Universal, fast method for iPad forensics imaging via USB adapter

Luis Gómez-Miralles
Computer forensics and electronic evidence investigator
INCIDE - Investigacion Digital, S.L.
Valencia, Spain
pope@lgomez.es

Joan Arnedo-Moreno
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Barcelona, Spain
jarnedo@uoc.edu

Abstract

The Apple iPad is a popular tablet device presented by Apple in early 2010. The idiosyncracies of this new portable device and the kind of data it may store open new opportunities in the field of computer forensics. Given that its design, both internal and external, is very similar to the iPhone, the current easiest way to obtain a forensic image is to install an ssh server and some tools, dump its internal storage and transfer it to a remote host via wireless networking. This approach may require up to 20 hours. In this paper, we present a novel approach that takes advantage of an undocumented feature so it is possible to use a cheap iPad accessory, the Camera Connection Kit, to image the disk to an external hard drive attached via USB connection, greatly reducing the required time.

Keywords: forensics, iPad, cybercrime, digital investigation, Apple.

1. Introduction

Portable devices have become a very important technology in our society, allowing access to computing resources or services in an ubiquitous manner. On that regard, mobile phones have become the clear spearhead, undergoing a great transformation in the last years, slowly becoming small computers that can be conveniently carried in our pockets and managed with one hand. However, as user requirements start including new functionalities beyond those that a mobile phone can realistically offer, advanced portable devices have been developed in order to fulfill them. Such devices try to reach a compromise between a high degree of portability, usability and the ability to provide such advanced functionalities (for example, being able to read or process documents).

The latest contender in the field of embedded portable devices is the Apple iPad, a tablet computer which tries to take advantage of its ancestor's success, the iPhone. It was announced by Apple in January 2010 and launched in the U.S.A. and Europe between April and May 2010. After 80 days in the market, 3 million units had been sold [1]. Given its popularity, it becomes evident that as such devices become widespread, they will also become more common and relevant as sources of evidence from a computer forensics standpoint, providing data about their users. Such data can become very important in cases of crime investigation, where it can be used as evidence in Court or can provide valuable clues to investigators. Since advanced portable devices are usually closed embedded systems with their own idiosyncracies, not actually being fully fledged PCs, forensic data acquisition presents some interesting challenges. That is specially relevant when it is necessary to use non-invasive methods, maintaining the device in the same state (or as similar as possible) as the one it was before the analysis began.

Currently, the easiest method to obtain a forensic image of an iPad device (which can also be basically applied to an iPhone) is to install an ssh server and some tools, retrieve its internal storage contents and transfer the data to a remote host via wireless networking. This approach can take up to 20 hours. In this paper, we present a different approach which relies on a local USB connection with help of a cheap and easily available peripheral, the Camera Connection Kit. This approach greatly reduces the time needed to create a system image. Furthermore, as an additional contribution, the presented keeps a compromise in the amount data which is modified during the acquisition process.

The paper is structured as follows. Section 2 provides an overview of the iPad architecture, focusing on those characteristics specially relevant from a forensic analysis standpoint. In Section 3, a literature review of the current state of iPhone/iPad forensics is presented. The proposed forensic data acquisition method is described in Section 4. Concluding the paper, Section 5 summarizes the paper contributions and outlines further work.

2. iPad architecture overview

From the external point of view, the iPad is basically a big (24x19 cm.) iPhone with a 9.7" screen, providing a resolution of 1024x768. While its internals are very similar to those of its antecesor, the iPad's bigger form factor makes it suitable for longer periods of use, which has motivated the apparition of lots of different applications of every kind. Therefore, the iPad is able to perform tasks perviously reserved to common computers or, up to some point, netbooks.

2.1. Main features

The basic iPad internals are:

- Processor: A custom Apple *A4* ARM processor based on a single-core Cortex-A8, running at a 1 GHz.
- Volatile storage: 256 MB DRAM.
- Non-volatile storage: 16, 32 or 64 GB solid state storage drive.
- Wireless connectivity: 802.11 a/b/g/n and Bluetooth 2.1, the same as every iPhone.
- In addition, the 3G model features an A-GPS (Assisted GPS), and hardware for communicating over UMTS/HSDPA (820, 1900 and 2100 MHz) and GSM/EDGE (850, 900, 1800 and 1900 MHz.

In the process described in this paper, we will use the wireless (802.11) network, and the iPad "Dock" connector, described in the next section.

