Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 916262, 10 pages
http://dx.doi.org/10.1155/2015/916262

Research Article

AirPrint Forensics: Recovering the Contents and Metadata of Printed Documents from iOS Devices

Luis Gómez-Miralles and Joan Arnedo-Moreno

Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya, Roc Boronat Street 117, 7th Floor, 08018 Barcelona, Spain

Correspondence should be addressed to Luis Gómez-Miralles; pope@uoc.edu

Received 10 June 2013; Accepted 22 October 2013

Academic Editor: David Taniar

Copyright © 2015 L. Gómez-Miralles and J. Arnedo-Moreno. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Since its presentation by Apple, both the iPhone and iPad devices have achieved great success and gained widespread popularity. This fact, added to the given idiosyncrasies of these new portable devices and the kind of data they may store, opens new opportunities in the field of computer forensics. In 2010, version 4 of the iOS operating system introduced AirPrint, a simple and driverless wireless printing functionality supported by hundreds of printer models from all major vendors. This paper describes the traces left in the iOS device when AirPrint is used and presents a method for recovering content and metadata of documents that have been printed.

1. Introduction

Information technologies have grown rapidly in the last decades, changing the way we live, work, and communicate. Portable devices such as smartphones and tablets have evolved from simple phones and agendas into literally full-fledged, always-online computers. In this scenario, where mobile devices become ubiquitous, privacy and cybersecurity become a great concern since such devices may contain huge amounts of sensible valuable data about us: contacts, calendar, e-mails, and photographs as well as a pile of logs: phone calls, chat, geographic positions, and so forth.

The practice of digital forensics has needed to adapt quickly to the emerging mobile technologies. We once had a homogeneous personal computer market, mainly dominated by a few different Windows versions, with minor representations of Mac OS or Unix-based systems. Now we find that the most personal devices, the ones that always accompany their users and are more prone to contain sensitive information, run software environments which simply did not exist a few years ago, namely, Android and iOS. Furthermore, because of the competitive nature of the market, with each new version of these systems, new functionalities are added in order to appeal to a greater set of users and thus become their device of choice. However, some of these new features may manage personal user data and are worth analyzing from a forensic investigation standpoint.

This paper focuses on the AirPrint feature of iOS devices (iPhone, iPad, and iPod Touch), which allows them to print wirelessly to compatible printers [1]. In a previous paper [2] we observed that printing a document through AirPrint leaves a trace in the filesystem of the iOS device in the form of a temporary file containing the printed content and with a specific metadata that allows for the identification of this precise kind of files in the filesystem. This paper extends our previous research to analyze if these temporary files can be recovered even in modern iOS versions which use hardware-based data encryption. Considering the rise of mobile devices and applications in general [3, 4] and the hundreds of millions iOS devices in particular [5], any available process which allows to recover user data becomes especially relevant from both a computer forensics and a privacy concern standpoint.

The main contribution of this paper is the exposition of a method to recover from an iOS device the contents and metadata of documents printed through AirPrint, even in modern devices which feature hardware-based data encryption. Analyzing the behavior of AirPrint posed an interesting challenge since iOS is a closed operating system and lacks public documentation about many internal aspects. In addition, even when the mobile threat landscape has been covered by other authors [6], there seems to be no additional research on how AirPrint works behind the scenes and the forensic traces it may leave. Several authors [7, 8] have reviewed the existing (mostly commercial) forensic investigation tools for iOS devices; however, analysis of AirPrint activity does not seem to be covered by any of the existing software solutions.

This paper is structured as follows. First of all, in Section 2, AirPrint and its mode of operation both from a user's and technical standpoints are presented. The analysis of the traces left by AirPrint and the information they contain is shown in Section 3. In Section 4, basic experiments are performed to assess the recoverability of the AirPrint traces in devices where encryption has been purposely disabled. Following, in Section 5, the recoverability is evaluated for the modern devices and OS versions that feature data encryption. Finally, concluding the paper, Section 6 summarizes the paper contributions and outlines further work.

2. Description of AirPrint Network Printing

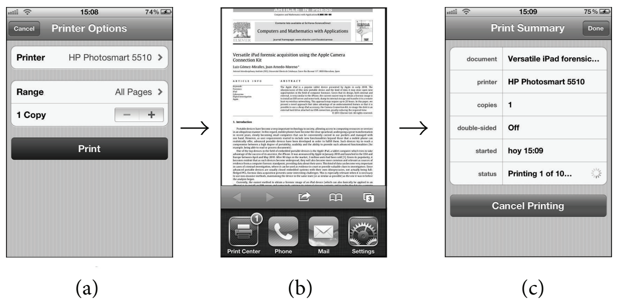
Briefly explained, AirPrint is an iOS feature that allows applications to send content to printers using the iOS device's wireless connection. Apple is directly quoted [1]: "*AirPrint automatically finds printers on local networks and can print text, photos and graphics to them wirelessly over Wi-Fi without the need to install drivers or download software*."

Apple announced AirPrint in September 2010. Two months later, iOS 4.2 was released for the iPhone, iPad, and iPod Touch, being the first iOS version to offer this feature to users. Its standard functionality allows printing only to specific, AirPrint-enabled printers. Nevertheless, as of July 2013 there were more than seven hundred AirPrint-enabled printer models in the market from sixteen different vendors [9]. Apple does not support sharing a common printer via the computer it is connected to, even when it was possible with some Mac OS X 10.6.5 beta versions; however, it can be done by using software tweaks such as AirPrintActivator [10].

