

iOS Forensics Lunch & Learn

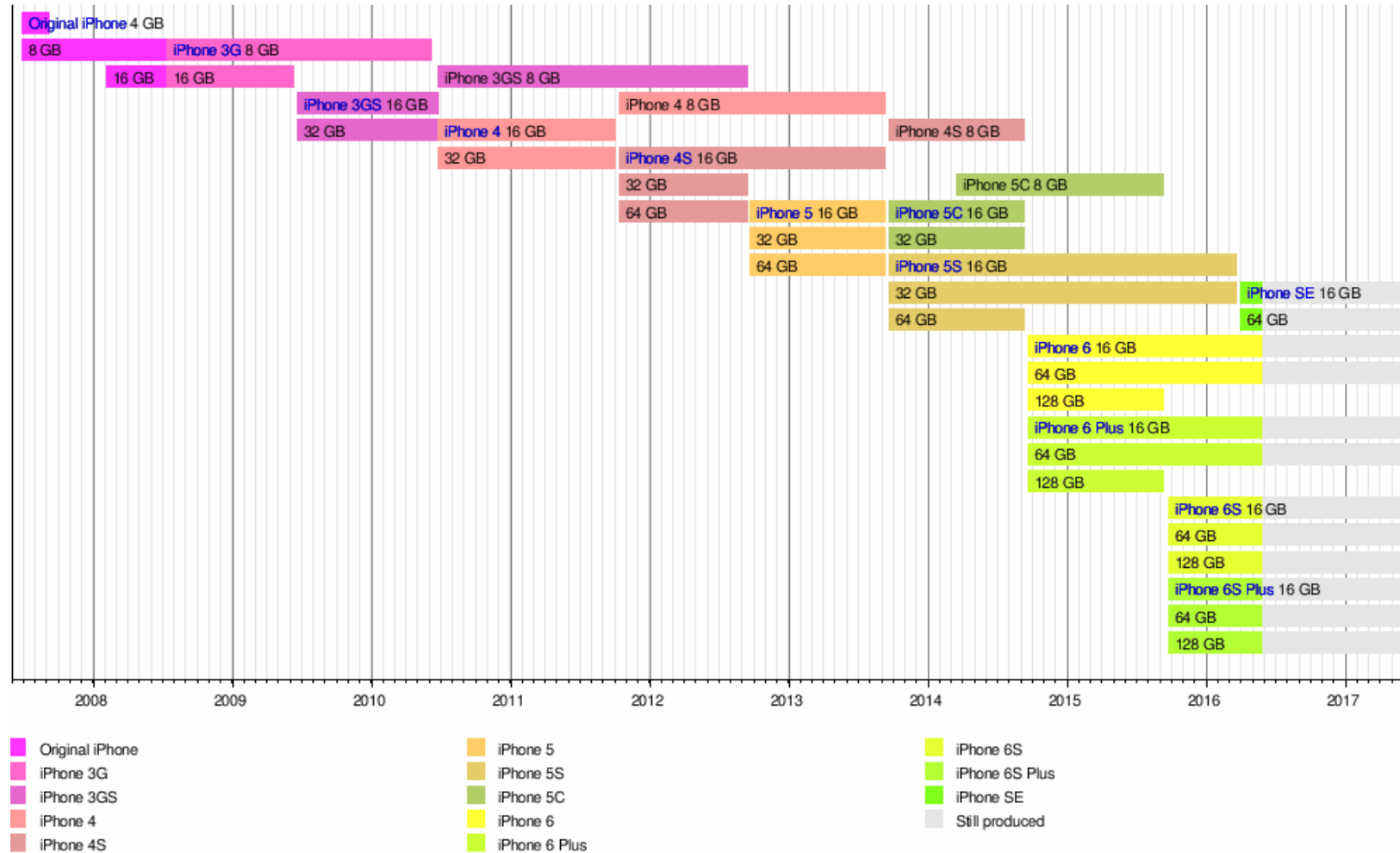
Dan O'Day

June 2, 2016

Caveat Emptor

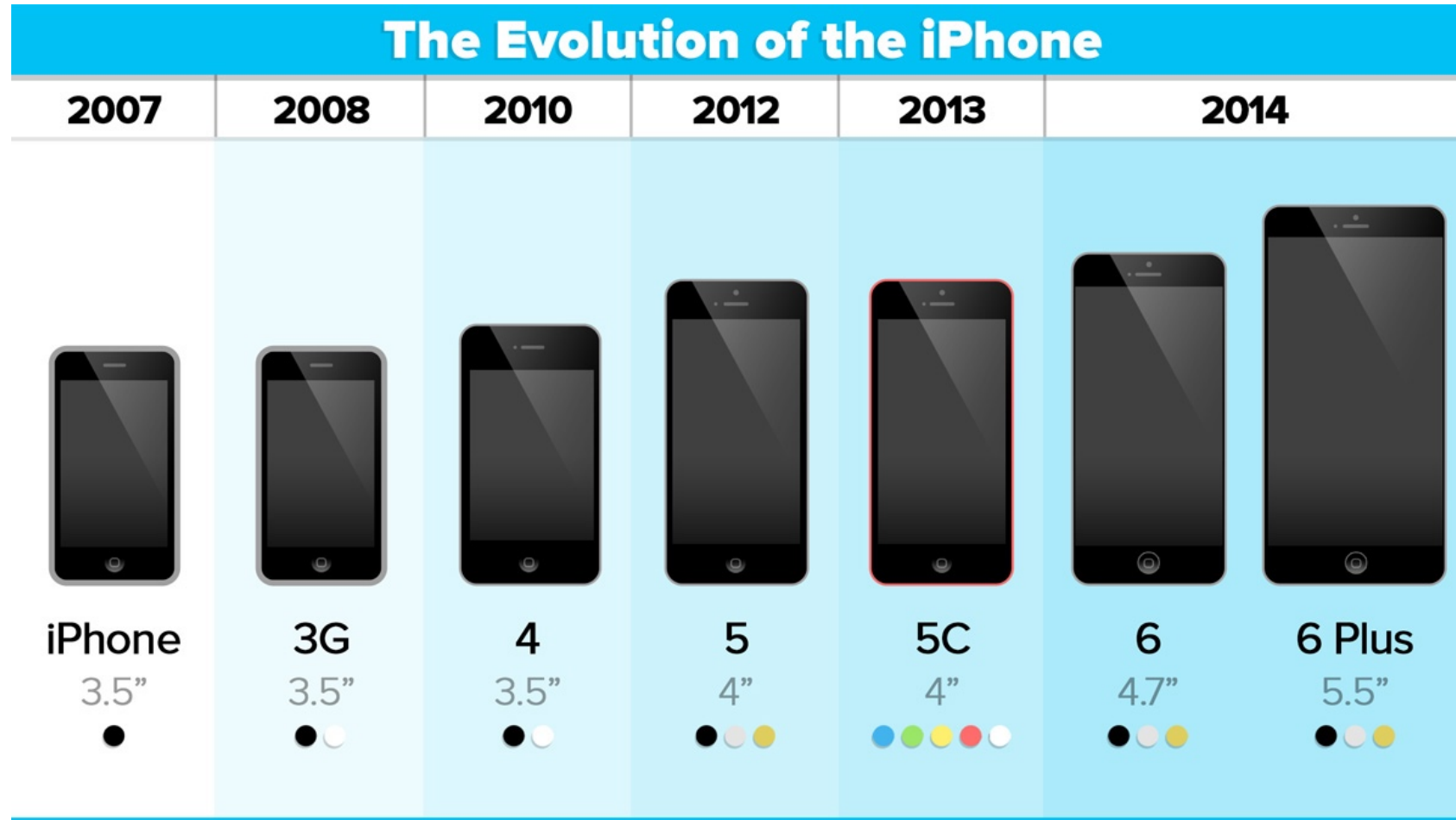
- This is more so a talk on iOS Security and challenges to iOS forensics
- Due to time constraints, I can't cover much more today

iPhone timeline



Source: https://en.wikipedia.org/wiki/IPhone_6S#Timeline_of_models

iPhone timeline (continued)






Source: <http://www.reglobe.in/blog/best-mobile-will-iphone-dethroned/>

iPhone timeline (continued)



