EILEEN M. DECKER
United States Attorney
PATRICIA A. DONAHUE
Assistant United States Attorney
Chief, National Security Division
TRACY L. WILKISON (California Bar No. 184948)
Chief, Cyber and Intellectual Property Crimes Section
Assistant United States Attorney
1500 United States Courthouse
312 North Spring Street
Los Angeles, California 90012
Telephone: (213) 894-2400
Facsimile: (213) 894-8601
Email: Tracy.Wilkison@usdoj.gov

Attorneys for Applicant
UNITED STATES OF AMERICA

UNITED STATES DISTRICT COURT

FOR THE CENTRAL DISTRICT OF CALIFORNIA

IN THE MATTER OF THE SEARCH OF AN APPLE IPHONE SEIZED DURING THE EXECUTION OF A SEARCH WARRANT ON A BLACK LEXUS IS300, CALIFORNIA LICENSE PLATE #5KGD203	ED No. CM 16-10 (SP) DECLARATION OF TRACY L. WILKISON IN SUPPORT OF GOVERNMENT'S REPLY IN SUPPORT OF MOTION TO COMPEL AND OPPOSITION TO APPLE INC.'S MOTION TO VACATE ORDER; EXHIBITS 1-16 Hearing Date: March 22, 2016 Hearing Time: 1:00 p.m. Location: Courtroom of the Hon. Sheri Pym

DECLARATION OF TRACY L. WILKISON

I, Tracy L. Wilkison, declare as follows:

1. I am an Assistant United States Attorney in the United States Attorney's Office for the Central District of California. I am one of the attorneys who represent the government in the instant matter.

2. Attached hereto as Exhibit 1 are true and correct copies of the article *Apple's Lawyer: If We Lose, It Will Lead to a 'Police State,'* by David Goldman and Laurie Segall, published on February 26, 2016, available at http://money.cnn.com/2016/02/26/technology/ted-olson-apple/index.html, and printed on March 9, 2016; and the article *Tim Cook: FBI Is Asking Apple to Create 'Software Equivalent of Cancer,'* by Mikey Campbell, published on February 24, 2016, available at http://appleinsider.com/articles/16/02/24/tim-cook-fbi-is-asking-apple-to-create-software-equivalent-of-cancer, and printed on March 9, 2016.

3. Attached hereto as Exhibit 2 is a true and correct copy of *U.S. Sec. and Exch. Comm'n Form 10-K Annual Report for Apple Inc.* (filed on Oct. 28, 2015), available at http://investor.apple.com/secfiling.cfm?filingID=1193125-15-356351&CIK=320193.

4. Attached hereto as Exhibit 3 is a true and correct copy of the English Language portions of the Apple Inc. ("Apple") document *iOS 9.0 Software License Agreement for iPhone, iPad, and iPod Touch*, available at http://images.apple.com/legal/sla/docs/iOS9.pdf, and printed on March 5, 2016.

5. Attached hereto as Exhibit 4 is a true and correct copy of Apple document *Terms and Conditions*, available at http://www.apple.com/legal/internet-services/itunes/us/terms.html, and printed on March 5, 2016.

6. Attached hereto as Exhibit 5 is a true and correct copy of the Internet archive stored version of Apple's statement *Our Commitment To Customer Privacy Doesn't Stop Because Of A Government Information Request* (March 31, 2015),

available at http://web.archive.org/web/20150331005807/http:/www.apple.com/privacy/government-information-requests/, and printed on February 29, 2016.

7. Attached hereto as Exhibit 6 is a true and correct copy of the Brief of Appellant, *United States v. Mountain States Telephone & Telegraph Company*, No. CA 78-2366.

8. Attached hereto as Exhibit 7 is a true and correct copy of document *California Government's Budget 2015-16 Enacted Budget Detail*, available at http://www.ebudget.ca.gov/2015-16/Enacted/agencies.html, and printed on March 4, 2016.

9. Attached hereto as Exhibit 8 is a true and correct copy of Apple document *Report on Government Information Requests (January 1–June 30, 2015)*, available at https://www.apple.com/nz/privacy/docs/government-information-requests-20150914.pdf, and printed on March 5, 2016.

10. Attached hereto as Exhibit 9 is a true and correct copy of the article *Apple Adds State-Controlled China Telecom as Data Center Provider*, by Lorraine Luk, published on August 15, 2014, available at http://blogs.wsj.com/digits/2014/08/15/apple-adds-china-telecom-as-data-center-provider/, and printed on March 5, 2016.

11. Attached hereto as Exhibit 10 is a true and correct copy of the article *Apple Tweaks Wi-Fi in iPhone to Use China Protocol*, by Owen Fletcher, available at http://www.pcworld.com/article/195524/article.html, and printed on March 9, 2016.

12. Attached hereto as Exhibit 11 is a true and correct copy of the report *Cyber Security in China: Internet Security, Protectionism and Competitiveness: New Challenges to Western Businesses*, by Hauke Johannes Gierow, Issue 22 China Monitor (published April 22, 2015), available at http://www.merics.org/fileadmin/templates/download/china-monitor/150407_MERICS_China_Monitor_22_en.pdf, and printed on March 9, 2016.

13. Attached hereto as Exhibit 12 is a true and correct copy of the article *While It Defies U.S. Government, Apple Abides By China's Orders—and Reaps Big Rewards*,

by David Pierson, published on February 26, 2016, available at http://www.latimes.com/business/technology/la-fi-apple-china-20160226-story.html, and printed on March 9, 2016.

14. Attached hereto as Exhibit 13 is a true and correct copy of Apple document *Legal Process Guidelines U.S. Law Enforcement*, published September 29, 2105, available at http://www.apple.com/privacy/docs/legal-process-guidelines-us.pdf, and printed on March 5, 2016.

15. Attached hereto as Exhibit 14 is a true and correct copy of Apple statement *Answers To Your Questions About Apple And Security*, available at http://www.apple.com/customer-letter/answers/, and printed on February 28, 2016.

16. Attached hereto as Exhibit 15 is a true and correct copy of the article *Apple Is Said to Be Trying to Make it Harder to Hack iPhones*, by Matt Apuzzo and Katie Benner, version published on February 24, 2016, available at http://www.nytimes.com/2016/02/25/technology/apple-is-said-to-be-working-on-an-iphone-even-it-cant-hack.html?_r=0, and printed on February 29, 2016.

17. Attached hereto as Exhibit 16 is a true and correct copy of Apple's Response to Court's October 9, 2015 Memorandum and Order, In Re Order Requiring Apple Inc. To Assist In The Execution Of A Search Warrant Issued By This Court, No. 15-MC-1902 (E.D.N.Y. Oct. 19, 2015).

