

SIM and USIM Filesystem: a Forensics Perspective

Antonio Savoldi and Paolo Gubian
University of Brescia
Department of Electronics for Automation
Via Branze 38, I25121 Brescia, Italy
{antonio.savoldi,gubian}@ing.unibs.it

ABSTRACT

The main purpose of this paper is to describe the real filesystem of SIM and USIM cards, enlightening what the official standard reference does not say. By analyzing the full filesystem of such embedded devices, it is possible to find a lot of undocumented files usable to conceal sensitive and arbitrary information that are unrecoverable with the standard tools normally used in a forensic field. In order to understand how it is possible to use a SIM/USIM for data hiding purposes, the paper will present a tool capable of extracting the entire observable memory of these devices together with the effective filesystem structure. Further, some practical examples regarding the data hiding procedure as a proof of concept will be analyzed and discussed.

Categories and Subject Descriptors

C.3 [Special-Purpose and Application-Based System]: Smartcards; E.5 [Files]: [backup/recovery, sorting/searching]

General Terms

Filesystem analysis

Keywords

filesystem, SIM/USIM card, imaging tool, data hiding

1. INTRODUCTION

There are many commercial tools used in order to extract and decode the raw data contained by SIM/USIM cards, although none of them is capable of revealing the real filesystem's structure and, consequently, to discover the multitude of nonstandard files which are hidden into such devices. This is what can be done with SIMBrush¹ [6][7], a new open source tool, developed in ANSI C for Linux

¹The tool can be required by contacting the author. More information are available at <http://www.ing.unibs.it/~antonio.savoldi>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07 March 11-15, 2007, Seoul, Korea

Copyright 2007 ACM 1-59593-480-4/07/0003 ...\$5.00.

and Windows platforms, aimed at extracting the observable portion of the filesystem of a SIM/USIM card. The real news for the digital forensics research is to know what is really concealed and, potentially concealable, in a standard SIM/USIM card's filesystem, thus demonstrating that data hiding is possible in such devices.

It is possible to analyze which types of data are extractable, from an ordinary SIM/USIM card, with commercial and proprietary tools, highlighting the data having an evidential value. *SIM Card Seizure* [5], which is part of the Paraben's Cell Seizure suite, is able to extract a proper subset, about 29 files, of the standard reference elementary files illustrated in ETSI 100-977 [14]. Other similar tools, for example *DataPilot* [8] or *Cards4Labs* [9], are again proprietary or their use, as in the latter case, is restricted to law enforcement personnel. In the open source arena there are, in the authors' knowledge, only two examples. The first, TULP2G [10][15], is a framework, developed by NIS (the Netherlands Institute of Forensics), implemented in C# and whose utility is in the recovery of data from electronic equipment such as cellular and SIM cards. The second example, BitPim [3], is a program that allows to manipulate, at the logical level, many CDMA phones branded LG, Samsung, Sanyo and other manufacturers. More information about mobile forensics tools can be found in [12].

From a forensics point of view, the SIMBrush tool can be placed in the imaging technologies group of techniques. Its mission is to extract from SIM and USIM cards, defined as the physical item, the information stored in them, to the widest possible extent, and to produce as output what is called a primary image. Subsequently, this primary image can be used during the investigation instead of the physical item itself, thus preserving the digital integrity of such a fundamental object. Throughout the rest of the paper we will briefly describe what is notable for this tool and, how it is possible to reconstruct the entire SIM's and USIM's filesystem. After that, some users' scenario will be shown with regard the presentation of notable data present in such devices. Finally, after having understood the real structure of the SIM/USIM filesystem, it will be shown how it is possible to use nonstandard locations to conceal arbitrary data, giving a practical demonstration of the effectiveness of the method.

2. TECHNOLOGY BACKGROUND

SIM stands for *Subscriber Identity Module* and together with the *Mobile Equipment*, that is the user cellular phone, constitute the *Mobile Station*, which defines, in the GSM

(*Global System for Mobile Communications*) system, the so called “end user part”. The evolution of such a pervasive system, which counts more than one billion of users in the world, is the UMTS (*Universal Mobile Telecommunications System*), which increases, with respect to the GSM counterpart, the bandwidth for data exchange. In this case we must consider, in the end user part of the network, the USIM, that is *Universal Subscriber Identity Module*, together with the *Mobile Equipment*. Substantially, there are not big differences between SIM and USIM, from the filesystem structure point of view, although the USIM contains more data, defined by the ETSI standard reference.

