Symbian Smartphone Forensics and Security: Recovery of Privacy-Protected Deleted Data

Vrizlynn L.L. Thing and Darell J.J. Tan

Digital Forensics Lab Cryptography & Security Department
Institute for Infocomm Research, Singapore
{vriz,jjdtan}@i2r.a-star.edu.sg

Abstract. In this paper, we discuss our proposed method to acquire privacy-protected data from Symbian smartphones running the latest OS version 9.4, S60 5th Edition, and smartphones running the prior OS version 9.3, S60 3rd Edition. We then present our reverse-engineering analysis work on the active and deleted Short Message Service (SMS) message recovery from the on-phone memory in the Symbian smartphones. We describe the encoding and format of the raw data of the SMS messages so as to achieve an automated parsing and recovery of the messages. Our experiments on various sent, received, draft and deleted messages showed that we were able to recover both the active (in its entirety) and deleted SMS messages (partially) correctly and automatically.

Keywords: Symbian forensics, security, memory analysis, mobile phones, smartphones, data acquisition, deleted SMS message recovery.

1 Introduction

As mobile phones are becoming increasingly prevalent and are constantly evolving into "smarter" devices (i.e. smartphones with higher processing power and enhanced features), capabilities to perform in-depth forensics on these devices also become essential. However, most current mobile phone forensics tools are still restricted to the acquisition and analysis of basic active files and data (i.e. logical data acquisition) on the Subscriber Identity Module (SIM), memory cards and the internal flash memory [1–7].

In the event that private application data is isolated and data-caging is in place, such security mechanisms prevent in-depth acquisition of important evidentiary data. For example, current Symbian deleted SMS recovery tools [8] have a limitation as they are only capable of recovering deleted SMS messages residing in the SIM card. SMS entries on the SIM card are marked as deleted or active. To undelete an SMS, the tools simply change the state flag in the allocation table from "free" to "in use". However, the real challenges arise when it is necessary to recover deleted SMS messages residing in the internal phone memory originally. This scenario is common due to it being a default configuration in smartphones these days (because of the memory limitation in the SIM

T.W. Chim and T.H. Yuen (Eds.): ICICS 2012, LNCS 7618, pp. 240–251, 2012.

card). Therefore, the capability to recover such deleted SMS messages to aid in forensics investigation is very important and necessary.

In this work, we focus on the recovery of deleted SMS messages from Symbian smartphones. We propose a method to conduct an in-depth evidentiary data acquisition from Symbian smartphones running the latest OS version 9.4, S60 5th Edition, and the prior OS version 9.3, S60 3rd Edition[1]. The acquisition method supports the retrieval of the relevant SMS message data from the Symbian smartphones. We also design an SMS recovery tool which accesses the associated data to reconstruct the deleted SMS messages.

Our main contributions in this work include: (i) creating our customised certificate and utility module to secure the required access to the phone, while at the same time, preserving its security protection mechanism against other softwares (ii) reverse-engineering and analysing the relevant data to support the reconstruction of deleted SMS messages, (iii) building the tool to acquire the data and reconstruct deleted SMS messages residing in the on-phone memory of Symbian smartphones, and (iv) experimenting on the smartphones which run the latest Symbian OS version 9.4, S60 5th Edition, and older OS version 9.3, S60 3rd Edition (i.e. Nokia N97 and E72, respectively). To the best of our knowledge, this is the first work on the recovery/carving of privacy protected deleted data from the Symbian smartphones since the introduction of its platform security framework in the S60 3rd Edition OS.

The rest of the paper is organised as follow. In Section 2, we present an overview of the existing work on mobile phone forensics research. We describe the in-depth acquisition and reverse-engineering experimental work with regard to the Symbian privacy-protected SMS data in Section 3. Future work is described in Section 4 and conclusions follow in Section 5.

2 Mobile Phone Forensics Research

In an early work [1], Willassen researched on the forensic investigation of GSM phones. The author presented the types of data of forensic relevance, which can exist on the phones, the SIM and the core network, and emphasized the need for more sound mobile forensics procedures and tools.

In [2], Casadei et al. presented their SIMbrush tool developed for both the Linux and Windows platforms. The tool relied on the PCSC library and supported the acquisition of the entire file system, including the non standard files, on the SIM. However, files with restricted read access conditions could not be extracted.

