A (not-so-quick) Primer on iOS Encryption

David Schuetz
Senior Consultant, NCC Group

@DarthNull
david.schuetz@nccgroup.trust

Background

Ancient History

nccgroup

- October 2014: "Apple's commitment to your privacy"
  - Changes in iOS 8
  - "Apple cannot bypass your passcode"
  - "…not technically feasible…to respond to government warrants"
- Raised lots of questions:
  - What does that mean? What did they do before?
  - What about other attacks? Forensics?

- Suddenly got a lot more important

Apple "deluged" by police

- CNET, May 2013, claims "Apple can bypass the security software":
  - Big backlog (7 weeks, one case took 4 months)
- Plus Kentucky, NY, San Bernardino, etc.

CNET › Tech Industry › Apple deluged by police demands to decrypt iPhones

Apple deluged by police demands to decrypt iPhones

ATF says no law enforcement agency could unlock a defendant's iPhone, but Apple can "bypass the security software" if it chooses. Apple has created a police waiting list because of high demand.

by Declan McCullagh @declanm / May 10, 2013 4:00 AM PDT

What does it MEAN?!?

- Backlog implies:
  - Can’t just plug in and use a magic key
  - Could brute force passcodes, conceivably

- “Apple can afford a LOT of GPU crackers…”
  - It doesn’t work that way

So how does iOS encryption work?

- It's complicated, but also fairly comprehensive
- Some early details figured out by researchers
  - Examining and understanding published APIs
  - Reverse engineering, breaking

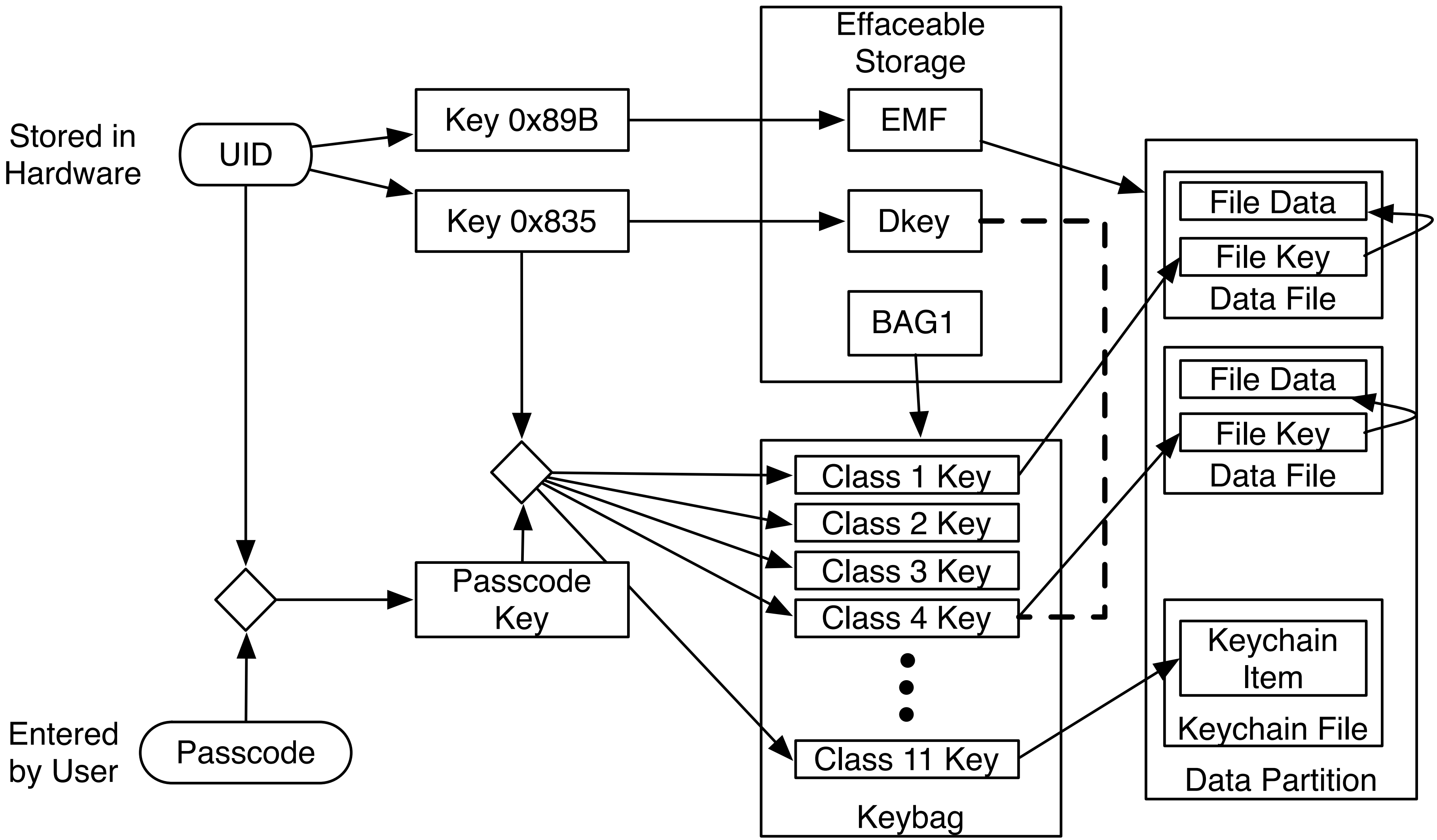
- Apple publishes an "iOS Security" paper
  - Beginning in May 2012
  - Updated multiple times since then
  - Covers encryption, Apple Pay, lots of other things