Long before the introduction of AirPrint, different solutions [11, 12] tried to fill in this gap. Usually, such solutions involved iOS applications capable of opening different file formats and sending them to a desktop computer, running a companion application, which would in turn send the document to the printer itself. Some printer vendors developed specific clients; however, none of these solutions were ever widely spread among users. Currently, with AirPrint working out-of-the-box and embedded into all applications, it is hard to believe that new users will consider using a specific, usually paid application to handle printing, except maybe in some very particular environments, such as cases where the use of these kind of applications was consolidated before AirPrint was launched or some advanced capabilities are required by power users.

From a user's standpoint, Figure 1 summarizes the printing process, showing the screens it is actually possible to interact with, as seen on an iPhone.

In the client side (iOS device), AirPrint-enabled applications contain a "Print" button that, when pressed, will present an extremely simple menu (Figure 1(a)), with only two or three available options:



Printer HP Photosmart 5510
Range All Pages
1 Copy

document	Versatile iPad forensic...
printer	HP Photosmart 5510
copies	1
double-sided	Off
started	hoy 15:09
status	Printing 1 of 10...

Figure 1: Step-by-step AirPrint options screen on an iPhone.

(1) *Printer:* this option opens a list of all AirPrint-enabled printers found in the local network, showing a "name" and "description" field for each one.

(2) *Range:* (optional) defaulting to "all pages", this option opens a selector which allows the user to choose a range of pages to be printed rather than all the document.

(3) *Copies:* this option specifies the number of copies to be printed.

(4) Depending on the printer features, additional parameters such as duplex printing can be controlled.

(5) *Print:* this button proceeds to send the job to the printer.

The user cannot specify any other kind of information usually available in printing menus, such as paper size or orientation and printing quality. Everything is automatically handled by AirPrint, using some default options. When the user chooses to "Print" the job, the device shows some brief messages ("*Contacting Printer*"; "*Preparing page (...) of (...)*"; "*Sending to Printer*"). However, depending on how long the print job is, these messages may be barely visible or last for several seconds.

After the job has been sent to the printer, the printing menu disappears and the application returns to its previous state. At this point, invoking the list of recently used applications, by double-clicking the device "Home" button, reveals a *Print Center* application (Figure 1(b)). Unless the user somehow knows this application has started running in background, it may be difficult for him to find it, since no active feedback is provided during the printing process, thus being invisible at casual glance.

Opening the *Print Center* application, the user can see the list of running and pending printing jobs, check their details, and cancel them (Figure 1(c)). When the last job finishes, that is, the moment the printer ejects the last page, the application closes and does not appear anymore in the list of recently used applications. As far as it is known, there is no way to open the *Print Center* as a standalone application. It is only executed while there are jobs being printed.

From a technical standpoint, the AirPrint service is known to use the standard IPP protocol at network level for printer management, and Bonjour/Zeroconf [13] for service discovery. A comprehensive description of the printing architecture and its underlaying API in iOS devices can be found in [14].

3. Forensic Traces Left by AirPrint

This section presents the preliminary information that must be considered before more in-depth forensic investigation may proceed. Mainly, it is important to assess whether any traces are left in the device after having printed a document using AirPrint, and if so, how they can be discovered, how they behave, and which useful information can be extracted from them. All this information was discovered through some basic experiments.

3.1. Preliminary Setup. For the experiments described in this section, the following equipment was used:

(i) iOS device #1: Apple iPhone 4, 16 GB (model `A1332`) running iOS 4.3.3 (`8J2`) and 5.0 (`9A334`), both jailbroken using the `redsn0w` software [15].

(ii) iOS device #2: Apple iPhone 3G, 8 GB (model `A1241`) running iOS 4.2.1 (`8C148`), jailbroken using `redsn0w` and enabling multitasking and AirPrint.

(iii) Desktop computer information: Apple MacBook Pro (model `MacBookPro5,1`) running Mac OS X 10.7.2 (`11C74`).

(iv) Printer: HP Photosmart 5510 B111a (model `CQ176B`) running firmware version `EPL2CN1122AR`.

(v) Wireless connectivity: all devices were connected to a wireless 802.11g network to perform the experiments.

(vi) Physical connectivity: all devices were using physical wires for AC power only.

3.2. The "Jailbreak" Process. Given that iOS enforces the device to run only code signed by Apple (downloaded from the App Store), during our experiments we used the "jailbreak" technique to bypass that restriction in order to have full access to the devices and be able to run shell commands on them. The jailbreak process is exempted from prosecution under the anticircumvention section of the U.S. Digital Millenium Copyright Act [16], and it has been very useful for forensic research in the past [17, 18].

Both iOS devices were jailbroken in order to gain full access and install an SSH server and basic UNIX tools.

3.3. Traces Found. Once we were able to execute code in the device, we invoked a series of commands before, during, and after printing, and we compared the results looking for remarkable differences. The most relevant commands used were:

(1) `find / -type (b,c,d,f,l,p,s)` for listing, respectively, all the block special devices, character special devices, directories, regular files, symbolic links, FIFOs, and sockets, in the filesystem.

(2) `netstat -an -f inet` for listing any current network connections. This would show active client-server activity as well as inactive servers awaiting for incoming petitions.

(3) `ps aux` for getting information about running processes.

