International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 03 – Issue 03, May 2014

iOS Security and Privacy: Authentication Methods, Permissions, and Potential Pitfalls with Touch ID

Stephen J. Tipton, Daniel J. White II, Christopher Sershon, and Young B. Choi
Department of Business, Leadership, and Information Systems
College of Arts & Sciences
Regent University
1000 Regent University Drive
Virginia Beach, VA, 23464
USA
e-mail: ychoi {at} regent.edu

Abstract — **We investigate the conceptual problem with iOS security, surveying a number of issues encompassing the utilization of biometric authentication. Specifically, we suggest that data collected from fingerprint scans for the purpose of authenticating an iOS user may be recorded by Apple, generating privacy issues. Furthermore, the potential for faking fingerprints, in addition to potential usability issues will be explored. A historical perspective of iOS versions is surveyed from a security standpoint, as well as its future potential moving forward with sophisticated authentication methods.**

Keywords-iOS; Apple; security; access; users; biometrics

I. INTRODUCTION

iOS 7 was released to public fanfare in early September 2013. Aside from the drastic change in user interface and experience is the method in which iPhone users are able to optionally unlock their devices. Some security experts have raved at the new feature; however, some warn that while fingerprints work well, they may not work well for everyone [1]. Adding complication, some security experts point to decade-old findings outlining the ability to fake fingerprints with gelatin-based fake fingers. While Gartner optimistically points to Apple's iOS 7 release as a "significant step forward," [2] others find the fingerprint reader to be more of a convenience than a security measure [3].

We will survey the conceptual problems with iOS security, surveying a number of issues involving the utilization of Touch ID, Apple's recent feature in implementing biometric authentication. The survey considers the controversial idea that data collected from such biometric user authentication using fingerprint scans may be recorded by Apple, implying privacy issues. Furthermore, we will examine the potential for falsifying fingerprints, in addition to potential issues surrounding its usability. The latter point is interesting, considering the intention of Apple's usability-first strategy; this is discussed within the historical overview of iOS, the underlying mobile operating system powering the iPod Touch, iPhone, and iPad devices.

How will Apple address the controversial skepticism that some users may consider with the new biometric features in iOS 7-based devices? In addition to discussions involving the various security implementations Apple utilizes in their iOS development, pitfalls of the Touch ID feature, as well as the future of security-minded innovation will be discussed.

II. OVERVIEW OF iOS HISTORY

Since Steve Jobs' introduction of the iPhone, the Apple-designed smartphone, along with its iPad and iPod Touch counterparts have redefined the paradigm of mobile computing. Since 2007, Apple has consistently improved iOS into "one of the most feature-rich and well-supported platforms on the market" [4]. Consequently, as the years have passed, the conceptual issue of security could not be overlooked. The following high-level overview surveys each iteration of iOS beginning with iOS 1 and culminating with iOS 7; the latter featuring one of the more controversial security measures discussed later in this composition. The purpose of an historical overview is to gain an understanding that iOS, as a tremendously evolving platform, must certainly establish an equally-evolving and viable security approach.

A. iOS 1.0

With the original iteration of iOS, Apple's strategy was to focus on solidifying the core experience rather than competing from a feature standpoint with well-established mobile systems including Windows Mobile, Palm OS, and BlackBerry. From a level of security, iOS 1 was “almost entirely locked down to hackers and developers” [4].

In comparison to the competition, the original iPhone had no support for third generation mobile telecommunications technology (3G), multitasking, or third-party applications. Usability was very limited as well. Users could not copy or paste text, could not make e-mail attachments, and could not take advantage of Multimedia Messaging Service (MMS) capabilities.

Many other features were strategically left out of the initial iteration of iOS. However, as per Apple’s strategy with iOS 1, the core user interface was revolutionary in the age of smartphones. Prior to the introduction of iOS, any touch interaction with a similar device (such as Palm OS) was limited to the use of a stylus and a resistive touchscreen. Apple’s strategy was to implement touch interaction as its primary interaction model, moving towards a more natural element of interaction with inertial scrolling and pinch-to-zoom features.

