Faux Disk Encryption

Realities of Secure Storage on Mobile Devices

Daniel A. Mayer
@DanlAMayer

Drew Suarez
@utkan0s

Who we are

nccgroup
freedom from doubt

Daniel Mayer

- Principal Security Consultant with NCC Group
- Developer of idbtool.com, iOS pentesting tool

Drew Suarez

- Senior Security Consultant, Research Director with NCC Group
- CyanogenMod (OSS) Device bringup / Wiki

NCC Group

- UK Headquarters, Worldwide Offices
- Softare Escrow, Testing, Domain Services

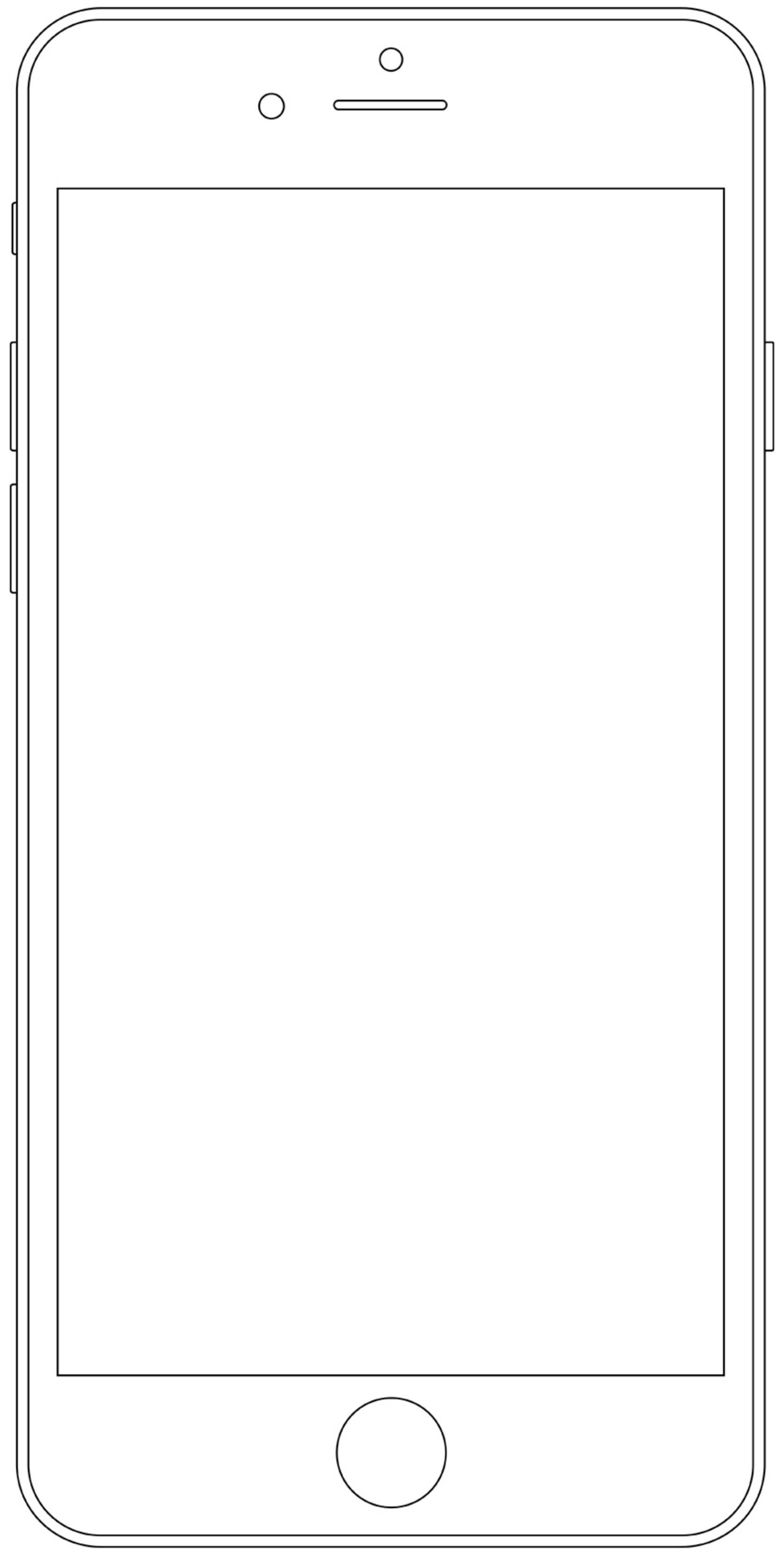
Outline

1. Introduction

2. Secure Storage on iOS

3. Secure Storage on Android

4. Where does this leave us?



Apps Dominate Mobile

Traditional

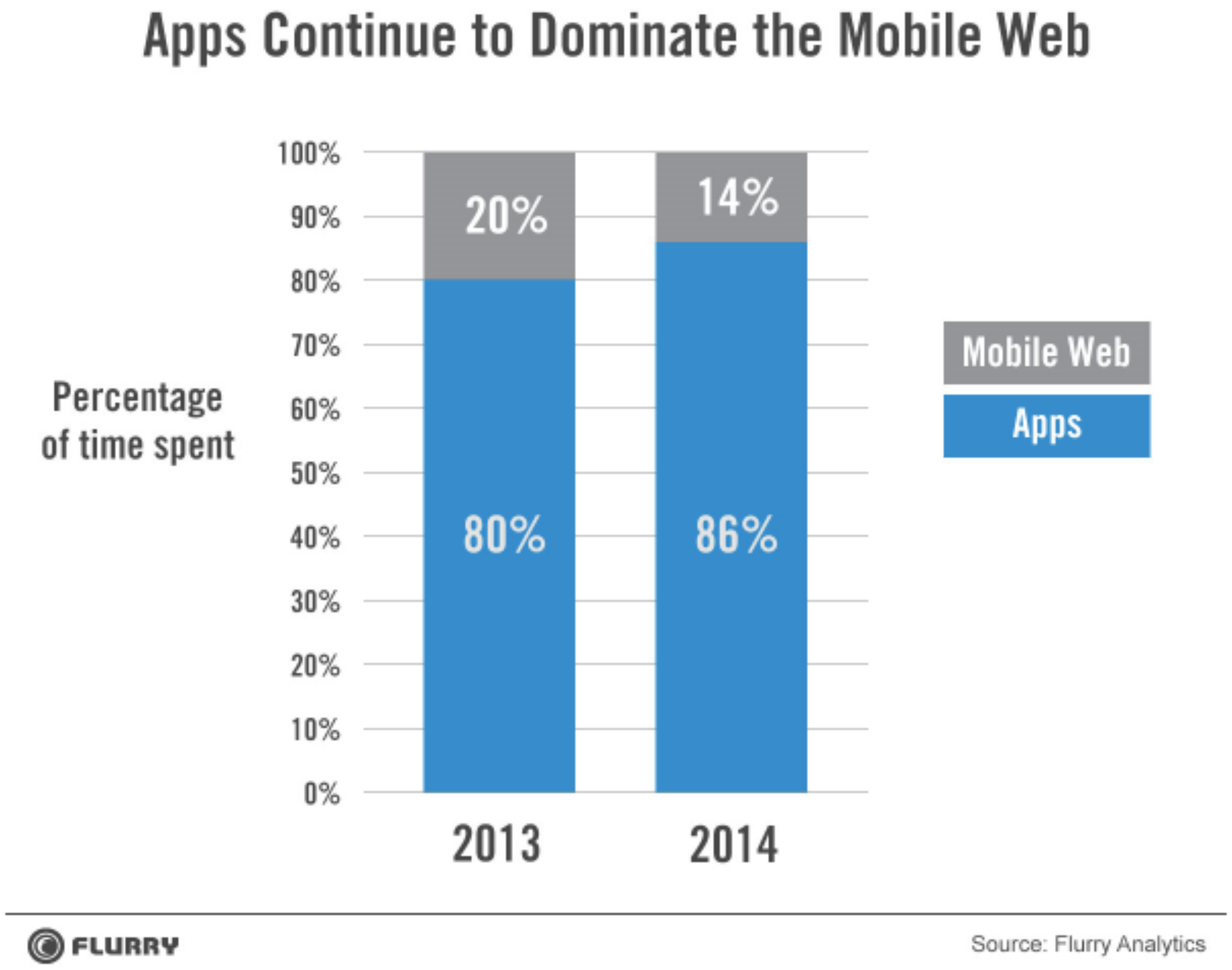
All data stored on server

Tight controls

Mobile

Data stored on device

Difficult to control



Challenge: Device Mobility

Data is being carried around

Devices prone to loss/theft [1]

- 1.4 million phones lost
- 3.1 million stolen
- (US, 2013)

[2]

Challenge: Data Accessibility

nccgroup
freedom from doubt

Local Data

Data cached and stored on the device

Credentials

Usernames / passwords

Access tokens

Challenge: Usability

Known security controls reduce usability

There is no absolute security

Remote Attacker

There is no absolute security

Coffee Shop Attacker

There is no absolute security

Casual Thief

There is no absolute security

Targeted Attacks

There is no absolute security

Nation States

There is no absolute security

Mobile Data Security

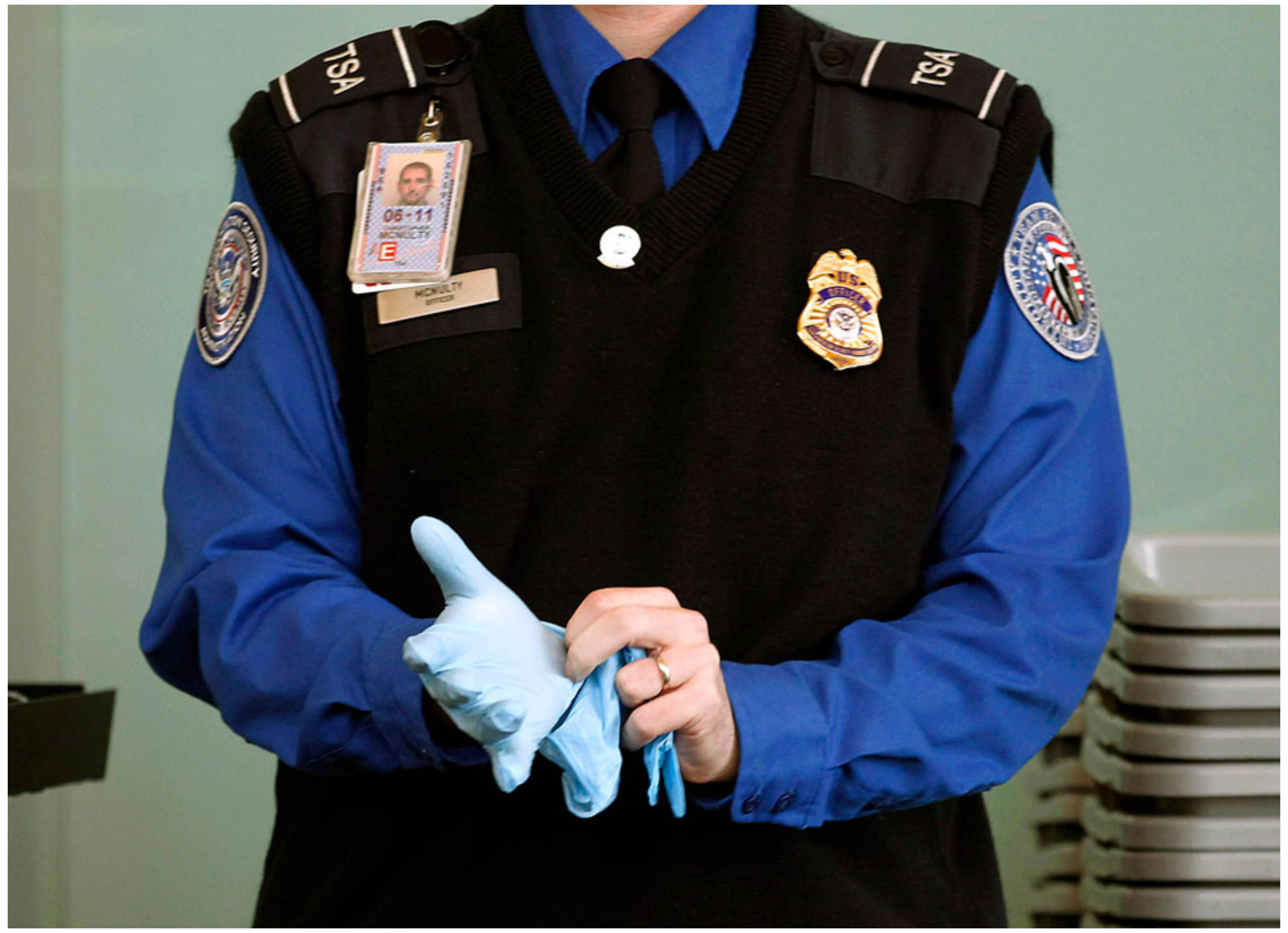
A Word on Full-Disk Encryption

nccgroup
freedom from doubt

Encrypts files stored on the file-system
- Transparently decrypted when read
- Transparently encrypted when written

Protection only when device is turned off
- In combination with strong passcode!

