Overcoming Impediments to Cell Phone Forensics

Wayne Jansen
NIST
100 Bureau Dr., STOP 8930
Gaithersburg, MD 20899

Aurélien Delaitre
NIST
100 Bureau Dr., STOP 8930
Gaithersburg, MD 20899

Ludovic Moenner
NIST
100 Bureau Dr., STOP 8930
Gaithersburg, MD 20899

Abstract:
Cell phones are an emerging but rapidly growing area of computer forensics. While cell phones are becoming more like desktop computers functionally, their organization and operation are quite different in certain areas. For example, most cell phones do not contain a hard drive and rely instead on flash memory for persistent storage. Cell phones are also designed more as special-purpose appliances that perform a set of predefined tasks using proprietary embedded software, rather than general-purpose extensible systems that run common operating system software. Such differences make the application of classical computer forensic techniques difficult. Also complicating the situation is the state of the art of present day cell phone forensic tools themselves and the way in which tools are applied. This paper identifies factors that impede cell phone forensics and describes techniques to address two resulting problems in particular: the limited coverage of available phone models by forensic tools, and the inadequate means for validating the correct functioning of forensic tools.[1]

1. Introduction

Nearly a billion cell phones were sold worldwide in 2006 and projections for 2007 and beyond continue to rise. Over the last decade the capabilities and features of cell phones, such as increases in performance and storage capacity, and additions of document and multimedia handling functionality, have also continued to improve rapidly, turning cell phones into data reservoirs that can hold a broad range of personal and organizational information.

From an investigative perspective, digital evidence recovered from a cell phone can provide a wealth of information about the user, and each technical advance in capabilities offers greater opportunity for recovery of additional information. While the outlook should be positive, a number of factors conspire to impede progress in cell phone forensics.

1.1 Current Conditions

Forensic software tools are a primary means for recovering digital evidence from cell phones. Unlike the situation with personal computers, mobile phone manufacturers employ many different proprietary operating systems and storage structures. Data recovery is usually carried out through logical instead of physical acquisition, using one or more protocols supported by the device. The protocols include standardized and proprietary device synchronization protocols, command interface protocols, and diagnostic protocols.

Six manufacturers control about 80 percent of the cell phone market at any one time; the top two, Nokia and Motorola, led the group in 2006 with more than 50 percent [1, 2]. Approximately fifty other manufacturers hold the remaining 20 percent share of the market. New manufacturers occasionally enter the marketplace replacing others that leave. For example, the widely advertised iPhone from Apple is a new entrant this year. The number of models of phones that appear on the world market each year is considerable, with new releases from major manufacturers continually appearing throughout the year. Models of older functioning phones, though out of date, can remain in use for years after their initial release. Phone models introduced into one national market can also be used in other market areas by replacing the identity module of a phone (e.g., a GSM subscriber identity module) with one from another carrier or through roaming features.

New phone models often have functional differences from previous models that a forensic phone tool needs to take into account to recover and report data properly. When a new phone appears, a tool manufacturer must decide whether to adapt its tool for the phone, purchase exemplars for study, create and test an update containing support for the phone, and finally distribute the tool update to the user. Tool updates need to be issued periodically to minimize this latency period and keep the software current with the latest available phone models. Complicating things further, variations in data storage

[1] Certain commercial products and trade names are identified in this paper to illustrate technical concepts. However, it does not imply a recommendation or an endorsement by NIST.

location assignments can occur in a specific model of phone that is subsidized and supplied by different network carriers, due to adaptations made for the carriers by the manufacturer. Firmware updates sent out by a network carrier can also affect data locations [3].

The time required for needed tool updates to become available, therefore, can be lengthy, putting forensic specialists constantly behind the power curve. At times the situation may necessitate turning to alternative means to acquire data from a recently released model of phone. Most cell phone forensic specialists use a collection of both forensic and non-forensic tools along with other accessories to form their "toolbox." Tools not designed specifically for forensic purposes are questionable, however [4]. Some contend that the current situation is likely to continue, keeping the cost of examination significantly higher than if a few standard operating systems prevailed [5].