By reviewing the lists of files and directories generated with the `find` commands explained above, it was observed that when a device prints via AirPrint for the first time the following folder is created:

```
/var/mobile/Library/com.apple.printd/
```

We observed that everytime a document is sent to a printer, a new file named `1.pdf` is created under this folder. With additional tests, it was observed that this PDF file exists in disk only while the document is being printed. The moment the printer ejects the last page and considers the job finished, the PDF file is deleted. This is also the moment at which the *Print Center* application disappears from the list of recently used applications.

By printing some documents and copying the resulting temporary PDF files to another location before their deletion and then examining them we observed that these files are regular PDF files with the same content that is being sent to the printer (no matter whether it was originally in PDF format or not). Hence, an obvious trace is being left in the filesystem, and it reflects exactly what was printed.

It must be noted that, in some of the preliminary tests, before the physical printer was available, we set up a virtual PDF printer in a Mac computer and shared, making it look like an AirPrint printer. It worked as expected and it was possible to print to it from an iPhone; however, the printing of the document (in this case, the generation of a file in the hard drive of the Mac) was much shorter than the actual printing of a page through a real printer with real ink and paper. We observed that this greatly reduces the chances of the temporary files being flushed to physical storage from the buffer cache in the iOS device before deletion and thus their chances of recoverability. Therefore, to obtain accurate results, the experiments need to be performed with a real printer.

3.4. Properties of the AirPrint Temporary Files. From the execution of the different tests we extracted the following conclusions. Unless otherwise specified, every finding applies to all available iOS versions with AirPrint support (versions 4.2 through 6.1 and possibly later versions as well).

(1) For every print job sent via AirPrint, a file with the name of `1.pdf` is created in the directory

```
/var/mobile/Library/com.apple.printd/
```

(2) This file is in PDF format, containing the document sent to the printer. This observed behavior is consistent across internal iOS applications (Mail, Safari) as well as third party ones (GoodReader, Papers, Keynote, . . .). The only exception found is the iOS Photos app, which seems not to generate

any temporary files on disk when printing, thus not leaving these traces.

(3) The file `1.pdf` is deleted as soon as the printing job finishes. The timing observed indicates that this happens not just after finishing the task of submitting the job to the physical printer, but after the document has been completely printed.

(4) When one job is being printed, subsequent jobs arriving to the queue generate files named `2.pdf`, `3.pdf`, and so forth. The behavior observed suggests that in iOS 4 the counter resets as soon as the queue is empty (if a new job arrives later it will be named `1.pdf` again), whereas starting from iOS version 5 the counter seems to keep increasing (each new job gets a higher number even if the queue is empty) until the device reboots.

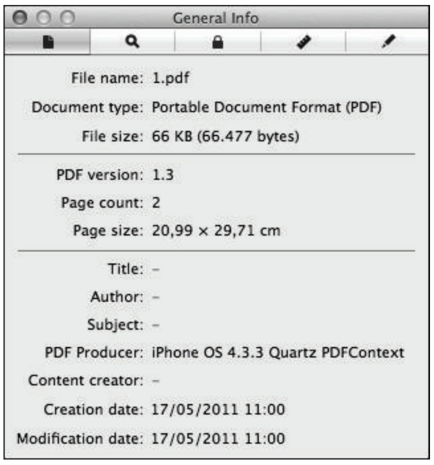
(5) When a job asks for more than one copy of the same document, the temporary PDF file contains only one copy of it. The information on the number of copies to be printed is being sent to the printer in a separate channel (standard PS commands or similar).

(6) When a page range is specified, the temporary PDF file contains only this page range. There is an exception when some applications print files that are themselves PDFs, which is studied later in Section 3.5.

3.5. PDF Metadata of the Temporary Files. Using a standard PDF reader, the metadata contents inside different temporary files generated by AirPrint were extracted and compared. The use of document metadata in forensic investigations has proven useful in different scenarios before [19–22]. A good analysis of the PDF format itself from a forensics point of view, considering its security and privacy aspects, can be found in [23].

Generally, all the temporary PDF files created by AirPrint can be identified as such by their metadata: they all show the same "PDF producer" entry ("*iPhone OS x.y.z Quartz PDFContext*," with *x.y.z* being the iOS version number), and the creation and modification dates both indicate the date and time when the document was sent to the printer (see Figure 2). Therefore, knowing what has been printed from a device looks as simple as recovering deleted PDF files from it and focusing on those with the strings "*iPhone OS x.y.z Quartz PDFContext*." Moreover, the creation and modification dates contained inside those PDF files in the form of PDF metadata will indicate precisely the printing date and time.

Only one exception was found to this behavior. When printing PDF files, that is, when the content to be printed is itself in PDF format, some applications behave like described earlier, but others (including iOS built-in applications such as Safari or Mail) actually copy the original PDF file to the `com.apple.printd` directory as `1.pdf` instead of generating a new PDF file with fresh metadata. In these cases, PDF metadata cannot be used to tell whether one given file is a trace from AirPrint printing: the only peculiar thing about that PDF file is the fact that it resides under such directory.



File name:	1.pdf
Document type:	Portable Document Format (PDF)
File size:	66 KB (66.477 bytes)
PDF version:	1.3
Page count:	2
Page size:	20,99 × 29,71 cm
Title:	–
Author:	–
Subject:	–
PDF Producer:	iPhone OS 4.3.3 Quartz PDFContext
Content creator:	–
Creation date:	17/05/2011 11:00
Modification date:	17/05/2011 11:00

FIGURE 2: Metadata of one temporary PDF file generated by AirPrint, as shown by the OS X "Preview" tool.