Need more fine-grained control



Secure Data Storage
…on iOS

iOS Boot/App signing

Apple Hardware + Apple Software

Boot Chain Completely Signed

- Hardware root of trust (ROM) contains Apple CA

iOS Updates

- Signed by Apple
- Downgrades not allowed

App Signing

- All code running on iOS must be signed by Apple

[3]

Bootstrapping Encryption

Device Passcode

- Not stored on device
- Derive encryption key when entered
- Wipe key when device is locked

Problems

- Users choose weak passcodes [1]
- Prone to offline brute-force attacks

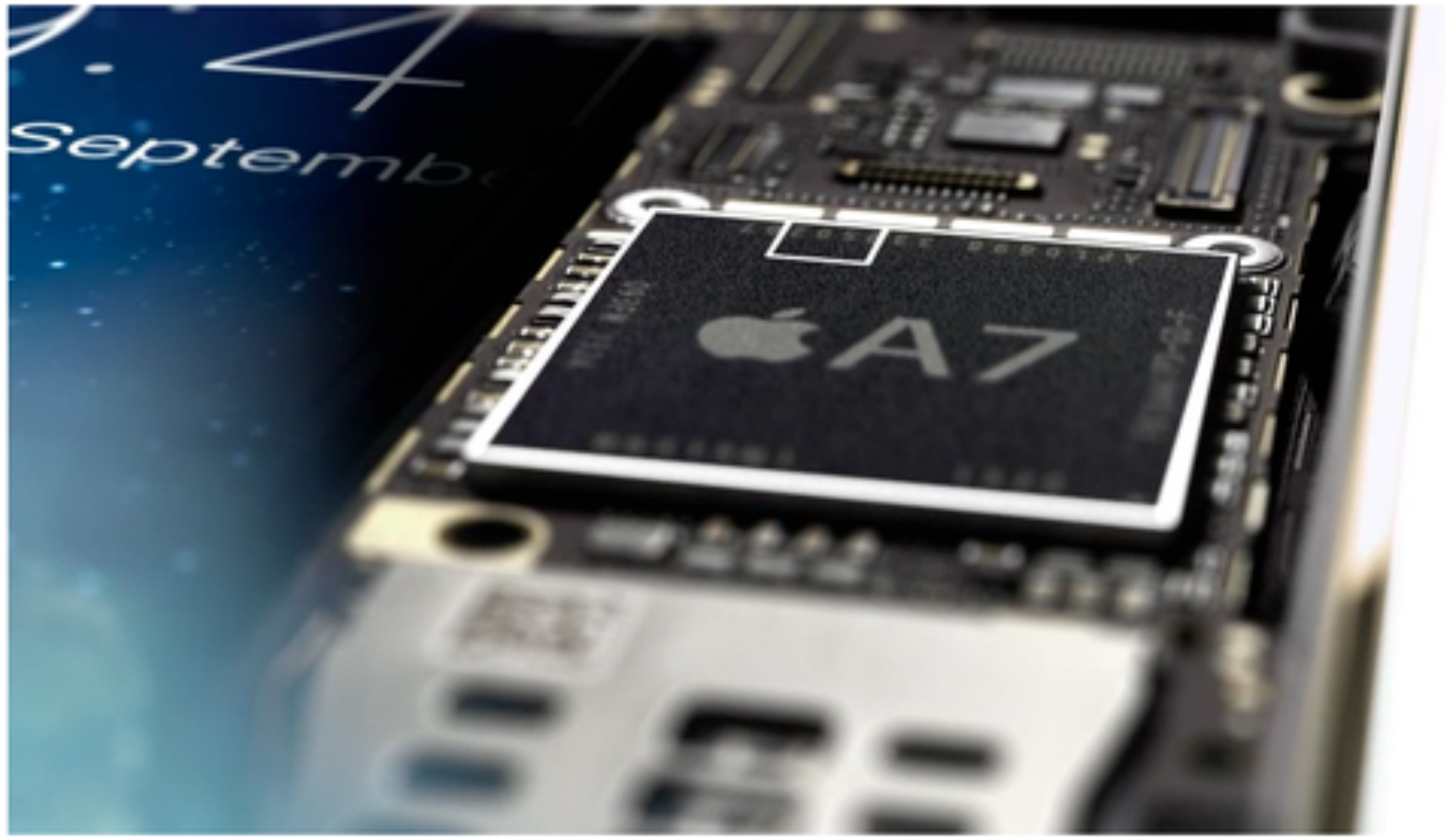
Hardware Root of Trust

Tie Encryption to a Device

Unique encryption key per device

Cannot be read by operating system

Can "ask" Secure Enclave to decrypt

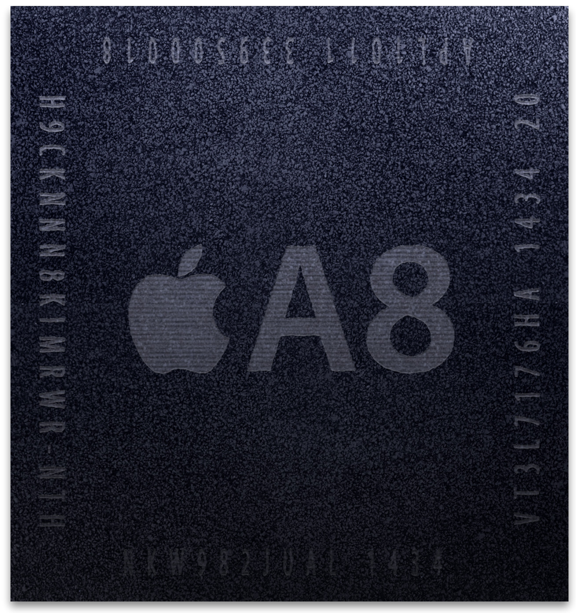


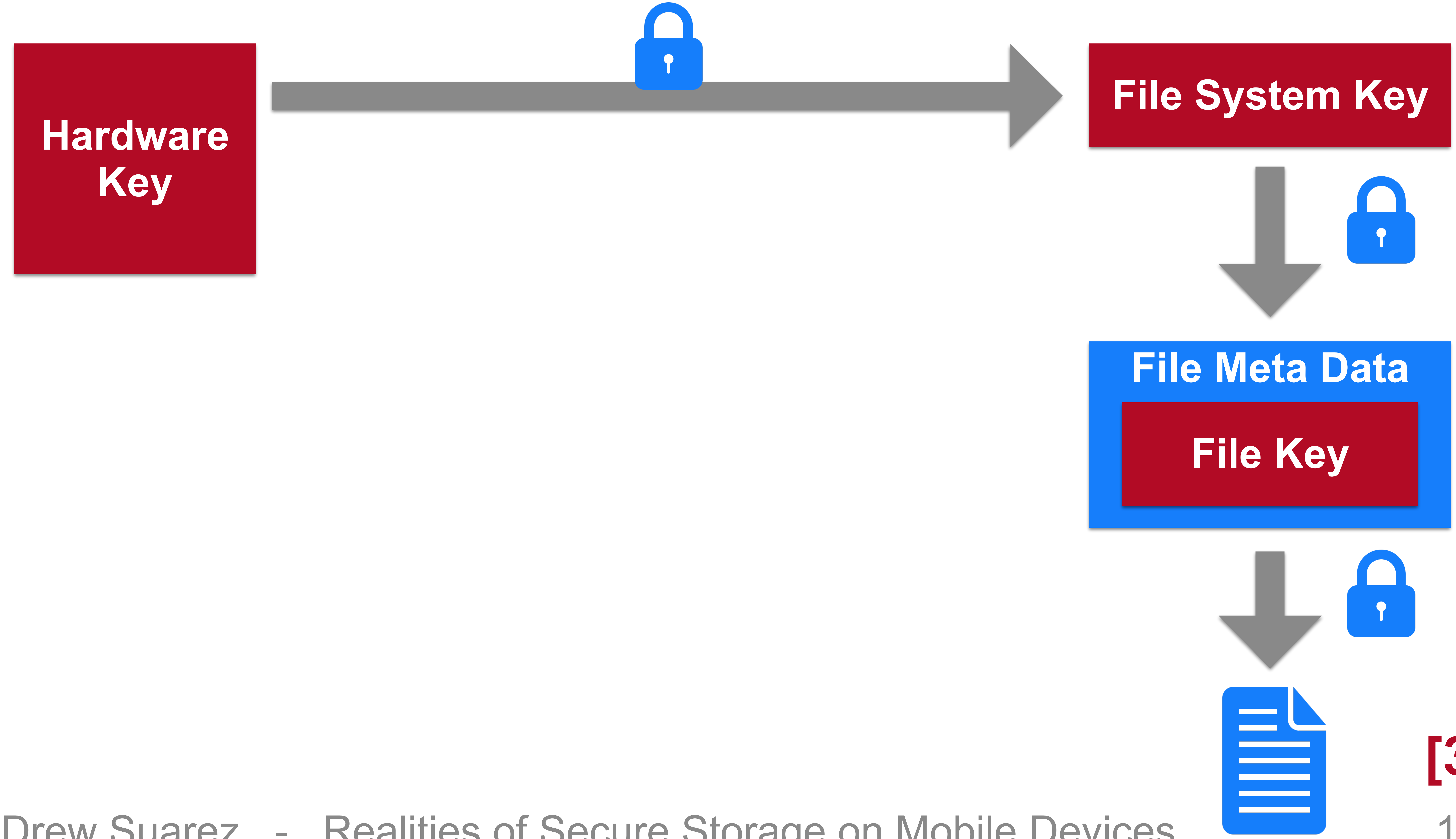
Hardware Controls

Enforce brute-force controls

Enforce device-wipe

iOS Encryption Hierarchy

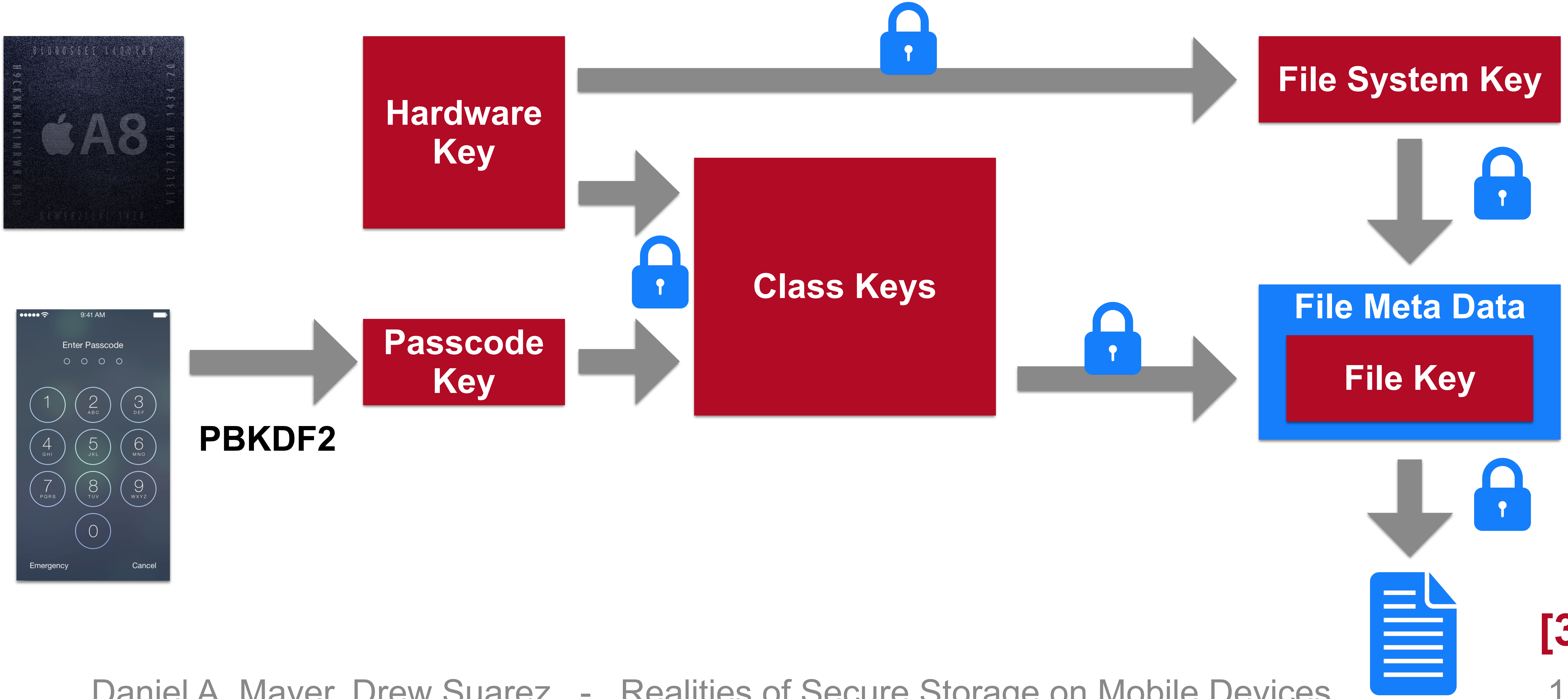




[3]