Phone managers are sometimes turned to as a way to recover data automatically when no suitable forensic tool is available. Phone managers are often available directly from the manufacturer of the phone and kept up to date with support for newly released models. The software allows user data to be synchronized with a desktop computer and changes to be made through the user interface. Since phone managers have the ability to both read and write data to a phone, they can be problematic from a forensic perspective, if used without applying proper testing and procedural controls. Many anecdotes abound of a practitioner accidentally or unknowingly writing data to a phone using such a tool. In one case, a forensic specialist, managing his personal phone using a non-forensic tool, was assigned an urgent task to examine a seized phone that required the same tool, and in the process accidentally merged his personal data with that recovered from the seized phone.

Forensic tools are also imperfect. In the rush to apply a tool, proper validation procedures may be overlooked. This is particularly true of updates to or new versions of a tool that has been validated earlier. Product training more often than not neglects tool validation, emphasizing instead tool functionality and use. Yet subtle and debilitating regression errors have occurred occasionally with software tool updates or new versions of tools, and are likely to continue to happen in the future.

Tool validation can be time consuming and complicated. It requires the population of data onto a device, followed by the manual comparison between what was populated and what the tool recovered. As device capacities and functional capabilities improve, the task also becomes more substantial. Furthermore, constructing test data that reflects important but troublesome areas and affects significant portions of memory adds to the burden.

1.2 Plausible Improvements

When taken together, all of the aforementioned factors significantly impede the practice of cell phone forensics. Many of the prevailing conditions are not readily resolved or likely to be changed. Nevertheless, it raises the question "How can the situation be improved?" In considering possible improvements, two solutions surfaced. The first is to develop a forensically sound way to address the problem of latency in coverage of newly available phone models by forensic tools. The approach, called phone manager protocol filtering, builds on the functionality of phone managers available from device manufacturers. The second solution is to provide a means to establish a baseline for validating the correct functioning of forensic tools. The approach, called identity module programming, populates the identity modules of certain classes of phones with reference test data, which can be used as a baseline for validation of forensic tools that recover evidence from these devices. The remainder of this paper outlines both solutions.

2. Phone Manager Protocol Filtering

As mentioned earlier, phone managers are a potential tool for automated data recovery of common types of core user data, such as phonebook entries and photos. A phone manager available from the cell phone's manufacturer is often kept up to date for the phone and also other phone models in the product line. For example, both Nokia and Motorola follow this approach for their cell phones. However, phone managers are not forensic tools. Additional steps must be taken to safeguard against altering data on the phone, including validating the phone manager's operation, producing a cryptographic hash of the acquired data, and testing and verifying the procedures to be followed. Even an experienced forensic specialist taking all available precautions could accidentally write data to a phone using such a tool.

Phone managers typically use the same protocols as forensic tools to recover data. Forensic cell phone tools avoid the problem of altering data on a phone by restricting the command options of the protocol used to communicate with the device to only those that are either known to be safe or involve very minor forensic issues. An obvious way to gain the same advantage for phone managers is to apply a filter somewhere between the phone manager application and the device being managed, which blocks harmful protocol commands from propagating. Filtering is an often used technique in computer forensics, commonly implemented in hardware or software write blockers for disk and USB device interfaces.

Most phone managers run under the Windows operating system and are distributed in binary form for installation. Figure 1 gives a general overview of the possible locations to implement a phone manager filter – at the programming interfaces between phone manager code and the communication library files, between the library files and the communication stack, within the communication stack, and between the communication stack and the device. After reviewing the alternatives, the approach selected was to avoid interception at the communications stack or at the device interface and instead move further upstream and target the software programming interface to the library.

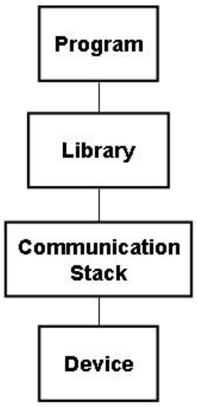

Figure 1: Filter Placement

Communications with cell phones occurs over a serial COM or USB port. Most serial port data transmission for Windows systems is done the same way as writing to a file. For example, the WriteFile function can be used to send data via a serial COM port. The same function also works with virtual serial ports established over USB, infrared, or Bluetooth communications. The filter could intercept the call to the application programming interface (API) for this function to capture the data, interpret the content, and return an appropriate response to the phone manager. Similarly, calls to other related functions, such as CreateFile and ReadFile, would need to be intercepted for the filter to work overall. The techniques used to insert code that can intercept commands at an API are the focus of the remainder of this section.