B. iOS 2.0

The introduction of the App Store, Apple’s iOS software distribution platform, was a prolific feature in the next iteration of iOS. Although not necessarily groundbreaking, Apple managed to make this feature appear new and innovative with its system for developing, browsing for, and installing third-party applications, also known as *apps*.

In a significant change of direction from the method in which mobile apps were previously distributed, Apple integrated the App Store both on the device as well as within iTunes; this decision eased the ability to browse and install third-party apps. Because of the direct pairing of the App Store with the established customer base within iTunes, users would not be required to re-enter credit card information to make purchases. Naturally, this resulted in more purchases being driven by “impulse buys” [4].

C. iOS 3.0

The third generation of iOS introduced copy-paste functionality, utilizing text magnification selection, a touch-friendly feature “well ahead of the competition in terms of usability” [4]. While this feature arrived far later than originally hoped for, it coincides with Apple’s strategy to solidify the user interface and user experience.

Apple also introduced push notifications for third-party apps, which eventually would become more of an annoyance than a convenience. iOS 3 also introduced the ability to directly purchase movies, TV shows, and books from the device’s iTunes store; efforts in expanding its revenue model to support in-app purchases and subscriptions were also featured. iOS 3 devices also included the new “Find My iPhone” feature, in addition to market-leading parental controls, and Bluetooth support.

D. iOS 4.0

Organizational features such as app folders, as well as usability features such as multitasking, Wi-Fi tethering, spell-check, and customized Spotlight searching were introduced in iOS 4, along with numerous other features.

Although multitasking in its natural form wasn’t supported, this iteration of iOS did offer its app developers a number of services that could run in the background including, but not limited to, local notifications allowing queued alerts; task completion to support the completion of uploads, (e.g., when a user closes an app); the ability to play music in the background; and, navigation apps maintaining the device’s current location when users switch to a different app.

E. iOS 5.0

Of more notoriety with the unveiling of iOS 5 was the introduction of Siri, a virtual assistant intended to replace Voice Control. Originally available only on the iPhone 4S, Siri is able to go beyond simply connecting phone calls, communicating “with everything from [a user’s] calendar to [computational knowledge engine] Wolfram Alpha,” [4] allowing users to ask questions in a natural language.

iOS 5 also unveiled iMessage, its messaging competition to BlackBerry Messenger. Like its competitor, iMessage is able to show delivery receipts and send multimedia messages.

A more revolutionary change with iOS 5 was the de-coupling of the device from a PC, once a tedious requirement for the purpose of activation. As a result, any iDevice could now theoretically become a user’s sole computing device; [4] therefore, revolutionizing the paradigm of mobile computing.

iTunes Wi-Fi sync, over-the-air updates, and iCloud were also major new features made available with iOS 5.

F. iOS 6.0

iOS 6 welcomed one feature that may prove beneficial to the future of the mobile payments market with the introduction of Passbook. Passbook collects rewards cards, payment types, tickets, coupons, boarding passes, and anything else that contains a barcode; its inclusion is clearly strategic, as it could propel Apple into direct competition with Google Wallet, provided it follows through with integration of its iTunes Store and collection of active credit card numbers [4].

With the release of iOS 6, the integration of Google Maps as the map provider was replaced with Apple Maps. Also removed was the more usability-related requirement to be on a Wi-Fi network in order to use FaceTime. Noted as a use-limited feature with its origination in iOS 4, due to the availability restriction, FaceTime is now available on both LTE and 3G networks. Users are now able to add their phone numbers just as they would an e-mail address to help facilitate adoption on 3G and LTE iPads, enabling the user to receive FaceTime calls on their iPads using the phone number [4].

G. iOS 7.0

iOS 7 brought extensive changes beginning with a complete visual overhaul, in addition to the introduction of the Control Center. The latter feature is Apple’s rendition of a quick-toggle settings panel similar to what is offered on Android devices.