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that this declaration is executed in Los Angeles, California, on March 9, 2016.

Tracy L. Wilkison
Assistant United States Attorney

EILEEN M. DECKER
United States Attorney
PATRICIA A. DONAHUE
Assistant United States Attorney
Chief, National Security Division
TRACY L. WILKISON (California Bar No. 184948)
Chief, Cyber and Intellectual Property Crimes Section
Assistant United States Attorney
1500 United States Courthouse
312 North Spring Street
Los Angeles, California 90012
Telephone: (213) 894-2400
Facsimile: (213) 894-8601
Email: Tracy.Wilkison@usdoj.gov

Attorneys for Applicant
UNITED STATES OF AMERICA

UNITED STATES DISTRICT COURT

FOR THE CENTRAL DISTRICT OF CALIFORNIA

IN THE MATTER OF THE SEARCH OF AN APPLE IPHONE SEIZED DURING THE EXECUTION OF A SEARCH WARRANT ON A BLACK LEXUS IS300, CALIFORNIA LICENSE PLATE #5KGD203	ED No. CM 16-10 (SP) SUPPLEMENTAL DECLARATION OF CHRISTOPHER PLUHAR IN SUPPORT OF GOVERNMENT'S REPLY IN SUPPORT OF MOTION TO COMPEL AND OPPOSITION TO APPLE INC.'S MOTION TO VACATE ORDER Hearing Date: March 22, 2016 Hearing Time: 1:00 p.m. Location: Courtroom of the Hon. Sheri Pym

SUPPLEMENTAL DECLARATION OF CHRISTOPHER PLUHAR

I, Christopher Pluhar, declare and state as follows:

1. I am a Supervisory Special Agent ("SSA") with the FBI, and I have knowledge of the facts set forth herein and could and would testify to those facts fully and truthfully if called and sworn as a witness.

A. The Subject Device Was Off When Seized

2. In paragraph 8 of my declaration dated February 16, 2016 (the "Initial Declaration"), I explained that the Subject Device was "locked" because it presented a numerical keypad with a prompt for four digits. To add further detail, on December 3, 2015, the same day the Subject Device was seized from the Lexus IS300, I supervised my Orange County Regional Computer Forensics Laboratory ("OCRCFL") team who performed the initial triage of the Subject Device, and observed that the device was powered off, and had to be powered up, or booted, to conduct the triage. Upon power-up, we observed that the device was protected with a four-digit passcode (because it displayed a number pad with four spaces), and was running iOS9. I confirmed with two FBI Evidence Response Team agents that the device was found in the center console of the Lexus IS300 described in the search warrant, and that it was found there powered off.

B. Accessing the iCloud Back-Ups

3. As described in paragraphs 5 and 6 of my Initial Declaration, after the shootout on December 2, 2016, the Subject Device was seized pursuant to the search warrant on December 3, 2016. After case agents and forensic examiners from the OCRCFL met with personnel (including Information Technology ("IT") personnel) from the San Bernardino County Department of Public Health ("SBCDPH"), I then met personally on December 6, 2015 with IT specialists at the SBCDPH to gather more information about the Subject Device and the SBCDPH account(s) associated with the Subject Device. I learned from SBCDPH personnel that the department had deployed a mobile device management ("MDM") system to manage its recently issued fleet of iPhones, that the MDM system had not yet been fully implemented, and that the

necessary MDM iOS application to provide remote administrative access had not been installed on the Subject Device. As a result, SBCDPH was not able to provide a method to gain physical access to the Subject Device without Farook's passcode.

4. As described in paragraph 7 of my Initial Declaration, the Subject Device is owned by SBCDPH. I learned from SBCDPH IT personnel that SBCDPH also owned the iCloud account associated with the Subject Device, that SBCDPH did not have the current user password associated with the iCloud account, but that SBCDPH did have the ability to reset the iCloud account password.

5. Without the Subject Device's passcode to gain access to the data on the Subject Device, accessing the information stored in the iCloud account associated with the Subject Device was the best and most expedient option to obtain at least some data associated with the Subject Device. With control of the iCloud account, the iCloud back-ups of the Subject Device could be restored onto different, exemplar iPhones, which could then be processed and analyzed.

a. As described in Apple's security documentation, a "passcode" is a component of the encryption key that protects the device itself, which is distinct from the "password" associated with an Apple ID needed to access Apple's Internet Services, such as iCloud. *See* Apple's iOS Security for iOS 9.0 (Sept. 2015) ("iOS Security") attached to the Declaration of Nicola T. Hanna as Exhibit K; *id.* at 11-12 (describing passcode's role in creating device's class key); *id.* at 38 (describing different password requirements for Apple ID needed for Apple's Internet Services); *id.* at 41 ("Users set up iCloud by signing in with an Apple ID"). Each iCloud account is associated with a specific Apple ID.

b. Therefore the pass*word* necessary to access the iCloud account associated with the Subject Device is unrelated to the pass*code* needed for physical access to the Subject Device itself.

6. While in discussions with SBCDPH IT personnel, I also spoke with Lisa Olle, attorney for Apple Inc. Ms. Olle provided me various pieces of useful information

about the iCloud account associated with the Subject Device, including information about the existing back-ups, confirmation that the entire iCloud account had already been preserved by Apple in response to an FBI request for preservation, and that the remote-wipe function was not activated for the Subject Device. Ms. Olle advised that once the search warrant was received by Apple, there would be an unknown time delay for Apple to provide the Subject Device iCloud account data.

7. After that conversation with Ms. Olle, and after discussions with my colleagues, on December 6, 2015, SBCDPH IT personnel, under my direction, changed the password to the iCloud account that had been linked to the Subject Device. Once that was complete, SBCDPH provided exemplar iPhones that were used as restore targets for two iCloud back-ups in the Subject Device's iCloud account. Changing the iCloud password allowed the FBI and SBCDPH IT to restore the contents of the oldest and most recent back-ups of the Subject Device to the exemplar iPhones on December 6, 2015. Once back-ups were restored, OCRCFL examiners processed the exemplar iPhones and provided the extracted data to the investigative team. Because not all of the data on an iPhone is captured in an iCloud back-up (as discussed further below), the exemplar iPhones contained only that subset of data as previously backed-up from the Subject Device to the iCloud account, not all data that would be available by extracting data directly from the Subject Device (a "physical device extraction").