2.1 API Interception

API hooking is a term used to describe intercepting calls to a function for some purpose, usually to customize and extend its functionality and also to monitor aspects of an application. The target function may be in an executable application, a library, or a system DLL. In the case of Windows operating systems, the functions of interest are part of the so-called Win32 API. Hooking Win32 APIs is not new; security add-ons, such as personal firewalls and anti-virus applications, as well as malicious code, such as rootkits, have used these techniques to insert themselves seamlessly into an operating system. The interception process is performed at run time against a running process rather than modifying static binary images at rest.

Several different techniques have been used to hook Windows APIs. A common way is to alter the import address table (IAT) of a given module and replace the target function with the substitute function. The IAT contains the address of each imported function and used by the loader to map function calls to entry points of loaded routines. Alternatively, an unconditional jump can be inserted in the first few bytes of a target function to change the flow of execution to the substitute function. When the substitute function completes its task, control is returned to the modified function or, optionally, back to the calling program.

The approach being used for the phone manager filter is to have the substitute function serve as a wrapper for the target function, as illustrated in Figure 2 [6]. The first few instructions of the target function are replaced with a jump to the filter function, and the replaced instructions from the target function are preserved in a so-called trampoline function. The trampoline function acts like a relay, ending with a jump back to the target function to complete processing after the preserved instructions are executed. The filter function can either call the trampoline function to invoke the target function, or return directly to the calling program and bypass the target function altogether. The target function is also adjusted to return control to the filter function upon completion to allow the filter to perform any needed post function operations.

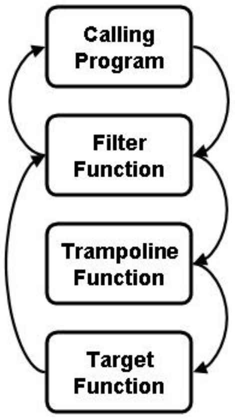

Figure 2: API Interception

2.2 Protocol Considerations

The Nokia PC Suite provides a good example of a candidate phone manager for protocol filtering. The current version for the U.S. market supports

Byte	0	1	2	3	4 - 5	6 - n	n+1 - n+2
Contents	Frame ID	Destination	Source	Command	Length	Data	Checksum

Figure 3: FBUS Frame

approximately 75 models, including the very latest. The versions for other countries support about the same number of models, some of which are different from the models in the U.S. version. PC Suite can be used for a number of things, including copying personal data (e.g., phonebook entries) to a computer for safekeeping, transferring images, video clips, and other files from the phone to a computer, and viewing contacts and messages on a device. Certain features work only when used with those models of Nokia phone that embody compatible functionality. Various types of communications with the phone are supported, including serial COM and USB cables. Wireless options also exist.

The Nokia PC Suite uses a proprietary protocol called the FBUS protocol to perform its functions. The FBUS protocol is used to extract the phone book, call logs, SMS messages and calendar entries from the phone. Another protocol, OBEX, which rides over the FBUS frames, is also used to extract media files, ring tones and downloaded applications that are present. The physical interface is a bidirectional serial communication bus that runs at 115,200 bps [7].

The FBUS frame is byte oriented. Figure 3 illustrates its composition. The first byte of the frame, byte 0, holds the hexadecimal value of the identifier for the FBUS protocol. The value 1E is the frame identifier for cable. Bytes 1 and 2 respectively contain the destination and source addresses [7, 8]. For data sent to the phone, the destination address is 00. The source address for the personal computer is 10 or 0C. Byte 3 contains the command identifier, which potentially supports up to 256 (i.e., 2^8) commands. Bytes 4 and 5 hold the length of the data that follows. The bytes following byte 5 convey the data segment of the frame. The last byte of the data segment contains a 3-bit sequence number. The last two bytes of the frame contain a checksum [7, 8]. Only frames of an even length are transmitted. A byte of all zeros is inserted before the checksum, if needed, to make the total length of the frame even.

The FBUS protocol is an acknowledged request-response protocol, with the phone manager issuing command requests and the phone answering [7, 8]. Responses use the same command identifier as the request being answered, but reverse the source and destination address. Every request or response, except for the first request, is prepended with an acknowledgment frame indicating receipt of the last protocol element sent by the other party, as illustrated in Figure 4. This convention means that the filter needs to send a properly constructed receipt acknowledgment for any blocked command, in addition to providing an appropriate response. Otherwise, the phone manager will resend the disallowed frame.

