Fraunhofer
SIT

iOS Keychain Weakness FAQ

Further Information on iOS Password Protection

Jens Heider, Rachid El Khayari

Fraunhofer Institute for
Secure Information Technology (SIT)

December 5, 2012

Updated versions can be found at:
http://sit4.me/ios-keychain-faq

Contact person:
Dr. Jens Heider
Fraunhofer Institute for Secure Information Technology (SIT)
Rheinstraße 75, 64295 Darmstadt, Germany
Email: jens.heider@sit.fraunhofer.de
Phone: +49 (0) 61 51/869-233

Revision history

1.9 2012-12-05

added: Appendix A Protection Class Overview, p. 15
added iOS 6.0.1 keychain entry classification table
updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
iOS 6.0.1 is affected
updated: 2.19 Which devices are in danger?, p. 11
added iPad4, iPad mini and iPhone 5 to the list of currently unaffected devices

1.8 2012-07-16

added: 2.20 Is the SIM PIN affected?, p. 11
SIM PIN can be extracted
updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
iOS 5.1.1 is affected
updated: 2.19 Which devices are in danger?, p. 11
added iPad3 to the list of currently unaffected devices
updated: Appendix A Protection Class Overview, p. 14
SIM PIN and Bluetooth Link Keys classification added to table
updated: Appendix A Protection Class Overview, p. 14
keychain entry classifications updated for iOS 5.1.1 release

1.7 2012-05-10

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
iOS 5.1 is affected
updated: Appendix A Protection Class Overview, p. 14
keychain entry classifications updated for iOS 5.1 release
updated: Matthias Boll left the team and Rachid El Khayari entered as co-author

1.6 2012-02-27

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
iOS 5.0.1 is affected
updated: 2.3 Are X.509 certificates also affected?, p. 6
certificates in lower class than passwords
updated: 2.19 Which devices are in danger?, p. 11
added iPad2 and iPhone 4S, potentially affected via Absinthe jailbreak
updated: Appendix A Protection Class Overview, p. 13
keychain entry classifications updated for iOS 5.0.1 release

1.5 2011-11-22

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
iOS 5 is affected
updated: 2.12 Is there a patch available?, p. 8
iOS 5 provides some fixes
updated: 2.13 What are mitigation options from your experience?, p. 9
updated to reflect iOS 5 changes
updated: 2.16 Some passwords are not accessible with the shown method. Doesn't this prove the general security of the used concept?, p. 10
updated to reflect iOS 5 changes
updated: 2.17 Does the presented attack still work in iOS 5?, p. 10
updated with final iOS 5 test results
updated: 2.19 Which devices are in danger?, p. 11
added iPhone 4S (currently unaffected)
updated: Appendix A Protection Class Overview, p. 13

keychain entry classifications of final iOS 5 release

1.4 2011-09-23

updated: 2.15 Is the iOS keychain in general insecure?, p. 9
Fixed broken reference

1.3 2011-09-13

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
(Tests of iOS firmware 5.0 beta7 indicate an improved keychain implementation; further results will be released for final version)
updated: 2.12 Is there a patch available?, p. 8
updated: 2.17 Does the presented attack still work in iOS 5?, p. 10

1.2 2011-09-01

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
(iOS firmwares 4.3.4, 4.3.5, 5.0 beta2 added to affected versions, description of the protection classes)
added: Appendix A Protection Class Overview, p. 13 (up-to-date tables of keychain entry classifications)
added: 2.17 Does the presented attack still work in iOS 5?, p. 10
added: 2.18 What are the effects when no passcode is set?, p. 11
added: 2.19 Which devices are in danger?, p. 11

1.1 2011-05-06

updated: 2.1 Which versions of iOS are affected by the attack method?, p. 5
(iOS firmware 4.3.3 added to affected versions)

1.0 2011-04-20 (First version)

Abstract

This paper summarizes answers to frequently asked questions about the iOS keychain weakness described in [HB11]. It provides further information for security evaluators on the impact of the findings.

1. Introduction

The *keychain* in Apple iOS devices is intended as secure storage for sensitive user data that should be protected even if an attacker has physical access to the device. Therefore, the keychain stores secrets for services used by the operating system (e.g., WiFi passwords, VPN credentials, etc.) but also credentials stored by 3rd party apps.

The protection of the keychain is of vital concern for device users, as an unauthorized access to contained passwords, keys and digital identities may be used to access sensitive data, to cause loss of data and may inflict financial loss.