Multitasking was largely updated in this release, both in interaction and implementation.

Double-clicking the home button now reveals full page previews of currently open apps; users can close apps by swiping them away in an upward motion, replacing the original tile-based close button.

This iteration became the first iOS iteration to arrive with not one, but two varying phones — the 5S and the 5C. With the 5C, users could opt for a reduced-cost alternative, with a colorful plastic shell versus the white or black aluminum shell. More notable, however, was the fact that the 5S introduced Touch ID, a biometric sensor that allows users the ability to optionally unlock the phone using a fingerprint, as well as make App Store purchases without providing a password.

III. IMPLICATIONS OF iOS SECURITY

In retrospect, the original iPhone shipped with the stripped-down iOS 1, a decision that exposes the following:

- There was a reduced attack surface.
- All processes ran as root, due to no privilege separation.
- There was no existence of code-signing enforcement.
- There was no data execution prevention (DEP).
- There was no address space layout randomization (ASLR).
- There was no sandboxing.

Essentially, if a vulnerability was exposed on the device, its exploitation would be very simple [5].

A. Usage Statistics

As of 2011, smartphones were approaching 25 percent of the mobile market [6]. The comScore MobiLens and Mobile Metrix has since reported as of October 2013, that smartphone usage has increased to a 60.8 percent original equipment manufacturer (OEM) market share, of which Apple leads with 40.7 percent [7]. As an astounding result, the increase in popularity is forcing more organizations to address security concerns on both personal and company-provided devices.

B. Authentication Implications

One concern worth noting is the inconvenient, as well as shoulder-surfing-prone utilization of a device PIN for the purpose of authentication. Primarily due to the utilization of an on-screen virtual keyboard, password input on a smartphone increases the risk of a bystander observing what is being entered [8]. Additionally, promoting the use of more stringent passwords compounds this susceptibility, as pointed out in Schaub, Deyhle, and Weber’s in-depth study of password entry usability and shoulder-surfing susceptibility with on-screen devices.

As an alternative method of user authentication, the release of iOS 7 introduces Touch ID, a biometric sensor allowing users the ability to unlock the device using a fingerprint, as well as make App Store purchases without the use of a passcode. The latter ability has since been patched in the release of the iOS 7.0.1 update [9]. This feature is available only on the iPhone 5S model.

In response to greater shoulder-surfing concerns in parallel with more stringent password requirements, Apple has countered with the fingerprint-based authentication mechanism’s reduction in requiring a manually-entered password in citing the growing practicality of utilizing “longer, more complex” passwords [10].

A passcode is “the first line of defense” against attackers (or innocent snoopers, such as a child) with physical access to a device [11]. One software implication addressed in both iOS and Android devices is a safeguard against offline brute force, or exhaustive password lookups. Strong passwords capable of withstanding brute force attacks have underlined the need in which "users may choose to lock their smartphones and set limit attempts after which a lock protocol is initiated" [12]. This implication could continue to recede with the further utilization of biometric features, although not without controversy.

IV. AUTHORIZATION AND PERMISSIONS

When an iOS device is initially turned on, the application processor executes code which is located in the device's read-only memory, known as the Boot ROM. This code, which contains the Apple Root CA public key, is implemented into the chip during manufacture. The public key confirms that the Low-Level Bootloader (LLB) is digitally signed by Apple prior to loading. This is the first instance in the chain of trust, known as the secure boot chain. Apple's primary goal in building the secure boot chain into iOS is to ensure that the software is not interfered with, even at its most basic level. This goal also includes restricting iOS to only run on appropriate Apple devices [13]. One of Apple's requirements in the iOS secure boot chain is that every step in the chain must ensure that its preceding step is signed by Apple before it allows anything else to execute [13].