- This talk focuses on Encryption

Basics of iOS Encryption

How iOS encryption works

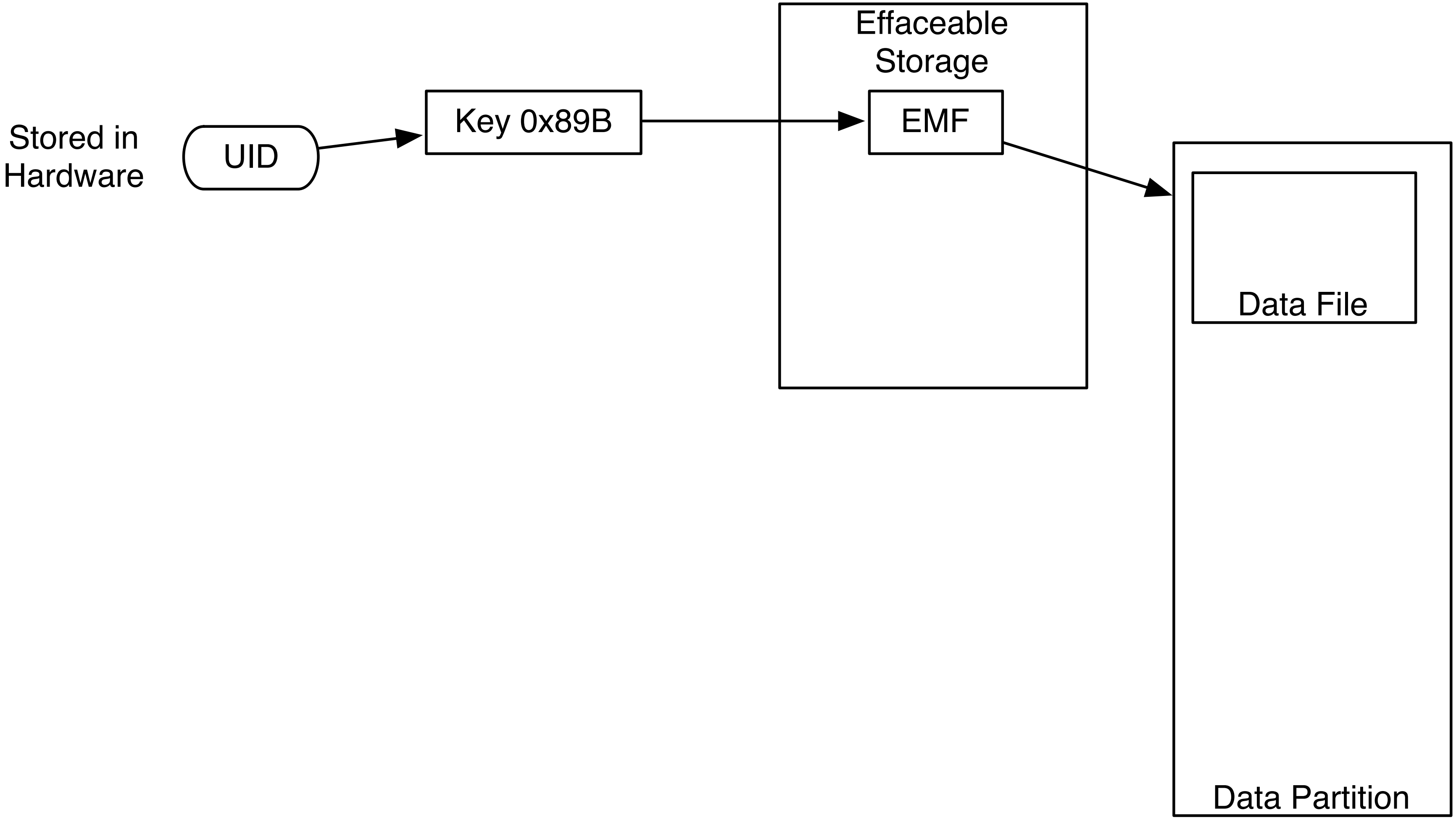
nccgroup



Full disk encryption

- iPhone 3GS / iOS 3
  - Dedicated AES processor
  - Located in DMA channel between CPU and Disk
- Generate a random key (EMF key)
- Encrypt EMF key using a hardware-derived key (0x89b)
- Store encrypted EMF key in special disk area
- Use this to encrypt filesystem metadata