2.2. Connectors and buttons

The iPad connectors and buttons are very similar to the iPhone's. When placed over the short edge with the round button in the center, we find:

- Top left: a 3.5" jack capable of functioning simultaneously for several audio functionalities.
- Top right: "Lock" button.
- Right edge, near the top: volume and mute controls. In the previous iOS 3 branch, the mute button was a rotation lock switch instead; this function has since been moved to a 'software switch' in the device's graphical interface.
- Bottom center, frontal face: round "Home" button.
- Bottom center, in the edge (below the "Home" button): Apple standard 30-pin "Dock" connector, the same used in every iPhone and most iPods.

.

Figure 1 shows the function of each button. Note that the "Lock" button performs several functions: when the device is off, it will turn it on; when the device is on, a short press will put the device to sleep or wake it form sleep, and a long press will show a dialogue to turn it off. For clarity, in this paper we will keep referring to this button as the "Lock" button. Button configuration is important since, as will be explained in Section 4.1.1, it may be necessary to put the device in *DFU mode* ('Download Firmware Update') in order to setup the device for forensic imaging. When this is needed, installed software usually instructs the user to press a particular combination of these buttons to have the device enter DFU mode.

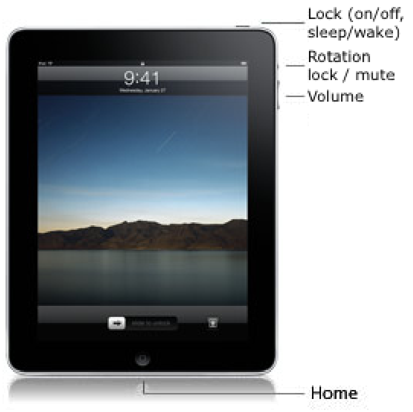


Figure 1. iPad button configuration (iOS 4 and higher).

2.3. Partition scheme

As noted by Zdziarski [2], all devices belonging to the iPhone family contain two partitions:

1) A huge *user data* partition, holding all extra applications installed as well as all the user's data.
2) A small *system* partition containing iOS and the basic applications.

From a forensics standpoint, as far as the user data partition is concerned, some iPad applications which may hold relevant data include enterprise or office software, such as QuickOffice Connect Mobile Suite [3] or Apple's iWork suite [4], [5]. They can all contain text documents or spreadsheets, which are prone to including sensitive or financial information. Although similar applications existed in the iPhone, allowing for direct document editing with no need for an external computer, the iPad's form factor will no doubt boost the existence of documents stored only within the device (and not, for instance, in the suspect's main computer), being edited here and never travelling outside the iPad (with the possible exception of device backups performed by iTunes).

Another possible source of information lies within Apple's *AirPrint* framework released in November 2010 as a feature of iOS 4.2 [6], which provides native printing capabilities to the iPhone and iPad. But long before AirPrint existed, other applications such as PrintCentral [7], already allowed the user to send most document types to a remote printer (connected to a computer with the appropriate server software). These applications' disk caches are likely to hold relevant information such as copies of printed documents.

The system partition contains the base iOS software that comes bundled inside every iOS software update (which explains why they weight hundreds of megabytes). This includes the core operating system and graphical user interface, as well as the standard set of bundled applications such as: Safari, Mail, Calendar, iPod, etc. Note that only the application binaries themselves lie within this partition, whereas the relevant data (for instance, user mail) is stored in the data partition.

3. Current work on IPad forensics

A very basic approach to acquiring user data is connecting the device via the standard USB cable to a computer running iTunes, Apple's multimedia player which is in charge of synchronizing content to the device. Using its AFC protocol (*Apple File Connect*), iTunes syncs existing information (contacts, calendar, email accounts, apps...) and can even retrieve a complete backup of the device; however this presents two problems:

1) The device needs to be correctly paired with the iTunes software in order to sync.
2) Even if the investigator has access to an iTunes backup of the device (say, found in the suspect's main computer), it will not contain unallocated space, from which deleted data can be recovered.

Consequently, more sophisticated methods are required. However, the iPad is distributed as a closed device, meaning that access to its internals is limited and only a those applications approved by Apple may be installed or executed. With these set of restrictions in place, it is extremely difficult to acquire any kind of meaningful forensic data. Fortunately, even though the iPad is a very new device, its internal architecture is very similar to the iPhone's and forensic approaches may be easily ported to the iPad.