C. Not All Data on an iPhone is Backed Up to the iCloud

8. Subsequently, a search warrant was issued on January 22, 2016, to obtain the preserved contents of the Apple ID and iCloud account associated with the Subject Device. Review of the iCloud search warrant results that were received from Apple on January 26, 2016 is ongoing, but review of this data is difficult compared to the data restored to the exemplar iPhones due to the manner in which it has been formatted and delivered by Apple.

9. The results of the iCloud search warrant confirm that the last Subject Device back-up to the iCloud account was on October 19, 2015 (approximately 6 weeks

before the December 2, 2015 attack in San Bernardino), as stated in paragraph 8 of my Initial Declaration. According to the logs contained in those results, on October 22, 2015, it appears that the "iForgot" web-based password change feature was used for the account associated with the Subject Device. I know based on my experience, and review of Apple's website, that "iforgot.apple.com" provides iCloud customers with the ability to reset the password associated with their iCloud account over the Internet.

10. Regarding iCloud back-ups, I know from training and experience as a mobile device forensic examiner, and consultation with other FBI technical experts that, in general, cloud-based back-ups of physical devices contain only a subset of the data that is typically obtained through physical device extractions.

a. For example, with iCloud back-ups of iOS devices (such as iPhones or iPads), device-level data, such as the device keyboard cache, typically does not get included in iCloud back-ups but can be obtained through extraction of data from the physical device. The keyboard cache, as one example, contains a list of recent keystrokes typed by the user on the touchscreen. From my training and my own experience, I know that data found in such areas can be critical to investigations.

b. I also know that the Apple iOS allows users to change settings on the device to exclude certain apps from including their user data in iCloud back-ups, but the user data associated with apps excluded from iCloud back-ups by the user may still be obtained via physical device extraction.[1] I consulted with an OCRCFL examiner who reviewed the exemplar iPhones that were used as restore targets for the iCloud back-ups of the Subject Device. Each of the restored exemplars includes restored settings, and those settings showed that, for example, iCloud back-ups for "Mail," "Photos," and "Notes" were all turned off on the Subject Device.

11. For these reasons, iCloud back-ups as currently implemented are not

[1] I also know that developers of iOS apps have the ability to design their apps to specifically exclude app user data from iCloud back-ups, but the user data associated with those apps may still be obtained via physical device extraction.

considered a comprehensive method of extracting all available stored data from an iOS device. For iOS devices, as well as other mobile device platforms, back-ups such as those made to iCloud can provide valuable evidence, but forensic examiners rely on physical device extractions to obtain the most data available from mobile devices. Therefore, even if it had been possible, via any means, to initiate a fresh iCloud back-up of the Subject Device, so that it included information through December 2, 2015, the FBI would still need to conduct a physical device extraction of the Subject Device in order to obtain all potential evidence from the Subject Device.

12. Before seeking the February 16, 2016, Order, in a phone conversation of which I was a part, the government explained to Lisa Olle and Erik Neuenschwander, among others from Apple, in detail its proposal for technical assistance including specifics of the three desired functions and how they might be achieved as embodied in the Order. After hearing the government's proposal, the Apple representatives declined to discuss the feasibility of the government's proposal and instead provided a list of alternative ways the government might be able to access some of the data on the Subject Device. Although the FBI had already explored these avenues, I, and others from the technical team re-explored them at the suggestion of Apple representatives. We again determined that none provided any means to access the full set of data on the Subject Device. In a subsequent phone conversation with Erik Neuenschwander and Lisa Olle, we explained that the alternatives they had suggested did not work. Erik Neuenschwander and Lisa Olle declined to discuss the feasibility of the government's proposal as embodied in the Order.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that this declaration is executed at California, on March 9, 2016.

Christopher Pluhar
Supervisory Special Agent
Federal Bureau of Investigation

EILEEN M. DECKER
United States Attorney
PATRICIA A. DONAHUE
Assistant United States Attorney
Chief, National Security Division
TRACY L. WILKISON (California Bar No. 184948)
Chief, Cyber and Intellectual Property Crimes Section
Assistant United States Attorney
1500 United States Courthouse
312 North Spring Street
Los Angeles, California 90012
Telephone: (213) 894-2400
Facsimile: (213) 894-8601
Email: Tracy.Wilkison@usdoj.gov

Attorneys for Applicant
UNITED STATES OF AMERICA

UNITED STATES DISTRICT COURT

FOR THE CENTRAL DISTRICT OF CALIFORNIA

IN THE MATTER OF THE SEARCH OF AN APPLE IPHONE SEIZED DURING THE EXECUTION OF A SEARCH WARRANT ON A BLACK LEXUS IS300, CALIFORNIA LICENSE PLATE #5KGD203	ED No. CM 16-10 (SP) DECLARATION OF STACEY PERINO IN SUPPORT OF GOVERNMENT'S REPLY IN SUPPORT OF MOTION TO COMPEL AND OPPOSITION TO APPLE INC.'S MOTION TO VACATE ORDER; EXHIBITS 17-30 Hearing Date: March 22, 2016 Hearing Time: 1:00 p.m. Location: Courtroom of the Hon. Sheri Pym

DECLARATION OF STACEY PERINO

I, Stacey Perino, declare as follows:

1. I am an Electronics Engineer with the Federal Bureau of Investigation ("FBI"). I have knowledge of the facts set forth herein and could and would testify to those facts fully and truthfully if called and sworn as a witness.

2. I received a Bachelor of Science in Mechanical Engineering from Colorado State University in 1991. I received a Bachelor of Science Degree in Electrical Engineering from the University of Colorado in 1996. I have been employed as an Electronics Engineer with the Federal Bureau of Investigation since 1996. From 1996 to 2001, I was an electrical engineer in the FBI's Cryptologic and Electronic Analysis Unit ("CEAU") developing both hardware and software solutions to recover data from electronic devices. From 2001 to 2009, I was the Program Manager for the Embedded Engineering Program within that same unit in the FBI. During this time I managed the technical efforts for a team of electrical engineers, computer engineers and computer scientists, composed of both government and contractor personnel with a focus on the recovery and presentation of data from electronic devices. In 2009, I became the Technical Director of the CEAU, a position I still hold.

3. This declaration is made in support of an application seeking an order from the Court compelling Apple Inc. ("Apple") to assist the FBI in its effort to search a cellular telephone, Apple make: iPhone 5C, Model: A1532, P/N:MGFG2LL/A, S/N:FFMNQ3MTG2DJ, IMEI:358820052301412, on the Verizon Network ("Subject Device").