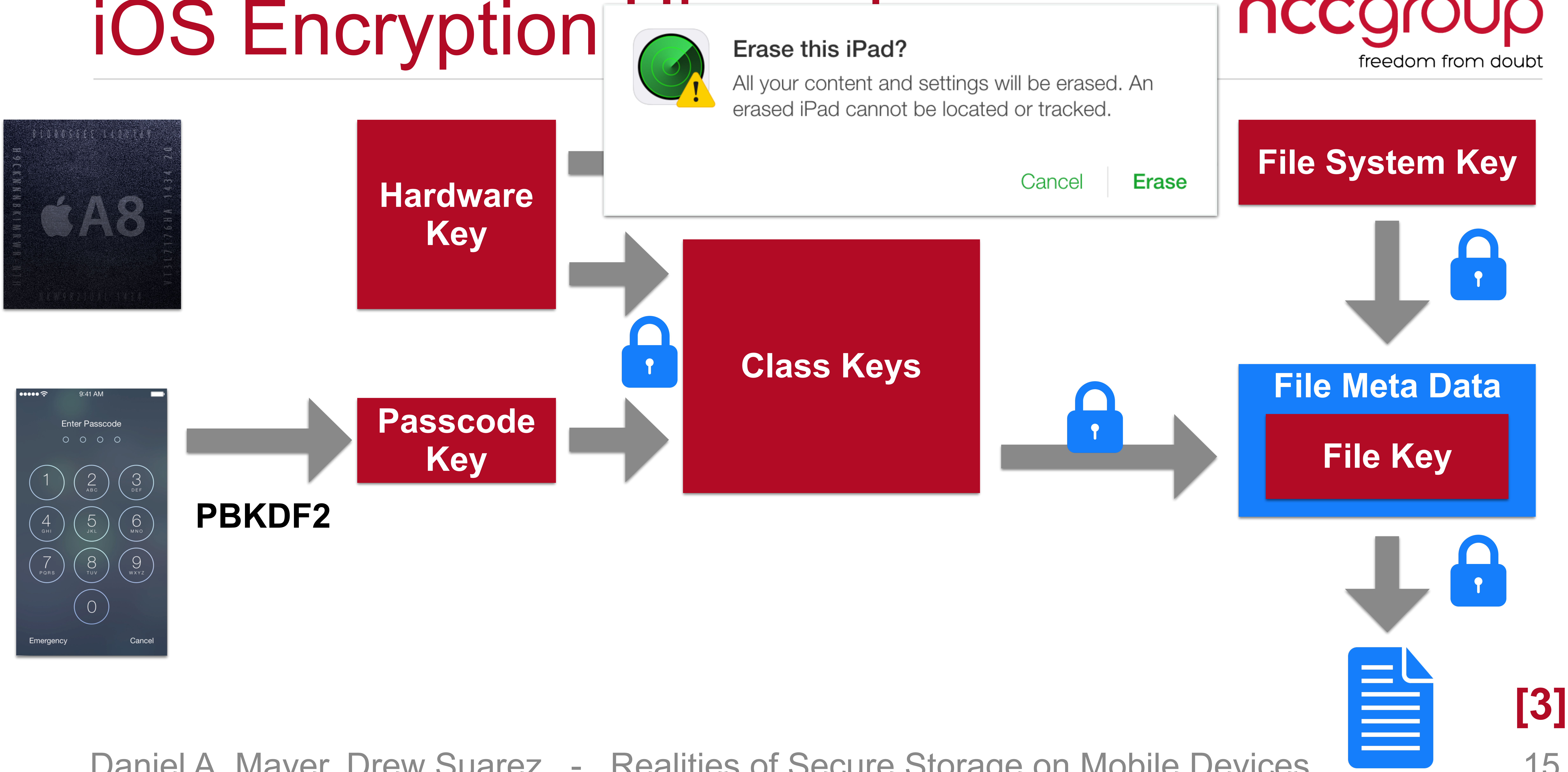
iOS Encryption Hierarchy

nccgroup
freedom from doubt



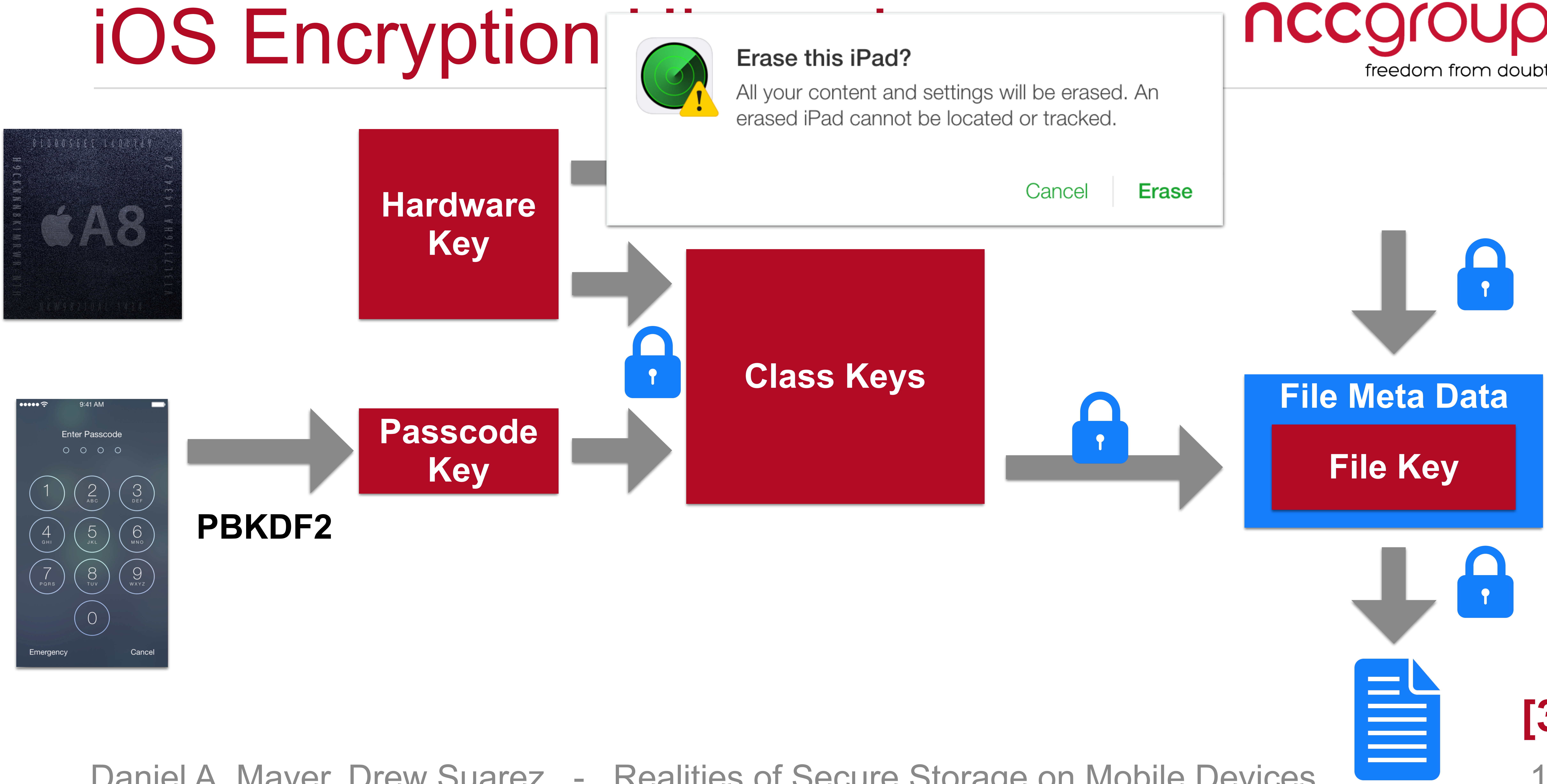
[3]

iOS Encryption



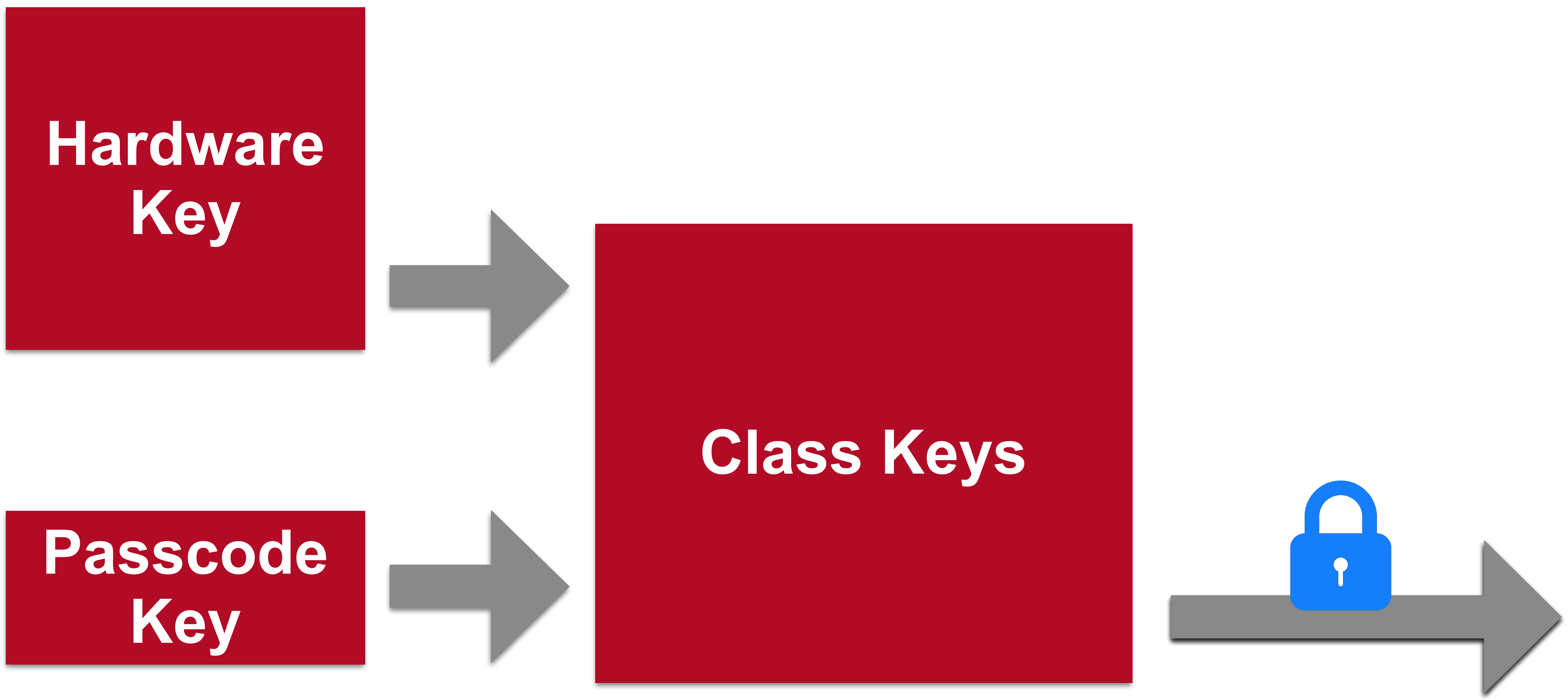
[3]

iOS Encryption



[3]

iOS Encryption Hierarchy



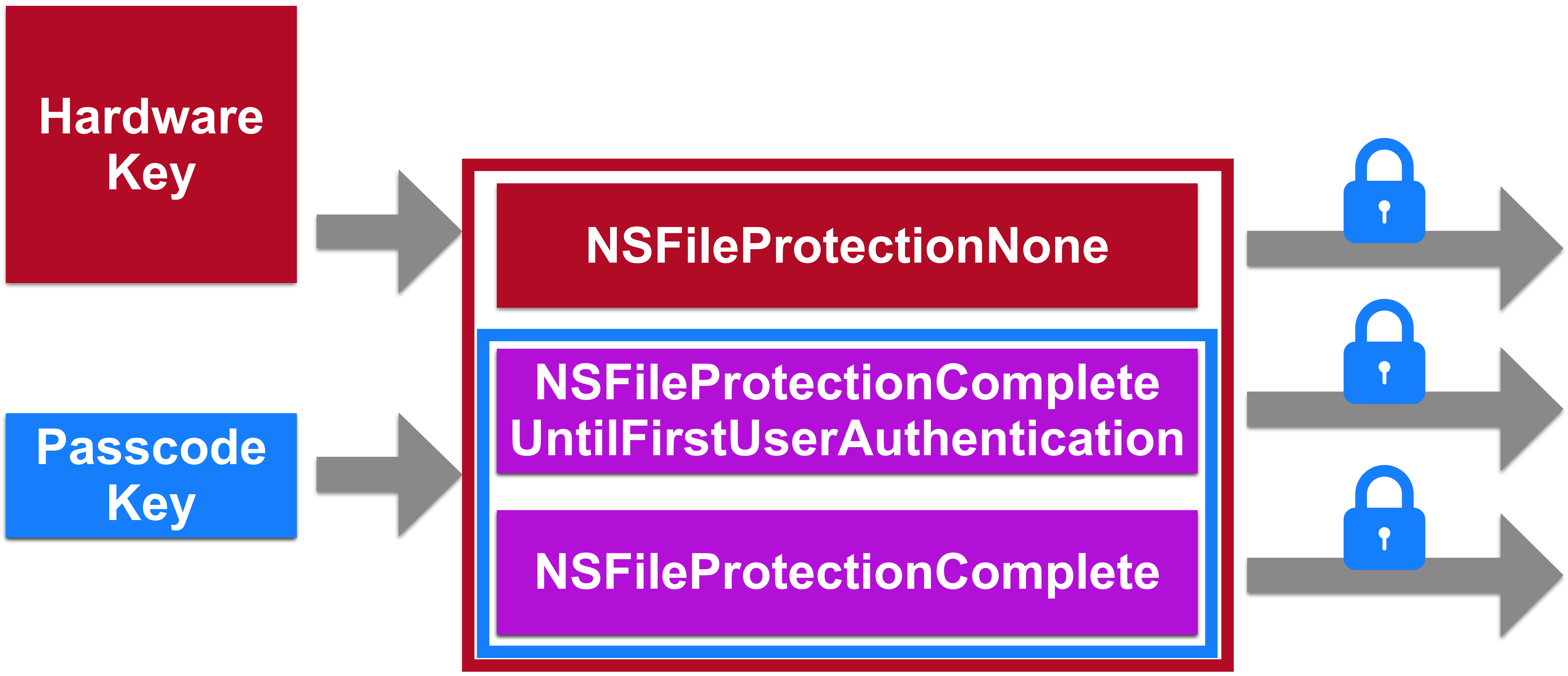
[3]

iOS Encryption Hierarchy

[3]

iOS Encryption Hierarchy

nccgroup
freedom from doubt



[3]

iOS Keychain

nccgroup
freedom from doubt

Structured Data Store

- Lives in SQLite database
- Entries individually encrypted

Main Criticism

- Data not deleted when app is uninstalled!

Keychain

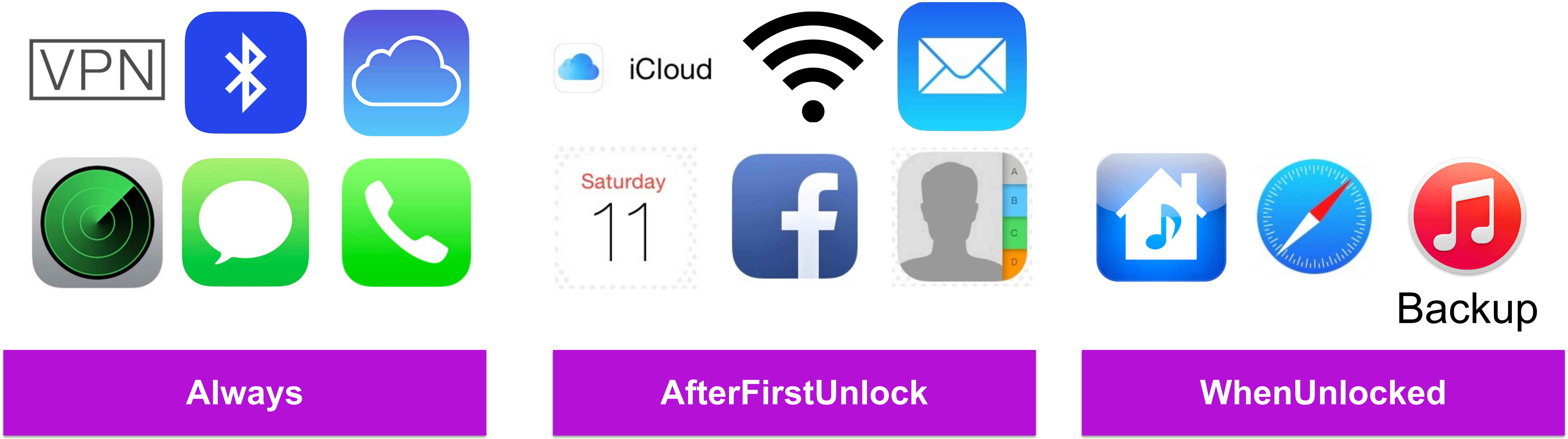
File Protection (NSFileProtection)	Keychain Class (kSecAttrAccessible)	Effect
None	Always	No protection.
UntilFirstUserAuthentication	AfterFirstUnlock	Protected from boot until user unlocks.
Complete	WhenUnlocked	Protected when device is locked.
N/A	WhenPasscodeSet	Only store if passcode is set.