On July 6th 2007, just one week after the iPhone was launched, George Hotz announced [8] the existence of a method to get a full, interactive shell. This was the first step towards bypassing Apple's restrictions on their devices, making it possible to execute any program and not only those approved by Apple; a process that has been named **jailbreaking**. In other mobile platforms, such as those running Google's *Android* operating system, a similar process exists which is known as *rooting*. Vendors usually dislike this technique, although in most countries it is legal or at least not definitely illegal. If you ever need to defend this in court, you can do a brief explanation of why jailbreaking the device: to get full access to the system, and thus to the information stored in it, which is crucial in criminal cases which require forensic analysis of these kind of devices.

The jailbreaking process modifies the system partition without alteration of the data partition, which means that it does not alter the user's data, a very important requisite. Even if we assume that some current or future jailbreak methods will modify the user data partition, we can still obtain plenty of useful information, as long as we know what alterations we are responsible for. Ever since their development, the jailbreak tools have been updated to support every new iPhone model and every new iOS version. This method may also be applied to an iPad and, in fact, all the two major forensic approaches in order to recover a complete image from the device are ultimately based on jailbreaking.

The main approach was proposed by Zdziarski [2], who noted that *the iPhone can communicate across several different mediums, including the serial port, 802.11 Wi-Fi, and Bluetooth. Due to the limitations of Bluetooth on the iPhone, the two preferred methods are via the serial port and Wi-Fi*. He proposed a basic method for obtaining a forensic image of the iPhone without tampering the user data partition by jailbreaking the device and using SSH access and the `dd` and `netcat` standard UNIX tools, which by that time had already been ported as a part of the growing iPhone jailbreaking community. Similar methods are explored by Rabaiotti [9] against a Microsoft Xbox. There was not, however, a known, public way to communicate with the device via its serial port, so Zdziarski had to send the forensic image via the device Wi-Fi interface, which is quite slow.

Alternate approaches are provided by some forensics software vendors [10], [11], which have developed solutions that use rather uncommon techniques to get a dump of the solid state storage drive. This is often accomplished by using exploits against more or less known bugs on specific iOS versions in order to execute arbitrary unapproved code, which is actually the same *jailbreakers* do in order to free their devices. However, these vendors do not need to install a complete set of tools in the device. Instead, they tend to upload a tiny, small-footprint software agent which ideally will take control of the system, dump the solid state storage drive through the serial port (dock connector), and will then reboot the device without copying any data to the iPad internal storage.

These methods offers some advantages over the jailbreak approach, being a more straightforward process, simpler to the investigator and leaving little or no footprint on the acquired system. However, it also has some weak points.

First and most important, any propietary method ultimately makes use of an exploit against vulnerabilities of the iOS version of the device, because this is the only way of take such control of the device bypassing every vendor restriction. With every iOS update (usually every few months, downloaded via iTunes), the forensics software must be updated, usually because bugs exploited in previous versions are fixed in the newer version; but even if an exploit still works, exploitation parameters such as memory addresses are very likely to change.

Jansen [12] identified *"the latency in coverage of newly available phone models by forensic tools"* as one of the problems for forensic specialists working with mobile devices. Jailbreaking in the iPad has been moving in a timeframe of barely 1-5 days following iOS updates. We consider very realistic that at some point in the near future, jailbreak updates will be available days or even weeks before some particular forensics software products get the same needed updates.

In addition, many of these proprietary methods are closed and lack any public documentation. Therefore, they are difficult to audit and it cannot be guaranteed that no footprint is actually left on the device. Knowing the process the device is going through, and the precise alterations that this process causes to the device, is a good practice and very important to the forensic investigator.

Therefore, even though some of the proprietary methods may be suitable for the analysis of the device under common circumstances, vendors of such products may fail to release in time an update to support newer iOS versions; they may even not release it at all if, say, the product is discontinued.

Our approach offers what appears to be the best possible throughput, and this is acomplished with a generic UNIX approach via jailbreaking, which is likely to live longer than most iPad forensics software products, thus guaranteeing that it can be applied in future iOS versions.

4. Forensic data acquisition on iPad devices

In this chapter we will describe our method for fast iPad imaging via a USB connection. The method is divided in two general phases: device setup and imaging. Each phase is also divided in several substeps which must be sequentially followed. We will not provide detailed instructions about how to jailbreak an iPad. The description will focus on our technique to recover an image of user data from an already jailbroken iPad.

4.1. Device setup

As mentioned in Section 3, before any forensic analysis may be attempted, a special device setup is required in order to bypass the access restrictions installed by the manufacturer. Once this phase is complete, low level access to the device is actually possible. In addition, it is necessary to install the extra packages needed for our proposed imaging approach.