Every SIM/USIM card is a smart card, standardized by ISO: in particular, SIMs are contact (as opposed to contactless) smart cards, which are specified by ISO standard 7816 [11]. They contain a microprocessor, three types of memory, which are RAM, ROM and EEPROM and, finally, some integrated logic to manage the security issues. The SIM can be considered as a black box interfacing with *Mobile Equipment* by a standard API (Application Program Interface). The filesystem is stored in an internal EEPROM and has a hierarchical structure with a root called *Master File* (MF). Basically, there are two types of files: directories, called *Dedicated Files* (DF) and files, called *Elementary Files* (EF). The main difference between these two types of files is that a DF contains only a *header*, whereas an EF contains a header and a *body*. The header contains all the meta-information that relates the file to the structure of the filesystem (available space under DF, number of EFs and DFs which are direct children, length of a record, etc.) and security information (access conditions to a file), whereas the body contains information related to the application for which the card has been issued. In an ordinary SIM/USIM card three types of EF are possible:

- **transparent EF:** these files are organized as a sequence of bytes. It is possible to read all or only a subset of their contents by specifying a numeric interval.
- **liner fixed EF:** the atomic unit for these files is the record, instead of the byte. A record is a group of bytes that has a known coding: every record of the same file represents the same kind of information. In a linear-fixed EF, all the records have the same length.
- **cyclic EF:** these files implement a circular buffer where the atomic unit of manipulation is the record. Therefore, the concepts of first and last are substituted by those of previous and next.

As said previously, there are a lot of files in an ordinary SIM/USIM card which can be subdivided as follows:

- **Information about the subscriber:** the SIM stores the International Mobile Subscriber Identity (IMSI) in specific file called EF_{IMSI} , which is a unique identifier for each subscriber in the system, as specified in [13]. Information about preferred languages, stored in EF_{LFP} and EF_{ELP} , could be of help in determining the subscriber’s nationality. Mobile Station ISDN (MSISDN), stored in EF_{MSISDN} , could be used to retrieve the calls originated by the user towards other phone numbers.

- **Information about acquaintances:** subscribers can maintain a list of the numbers they call or they are called from more frequently or that are of importance to them, precisely in EF_{ADN} , EF_{FDN} and EF_{BDN} . Furthermore, subscribers could be registered to one or more groups of subscribers if the so called “multicalls” are enabled.
- **Information about SMS traffic:** it is possible to read SMS messages (EF_{SMS}) sent and received by the subscriber out of the SIM card, and to see for every received SMS whether it has been read or not.
- **Information about subscriber:** the SIM stores, in the elementary file called EF_{LOCI} , the last area where the subscriber has been registered by the system.
- **Information about calls:** the last numbers dialed are stored in a file in the SIM filesystem, which is EF_{LND} . The key used to encrypt the last call is stored there too, precisely in EF_{Kc} .
- **Information about the provider:** it is possible to extract the provider name (EF_{SPN}), and the mobile network used for communications ($EF_{PLMN_{sel}}$), along with mobile networks that are forbidden to the subscriber (EF_{FPLMN}).
- **Information about charge:** some SIM cards maintain information about residual credit. In particular, the elementary file EF_{ACM} indicates the total time of the calls executed from the last charge; the cost for unit time is defined in EF_{PUCT} .
- **Information about the system:** every SIM card has a unique ID stored in it, stored in EF_{ICCID} . All the services to which the subscriber could be enabled, along with the actual status of habilitation, are stored in the SIM, precisely in EF_{SST} .

The operations allowed on the filesystem are coded into a set of commands [14] that the *interface device* (IFD), which is the physical device capable of interfacing with a SIM and setting up the communication session, issues to the SIM, then waiting for the responses and, among these, only few are really important in the SIMBrush tool:

- **SELECT:** this command selects a file for use and makes the header of that file available to the IFD;
- **GET RESPONSE:** in SIM cards, if some data is to be communicated from the card to the IFD after a command, it is the IFD itself that has to request it, using this command. What is really important to note is that there is no command to delete or create files and no command to quickly browse the entire filesystem.
- **READ BINARY:** with this command it is possible to read the body of a transparent EF.
- **READ RECORD:** this command permits to read the record, considered as an atomic unit, in elementary files such as linear fixed and cyclic.
- **VERIFY, CHANGE, ENABLE, DISABLE, UNBLOCK CHV:** these commands permit the management of *Card Holder Verification 1* (CHV1) and *Card*

Table 1: Access conditions for SIM/USIM cards.