[4]

Usability vs. Security

Data Accessibility

Some data must be accessible when device is in use



Tackling Usability

TouchID

- Usability feature
- Controlled by Secure Enclave
- Encourages users to set passcode
- Simply protects passcode-based key

https://www.youtube.com/watch?v=vI3OvT4b-sA

Advanced Controls

User Presence for Keychain

Requires users to enter Passcode (or TouchID)

Local Authentication

OS-level API

Not tied-in with crypto

Bypassable when jailbroken [5]

Use Keychain User Presence instead

Security Threats - Jailbroken

nccgroup
freedom from doubt

Jailbreaks Do

Allow execution of unsigned code

Disable some OS-level protections

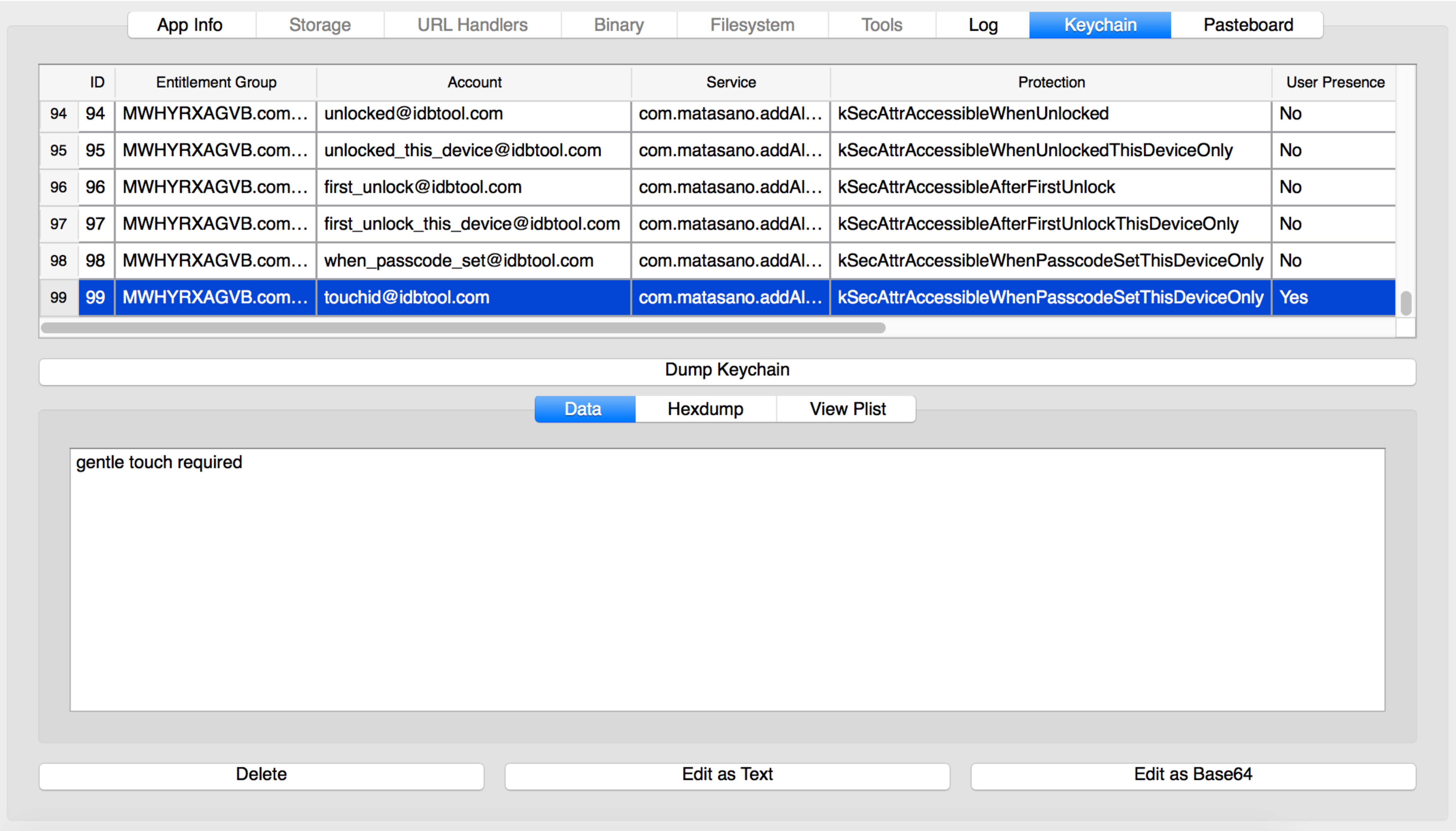
Jailbreaks Don't

Disable Sandboxing for
App Store apps

What About Secure Storage?

Passcode may prevent public jailbreaks

Access to all non-protected data



http://idbtool.com

Security Threats - Non-Jailbroken

Malicious Applications

- Asking for access to personal data
- Apps attacking other apps via IPC mechanisms

Evil Maid-Style Attacks

- Jailbreak device
- Backdoor OS / App

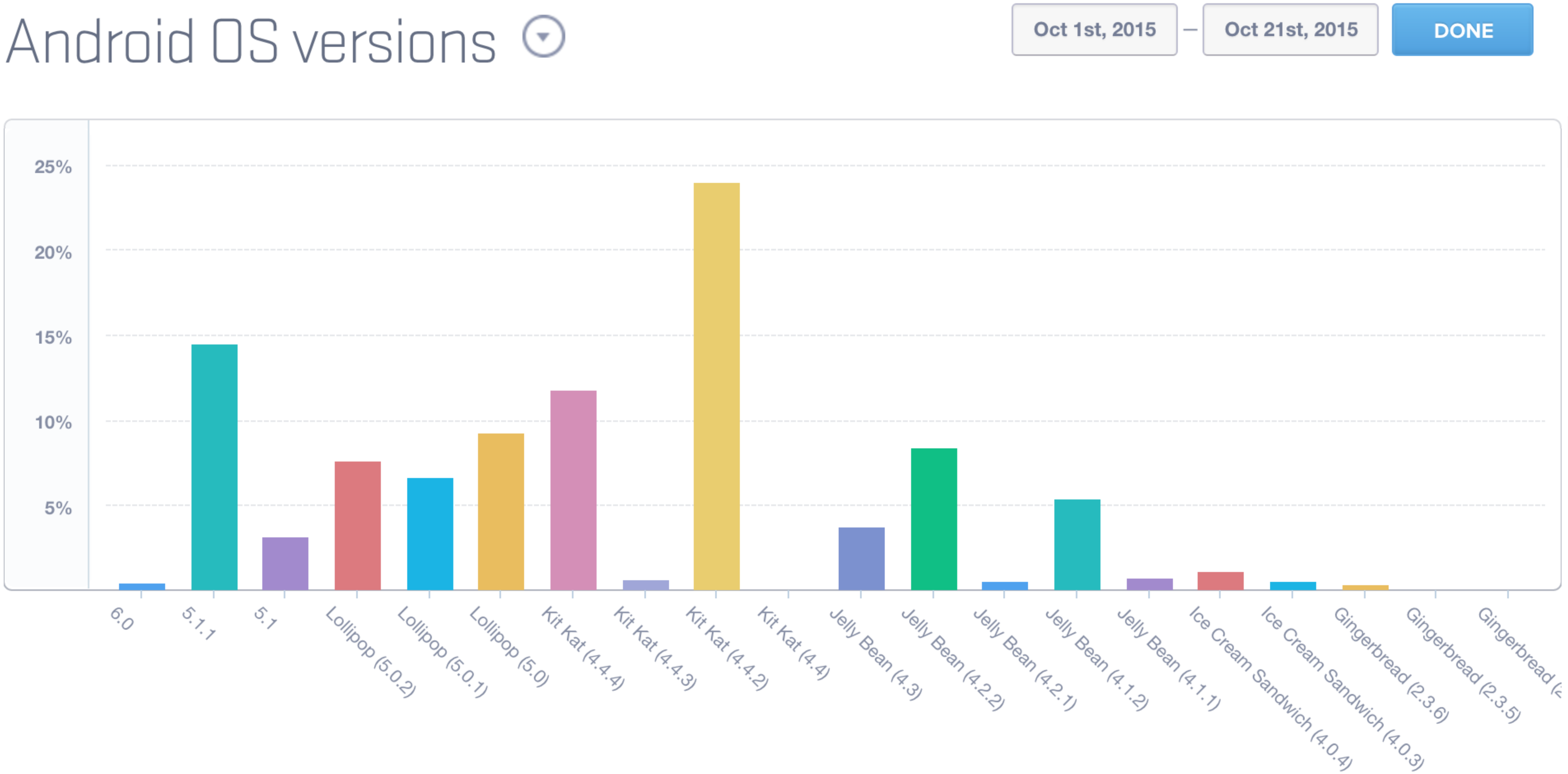
Secure Data Storage
…on Android

Evolution of Android Security

freedom from doubt

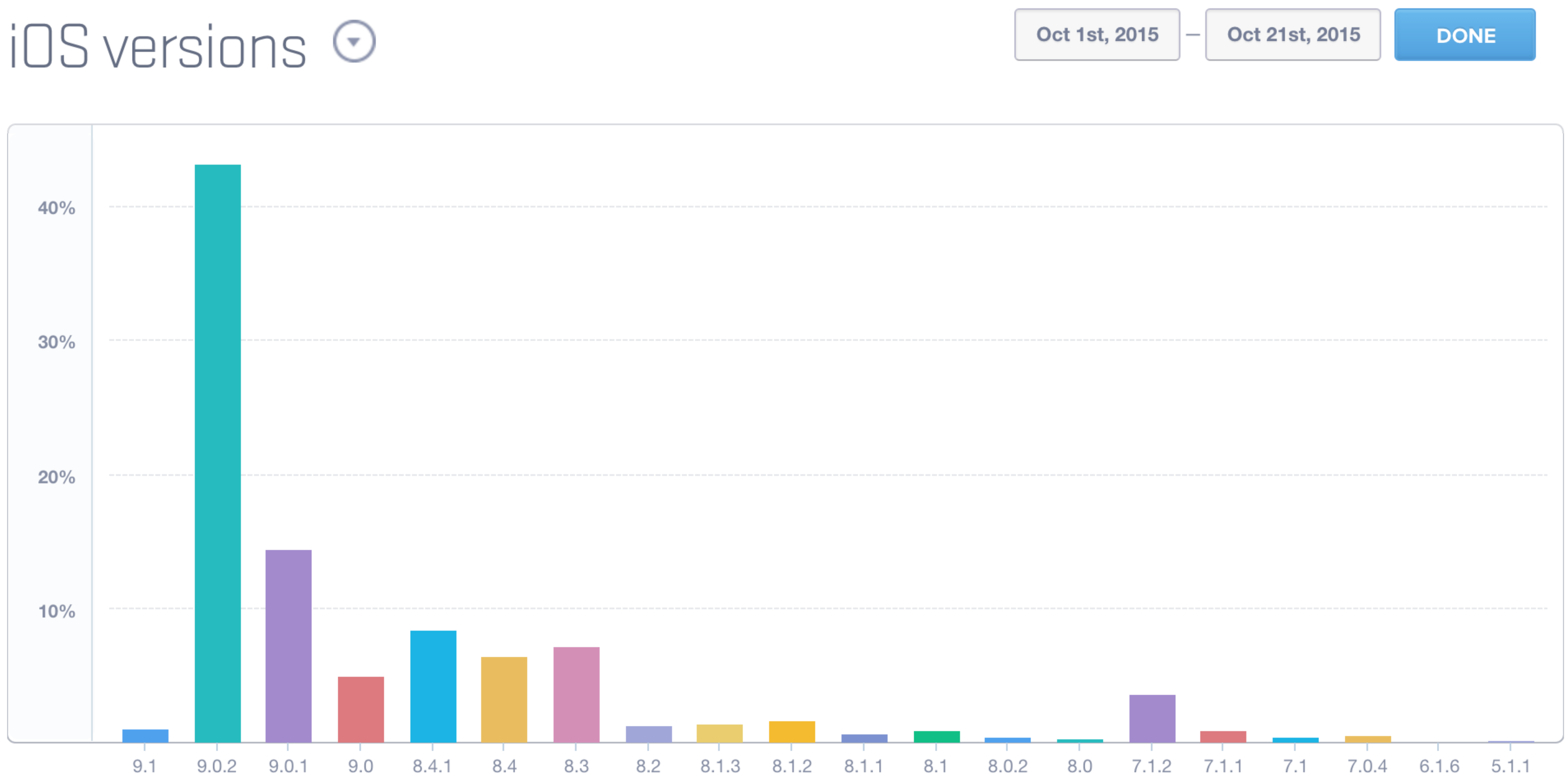
Feature	4.0	4.1	4.2	4.3	4.4	5.x
ASLR	X	X	X	X	X	X
DEP/PIE		X	X	X	X	X
Restricted `logcat`		X	X	X	X	X
Restricted `adb`			X	X	X	X
Manifest Export Security			X	X	X	X
Secure Random from OpenSSL			X	X	X	X
Untrusted Application Malware Scanning			X	X	X	X
SELinux (Permissive)				X	X	X
SELinux (Enforcing)					X	X
KeyStore Hidden Keys*				X	X	X
No `setuid/getuid, nosuid`				X	X	X
Text Relocation Protection				X	X	X
`dm-verity`					X	X
TEE signing of KEK						X
`forceencrypt`						X*

Adoption of Android Security



[6]

Flash back to iOS Adoption..