iOS 3 - FDE



Effaceable Storage
Stored in Hardware
UID
Key 0x89B
EMF
Data File
Data Partition

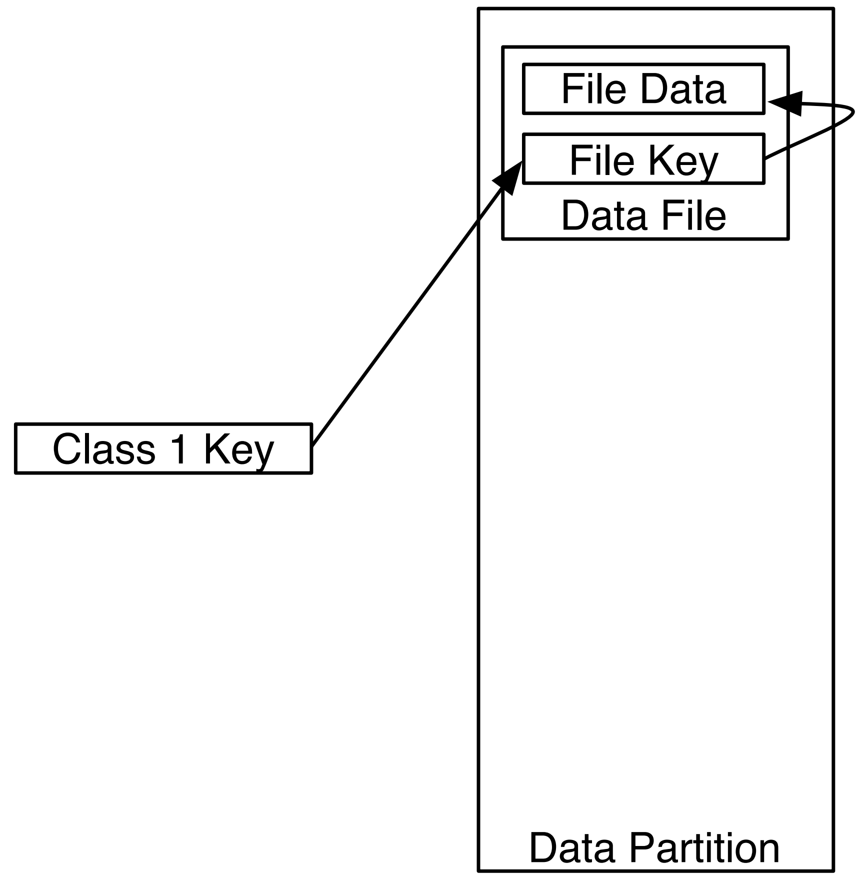
Advantages

- Advantages
  - Fast wipe
  - Can't access / modify data directly (without OS)
  - Can't transfer chips to another device
- Limitations
  - Filesystem access grants access to everything
  - No additional protections when locked

File-level encryption

- Data Protection API introduced in iOS 4
- Random encryption key created for each file
- File key is encrypted using a class key
- Encrypted file key stored with file metadata