In [3], Kim et al. presented a tool to acquire the data from a Korea CDMA mobile phone's internal flash memory. The tool communicated with the phone

[1] S60 4th Edition does not exist and the next edition after the 3rd Edition is the 5th. Since S60 3rd Edition, Symbian phones begin to use a "hardened version" of its OS, which includes capabilities restrictions and a platform security framework. Versions prior to the 3rd and 5th Edition are not relevant to our work as forensics acquisition was technically easier [5].

through the RS-232C serial interface and was able to acquire the existing files on the phone using the underlying Qualcomm Mobile Station Modem diagnostic mode protocol.

In [4], Mokhonoana and Olivier proposed an on-phone forensic tool to acquire the active files from a Symbian OS v7 phone and store it on the removable media. Instead of interfacing with the PC connectivity services, the tool interacted with the operating system to perform a logical copy of the files. The tested phone was Sony Ericcson P800. One main limitation of the tool was that those files in use could not be copied (e.g. call logs, contacts).

In [5], Distefano et al. proposed the mobile phone internal acquisition technique on the Symbian OS v8 phones. The mobile phone data was acquired using a tool residing on the removable media, instead of the PC/mobile phone USB connection based approach. The tool utilized the Symbian S60 File Server API in the read-only mode. The authors carried out experiments comparing the tool with Paraben Device Seizure (USB connection to phone) and P3nfs (Remote access through Bluetooth). The tool took a longer time to perform the acquisition. However, it was able to acquire more data compared to the P3nfs. When compared with the Paraben Device Seizure, lesser data was acquired. However, the authors observed that the larger data size from Paraben was due to the additional information from its acquired data management.

In [6], Jansen et al. proposed a phone manager protocol filtering technique by intercepting the data between the phone and the phone manager. The objective was to address the latency in the coverage of newly available phone models by existing forensic tools. The authors also proposed an identity module programming technique, to populate the phone's SIM with reference test data, so as to provide a baseline for the validation of SIM forensic tools.

Surveys on Existing Tools

In [9], Jansen and Ayers evaluated the state-of-the-art SIM forensic tools to understand the capabilities and limitations in their data acquisition, examination and reporting functions. The tools surveyed included Cell Seizure, GSM .XRY, MOBILedit! Forensic, TULP 2G, Forensic Card Reader, Forensic SIM, SIMCon and SIMIS. It was observed that most information such as the IMSI and SMS/EMS could be found by the tools.

In [10], Bhadsavle and Wang evaluated the effectiveness of the Paraben Device Seizure on a test data populated T-Mobile locked SIM. They determined that 100% of the test data were retrieved.

In [11], Williamson et al. studied the performance of different mobile phone forensic tools (i.e. TULP 2G, MOBILedit! Forensic, Cell Seizure and Oxygen Phone Manager) on the Nokia phones. The authors concluded that some tools failed to deliver some promised features (e.g. MD5 hash was not found in MOBILedit!, SHA1 hash was not found in Cell Seizure).

In [12], Ayers et al. conducted a comprehensive study on the current mobile phone forensic tools and presented their findings in the NIST report. The

evaluated tools included the Paraben Device Seizure, Pilot-Link, GSM .XRY, Oxygen Phone Manager, MOBILedit!, BitPIM, TULP 2G, SecureView, PhoneBase2, CellDEK, SIMIS2, ForensicSIM, Forensic Card Reader, SIMCon and USIMdetective. The authors presented each tool's capabilities and limitations on a range of mobile phones, covering different operating systems, processor types, and hardware components. Some examples of the tested phones included Samsung SGH-i300, Motorola MPX220, Sony Ericsson P910a, Nokia 7610 and BlackBerry 7780.

In [7], Hoog presented the existing forensic evidence acquisition tools for the Android phone. The Android Debug Bridge (ADB) enabled interaction with the phone over the USB connection. Therefore, active files on the phone can be retrieved through the "adb pull" command. Other tools such as the Nandroid backup and Paraben Device Seizure also supported the extraction of files residing on the phone.

In [8], the SIM Manager tool attempted to recover deleted SMS messages from the SIM card. The SMS messages were stored in a file on the SIM card and each slot in the file contained an SMS. When an SMS was deleted, the slot was marked as "free". To undelete the SMS, the tool can mark the SMS slot as "in use". However, if the deleted SMS messages were stored in the internal phone memory instead, it would not be possible to recover them after deletion by using this tool.