Level	Access conditions
0	ALWays
1	CHV1
2	CHV2
3	Reserved for GSM future use
4 to 14	ADM
15	NEVer

Holder Verification 2 (CHV2) authentication codes, known also as *Personal Identification Number 1 (PIN1)* and *Personal Identification Number 2 (PIN2)*.

Thus, in SIMBrush’s core algorithm, only these commands are used, preserving, in this way all integrity data in the filesystem, all data being extracted in read only access mode. It is interesting to note the *access conditions* which are indicated in the files header: these acts as constraints to the execution of commands which protect files from unauthorized manipulation, and only for the duration of their authorization. In particular these constraints, specified by some bytes in the header of each elementary file, are related to one set of specific commands issuable to a card, namely *update*, *read*, *increase*, *rehabilitate* and *invalidate*.

Table 1, related to the access conditions, specifies which level of authorization is required to issue the mentioned commands to a SIM/USIM card. This is important to understand and to translate the part of the header, in each elementary file, related with this access conditions.

The aforementioned access conditions can be explained as follows:

- **ALW:** the command is always executable on the file;
- **CHV1:** the command is executable on the file only if one among *Card Holder Verification 1* code or *Unblock Card Holder Verification 1* (called also PUK1, that is the Personal Unblocking Code 1) code has been successfully provided;
- **CHV2:** same as CHV1, but using *Card Holder Verification 2* code or *Unblock Card Holder Verification 2* (called also PUK2, that is the Personal Unblocking Code 2);
- **ADM:** allocation of these levels is a responsibility of the administrative authority which has issued the card: the card provider or the telephone provider which gives the card to its subscribers.
- **NEV:** the command is never executable on the file;

As will be clear in the latter section of this paper, these access conditions, for nonstandard files, play a key role to conceal arbitrary data in such files. It is also mentionable that only having the PIN1 is possible to extract all the interesting content from the SIM/USIM card. Indeed, it has been used only standard methods in order to extract digital data. Alternatively, it is possible unblock the card with the proper PUK1 code.

2.1 The Filesystem Extraction

Standard programs, like that developed by Shenoi[4], can extract only the standard elementary files starting from the selection rules defined in the reference standard [14]. First, the standards say that the filename is univocal and, for example, “3F00” identifies the root of the filesystem, which is the master file of a SIM/USIM card. Second, the SELECT command may be executed with any file specified with the relative ID with no restrictions. This leads to the opportunity to “brush” the logical ID space by issuing a SELECT command for each possible ID, from “0000” to “FFFF”, obtaining a warning from the SIM when the ID does not exist in the filesystem, or the header of the file when it does. In a SIM’s filesystem there is the concept of *current file* and *current directory*.

The SIM’s files are hierarchically selectable with the following constraints, specified in the reference standard [14]:

- MF is always selectable independently of the current directory.
- the current directory is always selectable.
- the parent of the current directory is selectable.
- any DF which is an immediate child of the parent of the current directory is selectable.
- any file, EF or DF, which is an immediate child of the current directory is selectable.

With these rules in mind, it is possible to select, for example, the file “6F3C” (SMS) by issuing two SELECT commands: the first to select the DF 7F10, which is the father of this EF, and the second to select the file, that is, in this case, the SMS elementary file. As is notable from the standard reference, the SIM filesystem has an n-ary structure and it is easy to extract the standard part of every SIM/USIM, just with only a few commands, by reading directly all the elementary files declared in the standard reference. This is, in the authors’ knowledge, the approach used in all commercial and open-source tools. With the objective to acquire all the observable memory of a SIM, that is the data accessible with standard methods, we must define the general selection rule by which it is possible to “brush” the entire logical address space of EEPROM. Thus, aware of the selection rules, defined by reference standards, it is possible to associate a set of files and directories to each of the above mentioned groups:

- **MF_SET:** it has only a single element: the Master File;
- **CURRENT_SET:** it has only a single element: the current directory;
- **PARENT_SET:** it has only a single element; the parent of the current directory;
- **DF_SIBLINGS_SET:** contains all children of the parent of the current selected element;
- **SONS_SET:** contains all files which are children of the current directory;