A. Installation Authorization Server

When iOS is installed or updated on a device, iTunes communicates with Apple's installation authorization server, sending it three specific requirements before iOS can be downloaded, installed, or updated. Each device transmits a number of cryptographic measurements for every software item of the installation package, a nonce, and its unique ID (ECID). Once these requests are fulfilled, the authorization server verifies the series of measurements for each installation item. When a measurement's match is found, the server adds the device's ECID to that measurement and signs it. The ECID makes the authorization of each measurement unique, apart from any other iOS-related authorizations. Additionally, this authorization verifies that the item measurement that has booted on the device, along with the ECID, generates an exact match to the signature on the authorization server [13]. This helps Apple to achieve its goal of ensuring that each iOS download, update, and installation are strictly Apple-provided [13]. In addition to verifying the authenticity of each iOS download and installation, the authorization server also ensures that each authorization is for a specific iOS device and that any outdated versions of the software cannot be duplicated from one device and installed on another. For this purpose, a nonce must be sent to the server. The nonce, which is a random anti-replay value, prevents potential attackers from saving the authentication server's reply and using it to downgrade a user's device to a previous iOS version, opening more software vulnerabilities [13].

B. Application Permissions

iOS's secure boot chain provides a solid security foundation for Apple's mobile users, however it is not the complete summary of Apple's iOS security efforts. iOS also prevents third-party applications from executing non-permitted code, such as self-modifying code by requiring an Apple-issued certificate from all installed programs [13]. The OS also contains an application permission structure that isolates third-party applications with means to prevent them from gaining access to files belonging to other programs, as well as making unauthorized changes to the device. This permission structure is designed in such a way that each installed iOS application is randomly assigned a unique home directory where it will store all of its files [13]. In alliance with this permission structure, iOS APIs also restrict applications from expanding their own permissions to gain access to other programs or system files for the same reason [13]. With further regard to protecting system files, Apple mounts the entire iOS partition in read-only on each device as an effort to prevent them from being accessed or modified by other installed applications [13].

Third-party application access to user information, whether it is stored on the iOS device or on iCloud, is regulated using declared entitlements. These entitlements are key-value pairs that provide an authentication layer to the applications. They cannot be altered because they are signed digitally. iOS is also designed so that the sharing of information between different applications is only allowed between one sending program and one receiving program that use a custom URL or a shared keychain access group, preventing a direct exchange of information between one application and another [13].

C. Built-In AES Engine

Starting with the iPhone 3GS and any of its subsequent generations, including iPods and iPads, Apple began building 256-bit AES encryption directly into the hardware of its mobile devices. Each of these iOS devices contain an ID (UID) and a device group ID (GID), which are both 256-bit AES keys embedded into the devices' processors during assembly. These keys cannot be read by any application—only their results of encryption and decryption can be seen. The UID and the GID are planted into the processor's silicon to prevent them from becoming corrupted or modified; only the AES engine is provided access to them [13]. One notable function of the UID is that it allows data to be cryptographically tied to only one device, preventing any possibility of data being accessed if the memory chips are physically removed and used in another device [13]. When new files are created in iOS, a new 256-bit key is generated for each one and is sent to the

AES engine in the device's hardware. These keys are used to encrypt each file when written to the flash memory [13].

Despite Apple's built-in AES scheme that dates back to the iPhone 3GS, the SANS Institute claims that any devices that did not come with iOS 4 or later installed must be specially configured for this 256-bit encryption to function. Even if an iOS device was eventually upgraded to iOS 4 or later, it requires the AES hardware to be properly configured. On the other hand, devices that were originally purchased with iOS 4 and above already contain the AES encryption feature configured right out of the package [14].

D. The NSA's iOS Security Skepticism

In skepticism of iOS's security capabilities, the NSA believes that it is certainly possible for an attacker to take control of an iOS device remotely by compromising its security. If this were the case, the attacker may also be able to access and gather a large sum of information about the targeted mobile user. There are several different methods in which iOS intruders can obtain data after they have taken control of a device. These methods can include retrieving audio files, controlling cameras, seeing the device's geo-location, and reading and stealing other data. More particularly, attackers may hunt for information such as the user's credentials—whether they are stored on the device or in a remote location. In fact, attackers can even impostor the owner from remote locations in the form of identity theft, deceiving other people and computers on the network that the device is connected to [15].