iOS 4 - Data Protection API

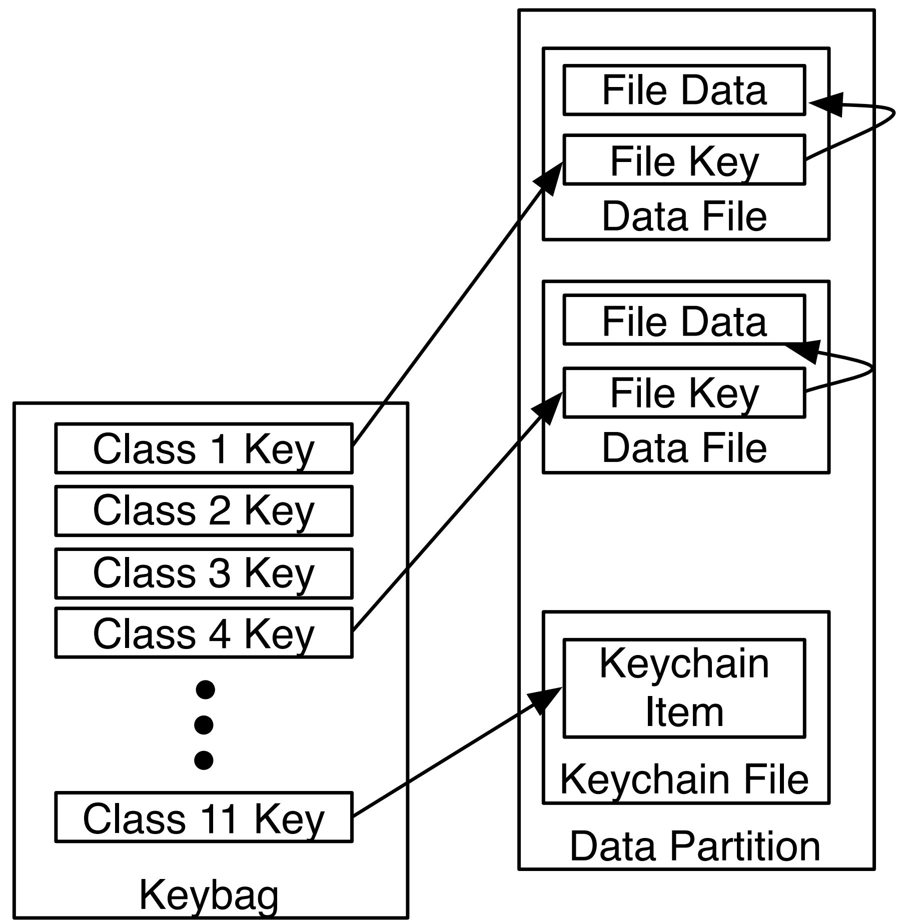


Multiple classes

- Default class:
  - iOS 4 - 6 is "no protection"
  - iOS 7 - 9: Complete until First Authentication
  - Most system apps through iOS 7 still used None

Protection Class	Description
None	No additional encryption
Complete Unless Open	Asymmetric, for locking while writing
Complete Until First User Authentication	Encrypted after reboot, until first time unlocked
Complete	Encrypted whenever device is locked