[7]

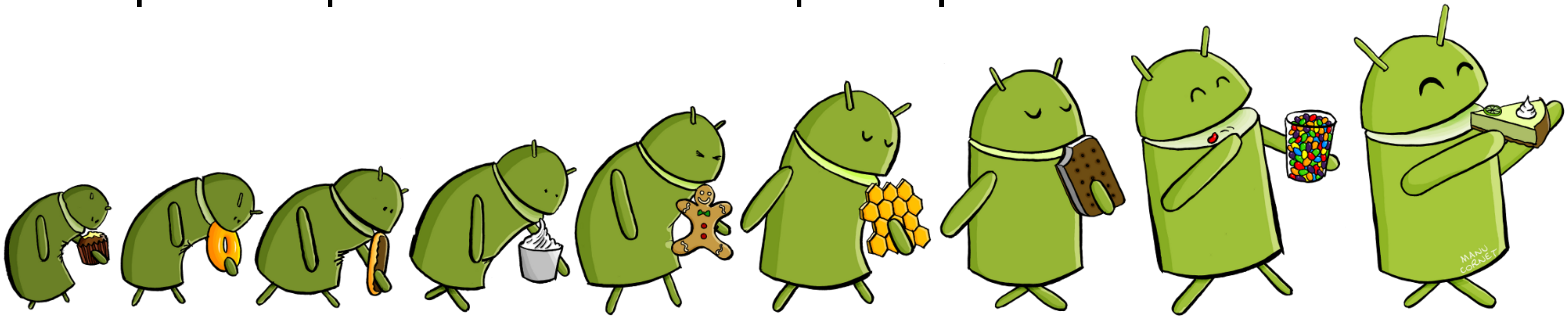
Impact on Application Devs

Developers face different platform versions and security APIs

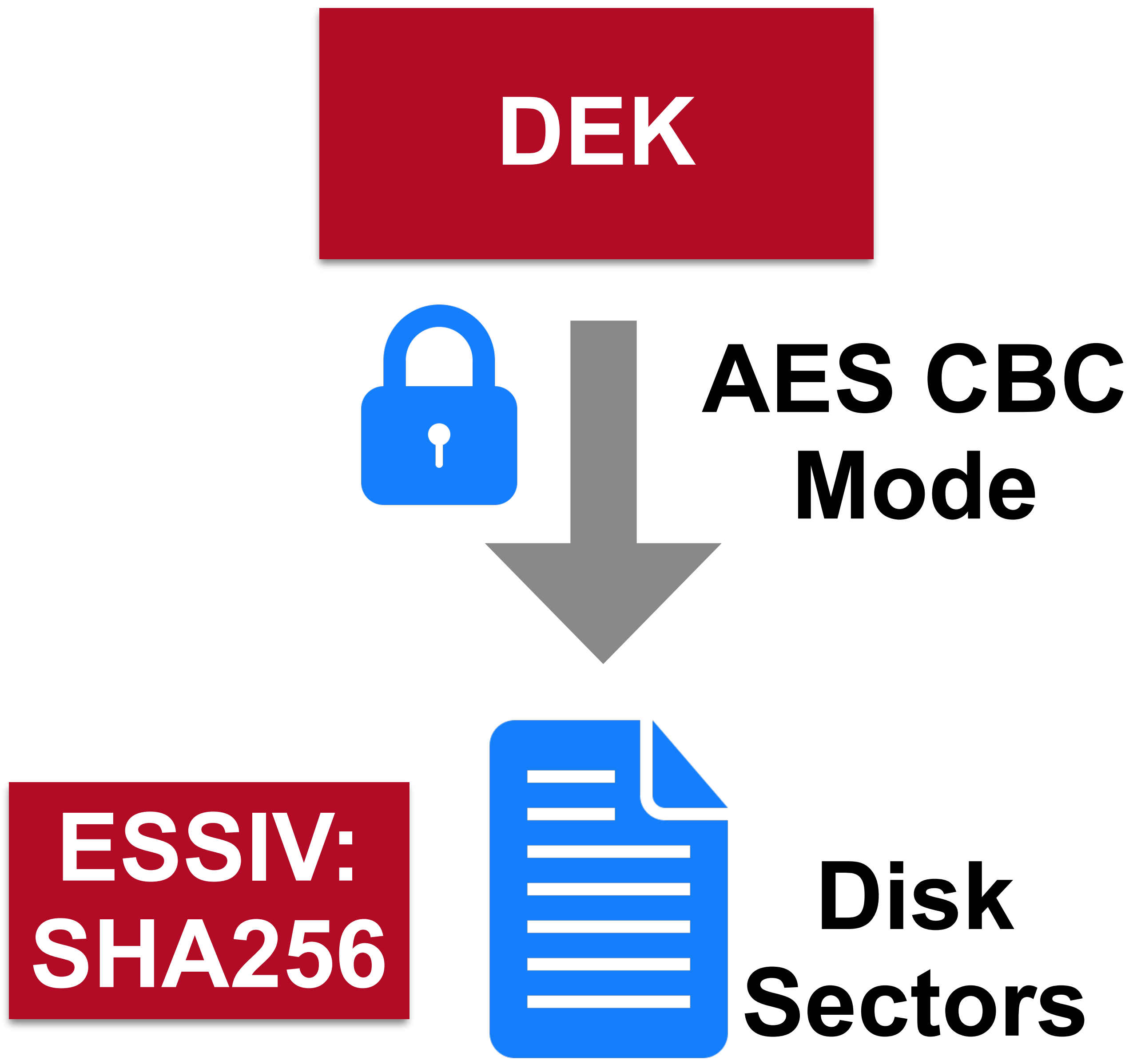
- Code complexity and inconsistent behavior
- Access to more secure functionality is not available for all users

Security improvements available via latest version

- Complicated problem of an OTA update process

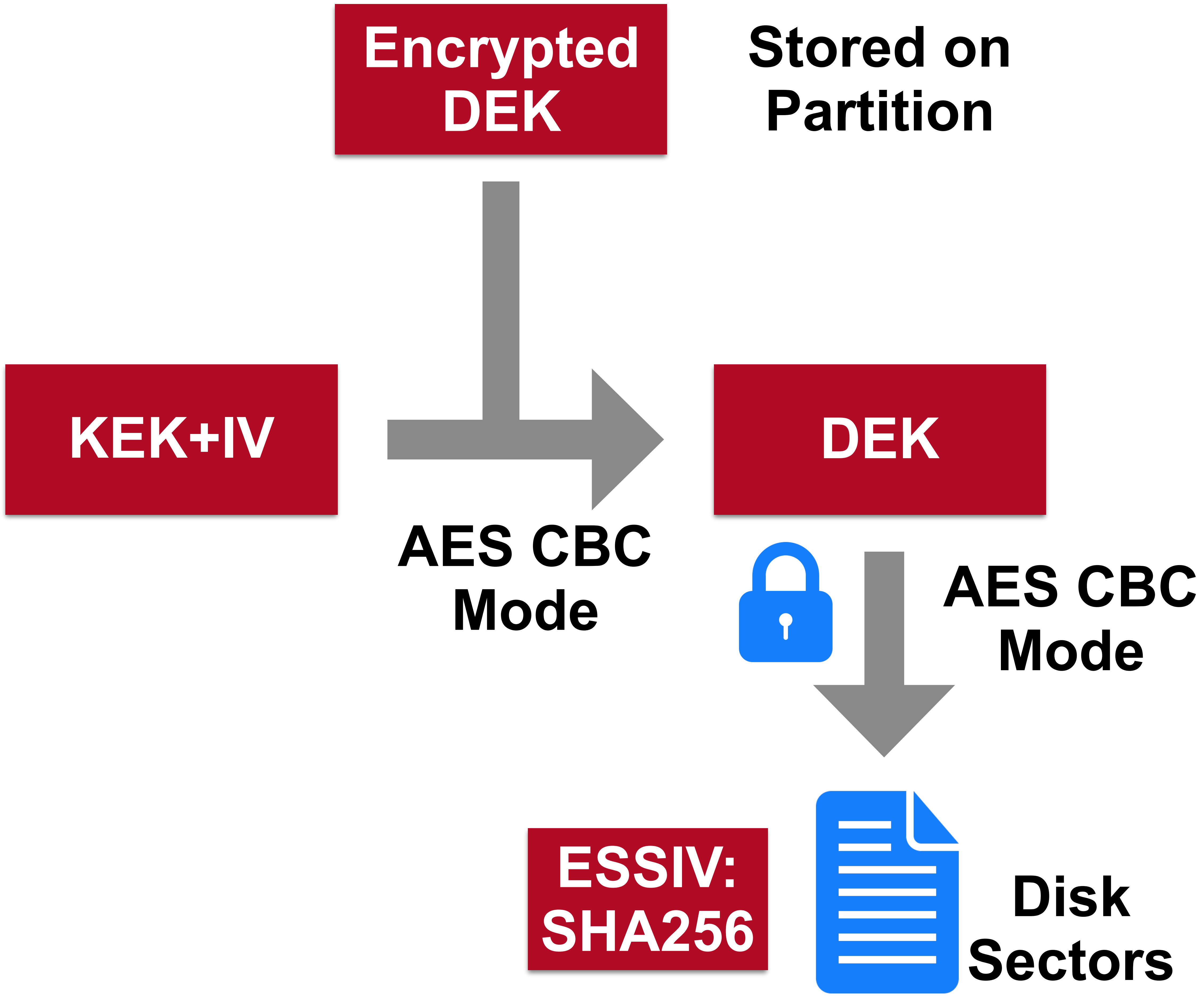


How Android Encryption Works



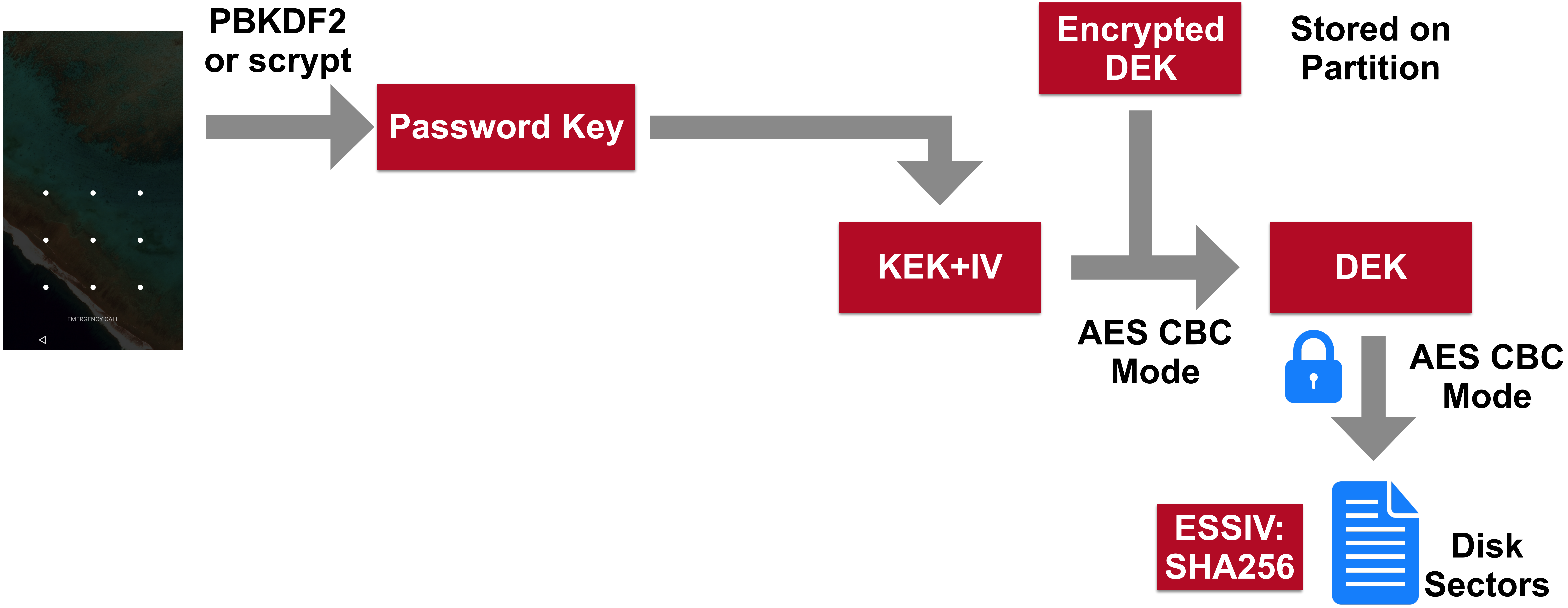
[8,9]

How Android Encryption Works



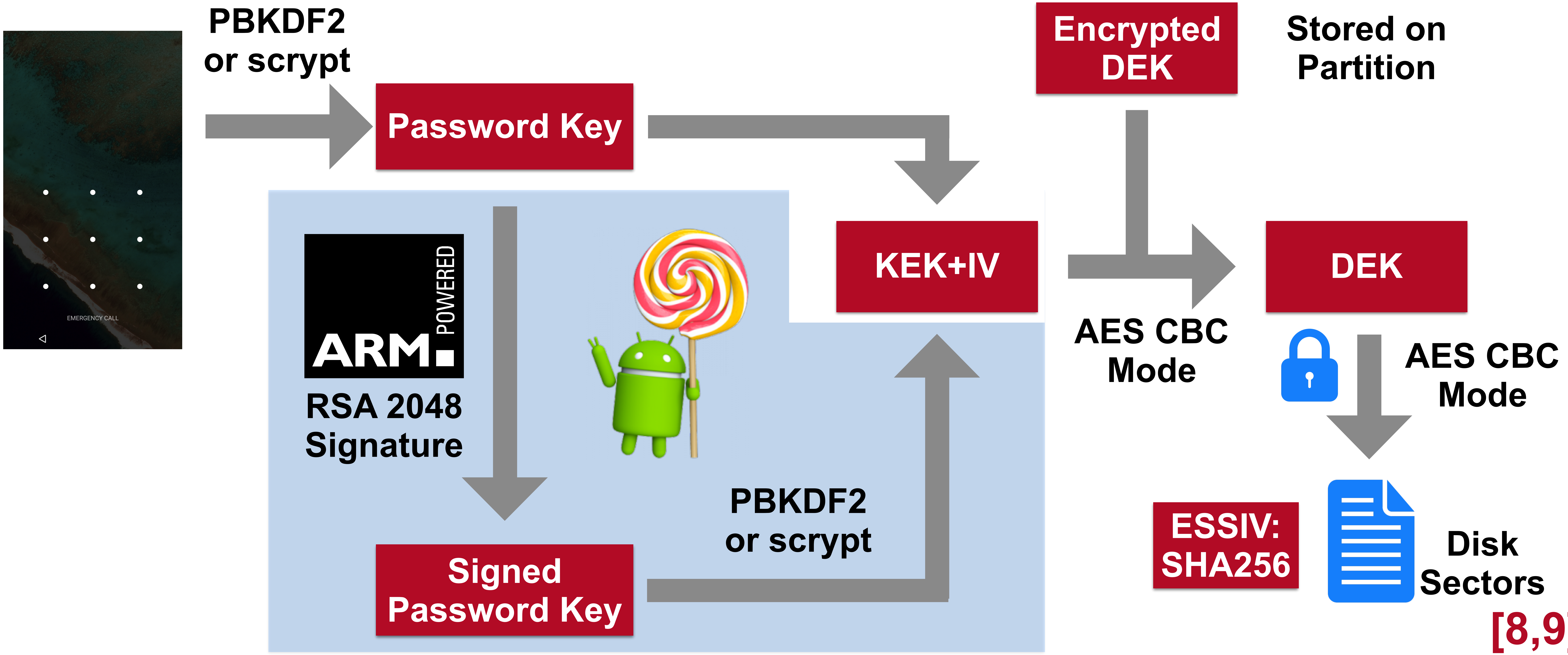
[8,9]