As we can observe from the above-mentioned research, they focus on the acquisition and analysis of active files and data on the phones, with the exception of [8]. In the event of the need to recover deleted sensitive data from smartphones, it is often not possible to use the existing tools. They are either unable to perform a sufficient level of acquisition (due to privacy protection and data-caging present in smartphones) or an in-depth analysis requiring a reverse-engineering effort (due to the different encoding approaches of different OSes and applications). To support the reconstruction of deleted data, an in-depth acquisition and analysis of user data from the phones is required. In our work, we focus on designing and developing an acquisition and analysis tool for recovering deleted privacy protected data from the Symbian smartphones. Our work mainly involves obtaining a privileged access to the phone's privacy protected data in the Symbian smartphones' internal memory and subsequently reverse-engineering the acquired data for the recovery of deleted data. We describe our work using the example of deleted SMS recovery, in this paper.

3 Symbian Deleted SMS Recovery

The main challenge in performing an in-depth data acquisition from the internal memory of the Symbian smartphones arises from its built-in security mechanism, which prevents applications from accessing or even viewing private data of the other applications. Such data-caging mechanisms are also present in smartphones

running other OSes due to the need for privacy protection as smartphones contain an increasingly substantial amount of confidential and sensitive information such as saved passwords, application configuration settings and data, emails, contacts, notes, calendar of personal and business schedules, and messages.

3.1 Symbian AllFiles Capability

Since Symbian S60 3rd Edition OS, the platform security architecture was put in place to restrict access to sensitive functionalities. Applications or tools without having been granted the required capability would not be able to access the restricted Symbian APIs. The AllFiles capability in Symbian allows an unrestricted read access to the entire filesystem. We describe the method to obtain this capability, as follow.

In Symbian, the loader can only run executable files from the \sys\bin directory (from any drive). However, a process with the DiskAdmin capability can access the mapdrive API that maps sub-directories to unused drive letters on the phone [13]. We obtained the DiskAdmin capability through our Symbian Developer Certificate [14]. Therefore, it becomes possible to create a sub-directory \sys\bin under any directory in the phone, place executable files in it and then map it to a new drive letter, effectively placing these executable files into the valid executable path. Details on this mapping technique can be found in [13].

In addition, to eliminate the need to integrate the above-mentioned executable file loading module into all tools and applications requiring high-level privileges, we generate a customised Symbian Authority Certificate and place it into the phone using the mapdrive API. This is achieved by triggering the Symian Certificate Store (SWICertStore) Updater, so that the phone can store our customised certificate. With this certificate residing in the phone, we are now able to sign our tool to provide it with any required high-level privilege (i.e. Allfiles capability). The main advantage of this approach is that, unlike the existing HelloOX2 hack [15], which disables the Symbian install server's security mechanism, we do not compromise the security of the phone by leaving it vulnerable to malwares. We can authorise the installation of specific tool/s by signing it with this higher authority certificate, while the phone still preserves its security mechanism to guard against other unauthorised softwares.

3.2 Reconstruction of Active and Deleted SMS Messages

We observed that the data that is associated with the deleted SMS messages exists in the \Private\1000484b\Mail2\Index file of the phone's internal memory. Enabled by the AlLfiles capability (in Section 3.1), this file is now exposed. We retrieved this private Index file to support our analysis. Other than this Index file, there exist several sub-folders in the \Private\1000484b\Mail2 folder. The files within these sub-folders are associated with the active SMS messages. We conducted 50 experiments to obtain the SMS test data for analysis purpose.

Reconstruction of Active SMS Messages

Each file in the \Private\1000484b\Mail2 corresponds to an active SMS message. Based on our analysis on the data to identify and retrieve relevant information such as the contact number, the contact name, the message contents, the GMT setting and the timestamp information, we observed that each file can be consistently parsed according to the format in Table 1 to obtain its corresponding SMS message.

Table 1. SMS Header and Data Format

Offset	Length	Description	Value(s) and Meaning
0	4	SMS header marker	0x683C0010
16	1	SMS type	0x04 (Incoming) or 0x08 (Outgoing)
21	1 or 2	Packet length	
+36	1 or 2	SMS message length	
+0	<length>	SMS message	

The "offset" column in Table 1 refers to the absolute offset location within the file. A plus sign prefix, if present, represents a relative offset from the end of the previous entry.