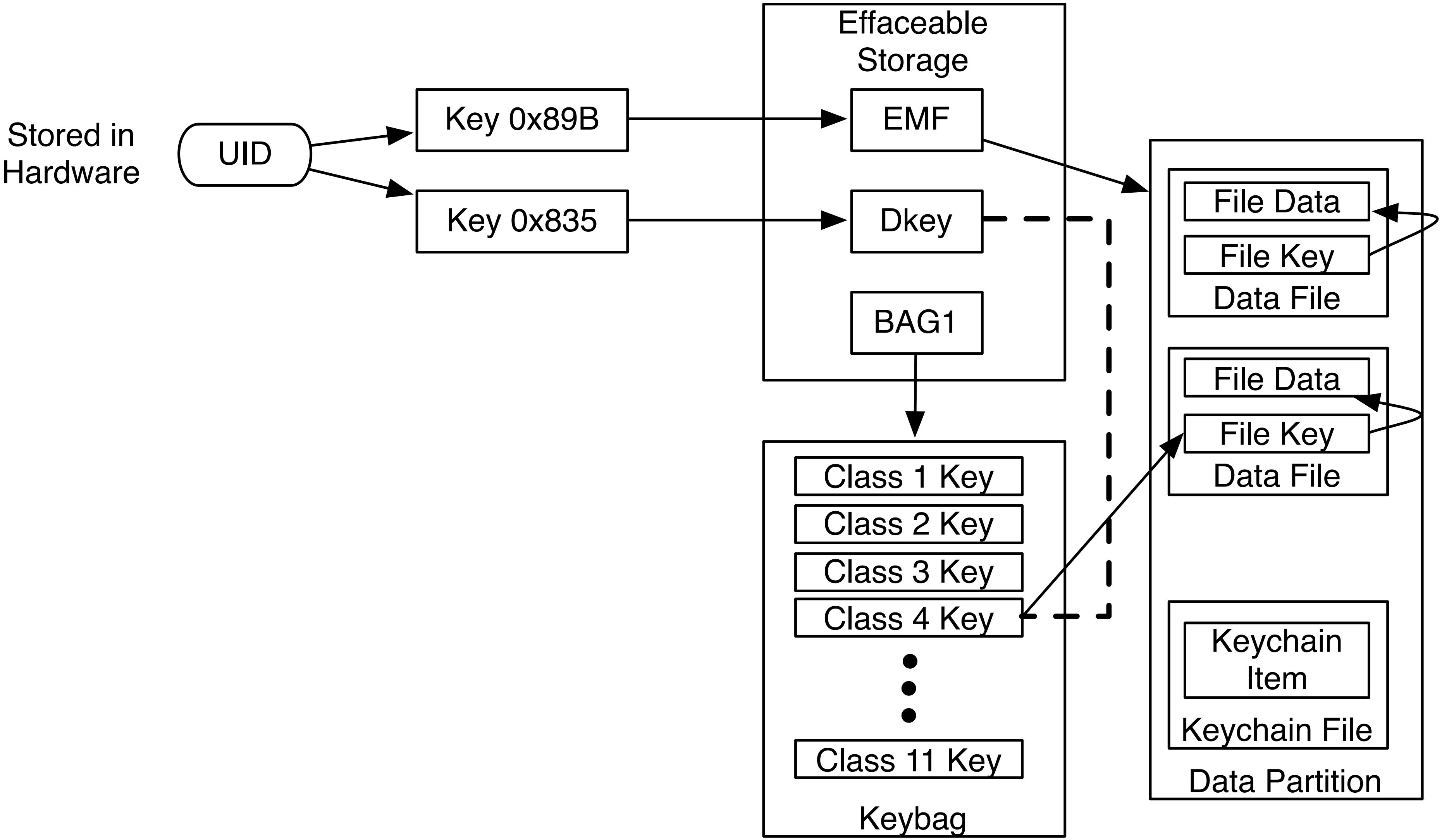
Class keys in the keybag



Data Protection: None

- Class 4 or D is File Protection “None” class
- Random Dkey generated
- Encrypted with key 0x835, derived from UID
- Encrypted key stored in effaceable storage