Source: <http://www.forbes.com/sites/gordonkelly/2015/09/10/iphone-6s-vs-iphone-6s-whats-the-differences>

iOS timeline

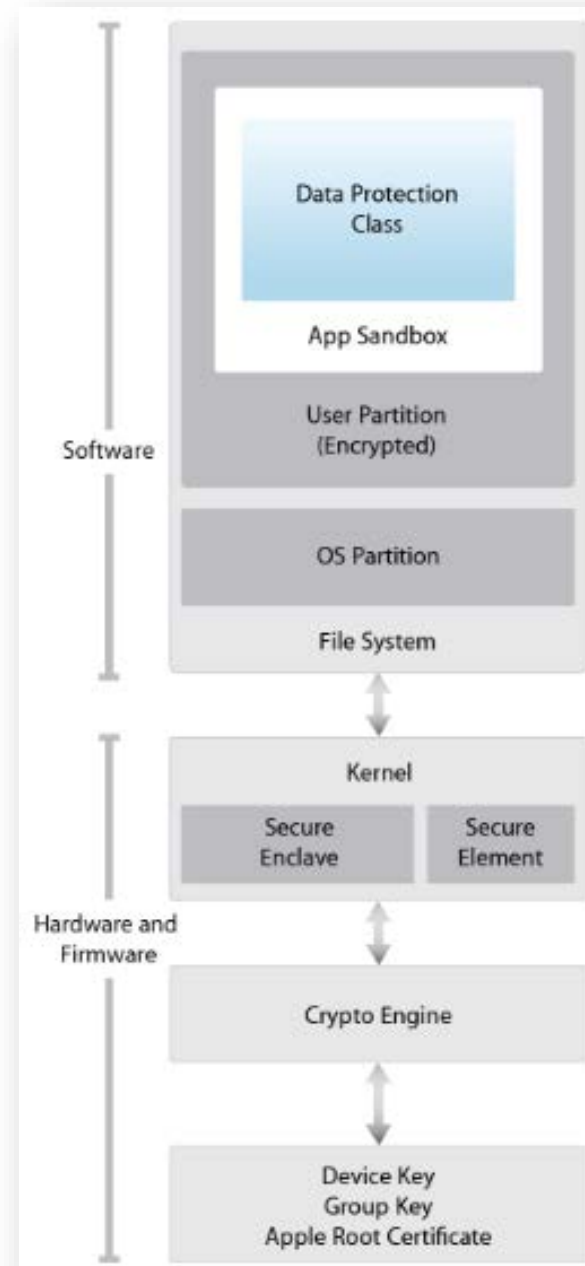
Legend:	 Discontinued	 Current	 Beta
----------------	---	---	--

Version	Build	Release date
3.1.3	7E18	February 2, 2010
4.2.1	8C148	November 22, 2010
5.1.1	9B206	May 7, 2012
6.1.6	10B500	February 21, 2014
7.1.2	11D257 11D258	June 30, 2014
8.4.1	12H523 (Apple TV Software 7.2.1)	February 25, 2016
9.3.2	13F69	May 16, 2016
9.3.2	13Y772 (tvOS 9.2.1)	May 16, 2016
9.3.3 Beta 1	13G12	May 23, 2016
9.3.2 Beta 4	13Y807 (tvOS 9.2.2 Beta 1)	May 23, 2016

Source: https://en.wikipedia.org/wiki/iOS_version_history

iOS 9.3 Security

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf



iOS 9.3

New in iOS 9.3:

- Night Shift
- Enhanced Protection for Notes
- Expanded Health Dashboard
- CarPlay Enhancements
- Classroom Enhancements
- Wi-Fi Calling for Verizon customers

Fixes:

- AppleUSBNetworking (USB DDoS)
- FontParser (RCE via malicious PDF)
- Nghttp2 updated to v1.6.0 to prevent RCE (HTTPProtocol)
- IOHIDFamily (app can determine kernel memory layout)
- Numerous kernel & WebKit patches
- Safari 9.1
- Unix Epoch bug