Considering this, an eventual automated tool aimed at recovering the traces left by AirPrint should go further than just carving for PDF files: it should correctly interpret the internals of the HFSX filesystem and tell whether each recovered file existed within the `com.apple.printd/` directory or somewhere else.

4. Recoverability of AirPrint Traces without iOS Data Protection

Given that the temporary PDF files generated by AirPrint only exist in the filesystem for a limited time and are deleted when the document has been printed, the possibility that, depending on disk scheduling and other factors, these files might never be actually flushed to the physical disk must be considered. In that case, they are unrecoverable by a subsequent forensic analysis. This section studies such possibility and assesses the probability that a given trace may be actually recovered at a later time.

4.1. Preliminary Setup. For the experiments described in this section, the following equipment was used:

(i) iOS device: Apple iPhone 3G, 8 GB (model `A1241`) running iOS 4.2.1 (`8C148`), jailbroken using `redsn0w` and enabling multitasking and AirPrint.

(ii) Desktop computer information: Apple MacBook Pro (model `MacBookPro5,1`) running Mac OS X 10.7.2 (`11C74`).

(iii) Printer: HP Photosmart 5510 B111a (model `CQ176B`) running firmware version `EPL2CN1122AR`.

(iv) Wireless connectivity: all devices were connected to a wireless 802.11g network to perform the experiments.

(v) Physical connectivity: all devices were using physical wires for AC power only.

By using older equipment (an iPhone 3G) in this set of experiments, we had a device without iOS "data protection" mechanisms (hardware-based encryption), which allowed us to analyze the behavior of AirPrint without having to avoid the added pitfall of encryption.

The AirPrint feature depends on the multitasking capabilities of the device, which are disabled by default in older models (such as an iPhone 3G). However, it is possible to enable those features during the jailbreak process using the `redsn0w` tool. Having AirPrint capabilities and an unencrypted filesystem made this device the perfect testbed for our experiments.

4.2. Experiments. There are several factors that increase the chances of the temporary PDF files being flushed to disk, making them potentially recoverable in the future. Some of these factors are listed as follows:

(i) Documents that take a long time to be printed because they have many pages or because they contain graphics. Note that, when sending a document to the printer, the user can control very few options: printer, page range, number of copies, and nothing else. There is neither an option for printing in "draft mode" nor one for using only the black cartridge, meaning that everything sent via AirPrint is printed in full color, good quality. . . and may require quite a lot of time to finish.

(ii) The quality of the wireless link between the iOS device and the printer. It can also affect the time needed to transmit and print the document.

(iii) Documents that are sent while there are other printing tasks running.

(iv) Periods of printing interruptions due to the need of human interaction, such as the printer running out of paper or out of ink.

In order to test the recoverability of AirPrint traces in the form of deleted temporary files, a series of experiments was run. The goals of these experiments were twofold. The first goal is to determine whether the temporary files generated by AirPrint are actually written to the physical disk at some point and thus may be recoverable after deletion. The second goal is to asses whether, even if the previous case is true, each of those temporary files would still be recoverable after generating more of them (i.e., what are the chances that the AirPrint temporary files overlap each other in disk, always overwriting the same space and thus making each other unrecoverable?).

Some steps in the testing process involved printing documents, while others were just aimed at simulating some casual user activity in the device (mail browsing, software update, and reboot). For those tests that involve printing, 10 documents of a given kind are printed in each test. Five of the tests involve printing, which means that, after completing all the tests, 50 documents had been printed.

After each of the tests, a dump of the iPhone filesystem was obtained. For those tests that involved printing, the dump process was not started until the printer had finished printing all of the submitted jobs sent.

The test was performed as follows:

(1) Various sets of 10 items each were printed using different applications (Safari, Photos, Mail, GoodReader, . . .).

(2) A dump of the device storage was obtained after each set of 10 items.

(3) Additional batches of activity were performed in the device (download email, install a software update, and reboot the device) and additional storage dumps were obtained after performing each of these tests.

We transferred the disk image on-the-fly via Wi-Fi to the desktop computer using the method proposed by Zdziarski [18], based on the `dd` and `netcat` commands.

Each filesystem dump was analyzed in order to determine whether the temporary files generated by AirPrint at each stage were recoverable both immediately after their generation and at later stages.

With no data encryption in place, we were able to use the *file carving* [24] method for recovering deleted files. Carving is a data recovery method consisting of going through a raw data stream (a filesystem dump in this case) looking for possible "headers" and "footers" (beginnings and ends) of known, chosen file types, such as JPEG pictures, MPEG video files, and PDF documents. The more strict a file type specification is, the easier it is for the carving tool to identify and recover that file type. In addition, some tools perform sanity checks such as establishing a file size limit or checking each recovered item against its format specification to determine whether it may be corrupt.

One of the benefits of the carving technique is that it can be used, with more or less success, over any kind of data, be it a known or unknown filesystem, a portion of it, or something completely different, such as network captures or RAM memory dumps. Note, however, that many tools implement specific strategies for common filesystems in order to improve overall success rate and extract items only from the unallocated space, skipping the disk space used by normal existing files. This is something very useful when the user just wants to focus on recovering deleted files.

This technique was applied by means of the open source, widely used `photorec` tool [25], which has a long track of usefulness in this kind of scenarios [24, 26] even on mobile devices [27].