With the previous publication on *Practical Consideration of iOS Device Encryption Security* the risks should be highlighted that accompany losing a locked iOS device regarding confidentiality of passwords stored in the keychain [HB11]. The paper presented results of hands-on tests that showed the possibility for attackers to reveal some of the keychain entries. For the described approach, the knowledge of the user's secret passcode was not needed, as the protection provided by the passcode was bypassed.

As intended, the publication of our findings has provided the possibility for a public discussion on the iOS keychain security. Also others have found weaknesses in the keychain concept, but seem to have chosen to keep this for their own [For11]. However, the general reaction we observed as the result of the publication was very positive. Many requests on further details and other aspects have provided the opportunity to increase the knowledge about the provided protection.

This paper should provide answers from our perspective to many important questions arisen from the initial publication. As for the initial publication, the intention is to provide as much insights as possible to enable others to understand and evaluate the impact of the observed security design, but without causing too many benefits for the wrong hands.

2. FAQ

2.1. Which versions of iOS are affected by the attack method?

As of this writing, all versions of iOS up to (and including) iOS 6.0.1 are affected by the described attack method. For all the devices mentioned in section 2.19 exist exploits to perform the attack. This means that in all these iOS versions at

least the entries marked *Always* in Appendix A are vulnerable to an attacker with physical access to the device. Although still affected, iOS 5 introduced some improved protection to the keychain implementation, those changes are described in section 2.17.

The *Always* protection class means that the entry is decryptable without user interaction right after the device finished booting, the encryption is not tied to the user passcode. These entries can easily be retrieved by an attacker. On the other hand, *AfterFirstUnlock* requires the passcode and is decryptable after a user unlocked the screen and entered his passcode after a device boot. The third, passcode-dependent class *WhenUnlocked* is only readable while the device is not locked. This prevents access when the device is locked again, whereas *AfterFirstUnlock* entries stay open after the first unlock. Each class exists in a migratable and non-migratable (*ThisDeviceOnly* suffix) variation, with the latter tying the encryption to a specific device. In our described attack model this differentiation has no influence on the attack. All the classes are documented in [App10a].

However, in iOS 3.x and prior versions *all* entries of the keychain can be accessed by attackers. In these versions all keychain entries are encrypted *only* with the device key accessible on the device. Therefore, for those devices, the knowledge of the passcode is not needed to decrypt all keychain entries.

2.2. Are also the credentials of App XY affected?

This depends on how the app stores the credentials and how it uses the keychain. If the developer chose an alternative way of storing the user's credentials (instead of using the keychain), it depends on the strength of this proprietary protection.

If the developer chose to store the data in the keychain and instructed the system to make the stored credentials available even when the device is locked by setting the accessibility to *kSecAttrAccessibleAlways* (cp. [App10a]), then the app is affected. Otherwise the credentials of the app are *not* affected by the attack method.

2.3. Are X.509 certificates also affected?

Yes, certificates stored in the keychain by the OS or apps can be affected by the weakness as well. Like for passwords the accessibility depends on the applied protection class (see section 2.14). In certain scenarios certificates are even ranked in lower security classes than the corresponding passwords. In iOS 5.0 e.g. VPN passwords are stored in the *AfterFirstUnlock* protection class while VPN client-certificates and private keys are stored in the *AlwaysThisDeviceOnly* protection class.

2.4. Do iOS configuration profiles provide a better protection than manual settings?

No. After acceptance of the configuration profile, the device stores the credentials in the data structures as if entered manually in the setup dialogs.

2.5. Are credentials in encrypted iOS configuration profiles protected?

No. Configuration profiles of the iPhone Configuration Utility (iPCU) can be exported with the setting "signed and encrypted". This is meant purely to secure the channel between iPCU and the device. If the device has the matching CA certificate to verify the signature it decrypts the configuration file and installs it. The settings and passwords from the configuration profile are now stored like any other in the keychain. This means the export settings of the iPCU do not provide any additional security on the device and against the described attack method.

2.6. Do you provide the scripts used in the demo?

No, please note that we do not publish our scripts, parts of it or further details on the used system functions. The only exception is made for law enforcement agencies with direct authorization of the judiciary.