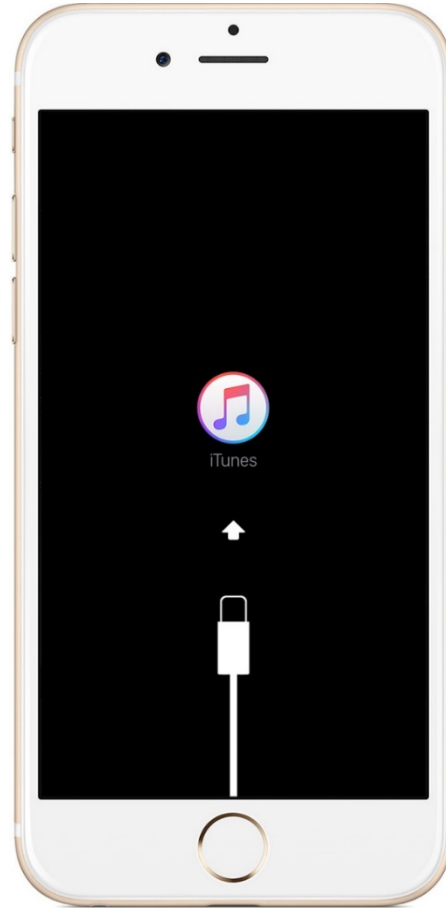
Source: <https://support.apple.com/en-us/HT206166>

iOS 9.3 secure boot chain

- Each step of startup process verified in “chain of trust” via cryptographically-signed components
 1. Code executed from Boot ROM (hardware “root of trust” laid down during chip fabrication). Boot ROM contains Apple Root CA public key which is used to verify signature of Low-Level Bootloader (LLB)
 2. If LLB is verified, it verifies and runs iBoot (iBoot runs recovery mode also). The now defunct iDroid Project developed a (now deprecated) open-source version called “openiBoot”
 3. If iBoot checks out, it verifies and runs the iOS kernel
 4. For devices with cellular access, the baseband subsystem has its own secure boot process
 5. For devices with A7+ processors, the Secure Enclave (SE) coprocessor also has its own secure boot process (A7 processor is a 64-bit system on a chip that first appeared in the iPhone 5s; 6 and 6 Plus have A8, and 6S and 6S Plus have A9)
- Failure to verify the next component in the chain after the LLB results in the device entering recovery mode
- Failure to load/verify LLB results in the device entering device firmware upgrade (DFU) mode

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 recovery mode



Source: <https://support.apple.com/en-us/HT201263>

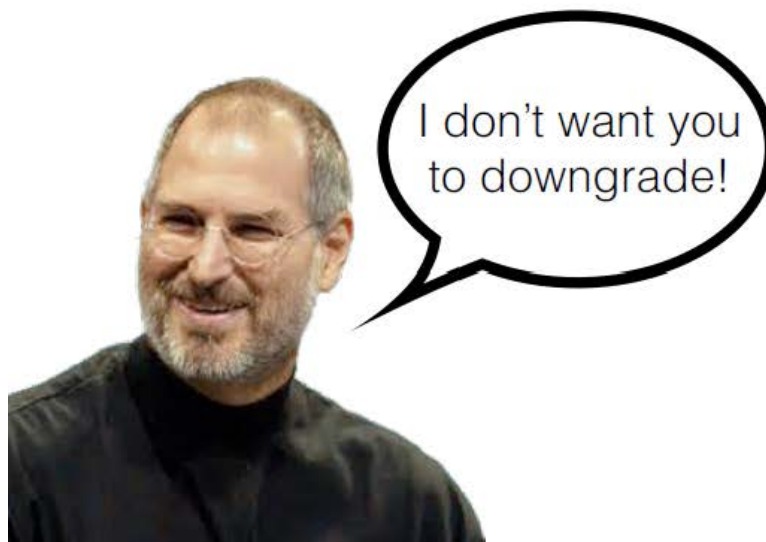
iOS 9.3 System Software Authorization

- Apple uses a process called System Software Authorization to prevent downgrades
 - This is enhanced by the Secure Enclave (SE) in A7+ chips
- The entire upgrade process is secured using a list of cryptographic “measurements” for each part of the installation bundle, which includes using the device’s unique ID (ECID) and Apple’s digital signature protocol
 - Prevents copying an old iOS version from one device to another
 - Also uses a “nonce” (anti-replay value) to prevent saving a server’s response and using it to tamper with a device or alter software
- Latest firmware (*.ipsw) files can be downloaded from <http://www.iclarified.com/750/where-to-download-iphone-firmware-files-from>