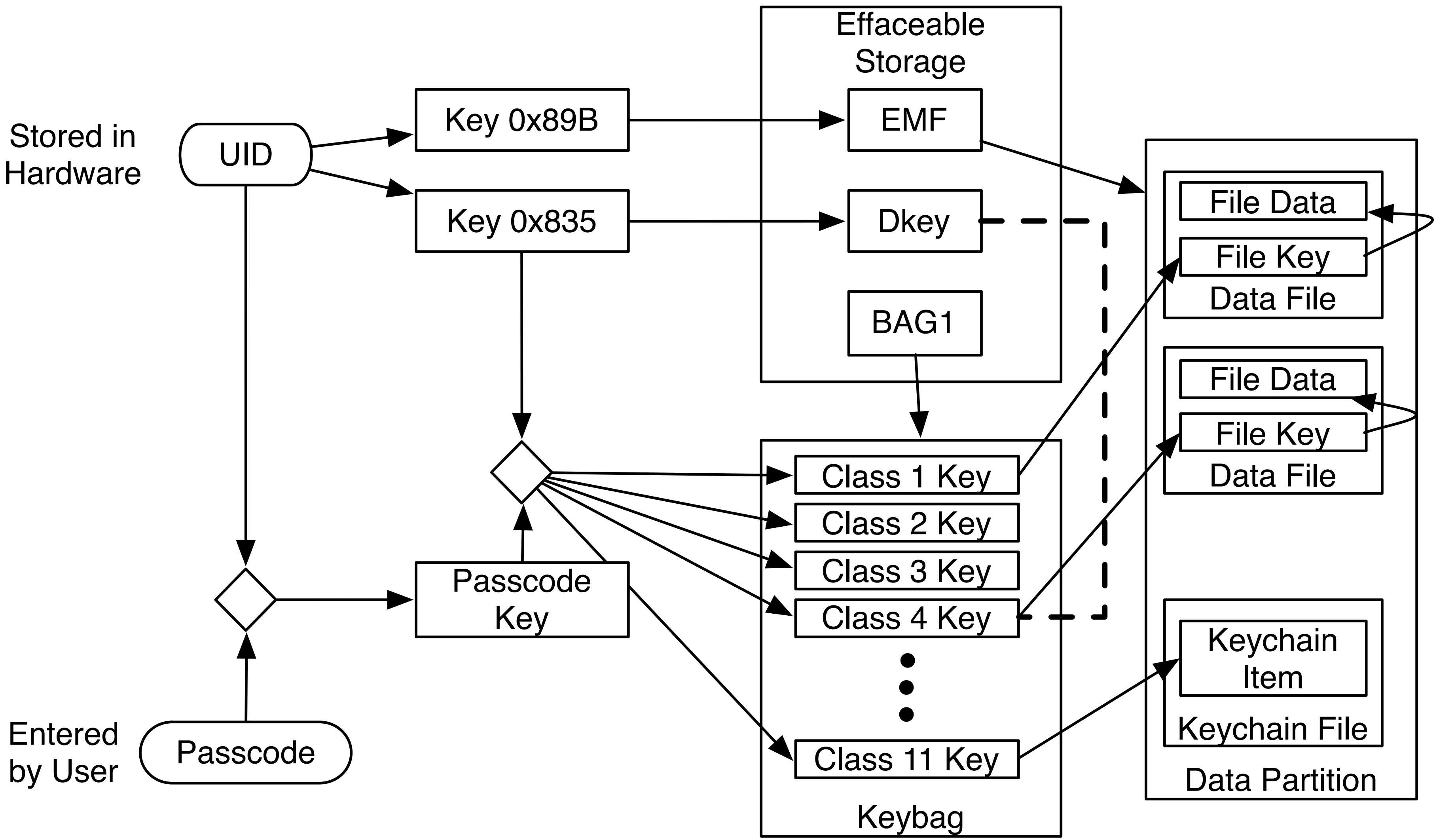
Default protection key



Class key protection

- Each class key is also wrapped or encrypted
  - Using the user's passcode key
- Entire keybag is encrypted
  - Using a bag key (stored in effaceable storage)
- When passcode is changed, old bag keys deleted

Passcode and keybag



Passcode KDF

- PBKDF2, using Passcode, Salt, UID, variable iterations
- Work factor depends on device
  - Constant time — approx. 80 mS / attempt
- A7 onward add a 5 second delay
- Depends on UID, which can't be extracted from phone
  - Not possible to bring to your cracking cluster

Brute forcing passcode

- Must be performed on the device
  - Signed external image
  - Using a bootrom vulnerability
- 80 mS per attempt
  - Now up to 5 sec, so multiply table by ~62
- Attempt escalation, auto-wipe are part of UI
  - When booted from external image, no limits

Complexity	Time
4-digit numeric	15 min
6-digit numeric	22 hours
6-char lowercase	286 days
6-char mixed case	50 years

Locking…

- FileProtectionComplete key removed from RAM
    - All Complete protection files now unreadable
- Other keys remain present
    - Allows connection to Wi-Fi
    - Lets you see contact information when phone rings
- [I once found an edge case where this doesn't happen…]

Changing passcode…

- The system keybag is duplicated
- Class keys wrapped using new passcode key (encrypted with 0x835 key, wrapped with passcode)
- New BAG key created and stored in effaceable storage
  - Old BAG key thrown away
- New keybag encrypted with BAG key

Rebooting…

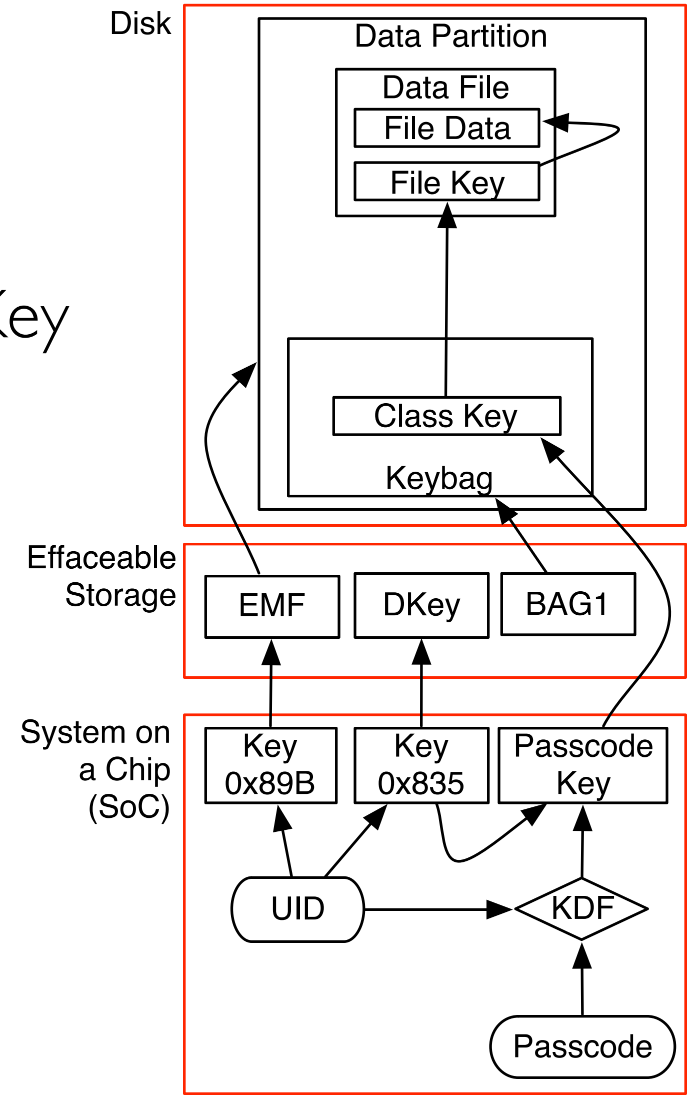
- File Protection Complete key lost from RAM
- Complete until First Authentication key also lost
- Only "File Protection: None" files are readable
  - And then only by the OS on the device
    - Because FDE