With these sets it is possible to define, given the current directory, which are the files and directories that are selectable, and this is explained by the following relationship:

$$\begin{aligned} \text{SELECTABLE_SET} = & \text{MF_SET} \cup & (1) \\ & \text{CURRENT_SET} \cup \\ & \text{PARENT_SET} \cup \\ & \text{DF_SIBLINGS_SET} \cup \\ & \text{SONS_SET} \end{aligned}$$

Because the relationship between the set of every possible current directory and the set of possible SELECTABLE_SET is univocal, it is thus possible reconstruct the entire filesystem, finding the missing part of it, which is, at each level of the n-ary tree, the set of sons:

$$\begin{aligned} \text{SONS_SET} = & \text{SELECTABLE_SET} \setminus & (2) \\ & (\text{MF_SET} \cup \\ & \text{CURRENT_SET} \cup \\ & \text{PARENT_SET} \cup \\ & \text{DF_SIBLINGS_SET}) \end{aligned}$$

With this relationship it is possible to reconstruct the entire filesystem tree; in particular only the header of any file (DF or EF) is ever extractable, while the body, which is the most important part of one elementary file, is subject to access condition limitations.

Presently, SIMBrush is able to extract the body of those files whose access conditions are ALW and CHV1/CHV2, and the latter case is possible only if the appropriate code is provided, that is when PIN1 or PIN2 are provided. The main algorithm is based on the construction of a binary tree, which is a suitable data structure for SIM card data, being this structure equivalent to an n-ary tree. We can explain the main elements of this pseudo-code:

```

Procedure Build_Tree
  Expand_DF(PARENT_SET = 0,
            CURRENT_SET = {MF},
            DF_SIBLINGS_SET = 0);
End
Procedure Expand_DF(PARENT_SET: NODE,
                   CURRENT_SET: NODE,
                   DF_SIBLINGS_SET: NODE)
  Select(CURRENT_SET);
  SELECTABLE_SET = Brush(CURRENT_SET);
  SONS SET = SELECTABLE_SET \
    (MF_SET U
     CURRENT_SET U
     PARENT_SET U
     DF_SIBLINGS_SET);
  For each node N belonging to SONS_SET,
    Place_in_tree(N);
    If N equal DF Then
      Expand_DF(PARENT_SET = CURRENT_SET,
                CURRENT_SET = N,
                DF_SIBLINGS_SET =
                DF_SIBLINGS_SET \ {N});
End

```

- **Build_Tree:** this procedure initializes the parameters of the recursive function **Expand_DF**.
- **Expand_DF:** is the recursive function that, starting from the filesystem root, brushes the ID space, searching all existing EFs and DFs and finding all sons of the current node, which are placed, dynamically, in a binary tree data structure. For each son, if this is an EF

then it is placed in the data structure; otherwise, if it is a DF then the Expand_DF function acts recursively, updating all interested sets.

- **NODE:** defines the main data structure to store all filesystem's data.
- **Select:** sends a SELECT command to the SIM card.
- **Brush:** this function selects a Dedicated File, passed as argument, which becomes the current DF, and brushes the entire logical ID's space, obtaining the SELECTABLE set as a result related to such DF.

Despite the simplicity of this solution in order to extract the entire part of filesystem of SIM/USIM cards, it is necessary explain clearly why such non standard files are valuable from a forensics perspective.

3. THE HIDDEN PART OF FILESYSTEM

The SIMBrush tool produces as output one XML (eXtensible Markup Language) file, which contains the raw data; the next step is to decode these data in a comprehensible form suitable for the forensics practitioners to derive useful data that could become digital evidence after the analysis. This part is done by a second tool, written in Perl, whose function is to translate clearly the rough data into a suitable form, which is the output XML file. This file replicates the tree structure of the input file with the complete translation of standard files following the guidelines of reference standards [14] and translating only the header for nonstandard files. This XML file, resulting from the translation process, can be viewed, similar to *Autopsy Forensics Browser* [1], in any type of web browser to maximize the simplicity of reading. XML-DOM (XML-Document Object Model) [2] has been used for the interpretation script in order to manage in an easy way and as an n-ary tree the XML file produced by SIMBrush.