2.7. What data may get lost with the described attack method?

We used a so-called *custom ramdisk* for the jailbreak. This technique leaves everything in place on the iOS device, so no user data gets lost. After mounting the filesystem of the device, all files are accessible. Some entries in these files are encrypted additionally. In our publication we showed that it is possible to decrypt some of these encrypted items of the keychain without knowing the passcode.

2.8. Can changing the root password prevent the attack?

No. See also section 2.9.

However, if the device is already jailbroken, it is of course important to change the passwords for the accounts *root* and *mobile*. Otherwise attackers can also perform serious remote attacks via wireless networks.

2.9. Does jailbreaking / modifying a device prevent an attacker from accessing secrets in the keychain?

No, the access possibilities to the iOS via a custom boot are that low level that an attacker can revert all changes to the device during the jailbreak step with-

out loosing user data.

In the case of a changed root password from the default "alpine" to something unknown to the attacker, simply installing an SSH certificate (together with the SSH server) during the jailbreak will do. Also other device modifications (e.g., changed SSH configuration, changed file permissions, etc.) can not prevent an attacker from reverting these changes during the jailbreak step as the attacker already possesses root privileges in this step.

In the meantime we have changed our scripts to work also on already jailbroken devices to demonstrate that this is no protection. Even worse: on a jailbroken device a trojan app downloaded e.g. from Cydia store can read *all* keychain entries (and send it to the attacker) when the user has started the app.

2.10. Did you inform Apple Inc. prior to publication?

Yes. Our mission is helping industry to understand security, and to work with industry in developing security solutions. Our goal is not publishing hacks, but rather to be on the constructive side. We are well aware that the world benefits more from a constructive approach to security than from quickly publishing hacks.

This work came out of client requests to analyze the security of the Apple iOS, which is a major concern for many enterprises. As pointed out in our report, the problem is caused as a direct consequence of a trade-of between functionality and security.

We're in contact with Apple regarding this issue since September 2010 and of course they were aware of the possibilities an attacker has after jailbreaking a device.

Prior to publication we contacted Apple again, but did not receive a comment on how or if this will be addressed in the future.

2.11. Did Apple Inc. announce a timeline for mitigations?

As of this writing we're not aware of a planned timeline.

2.12. Is there a patch available?

No, as of this writing Apple has not announced a patch or configuration option that changes the observed behavior for iOS 4.

But the observed changes in iOS 5 provide some significant security improvements (see section 2.17), therefore an update is strongly recommended security-wise. However, these changes are not documented by Apple and a configuration option to classify keychain entries is still not available.

2.13. What are mitigation options from your experience?

The possibilities for protection depend on the use cases, the installed apps and the environment in which the iOS device shall be used and protected. In iOS 4, we proposed to switch from MS Exchange to IMAP / SMTP / ICAL servers (to prevent the vulnerable push passwords) and to refrain from using affected VPN and WLAN configurations by using other protection mechanisms as possible mitigation options. However, this may require major changes to the infrastructure that should be done carefully to prevent other flaws. Instead, we recommended to update to iOS 5.

For iOS 5, we can confirm that the use of MS Exchange with passwords is secured now. But this does not include user certificates for authorization, because these are still accessible for an attacker. The same changes apply to Virtual Private Networks. The updated security of other credentials can be found in Table A in Appendix A.

For organizational measures please also refer to the conclusion section in [HB11].

2.14. I'm an app developer. How can I protect the credentials of my apps?

Set the keychain item accessibility value to *kSecAttrAccessibleWhenUnlocked* or kSecAttrAccessibleAfterFirstUnlock. This prevents keychain access without knowledge of the passcode. If you do not specify the setting, the current default is kSecAttrAccessibleWhenUnlocked, but to prevent being affected by future changes of this default you should explicitly define your protection class. Refer to Apple developer portal [App10a] for further details.

2.15. Is the iOS keychain in general insecure?

No, the iOS keychain can provide sufficient security for stored items, if *both* of the following conditions are met:

- Items are stored with a protection class that makes them only available when the device is unlocked. (see Section 2.14)
- A strong passcode of 6 alphanumeric digits is enforced (reduces the risk[1] of brute-force attacks).

1 according to [App10b]: ≈50ms per password try; → ≈20 tries per second; → ≈1.7 years for a 50% change of guessing the correct passcode for a 6-digit alphanumeric code with base 36. The standard simple code of 4 numeric digits would be brute-forced in less than 9 minutes. Based on the assumption that the counter for wrong tries in the iOS can be bypassed, as it is not hardware-based. [BS11b] provides a tool for passcode bruteforce attacks.