Every dump obtained during the tests was carved for PDF files using the default `photorec` parameters. Fine-tuning these parameters would have probably improved the recovery success rate; however, after the tests were completed, it was found unnecessary given that the results achieved were indeed positive.

4.3. Results. The results of trace extraction using the photorec tool were analyzed from two different standpoints:

(1) *Recoverability.* Tenths of the temporary PDF files generated by AirPrint were successfully recovered from each filesystem dump, which confirms that those files are indeed flushed to disk.

(2) *Persistence.* The artifacts created during our first tests were still recoverable after the last tests. This suggests that, probably due to iOS file allocation strategies, the temporary files generated by AirPrint are not likely to overwrite each other.

The results of these tests, which can be seen in more detail in [2], show that under iOS 4 and with no data encryption in place the temporary files generated by AirPrint are indeed written to disk and are potentially recoverable even after rebooting the device or turning it off.

Considering that the artifacts are stored in unallocated space, it is unavoidable that using the device for long periods of time reduces the chance of recovering such artifacts, as new data stored in the device may overwrite that disk space. However, the tests show that the probability does not decrease very quickly, and there is at least a good chance to recover most of the traces.

5. Impact of iOS Data Protection on the Recoverability of AirPrint Traces

In this section we describe a new set of experiments aimed at assessing whether the traces left by AirPrint in the device's filesystem can be recovered even when modern iOS data encryption mechanisms are in place.

5.1. Preliminary Setup. The experiments described in this section were carried out using the equipment described below:

(i) iOS device: Apple iPhone 3GS, 32 GB (model `A1303`) running iOS 6.1 (10B141), jailbroken using the `evasi0n` software [28].

(ii) Desktop computer: Apple iMac (model `iMac13,2`) running OS X 10.8.4 (`12E55`).

(iii) Printer: HP Photosmart 5510 B111a (model `CQ176B`) running firmware version: `EPL2CN1122AR`.

(iv) Wireless connectivity: all three devices were connected to a wireless 802.11g network to perform the experiments.

(v) Physical connectivity: the printer and desktop computer used physical wires for AC power. In addition, the iOS device was connected most of the time to the desktop computer using the standard Apple USB to 30-pin cable; this powered the device and served as the transmission channel when dumping the device internal storage to the desktop computer.

5.2. Mechanisms to Bypass iOS Data Protection. As we introduced in previous sections, all current iOS devices offer hardware-based encryption, backed with software support at the OS and application level. As noted by Casey et al. [29], "the increasing use of full disk encryption can significantly hamper digital investigations, potentially preventing access to all digital evidence in a case."

The data encryption mechanisms that Apple calls "data protection" were introduced in iOS version 4 (June 2010) and stood publicly unbreakable for nearly one year until, in May 2011, a software firm announced a product capable of bypassing this encryption [30]. This product is restricted to "*established law enforcement, intelligence and forensic organizations as well as select government agencies*."

At the same time, Bedrune and Sigwald published [31] details about iOS data protection shortly after they released the tools and source code capable of breaking this encryption [32]. Even if the device is locked with a passcode, the tools include a bruteforce script that runs in the iOS device itself and obtains the user-defined passcode, unless it is an alphanumeric code, something rarely seen in these devices, although supported. As of July 2013, their toolkit has been updated and works successfully with supported devices even under iOS version 6.1.

Nowadays most forensic tools support a similar functionality to one extent or another.

5.3. How iOS Data Protection Affects the Recovery of Deleted Files. One of the features of iOS encryption is that it relies on per-file encryption keys, which means that each file in the filesystem is encrypted using a different key. This, in turn, means that recovering a deleted file is more difficult than just retrieving the portion of disk space where the contents of this file reside: one must also recover the necessary filesystem metadata containing the encryption key for that given file.

For this reason, commercial tools, even when able to defeat encryption, do not recover deleted files so far. Instead, these tools usually opt for (a) examining the device backups stored in iTunes in a local computer rather than device itself or (b) just query the device for as much information as possible using standard APIs; for instance, get every sent and received SMS from the Messages application, list recent lookups from the Maps application, or query the Phone application for the recent calls log. Both these approaches, however, overlook any deleted data in the device, skipping a lot of information that could be relevant to the forensic investigator.

Given that the file contents are encrypted, the carving technique cannot be used to look for files (allocated or not). However, a smarter tool looking for deleted file/directory entries in the HFSX filesystem should succeed at recovering these files, even when their contents are encrypted.

Bedrune and Sigwald's `ios_examiner` tool [32] recently incorporated an `undelete` function which applies a novel technique [33] based on using the additional data stored in the filesystem's transaction journal in order to improve the recovery results. This tool is still very recent and under improvement, but it is expected that commercial forensic application developers may include it in later versions of their products or, at least, use similar techniques to allow forensic tools to analyze an encrypted filesystem.

5.4. Adapting the `Iphone-Dataprotection` *Toolkit.* As we started new experiments, we observed that it was certainly possible to recover deleted files from our test devices using the `iphone-dataprotection` toolkit. However, only certain file types were recovered (JPG pictures, SQLite databases, XML files, . . .), whereas no PDF files were recovered at all.

```
magics = ["SQLite", "bplist", "<?xml", "\xFF\xD8\xFF", "\xCE\xFA\xED\xFE",
          "\x89PNG", "\x00\x00\x00\x1CftypM4A", "\x00\x00\x00\x14ftypqt", "\x25PDF-"]
```

Algorithm 1

```
knownExtensions = (".m4a", ".plist", ".sqlite", ".sqlitedb", ".jpeg", ".jpg",
                   ".png", ".db", ".json", ".xml", ".sql", ".pdf")
```

Algorithm 2

There are two modifications that must be applied to the software to have it recover PDF files.