Variable-Length Length Field

As shown in Table 1, a variable-length field is used for representing the length of the packet and message. This representation is very similar to the UTF-8 encoding, except that the lower bits were used here. If the lowest significant bit was set, the length field was extended to the following byte. Otherwise, the length field only occupied one byte. The following algorithm was designed to determine the actual packet and SMS message length.

1. Read the first "length" byte. Take note of the lowest significant bit.
2. Right-shift the byte by one bit. Assign this as the length.
3. If the lowest significant bit was set, right-shift the length by another bit. Read the next byte, left-shift it by six bits and logical OR it with the length in step 2.
4. Subtract one from the length to obtain the actual length.

The remaining data may contain an optional "sent SMS" data block (for outgoing message type only) and other essential information such as the timestamp, contact number or contact name. Table 2 shows the sent SMS data format and Table 3 shows the remaining data following either the SMS header (and message data) or the "sent SMS" data block (if present). The actual length for the contact name, contact number, SMS centre number and sender number has to be computed by dividing the provided length information by four.

Typically, the type of SMS (whether sent or received) can be determined from the message type indicator (1 byte at offset 16 as shown in Table 1). However, during our investigation, we observed a special case, which is the "PAIF message". The PAIF message is a configuration request SMS, which was sent by the

Table 2. Sent SMS Data Format

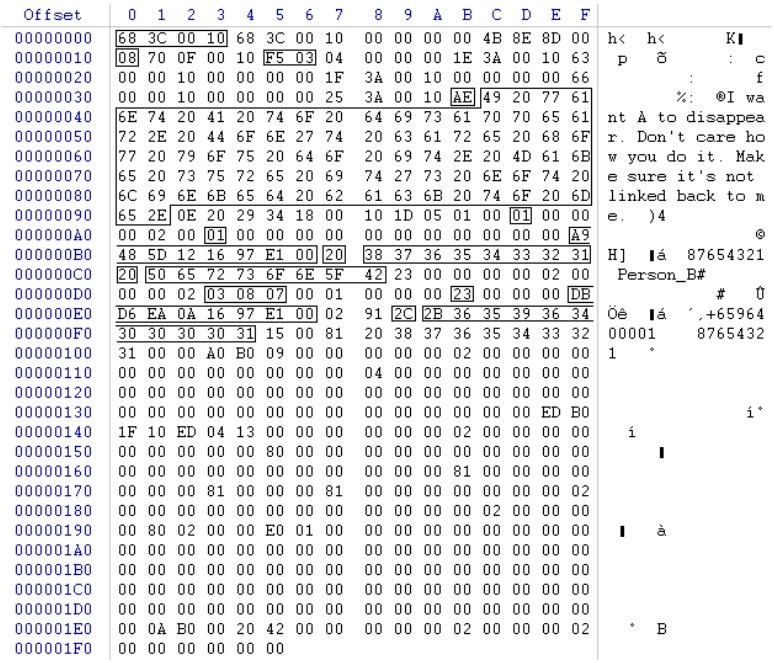
Offset	Length	Description	Value(s) and Meaning
+11	1	Sent SMS marker	0x01 (Data present) or 0x00 (Data not present, no need to parse further)
+5	1	Sent flag	0x00 (Not sent, draft SMS) or 0x01 (Sent SMS)
+11	8	Sent timestamp	Time when SMS was sent
+0	1	Contact number length	
+0	<length>	Contact number	e.g. "+6512345678"
+0	1	Contact name length	
+0	<length>	Contact name	e.g. "Person A"

Table 3. Remaining SMS Data Format

Offset	Length	Description	Value(s) and Meaning
+8 or +10	3	GMT value	For GMT offset computation (+8 if sent block not present, otherwise +10)
+5	1	SMS index entry number	Entry number (incrementally generated)
+3	8	SMS creation timestamp	Time when SMS was created (applicable for draft and sent SMS only)
+2	1	SMS centre number length	
+0	<length>	SMS centre number	e.g. "+6512345678"
+2	1	Sender number length	Only applicable for received message
+0	<length>	Sender number	e.g. "+6587654321" (Only applicable for received message)
+2	8	SMS received timestamp	Only applicable for received message