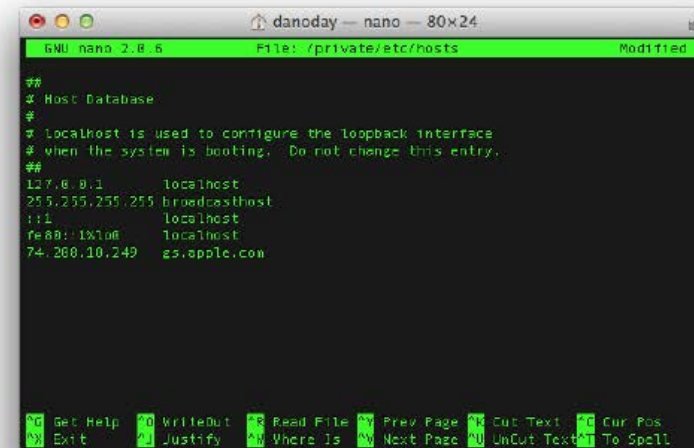
Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 System Software Authorization (continued)

It used to be so much simpler before the nonce and ECID integration! Not anymore...



```
T0huWav@hu:~ danoday$ sudo nano /private/etc/hosts
```



```
GNU nano 2.8.6 File: /private/etc/hosts Modified
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1 localhost
255.255.255.255 broadcasthost
::1 localhost
fe80::1%lo0 localhost
74.208.10.249 gs.apple.com

^G Get Help ^O WriteOut ^R Read File ^V Prev Page ^X Cut Text ^C Cur Pos
^Y Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell
```

iOS 9.3 Secure Enclave (SE)

- SE is a coprocessor in A7+ processors
- Provisioned during fabrication with its own UID that is inaccessible to other parts of the system and *is unknown to Apple*
 - On startup the UID is entangled with a temp (“ephemeral”) key and is used to encrypt SE’s allocated memory space
- Encrypted memory and hardware RNG
- Has its own secure boot process
- Provides all cryptographic operations for Data Protection key management
 - *And...* now also maintains Data Protection integrity even in the event of kernel compromise!
- SE processed fingerprint data from Touch ID sensor
 - The processor forwards the data to the SE but can’t read it due to it being encrypted with a session key that is negotiated using the device’s shared key (lions and tigers and AES key wrapping... oh my!)

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 Touch ID

- Touch ID does not obviate the need for a device password
- The scanned fingerprint is stored in SE's encrypted memory while being analyzed, then is discarded
- The resulting nodes map is stored encrypted without identifying information and can only be read by the SE (it is [allegedly] never sent to Apple nor backed up to iCloud or iTunes)
- If Touch ID is disabled, the keys for Data Protection (stored in SE) are discarded *every time* the device is locked
- If enabled, the Data Protection keys aren't disabled but rather wrapped with a key given to the Touch ID subsystem inside the SE. The key is provided after a successful unlock if the fingerprint validates
- The Touch ID keys are discarded
 - every reboot
 - after 48 hours
 - after 5 failed Touch ID recognition attempts

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

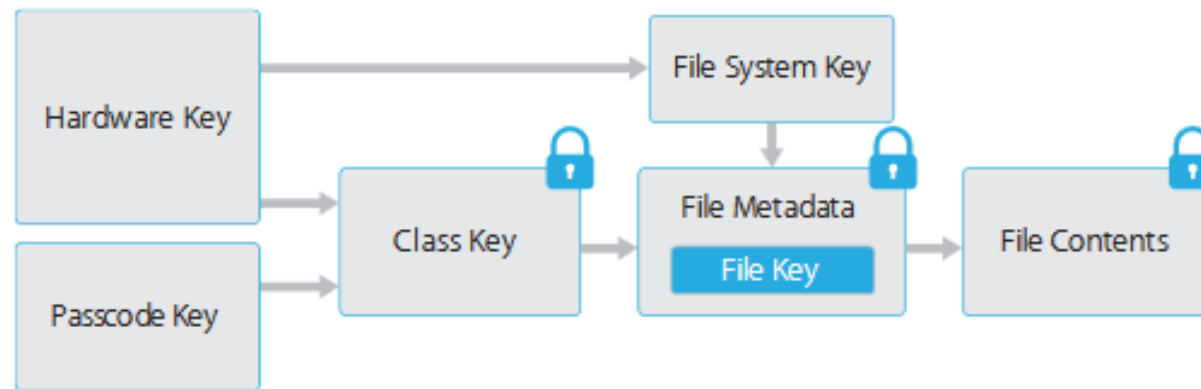
iOS 9.3 encryption

- iDevices have dedicated AES 256 crypto engines built into DMA path between flash storage and main system memory
- UID and GID AES-256-bit keys are fused into application processor and SE during manufacturing
 - No software/firmware has direct read access
 - Apple does not record the UID
 - GID is common to a class of devices (e.g., all devices using A9 processor)
 - UID and GID not accessible via JTAG nor other debugging interfaces