The undelete algorithm used by the tool considers that a file is correctly recovered only if the initial bytes of the file match a given set of patterns. The stock list includes a limited set: SQLite databases, XML files, binary property lists, JPEG pictures, Mach-O executable binaries, PNG graphics, and M4A audio files. In order to have the tool recover deleted PDF files, the file signature of the PDF type must be added in `hg/python_scripts/hfs/journal.py` (lines 58-59) as shown in Algorithm 1.

In order to have these files stored in a separate directory, we declare `.pdf` as a known extension by modifying `hg/python_scripts/nand/carver.py` (line 119) as shown in Algorithm 2.

After performing this modification we observed that the `ios_examiner` tool successfully recovered (where technically possible) deleted PDF files. In fact it is even possible to acquire one or more dumps of the device's internal storage using the original (unmodified) tool, apply our described modifications afterwards, and then run the modified tool against the acquired images to recover any deleted PDF files they might contain, some of which can be traces left by the use of AirPrint, whereas others will be regular PDF files that have been deleted or reallocated in disk for whatever reason.

Given a set of recovered PDF files, we wrote a Perl script that outputs a CSV table indicating which of the files correspond indeed to contents printed through AirPrint, and if so, when were the documents printed and under which iOS version as shown in Algorithm 3.

5.5. Experiments. In this series of experiments we wanted to verify whether AirPrint traces were recoverable in scenarios where iOS data protection is enabled and analyze how the amount of free disk space affects the recoverability rate.

A detailed description of the whole testing process follows:

(1) Fill the device with some applications (GoodReader, plus Apple's Podcasts, iBooks, iTunes U, Find My Friends, and Find My iPhone) and multimedia content. Setup an iCloud account for activating the Find My iPhone service and start syncing email, contacts, calendars, reminders, Safari tabs, notes, photos, documents, and data.

(2) After a period of 24 hours for any massive syncing activity to take place, the device storage as reported in "Settings" is 4.1 GB available of a total of 28.3 GB (i.e., 15% free space).

(3) Reboot the device to start from a clean state.

(4) Use the GoodReader application to print a fixed set of 20 documents amounting to a total of 109 paper pages; send each document to the printer only when the previous one has been completely printed.

(5) Turn the device off and acquire forensic image #1.

(6) Boot the device. Remove all optional Apple applications as well as multimedia content (audio, video) and iCloud accounts added in step (1).

(7) After a period of 24 hours for any deletion activity to take place, the device storage as reported in "Settings" is 27.2 GB available of a total of 28.3 GB (i.e., 96% free space).

(8) Reboot the device to start from a clean state.

(9) Repeat step (4).

(10) Turn the device off and acquire forensic image #2.

(11) Recover AirPrint traces from both images and compare the results.

5.6. Results. Table 1 shows the results for each individual file. In each case, we were able to recover between 5% and 10% of the documents printed through AirPrint, extracting the following conclusions:

(i) It is possible to recover the full content of documents printed through AirPrint as well as relevant metadata such as print date and iOS version. In some cases the recovered PDF files may be corrupt but still contain details such as iOS version used and print date.

(ii) The low recoverability rate observed (5–10% in realistic scenarios) could be due to the disk scheduling algorithm used in the iOS operating system (in this particular version at least). This would also explain the fact that the success rate keeps constant regardless of the amount of free disk space.

(iii) At any particular iOS version (existing or future), a change in the disk scheduling subsystem could boost the success rate significantly. Additional work

```
#!/usr/bin/perl
print "Filename,iOS version,Print date\n";
while( $file = shift(@ARGV) ) {
  $ios = ”;
  $date = ”;
  $pdfinfo = ‘pdfinfo $file 2>&1’;
  @metadata = split( /\n/, $pdfinfo );
  if( @metadata[0] =~ m/iPhone OS.* Quartz PDFContext/ ) {
    ($ios = @metadata[0]) =~ s/.*iPhone OS ([0-9.]+) Quartz PDFContext/iOS \1/;
    ($date = @metadata[1]) =~ s/CreationDate: //;
    print "$file,$ios,$date\n";
  } else { print "$file,does not look like an AirPrint temporary file.\n"; }
}
```

Algorithm 3

Table 1: Recoverability of AirPrint temporary artifacts under iOS 6.

#	File	Size (bytes)	Size (pages)	Recovered?
1	000001.DOC	40.960	2	In image #1
2	000002.DOC	57.856	2	In image #2 and partially in #1
3	000003.DOC	55.808	2	No
4	000004.DOC	175.616	4	No
5	000005.DOC	180.736	5	No
6	000006.DOC	67.584	5	No
7	000007.DOC	179.200	30	No
8	000008.PPT	302.592	12	No
9	000009.PDF	39.586	4	No
10	000010.PDF	120.441	4	No
11	000011.PDF	31.367	1	No
12	000012.PDF	22.857	6	No
13	000013.PDF	38.638	2	No
14	000015.PDF	55.964	2	No
15	000016.PDF	150.586	4	No
16	000018.PDF	94.424	9	No
17	000019.PDF	124.152	3	No
18	000020.PDF	4.755	2	No
19	000021.PDF	4.521	2	No
20	000022.PDF	21.235	8	No

is needed to assess whether the results observed are kept consistent across different device models and iOS versions.