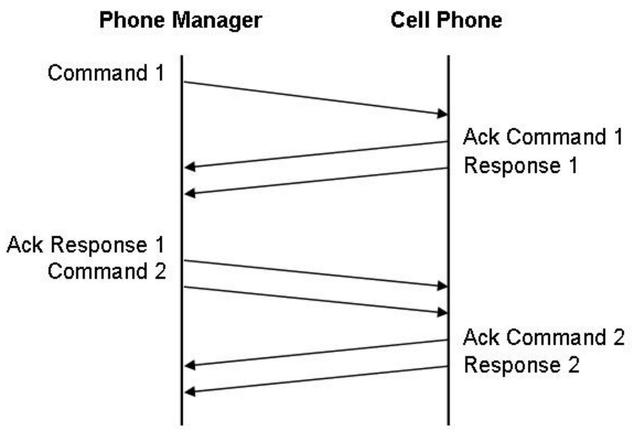

Figure 4: FBUS Communication

Table 1 illustrates the FBUS protocol exchanges used by two different forensic tools to acquire the identifier of the handset, known as the International Mobile Equipment Identifier (IMEI) from the same Nokia 6101

Table 1: IMEI Recovery

	(Hex) Request / Response	**(ASCII) Request / Response**
PhoneBase	1E 00 10 1B 00 07 00 01 00 00 41 01 41 00 0E 1C	. . . . A . A . . .
	1E 10 00 7F 00 02 1B 01 05 6C 1E 10 00 1B 00 1C 01 39 00 01 00 01 41 14 00 10 **33 35 36 36 36 31 30 30 35 37 30 34 30 39 32** 00 01 42 5B 50	. . . ▯ . . . . . l . . . . 9 . . . . A . . . **3 5 6 6 6 1 0 0 5 7 0 4 0 9 2** . . B [P

	(Hex) Request / Response	(ASCII) Request / Response
Secure View	55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 … (9 more rows) 1E 00 10 1B 00 07 00 04 00 00 41 01 60 00 2F 19	U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U … . . . . A . ` . / .
	1E 10 00 7F 00 02 1B 00 05 6D 1E 10 00 1B 00 1C 04 39 00 01 00 01 41 14 00 10 **33 35 36 36** **36 31 30 30 35 37 30 34 30 39** **32** 00 01 45 5E 57	. . . ▯ . . . . . m . . . . 9 . . . . A . . **. 3 5 6 6** **6 1 0 0 5 7 0 4 0 9 2** . . E ^ W

cell phone. The value of the IMEI is 356661005704092, highlighted in bold within the response entry. Both forensic tools send a request with the command of 1B to recover the IMEI. The second tool listed prefixes the request with a series of synchronization characters of 55 hexadecimal. Receipt of the request is acknowledged by the phone with an acknowledgment (i.e., command value of 7F hexadecimal), immediately followed by the response containing the value of the IMEI.

Because the FBUS protocol is proprietary, the function of all command identifiers is not known. However, over the years many of the commands have been determined through experimentation by various parties. Furthermore, the communications of forensic tools, such as the ones mentioned above, can be monitored to identify commands considered safe by tool manufacturers. To avoid propagating frames containing unsafe commands to a phone, the phone manager filter incorporates a white list of known commands considered safe; all other command frames are blocked.

Initial testing of the prototype implementation indicates that the approach could provide a practical and effective solution for addressing the latency in forensic tool coverage of available phones. Intercepting low-level Windows APIs, as opposed to higher-level internal APIs in the application, should also allow the solution to be used with phone managers from other cell phone manufacturers. Reprogramming the filter for the different protocols involved would, needless to say, be required. As with any forensic tool, the resulting filtered phone manager program requires validation before its use. The next section, though not pertaining directly to validation of forensic tools for handsets, gives an idea of the rigor needed.

3. Identity Module Programming

Subscriber Identity Modules (SIMs) are synonymous with mobile phones and devices that interoperate with GSM cellular networks. Under the GSM framework, a cellular phone is referred to as a Mobile Station and is partitioned into two distinct components: the Subscriber Identity Module (SIM) and the Mobile Equipment (ME). As the name implies, a SIM is a removable component that contains essential information about the subscriber. The ME, the remaining radio handset portion, cannot function fully without one. The SIM's main function entails authenticating the user of the cell phone to the network to gain access to subscribed services. The SIM also provides a store for personal information as well as operational information. Another class of SIMs being deployed in third generation (3G) Universal Mobile Telecommunications Service (UMTS) networks is UMTS SIMs (USIMs). USIMs are enhanced versions of present-day SIMs, containing backward-compatible information.

