AFAUC – anti-forensics of storage devices by alternative use of communication channels

Harald Baier, Julian Knauer
da/sec - Biometrics and Internet Security Research Group
Hochschule Darmstadt, Darmstadt, Germany
Email: {harald.baier,julian.knauer}@cased.de

Abstract

Since the end of the 1990ies side channel attacks became a very prominent branch of cryptography. In other areas of computer security, however, side channels are not well studied. It is the primary goal of this paper to raise the awareness of the community about the potential existence of side channels during a forensic investigation. We present a concept called AFAUC – anti-forensics of data storage by alternative use of communication channels. The general idea is to confuse the investigator by abusing a communication channel for unintended purposes. As a concrete example of AFAUC, we access a storage device through its diagnostic interface to obfuscate data on that device. More precisely, we analyse if it is feasible in practice to abuse an existing communication channel, which was designed for a different purpose, to hide data in an area of a hard disc drive (HDD), which is not accessible by an investigator and which is different from the well-known Host Protected Area and Device Configuration Overlay, respectively. The basic idea is to access the HDD via its diagnostic interface in an unintended manner and to manipulate its size in the firmware setting. We show that this is possible even without any expensive tool for a Samsung HDD. Evaluation including a test in a law enforcement laboratory revealed that the hidden data would not be detected in an ordinary case. Hence AFAUC may be used by skilled, but not well-funded users. Although AFAUC is a classical dual-use technology, we would like to initiate the community to come up with further alternative use cases of communication channels to support users in oppressive countries to defend themselves. In contrast to the underground economy these users are typically not well-funded and thus depend on reliable anti-forensic methods.

Keywords

Digital forensics techniques and tools, anti-forensics, HDD security, obfuscating data, reverse engineering, ATA.

I. Introduction

Anti-forensic techniques tend to make use of obfuscation techniques to prevent investigators to access data. Typical techniques are the use of encryption, steganography, or programme packers [1], [2], [3], [4]. In the first case the investigator detects a data structure, but he is unable to use it as evidence as he only has access to the meaningless ciphertext. Steganography, on the other side, aims at hiding the existence of data by embedding it in unsuspicious files. In both cases it is a non-trivial task to find the actual data to support hypothesis within an investigation.

A further well-known approach in computer forensics to deceive investigators is to hide data somewhere on the disc. Common approaches in the past were the slack space of a FAT or an NTFS file system (e.g., the formerly software slacker from the Metasploit framework [5]), journals of a journaling file system (e.g., ext3 [6]) or in areas, which are reserved for inodes [7]. However, data in these locations is easily accessible and may hence be detected by file carving methods like `foremost` or `scalpel`.

Anti-forensic methods are dual-use, that is they may be used by both the white hats (e.g., users in an oppressive state) or the black hats (e.g., the underground economy). While the first use case is important to support people to store and distribute independent information, the latter use case is an objectionable secondary effect. As a consequence within the first use case investigators should not be able to detect the anti-forensic protected data, while they should be aware of the anti-forensic methods in the latter one.

We were inspired by side channels and aim at promoting a similar idea, which we call *alternative use of communication channels* as potential anti-forensic techniques. Alternative use means to communicate via an interface, which is primarily aimed at a different purpose, e.g., for maintenance. With respect to white hats, the alternative use may be a technique to provide methods to hide data for benign purposes. On the other hand in case of the black hats investigators should be aware of the potential existence of abusing communication channels to obfuscate data. Hence our primary goal is to raise awareness in the forensic community about this weaker form of side channels to hide data.

As a sample use case we extend the idea to use hidden partitions of hard disc drives (HDD), which are already known to the community (e.g., [2], [8], [6]). The ATA standard [9] provides the possibility to generate partitions on the HDD, which are only accessible after issuing certain ATA commands to the HDD. The ATA standard describes two such areas, the Host Protected Area (HPA) and the Device Configuration Overlay (DCO). Without sending specific ATA commands to the HDD it refuses access to these areas. Although the knowledge about the existence of an HPA or a DCO is not widespread under computer scientists, these areas are well-known in the computer forensics community and thus part of any investigation.

A. *Contributions*

The ATA commands to set hidden partitions inspired our work presented in that paper. Our main goal is to describe a general methodology to access a storage medium via a communication channel, which was designed for a different purpose. While the idea of abusing an existing interface for unintended purposes is a basic idea in IT security, we aim at raising the awareness that this may be used as a dual-use anti-forensic method, too. The technical idea is to establish a communication channel to its firmware by reverse engineering the firmware commands. Based on our work [10], [11] as a proof of concept we show that it is feasible without expensive toolkits to create a hidden data partition on an HDD, which is typically not revealed by investigators. We call our approach AFAUC – anti-forensics of data storage by alternative use of communication channels. Our practical use case obfuscates data on a Hard Disc Drive.