2.16. Some passwords are not accessible with the shown method. Doesn't this prove the general security of the used concept?

Well, though it is indeed good news to find passwords e.g. stored by the Safari browser to be encrypted with the passcode[2], in the field the threat to a user is exposed by any available stored secret. Available credentials can be exploited to gain access to other accounts. Additionally, it is also debatable how many different passwords are really used by common users. Of course, users are instructed to refrain from using the same passwords over and over again, but this advice cannot be enforced.

Therefore, from our perspective, the problems with the concept that threatens users having higher security demands are

- the non-configurable trade-off between functionality and security,
- the unavailability of security information about the actual provided protection for specific keychain items (caused by the differences of the protection classes),
- and the non-disclosure of the actual protection scheme preventing efficient in-depth evaluation of the protection strength.

2.17. Does the presented attack still work in iOS 5?

Our tests of the final release of iOS 5 show some changes to the keychain implementation. Now all data fields, not only the secret, are saved in an encrypted dictionary structure. Account names and descriptions are no longer stored as clear-text in the database but in hash form. Bédrune and Sigwald describe those changes in [BS11a]. This effectively means that an attacker is now prevented from retrieving every account name, he only knows those where he also knows the secret.

The iOS 5 tests also confirmed that the attack still works. It might have an even larger impact due to the introduction of iCloud and the increasing ubiquity of smartphones. On a positive note, Apple has changed the protection classes for a few items to be more secure, in particular Microsoft Exchange, VPN and calendar passwords. Those entries are no longer decryptable without passcode knowledge in iOS 5.

The Table A in Appendix A shows the keychain entries in iOS 5 and how they are protected. For some of them the specific use is currently unknown and subject to further research.

2 See tables in Appendix A

2.18. What are the effects when no passcode is set?

While being very convenient, this would eliminate the security provided by the iOS keychain. All entries, regardless of protection class, will be accessible with our attack. We strongly encourage enabling the lock screen passcode.

2.19. Which devices are in danger?

The attack described in [HB11] currently works for iPod Touch 1-4, iPhone 3G/3GS/4 and iPad. For all these devices exists a bootrom exploit which can be used to make the filesystem changes and disable the code signing that our attack needs.

However, newer versions of these devices, which use A5/A5X/A6 processors are currently not affected. This refers to the iPod Touch 5, iPad 2/3/4, iPad mini and iPhone 4S/5.

Nevertheless these newer devices are far from being completely secure. An exploit for the devices has already been published (Absinthe Jailbreak / Rocky Raccoon) but at the current moment the workflow of the jailbreak tools requires the targeted device to be unlocked. Anyhow, the keychain weakness is on the operating system level, which means that malicious software run on the iPad 2/3/4 or iPhone 4S/5 can access the keychain like our script. And once a device is jailbreakable via bootrom exploit it is rendered vulnerable again.

2.20. Is the SIM PIN affected?

At the writing of this document the SIM PIN **is** affected, as it can be extracted from the keychain with our presented approach.

According to a document on iOS Security [Inc12] recently published by Apple Inc., the keychain contains a copy of the SIM PIN which is protected by the protection class *AlwaysThisDeviceOnly*. After investigating this previously unpublished fact, we verified that the SIM PIN is truly stored in the keychain, after the user has unlocked the SIM.

From our perspective, storing the user's SIM PIN for a longer period than required for completing the authentication process violates the 3GPP standard for USIM and IC card requirements. In section 5.3 of [3GP11] the standard states:

"*User related security codes such as PIN and Unblock PIN may only be stored by the ME during the procedures involving such a code and shall be discarded by the ME immediately after completion of the procedure.*"

However, in our tests, the SIM PIN entry is kept on the iPhone during operation and standby times and is only deleted by a clean-up process during a normal system shutdown.

This violation results in a direct loss of security, because if the device is rebooted with a hard reset instead of a clean shut down, the clean-up process does not

delete the SIM PIN from the keychain, which makes it extractable with our approach. In other words, it is possible to extract the SIM PIN from the keychain without knowing the passcode, as long as the device is stolen or is found powered on. This approach is demonstrated in the video *Extracting SIM PIN from iPhone* [HEK12].