At its core, a SIM is a special type of smart card that typically contains a processor and between 16 and 256 KB of persistent electronically erasable, programmable read only memory (EEPROM). It also includes random access memory (RAM) for program execution, and read only memory (ROM) for the operating system, user authentication and data encryption algorithms, and other applications. The hierarchically organized file system of a SIM resides in persistent memory and stores such things as names and phone number entries, text messages, and network service settings. Depending on the phone used, some information on the SIM may coexist in the memory of the phone or reside entirely in the memory of the phone instead of available memory on the SIM.

Some of the earliest general-purpose forensic tools for mobile phones targeted SIMs, not only because of detailed specifications available for them, but also because of the highly relevant and useful digital evidence that could be recovered. A recent assessment of the capabilities of present day forensic tools to recover evidence from SIMs, however, noted discrepancies between the test data placed on a SIM and that recovered and reported in every tool [9]. They include the inability to recover any data from certain SIMs, inconsistencies between the data displayed on screen to the user and that generated in the output reports, missing truncated data in reported or displayed output, errors in the decoding and translation of recovered data, and the inability to recover all relevant data. Moreover, updates or new versions of a tool, on occasion, were less capable than a previous version

Validating each version of a forensic SIM tool is an essential quality assurance measure. The results aid in deciding how to compensate for any noted shortcomings or whether to switch to a new version or update of the tool that may be available. Validation should be carried out when first choosing a forensic tool to ensure its acceptability and redone when updates or new versions of the tool become available to maintain consistency of results. Validating a tool entails defining a comprehensive set of test data, loading it onto the device, and following defined procedures to acquire and recover the test data for comparison [10].

While tool validation is essential, building reference SIMs that contain comprehensive test data can be time consuming and difficult to carry out, normally requiring the use of various SIM editing tools and handsets to populate the data. For example, variances exist between SIMs from different manufacturers, such as dissimilar file capacities allocated for the same set of entries (e.g., phonebook list) and diverse sizes for the same data fields (e.g., name). Different character encodings may also apply for various languages of interest (e.g., English versus Asian characters). For many, a comprehensive validation effort is beyond their means and a lesser tack is taken. The focus of the remainder of this section is an approach for automating the population of reference test data onto the file system of a SIM, which attempts to address those differences and simplify the process.

3.1 File System Considerations

The file system of a SIM is organized as a hierarchical tree structure, composed of three types of elements: the root of the file system (MF), subordinate directory files (DF), and files containing elementary data (EF) [11]. Figure 5 illustrates the structure of the file system. The EFs under DF_{GSM} and $DF_{DCS1800}$ contain mainly network-related information for different frequency bands of operation. The EFs under $DF_{TELECOM}$ contain service-related information.

Each element of the file system has a unique numeric identifier assigned. The identifier can be used to reference an element when performing an operation, such as reading the contents of an EF, in the case of a forensic tool [12]. Operations are accomplished through command directives called Application Protocol Data Units (APDUs). A phone handset uses APDUs when communicating with a SIM [11]. The APDU protocol is a simple command-response exchange, with a single response to each command issued. The APDU protocol must be used to convey commands to perform update operations on a referenced EF to populate it with test data.

SIMs use three structures for EFs: transparent files, linear fixed files, and cyclic files. Transparent files are a sequence of bytes that can be accessed via an offset. Linear fixed files are a list of records of the same length that can be accessed by absolute record number, via a record pointer, or by seeking a record by pattern. Cyclic files comprise a circular queue of records maintained in chronological order, which are accessible the same as with linear fixed records, with the oldest overwritten if storage is full.