The basic idea is to overwrite the maximum addressable user sector, which is stored within the firmware configuration. By manipulating this firmware data field the drive capacity is reduced. The data within this newly created partition is not accessible until the original settings are restored. Thus data can be hidden in this location. Our approach is to gain access to the firmware configuration by exploiting the functionality of the diagnostic interface in an unintended manner, which is available on some hard disc drives. We connect to the interface through a self-made RS-232 transceiver and manipulate the configuration to create a new data partition. The actual monetary costs of this process are negligible, however, it requires some technical skills and time.

As proof of concept we make use of a Samsung SP2504C P120S, where we reduce the accessible capacity from 250 GiB to 249.8 GiB. Unlike a Host Protected Area or the use of the Device Configuration Overlay, this manipulation is neither detected by common ATA-tools like `hdparm` nor by widespread commercial tools like `PC-3000` [12] or `TD1` [13] used by forensic researchers and data rescue companies. For instance, this could be verified with the help of the IT forensic department of a German law enforcement agency and the use of their professional PC-3000 suite. The possibility to modify the device configuration without any warnings or signals from the device shows the lack of

ATA command	**Description**
`IDENTIFY DEVICE`	Host is capable of requesting all necessary information about the target, e.g., serial number, firmware revision, security status, total number of user addressable sectors.
`READ NATIVE MAX ADDRESS`	This command returns the native device size with factory default settings. It is mandatory for devices implementing the Host Protected Area (HPA) feature.
`SET MAX ADDRESS`	This command changes the size of the maximum addressable user data area. The new setting is returned by `IDENTIFY DEVICE`.
`DEVICE CONFIGURATION IDENTIFY`	This command returns the actual features of a disc including its physical size, i.e., the real maximum Logical Block Address (LBA) of the device.

Table I
OVERVIEW OF RELEVANT ATA COMMANDS

important sanity checks in the firmware data structures. More details about AFAUC may be found in [10].

A further aim of this paper is to raise the awareness of forensic researchers and investigators to have a careful look at HDD device settings while analysing hard disc drives and look for suspicious device settings.

B. Notation and Terms used in this Paper

Within this paper we refer to terms related to the AT Attachment (ATA) standard. The first term is the *Host Protected Area* (HPA), it is used to enable a data partition, hidden and inaccessible for the operating system. The second term is *Device Configuration Overlay* (DCO), it can also be used to create a hidden data partition. We also need to define the names *host* and *target*. While the host is always the computer from which all actions are originated, the target is the device connected to the host. In the manner of this paper a target represents an HDD. Throughout this paper we use a mono spaced typeface for ATA command sequences defined by the ATA standard, e.g., `IDENTIFY DEVICE`.

C. Organisation of this Paper

The rest of the paper is organised as follows: In the subsequent Sec. II we present the foundations of the ATA standard and its commands, which are necessary to understand our approach. We additionally sketch current techniques to obfuscate data areas on HDDs. Then, Sec. III introduces our general methodology used as paradigm of AFAUC. In Sec. IV we describe details of our side channel access to the firmware configuration via the diagnostic interface. Next Sec. V shows our reconfiguration of the HDD setting, which is evaluated with support from the IT forensic department of a German law enforcement agency in Sec. VI. Finally Sec. VII concludes our paper and points to future work.

II. FOUNDATIONS OF THE ATA STANDARD AND HIDDEN AREAS

In this section we introduce the foundations of the ATA standard [9], which are necessary to explain our anti-forensic approach AFAUC in the subsequent sections. As our approach is inspired by the two well-known hidden data partitions HPA and DCO [14], [8], we introduce these areas, too.

The AT Attachment (ATA) [15] aims at standardising an interface to connect storage media devices like a Hard Disc Drive or optical discs to its host. The ATA standard enumerates a bunch of commands to read or set parameters of the device. ATA is maintained by the Technical Committee 13 (TC13) of the InterNational Committee on Information Technology Standards (INCITS). All relevant ATA commands of our approach are listed in Table I.

```
LBA                                         488000001            488397168
  0                                             |     488245121          |
  |                                             |         |              |
  ------------------------------------------------------------------------
  |          User area                          |  HPA    |     DCO      |
  ------------------------------------------------------------------------
                                                |         |              |
                              IDENTIFY DEVICE __|         |              |
                                        READ NATIVE MAX __|              |
                                        DEVICE CONFIGURATION IDENTIFY  __|
```

Figure 1. Data area layout with HPA and DCO enabled.

```
$ hdparm -N /dev/sda
     /dev/sda:
        max sectors = 488000001/488245121, HPA is enabled
```

Figure 2. Detection of a Host Protected Area

We explain the usage of the basic ATA commands by means of two common ways to create an additional hidden data partition: The *Host Protected Area* (HPA) and the *Device Configuration Overlay* (DCO). A sample disc layout is given in Fig. 1.

An HPA is the first and oldest method to create a hidden data area. It was introduced with ATA standard version 4 in 2001 [16] and is designed to provide system vendors a separate data partition to store recovery utilities or backups of the operating system [14].