4.1.1. Jailbreak the device. The actual way to perform the jailbreak varies depending on the iOS version installed on the device. An iPad running iOS 3.2.1 (the initial iOS version preinstalled in most iPads) can be jailbroken by just browsing to `http://www.jailbreakme.com`, a website that exploits a known vulnerability in Safari to take control of the system. The exploits themselves and related documentation can be found at [13]. For a complete, up-to-date chart about jailbreaking tools for each iOS version, refer to [14].

Many jailbreaking tools (`redsn0w`, `PwnageTool`, etc) will require the user to put the device into *DFU mode* with a combination of presses of the “Lock” and “Home” buttons. When this is needed, the software will give the user the necessary instructions.

Should we find a device with a recent iOS version for which no jailbreak procedure exists, it could be acceptable to downgrade to the latest jailbreakable version, although this should be done only as a last resort, and always documenting the steps taken. This would rarely succeed, however, because Apple does not allow to downgrade a device’s iOS version after a newer version has been available for some time. There are some workarounds for this but they are not of use in our scenario because they require that we have previously saved some crucial data before installing its present iOS version. Anyway, it is very unlikely that we hit a non-jailbreakable iOS. Take, for instance, iOS 4.2.1 (the first iOS 4 release for the iPad): it was released in November 22 2010, and the appropriate tool for jailbreaking (in that case *redsn0w*) was released the next day [15].

Once a new iOS version has been released, the first jailbreak methods will probably be *tethered*: a tethered jailbreak means that it is only effective as long as the operating system is running. The moment it is rebooted (not when the device is locked), the jailbreak is lost, meaning that two things will happen temporarily until the device is rebooted again into a tethered jailbreak state with the appropriate tool: (1) any jailbreak software installed will not work; and (2) some internal applications (for instance the Safari web browser) may not work, or in the worst case, the whole device might not work at all. We state again that this is only a temporary state, until the device is jailbroken again. It would be acceptable to use a tethered jailbreak for imaging purposes, and in fact part of the tests performed in this paper have taken place over an iPad running iOS 4.2.1, for which tethered jailbreak is the only jailbreak method available at this time.

It is important to note that *jailbreaking* a device does not mean *carrier-unlocking* it. Jailbreak is just a precondition for carrier unlocking. Our proposal needs not perform carrier unlocking, and in fact this is rarely needed in the iPad given that it is usually sold carrier-free.

4.1.2. Charge the battery. It may seem obvious, but it is necessary to have the battery charged to, at least, about 20%. This is because during the imaging process, the iPad’s dock connector will be used for USB data transfer, so it will not be possible to plug the device to a power point.

4.1.3. Run Cydia and upgrade available packages. After the device has been jailbroken, a new application labeled *Cydia* [16] will appear in the home screen. This is the software manager that allows installing software not approved by Apple.

When run for the first time, Cydia initializes the device’s filesystem and exits. In the next execution, it presents a *Who are you?* prompt, offering three choices; we must choose

'Developer (no filters)', as it offers the widest range of software. Afterwards, if there are available updates to install, it is recommended to perform a 'Complete upgrade'. The device will then restart. We re-open Cydia and, if asked for upgrades, we repeat the process.

4.1.4. Install required software packages. Once Cydia has finished upgrading itself, we use the 'Search' function in Cydia to find and install the following packages:

- *openssh*. This package contains the SSH server that we will use to access the iPad.
- *coreutils*. This package contains the `split` command, which is needed due to reasons that will be exposed later.

The most important tool for this procedure, `dd`, need not be installed, as it is contained inside the essential *coreutils-bin* package, which is installed by default as part of the jailbreaking process.

4.1.5. Network and auto-lock settings. It is necessary to connect the iPad to a wireless network. Another computer in that network will be used to access the iPad via SSH.

Communication between the computer and the iPad will be over SSH, and thus, encrypted. However, we strongly advise to use encryption in the wireless network protocol, and ideally, to use an isolated network for the computer and the iPad only. This is because there is a small window of time in which the device will be accessible with default passwords. There is at least one known worm which penetrates jailbroken iOS devices using these default credentials [17], although nowadays it is nearly impossible to find that code in the wild.

To connect to a wireless network we use the relevant section inside the 'Settings' application. If no wireless network is available, a laptop can be used to create an ad-hoc network and have the iPad join it. The blue button next to the network name reveals the IP address in use (usually acquired via DHCP) and allows the user to manually specify an IP address if needed. The IP address must be noted, as it will be needed later for accessing the iPad from the remote computer.