The various types of digital evidence of interest to a forensic specialist exist in EFs scattered throughout the file system. Besides the standard files defined in the GSM specifications, a SIM may contain non-standard files established by the network operator [12]. The following general categories of evidence in standard elementary data files have importance [9]:

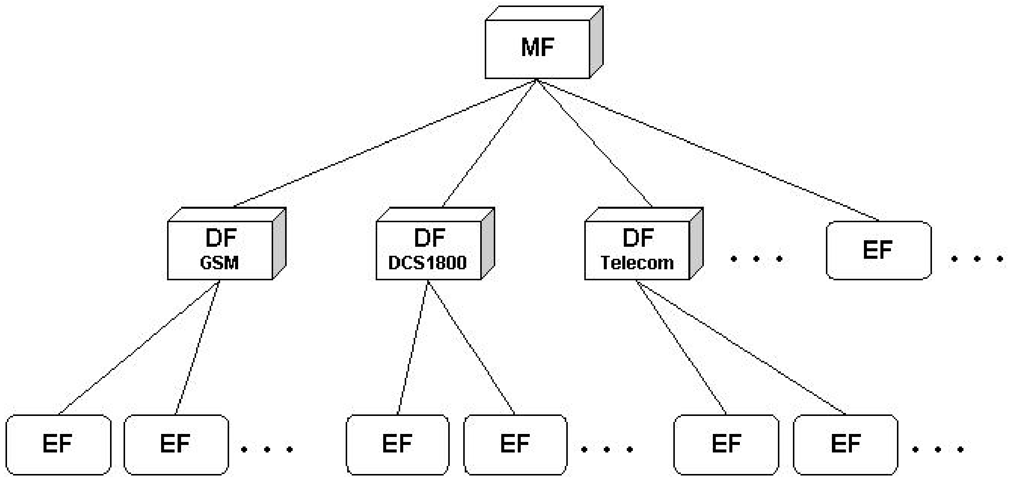

Figure 5: SIM File System

- Phonebook and Call Information, known respectively as the Abbreviated Dialling Numbers (ADN) and Last Numbers Dialled (LND).
- Messaging Information, including both Short Message Service (SMS) text messages and Enhanced Messaging Service (EMS) multimedia messages.
- Location Information, including Location Area Information (LAI) for voice communications and Routing Area Information (RAI) for data communications.

News articles of high profile cases occasionally contain illustrative examples where such recovered evidence was used successfully in an investigation. The following are two examples:

- Text Message and Call Data [13] – "A pastor of the Pentecostal congregation in the small community of Knutby was sentenced to life in prison for persuading one of his lovers (the au pair) to shoot and kill his wife and trying to kill the husband of another mistress. Two days after the murder, the pastor's au pair Sarah S. claimed that she did it. Despite her claims … the police believed she had an accomplice."
  "The strongest evidence against the pastor was the extensive communication through text messages and voice calls between him and the au pair on the day of the murder and just before that. What they did not know was that their (anonymously sent and) carefully deleted text messages were possible to recover."
- Location Data [14] – "Mr Bristowe told BBC News Online: 'It was mobile phone evidence which made the police look more closely at Huntley. He had been Mr. Useful, helping them to search the college grounds, but when they checked Jessica's phone and discovered when and where it had been switched off alarm bells began to ring… (Jessica's phone) disengaged itself from the network, in effect it says goodbye' at 1846 BST on the Sunday when the girls disappeared. Jessica's phone contacted the Burwell mast when it was turned off."
  "'The police provided us with a map of the route they thought the girls would have taken, and the only place on that route where the phone could have logged on to Burwell (and disengaged itself) was inside or just outside Huntley's house.' It is believed to be that crumb of crucial evidence which forced Huntley to change his story earlier this year and suddenly admit the girls died in his bathroom."

The failure of a forensic tool to correctly recover and report such relevant SIM data greatly impedes the ability of the forensics specialist and jeopardizes the credibility of the overall results.

3.2 Design and Implementation

The overall data flow of the identity module programmer (IMP) is given in Figure 6. Conceptually the process is straightforward. Reference data is read by the program and used to populate the SIM shown at the right. Any errors are logged and a summary of the results is reported, once the appropriate access conditions for the SIM (i.e., defined in Card Data) are enabled. The reference test data could be generated manually or automatically using a preprocessor.