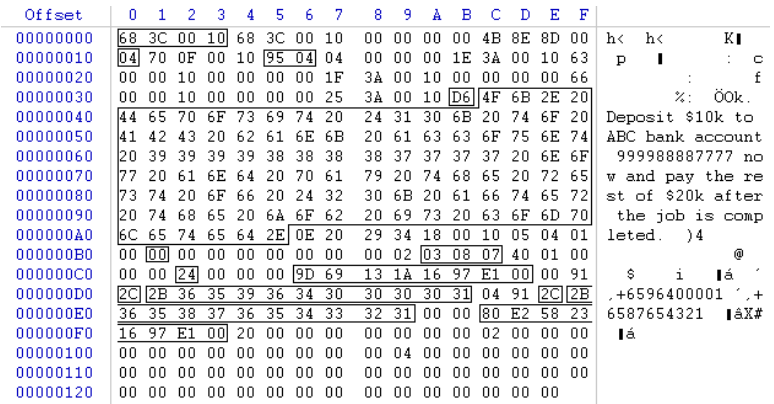
smartphone to the mobile service provider automatically. Upon investigation, the header indicates a received SMS (message type = 0x04) but it contains the optional "sent SMS" data block. To handle this special scenario, we first determine the SMS message type through its type indicator byte, and then override it as a Sent SMS if the Sent SMS header block is present. Otherwise, the SMS message is deemed to be a received SMS.

For an outgoing SMS, the "Sent flag" field within the "sent SMS" data block will determine whether the SMS has been sent. If it has not been sent, it will be classified as a draft SMS. Otherwise, it will be classified as a sent SMS. The timestamp represented the date and time as the number of microseconds since midnight, January 1st, 0 AD nominal Gregorian, as mentioned in [16]. An example of an active sent SMS message is shown in Figure 1 and an active received SMS message is shown in Figure 2.



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	68	3C	00	10	68	3C	00	10	00	00	00	00	4B	8E	8D	00	h< h< K▮
00000010	08	70	0F	00	10	F5	03	04	00	00	00	1E	3A	00	10	63	p õ : c
00000020	00	00	10	00	00	00	00	1F	3A	00	10	00	00	00	00	66	: f
00000030	00	00	10	00	00	00	00	25	3A	00	10	AE	49	20	77	61	%: ®I wa
00000040	6E	74	20	41	20	74	6F	20	64	69	73	61	70	70	65	61	nt A to disappea
00000050	72	2E	20	44	6F	6E	27	74	20	63	61	72	65	20	68	6F	r. Don't care ho
00000060	77	20	79	6F	75	20	64	6F	20	69	74	2E	20	4D	61	6B	w you do it. Mak
00000070	65	20	73	75	72	65	20	69	74	27	73	20	6E	6F	74	20	e sure it's not
00000080	6C	69	6E	6B	65	64	20	62	61	63	6B	20	74	6F	20	6D	linked back to m
00000090	65	2E	0E	20	29	34	18	00	10	1D	05	01	00	01	00	00	e.)4
000000A0	00	02	00	01	00	00	00	00	00	00	00	00	00	00	00	A9	©
000000B0	48	5D	12	16	97	E1	00	20	38	37	36	35	34	33	32	31	H] ▮á 87654321
000000C0	20	50	65	72	73	6F	6E	5F	42	23	00	00	00	00	02	00	Person_B#
000000D0	00	00	02	03	08	07	00	01	00	00	00	23	00	00	00	DB	# Û
000000E0	D6	EA	0A	16	97	E1	00	02	91	2C	2B	36	35	39	36	34	Öê ▮á ´,+65964
000000F0	30	30	30	30	31	15	00	81	20	38	37	36	35	34	33	32	00001 8765432
00000100	31	00	00	A0	B0	09	00	00	00	00	00	02	00	00	00	00	1 °
00000110	00	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ED	B0	í°
00000140	1F	10	ED	04	13	00	00	00	00	00	00	02	00	00	00	00	í
00000150	00	00	00	00	00	80	00	00	00	00	00	00	00	00	00	00	▮
00000160	00	00	00	00	00	00	00	00	00	00	00	81	00	00	00	00	
00000170	00	00	00	81	00	00	00	81	00	00	00	00	00	00	00	02	
00000180	00	00	00	00	00	00	00	00	00	00	00	00	02	00	00	00	
00000190	00	80	02	00	00	E0	01	00	00	00	00	00	00	00	00	00	▮ à
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001E0	00	0A	B0	00	20	42	00	00	00	00	00	02	00	00	00	02	° B
000001F0	00	00	00	00	00	00											