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 Data Protection

- Key system apps use it by default, including
 - Messages
 - Mail
 - Calendar
 - Contacts
 - Photos
 - Health
- Third-party apps no longer have to opt in after iOS 7 (they automatically receive this protection)



Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 passcodes

- Data Protection is automatically enabled upon setting up a device passcode
- Passcode is entangled with device's UID, which require brute-force attacks to be performed on the device itself
- An iteration count slows down brute-force attacks, and the device can be configured to wipe after so many failed attempts (it would take about 5.5 years to attempt all combinations of a 6-character alphanumeric passcode with lowercase letters and numbers)
- On devices with A7+ processor, delays are enforced by the Secure Enclave (even if device is restarted, delay is still enforced and timer starts over)

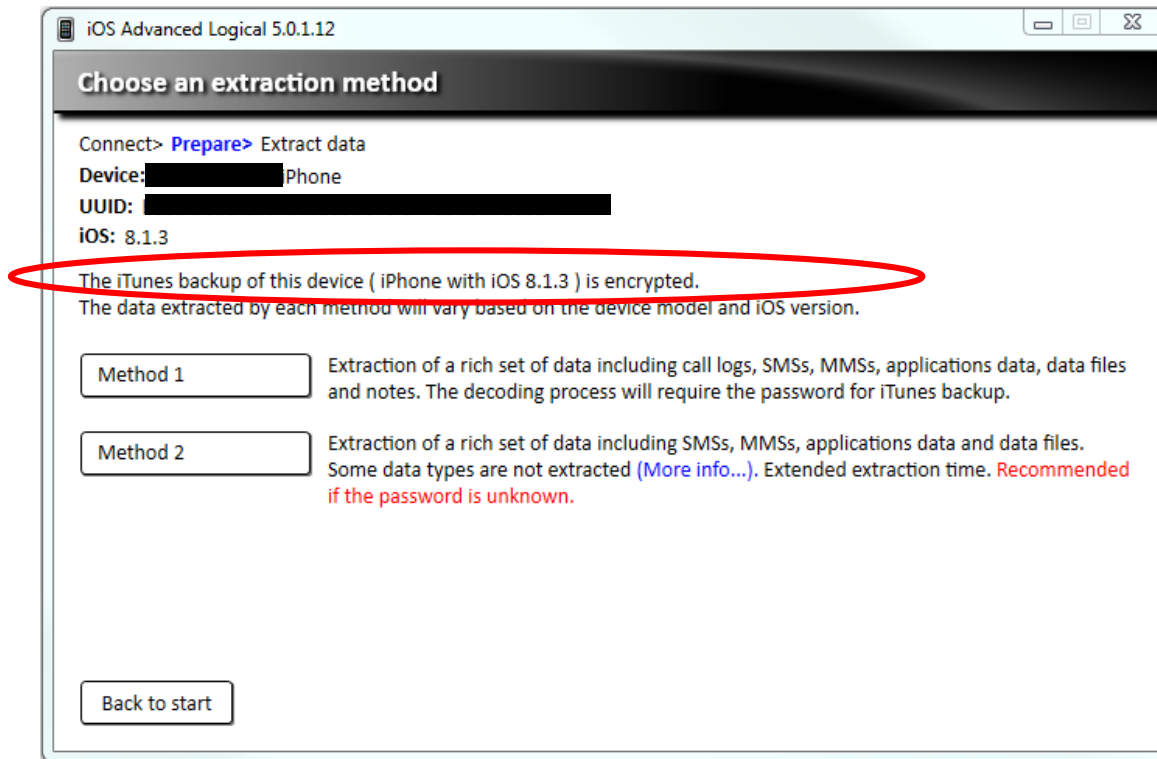
Delays between passcode attempts

Attempts	Delay Enforced
1-4	none
5	1 minute
6	5 minutes
7-8	15 minutes
9	1 hour

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 Keychain

- Stores application passwords and other short but sensitive data in a SQLite database
- During forensic acquisition, can only be decrypted if iTunes Backup Encryption is configured and password is known



Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

iOS 9.3 San Bernardino Terrorist iPhone 5c

- The FBI's court order was technically feasible *because the iPhone 5c doesn't have SE*
- As such, all passcode security was implemented in the main software (*.ipsw) since no separate processor exists to enforce this
- A firmware update could have disabled the obstacles to a brute-force attack (but could not have bypassed or otherwise provided the passcode itself)

"[Provide] the FBI with

- ***a signed iPhone Software file, recovery bundle, or other Software Image File ("SIF") that can be loaded onto the SUBJECT DEVICE.***
- ***The SIF will load and run from Random Access Memory ("RAM") and will not modify the iOS on the actual phone, the user data partition or system partition on the device's flash memory.***
- ***The SIF will be coded by Apple with a unique identifier of the phone so that the SIF would only load and execute on the SUBJECT DEVICE. The SIF will be loaded via Device Firmware Upgrade ("DFU") mode, recovery mode, or other applicable mode available to the FBI. Once active on the SUBJECT DEVICE, the SIF will accomplish the three functions specified in paragraph 2. The SIF will be loaded on the SUBJECT DEVICE at either a government facility, or alternatively, at an Apple facility; if the latter, Apple shall provide the government with remote access to the SUBJECT DEVICE through a computer allowed the government to conduct passcode recovery analysis."***

Source: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

Jailbreaking

Jailbreaking

- *CAVEAT: This is a view from 30,000 feet*
- Kernel must be patched (/private/etc/fstab)
 - iDevices have two slices (partitions) but system partition is read-only
 - A successful jailbreak must patch or bypass the checks for signed code in the chain of trust
 - Exploits target either Boot ROM or userland (the latter are more common)
- Pangu Team currently has best option for devices up to iOS 9.1 (Pangu Team has dominated iOS 9 jailbreaking thus far)
- No reliable methods for iOS 9.2+ yet