An HPA is detected by most of the common forensic software tools available, e.g., `hdparm`. To check if an HPA is present, the `IDENTIFY DEVICE` command will return a smaller target capacity than the result of `READ NATIVE MAX ADDRESS`. The difference determines the size of the HPA. On the other side an HPA is easily created by sending a `SET MAX ADDRESS` to the target with the new drive capacity as parameter. The command `SET MAX ADDRESS` can also be used to disable the HPA by setting the new size equal to `READ NATIVE MAX ADDRESS`.

As an example the listing in Fig. 2 invokes `hdparm` using the switch `-N` to compare the output of `IDENTIFY DEVICE` (488000001 sectors in our example) and `READ NATIVE MAX ADDRESS` (488245121 sectors). Thus an HPA actually is present.

The DCO is part of ATA since revision 6 in 2002 [17]. It is the second method to conceal parts of a disc provided by the ATA standard. It provides the possibility to manipulate certain HDD options including the physical device capacity. Vendors use the DCO e.g., to the change the behaviour and the appearance of an HDD. As an example, the vendor could change drive specific identification settings and the capacity of the device to sell one drive as many different models with different features. When the drive capacity has been reduced, the DCO area becomes inaccessible. In difference to the HPA, which can be detected by comparing the output of `IDENTIFY DEVICE` to `READ NATIVE MAX ADDRESS` a DCO is discovered by checking the size of `READ NATIVE MAX ADDRESS` against the output of the command `DEVICE CONFIGURATION IDENTIFY`. The listing shown in Fig. 3 makes use of `hdparm` to issue the `DEVICE CONFIGURATION IDENTIFY` command.

An HPA and a DCO can coexist, but the DCO has to be created before the HPA [14]. This is required due to the dependency of HPA and the output of `READ NATIVE MAX ADDRESS`. However, hiding

```
  $ hdparm --dco-identify /dev/sda
      /dev/sda:
      DCO Revision: 0x0002
      The following features can be selectively disabled via DCO:
          Transfer modes:
               udma0 udma1 udma2 udma3 udma4 udma5 udma6
          Real max sectors: 488397168
          ATA command/feature sets:
      [REMOVED]
```

Figure 3. Detection of a Device Configuration Overlay

information in these areas will not be successful from an attacker's point of view, as it is common practice to acquire data of these partitions, too. For instance, this is stated in the German IT forensic guideline from the German Federal Office for Information Security [18], p.26: *Completeness of the image: Reserved areas of mass storage media (e.g., HPA and DCO) must be detected reliably and deactivated during acquisition to get a complete image.*

III. Our methodology and related work

This section sketches our methodology to obfuscate data on an HDD. After that we review current work, which is related to this paper.

Side channels as used in cryptography inspired us to think about a similar use in computer forensics. In cryptography a side channel is typically used by a passive attacker, who observes information delivered by a device, which performs cryptographic operations. The attacker's goal is to gather information about a private cryptographic key. Sample side channels are power consumption (e.g., Differential Power Analysis by Kocher et al. [19]), timing observation (e.g., [20]), or recently acoustic emanations [21].

However, this sort of channels is used by passive attackers to gain information. A transfer to an anti-forensic approach means that we have to shift the user's role to become an active one. Once keeping in mind that a key paradigm of information security is to avoid unintended use of interfaces, we came up with our methodology of AFAUC. It comprises the following steps:

1) The first step is to identify a communication channel to access a storage device. In case of a classical HDD we have the classical primary interface for bulk data exchange, e.g., (S-)ATA, SCSI. However, we are more interested in some sort of 'side channel', that is we look for an interface, which is primarily aimed at a different purpose than bulk data exchange, e.g., for maintenance. This step is in good tradition of IT security to abuse an interface for unintended use (e.g., think of network-based attacks to an atomic power plant through a legacy modem connection, although the IT network is not accessible through LAN).
   In a previous version of this paper, we used the term *side channel* for this access interface. However, most of our communication partners recommended to make use of a different term. We therefore decided to make use of the notation *alternative use of a communication channel*. Nevertheless we are convinced that there should be a better expression than these two alternatives.
2) Once an interface is identified we have to connect to the storage device through this interface. For instance, we have to identify the role of pins, the voltage, the protocols, or data units.

3) If a connection is established a reverse engineering of interface commands is used to actually change the configuration of the device. Classical sources are manuals, the manufacturer's web site, Internet forums, etc. We have to keep in mind that we may not be successful, e.g., due to checksums over the altered data, backup copies differing from our manipulated configuration, or simply because the configuration data is encrypted (and because the configuration data is not targeted through this interface, it is not decrypted by our access).
4) We must adapt non-digital information to our manipulation. For instance, if we change the storage capacity of an HDD, we have to replace the original boilerplate by a manipulated one, too.
5) The manipulation shall not be detected by an investigator. We therefore propose to go through a sample data acquisition using wide-spread tools and check, that the manipulation is not detected.