How Android Encryption Works



[8,9]

How Android Encryption Works



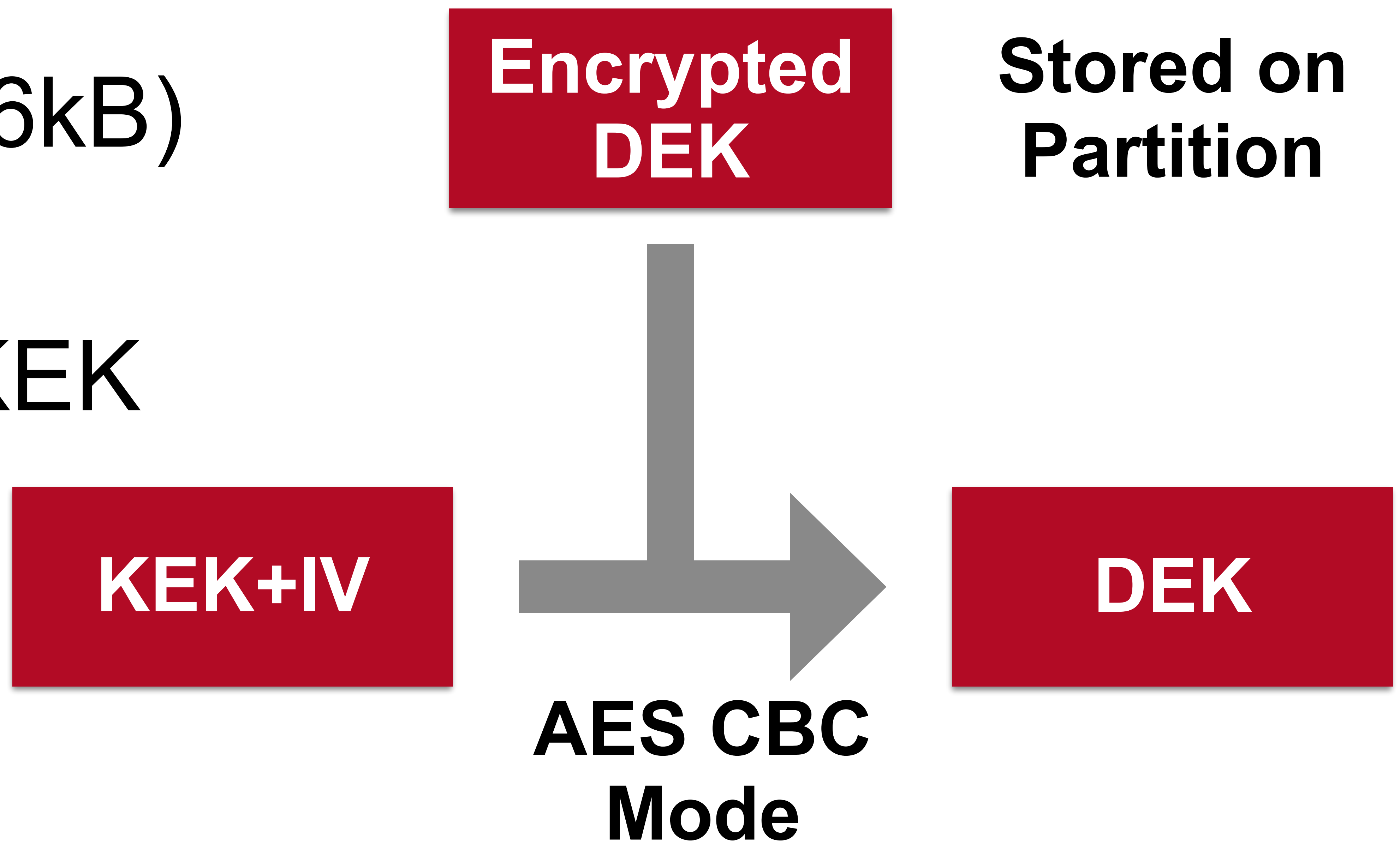
[8,9]

How Android Encryption Works

This protection only covers the userdata partition

Crypto footer

- Carved out of end of userdata partition (-16kB)
- Sometimes there is a dedicated partition
- Master key stored here encrypted by the KEK



LUKS-ish but not quite.

- Footer can only hold one decryption key

[8,9]

Android Credential Storage

System Credential Store allows for storage of

- VPN Keys
- WiFi
- Asymmetric keys

Encrypted by key derived from user's passcode

Can be hardware backed

- Private keys non-extractable, even as root
- Requires use of device in attack

Issues with KeyStore

- Inconsistent protections available to developers
- Unclear documentation and erratic behavior causes keys to be wiped (fixed in 5.0)
- Improved with Marshmallow

[10]

A look at Marshmallow changes

scrypt hashing of unlock passcode values

- Replaces weaker SHA-1/MD5 hash concatenation

Additional KeyStore improvements

- Added support to store symmetrical keys (without private API)
- Documented and refined KeyStore wipe behavior
- Additional properties for keys
  - Prevent unsafe modes (fixed IV’s, ECB mode, etc)
  - Explicitly define a key type

Nexus Imprint, etc

Allows for a more complex passwords

Secure payments, unlock capabilities

Stored securely in TEE

Sets defined standards for other OEMs

Google & OEMs

Wild inconsistencies among devices

- Boot loader security
- Hardware backed crypto storage
- TEE / TrustZone
- Boot image type

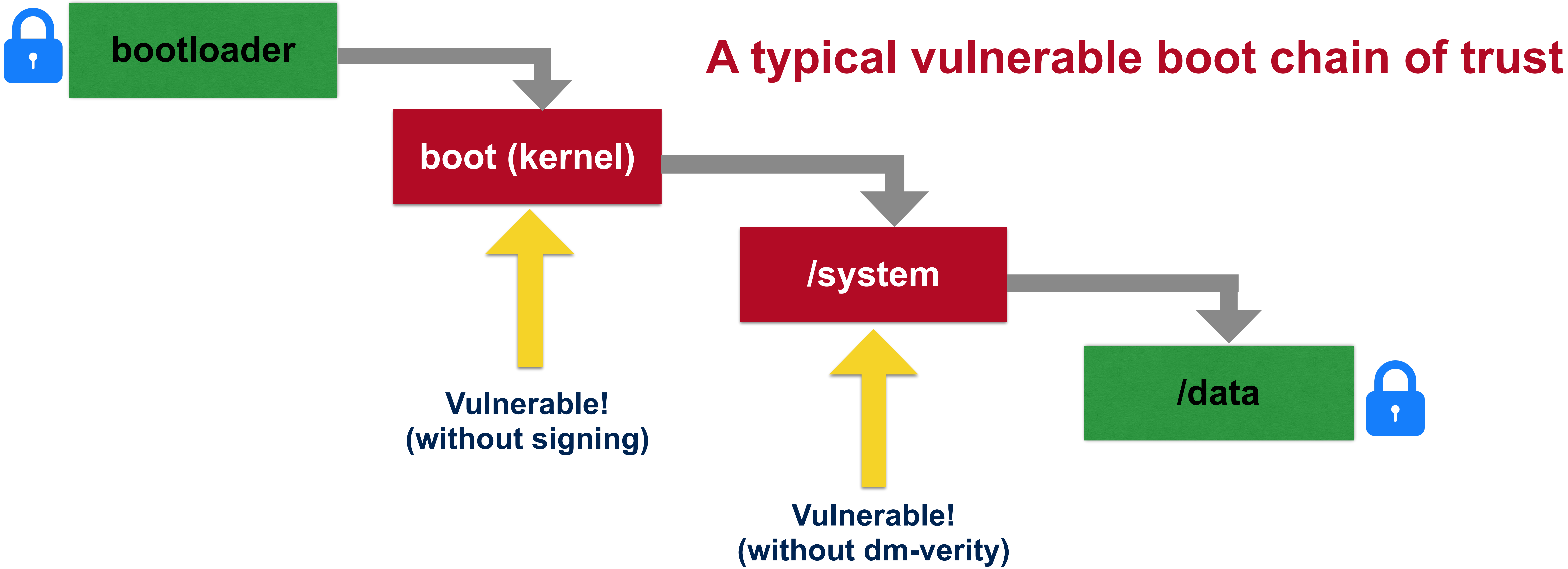
Different OEMs offer different protection schemes

- eMMC write protection
- Boot image signature verification
- Locked, locked but unlockable, permissive by default

Difficult problem to solve

- Challenging for Google to enforce consistent protections on the OEMs
- Apple has a distinct advantage in controlling the whole stack

Importance of Boot Security



Download Mode

Samsung specific boot loader interface for their Android devices

Internally, Samsung uses a tool called ODIN

- Interacts with the device and flash firmware images
- Check out heimdall if you want a cross-platform, open source version

Overly permissive!

Most devices allow direct write access!

- Except for a few US carrier protected models
- (Boot image signature verification)

[11,12]

lk (little kernel) Bootloader

nccgroup
freedom from doubt

Issues with lk used on many devices

“Fastboot boot command bypasses signature verification (CVE-2014-4325)” [13]

“Incomplete signature parsing during boot image authentication leads to signature forgery (CVE-2014-0973)” [14]

“Improper partitions bounds checking when flashing sparse images (CVE-2015-0567)” [15]

laf

Bootable partition named laf found on many LG devices

Communication via Send_Command binary (Windows)

- Also available as python script for all platforms
- Drops into a root shell
- Flash new images from shell

Fixed? Not quite.

- /dev/block/mmcblk0p1 - protected
- /dev/block/mmcblk0 - not protected
- dd + seek :)

[16]

Let’s revisit:
“FDE protects data when device is turned off”

Mobile "Evil Maid" Attacks

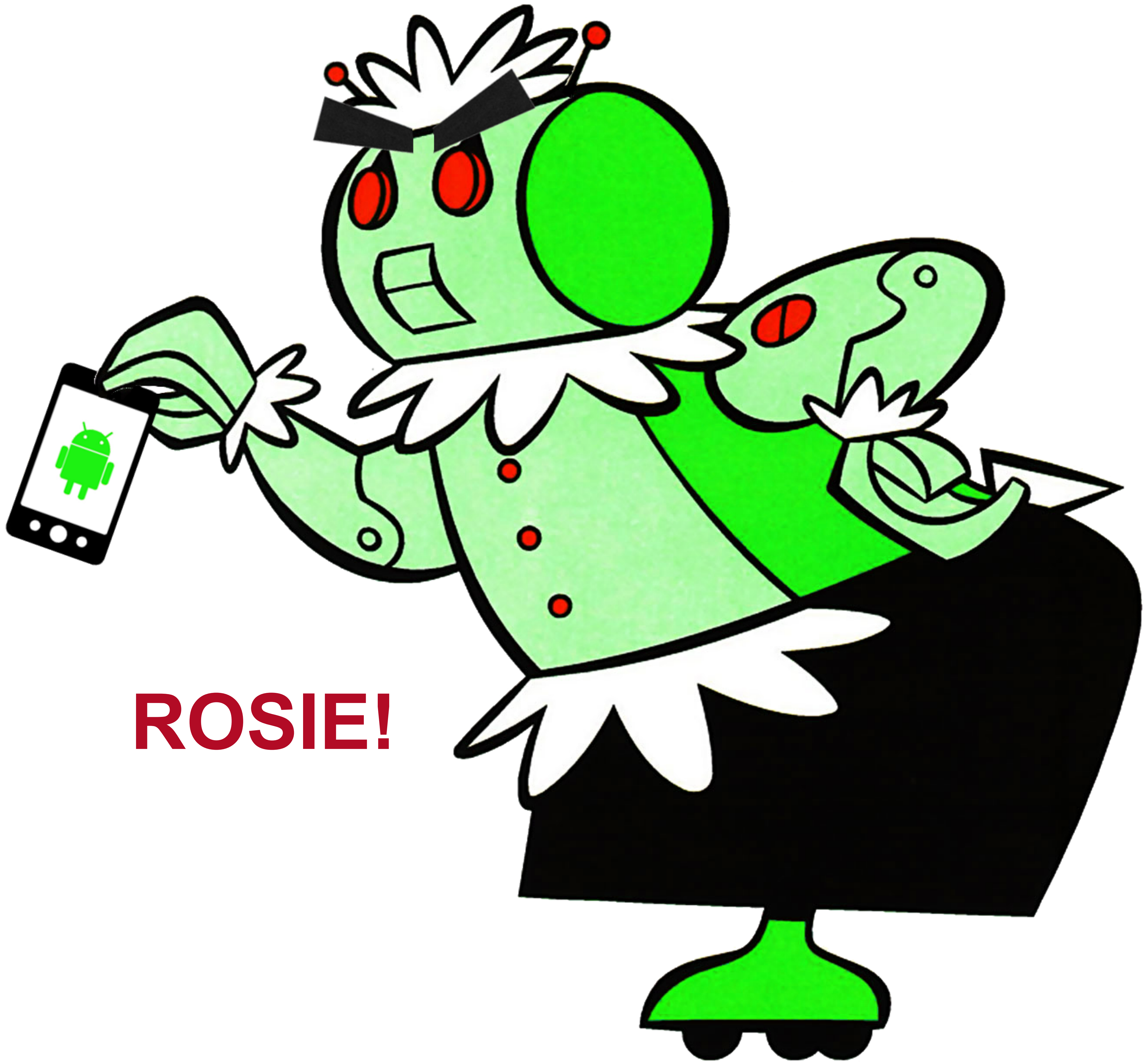
Exploit permissive bootloader

- Flash custom boot image
- Backdoor in kernel in image
- < 2 minutes (including reboots!)

