under grant No.LZ15F020003. The last author is also supported by a Cloud Technology Endowed Professorship.

The authors would like to thank the editor and the two anonymous reviewers for their constructive feedback.

References

Boddington, R., Hobbs, V., Mann, G., 2008. Validating Digital Evidence for Legal Argument. Security Research Centre Edith Cowan University.
Breeuwsma, I.M.F., 2006. Forensic imaging of embedded systems using jtag (boundary-scan). Digit. Investig. 3 (1), 32–42.
Breeuwsma, M., Jongh, M.D., Klaver, C., Knijff, R.V.D., Roeloffs, M., 2009. Forensic data recovery from flash memory. Small Scale Digit. Device Forensics J. 1.
Carrier, B., 2005. File System Forensic Analysis. Addison-Wesley Professional.
Cohen, M.I., 2007. Advanced carving techniques. Digit. Investig. 4 (3–4), 119–128.
Cusack, B., Simms, M., 2011. Evidential recovery from gps devices. J. Appl. Comput. Inf. Technol. 15 (1). ISSN 2230-4398. http://www.citrenz.ac.nz/jacit/jacit1501/2011cusack_evidentialrecovery.pdf.
de Sousa, J.P.C., Gondim, J.J.C., 2016. Extraction and analysis of volatile memory in android systems: an approach focused on trajectory reconstruction based on nmea 0183 standard. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 328–337.
DFRWS, 2007. Challenge. http://www.dfrws.org/2007/challenge.
D'Orazio, C., Ariffin, A., Choo, K.-K.R., 2014. iOS anti-forensics: how can we securely conceal, delete and insert data?. In: 47th Annual Hawaii International Conference on System Sciences (HICSS 2014), pp. 4838–4847.
Elstner, J., Roeloffs, M., 2016. Forensic analysis of newer TomTom devices. Digit. Investig. Int. J. Digit. Forensics Incid. Response 16 (C), 29–37.
Eterovicsoric, B., Choo, K.R., Mubarak, S., Ashman, H., 2017. Windows 7 anti-forensics: a review and a novel approach. J. Forensic Sci. 62, 1054–1070.
Garfinkel, S. Announcing frag find: finding file fragments in disk images using sector hashing. https://tech.groups.yahoo.com/group/linuxforensics/message/3063.
Garfinkel, S.L., 2007. Carving contiguous and fragmented files with fast object validation. Digit. Investig. Int. J. Digit. Forensics Incident Response 4, 2–12.
Garfinkel, S.L., Mccarrin, M., 2015. Hash-based carving: searching media for complete files and file fragments with sector hashing and hashdb. Digit. Investig. 14, S95–S105.
Iii, G.G.R., Roussev, V., August, 2005. Scalpel: a frugal, high performance file carver. In: Refereed Proceedings of the Digital Forensic Research Workshop, DFRWS 2005. Astor Crowne Plaza, New Orleans, Louisiana, USA.
Karresand, M., Shahmehri, N., 2006. Oscar – File Type Identification of Binary Data in Disk Clusters and RAM Pages. Springer US.
Karresand, M., Shahmehri, N., 2008. Reassembly of fragmented jpeg images containing restart markers. In: European Conference on Computer Network Defense, pp. 25–32.
Leom, M.D., Choo, K.-K.R., Hunt, R., 2016. Remote wiping and secure deletion on mobile devices: a review. J. Forensic Sci. 61 (6), 1473–1492.
Li, W.J., Wang, K., Stolfo, S.J., Herzog, B., 2005. Fileprints: identifying file types by n-gram analysis. In: Information Assurance Workshop, 2005. Iaw '05. Proceedings from the Sixth IEEE Smc, pp. 64–71.
Liao, Z.F., Li, Y., Peng, Y., Zhao, Y., Zhou, F.F., Liao, Z.N., Dudley, S., Ghavami, M., 2015. A semantic-enhanced trajectory visual analytics for digital forensic. J. Vis. 18 (2), 173–184.
Na, G.H., Shim, K.S., Moon, K.W., Kong, S.G., Kim, E.S., Lee, J., 2014. Frame-based recovery of corrupted video files using video codec specifications. IEEE Trans. Image Process. Publ. IEEE Signal Process. Soc. 23 (2), 517–526.
Pal, A., Memon, N., 2009. The evolution of file carving. IEEE Signal Process. Mag. 26 (2), 59–71.
Poisel, R., Tjoa, S., Tavolato, P., 2011. Advanced file carving approaches for multimedia files. In: IEEE International Conference on Computational Science and Engineering, pp. 1558–1565.
Quick, D., Martini, B., Choo, K.-K.R., 2013. Cloud Storage Forensics. Syngress, an Imprint of Elsevier.
Stamm, M.C., Liu, K.J.R., 2011. Anti-forensics of digital image compression. IEEE Trans. Inf. Forensics Secur. 6 (3), 1050–1065.
Van Eijk, O., Roeloffs, M., 2010. Forensic acquisition and analysis of the random access memory of TomTom GPS navigation systems. Digit. Investig. 6 (3–4), 179–188.
Wikipedia, Baidu map. https://en.wikipedia.org/wiki/Baidu_Maps.
Wikipedia, F-measure. https://en.wikipedia.org/wiki/F1_score.
Wikipedia, Nmea 0183. https://en.wikipedia.org/wiki/NMEA0183.
Zheng, Y., Zhang, L., Xie, X., Ma, W.Y., April, 2009. Mining interesting locations and travel sequences from gps trajectories. In: International Conference on World Wide Web. WWW 2009, Madrid, Spain, pp. 791–800.

Please cite this article in press as: Shi, et al., A novel file carving algorithm for National Marine Electronics Association (NMEA) logs in GPS forensics, Digital Investigation (2017), http://dx.doi.org/10.1016/j.diin.2017.08.004