Fig. 1. Active Sent SMS Message



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	68	3C	00	10	68	3C	00	10	00	00	00	00	4B	8E	8D	00	h< h< K▮
00000010	04	70	0F	00	10	95	04	04	00	00	00	1E	3A	00	10	63	p ▮ : c
00000020	00	00	10	00	00	00	00	1F	3A	00	10	00	00	00	00	66	: f
00000030	00	00	10	00	00	00	00	25	3A	00	10	D6	4F	6B	2E	20	%: ÖOk.
00000040	44	65	70	6F	73	69	74	20	24	31	30	6B	20	74	6F	20	Deposit $10k to
00000050	41	42	43	20	62	61	6E	6B	20	61	63	63	6F	75	6E	74	ABC bank account
00000060	20	39	39	39	39	38	38	38	38	37	37	37	37	20	6E	6F	9999888877777 no
00000070	77	20	61	6E	64	20	70	61	79	20	74	68	65	20	72	65	w and pay the re
00000080	73	74	20	6F	66	20	24	32	30	6B	20	61	66	74	65	72	st of $20k after
00000090	20	74	68	65	20	6A	6F	62	20	69	73	20	63	6F	6D	70	the job is comp
000000A0	6C	65	74	65	64	2E	0E	20	29	34	18	00	10	05	04	01	leted.)4
000000B0	00	00	00	00	00	00	00	00	00	02	03	08	07	40	01	00	@
000000C0	00	00	24	00	00	00	9D	69	13	1A	16	97	E1	00	00	91	$ i ▮á ´
000000D0	2C	2B	36	35	39	36	34	30	30	30	30	31	04	91	2C	2B	,+6596400001 ´,+
000000E0	36	35	38	37	36	35	34	33	32	31	00	00	80	E2	58	23	6587654321 ▮áX#
000000F0	16	97	E1	00	20	00	00	00	00	00	00	00	02	00	00	00	▮á
00000100	00	00	00	00	00	00	00	00	00	04	00	00	00	00	00	00	
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Fig. 2. Active Received SMS Message

Reconstruction of Deleted SMS Messages

We observed that each deleted SMS entry in the index file contains the start marker, "0x6A0F00102C". To extract a deleted SMS, this start marker is firstly located. The index file is parsed according to the format shown in Table 4.

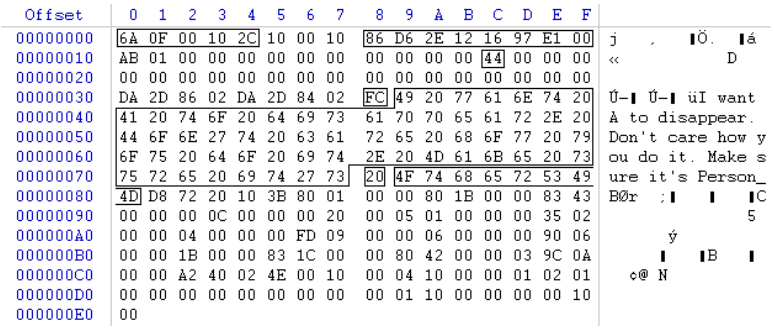
Table 4. Deleted SMS Header and Data Format

Offset	Length	Description	Value(s) and Meaning
0	5	SMS start marker	0x6A0F00102C
8	8	Timestamp	Sent or received timestamp
28	1	SMS message type	0x44 (sent or draft message) or 0x00 (received message)
56	1 or 2	SMS message length	
+0	<length>	SMS message	
+0	1	Contact information length	
+0	<length>	Contact information	Contact number or contact name (if present in address book)
+2	8	SMS received timestamp	Only applicable for received message

The maximum number of bytes that each entry can hold as a deleted SMS message is observed to be 64 bytes. This is indicated by the length information of "0x0102". Therefore, if the original message is less than or equal to 64 bytes, the entire message can be recovered. Otherwise, only a partial deleted SMS is recoverable. The length information provided in the table is also of a variable length. The actual length is computed by simply dividing the provided length information by four. If the first byte is "0x01", the length information is contained in this first byte and its following byte. Otherwise, if the first byte is in the range of "0x04" to "0xFC", the length information is provided by this byte alone. In addition, no GMT setting information was provided in the index file entries for the deleted SMS messages. To retrieve the next deleted entry, a search for the next start marker is conducted and the data has to be parsed accordingly (as shown in Table 4). This procedure is repeated until no more deleted SMS start marker is found. An example of a deleted sent SMS record is shown in Figure 3 and a deleted received SMS record is shown in Figure 4.