The traces of the printing activity from step (9) should be more easily recoverable given that in step (7) we tried to improve the recoverability rate by freeing most disk space (to reduce the probability that some new file, log entry, etc. could overwrite the AirPrint traces once they've been deleted) and by reducing most of the device's background activity (iCloud syncing, e-mail activity, . . .). Hence, the second image should contain a higher number of AirPrint traces than the first one.

In contrast to what could be reasonably expected, we observed that in each case we recovered only one or two AirPrint temporary files out of the 20 possible. It could be thought that only the latest jobs are being recovered; however the traces we found corresponded to the first jobs sent to the printer rather than the last ones.

We performed additional experiments introducing circumstances such as print interruptions due to lack of paper and loss of network link between the device and the printer. In such circumstances, the temporary files remain much longer in the iOS device's filesystem and will persist for an undetermined amount of time (even some time after rebooting the device). Under these conditions we saw the success rate increase to 15%.

6. Conclusions and Future Work

This paper analyzes the forensic traces left by usage of the AirPrint functionality in iOS based devices. We have developed a method which leverages publicly available tools to recover from an iPhone or iPad the contents of documents that have been printed using the standard AirPrint feature. The recovery of these artifacts can be valuable from the point of view of a forensic investigation in scenarios such as information leak or distribution of inadequate content; however it could also pose a privacy risk to the user community.

The traces described could persist even after the original file has been deleted, or if the original file resides inside some "vault-type" application which protects its contents on disk with an additional layer of encryption.

With modern iOS 6 data encryption mechanisms in place, the described method still succeeded in recovering between 5 and 15% of the documents printed through AirPrint. We believe the success rate can depend on factors such as the disk scheduling strategy, and thus different iOS versions could throw different results.

Considering the use case of AirPrint in domestic scenarios, probably home users will not be particularly concerned about this finding, a possible exception to this being explicit graphic content. In this aspect it is interesting to note that Apple's stock Photos application specifically did not generate, in our experiments, the temporary files described in this paper, whereas other 3rd party applications offering added security to store this kind of information are likely to generate the standard AirPrint traces when printing documents. As a general solution, in order to limit the possibility of recovering data from the filesystem, some techniques aimed at performing a secure deletion could be adopted, such as the one presented in [34] for the Android OS.

Further work must be carried out to assess whether it is possible to capture the network traffic generated while printing using techniques similar to [35] and recover the contents of the documents being printed. It would also be interesting to extend the research presented in this paper across a wider range of devices, iOS versions, and 3rd-party applications and to examine if similar issues affect other printing solutions for iOS devices and other mobile devices.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was partly funded by the Spanish Government through projects TIN2011-27076-C03-02 "CO-PRIVACY" and SMARTGLACIS (TIN2014-57364-C2-2-R).

References

[1] Apple, *Apple's AirPrint Wireless Printing for iPad, iPhone & iPod touch Coming to Users in November, 2010*, Apple, 2010, http://www.apple.com/uk/pr/library/2010/09/15airprint.html.

[2] L. Gomez-Miralles and J. Arnedo-Moreno, "Analysis of the forensic traces left by airprint in apple iOS devices," in *Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA '13)*, pp. 703–708, IEEE, Barcelona, Spain, March 2013.

[3] O. Bohl, S. Manouchehri, and U. Winand, "Mobile information systems for the private everyday life," *Mobile Information Systems*, vol. 3, no. 3-4, pp. 135–152, 2007.

[4] S. Caballé, F. Xhafa, and L. Barolli, "Using mobile devices to support online collaborative learning," *Mobile Information Systems*, vol. 6, no. 1, pp. 27–47, 2010.

[5] T. Cook, Apple WorldWide Developers Conference keynote, 2013, http://www.apple.com/apple-events/june-2013/.

[6] A. Castiglione, R. de Prisco, and A. de Santis, "Do you trust your phone?" in *E-Commerce and Web Technologies*, vol. 5692 of *Lecture Notes in Computer Science*, pp. 50–61, Springer, Berlin, Germany, 2009.

[7] A. Hay, D. Krill, B. Kuhar, and G. Peterson, "Evaluating digital forensic options for the apple iPad," in *Advances in Digital Forensics VII*, vol. 361 of *IFIP Advances in Information and Communication Technology*, pp. 257–273, Springer, Boston, Mass, USA, 2011.

[8] A. Levinson, B. Stackpole, and D. Johnson, "Third party application forensics on Apple mobile devices," in *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS '11)*, pp. 1–9, January 2011.

[9] Apple, *iOS: AirPrint 101*, Apple, 2011, http://support.apple.com/kb/ht4356.

[10] Netputing, *AirPrint Activator*, 2011, http://netputing.com/airprintactivator.

[11] Avatron Software Inc, *Print Sharing*, 2011, http://avatron.com/apps/print-sharing/.

[12] EuroSmartz, PrintCentral for iPad, iPhone or iPod Touch, 2012, http://mobile.eurosmartz.com/products/printcentral.html.

[13] D. Steinberg and S. Cheshire, *Zero Configuration Networking: The Definitive Guide*, O'Reilly, 2005.

[14] Apple Inc, How Printing Works in iOS, 2011, https://developer.apple.com/library/ios/.