Still in the 'Settings' application, section 'General', the 'Auto-Lock' option must be set to 'Never'. This will prevent the device from going into sleep mode while the forensic image is being generated, which could interrupt the process. When not in use, the device should be locked (using the Lock button; see section 2) in order to save battery.

We have not tested whether the multitasking capabilities and persistent Wi-Fi in iOS 4 would allow the imaging process to take place while the device is locked. Anyway, given that imaging is a long process that can take more than hour in the biggest devices, we recommend to keep the device awake all the time.

Local access approach. We found at least two ways to apply this method without using a remote computer, although both of them introduce additional complications to the process.

On one hand, it may be possible to install *MobileTerminal* instead of *openssh*, and use the terminal application in the iPad itself to mount the hard drive and image to it. However, at the time of this writing, *MobileTerminal* does not work in iOS versions 4.x, and this software has a history of long delays before being updated to support newer iOS versions.

On the other hand, another approach is to install *openssh* and run an SSH client on the iPad itself. There are many such applications in Apple's App Store, although the fact of keeping this application running during the image generation is likely to alter data and will possibly corrupt the image. Thus, we prefer to use a remote computer and leave the iPad as untouched as possible. Remounting the partition read-only is not a possible solution in this case, as will be explained in Section 4.2.1.

4.2. Device imaging

Once the device is connected to a wireless network, another computer in that same network is used to connect to the iPad via SSH. Using this connection, it is possible to remotely issue commands to the device to initiate the imaging process.

At this point the iPad is accessible via the standard password `alpine`, which works for both the standard `mobile` user as well as for the `root` user, which has full access to the device. The correct way to proceed would be to access the iPad via SSH as the `root` user, and immediately change its password and the password of the `mobile` user account, using the `passwd` command.

4.2.1. Mounting a USB hard drive. In this step we will use Apple's Camera Connection Kit for the iPad [18] in order to access an external USB hard drive. According to Apple, *"the iPad Camera Connection Kit gives you two ways to import photos and videos from a digital camera: using your camera's USB cable or directly from an SD card"* [18]. Thus, it consists of two adapters, one of them being a SD card reader, and the other offering a USB female connector; both of these adapters can plug (one at a time) to the iPad's dock connector, placed in the base, below the "Home" button.

Initial vendor information suggested that the USB adapter only uses the PTP protocol [19] to access the images stored in a camera, and that an actual camera, with its camera-to-USB cable, should be plugged into this connector for the adapter to import the pictures. When this is done, the *Photo* application launches and allows the user to transfer photos and videos from the connected media to the iPad's internal memory.

We have found, however, that the iPad implements the *USB mass storage device class* protocol. Thus, the iPad may mount the disk inserted (regardless of whether it is a hard or solid state storage drive) looking for a `/DCIM` directory as per CIPA DCF standard [20]. If this folder exists, the Photo application will open, allowing the user to import contents; if the folder is not found, the device is unmounted and ignored. We have exploited this undocumented feature to manually mount an external USB hard drive with the appropriate parameters.

As for the filesystems supported, we have been successful in mounting FAT and HFS+ (the standard Macintosh filesystem, which is also the one used for the iPad internal storage). An important issue for Windows users is that their operating system will refuse to format a drive larger than 32 GB as FAT [21], although it can normally mount much bigger FAT partitions and work with them flawlessly. These users will need to use externals tools such as Fat32Format [22]. Mac and Linux users will have no trouble with their standard *Disk Utility* and *mkfs.msdos* tools, respectively.

When we have connected the USB external drive to the iPad (see Figure 2), we can check its presence within the SSH session by running the folowing command:

```
ls /dev/disk1
```

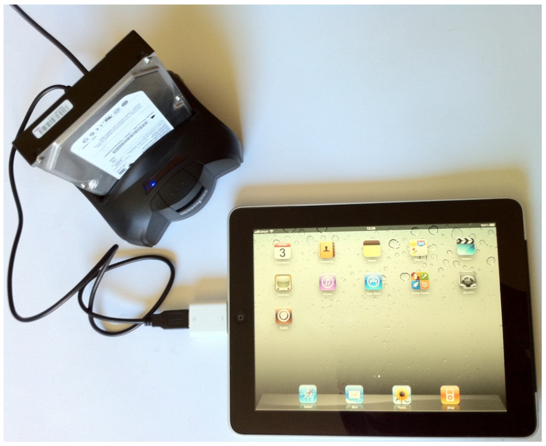


Figure 2. iPad connection to external hard drive via Camera Connection Kit.