E. User Security Services and Recommendations

One emphasized risk of using the iOS software, noted by the NSA in "Security Configuration Recommendations for Apple iOS 5 Devices," is the possibility of user data falling into the wrong hands as a result of users losing their devices [15]. To counter this consequence, iOS users have the advantage to use services such as ActiveSync, MDM, or iCloud to allow remote wiping [15]. Remote wiping, which completely erases all data from a missing device, is executed by removing the device's block storage encryption key from Effaceable Storage in a secure fashion. Discarding this particular key makes all data on the device unreadable for anyone who finds it [13].

In addition to these remote wipe services, "Find My iPhone" is an Apple-developed application that can be downloaded on an iOS device so that it can be spotted on a map if it becomes lost or stolen. There are also a number of other geo-location services available, similar to this one [15]. The NSA also encourages the use of passcodes to secure an iOS device since they not only prevent someone from gaining unauthorized access to the device, but they also provide protection from other network users who may attempt to "snoop" for hidden information. The NSA names this potential security threat "Data Compromise due to Casual Access Attempt" [15]. In addition to locking an iOS's device's content, passcodes also include a degeneration for encryption keys. Since encryption keys are not stored on the device itself, it is impossible for locked data to be accessed without the passcode. Both four-digit and arbitrary-length passcodes are supported in iOS [13].

The only way for an attacker to try breaking into an iOS device is by brute-force attack since the owner's passcode is also meshed in with the device's UID. In order to make brute-force attacks even more frustrating, a large iteration count is set in place so that each break-in attempt consumes about 80 milliseconds. In other words, "it would take more than five and a half years for someone to try all possible combinations of a six-character alphanumeric passcode with lowercase letters and numbers" [13]. Escalating time delays are also enforced between a certain number of failed passcode attempts, which adds even more frustration for attackers. Device owners can also adjust this feature to their preferences in settings, and are also able to configure the device so that all of their data is automatically wiped after ten failed passcode attempts [13].

Network security features are also incorporated into iOS and can be easily accessed for configuration. iOS devices are certainly capable of connecting to VPN servers where many different protocols and authentication methods are supported [13]. A wide range of standardized Wi-Fi protocols are also supported in iOS such as WPA2 Enterprise, for example. WPA2 Enterprise features a 128-bit encryption schema, providing iOS users with a premium level of data protection while connected to the Internet via Wi-Fi [13]. Because iOS devices are designed to have only one user, unlike laptops and desktops, their authorization controls are founded on different connections that reach them. Because of this, the SANS Institute recommends that iOS users disable any unnecessary connections and increase their focus on configuring connections that are necessary with the best security in mind [14].

V. PITFALLS OF BIOMETRIC SECURITY FEATURES

One of the biggest additions to iOS7 has been the use of biometrics as part of the system's security. While biometrics has received a reputation for being one of the best ways to authorize a person, it does not mean that it is without flaws. The inherent flaws of biometrics along with the commercial storage of a person's biometric signature have made the iOS7 one of the most controversial operating systems currently on the market.

One of the best ways to secure systems is to use something that only the authorized user can possess, such as a lock with only a single key. In theory, biometrics satisfies this description because items are encrypted with keys made from the unique features of the user's physical body. However, when put into practice, biometrics can become very vulnerable. A biometric scan can be subject to reverse engineering and hacking in the same way a lock can be subjected to the exploits of a lock pick. In 2004, members of the Information Systems Audit and Control Association released a report that identified and explained eleven security vulnerabilities with biometrics [24].

A. Spoofing

Spoofing is one of the common vulnerabilities that biometrics experience [18]. With spoofing, an attacker can use an artificial finger to fake out the sensor. Faking fingerprints may be achieved by lifting fingerprints off of something the victim has touched, and subsequently copied by using special materials. This is similar to a user leaving a key to their front door out where a criminal can grab it, copy it at a hardware store, and then return it to where it was found. This is a huge problem because it provides the hacker complete access to the account without the user ever knowing how or when access was gained.