[15] iPhone Dev Team, redsn0w, 2011, http://blog.iphone-dev.org/post/5239805497/tic-tac-toe/.

[16] US Copyright Office, *Rulemaking on Exemptions from Prohibition on Circumvention of Technological Measures that Control Access to Copyrighted Works*, US Copyright Office, 2010, http://www.copyright.gov/1201/2010/.

[17] J. R. Rabaiotti and C. J. Hargreaves, "Using a software exploit to image RAM on an embedded system," *Digital Investigation*, vol. 6, no. 3-4, pp. 95–103, 2010.

[18] J. Zdziarski, *iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets*, O'Reilly Media, 2008.

[19] F. Buchholz and E. Spafford, "On the role of file system metadata in digital forensics," *Digital Investigation*, vol. 1, no. 4, pp. 298–309, 2004.

[20] A. Castiglione, A. De Santis, and C. Soriente, "Taking advantages of a disadvantage: digital forensics and steganography using document metadata," *Journal of Systems and Software*, vol. 80, no. 5, pp. 750–764, 2007.

[21] A. J. Clark, "Document metadata, tracking and tracing," *Network Security*, vol. 2007, no. 7, pp. 4–7, 2007.

[22] M. S. Olivier, "On metadata context in database forensics," *Digital Investigation*, vol. 5, no. 3-4, pp. 115–123, 2009.

[23] A. Castiglione, A. De Santis, and C. Soriente, "Security and privacy issues in the portable document format," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1813–1822, 2010.

[24] M. I. Cohen, "Advanced carving techniques," *Digital Investigation*, vol. 4, no. 3-4, pp. 119–128, 2007.

[25] CGSecurity, *PhotoRec, Digital Picture Recovery*, 2009, http://www.cgsecurity.org/wiki/PhotoRec.

[26] S. L. Garfinkel, "Forensic feature extraction and cross-drive analysis," *Digital Investigation*, vol. 3, pp. 71–81, 2006.

[27] I. Pooters, "Full user data acquisition from Symbian smart phones," *Digital Investigation*, vol. 6, no. 3-4, pp. 125–135, 2010.

[28] Y. D. Wang, N. Bassen et al., evasi0n, 2013, http://evasi0n.com/.

[29] E. Casey, G. Fellows, M. Geiger, and G. Stellatos, "The growing impact of full disk encryption on digital forensics," *Digital Investigation*, vol. 8, no. 2, pp. 129–134, 2011.

[30] ElcomSoft, ElcomSoft investigates iPhone hardware encryption, provides enhanced forensic access to protected, 2011, http://www.elcomsoft.com/PR/eppb110524en.pdf.

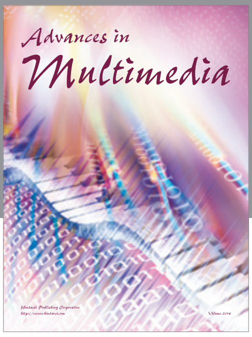
[31] J. B. Bedrune and J. Sigwald, *iPhone Data Protection in Depth*, HITB, Amsterdam, The Netherlands, 2011.

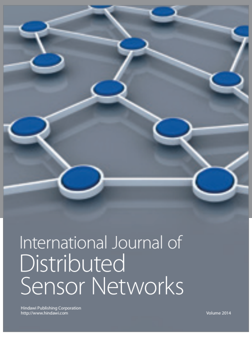
[32] J. B. Bedrune and J. Sigwald, iPhone data protection tools, 2011, http://code.google.com/p/iphone-dataprotection/.

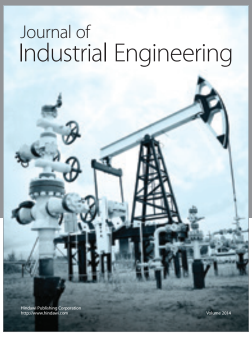
[33] A. Burghardt and A. J. Feldman, "Using the HFS+ journal for deleted file recovery," *Digital Investigation*, vol. 5, supplement, pp. S76–S82, 2008.

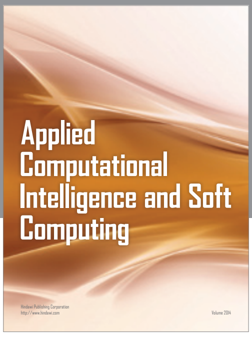
[34] A. Castiglione, G. Cattaneo, G. De Maio, and A. De Santis, "Automatic, selective and secure deletion of digital evidence," in *Proceedings of the 6th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA '11)*, pp. 392–398, October 2011.

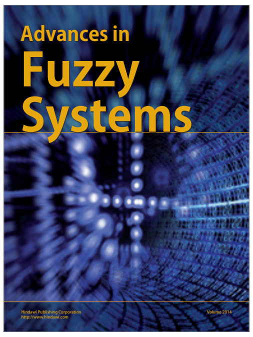
[35] A. Castiglione, G. Cattaneo, G. de Maio, and A. de Santis, "Forensically-sound methods to collect live network evidence," in *Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA '13)*, pp. 405–412, Barcelona, Spain, March 2013.

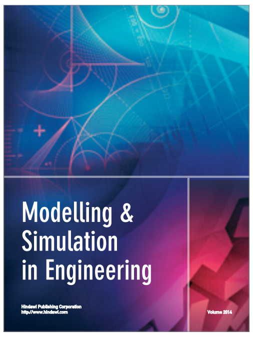












Hindawi

Submit your manuscripts at
http://www.hindawi.com