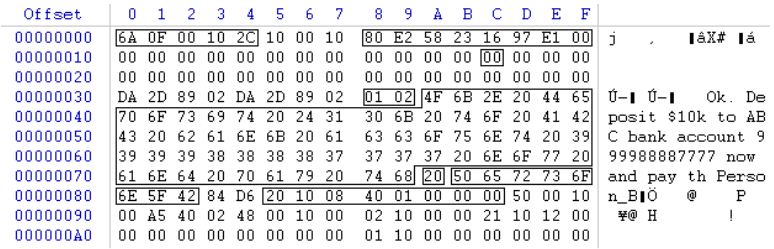
Further Memory Dump Investigation

We conducted an investigation to find out if the rest of the deleted SMS message (i.e. after the 64th byte) can be retrieved from the raw data space. We developed an internal phone memory dump tool, which accessed the Symbian TBusLocalDrive API, to perform an internal phone memory dump. The dump was performed on the internal phone memory as it was the configured location to store the SMS messages. The tool required another manufacturer-approved capability, the TCB capability, which we signed using our certificate and installed on to the Symbian phones. We then made an attempt to search the entire internal phone memory dump for the non-recoverable part of the deleted SMS message contents (i.e. >64 bytes). The contents could not be found.



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	6A	0F	00	10	2C	10	00	10	86	D6	2E	12	16	97	E1	00	j , ▮Ö. ▮á
00000010	AB	01	00	00	00	00	00	00	00	00	00	00	44	00	00	00	« D
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	DA	2D	86	02	DA	2D	84	02	FC	49	20	77	61	6E	74	20	Ú-▮ Ú-▮ üI want
00000040	41	20	74	6F	20	64	69	73	61	70	70	65	61	72	2E	20	A to disappear.
00000050	44	6F	6E	27	74	20	63	61	72	65	20	68	6F	77	20	79	Don't care how y
00000060	6F	75	20	64	6F	20	69	74	2E	20	4D	61	6B	65	20	73	ou do it. Make s
00000070	75	72	65	20	69	74	27	73	20	4F	74	68	65	72	53	49	ure it's Person_
00000080	4D	D8	72	20	10	3B	80	01	00	00	80	1B	00	00	83	43	BØr ;▮ ▮ ▮C
00000090	00	00	00	0C	00	00	00	20	00	05	01	00	00	00	35	02	5
000000A0	00	00	04	00	00	00	FD	09	00	00	06	00	00	00	90	06	ý
000000B0	00	00	1B	00	00	83	1C	00	00	80	42	00	00	03	9C	0A	▮ ▮B ▮
000000C0	00	00	A2	40	02	4E	00	10	00	04	10	00	00	01	02	01	¢@ N
000000D0	00	00	00	00	00	00	00	00	00	01	10	00	00	00	00	10	
000000E0	00																

Fig. 3. Deleted Sent SMS Message



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	6A	0F	00	10	2C	10	00	10	80	E2	58	23	16	97	E1	00	j , ▮âX# ▮á
00000010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	DA	2D	89	02	DA	2D	89	02	01	02	4F	6B	2E	20	44	65	Ú-▮ Ú-▮ Ok. De
00000040	70	6F	73	69	74	20	24	31	30	6B	20	74	6F	20	41	42	posit $10k to AB
00000050	43	20	62	61	6E	6B	20	61	63	63	6F	75	6E	74	20	39	C bank account 9
00000060	39	39	39	38	38	38	38	37	37	37	37	20	6E	6F	77	20	99988887777 now
00000070	61	6E	64	20	70	61	79	20	74	68	20	50	65	72	73	6F	and pay th Perso
00000080	6E	5F	42	84	D6	20	10	08	40	01	00	00	00	50	00	10	n_B▮Ö @ P
00000090	00	A5	40	02	48	00	10	00	02	10	00	00	21	10	12	00	¥@ H !
000000A0	00	00	00	00	00	00	00	00	01	10	00	00	00	00	00	00	