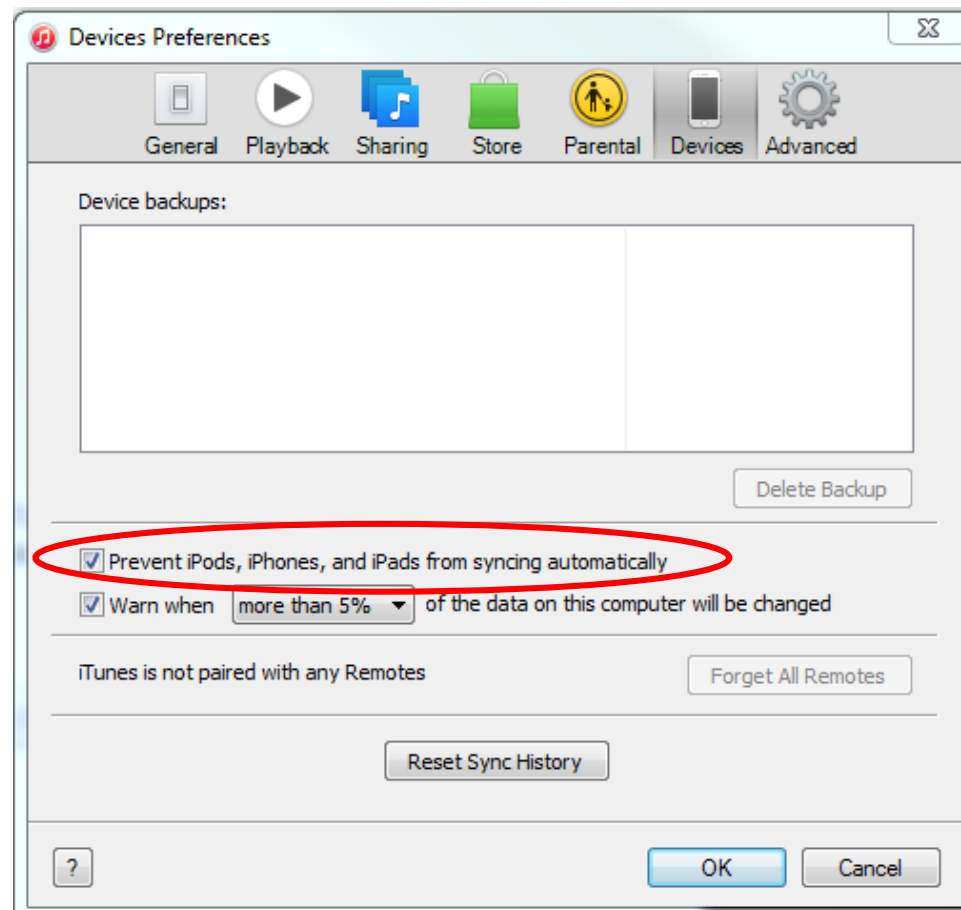
Jailbreak types

- **Untethered** – device reboots and kernel patches itself (no computer needed)
- **Tethered** – device needs computer to assist with patching kernel every time the device reboots (the device must be “re-jailbroken” on every boot)
- **Semi-tethered** – device needs computer to assist with patching kernel every time the device reboots *in order to run modified code* (but the rebooted but unpatched device is still usable for simple tasks such as making calls, texting, etc.)

Forensic environment

Forensic environment

- Install iTunes
- Warning! iTunes will automatically sync with devices you plug in!

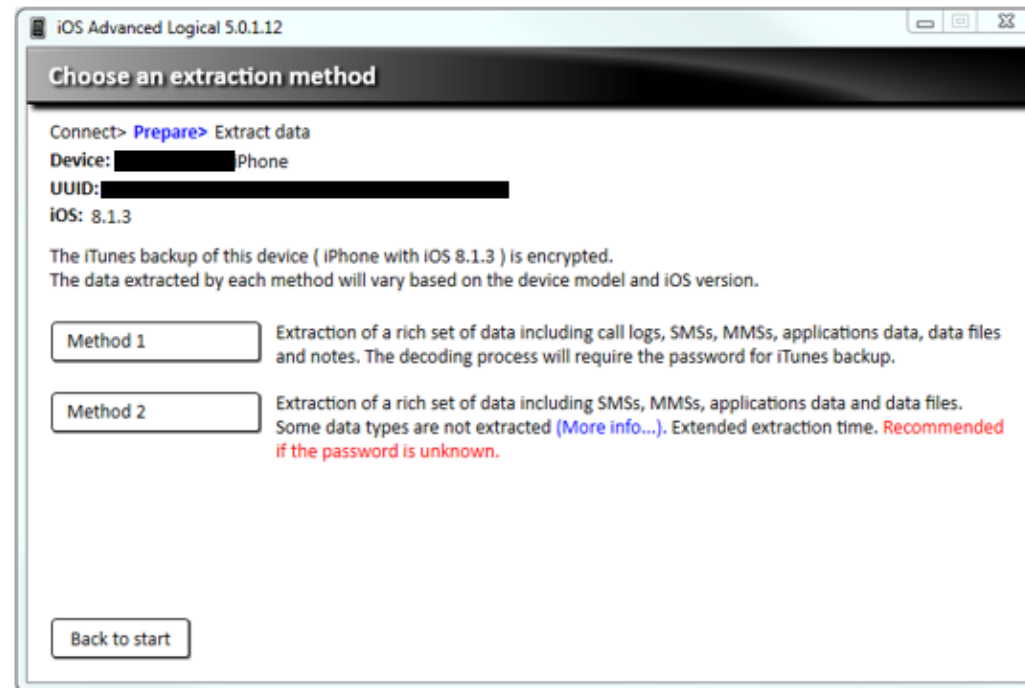


Forensic acquisition – physical vs. logical

- No physical acquisition for devices using hardware encryption of slice 2 (user data)
 - No physical acquisition for iPhone 4s+
- So much easier with iPhone 4 and earlier
 - Jailbreak device
 - ssh into device
 - Default root password is 'alpine'
 - dd dump data
- Cellebrite has its own proprietary “advanced logical”, which means very little after iOS 8 was introduced
- Always acquire via every method available

Forensic acquisition

- Numerous forensics tools (Cellebrite, XRY, etc.)
- An iTunes Backup will get a lot also
- To parse iTunes backup, you can use a forensics tool or something free such as iPhone Backup Extractor (available from <http://supercrazyawesome.com>)
- Cellebrite has two “advanced logical” acquisition methods, the second is essentially an iTunes backup



Thank you