The iPad internal storage disk is assigned the node name `/dev/disk0`, so the presence of a `/dev/disk1` implies that the newly connected hard drive has been correctly recognized. If we get an error and there is no `/dev/disk1`, the drive has not been recognized. In our tests, this was usually accompanied by a dialog in the screen complaining that "this device requires too much power", when trying to connect certain big solid state storage drives and some portable hard drives that take power from USB only. Under iOS version 4 the problem gets bigger because the USB port will no longer emit 100 mA (as it did under iOS 3.x) but only about 20 mA [23]. We found that best results were achieved using a full-size external hard-drive with its own power adapter, or connecting the drive to a powered USB hub.

This command mounts the first partition of the external drive in the `/mnt` directory of the device:

```
mount -t msdos /dev/disk1s1 /mnt
```

We were equally able to mount HFS+ partitions using the `-t hfs` parameter. Due to the Macintosh EFI support, finding the correct partition name for HFS-formatted disks can be tricky. To view the full list of available partitions, we used the command `ls /dev/disk1*`, and we tried to mount all of them until we succeeded.

Zdziarski [2] recommended immediately remounting the data partition in read-only mode (`umount -f /private/var; mount -r /private/var`) prior to beginning the actual imaging. However, in our tests, we found that in both iOS 3 and iOS 4 the system halted if the partition was unmounted; and forcing its remount with `mount -fru` was not supported either.

It must be noted that imaging a mounted partition may alter the integrity of the filesystem contained in the resulting image. In fact we found out that it is possible to end up with images that are unmountable. In order to reduce this risk, no other activity should be taking place in the iPad (neither via the touch screen, nor through the network) while imaging.

Once the disk has been mounted, the command `df -h /mnt` can be used to show its free space and confirm that the drive had been correctly recognized.

4.2.2. Obtaining the forensic image. At this point the working directory was changed to that where the external drive was mounted and the imaging process started with the command:

```
dd if=/dev/rdisk0s2 bs=32M | split -b
4000m - part-
```

The full command can be explained as follows:

- `dd` - The command `dd` is invoked,
- `if=/dev/rdisk0s2` - Taking as Input File (i.e. reading from) the device `rdisk0s2`, which corresponds to the second slice of the iPad's internal storage, containing the data partition. Due to the partition scheme used in Mac OS and iOS, it is equally acceptable to image `/dev/rdisk0s2s1`.
- `bs=32M` - Using a block size of 32 MB; actually we found that the process works, with similar throughput, for values of 1M and multiples of it.

- `| split` - Instead of writing all these data to a huge file in disk, the data is split.
- `-b 4000m` - Split file size, into smaller files of 4 GB (4000 MB) each.
- `-` This dash means the input content to be split is coming from the previous command, in this case `dd`.
- `part-` - And this is prepended to the name of the output files. The suffix will be two letters, starting with `aa`, as this is the default behavior for the `split` command.

As a result, several 4 GB chunks named `part-aa`, `part-ab`, etc. were generated. Splitting the image in smaller 4 GB files would not be necessary when imaging to a HFS-formatted (Mac) drive.

When finished, the target drive **must** be unmounted before disconnecting it from the iPad. This can be done by either turning the iPad off or unmounting the drive by exiting the `/mnt` folder and running `umount /mnt`.

4.2.3. Reconstructing the image.

In order to obtain a full image that can be processed using standard tools, all the fragments must be concatenated. This can be done with a variety of tools in different systems, but a simple command that will probably work in Mac, Linux, and Windows, would be:

```
cat part-* > ipad.dmg
```

The resulting image can then be treated by the methods described in [2] to recover data such as: emails, address book contacts, pictures and videos, Google Maps data, and so on.

As far as image reconstruction is concerned, it must be noted that, starting with iOS version 4, Apple introduced a layer of hardware encryption services [24], which, if activated in the device, will result in partial encryption of the imaged data. Altogether with this, we found a new `protect` option for the `mount` command, which is by default applied to the data partition. We have failed to find any documentation about this parameter, although interestingly enough, the string *protect* also appears inside the Mac OS X `mount` command. Nevertheless, the inclusion of this iOS version into the iPad is still very recent at the time of this writing, so we didn't have much time to experiment with it.