In the following sections we show that this methodology works to implement AFAUC: a diagnostic interface of an HDD may be used as a communication channel to access an HDD and reverse engineering of the firmware commands is used to create an obfuscated data partition.

We finish this section by pointing to related work. General anti-forensic related work (e.g., [1], [2], [3], [4], [5]) was already mentioned in the introduction. Above we discussed some related work with respect to side channels (e.g., [19], [20], [21]). It remains to mention storage medium specific related literature. Actually only few papers deal with this topic. A more sophisticated approach compared to our one is from [22], who show how to install a back-door on an HDD such that the HDD controller replaces data while writing on the disc. There are two main differences to our approach: first, the key scenario is an attacker, who aims at remote control of the HDD. Second, [22] requires more skills than our approach AFAUC and thus limits the user group. At 44CON 2013 Travis Goodspeed presented a disc, which functions normally for a legitimate user. However, the disc erases itself in response to any attempt to image the disc with common tools like `dd` [23]. In contrast to our approach the investigator is aware of the anti-forensic measures.

IV. Accessing an HDD via diagnostic interface

In this section we give some insights in our access channel to the firmware of an HDD. Our idea is to follow the common attacker paradigm of using a *communication channel* in an unintended way to get unauthorised access to devices. More precisely we make use of the *diagnostic interface*, which is implemented by some HDD vendors (e.g., Samsung, Seagate) to implement a serial interface for diagnostic purposes. The access process is similar to that described in [10], [11].

The HDD operates a *Micro Controller Unit* (MCU) and a dedicated firmware [24]. MCUs may be accessed from the host via a serial interface. We call these components a *diagnostic interface*. The existence of diagnostic interfaces is unknown to the vast majority of computer users. Its functionality varies a lot between different HDD models and even within the same series of devices. Its common purpose is to provide diagnostic tools to analyse or to measure the health of an HDD.

By establishing a connection to the interface, the user gains access to a diagnostic application running within the HDD firmware. On the host side we have to run a terminal software to communicate with the diagnostic application. For Linux a common terminal is `minicom` [25], for Windows one may use `hyperterminal` [26].

As the diagnostic interface is not documented by the vendors, we need to perform reverse engineering to access the HDD in this way. To establish a connection it is at first necessary to locate the port on the target. The port itself is an RS-232 interface, which requires an RS-232 transceiver [27] to convert the logic signals between host and target. An RS-232 connection requires at least three connections: receive (RX), transmit (TX), and a common ground (GND). The RS-232 module we use for our research also requires an additional power supply (V$_{CC}$) for the integrated circuit.

Locating the access port on the target is in general not easy. In our proof of concept, we make use of a Samsung SP2504C P120S HDD, where the port is located at the configuration pins on the back of the HDD. Seagate HDDs use a similar topology. S-ATA devices, on the other hand, make use of a dedicated port. Once the access port is found, we have to solve the difficulty of determining the correct settings for the terminal application. It is important to set the correct transfer parameters within the terminal application, otherwise it is not possible to communicate with the target as the output and input characters cannot be interpreted correctly and the information on the screen will look garbled. Instead of following a black box reverse engineering approach we try to find the relevant information with the correct pin-out and terminal settings in discussion boards on the Internet. Actually a forum dealing with hard drive technology and recovery [28] yields hints, which we could validate successfully for our hardware. Sample pin-out configurations we retrieved are shown in Fig. 4.

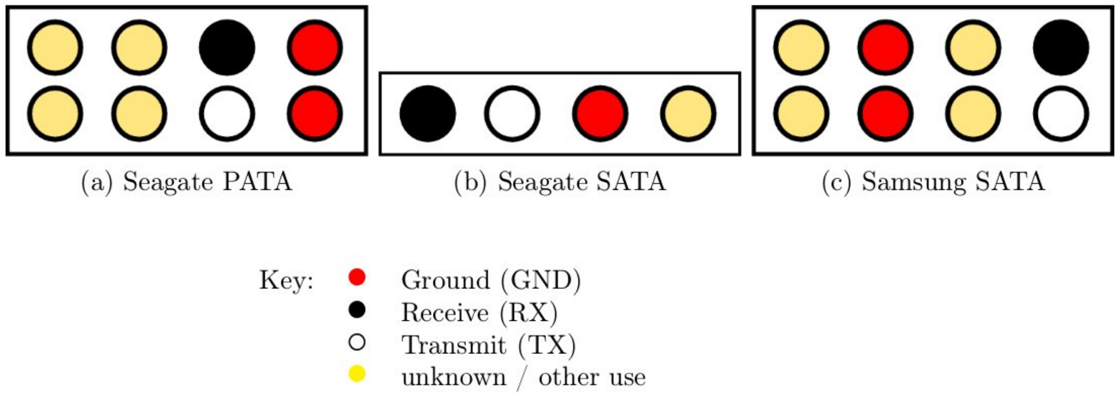

Figure 4. HDD pin-out configuration for 3 different HDD models.