With the *Document Object Model* paradigm every XML file is treated as an n-ary tree; in this way it is possible to manage every standard file, applying the correct translation procedure, so that the output results is an expanded and translated version of the input file. It is possible to see an example of translation about the contents of one record of SMS (6F3C) file: all the translated information is clear and simple to understand, being visible in textual format in any type of web browser. We can look at the raw data related to the contents of one SMS record:

```

<content>
01 07 91 93 33 85 28 02 00 04 04 85
94 61 00 00 50 70 40 90 60 25 80 A0
D4 64 13 14 B6 DB D3 F3 37 E8 2C 0F
D3 EB 69 FA 1B 44 0C 83 E2 F5 F2 9C
FE 06 B5 DF ED B2 9B FE 06 35 C3 78
7C 1A 74 0D 42 41 F4 34 48 5E 3E 87
D9 61 90 0C 34 AF BF DD 65 79 BA 0C
22 87 41 F3 71 58 9E 1E 87 E5 65 50
D9 4D 97 BF 41 69 36 68 16 7B C1 70
2F 58 0D 44 0E 83 A6 F5 B7 BB 2C 4F
97 41 CD 30 1E 9F BE 06 A1 20 54 DA
0D BA 06 A1 20 72 1A 44 4D 36 41 2F
A8 FC DD 7E EB D3 6F 77 3A 05 4A BA
CD 6F 50 98 0D 8A C5 72 FF FF FF FF
FF FF FF FF FF FF FF FF
</content>

```

With the help of reference standards [14] it is easy to extract all the information about the SMS message, such as

the time (date and hour) when the message has been sent, the length of SMS, the number of the sender, the number of the service center, and finally, the textual message.

```
<content>
<Date>04 Jul 05</Date>
<Hour>09.06.52</Hour>
<Length_SMS>160</Length_SMS>
<Number>4916</Number>
<Number_Service_Center>
393358822000
</Number_Service_Center>
<Status>01</Status>
<Text>TIM avviso gratuito Da questo momento
Maxi WAP ti regala 2 suonerie da scaricare
entroil 31/08/05 da SuonerieMaxiWAP
(in WAP di TIM/Promozioni)
</Text>
</content>
```

SIMBrush and the interpretation tool have been tested on several SIM and USIM cards of different sizes, providers and ranging between 16 Kbytes and 128 Kbytes. It is possible to extract every kind of data, among those defined in the reference standard [14]. Analogously to Windows filesystem, it is possible to derive the entire *File Allocation Table* of the SIM/USIM card, that is the complete and essential part of a filesystem regarding all the meta-information of the files. To demonstrate this, a partial list of standard and non-standard files related to a 128 Kbytes SIM card, has been reported, as seen in table 2.

The choice to represent the filesystem in this tabular form was dictated by the huge quantity of information related to each node of the n-ary tree of the filesystem. Substantially, each row of the table is the interpretation of the header for all filesystem dedicated and elementary files, standard and nonstandard. There are seven fields which refer to *ID*, *standard name*, *file type* (DF, EF or MF), *privileges*, which are related to the constraints on the execution of commands such as *Update*, *Read*, *Increase*, *Rehabilitate* and *Invalidate*, *structure of file* (transparent, linear fixed or cyclic), the field related to *father* of nodes, important to see the real structure of the n-ary tree, and, finally the *size* of the elementary files. This meta-information can also be seen in the XML file of output, as it is visible by analyzing the following snippets of code. The ICCID (Integrated Circuit Card ID) file is clearly visible, either in the raw and in the translated version.

```
<ef>
<header>0000000A2FE204000FF55501020000</header>
<body>
<content>98931000006092643586</content>
</body>
</ef>
```

By inspecting the translated header, it is clear that it is impossible to alter the body of the file, being the *acUPDATE* field fixed on the *NEV* value, which means that it is impossible to modify this elementary file, whose size, as it is readable, is of 10 bytes.

```
<EF>
<ICCID description="EFICCID '2FE2' (ICC
Identification): This EF provides a
unique identification number for the SIM.">
<content>98931000006092643586</content>
<header>
<ID>2FE2</ID>
<SIZE>10</SIZE>
<acINCREASE>NEW<acINCREASE>
```

```
<acINVALIDATE>ADM</acINVALIDATE>
<acREAD>ALW</acREAD>
<acREHABILITATE>ADM</acREHABILITATE>
<acUPDATE>NEV</acUPDATE>
<status>
File invalidated#
File not readable or updatable when
invalidated#
</status>
<structure>transparent</structure>
</header>
</ICCID>
</EF>
```