In this scenario, imaging the partition is possible although the resulting image may not be mountable. We think that this is probably due to a layer of encryption, which could be circumvented if the keys are retrieved from the live system after gaining SSH access. Still, carving tools such as Scalpel [25] may be able to recover certain file types.

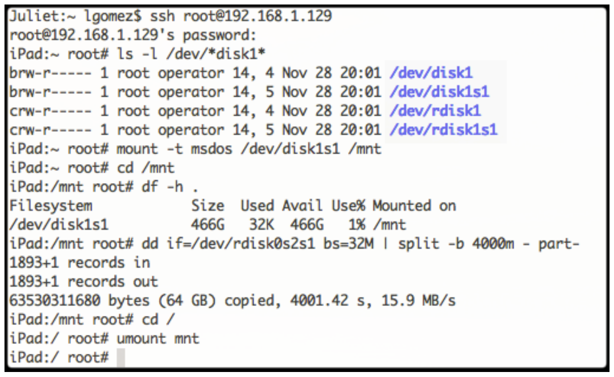
If the device is just passcode-protected, jailbreaking and accessing via SSH is equally possible. The simplest jailbreaking methods working in user-land, such as `jailbreakme.com` will not work given that we are unable to obtain initial access to the device, but other methods (*redsn0w*, *Pwnage Tool*...) could work.

4.3. Performance Results

We performed several experiments measuring the speed of our imaging process proposal via USB connection using the Camera Connection Kit. As can be seen in Figure 3, the process offers a measured throughput of 15.9 MB/s or 0.95 GB/min. This was the highest transfer rate we could achieve, always using common serial-ATA hard drives connected through standard USB-to-SATA adapters. The Figure represents the output of imaging a 64 GB iPad running iOS 4.2.1 to a Seagate ST3500418AS drive.

In comparison, we tested the Zdziarski method over an ad-hoc 802.11n network operating at its maximum theoretical rate (108 Mbps), and we obtained a throughput of barely 1 MB/s, which means that a 16 GB iPad would be imaged in 5 hours and a 64 GB one would require about 20 hours. Our USB approach results in a speed boost of 15x over traditional Wi-Fi imaging.

Forensics software vendors do not seem to release specifications about the imaging times needed by their methods; we could only find that information about Jonathan Zdziarski who states [26] about *"the latest version of the Zdziarski method, which is used in the automated tools available free to law enforcement agencies worldwide"*: *"about 15-30 minutes is all it takes, regardless of whether you're imaging a 4GB iPhone or a 32GB iPhone 3G[s]"*. Assuming he is able to image 32 GB in 'about 30 minutes', we think we have come to the same limit. This is probably the maximum transfer rate of the device's serial port, although it is hard to tell whether this is a physical limit of the port or a software matter that could be improved in future iOS versions.



```
Juliet:~ lgomez$ ssh root@192.168.1.129
root@192.168.1.129's password:
iPad:~ root# ls -l /dev/*disk1*
brw-r----- 1 root operator 14, 4 Nov 28 20:01 /dev/disk1
brw-r----- 1 root operator 14, 5 Nov 28 20:01 /dev/disk1s1
crw-r----- 1 root operator 14, 4 Nov 28 20:01 /dev/rdisk1
crw-r----- 1 root operator 14, 5 Nov 28 20:01 /dev/rdisk1s1
iPad:~ root# mount -t msdos /dev/disk1s1 /mnt
iPad:~ root# cd /mnt
iPad:/mnt root# df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/disk1s1          466G   32K  466G   1% /mnt
iPad:/mnt root# dd if=/dev/rdisk0s2s1 bs=32M | split -b 4000m - part-
1893+1 records in
1893+1 records out
63530311680 bytes (64 GB) copied, 4001.42 s, 15.9 MB/s
iPad:/mnt root# cd /
iPad:/ root# umount mnt
iPad:/ root#
```

Figure 3. Throughput of system imaging a 64 GB iPad.

5. Conclusions and Future Work

In this paper, we have presented a novel approach that takes advantage of a hidden feature in the iPad's USB adapter so it is possible to use a cheap, universally available $30 accessory to image the device directly to a USB drive attached to it. The main contribution of this approach is resulting speed boost to the process, which greatly outpaces existing traditional Wi-Fi approaches, becoming one of the fastest ways to obtain a complete forensic dump of Apple's iPad. In fact, we have apparently reached the speed limit of the iPad's dock connector.

Up to this day, similar transfer rates could only be achieved using commercial tools which are paid and/or restricted to law enforcement agencies, opaque to the scientific community and undocumented. Therefore, it is difficult to assess what is really happening during the imaging process and whether the original data is being somehow altered.