If all settings are correct, and a connection is established, the diagnostic interface will present a command prompt to the user. However, as the command interface is not documented, again a reverse engineering approach is required to issue commands to the device and to gather information about the device's properties. We will describe in Sec. V our proceeding to reconfigure the device settings of the Samsung HDD.

V. Reconfiguring Device Settings

The goal of this section is to describe a method to create an additional data partition, which is inaccessible and undetectable by common forensic data acquisition methods. This is achieved by manipulating the device capacity without using any ATA feature. Section VI evaluates our approach. As before our sample HDD is a Samsung HDD SP2504C P120S HDD, but the reverse engineering should be successful for any device with an established communication via the diagnostic interface. The reason to select the Samsung HDD is rather simple: it was left over in our research group.

We establish a side channel to the target by using the method of Sec. IV and the terminal settings `57600 BAUD, 8-N-1` (see e.g., [29]). If the target device is now powered on, the target will immediately print the firmware boot messages on the terminal. After the initialisation process is complete, the firmware will wait for user input with the prompt `ENG>`.

However, a web-based reverse engineering approach was only partly successful as only few information about the Samsung's firmware commands are available. Inspired by [30] our hypothesis was that commands comprise one or two capital letters and up to two arguments. We could successfully

Command	Parameter	Description
`RM`	ModuleIndex	Read Module: reads the module with ModuleIndex from the service area into the device memory.
`MW`	Offset DataWord [DataWord...]	Memory Write: modify the contents of the memory by writing one or multiple DataWord beginning at Offset.
`WM`	ModuleIndex	Write Module: writes the memory buffer back to the service area as module ModuleIndex.

Table II
IMPORTANT SAMSUNG DIAGNOSTIC COMMANDS

validate this assumption by trial-and-error. An important mnemonic to communicate with the Samsung HDD is the help instruction `HE`, which yields an overview over available commands. Additionally `HE 1` provides a more fine grained help information, e.g., if a command accepts or requires arguments.

For our approach AFAUC the relevant commands to read and write data to and from the target are listed in Table II. [30] provides some details about the usage of these commands, the meaning of the mnemonics, however, is due to our interpretation of the firmware behaviour after issuing the respective instruction. Hence we are able to have full access to the HDD to generate a hidden partition without using ATA commands.

The firmware configuration of the Samsung HDD is split into different parts, which are called *modules*. A module is referenced by its index, which is a non-negative integer. In order to load the first module of index 0 into the device memory we have to issue the command `RM 0`. Additionally the content of the module is written to the terminal as data words of 2 *bytes*, that is the smallest addressable unit is a `WORD` of 16 bits. Each loaded module links it start address to `W:005B00`. After loading a module into the device memory we can modify its content by issuing the command `MW`. This command requires two arguments: first, the offset within the module, where the data will be written to, and second the actual data. Finally, the modified module is flushed to the service area using the command `WM`.

For ease of investigation we implement a parser to create a binary dump of each module, which is written to a file on the host system. In the first step, we dump the first 20 modules. An interesting property is that each module has a length of $512 \cdot k$ bytes (k is a positive integer) and thus a multiple of the HDD block size of 512 bytes.

To examine the purpose of each module, we inspect them with a hexdump viewer. A first observation is that each module stores an ASCII string of length 8 bytes in the beginning. For instance, module 0 starts with the string `FSI_____` (where `'_'` means a SPACE, i.e., ASCII character $0x20$), module 6 with the string `CONFIG__`, and module 16 with the string `SECURITY`. Obviously the Samsung HDD stores the name of the module describing its purpose within this data field. As our aim is to reconfigure the size of the HDD, module 6 attracts our interest as its first bytes contain the ASCII string `CONFIG`. The output of the command `RM 6` on our terminal is listed in Fig. 5.

Modern devices use the Logic Block Address (LBA) scheme to access an HDD block. As shown in Fig. 3 we know from the output of the command `hdparm --dco-identify` that the maximum LBA of the Samsung HDD is $488397168 = 0x1d1c5970$. As the Samsung HDD accesses its data in units of a `WORD` of 16 bits, we translate this value to a little-endian hexdump string with respect to this unit, which results in the string `5970 1d1c`. This value can be found in the `CONFIG` module at byte offset `0x26`. However, in terms of the Samsung addressing we have to convert this offset relative to a `WORD` unit (i.e., $\frac{0x26}{0x2} = 0x13$) and relative to the start offset `W:005B00`. Hence to reconfigure the disc

```
ENG> RM 6
W:005B00 434F 4E46 4947 2020 0000 0000 0000 0004
W:005B08 0000 0001 0002 0003 0000 0001 0002 0003
W:005B10 3FFF 0010 003F 5970 1D1C 0000 0000 0000
W:005B18 2459 005B 65B4 01D2 7242 0349 7D93 04BE
[...]
```