Analyzing table2, it can be seen that nonstandard files labelled with NS are the most predominant part and the goal is to demonstrate that some of this files are usable for data hiding purposes. The dedicated file “7F21” contains the same elementary file of DF 7F20, for backward compatibility through mobile equipment belonging to Phase 1 DCS 1800 [14] and there are no apparent inconsistencies related to the presence of files with the same ID because the standard explains such a fact. There are also other files placed in nonstandard directories, such as the “7FBB” DF, but unfortunately, even looking at non official reference standards, there is no official interpretation about this DF. It is arguable that some of these replicated standard files are placed in nonstandard locations for backward compatibility.

By analyzing, for example, the nonstandard elementary files under the DF “5FFF”, namely EFs ranging from “1F0C” to “1F3F”, it is visible that these files are modifiable with the *Update* command, being the privilege for this command *CHV1*. This means that everyone having the PIN1 of the card is authorized to store arbitrarily data by replacing the content of those existing.

This fact raises a new question about the concrete possibility to hide information in nonstandard locations of SIM cards and it is clearly an open issue, analyzable also from the steganography point of view. Now it is possible to describe how the data hiding can be applicable on SIM/USIM cards.

Initially, we can start with the full extraction of meta-data of all observable files present in the SIM. This can be done with the SIMBrush tool, in this way, defining a complete map of files in the SIM similar to the traditional *File Allocation Table*.

Subsequently, we can verify all nonstandard files which are modifiable when the access conditions for the *Update* command is equal to *ALW* or *CHV1* (the files are thus modifiable with the users’ privileges). The second stage of the experiment consists in the selection of those nonstandard files which have the relevant access privileges, as said previously, and in the writing of arbitrary data, whose length is determined by the size of the files.

This can be done with the *Update* command as specified in the standard reference. We have discovered that for a standard GSM SIM card of 128 KBytes, of TIM provider, there is a non-standard space of 16549 bytes, while the total engaged space by all files (standard and nonstandard) is 111959 bytes, as is verifiable summing up the size of files regarding the analyzed SIM card visible in table 2. The non-standard space can be defined as *Global Writable Slack Space (GWSS)*: it defines the nonstandard area of the SIM card that is really writable and thus modifiable arbitrarily. It has been possible to make this measure with the information of files size obtainable from the header of each elementary file.

This fact proves that it is possible to modify the integrity of data in the SIM card which can be considered as a covert channel, being capable of containing a lot of arbitrary data, arbitrarily definable, which is not accessible in a standard way.

In order to demonstrate how this is possible we present two basic procedures written in Perl that can be used to store and retrieve hidden data. In the first procedure, the non-standard elementary file "1F08" is chosen to modify its content. Having the *File Allocation Table* of the SIM it is quite trivial to select an arbitrary non-standard file and, subsequently, see the relative content.

```
sub Read {
  my @path = ('7f10','5fff','1f08');
  ($erg,$txt) = &SelectGSMFile(@path);
  print "ERG: $erg -- $txt\n\n";
  return $erg . " while getting " . $i
    if $erg;
  $txt=&DoAPDU("a0 b0 00 00 00");
  if ($txt =~ /^67([0-9a-f]{2})$/i)
  { $txt=&DoAPDU("a0 b0 00 00 " . $1);
    print "TXT: $txt\n"; }
  return "404 can't read " . $descr .
    ":" . $txt . "\n" unless $txt =~ s/9000$/;
}
```

After having read the content of the non-standard file, it is possible to write an arbitrary string in hexadecimal code that overwrites the previous one. In this example the body of the file is written with the user-defined string containing 20 null characters.

```
sub Write {
  my @path = ('7f10','5fff','1f08');
  ($erg,$txt) = &SelectGSMFile(@path);
  $cmd='00000000000000000000000000000000000000000000';
  $cmd = &DoAPDU(sprintf("A0 d6 00 00 14 %s",$cmd));
  return "402 write record $i: $cmd"
    unless $cmd =~ /9000$/;
}
```