4. In addition to relying on my own education, training, and experience, in preparing this declaration, I have reviewed the following:

a. The Declarations of Christopher Pluhar dated February 16, 2016 ("Initial Pluhar Declaration") and March 9, 2016, the Application filed in Case No. 16-

10 in the Central District of California, and the Court's Order in the same case calling for a software image file or "SIF" to be prepared by Apple (the "Order").

b. The Declaration of Erik Neuenschwander dated February 25, 2016 ("Neuenschwander Declaration").

c. Apple's "iOS Security" for iOS 9.0 or later dated September 2015 ("iOS Security"), attached to the Declaration of Nicola T. Hanna as Exhibit K.

d. Documentation from the website of the information technology company Sogeti, attached hereto as Exhibit 17, available at http://esec-lab.sogeti.com/static/publications/11-hitbamsterdam-iphonedataprotection.pdf.

e. The repository of code stored at https://code.google.com/archive/p/iphone-dataprotection, described as "ios forensics tools," and "Tools and information on iOS 3/4/5/6/7 data protection features."

f. Cellebrite Physical Extraction Manual for iPhone & iPad (Rev 1.3), attached hereto as Exhibit 18.

g. Apple's "Cryptographic Services," attached hereto as Exhibit 19, available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/Security_Overview/CryptographicServices/CryptographicServices.html.

h. Materials from Apple's "Code Signing Guide":

i. Exhibit 20, "About Code Signing," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html.

ii. Exhibit 21, "Code Signing Overview," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/AboutCS/AboutCS.html.

iii. Exhibit 22, "Code Signing Tasks," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/Procedures/Procedures.html.

iv. Exhibit 23, "Code Signing Requirement Language," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/CodeSigningGuide/RequirementLang/RequirementLang.html.

i. Materials from Apple's "Cryptographic Services Guide":

i. Exhibit 24, "About Cryptographic Services," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/Introduction/Introduction.html.

ii. Exhibit 25, "Cryptography Concepts In Depth," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/CryptographyConcepts/CryptographyConcepts.html.

iii. Exhibit 26, "Encrypting and Hashing Data," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/GeneralPurposeCrypto/GeneralPurposeCrypto.html.

iv. Exhibit 27, "Managing Keys, Certificates, and Passwords," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/KeyManagementAPIs/KeyManagementAPIs.html.

v. Exhibit 28, "Glossary," available at https://developer.apple.com/library/mac/documentation/Security/Conceptual/cryptoservices/Glossary/Glossary.html.

j. Apple's "Unauthorized Modification of iOS Can Cause Security Vulnerabilities, Instability, Shortened Battery Life, and Other Issues," attached hereto as Exhibit 29, and available at https://support.apple.com/en-us/HT201954.

k. Apple's "Code Signing," attached hereto as Exhibit 30, and available at https://developer.apple.com/support/code-signing/.

5. This Declaration relies on Apple's publicly disseminated descriptions of how its own devices, operating system, security features, and software operate. Apple's source code is not, however, publicly available. Therefore the descriptions below do not

rely on my having reviewed Apple's source code, rather they rely upon Apple's own description of its devices, operating system, security features, and software, as well as on my training and experience in both observing and/or conducting the tests described in this document, directing the CEAU embedded engineering analysis of Apple devices and software, and reviewing other open source materials describing Apple mobile device technologies.

A. Purpose of this Declaration

6. In this declaration, I discuss the following topics:

a. The SIF called for in the Order could run *only* on the Subject Device. To explain this, I first provide some background on public key cryptography (Part B.1) and Apple's use of it and code signing to prevent the use of unauthorized code on its products (Part B.2). The Order provides that the SIF would only run on the Subject Device. Apple already requires that iOS updates include a unique device identifier for the Subject Device (Part B.3). Because an iPhone requires Apple to have cryptographically "signed" code before an iPhone will run it, and changing a unique device identifier within the SIF would invalidate Apple's signature, the SIF would not run on other iPhones. (Part B.3.)

b. The SIF called for by the Court's Order would perform functions that already exist in open source software for older devices and operating systems. In other words, code already exists that will bypass the auto-erase and time-delay functions and permit electronic submission of passcodes, but would need to be updated and modified for newer operating systems. (Part C.) That software, however, cannot run on the Subject Device without Apple's "signature."

c. The data contained on the Subject Device can be decrypted *only* on the Subject Device. This is because the encryption key includes a unique identifier that exists only on the Subject Device. (Part D.) Because the decryption must occur on the Subject Device, and because only Apple-signed software can run on the Subject Device

(Part B.2), any code or software tools needed to assist in testing passcodes (even code that includes components that already exist, Part C) must be signed by Apple.

d. Because the Subject Device was powered off when it was seized, it was not possible for it to back itself up to iCloud without the passcode. (Part E.)

B. The SIF Called for by the Order Would Run Only on the Subject Device

1. General Background on Public Key Cryptography

7. Generally, encryption and decryption are the processes of first converting intelligible "plaintext" into unintelligible "ciphertext," and second converting the ciphertext back into plaintext, respectively.

8. While encryption is designed to protect the confidentiality of information, a separate issue that arises in cryptology is authentication. Public key cryptography provides a method to both send messages securely, even when using a non-secure channel, and to validate the messages that are received. A properly implemented cryptographic signature gives the receiver reason to believe the message was sent by the person claiming to be the sender. A cryptographic signature also prevents modification of the original message by anyone other than the signer.

9. Public key encryption uses a complex operation that involves two different keys, a public key and a private key. A public key cryptosystem uses one key to encrypt (or to sign) a message and a different key to decrypt (or verify) the same message. (For this reason it is also referred to as asymmetric.) One of the essential properties of a public key cryptosystem is that it is too difficult—computationally infeasible—to determine a person's private key knowing only that person's public key.

10. The public key is made globally available while the private key is kept confidential. This allows anyone who is a member of the system to use the "phone book" of public keys to send a private message to any other member using the recipient's public key, but it allows only the recipient to open it using that person's private key.

Each key pair is unique to an individual member of a properly implemented cryptosystem.

11. One of the other essential properties of a public key cryptosystem is that the encryption operation and the decryption operation used in the cryptosystem are inverse operations.[1] This means that if one started with a message, it would not matter if one used the encryption operation followed by the decryption operation, or the decryption operation followed by the encryption operation, either would yield the original message again.[2]

12. A more detailed example of how the public key cryptosystem works to sign a message is as follows:

a. Alice generates a public-private key pair, and publishes her public key.

b. Alice composes a Message to Bob, and uses her private key to compute or generate the Signature. (This is represented: Signature = D_{pri}(Message) , where D is the decryption operation.)

c. Alice sends Bob both the Message and the Signature.

d. Bob then uses Alice's public key to verify that the message was signed using her private key. Bob does this by running the inverse "encryption" operation on the Signature. (This is represented: E_{pub}(Signature) = Message, where E is the encryption operation.) If the result of that operation is the Message that Alice sent Bob, then Bob knows the message is not a forgery and came from Alice.