Wiping device…

- Effaceable storage is wiped, destroying:
  - DKey: All "File protection: none" files are unreadable
  - Bag key: All other class keys are unreadable
  - EMF key: Can't decrypt the filesystem anyway

Play it again!

- File is encrypted with a File Key
- File Key encrypted with Class Key
- Class Key encrypted with Passcode Key
- Passcode key derived from:
  - UID, 0x835, Passcode
- Keybag encrypted with Bag Key
- Entire disk encrypted with EMF Key
- EMF key encrypted using 0x89b
- 0x89b and 0x835 derived from UID



Weakness and Attacks

Breaking Through the Crypto

- Several ways to get around these protections
  - Jailbreaking devices
  - Simple bugs in the software
  - Forensic tools using obscure or broken features
  - Special boot-level capabilities
  - Collect from other locations ("To the cloud!")

Jailbreaking

- Exploits bugs in the operating system
- Bypasses code signing, sandboxes, etc.
- Needs to modify filesystem to maintain persistence

- Jailbreak process cannot bypass crypto on a locked device
  - But may weaken it
- Generally need to unlock, install, reboot device:

- Jailbreakers have much larger attack surface
  - Any app or system process on unlocked device

Bugs

- Lockscreen bypasses
  - Really just moving from one app to another
  - Crypto protections are still in place
  - Limited data accessibility
  - Usually fixed quickly
- Malicious apps
  - From app store
  - Side-loaded with enterprise certs
- OS-level problems

Forensic Capabilities

- No magic channels just for forensics tools
  - Frequently using same bugs found by community
  - Methods and capabilities often closely held
  - Difficult to fully ascertain
- Locked device
  - Face same obstacles as everyone else
- Unlocked device
  - Hidden or little-understood features
  - Special databases, logs, etc.
  - Treasure trove of info

Boot A New OS

- Multi-step boot process
  - LLB (low-level boot)
  - iBoot
  - OS boot
- Signature checks at each stage
- OS image encrypted for each device class
  - Key derived from "GID" code in SoC
- Bugs on early devices allowed bypassing signature
  - Fixed in iPhone 4S, iPad 2

The Cloud

- Server-side data storage very common
  - Generous "basic" app-data storage for free from Apple
  - User-paid iCloud data
  - Third-party cloud storage
  - App vendor servers
- Can't get data on phone? Go to the net
- Examples of iOS data stored on iCloud:
  - Backups
  - Notes, calendar entries, contacts
  - App-specific data
  - iCloud drive - iWork data, etc.

MDM or Desktop Sync

- Sync to iTunes gets lots of data
  - But no keychain, unless the backup is encrypted
- USB access on trusted desktop
  - Used to allow access to most all data
  - Now only works on beta versions of software
  - Could come back without warning (by design or not)
- Mobile Device Management
  - If enrolled and configured, can remotely unlock
  - Needs Wi-Fi access
  - If rebooted and no cellular data — no MDM.

Privacy Takes Center Stage

New Public Focus

- Encryption features fairly stable since iOS 4
- Why is this a big deal now?

- Software changes
- New hardware features
- Stronger public stance on privacy
  - Somewhat driven by post-Snowden concerns

New Data Protection Defaults

- iOS 7 defaults:
  - 3rd party apps: Complete Until First Unlock
  - System apps: None (except Mail)
- Now System Apps default to Until First Unlock
  - Most data unreadable after a reboot
- Also limited sandbox access over USB
  - Can no longer access all of app's files
  - Even when unlocked
  - Even with trusted computer

See for yourself

- iOS 7 phone:
  - Reboot, Call from landline
  - See full contact information (name, picture, etc.)
- iOS 8 or 9:
  - Reboot, call from landline, just see phone number
  - Unlock, lock again, call again
  - Now you see everything

Secure Enclave

- Introduced with iPhone 5S and iOS 7 in 2013
- Special sub-processor and storage
  - Separate hardened OS
  - Specially encrypted area on disk
- Handles many of the passcode features
  - Not sure whether failure counts stored there
  - Hardcoded 5 second delay
- Additional features added over time
  - Encryption and public keys
- Not very well understood at this point