The second most common risk to biometrics, specifically fingerprinting, is residual characteristics. This occurs when pieces of a victim's biometrics remain on the sensor, and the attacker uses it to gain access. This was made popular in 2012, according to Dimitriadis, when teenagers used a hack discovered by a security researcher involving gummy candies on the fingerprint scanners in grocery stores to charge items to accounts of other customers [18, 23]. Similar to the physical lock and key, residual templates are the digital equivalent of leaving the key to a house in the front door. While still as dangerous as someone copying the house key, the user will be able to track down when and where their biometrics were used.

A vulnerability that corresponds with residual templates, but carries the same magnitude as a copied template, is that of a similar template. This vulnerability occurs when a biometric system is not configured properly, or there is too much leeway given to a scan. A person with a similar fingerprint could manipulate the position of their finger to match the victim's signature, providing the attacker access to the victim's information. Physically, this is similar to bump key to gain entry to a locked room. A hacker can manipulate his or her fingerprint in the same way a lock pick knocks a bump key until its pins match the lock's tumbler; giving unlimited access without ever having to go through the trouble of finding and coping the victims original fingerprint.

B. Fake Templates

Another issue with biometrics is the use of fake templates. This vulnerability arises when there is a need to save biometric templates on a server. This method of storage puts the biometrics system at large risk since an attacker would merely need to gain access to the server and then upload fake templates of their fingerprints to the victim's account [18]. A physical example of this is a hacker creating a fake company badge in order for security guards to let him in the door. As long as the forged badge looks exactly how it appears in the template, there is no way of knowing if that person is actually authorized.

C. Transmission of Biometric Signatures

There is also a vulnerability that exists when the sensor and feature extractor exchange information, made possible by the iOS's vulnerability to man-in-the-middle attacks [25], a weakness known as the communication link. When the data is captured, the information from the biometric signature is sent from the sensor to the feature extractor. When this exploit is done, the attacker not only has access to the unique signal of the person's fingerprint, but also the elements of the encryption process, such as the matching algorithm. With these in hand, the attacker can then use them on a device at a later time [18].

Cross system risk is another potential vulnerability of biometrics [18]. This is due to a single point of failure within biometrics that subjects a person's account to a single password. An attacker could merely gain access to the victim's fingerprint and subsequently gain access to multiple systems and files that also use that fingerprint to protect them. This is a particularly dangerous weakness; if a person were to encrypt both classified and unclassified information with their fingerprint, an attacker could access the unclassified information and have access to both documents.

D. Brute Force

The final vulnerability biometrics faces is the simple brute force attack. This can be carried out in one of two ways: physically and digitally.

A physical brute force attack on biometrics uses the vulnerability that comes from sensor noise. This is where an attacker may use a flashlight or laser shined on to the sensor. By doing this, the attacker hopes to overload the sensor and then gain access.

A digital brute force attack has two methods of execution. First, this may occur similar to a brute force password crack. This is made possible because biometrics must compare a person's template to a multitude of other templates; there is not a good way to set up a lock out after several bad attempts. This allows an attacker to simply send an unlimited amount of data at the sensor until they eventually get the correct combination [18]. Second, it may be executed by analyzing the device's power consumption when it uses its biometric sensor. This weakness allows an attacker to potentially discover the software code and can also attempt to break the encryption with the information gathered [18].

E. Effects of Vulnerabilities on iOS 7

Once all the vulnerabilities are realized, it is easy to see which ones are going to pose a greater threat to iOS 7 users. First, there is the issue of spoofing, brute force attacks, and residual templates. These are vulnerabilities that affect all biometric devices; however, these pose a higher risk to users of iOS 7 since biometrics is being employed on mobile devices, specifically the smartphone. An attacker must simply steal the victim's phone or device, and they are able to work on it at their own leisure.