[1] This is represented as follows, where E() and D() denote the encryption and decryption operations, and M is the text of the message: M = E(D(M)) = D(E(M)).

[2] (*See* Ex. 19 at 2, diagram (Apple developer website, Cryptographic Services). *See generally* Ex. 25 at 5, 7 (Apple developer website, Cryptographic Concepts in Depth).)

e. In this example, Alice could also have encrypted the message using Bob's public key. Bob could then have decrypted the message using Bob's own private key.

2. Apple's Use of Public Key Encryption to Prevent the Use of Unauthorized Code on Its Products

13. Just as a cryptosystem can be used to "sign" messages, it can be used to "sign" executable code.[3] Specifically, a vendor can embed a public key into a device such that the public key cannot be altered. For any and all executable code modules, the vendor uses its private key to calculate and attach a signature. As the device loads code modules for execution, the device uses the embedded public key to calculate the signature and thus verify the module's integrity and authenticity. As long as the public key cryptosystem is unbroken and the embedded key cannot be modified within the device, the scheme guarantees that only code issued by the vendor (that has been cryptographically signed) will run on the device.

14. Apple implements this system to require that its devices use software that only Apple authorizes. Apple does this by programming the public key into Read Only Memory ("ROM"). ROM is hardwired during the manufacture of the semiconductor device and cannot be changed later through any software means. The firmware in ROM is the first code that executes on the processor when power is applied. According to Apple's Security documentation, Apple products have also stored "the Apple Root CA [certificate authority] public key" within boot ROM.[4] (iOS Security at 5.) The boot ROM code uses the public key to verify that the next code to load (which is stored in

---

[3] In simplified terms, software is generally written by programmers in "source code." That source code is converted (or "compiled") into what is referred to as "executable code" that is in a format that a computer processer can understand and "execute."

[4] Boot ROM is firmware that has been fused or hardwired into the processor during manufacturing. It cannot be changed.

memory outside the processor) has also been signed with Apple's private key. (iOS Security at 5.)

15. This system ensures that Apple controls all code loaded and run on the device from the initial power-on. Apple describes how it has implemented this process in what it refers to as its "Chain of Trust" on pages 5-10 of its iOS Security document, wherein each sequential step needed to boot up the operating system and run application software relies on—and requires—Apple's signature. Specific details include the following:

a. "Each step of the startup process contains components that are cryptographically signed by Apple to ensure integrity and that proceed only after verifying the chain of trust. This includes the bootloaders, kernel, kernel extensions, and baseband firmware." (*Id.* at 5.)[5]

b. "The Boot ROM code contains the Apple Root CA [certificate authority] public key, which is used to verify that the Low-Level Bootloader (LLB) is signed by Apple before allowing it to load. This is the first step in the chain of trust where each step ensures that the next is signed by Apple. When the LLB finishes its tasks, it verifies and runs the next-stage bootloader, iBoot, which in turn verifies and runs the iOS kernel." (*Id.*) A certificate authority is the entity that issues digital

[5] A bootloader is the initial code run on a processor that starts the system's hardware components and peripherals and prepares the hardware for the operating system or higher level code. There may be multiple bootloaders that are executed sequentially at startup. The kernel is the first part of an operating system that loads and is responsible for controlling access to the computer's hardware resources. The kernel generally runs in protected memory to which other parts of the operating system and application code cannot directly read or write. Kernel extensions provide a method for adding or changing functionality of Apple's kernel without recompiling/relinking the source code. A mobile device typically has multiple processors; the application processor running an operating system, such as iOS 9.02, with which the user interacts (via the screen and keyboard), and the baseband processor which handles network communications traffic and protocols. The application processor is responsible for starting (booting) the baseband processor. Therefore, the application processor provides the baseband processor with the code it needs to load and run. Thus, Apple's chain of trust calls for each of these steps to be verified, ensuring that the next steps are authorized by Apple before allowing them to run or execute.

certificates. Certificate authorities create the public/private key pairs, and are responsible for ensuring the security of the private key. Apple has built its own certificate authority and has created its own public/private key pair used in the iPhone. As noted above, the public key is permanently programmed into the ROM of the iPhone, while the private key is controlled and protected by Apple. Because only Apple possesses its private key, only Apple is able to sign software that will be loaded on its devices. By keeping the private key secret, Apple ensures that only software signed by Apple using its private key can be loaded on its devices during the boot process.

c. "This secure boot chain helps ensure that the lowest levels of software are not tampered with and allows iOS to run only on validated Apple devices." (*Id.*) "From initial boot-up to iOS software updates to third-party apps, each step is analyzed and vetted to help ensure that the hardware and software are performing optimally together and using resources properly." (*Id.*)

d. "This architecture is central to security in iOS, and never gets in the way of device usability. The tight integration of hardware and software on iOS devices ensures that each component of the system is trusted, and validates the system as a whole." (*Id.*)

16. "The startup process described above helps ensure that only Apple-signed code can be installed on a device." (*Id.* at 6.) If any component can be made to load code not signed by Apple, the chain of trust is broken. By beginning their chain of trust with the initial code and public key programmed into the device ROM, Apple has made it extremely difficult for anyone to defeat the chain of trust.

17. As a result of these features, an Apple iPhone is designed to only run code (the operating system and the many pieces of firmware and software that may operate within it) that are signed using Apple's keys.

3. Apple "Signs" iOS Updates for Its iPhones that Include a Unique Device Identifier, Ensuring It Only Works on One iPhone

18. While the features described above permit Apple to ensure that the devices it manufactures will use only an operating system or software that Apple has authorized (by signing it), Apple also relies on them to ensure that an operating system will work only on one specific Apple device. Specifically, during an iOS update, recovery, or Device Firmware Update (DFU) process, the device verifies that the code being loaded to it was digitally signed specifically for that device, and not for another device. This feature, enforced by the hardware-based chain of trust, allows Apple to ensure that any code loaded to the phone will only operate on a specific device.

19. Apple implements this process in the following manner. First, the device connects to a computer, for example through iTunes, and provides iTunes with unique information about itself—both its hardware and software. Second, iTunes sends this information from the device to an Apple server that builds the package of code needed to update or recover that device, packages it with the same unique information about the device, and returns it to the computer running iTunes. Third, upon receiving that package from the computer running iTunes, the device is required to read and recognize its own unique information before installing the operating system.