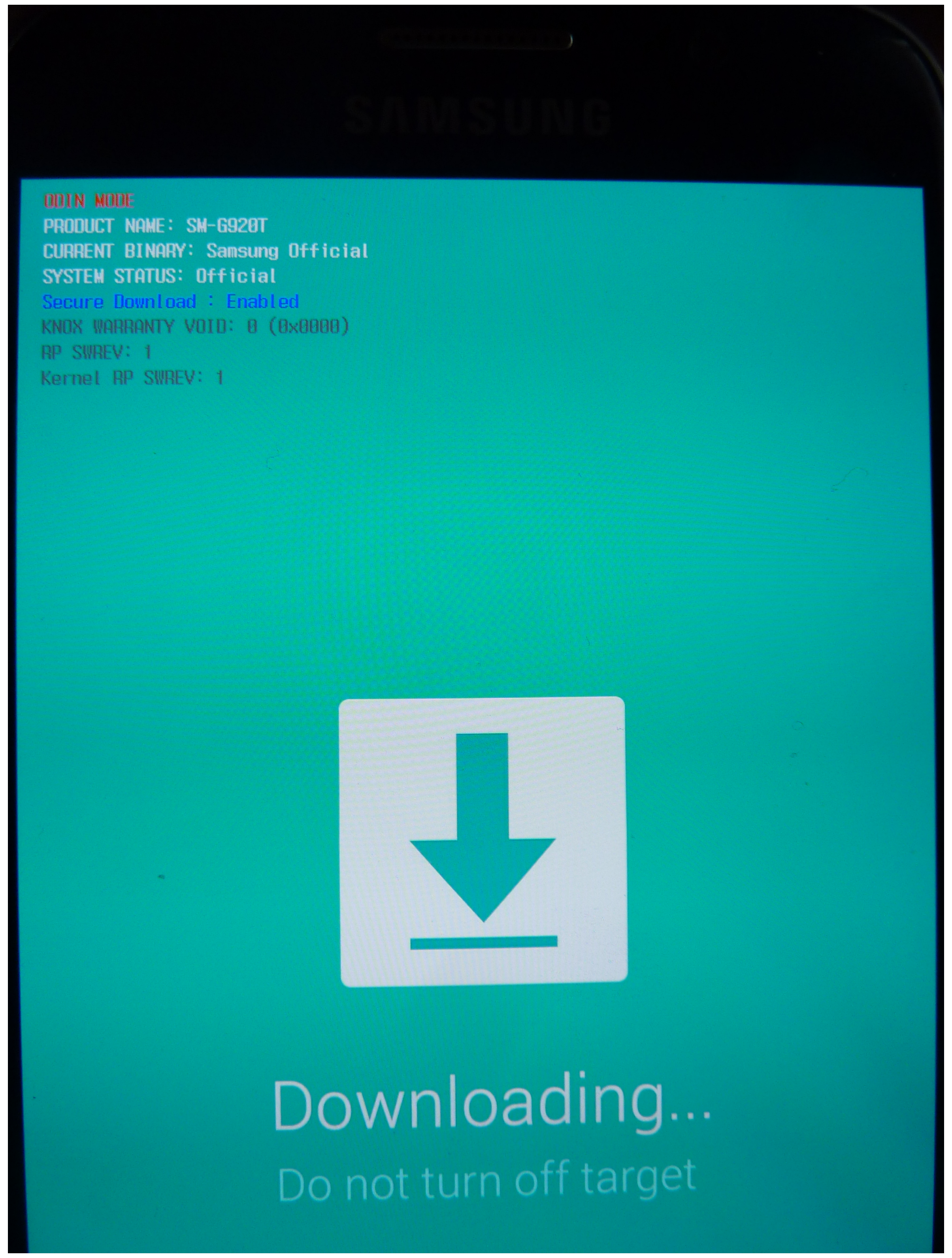
Give device back to user

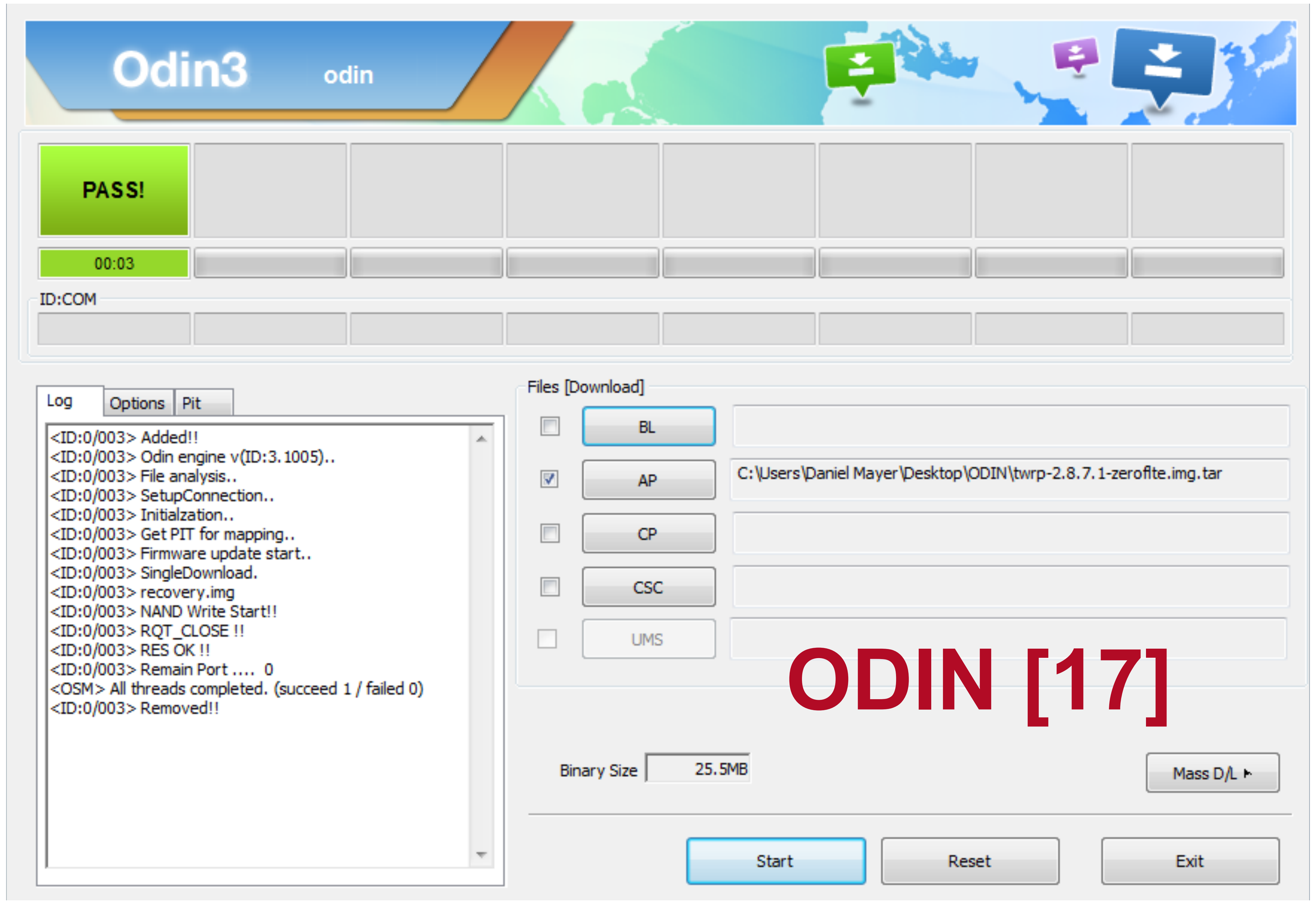
Profit!

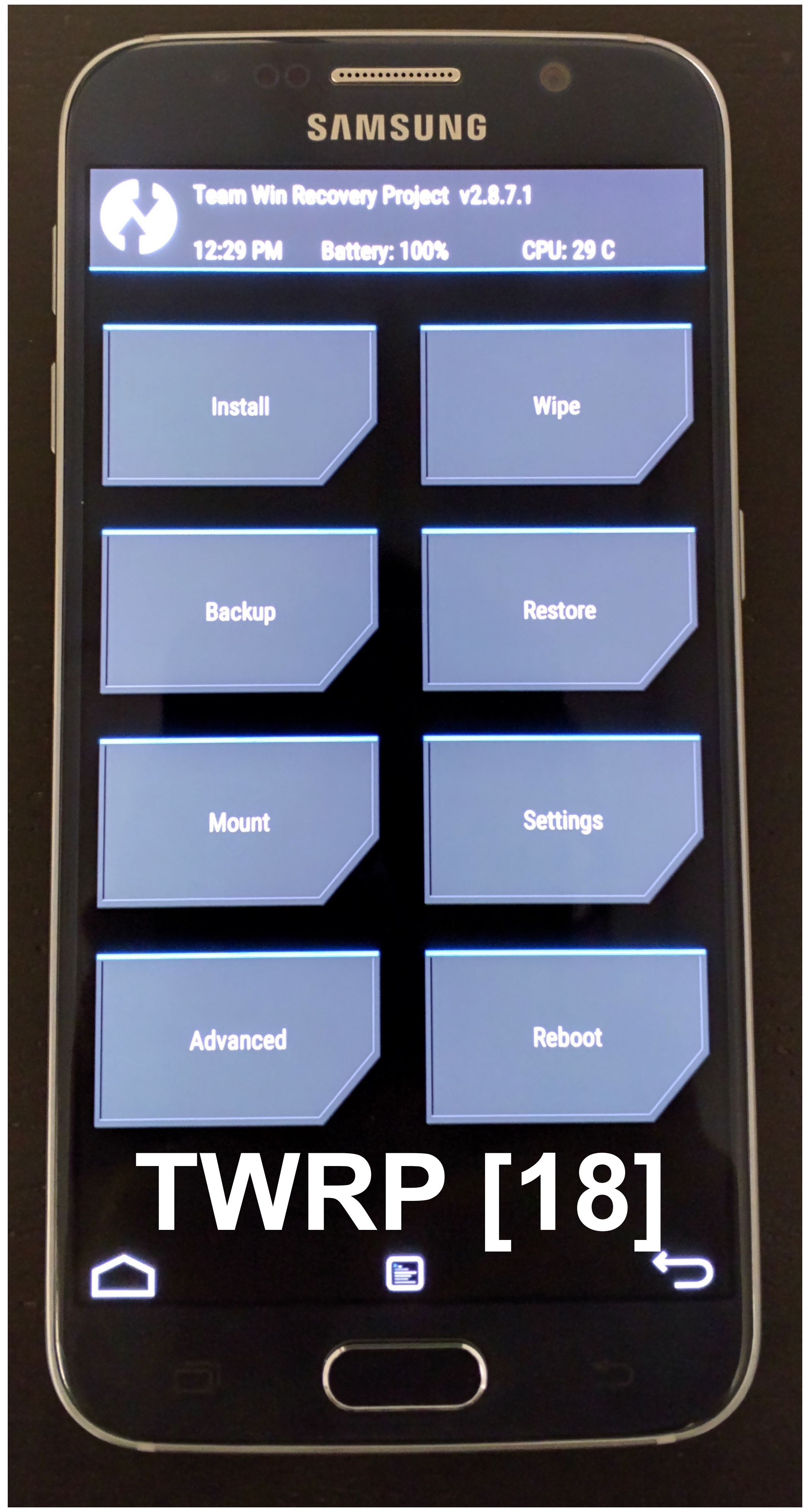
- Get encryption key…
- …or data exfiltration
- …or shells



Dev Step 1: Flash Recovery



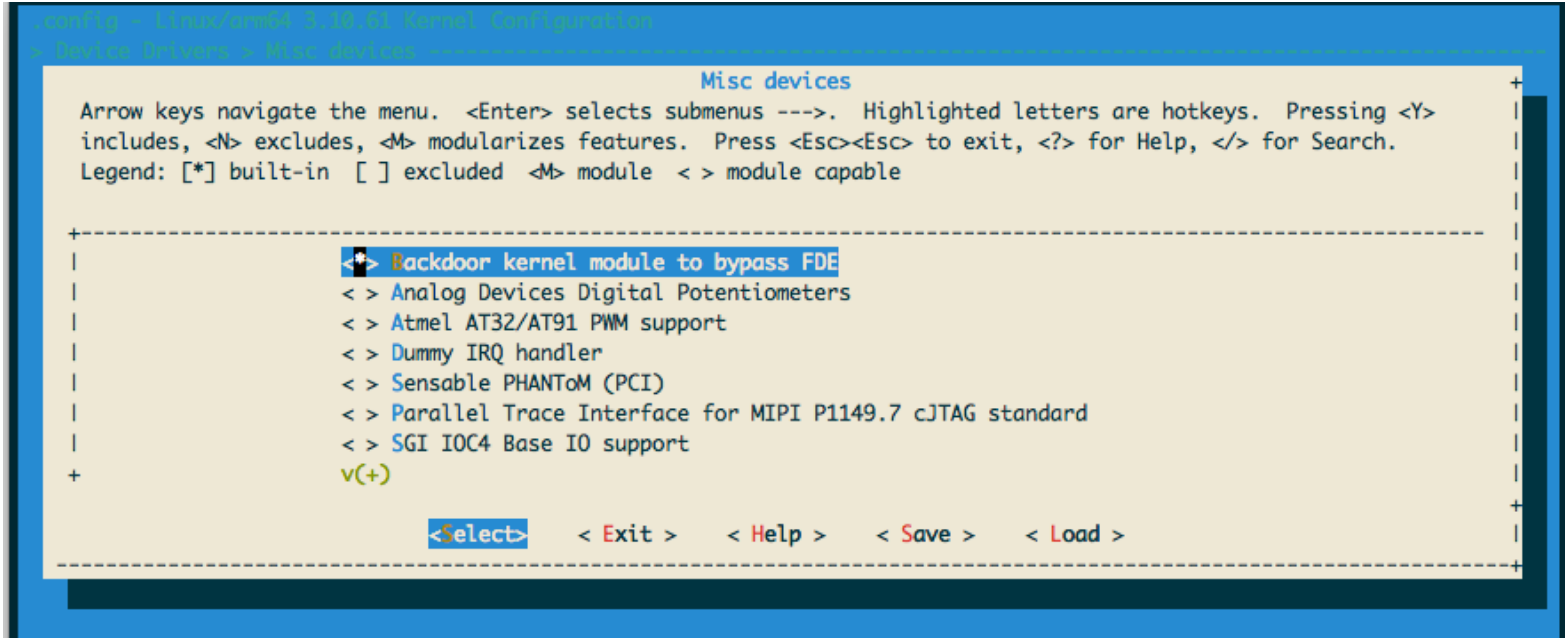




For more info on recovery… [19]