A. Protection Class Overview

Table 1: classification of iOS 4 keychain entries

Entry Description	Secret Type	kSecAttrAccessible
AOL Email	Password	AfterFirstUnlock
Apple Push	Certificate + Token	**Always-ThisDeviceOnly**
Apple-token-sync (mobile me)	Token	**Always**
Apps using default the protection	depends on App	WhenUnlocked
Backup Password	Password	WhenUnlockedThisDeviceOnly
CalDAV	Password	**Always**
GMail	Password	AfterFirstUnlock
GMail as MS Exchange	Password	**Always**
iChat.VeniceRegistrationAgent	Token	**Always-ThisDeviceOnly**
Identity Certificate (e.g. for VPN)	Certificate	**Always-ThisDeviceOnly**
IMAP	Password	AfterFirstUnlock
LDAP	Password	**Always**
Lockdown-identity	Certificate	**Always-ThisDeviceOnly**
MCEmail Account (probably created by iPCU profile)	plist with IMAP password	**Always**
MS Exchange	Password	**Always**
Passwords saved in Safari	Password	WhenUnlocked
SMTP	Password	AfterFirstUnlock
Visual Voicemail	Password	**Always**
VPN IPsec Shared Secret	Password	**Always**
VPN XAuth Password	Password	**Always**
VPN PPP Password	Password	**Always**
WiFi (Company WPA with LEAP)	Password	**Always-ThisDeviceOnly**
WiFi (WPA)	Password	**Always-ThisDeviceOnly**
Yahoo Email	Token + Cookie	AfterFirstUnlock

Table 2: classification of iOS 5.1.1 keychain entries

Entry Description	Secret Type	kSecAttrAccessible
AOL Email	Password	AfterFirstUnlock
Apple ID	Private Keys	WhenUnlocked
Apple ID Authentication Password	Token	AfterFirstUnlock-ThisDeviceOnly
Apple Push	Token	**AlwaysThisDeviceOnly**
Apple Ubiquity (iCloud)	Certificates + Private Keys	**AlwaysThisDeviceOnly**
Apple-token-sync	Token	**Always**
Apps using default class	depends on App	WhenUnlocked
APSClientIdentity	Certificate	**AlwaysThisDeviceOnly**
Backup Password	Password	WhenUnlocked-ThisDeviceOnly
Bluetooth Link Key	Key	**AlwaysThisDeviceOnly**
CardDAV	Password	AfterFirstUnlock
CalDAV	Password	AfterFirstUnlock
GMail Account	Password	AfterFirstUnlock
iChat message-prot.-key	Key	**AlwaysThisDeviceOnly**
Identity Certificate (e.g. VPN)	Certificate + Private Key	**AlwaysThisDeviceOnly**
ids identity (probably iChat/iMessage)	Certificate + Private Key	**AlwaysThisDeviceOnly**
IMAP	Password	AfterFirstUnlock
iMessage Encryption Key	Key	**AlwaysThisDeviceOnly**
iMessage Signing Key	Key	**AlwaysThisDeviceOnly**
iPhone Configuraton Utility	CA Certificates + Private Key	**AlwaysThisDeviceOnly**
LDAP	Password	WhenUnlocked
Lockdown-identity	Certificate + Private Key	**AlwaysThisDeviceOnly**
MCEmail Account (probably created by iPCU profile)	plist with IMAP password	**AlwaysThisDeviceOnly**
MS Exchange	Password	AfterFirstUnlock
Passcode policy settings	plist with hashes of old passcodes	WhenUnlocked-ThisDeviceOnly
Passwords saved in Safari	Password	WhenUnlocked
SIM PIN	PIN	**AlwaysThisDeviceOnly**
SMTP	Password	AfterFirstUnlock
Subscribed Calendars	Password	AfterFirstUnlock
Visual Voicemail	Password	**Always**
VPN Passwords	Password	AfterFirstUnlock
VPN Certificates	Certificate + Private Key	**AlwaysThisDeviceOnly**
WiFi	Password	**Always-ThisDeviceOnly*** AfterFirstUnlock**
Yahoo Email	Token	AfterFirstUnlock