As far as iPad forensics is concerned, a fast imaging method opens some interesting research lines for the future. The ones we find most interesting are live memory dump of the device, study of the iOS 4 encryption system, an autonomous imaging from the iPad to the connected USB drive eliminating the need of a network and a remote computer and analyzing of the forensic artifacts left by the AirPrint subsystem.

Acknowledgments

This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

References

[1] InformationWeek, "iPad is top selling tech gadget ever", 2010, http://www.informationweek.com/showArticle.jhtml?articleID=227700347.

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

[3] Quickoffice Inc., "Quickoffice Connect Mobile Suite for iPad on the iTunes App Store", 2010, http://itunes.apple.com/us/app/quickoffice-connect-mobile/id376212724.

[4] Apple Computer Inc, "Pages for iPad on the iTunes App Store", 2010, http://itunes.apple.com/us/app/pages/id361309726.

[5] Apple Computer Inc, "Numbers for iPad on the iTunes App Store", 2010, http://itunes.apple.com/us/app/numbers/id361304891.

[6] Apple Computer Inc., "Apple's AirPrint Wireless Printing for iPad, iPhone and iPod touch Coming to Users in November", 2010, http://www.apple.com/pr/library/2010/09/15airprint.html.

[7] EuroSmartz Ltd., "PrintCentral for iPad on the iTunes App Store", 2010, http://itunes.apple.com/us/app/printcentral-for-ipad/id366020849.

[8] George Hotz, "iPhone serial hacked, full interactive shell", 2007, http://www.hackint0sh.org/f127/1408.htm.

[9] J.R. Rabaiotti and C.J. Hargreaves, "Using a software exploit to image RAM on an embedded system ", *Digital Investigation*, vol. 6, pp. 95–103, 2010.

[10] Katana Forensics, "Lantern", http://katanaforensics.com/use-our-tools/lantern/.

[11] Forensic Telecommunications Services Ltd., "iXAM - Advanced iPhone Forensics Imaging Software", http://www.ixam-forensics.com/.

[12] Moenner L. Jansen W, Delaitre A, "Overcoming impediments to cell phone forensics", in *In Proceedings of the 41st Annual Hawaii International Conference on System Sciences*. 2008, pp. 483 – 483, IEEE CSP.

[13] Comex, "Comex 'star' GIT repository", 2010, http://github.com/comex/star.

[14] "Jailbreak Matrix", 2010, http://www.jailbreakmatrix.com/iPhone-iTouch-Jailbreak.

[15] iPhone Dev Team, "Thanksgiving with Apple", 2010, http://blog.iphone-dev.org/post/1652053923/thanksgiving-with-apple.

[16] Jay Freeman 'Saurik', "Bringing Debian APT to the iPhone", 2008, http://www.saurik.com/id/1.

[17] Sophos Security, "First iPhone worm discovered - ikee changes wallpaper to Rick Astley photo", 2009, http://nakedsecurity.sophos.com/2009/11/08/iphone-worm-discovered-wallpaper-rick-astley-photo/.

[18] Apple Computer Inc, "Apple iPad Camera Connection Kit", 2010, http://store.apple.com/us/product/MC531ZM/A.

[19] International Organization for Standarization (ISO), "ISO 15740:2008 – Electronic still picture imaging – Picture transfer protocol (PTP) for digital still photography devices ", 2008.

[20] Camera & Imaging Products Association, "Design rule for Camera File system: DCF version 2.0", 2010.

[21] Microsoft Corp, "Limitations of the FAT32 File System in Windows XP", 2007, http://support.microsoft.com/kb/314463/.

[22] Ridgecrop Consultants Ltd., "Fat32Format", 2009, http://www.ridgecrop.demon.co.uk/index.htm?fat32format.htm.

[23] 9to5 Mac, "iOS 4.2 emits less USB power on iPad, Camera Connection Kit crippled?", 2010, http://www.9to5mac.com/40091/ios-4-2-emits-less-usb-power-on-ipad-camera-connection-kit-crippled.

[24] Apple Computer Inc, "iOS 4: Understanding data protection", 2010, http://support.apple.com/kb/HT4175.

[25] LLC Digital Forensics Solutions, "Scalpel: A Frugal, High Performance File Carver", 2006, http://www.digitalforensicssolutions.com/Scalpel/.

[26] Jonathan Zdziarski, "iPhone Insecurity", 2010, http://www.iphoneinsecurity.com/.