Figure 5. Hex dump of module 6 with the starting ASCII string CONFIG

```
LBA                                       488000001      488297168
  0                                           |      488245121 |488397168
  |                                           |         |      |         |
  -----------------------------------------------------------------------
  |          User area                        |  HPA    | DCO  |  MOD    |
  -----------------------------------------------------------------------
                                              |         |      |
                             IDENTIFY DEVICE __|         |      |
                                       READ NATIVE MAX __|      |
                              DEVICE CONFIGURATION IDENTIFY  __|
```

Figure 6. Data area layout with HPA, DCO, and additional hidden data partition MOD

size we have to manipulate the two WORDs starting at offset W:005B13.

Our aim is to reconfigure the device to reduce its apparent physical capacity as returned by `hdparm --dco-identify`. Our modified HDD partitioning is shown in Fig. 6. We reduce the device capacity by $100,000$ HDD blocks, i.e., by about 50 MiB. Thus we create an obfuscated partition of roughly 50 MiB at the end of the disc.

At first we read the CONFIG module into the device memory by entering `RM 6`. This will display the full content of the module on the terminal as shown in Fig. 5. To verify that the correct module is loaded we validate to find the string `434F 4E46 4947` at offset W:005B00. Additionally we expect to find the original drive capacity encoding `5970 1D1C` at offset W:005B13. Next we need to translate the manipulated drive capacity $488297168 = 0x1d1ad2d0$ to the little-endian WORD encoded string `d2d0 1d1a`. Then we overwrite the original capacity by issuing the command `MW 5B13 D2D01D1A`. To make the changes permanent a final `WM 6` writes the modified memory content back to the service area with module index 6.

Upon the next device start, the changes will become active. All ATA commands, which depend on the drive capacity, will use this manipulated setting as reference.

VI. Evaluation

In this section we evaluate our approach AFAUC with respect to the following properties: first, we prove the *basic functional correctness* of our approach. Second, we discuss the user's strength in terms of *costs*. Our aim is to show that even a low-funded, but technically skilled user is able to implement AFAUC. Finally, we discuss the *practical relevance* of our anti-forensic attack and reveal that the common process model of investigators (e.g., law enforcement) does not reveal the hidden data partition.

A. Functional correctness and costs

We first show the basic functional correctness of our implementation by applying two tests using common data acquisition tools. The functional test is performed on an ordinary laptop running Linux and using only open source tools. The first one disables HPA and DCO of the original Samsung HDD and writes a file of length 512 bytes into the final HDD block, i.e., to LBA 488397167. This file contains the ASCII control pattern *Do you detect our AFAUC anti-forensic approach?*. To verify that our method works as expected, we create a full disc image after our manipulation (see Fig. 6) using `dd` and search for our test pattern in the image file. Fortunately it is not found. Our second functional test simply invokes the command `hdparm --dco-identify` on the manipulated device, which returns the size `Real max sectors: 488297168`. Hence our AFAUC proof of concept actually works.

Next we turn to the costs to implement AFAUC. If we neglect the personnel costs we only have to consider the components to build the RS-232 transceiver. The investment is less than 10 EUR and thus almost free of charge. However, putting AFAUC into practice requires technical skills, dedicated knowledge about HDDs and time to customise our implementation to different HDD models. Hence AFAUC is cheap in terms of monetary costs, but expensive in terms of time to adapt it. However, a motivated attacker is able to make use of AFAUC to obfuscate data.

B. Practical relevance

Finally, we turn to the practical relevance. We already showed in Sec. VI-A that the hidden data partition is not detected by the widespread opensource Linux tools `dd` and `hdparm`, respectively, where the latter tool directly invokes ATA commands. Thus an investigator using these tools does not detect our modified HDD configuration. We are now interested in the risk of detecting the obfuscated data partition if AFAUC is applied by an investigator, who additionally has access to powerful commercial tools. The setting is that we only care about the digital information, that is we do not take further information like boilerplates into account (it is easy to forge them, too). We tested the tools `PC-3000` [12] and `TD1` [13].

We first handed our Samsung HDD to specialists of the IT forensic department of a German law enforcement agency, who makes use of the professional PC-3000 suite [12] to investigate HDDs. PC-3000 is a combination of software and hardware components and accesses the device via its ATA interface. It seems that PC-3000 makes use of some proprietary ATA commands provided by the HDD manufacturers for maintenance and service access [31]. According to [12] and the law enforcement investigators the suite is even successful in case of damaged storage media. PC-3000 comes with a detailed documentation about different HDD models to support the investigator in acquiring and inspecting a typical HDD and to recover or repair a storage device. The law enforcement staff inspect the digital (!) configuration information of our Samsung HDD according to their common process model and does not reveal the hidden data partition of about 50 MiB. Thus the risk of detection is low. However, PC-3000 is able to access backup copies of the different firmware modules within the service area. Actually there are 3 backup copies of the original HDD setup. But PC-3000 does not perform a sanity check and the current process model of the law enforcement agency does not look at these backup copies, too.