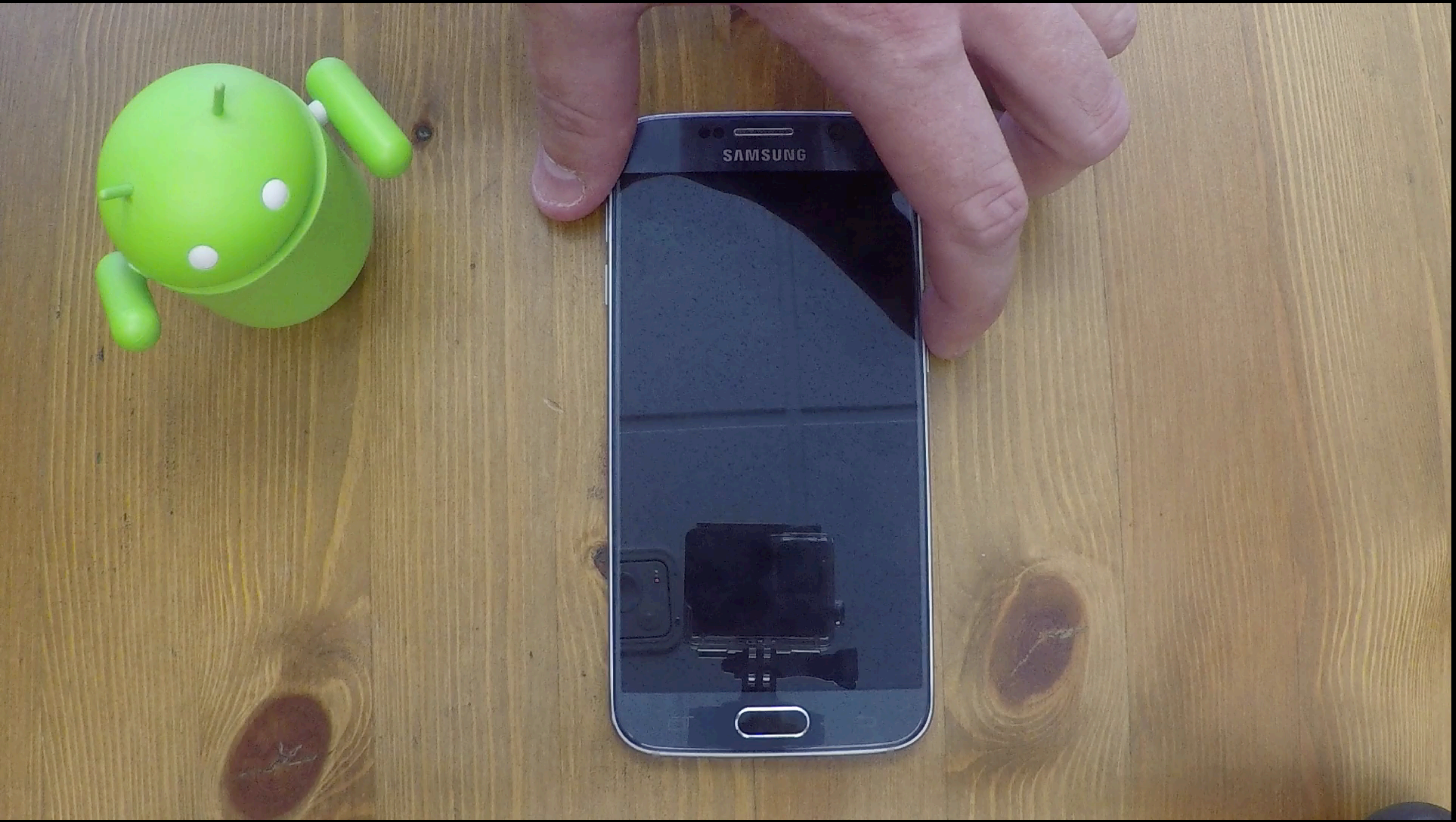
Dev Step 2: Backdoor the Kernel

nccgroup
freedom from doubt



Dev Step 3: Test Exploit

1. **Compile backdoored kernal**
2. **Create boot image**
3. **Flash boot image via recovery**
4. **Reboot and test**


SAMSUNG

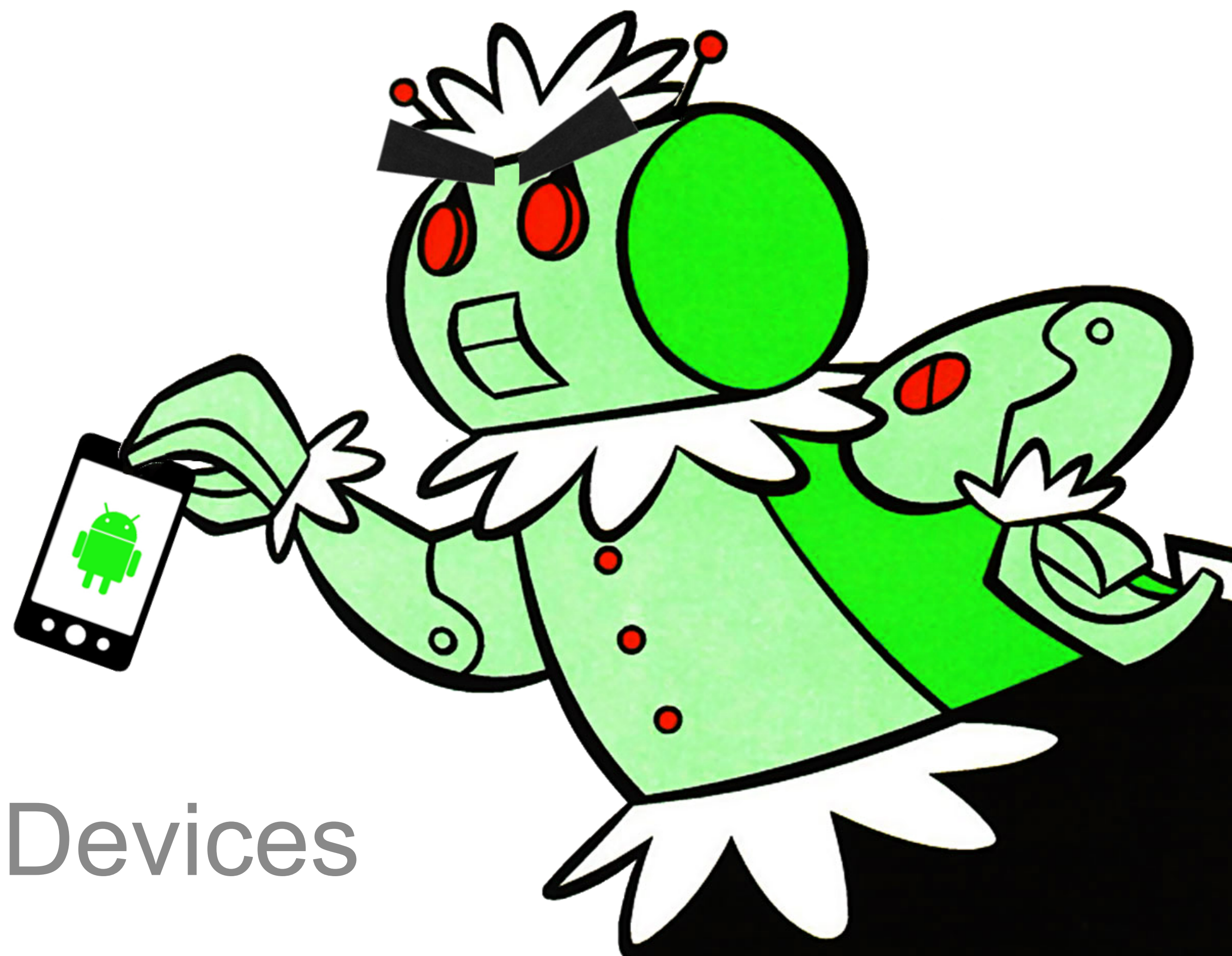
The Attack: Review

Possible on a number of OEM devices

This is not a new problem

Google provides mechanisms to prevent this

Similar attack possible in iOS, but requires jailbreak



A penny for your thoughts…?

Secure configurations by default!

Responsible bootloader unlock capabilities

Clearly documented security guarantees

Consistency among OEM partners

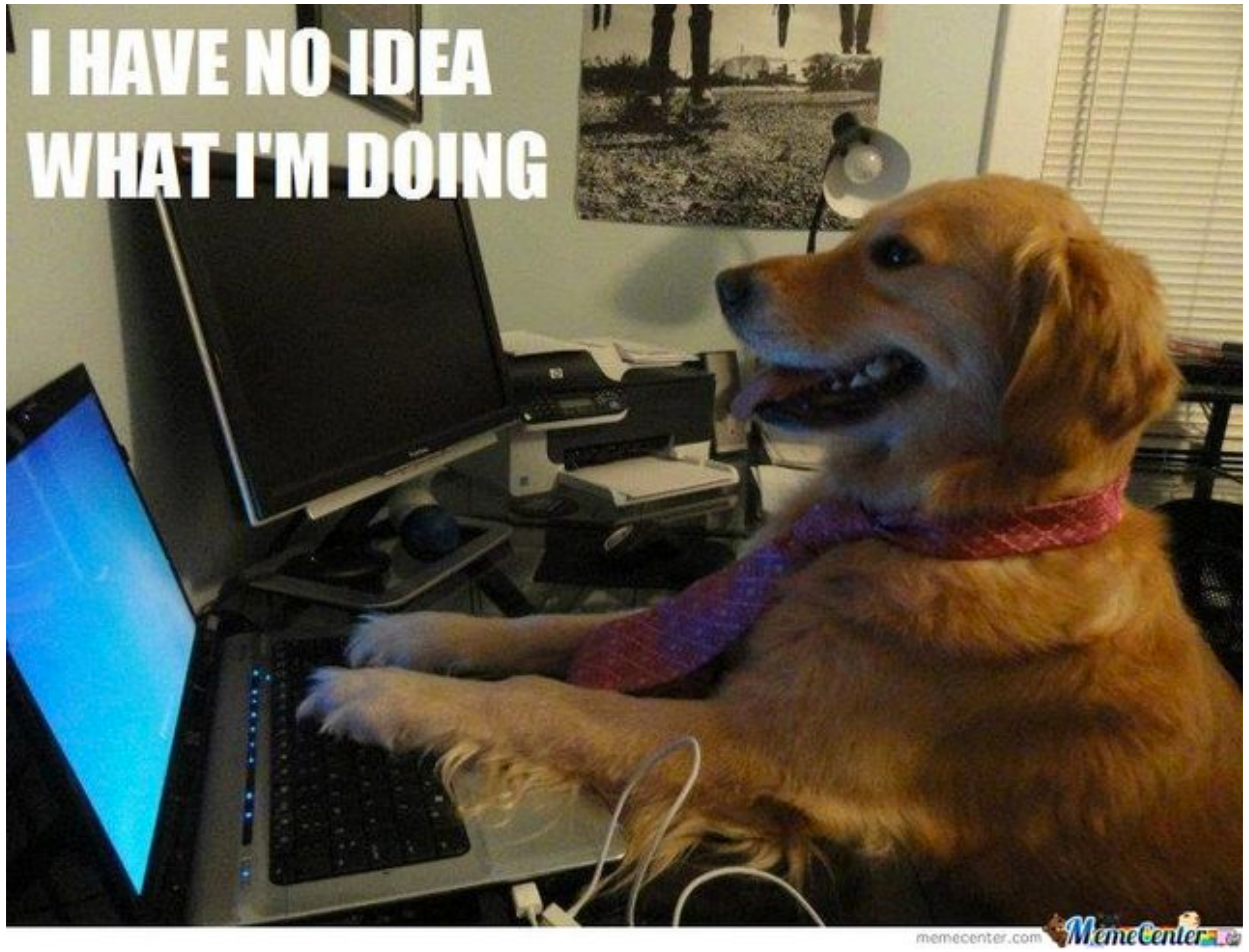
“Alternatives” to Platform Security

No Password? No Problem!?

What if users may not have set passcodes?

Custom App Sandboxes

- Add passcode to app
- Derive encryption key
- Encrypt data
- Wipe key!



Challenges

- Crypto is hard! [20]
- Not hardware backed, no brute-force protection

Online Apps

No Offline Storage

- Does data need to be offline?
- Consider storing server-side

Usability

- Login each time
- Long-lived token, back to storage problem

Where does this leave us?

Best Practices for Users

General

- Set a (strong) passcode!
- Use the latest OS available for your hardware

iOS

- Enable (remote) wipe

Android

- Choose your phone wisely
- Encrypt your device

Best Practices for Developers

General

Determine if data has to be stored locally

Case by case situation…

Android

Relying on platform security is challenging

Discussion: supporting old versions of Android

iOS

Use protection class that requires passcode

Warn user when no passcode is set

The Road Ahead

Usability

For Users

Beyond Passwords

Biometrics

For Developers

Consistency in platform

With sane, documented defaults

Black Hat Sound Bytes

1. **Security controls should be balanced with data sensitivity and threat model.**

2. **Protect data until access is actually needed.**

3. **Secure storage relies on the entire stack being secured.**

References

[1] Consumer Reports. Smart phone thefts rose to 3.1 million last year, Consumer Reports finds, May 2014

[2] http://www.engadget.com/2012/04/09/us-carriers-agree-to-build-stolen-phone-database-and-blacklist/

[3] Apple Inc. iOS Security - iOS 8.3 or later. https://www.apple.com/privacy/docs/iOS_Security_ Guide_Oct_2014.pdf, April 2015

[4] Apple Inc. Keychain Services Reference. https://developer.apple.com/library/ios/documentation/ Security/Reference/keychainservices/index.html, 2015

[5] SuccessID - TouchID override & simulation - https://hexplo.it/successid-touchid-override-simulation/

[6] https://mixpanel.com/trends/#report/android_frag

[7] https://mixpanel.com/trends/#report/ios_frag

[8] Android Security Internals: An In-Depth Guide to Android's Security Architecture, Elenkov, N., No Starch Press

[9[Android Explorations, Elenkov, N., http://nelenkov.blogspot.com/

[10] Google. Android Keystore Changes. https://developer.android.com/preview/behavior-changes. html#behavior-keystore.

[11] http://wiki.cyanogenmod.org/w/Template:Samsung_install

[12] http://forum.xda-developers.com/showthread.php?t=810130

[13] https://www.codeaurora.org/projects/security-advisories/fastboot-boot-command-bypasses-signature-verification-cve-2014-4325

[14] https://www.codeaurora.org/projects/security-advisories/incomplete-signature-parsing-during-boot-image-authentication-leads-to-signature-forgery-cve-2014-0973

[15] https://www.codeaurora.org/projects/security-advisories/lk-improper-partition-bounds-checking-when-flashing-sparse-images-cve

[16] http://forum.xda-developers.com/android/development/guide-root-method-lg-devices-t3049772

[17] http://forum.xda-developers.com/galaxy-s3/themes-apps/27-08-2013-odin-3-09-odin-1-85-versions-t2189539

[18] https://twrp.me/

[19] https://youtu.be/5W_s--lSqyo - Making Androids Bootable Recovery Work For You, Drew Suarez

[20] the matasano crypto challenges, http://cryptopals.com/

¡Gracias!

Questions?

Daniel A. Mayer — @DanlAMayer

Drew Suarez — @utkan0s

Slidedeck: https://speakerdeck.com/utkanos