Another issue for iOS 7 is the need to store templates on a central server. Since the fingerprint scan on an iPhone will allow a person to gain access to other Apple features, it has effectively created a single point of failure for the user [22]. An attacker can use a cross-system link and fake template vulnerabilities to gain access to all of their victim's information in a single attack. Because of this single point of failure, an attacker may not even need to upload a fake template to the server. This brings up one of the most controversial parts of iOS 7's biometrics. Because Apple will be saving user's biometrics on central servers, all an attacker would need to do is gain access to those servers [20]. Once access is acquired they now have access to the biometric scans of thousands of users. This threat is similar to the 2011 Sony PlayStation Network Hack. Attackers in this scenario only needed to gain access to a few servers and then gained access to all its user data.

VI. PRIVACY CONTROVERSY

A final item to consider in regard to the use of biometrics in iOS 7 is due to recent events with the information leaks at the NSA. Many users fear that since Apple is storing their biometric information, they may be using that data to track their users. Moreover, iOS 7 users fear that Apple may actually be turning this information over to government organizations without their consent. While much of this is speculation at this point, revelations of government activity in using data acquired from phone companies allows the consideration of this idea as reasonable.

It also poses several legal issues surrounding a person self-incriminating themselves by using their cell phones, or the possibility of someone being innocently charged because the biometric saved from their cell phone was similar to a criminal's. Since Apple or the government have not officially addressed this matter, this is a controversy that users can only speculate and debate about.

VII. CONCLUSION

With such a large amount of vulnerabilities, and with both the security and ethical controversy that surrounds the iOS 7, the question must be asked, what is Apple doing to address and fix these problems? The answer to this question is a complicated one. Apple keeps most of their security practices under lock and key. Users rarely get to learn exactly how they are being protected.

Courts attempts to clear up some of the speculation that is revolving around the controversies by pointing out that the company has set up a strict policy keep app developers from utilizing the fingerprint scanner [17]. This shows that Apple is unlikely to be selling personal information to anyone. PC Magazine has also reported that Apple has added a special feature to its Safari browser which will allow users of iOS 7 to turn off tracking, so that Apple cannot track their information.

iOS 7 also implements the iCloud Key Chain. As outlined in Eddy, this is a system designed to protect the user's passwords by encrypting them with a separate 256-bit AES encryption [19]. This, in addition to still utilizing a PIN code, can allow a user to use a fingerprint to protect their phone while using a separate PIN code to protect their passwords to all other services, such as iTunes [22].

In regard to the worry that Apple releases information to government organizations, Reisinger states that Apple is among other heavy-hitting tech companies that are making efforts to strengthen encryptions to prevent access from groups such as the NSA [21]. While there may still be paranoia in the air on the matter, it can be said that Apple is not going to risk its customer base by giving information to the government.

REFERENCES

[1] G. Smith. "iPhone fingerprint scanner comes with a catch." Huffington Post. September 10, 2013. http://www.huffingtonpost.com/2013/09/10/iphone-fingerprint-scanner_n_3900529.html

[2] N. MacDonald. "Apple's iOS 7 is a significant step forward." Gartner. August 14, 2013. http://blogs.gartner.com/neil_macdonald/2013/08/14/apples-ios-7-is-a-significant-step-forward/

[3] Kingsley-Hughes. "The iPhone 5s fingerprint reader: more about convenience than security." ZDNet. October 8, 2013. http://www.zdnet.com/the-iphone-5s-fingerprint-reader-more-about-convenience-than-security-7000021692/

[4] "iOS: a visual history." The Verge. December 13, 2011.

http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

[5] C. Miller; D. Blazakis; D. DaiZovi; S. Esser; V. Iozzo; and R-P. Weinmann. *iOS Hacker's Handbook*. Hoboken, NJ: John Wiley & Sons, 2012.