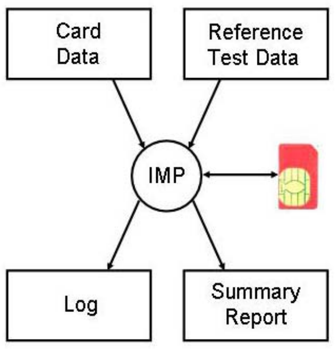

Figure 6: IMP Overview

For IMP to communicate with a SIM, the SIM must be removed from a phone and placed into an appropriate reader. Either a specialized reader that accepts a SIM directly or a general-purpose reader for a full-size smart card can be used, provided that it is compatible with the PC/SC (Personal Computer/Smart Card) specification, a popular general-purpose architecture for smart cards [15]. For full-size card readers, a standard-size smart card adapter is needed to house the SIM for insertion into the reader.

Reference data can be populated on a SIM only when the correct access conditions for an EF are satisfied to enable update (i.e., write) operations to be performed. However, different access conditions prevail for the various EFs of interest needing to be populated. Common access conditions include Personal Identification Number (PIN) verified and administrator code verified access. While PINs are usually available for most production SIMs, administrator codes are normally kept by the network carrier and not made available. One exception is test SIMs, which are available from most SIM manufacturers for development purposes. The PIN values and administrator access codes are usually provided by the manufacturers together with the test SIMs. As one might expect, test SIMs allow a greater range of reference data to be populated. Nevertheless, production SIMs can still form a useful baseline for validation, as long as EFs not populated by the tool are noted and taken into account during tool validation. Both types of SIMs can be used with IMP.

Because of the variation possible between SIMs, the defined reference test data may exceed the capacity of an EF or the size of the field. Attempts to exceed either type of limit are detected and processed by the SIM itself. Out of bounds references are denied and overly long data are truncated to the space available. IMP logs any deviations between the populated data and reference data as they occur. A summary of all reference test data populated by IMP appears in the output report, as well as the contents of certain EFs that could not be populated, which together provide a known definitive baseline for validation.

The initial set of reference data was drawn from test scenarios recently used in assessments of forensic SIM tools involving basic, location, EMS, and foreign language data. Basic data includes subscriber (e.g., the IMSI and ICCID elementary files), phonebook (i.e., the ADN elementary file), recent call (i.e., the LND elementary file), and SMS message related information. Besides common input data, known problematic input, such as the use of a special character for a phonebook name entry, were included. Foreign language data involves text messages and phonebook data that are expressed in a language other than English. EMS data consist of text messages more than 160 characters in length and containing black and white bitmap images or monophonic melodies. EMS messages can also contain formatted text with different font styles and fonts. Location data includes location-related information, such as the last location area or routing area where the phone disengaged from the network (i.e., the LOCI and LOCIGPRS elementary files).

```
<phonebookentry>
      <description enc="ucs2">阿家里面于</description>
      <address>
            <ton>international</ton>
            <npi>telephone</npi>
            <number>1444412345678</number>
      </address>
</phonebookentry>
```

Figure 7: Example XML Phonebook Entry

XML is used to represent test data for input to IMP. XML is a popular syntax, able to be processed by computers and, with some effort, also understood by humans. Many XML editors exist, as well as tools for defining data type descriptions and schemas against which data representations can be constructed and automatically verified. These characteristics motivated its choice. Figure 7 shows an example phonebook entry for an Asian name and an international telephone number encoded in XML.

One consideration in constructing the XML schema is defining ways to represent deleted entries in the test data. No delete operation exists for SIMs. Instead, deletion is accomplished by updating information in an elementary file with strings of hexadecimal "FF." The one exception involves SMS message content, by which a status flag is used to indicate a deleted entry instead of "FF" overwrite, allowing the content to be recovered. The structure of an elementary file affects the way deleted information is represented. For example, for linear fixed files, a record number could be used to specify the content of the indicated record, whereby a deleted entry is simply never referenced. However, that choice might induce errors in the reference data set, such as duplicate entries, which would not be automatically detectable by an XML validation tool. Instead of record numbers, however, data for such record entries could be listed sequentially and populated in the order of appearance. Delete entries can then be designated by a special tag, which results in the creation of a gap in the file structure.

Most forensic SIM tools run under the Windows operating system, making it a logical platform for implementing IMP. To allow other operating systems besides Windows to be supported, IMP was written in the Java programming language. IMP uses and extends an open source programming interface called Java Card Communication Access Library (JACCAL) to exchange APDUs with the SIM. A SAX parser is also used to interpret the reference test data represented in XML.