Fig. 4. Deleted Received SMS Message

4 Future Work

We plan to conduct further and more thorough investigations on the persistency of the deleted SMS messages. Our preliminary investigation results showed that the persistency of the deleted SMS messages does not depend on factors such as how long the phones have been left running or the active use of other applications. Instead, it depends on the messaging application. The deleted SMS messages were observed to be undetectable after transmission of "sufficient" multiple subsequent SMS messages (e.g. after the subsequent transmission of ten active SMS messages, we observed that one of the deleted SMS messages became undetectable). Another planned future work is the reconstruction of other Symbian privacy-protected deleted application data such as the multimedia messaging service (MMS) messages, emails, notes, and other common data types present in smartphones.

5 Conclusions

In this paper, we identified the need for an in-depth acquisition and analysis of the privacy-protected data in the Symbian smartphones, which were running the latest OS version 9.4, S60 5th Edition, as well as the prior OS version 9.3, S60 3rd

Edition. We obtained the necessary Symbian capability to access the privacy-protected data without compromising the phone's built-in security mechanism. This was achieved through the generation and installation of our customised high-level privilege certificate onto the phone through our developed mapdrive exploit.

In addition, we performed reverse-engineering on the acquired data to derive the encoding and format of the data so as to reconstruct both active and deleted SMS messages. Through our research, we also discovered the presence of a special scenario (i.e. PAIF message) and proposed the approach to handle it correctly when designing and developing our recovery tool. Our experiments on various sent, received, draft and deleted messages showed that we were able to recover both the active (in its entirety) and deleted SMS messages (up to a message length of 64 bytes) correctly and automatically.

References

1. Willassen, S.: Forensics and the GSM mobile telephone system. International Journal of Digital Evidence 2(1), 1–17 (2003)
2. Casadei, F., Savoldi, A., Gubian, P.: Forensics and SIM cards: an overview. International Journal of Digital Evidence 5(1), 1–21 (2006)
3. Kim, K., Hong, D., Chung, K., Ryou, J.-C.: Data acquisition from cell phone using logical approach. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 26 (December 2007)
4. Mokhonoana, P.M., Olivier, M.S.: Acquisition of a Symbian smart phone's content with an on-phone forensic tool. Department of Computer Science. University of Pretoria (2007)
5. Distefano, A., Me, G.: An overall assessment of mobile internal acquisition tool. In: Proceedings of the 8th Digital Forensics Research Conference (DFRWS), Digital Investigation, vol. 5(1), pp. S121–S127 (September 2008)
6. Jansen, W., Delaitre, A., Moenner, L.: Overcoming impediments to cell phone forensics. In: Proceedings of the 41st Hawaii International Conference on System Sciences (2008)
7. Hoog, A.: Android forensics, Presented at Mobile Forensics World 2009 (May 2009)
8. Dekart, Sim manager (February 2012), `http://www.dekart.com`
9. Jansen, W., Ayers, R.: Forensic software tools for cell phone subscriber identity modules. In: Conference on Digital Forensics, Association of Digital Forensics, Security, and Law (ADFSL) (April 2006)
10. Bhadsavle, N., Wang, J.A.: Validating tools for cell phone forensics, Southern Polytechnic State University, Technical Report CISE-CSE-08-05 (2008)
11. Williamson, B., Apeldoorn, P., Cheam, B., McDonald, M.: Forensic analysis of the contents of Nokia mobile phones. In: Proceedings of the 4th Australian Digital Forensics Conference (December 2006)
12. Ayers, R., Jansen, W., Moenner, L., Delaitre, A.: Cell phone forensic tools: An overview and analysis update, National Institute of Standards and Technology, Technical Report 7387 (March 2007)

13. Muller, B.: From 0 to 0 day on symbian - finding low level vulnerabilities on symbian smartphones. SEC Consult Vulnerability Lab Whitepaper (June 2009)
14. Nokia, Symbian signed developer certificate, http://www.developer.nokia.com/Community/Wiki/Developer_Certificate_(Symbian_Signed)
15. HelloOX2 Team, Helloox2 (April 2012), http://helloox2.com
16. Nokia, Symbian timestamp storage and manipulation (April 2012), http://library.developer.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/reference/reference-cpp/Kernel_Architecture_2/TTimeClass.html