We compare the output of PC-3000 information to the information of our AFAUC implementation and do not find any deviation. Hence we are convinced to provide a correct working anti-forensic approach. The direct monetary costs of a PC-3000 based manipulation are about 3000 EUR and thus rather low for a professional suite. On the other hand, the utilisation of PC-3000 requires experience with this tool and thus time.

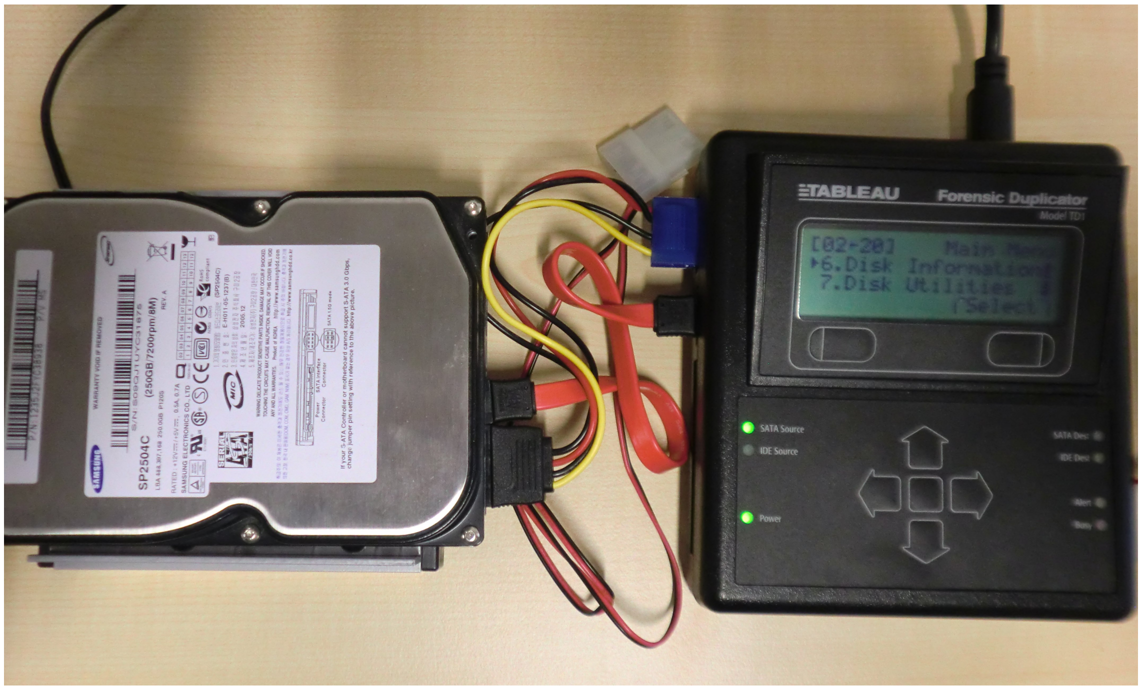

Figure 7. Evaluation setting of the TD1 imaging tool.

The results of our second commercial tool TD1 approve the results of PC-3000. This test was performed by investigators of a large German IT forensic department. A photo of the setting for the evaluation is shown in Fig. 7.

VII. Conclusion and Future Work

In this paper we provide a general methodology to access a storage medium via a communication channel in an unintended way and to establish a communication channel to its firmware by reverse engineering the firmware commands. Our approach AFAUC (anti-forensics of storage devices by alternative use of communication channels) is successful as it is feasible to implement it without expensive toolkits and the risk of detection is low.

Additionally we want to raise the awareness of forensic researchers and investigators to keep in mind that anti-forensic is a growing issue.

As a future work we aim at extending AFAUC to manipulate any backup copy in the service area, too, in order to minimise the risk of detection. Furthermore, a reverse engineering of the apparantly undocmented ATA commands, which are supposedly used by the PC-3000 suite, should yield a comparison to our approach.

Acknowledgment

We thank the IT forensic departments of both a German law enforcement agency and a private company in supporting us to evaluate our approach AFAUC. We are grateful for the anonyomous reviewer comments to improve the paper. This work was supported by the German Federal State of Hesse through its LOEWE funding of CASED.

References

[1] S. Garfinkel, “Anti-forensics: Techniques, detection and countermeasures,” in *The 2nd International Conference on i-Warfare and Security (ICIW)*, Monterey, CA, USA, March 2007.

[2] ——, “Forensics wiki,” http://www.forensicswiki.org/wiki/Anti-forensic_techniques, February 2013, last retrieved: Feb, 20th 2014.

[3] W. Henrique, “Anti forensics: Making computer forensics hard,” in *Code Breakers III*, September 2006.