After the modification, it is possible to extract again the entire image seeing the results of data hiding. Although this is a trivial example, by demonstrating the modification of one non-standard elementary file, we have verified that all such non-standard accessible files are modifiable. The following snippet of code represents the non-standard elementary file 1F 08, who size is 20 bytes, as it is visible from table 2.

```
<ef>
<header>000000141F0040015F555010200009000</header>
<body>
<content>22F2108000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</content>
</body>
</ef>
```

After having written the "null-string", as explained before, we can again extract the entire image and as a result obtain what is shown subsequently. In this case the body of the file is properly fixed to the user-defined value.

```
<ef>
<header>000000141F08040015F555010200009000</header>
<body>
<content>00000000000000000000000000000000000000000000</content>
</body>
</ef>
```

In our experiments we have considered, as already said, a lot of SIM cards of different size, ranging between 16 Kbytes and 128 Kbytes. Interestingly, only the 128 Kbytes SIM cards seem to have the highest content of non-standard files. The *GWSS* is about the 10% of the total EEPROM memory of the SIM.

4. CONCLUSIONS

In this paper we have presented a methodology that demonstrates how it is possible to conceal arbitrary data in the non-standard part of the SIM/USIM filesystem. This proof-of-concept is usable to say that ordinary tools used in the field of cellular forensics have a missing part. The recovery of the entire filesystem is only the initial part of the examination. The non-standard part of the filesystem can be analyzed with the steganalysis methods, trying to guess the coding used to store the arbitrary data. As future work we consider to improve the detection of hidden data in SIM/USIM cards, presenting a methodology to approach this problem.

5. REFERENCES

- [1] Autopsy Forensics Browser. Software available at: <http://www.sleuthkit.org/autopsy/>.
- [2] Document Object Model. Paper available at: <http://www.w3.org/DOM/>.
- [3] R. Binns. *BitPim*. Software available at: <http://bitpim.sourceforge.net/>.
- [4] G. Manes C. Swenson and S. Shenoi. Imaging and analysis of gsm sim cards. In *IFIP International Federation for Information Processing*, Springer Boston, pages 205–216, 2006.
- [5] Paraben Corporation. *Sim Card Seizure*. Software available at: <http://www.paraben-forensics.com/catalog/>.
- [6] A. Savoldi F. Casadei and P. Gubian. Simbrush: An open source tool for gsm and umts forensics analysis. In *Proceedings of Systematic Approaches to Digital Forensic Engineering, First International Work-shop*, Proc. IEEE, pages 105–119, 2005.
- [7] A. Savoldi F. Casadei and P. Gubian. Forensics and sim cards: An overview. *International Journal of Digital Evidence*, 5, 2006.
- [8] Susteen Inc. *DataPilot*. Software available at: <http://www.susteen.com/>.
- [9] Netherland Forensics Institut. *Card4Labs*. Software available at: <http://www.forensischinstituut.nl/NFI/nl>.
- [10] Netherland Forensics Institut. *Tulp2G, Forensic Framework for Extracting and Decoding Data*. Software available at: <http://tulp2g.sourceforge.net/>.
- [11] ISO. *Identification Cards - Integrated Circuit Cards with Contacts*. Paper available at: http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816.aspx.
- [12] *e-evidence info* The Electronic Evidence Information Center. *BitPim*. Software available at: <http://www.e-evidence.info/index.html>.
- [13] ETSI TS 300 927 v5.4.1. *Numbering, Addressing and Identification*. European Standard (Telecommunications series), 2000.
- [14] ETSI TS 100 977 v8.3.0. *Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface*. Paper available at: <http://www.id2.cz/normy/gsm1111v830.pdf>.
- [15] J. van den Bos and R. van der Knijff. Tulp2g an open source forensic software framework for acquiring and decoding data stored in electronic devices. *International Journal of Digital Evidence*, 4, 2005.

Table 2: A partial list of standard and nonstandard files extracted from a TIM 128 Kbytes SIM card.