20. Details of this process are as follows:

a. Apple maintains what it refers to as "the Apple installation authorization server," which is referred to herein as the "Installation Server." (iOS Security at 6.)

b. Whenever a device tries to upgrade its version of iOS, through the upgrade or recovery process, the device must first send to that server a set of information from the device. The information sent by the device includes "cryptographic measurements for each part of the bundle to be installed (for example, LLB, iBoot, the kernel, and OS image)." (*Id.*) Those measurements are a digest or partial digest of that component. (A digest can be a cryptographic hash, or the result of a similar algorithm

that generates a unique value, akin to a digital fingerprint, after it processes each part of the bundle.)[6] The device also sends a "nonce," or a random, one-time-use value.

c. Most importantly for ensuring the "personalization" of the software for use on a specific device, the device also sends "the device's unique ID (ECID)." (iOS Security at 6.) The ECID is a unique, device-specific identifier programmed into the phone hardware during manufacture. (*Id.* at 58 (defining ECID as "[a] 64-bit identifier that's unique to the processor in each iOS device. Used as part of the personalization process, it's not considered a secret").) Apple explains the use of these values in their iOS Security document. "These steps ensure that the authorization is for a specific device and that an old iOS version from one device can't be copied to another." (*Id.* at 6.)

d. Once the Apple Installation Server receives this information from the device, it builds a software package and digitally signs it using a private key that is not known to the public. The digital signature includes the ECID, nonce, and other cryptographic measurements in the signed data. Once the device receives the package, the device verifies from the signed data that the package is meant for it.

e. The device is also able to tell that the installation is current and is not a repeat of an older installation (which would result in a "downgrade" of the operating system). The device does so by checking the random, one-time nonce it had sent to the server was the one returned by the server in the signed package. "The nonce prevents an

[6] "In cryptography, hashes are used when verifying the authenticity of a piece of data. Cryptographic hashing algorithms are essentially a form of (extremely) lossy data compression, but they are specifically designed so that two similar pieces of data are unlikely to hash to the same value. . . . . With good hashing algorithms, collisions [messages that hash to the same value] are unlikely if you make small changes to a piece of data. This tamper-resistant nature of good hashes makes them a key component in code signing, message signing, and various other tamper detection schemes." (Ex. 19 at 3 (Apple developer website, Cryptographic Services).) By way of background, data compression that is "lossy" loses some qualities of the original data, such as when a compressed digital image loses resolution or appears "pixelated." In the cryptography context, what is important is that the resulting hash value is unique, not that it be capable of reformulating the entire original piece of data, hence it "loses" data by being reduced to a small but unique string of letters and numbers.

attacker from saving the server's response and using it to tamper with a device or otherwise alter the system software." (*Id.* at 6.)

21. The digital signature prevents any part of the returned package from being changed. If the software in the returned package is altered, the digital signature check will fail and the device will not load it. If the ECID is changed to that of another device, the signature check will fail and the device will not load the code.[7] In other words, unless someone can bypass the digital signature verification, allowing them to load unsigned code, the software cannot be changed to operate on a different device or perform a different function.[8]

22. The Order provides that the SIF would only run on the Subject Device. As shown in the preceding description of Apple's normal code signing process during an iOS update, Apple already has a mechanism in place to do this by including the ECID into the digital signature process. If this same or a similar process were used, the SIF could incorporate the ECID of the Subject Device, and then be signed by Apple. In that case, if the ECID of the SIF were changed to the ECID of another device, the signature check would fail and an Apple device would not load the code.[9]

---

[7] As described on Apple's developer website: "When a piece of code has been signed, it is possible to determine reliably whether the code has been modified by someone other than the signer." (Ex. 21 at 1 (Apple developer website, Code Signing Overview).) Among the purposes of code signing are to "ensure that a piece of code has not been altered," and to "identify code as coming from a specific source (a developer or signer)." (*Id.*)

[8] Because of the significance of the ability to digitally sign code and therefore cryptographically authenticate it, Apple's developer website explains that a "signing identity, no matter how obtained, is completely compromised if it is ever out of the physical control of whoever is authorized to sign code." (Ex. 22 at 2 (Apple developer website, Code Signing Tasks).)

[9] An additional measure to ensure the SIF would only run on the Subject Device could be to program the Subject Device's ECID directly into the software running in the SIF. In this scenario, the SIF would read the ECID of the device on which it was running, and compare that to the ECID of the Subject Device that had been programmed into it; if the two did not match, the software would exit. In other words, while the iOS update scenario described in this Part relies on the *device's* refusal to run the code without a valid Apple signature (which signature would be invalid by changing the ECID), the *SIF* could refuse to fully execute if it did not detect the Subject Device's

(footnote cont'd on next page)

23. For these reasons, the SIF called for by the Order would be permitted to run only on the Subject Device. In other words, the creation of the SIF, tailored and signed with the unique identifier of the Subject Device, would not undermine the security of other iPhones that also require Apple-signed code, because each iPhone has its own unique identifier. The SIF proposed by the Order would therefore not break Apple's chain of trust on its iPhones, or even on the Subject Device; Apple's assistance will keep the chain of trust intact.

24. Importantly, if somebody were to bypass the Apple digital signature process, the chain of trust would be broken. Causing an Apple device to allow itself to run software not signed by Apple is referred to as "jailbreaking" the device. Jailbreaks result from bugs or errors in different programs that can be exploited to run unsigned code on a device. To my knowledge, for the iPhone 5C, jailbreaks have been exclusively performed from a powered-on phone on which the passcode has been entered and the phone unlocked. Thus there are currently no published jailbreaks for an iPhone 5C where the passcode has not been entered at least once since powering on, and hence there are none that could be applied to the Subject Device.

C. Software Already Exists that Performs Similar Functions as the SIF

25. The security features created and implemented by Apple that are described above were challenged by researchers and hackers as previous iterations of iOS were released. Apple's current chain of trust structure has fixed previous issues, but the methods that have been published and used to test earlier versions of iPhones illustrate why the components used in the SIF already exist, and why it, like other previous tools, can be operated from random access memory ("RAM").

26. Paragraph 19 of the Neuenschwander Declaration states that Apple's "current iPhone operating systems designed for consumer interaction do not run in

---

ECID. This example is designed to illustrate that there is more than one way to cause the SIF to only load and execute on the Subject Device.