* if configured via iPhone Configuration Utility

** if configured on device

Table 3: classification of iOS 6.0.1 keychain entries

Entry Description	Secret Type	kSecAttrAccessible
Apple ID Authorization & Intermediate Cert.	Certificate	**Always**
Apple ID Authorization Private Key	Key	AfterFirstUnlock
Apple ID Authentication Password	Token	AfterFirstUnlockThisDeviceOnly
Apple Push	Token	**AlwaysThisDeviceOnly**
Apple ID iCloud	Token	AfterFirstUnlock
Apple ID Facetime	Token	AfterFirstUnlock
Apple iTunes Store	Token	AfterFirstUnlockThisDeviceOnly
Apple Ubiquity (iCloud)	Cert. + Private Keys	**AlwaysThisDeviceOnly**
Apple Token fmip	Token	**Always**
Apps using default class	depends on App	WhenUnlocked
APSClientIdentity	Certificate	**AlwaysThisDeviceOnly**
Backup Password	Password	WhenUnlockedThisDeviceOnly
Bluetooth Link Key	Key	**AlwaysThisDeviceOnly**
CardDAV	Password	AfterFirstUnlock
CalDAV	Password	AfterFirstUnlock
GMail / AOL / Yahoo Account	Password	AfterFirstUnlock
Identity Certificate (e.g. VPN)	Certificate + Private Key	**AlwaysThisDeviceOnly**
ids identity	Certificate + Private Key	**AlwaysThisDeviceOnly**
IMAP	Password	AfterFirstUnlock
iMessage Encryption Key	Key	**AlwaysThisDeviceOnly**
iMessage Signing Key	Key	**AlwaysThisDeviceOnly**
iPhone Configuraton Utility	CA Certificates + Private Key	**AlwaysThisDeviceOnly**
LDAP	Password	WhenUnlocked
Lockdown-identity	Certificate + Private Key	**AlwaysThisDeviceOnly**
MCEmail Account (probably created by iPCU profile)	plist with IMAP password	**AlwaysThisDeviceOnly**
MS Exchange	Password	AfterFirstUnlock
Passcode policy settings	plist with hashes of old passcodes	WhenUnlockedThisDeviceOnly
Passwords saved in Safari	Password	WhenUnlocked
SIM PIN	PIN	**AlwaysThisDeviceOnly**
SMTP	Password	AfterFirstUnlock
Subscribed Calendars	Password	AfterFirstUnlock
Visual Voicemail	Password	**Always**
VPN Passwords	Password	AfterFirstUnlock
VPN Certificates	Certificate + Private Key	**AlwaysThisDeviceOnly**
WiFi	Password	**AlwaysThisDeviceOnly*** AfterFirstUnlock**

* if configured via iPhone Configuration Utility

** if configured on device

References

[3GP11] 3GPP TS 21.111 V10.0.0. Technical Specification Group Core Network and Terminals; USIM and IC card requirements (Release 10). http://www.3gpp.org/ftp/Specs/archive/21_series/21.111/21111-a00.zip, April 2011. 11

[App10a] Apple Inc. Keychain item accessibility constants. http://developer.apple.com/library/ios/documentation/Security/Reference/keychainservices/Reference/reference.html#//apple_ref/doc/constant_group/Keychain_Item_Accessibility_Constants, September 2010. 6, 9

[App10b] Apple Inc. WWDC 2010, Core OS, Session 209 "Securing Application Data", Slide 24. http://developer.apple.com/videos/wwdc/2010/, 2010. 9

[BS11a] Jean-Baptiste Bédrune and Jean Sigwald. iOS 5 data protection updates. http://esec-lab.sogeti.com/post/iOS-5-data-protection-updates, October 14 2011. 10

[BS11b] Jean-Baptiste Bédrune and Jean Sigwald. iPhone Data Protection. http://code.google.com/p/iphone-dataprotection, August 2011. 9

[For11] Chris Foresman. Security expert: iPhone password hack shows flawed security model. http://arstechnica.com/apple/news/2011/02/six-minute-keychain-hack-highlights-busted-iphone-security-model.ars, February 2011. 5

[HB11] Jens Heider and Matthias Boll. Lost iPhone? Lost Passwords! Practical Consideration of iOS Device Encryption Security. http://www.sit.fraunhofer.de/Images/sc_iPhone%20Passwords_tcm501-80443.pdf, February 9 2011. 5, 9, 11

[HEK12] Jens Heider and Rachid El Khayari. Demo: Extracting SIM PIN from iPhone. http://www.youtube.com/watch?v=hpFdfv3--3s, July 2012. 12

[Inc12] Apple Inc. iOS Security. http://images.apple.com/ipad/business/docs/iOS_Security_May12.pdf, May 2012. 11