4. Conclusions

Cell phone forensics is an emerging discipline. Various impediments exist that create problems for forensic specialists working in this area, and need to be overcome for the discipline to flourish. The two techniques presented in this paper attempt to resolve two problems: the latency in coverage of newly available phone models by forensic tools, and the lack of readily available reference material to use as a comprehensive baseline for validating the correct functioning of forensic SIM tools.

The basic techniques described in the paper are extendable beyond the specific examples given. In the case of phone manager protocol filtering, the technique could be applied to phone managers from cell phone manufacturers other than Nokia, albeit with a filter programmed for the different protocols that may be involved. Similarly, the technique for populating SIMs could be applied to other types of identity modules in the marketplace, with the appropriate modifications applied. More important, the discussion will hopefully inspire others to step back and take a broader look at existing problems in this discipline, and consider better solutions than those given or address the other outstanding problems that remain.

5. References

[1] Nokia and Motorola Gain Market Share as Arena Grows, International Herald Tribune, Tech/Media November 22, 2006, <URL: http://www.iht.com/articles/2006/11/22/yourmoney/mobile.php>.

[2] Nokia and Motorola Account for Nearly 50% of Worldwide Sales, Mobiledia, August 25, 2005, <URL: http://www.mobiledia.com/news/35125.html>.

[3] Robert Vamosi, Cell Phone 'CSI,' CNET Reviews, May 25, 2007, <URL: http://reviews.cnet.com/4520-3513_7-6737586-1.htm>.

[4] Annalee Newitz, Courts Cast Wary Eye on Evidence Gleaned From Cell Phones, WIRED, May 10, 2007, <URL: http://www.wired.com/politics/law/news/2007/05/cellphone_forensics>.

[5] Tyler Moore, The Economics of Digital Forensics, Fifth Annual Workshop on the Economics of Information Security, June 2006, <URL: http://www.cl.cam.ac.uk/~twm29/weis06-moore.pdf>.

[6] Galen Hunt, Doug Brubacher, Detours: Binary Interception of Win32 Functions, 3rd USENIX Windows NT Symposium, Seattle, WA, July 1999, <URL: research.microsoft.com/~galenh/Publications/HuntUsenixNt99.pdf>.

[7] Wayne Peacock, An Introduction to Nokia F-Bus, Embedtronics, April 2005, <URL: http://www.embedtronics.com/nokia/fbus.html>.

[8] Paul McCarthy, Forensic Analysis of Mobile Phones, BS CIS Thesis, University of South Australia, School of Computer and Information Science, Mawson Lakes, October 2005, <URL: http://esm.cis.unisa.edu.au/new_esml/resources/publications/forensic%20analysis%20of%20mobile%20phones.pdf>.

[9] Wayne Jansen, Rick Ayers, Forensic Software Tools for Cell Phone Subscriber Identity Modules, Conference on Digital Forensics, Association of Digital Forensics, Security, and Law (ADFSL), April 2006, <URL: http://csrc.nist.gov/mobilesecurity/publication/pp-SIM%20tools-final.pdf>.

[10] Amanda Goode, Forensic Extraction of Electronic Evidence from GSM Mobile Phones, IEE Seminar on Secure GSM & Beyond, Digest No. 2003/10059, February 11, 2003.

[11] Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface, 3rd Generation Partnership Project (3GPP), TS 11.11 V8.13.0 (Release 1999), Technical Specification, June 2005.

[12] Casadei, F. et al., Forensics and SIM cards: an Overview, International Journal of Digital Evidence, Volume 5, Issue 1, Fall 2006, <URL: http://ieeexplore.ieee.org/iel5/10612/33521/01592525.pdf?isnumber=&arnumber=1592525>.

[13] Robert Burnett, Ylva Hård af Segerstad, The SMS Murder Mystery: the dark side of technology, Safety & Security in a Networked World: Balancing Cyber-Rights & Responsibilities, September 2005, <URL: http://www.oii.ox.ac.uk/microsites/cybersafety/extensions/pdfs/papers/robert_burnett.pdf>.

[14] Chris Summers, Mobile phones - the new fingerprints, BBC News Online, December 18, 2003, <URL: http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/1/hi/uk/3303637.stm>.

[15] PC/SC Workgroup (2005) Interoperability Specification for ICCs and Personal Computer Systems, Part 1 - Introduction and Architecture Overview, Revision 2.01.00, June 2005, <URL: http://www.pcscworkgroup.com/specifications/files/pcsc1_v2.01.0.pdf>.