RAM, but are installed on the device itself. To make them run in RAM, Apple would have to make substantial reductions in the size and complexity of the code." As the discussion below illustrates, the SIF would not be designed for "consumer interaction." Rather, the SIF would be designed only to test passcodes, and other similar tools that have previously been used for this purpose do run in RAM.

27. Those previous tools that are available cannot be used on the Subject Device because they are not signed by Apple, and the current chain of trust on the Subject Device requires Apple to have signed any software that will be allowed to run.

28. A more detailed description is as follows:

a. A previous bug allowed a cold-booted[10] iPhone to load a "minimal" operating system in memory (RAMdisk) that had not been signed by Apple. Previously, Apple iPhone versions 3GS and 4 contained a bug in the Apple boot ROM that allowed unsigned code to be loaded and run through Recovery or DFU mode. This vulnerability was published as the "limera1n" exploit. Other researchers analyzed the Apple boot process and published details of it, including the composition of the RAMdisk (*i.e.*, which software components were bundled into the RAMdisk) used in the Recovery mode and DFU mode process to update device firmware.

b. A passcode-recovery tool has already been developed that uses brute-force techniques. The information technology company Sogeti[11] analyzed Apple's encryption process demonstrating that any passcode "guessing" had to be performed by code running on the device and could not be done externally (further explained below in Part D). Other vulnerability researchers used this result to develop software that could brute force the passcode on a jailbroken device (iphone-dataprotection project[12]).

---

[10] Cold-boot refers to a phone that has been powered off and then powered back on but no passcode has been entered.

[11] (Ex. 17 (http://esec-lab.sogeti.com/static/publications/11-hitbamsterdam-iphonedataprotection.pdf).)

[12] (https://code.google.com/archive/p/iphone-dataprotection, "ios forensics tools," and "Tools and information on iOS 3/4/5/6/7 data protection features.")

c. From this open source research, several forensic tools were developed that combined (1) the boot ROM code signing defeat, and (2) brute-force passcode guessing. Examples include the Cellebrite UFED tool and an FBI-developed tool. Both the Cellebrite[13] and FBI tools utilize the boot ROM exploit, allowing iPhone 3GS and iPhone 4 devices to load and boot an unsigned RAMdisk containing code to brute force the device passcode. The passcode recovery process operated from RAM, and did not alter the system or user data area. The passcode recovery software did not require user interaction, and the entire process ran without use of the "Springboard" graphical user interface. Because these forensic tools ran from a RAMdisk and did not use the operating system that was stored on the device, these tools did not incur time delays or the auto-erase function (which are features implemented by the operating system installed on the device).

d. Apple addressed the bug, and subsequently a jailbreak (*i.e.*, allowing code unsigned by Apple) could only occur on an iPhone after it had been booted and unlocked. As described previously, a jailbroken phone is one that has had the chain of trust broken and can run unsigned code.[14] After Apple corrected the bug present in the

[13] Cellebrite is a private company that makes forensic data recovery tools for mobile devices. While I have not examined the source code for the UFED tool, based on the Cellebrite Physical Extraction Manual for iPhone and iPad (Rev 1.3) and the fact that the Cellebrite tool no longer supports iPhone 4S and later devices, I believe the UFED tool relied on the same ROM exploit. The manual states: "The extraction application does not load iOS but instead loads a special forensic utility to the device. This utility is loaded to the device's memory (RAM) and runs directly from there." The utility is loaded from recovery mode.

[14] The use of jailbroken phones discussed in this Part occurred in a testing environment. Outside of a testing environment, some users have jailbroken their phones to try to use software or services that Apple has not authorized, but Apple cautions that doing so presents "[s]ecurity vulnerabilities": "Jailbreaking your device eliminates security layers designed to protect your personal information and your iOS device. With this security removed from your iOS device, hackers may steal your personal information, damage your device, attack your network, or introduce malware, spyware or viruses." (Ex. 29 (https://support.apple.com/en-us/HT201954).) Furthermore, the jailbreaking process often results in deletion or alteration of data stored on the phone. As discussed in this Part, software already exists that performs certain functions that could be used in the SIF, and to the extent those software components could be used to undermine security, they (like the SIF) would only work on devices that had already assumed security vulnerabilities by being jailbroken.

iPhone 3GS and 4, all known jailbreaks have been applied from within the iPhone user interface, instead of during the boot process. There are publicly known jailbreaks for most recent iPhone OS versions (up to at least version iOS 9.0.2), but they can only be executed from an unlocked iPhone via the user interface, *i.e.*, after the iPhone had booted and had been unlocked. After these jailbreaks are applied, software that has not been signed by Apple may be run.

e. The same brute-force source code still works on jailbroken iPhones. A software project named "iphone-dataprotection" includes a passcode recovery program that can still be compiled, loaded, and run within a jail-broken Apple device. The FBI tool used essentially the same functionality as this project but executed it from a RAMdisk. The FBI recently tested the iphone-dataprotection passcode recovery software on a jailbroken iPhone 6 Plus running iOS 8.4 (in which the passcode had been entered once). With minor modifications this software still functioned and was able to recover the passcode without incurring time delays. The FBI also tested this passcode recovery software on a jailbroken iPad Air 2 running iOS 9.02. In this device the passcode recovery software functioned, but it did incur the time delays and most likely would have erased the device.[15] However, this test does verify that the passcode recovery code works, which has existed for many years and still functions essentially the same. This specific code would not run on the Subject Device "as is," because it is not signed by Apple and also because it would incur time delays and risk causing the device to erase, which would require further development and modifications to the kernel software.[16]

---

[15] It should be noted that the iPhone 6 and iPad Air 2 both use the more advanced A8 processor and the time delay and erase functionality has moved into a separate security controller called the Secure Enclave.

[16] For example, in previous versions of iOS the time delay and password try count resided in the "springboard" user interface, which is in part what allowed the passcode recovery software to work and to bypass the time-delay and auto-wipe features. In approximately iOS 8.4, that functionality moved from the Springboard and would require further modification to bypass the delay and wipe functions.

f. Only Apple can produce and sign the RAMdisk needed to run the passcode guessing code without first unlocking the iPhone. Beginning with the release of the iPhone 4S in 2011, Apple fixed the bug in the boot ROM. Since that time, the Apple chain of trust—which governs the boot process on an iPhone—has remained intact, preventing loading of unsigned RAMdisks. (The jailbreaks that have occurred on iPhones 5C or later have occurred after the boot-up process has occurred, and after a passcode has been entered; the chain of trust through the boot-up process remains intact on those phones.) However, the steps used in the Apple Recovery and DFU mode boot processes have not changed substantially since that time, and Apple's use of a RAMdisk to perform the updates and device recovery processes appear consistent with the methodology of the earlier devices. Without assistance from Apple to digitally sign the code, however, it has not been possible to continue development of these tools for newer devices. The passcode-guessing software employed by these tools has been tested within jailbroken devices running an iOS that has already been booted and unlocked; neither the FBI, nor others to my knowledge, however, have been able to integrate the software into a RAMdisk to test passcodes from a cold-booted iPhone device since the iPhone 4.