[4] R. Harris, “Arriving at an anti-forensics consensus: Examining how to define and control the anti-forensics problem,” in *DFRWS 2006*, August 2006.

[5] Metasploit, “Mafia – metasploit anti-forensics project,” http://www.metasploit.net/research/projects/antiforensics/, December 2008, no further support available.

[6] K. Eckstein and M. Jahnke, “Data hiding in journaling file systems,” in *DFRWS 2005*, August 2005.

[7] Grugq, “To the Art of Defiling,” in *Black Hat Asia 2003*, August 2003.

[8] B. Carrier, *File System Forensic Analysis*. Addison Wesley, 2010.

[9] T13, “AT Attachment 8 - ATA/ATAPI Command Set (ATA8-ACS),” http://t13.org/Documents/UploadedDocuments/docs2006/D1699r3f-ATA8-ACS.pdf, December 2008.

[10] J. Knauer, “Hard Drive Security,” Master’s thesis, CASED – Hochschule Darmstadt, http://www.dasec.h-da.de, 2012.

[11] J. Knauer and H. Baier, “Zur Sicherheit von ATA-Festplattenpasswörtern,” in *Proceedings of D-A-CH Security 2012*, Konstanz, DE, September 2012, pp. 26–37.

[12] “ACE Laboratory, PC-3000 for Windows UDMA,” http://www.acelaboratory.com/pc3000.udma.php, April 2012, last retrieved: february, 17th 2014.

[13] “TABLEAU, TD1 Forensic Duplicator,” http://www.tableau.com/, May 2013, last retrieved: February, 17th 2014.

[14] M. R. Gupta, M. D. Hoeschele, and M. K. Rogers, “Hidden Disk Areas: HPA and DCO,” *International Journal of Digital Evidence*, Fall 2006.

[15] T13, “AT Attachment 3 Interface (ATA-3),” http://www.t13.org/documents/UploadedDocuments/project/d2008r7b-ATA-3.pdf, January 1997.

[16] ——, “AT Attachment 4 with Packet Interface 6 (ATA-4),” http://www.t13.org/documents/UploadedDocuments/project/, 2001.

[17] ——, “AT Attachment 6 with Packet Interface 6 (ATA-6),” http://www.t13.org/documents/UploadedDocuments/project/, 2002.

[18] G. F. O. for Information Security, “Leitfaden ’IT-Forensik’,” http://www.bsi.bund.de/, March 2011, last retrieved: February, 20th 2014.

[19] P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” in *Advances in Cryptology – CRYPTO’99*. Springer-Verlag, 1999, pp. 388–397.

[20] P. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in *Advances in Cryptology – CRYPTO’96*. Springer-Verlag, 1996, pp. 104–113.

[21] D. Genkin, A. Shamir, and E. Tromer, “RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis,” http://www.cs.tau.ac.il/~tromer/acoustic/, December 2013, last retrieved: February, 17th 2014.

[22] J. Zaddach, A. Kurmus, D. Balzarotti, E.-O. Blass, A. Francillon, T. Goodspeed, M. Gupta, and I. Koltsidas, “Implementation and implications of a stealth hard-drive backdoor,” in *29th Annual Computer Security Applications Conference (ACSAC)*. ACM, 2013, pp. 279–288.

[23] T. Goodspeed, “A Fast Hard Disk with Active Antiforensics,” http://44con.com/speakers/2013/travis-goodspeed.html/, September 2013, last retrieved: February, 17th 2014.

[24] H. Messmer, *The Indispensable PC Hardware Book (3rd Edition)*. Addison-Wesley, 1999.

[25] M. van Smoorenburg, J. Lahtinen, A. C. de Melo, and J. Seymour, “minicom: Friendly serial communication program,” http://www.t2-project.org/packages/minicom.html, 2011.

[26] H. Inc., “Hyperterminal,” http://http://www.hilgraeve.com/hyperterminal, 2013.

[27] MAXIM, *True RS-232 Transceivers*, rev. 5 ed., MAXIM, Maxim Integrated Products, Inc. 120 San Gabriel Drive Sunnyvale, CA 94086 USA.

[28] “Unlocking ATA Password for Western Digital,” http://forum.hddguru.com/unlocking-ata-password-for-western-digital-t8374-20.html, 2006, last retrieved: Februar, 15th 2014.

[29] hddstudio, “Seagate diagnostic command list,” http://forum.javaxtreme.com/threads/103-Seagate-Diagnostic-Command-List, April 2010, last retrieved: February, 14th 2014.

[30] Andrej, “Palo does not come with the preparation of safe (translated),” http://www.hardw.net/forum/topic6825.htm, 2006.

[31] A. Vidström, “Computer Forensics and the ATA Interface,” FOI, Swedish Defence Research Agency Command and Control Systems Box 1165 SE-581 11 LINKÖPING Sweden, Technical report E7091, February 2006.