Public Commitment to Privacy

- Draws a line in the sand
  - "We sell products, not your information"
  - Wants customers to be in control of their data
- Technical advice for strong security choices
- Promise of transparency regarding government access

(Intense) Spotlight on Security

The Road to San Bernardino

- Gradual security improvements over years
- Snowden revelations
- Public commitment to privacy and security
- Beginnings of pushback from law enforcement
- San Bernardino attack
- FBI requests court to order assistance from Apple
- Strangers asking me about the case

What FBI asked for

- A way to bypass passcode guessing limits
- "Custom version of operating system"
- "Tailored to just this phone"

- Possible? Maybe. Probably.
- A good idea?
  - Apple spent nearly 100 pages explaining why not

- FBI eventually …. hired hackers? ….

How'd they finally get in?

- Many possibilities have been suggested
- Mostly just speculation
- Some ideas more likely than others
- Some ideas are… out there.

Probable Attack Surfaces

- Cryptography
  - Extensively used
  - Security highly dependent upon this being "safe"
- Hardware attacks
  - If you can hold it, you can own it
  - How much do you want to spend?
- Software bugs
  - They happen to everyone
  - A lot

Cryptographic Attacks

- To boot a hacked image:
  - Break into Apple and steal their secret keys
    - Other Apple services use tamper-resistant HSM
  - Break signature process
    - RSA signatures
    - SHA1 hashes
    - BootROM bug
- Major cryptographic break in AES
  - Allow derivation of UID and offline cracking
  - Allow direct decryption of data files

Hardware Attacks

- De-cap the SoC
  - Find the UID and extract it
  - Copy encrypted data from NAND
  - Brute-force passcode on a GPU cluster
  - Risky and expensive. No recovery path.
- Memory chip attacks
  - Prevent updating passcode failure count
  - Roll flash back to previous copy where count = 0
- Race condition
  - Detect failure before OS can update count

Software Attacks

- Race condition
  - Enter passcode, do something else REALLY FAST
- Lockscreen bypass
  - Wouldn't get much data
  - Could show springboard
  - Might show that phone had very little data anyway
- Other attacks
  - Code injection
  - DFU or iTunes Restore attacks
  - Wired or wireless attack surfaces

Likely Suspects?

- New Boot ROM bug
  - Boot hacked image containing passcode cracker
- Lockscreen bypass
  - Limited data extraction, but provides window
- Other bugs in lock screen
  - Allowing for interruption of timeout or failure counting
  - Attach a robot
- Hardware-level attacks on memory
  - Interrupting data writes or restoring earlier copy

How much could they get?

- Everything, right away?
  - Needs a major crypto bug
- Everything, eventually?
  - Passcode failure count bypasses
  - Hardware or software attacks
- Simple intel and general phone usage?
  - Lockscreen bypass

Recommendations

General Best Practices

- Good advice on Apple's Privacy and Security pages
- Select newer devices with Secure Enclave
- Select a long passcode
  - Alphanumeric is best
  - Even with 5-second delay in Secure Enclave
- Use TouchID for "typical" daily use
  - But don't forget the passcode!
- If you're arrested, turn off phone
  - Or quickly try to unlock with wrong finger
  - After a few tries, fingerprints disabled

Remaining Questions

From 2014 version of this talk

- Can Apple brute force passcodes?
  - Would they?
  - Could they be ordered to?
  - Has this happened already?

More Hardware Questions

- Is the crypto processor located within the SoC?
  - Pretty sure it is
- Can the Secure Enclave software be updated?
  - To alter the passcode failure protections?
  - Does it require device be unlocked?
- Are any of the SE functions in ROM?
- Where is the failure count located?
  - On SoC or flash?
- Will SE code enforce 10-try limit?

Conclusion

Conclusion

- iOS security highly dependent upon encryption
  - Complex and comprehensive
  - No publicly-known major design flaws

- Bypassing encryption depends on breaking passcode
  - Hardware attacks (potentially expensive)
  - Software bugs (usually fixed quickly)
  - Still a slow process
- Or breaking crypto in general
  - Which breaks EVERYTHING

- Users can fight back with strong passcode

References

- Apple "iOS Security" paper
- "iPhone data protection in depth" (Sogeti, HITB Amsterdam 2011)
- "Evolution of iOS Data Protection and iPhone Forensics: from iPhone OS to iOS 5", (Elcomsoft, Black Hat Abu Dhabi 2011)

Thank You

- David Schuetz
  - Senior Security Consultant at NCC Group
  - david.schuetz@nccgroup.trust
  - @DarthNull

nccgroup

UK Offices

Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Thame

European Offices

Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland

North American Offices

San Francisco
Atlanta
New York
Seattle

Australian Offices

Sydney