29. As set forth above in the previous paragraph, there are already software components available that perform some of the functions of the SIF called for by the Court's Order. Although code similar to what would be in the SIF already exists, it cannot be used on the Subject Device without Apple's signature because of Apple's robust security and code-signing practices.

D. The Encrypted Data on the Subject Device Must Be Decrypted on the Subject Device Itself

30. As described in paragraph 12 of the Initial Pluhar Declaration, an iPhone 5C running iOS 9 is encrypted using a combination of two components: one user-determined passcode, and one unique 256-bit key (referred to as a "UID") fused into the

phone itself during manufacture. (iOS Security at 12; *id.* 11 (diagram); Neuenschwander Decl. ¶ 13.) These two different components are discussed below.

31. According to Apple's documentation, the UID is unique to each device, is fused into the hardware, and is not known to Apple or anyone else, as described on page 10 of iOS Security:

> The device's unique ID (UID) . . . [is] fused . . . into the application processor and Secure Enclave during manufacturing. No software or firmware can read them directly . . . . The UIDs are unique to each device and are not recorded by Apple or any of its suppliers. . . . The UID allows data to be cryptographically tied to a particular device. For example, the key hierarchy protecting the file system includes the UID, so if the memory chips are physically moved from one device to another, the files are inaccessible. The UID is not related to any other identifier on the device.

32. I know from Supervisory Special Agent ("SSA") Pluhar that the Subject Device was powered off when the FBI found it. When the Subject Device was powered on, it displays a numerical keypad (like that on a telephone), and a prompts for four numbers to be entered.

33. With a four-digit numerical pin, there are only 10,000 possible passcodes. Testing 10,000 passcodes electronically would likely take less than a day, depending on how the SIF were configured.

34. Apple's iOS Security also explains that because its passcodes are permitted to be weak in that they can be only four numbers, Apple has included additional features to discourage brute-force attacks. These features are described in paragraphs 13 and 14 of the Initial Pluhar Declaration, and on page 12 of Apple's iOS Security (noting that iOS 9 iPhones (1) escalate time delays between failed passcodes, and can (2) be configured to wipe their contents after ten failed passcodes, to "discourage brute-force passcode attacks").

35. The UID is itself a strong encryption key. It is fused into the hardware and is both unknowable and unchangeable: it is always used the same way to create the encryption key. The only variable is the passcode.

36. Because both the UID, which is unique and embedded in the device itself, is a part of the encryption key (along with the user-generated passcode), the data that is stored on the Subject Device will need to be decrypted on the Subject Device. Because only Apple-signed software can run on the iPhone, and the decryption must occur on the Subject Device, any code or software tools needed to assist in testing passcodes must be signed using Apple's encryption keys.

E. Apple's iCloud Backup

37. I know from SSA Pluhar that the Subject Device was found in a powered-off state. Based on Apple's published documentation, open source research relating to Apple's encryption, and Apple press releases about iOS 8 and later encryption, I believe that (1) the device would not connect to a WiFi network until the passcode was entered, and (2) even if the device could be forced to perform an iCloud backup, the user data would still be encrypted with the encryption key formed from the 256 bit UID and the user's passcode.

38. Subsequent to seizing the Subject Device, the FBI performed several tests on exemplar phones to test whether a cold-booted iPhone could connect to a trusted WiFi network and perform a backup. The result of that testing was that cold-booted iPhones would not connect to a WiFi network.

a. To the best of my knowledge, a cold-booted iPhone will not connect to WiFi networks trusted by the Subject Device such as a home or work network until the passcode is entered. However, according to Apple and verified by the FBI, there are some WiFi networks inherently trusted by iOS, such as those operated by iPhone sponsors (referred to as carrier-sponsored WiFi). For example, an AT&T iPhone can automatically connect to an AT&T hotspot.

b. When the FBI tested a locked AT&T phone on which the passcode had been entered once by taking it to an area with an AT&T hotspot, the phone connected automatically to the hotspot, as indicated by the WiFi indicator on the top

banner of the lock screen display. Additionally the "Find My iPhone" service was used and was able to locate the iPhone, verifying that a phone in which the passcode has been entered will connect, even when screen-locked, to a trusted WiFi network.

c. The same test was also done with the phone first powered off and restarted, but with the passcode not having been entered. In this scenario, the test phone did not show any indication it was connected to the AT&T hotspot through the banner. Additionally, the "Find My iPhone" service was unable to locate the device. The results of these tests show that WiFi is not enabled on the device until after the passcode is entered.

d. Further tests were conducted by myself and a colleague in CEAU by taking an iPhone 5 running iOS 9.02 and an iPhone 6 Plus running iOS 9.2 into a radio-frequency shielded chamber to test their electronic emissions. Both iPhones were fully charged, connected to power and had their WiFi enabled. The same series of tests was done on both phones with identical results. When the iPhone was not protected by a passcode and was powered on in that chamber, it began to emit signals in the frequency band of 2.4 gigahertz (GHz), a common band for WiFi connections. This is consistent with the iPhone trying to detect a WiFi network. When the iPhone *was* protected by a passcode and was powered on in the same chamber without entering the passcode, no emissions in the 2.4 GHz frequency band were detected. This indicates that the WiFi was not active. When the passcode was entered, WiFi 2.4GHz emissions were detected. The phone was allowed to screen lock after the passcode had been entered. Again, 2.4GHz emissions were detected. Each phone was rebooted, no passcode entered, and left overnight in the chamber. No 2.4GHz signals were observed. These tests indicate the WiFi is not active on a cold-booted device until the passcode has been entered at least once.

e. The FBI does not know of any way to force an iPhone that has not had the passcode entered at least once since being powered on to perform an iCloud backup.

39. This result is consistent with Apple's security documentation, which states that data stored on the device is encrypted using a key that is a combination of both the UID (the device-specific unique identifier) and the passcode generated by the user. Unless the passcode is entered by the user, the entire encryption key could not be used to decrypt the data, and the data therefore could not be backed-up to an iCloud—at least in a state that could be recovered outside the device.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that this declaration is executed at Virginia, on March 9, 2016.

Stacey Perino

STACEY PERINO