[6] K. Wadner. *Security Implications of iOS*. SANS Institute, 2011. http://www.sans.org/reading-room/whitepapers/pda/security-implications-ios-33724

[7] "comScore Reports August 2013 U.S. Smartphone Subscriber Market Share." Comscore, October 2013. http://www.comscore.com/Insights/Press_Releases/2013/10/comScore_Reports_August_2013_US_Smartphone_Subscriber_Market_Share

[8] F. Schaub, R. Deyhle, & M. Weber. "Password entry usability and shoulder surfing susceptibility on different smartphone platforms." In Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia (p. 13). ACM. December, 2012.

[9] Kingsley-Hughes. "The iPhone 5s fingerprint reader." ZDNet. (See [3])

[10] Apple, Inc. "iOS Security." White paper. February 2014. http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf

[11] K. Wadner. *Security Implications of iOS*. SANS Institute. (See [6])

[12] R. A. D. E. K. Vala, L. I. B. O. R. Sarga, & R. A. D. E. K. Benda. "Security reverse engineering of mobile operating systems: a summary." In Recent Advances in Computer science. *Proceedings of the 17th International Conference on computers*. Rhodes Island, Greece. July, 2013.

[13] Apple, Inc. *iOS Security*. October 2012. http://www.apple.com/ipad/business/docs/iOS_Security_Oct12.pdf

[14] N. Hindocha; D. Mold; T. Neaves; & L. Neely. *iOS Platform Security*. August 2011. http://www.sans.org/score/checklists/ios-platform-sec.pdf

[15] National Security Agency. *Security Configuration Recommendations for Apple iOS 5 Devices*. March 2012. http://www.nsa.gov/ia/_files/os/applemac/apple_ios_5_guide.pdf

[16] Aresty, Bird, Darrow, Ferrera, Klosek, Lichtenstein, & Reader (2012). Cyber Law: Text & Cases. Mason, OH

[17] Couts, A. (2013, Sep 11). Don't Panic! Apple's Fingerprint Scanner Should Appease Conspiracy Theorist. Retrieved from http://www.digitaltrends.com/mobile/can-apple-hand-over-your-fingerprint-to-the-nsa/

[18] Dimitriadis, C. Polemi, D. (2004). Biometrics Risk and Controls. Retrieved from http://www.isaca.org/Journal/Past-Issues/2004/Volume-4/Pages/Biometrics-Risks-and-Controls.aspx

[19] Eddy, M. (2013, Sep 13). iOS7 makes the iPhone More Secure than Ever. Retrieved from http://www.pcmag.com/article2/0,2817,2424412,00.asp

[20] Miller, C.; Blazakis, D.; DaiZovi, D.; Esser, S.; Iozzo, V.; Weinmann, R-P. (2012) iOS Hacker's Handbook. Hoboken, NJ: John Wiley & Sons.

[21] Reisinger, D. (2013, Nov 15). Battle Brews as Tech Companies Attempt to Fend Off NSA Hacking. Retrieved from http://news.cnet.com/8301-1009_3-57612543-83/battle-brews-as-tech-companies-attempt-to-fend-off-nsa-hacking/

[22] Saran, C. (2013, Sep 12). Apple Tightens Security with iOS7. Retrieved from http://www.computerweekly.com/blogs/inspect-a-gadget/2013/09/apple-tightens-security-with-ios-7.html

[23] Smith, G. (2013). iPhone Fingerprint Scanner Comes With A Catch. Retrieved from http://www.huffingtonpost.com/2013/09/10/iphone-fingerprint-scanner_n_3900529.html

[24] Vala, R. A. D. E. K., Sarga, L. I. B. O. R., & Benda, R. A. D. E. K. (2013, July). Security Reverse Engineering of Mobile Operating Systems: A Summary. In Recent Advances in Computer science. *Proceedings of the 17th International Conference on computers*. Rhodes Island, Greece.

[25] Wadner, K. (2011) Security Implications of iOS. Retrieved from http://www.sans.org/reading-room/whitepapers/pda/security-implications-ios-33724