ID	Name	File Type	Privileges	Structure	Father	Size [byte]
3F00	MF	MF	—	—	—	—
2F00	NS	EF	ALW ² ,ALW,ADM,NEV,NEV,NEV	linear fixed	3F00	46
2F05	ELP	EF	ALW,CHV1,NEV,NEV,NEV	transparent	3F00	4
2F06	NS	EF	ALW,NEV,NEV,NEV,NEV	linear fixed	3F00	330
2FE2	ICCID	EF	ALW,NEV,NEV,NEV,NEV	transparent	3F00	10
2FE4	NS	EF	ALW,NEV,NEV,NEV,NEV	transparent	3F00	35
2FE5	NS	EF	ALW,NEV,NEV,NEV,NEV	transparent	3F00	6
2FFE	NS	EF	CHV1,ADM,NEV,NEV,NEV	transparent	3F00	8
7F10	DFTELECOM	DF	—	—	3F00	—
5F3A	NS	DF	—	—	7F10	—
4F21	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	500
4F22	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5F3A	4
4F23	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5F3A	2
4F24	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5F3A	2
4F25	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	500
4F26	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	1250
4F30	SAI	EF	CHV1,ADM,NEV,ADM,ADM	linear fixed	5F3A	128
4F3A	NS	EF	CHV1,CHV1,NEV,CHV2,CHV2	linear fixed	5F3A	7000
4F3D	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	75
4F4A	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	39
4F4B	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	70
4F4C	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	70
4F50	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	1280
4F61	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	340
4F69	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5F3A	500
5FFF	NS	DF	—	—	7F10	—
1F00	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	105
1F01	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	175
1F02	NS	EF	CHV1,CHV1,NEV,NEV,NEV	transparent	5FFF	11
1F03	NS	EF	ALW,ADM,NEV,NEV,NEV	linear fixed	5FFF	40
1F04	NS	EF	ALW,CHV1,NEV,NEV,NEV	transparent	5FFF	4
1F05	NS	EF	ADM,ADM,NEV,ADM,ADM	linear fixed	5FFF	640
1F06	NS	EF	ADM,ADM,NEV,ADM,ADM	linear fixed	5FFF	420
1F07	NS	EF	CHV1,ADM,NEV,ADM,ADM	transparent	5FFF	20
1F08	NS	EF	CHV1,CHV1,NEV,NEV,NEV	transparent	5FFF	175
1F09	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5FFF	100
1F0A	NS	EF	ADM,ADM,NEV,ADM,ADM	linear fixed	5FFF	16
1F0B	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	16
1F0C	NS	EF	CHV1,CHV1,NEV,CHV1,CHV1	linear fixed	5FFF	34
1F1E	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	70
1F1F	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	70
1F20	NS	EF	CHV1,ADM,NEV,ADM,ADM	linear fixed	5FFF	128
1F21	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	1280
1F22	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	340
1F23	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	500
1F24	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	1250
1F34	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	500
1F38	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	5FFF	500
1F3D	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5FFF	4
1F3E	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5FFF	2
1F3F	NS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	5FFF	2
1F40	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	700
1F41	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	100
1F42	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	13
1F43	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	11000
1F44	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	5000
1F45	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	800
1F52	NS	EF	ADM,ADM,NEV,ADM,ADM	transparent	5FFF	50
6F06	NS	EF	ALW,NEV,NEV,NEV,NEV	linear fixed	7F10	770
6F3A	ADN	EF	CHV1,CHV1,NEV,CHV2,CHV2	linear fixed	7F10	7000
6F3B	FDN	EF	CHV1,CHV2,NEV,ADM,ADM	linear fixed	7F10	364
6F3C	SMS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	7F10	5280
6F40	MSISDN	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	7F10	28
6F42	SMSP	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	7F10	84
6F43	SMSS	EF	CHV1,CHV1,NEV,ADM,ADM	transparent	7F10	2
6F44	LND	EF	CHV1,CHV1,NEV,ADM,ADM	cyclic	7F10	84
6F49	SDN	EF	CHV1,ADM,NEV,ADM,ADM	linear fixed	7F10	336
6F4A	EXT1	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	7F10	39
6F4B	EXT2	EF	CHV1,CHV2,NEV,ADM,ADM	linear fixed	7F10	39
6F4F	NS	EF	CHV1,CHV1,NEV,ADM,ADM	linear fixed	7F10	75
6F54	SUME	EF	ADM,ADM,NEV,NEV,NEV	transparent	7F10	22
C000	NS	EF	ADM,ADM,NEV,NEV,NEV	linear fixed	7F10	42

²The sequence of privileges is related to, as explained in the paper, the execution of a defined set of commands issuable to a SIM card, namely *Read*, *Update*, *Increase*, *Rehabilitate* and, finally, *Invalidate*.