HackInBo

Data and Network Security in iOS

Andrey Belenko

Agenda

- Data at rest
  - iOS Data Protection
  - Keychain
  - Application-level encryption
- Data in transit
  - TLS
  - Application Transport Security

Applications need to store data

iOS Data Protection (1)

- Embedded AES co-processor
- File encryption
- Keychain for storing secrets
- Protection classes
- Encryption can be tied to passcode
- Local backup can be encrypted
- iPhone 3Gs and iOS 4+

Protection Classes: Files

NSFileProtectionNone

Accessible at any time
No (special) protection
Encrypted with filesystem key

NSFileProtectionComplete

Accessible only when unlocked
Key depends on passcode
Key purged from memory on lock

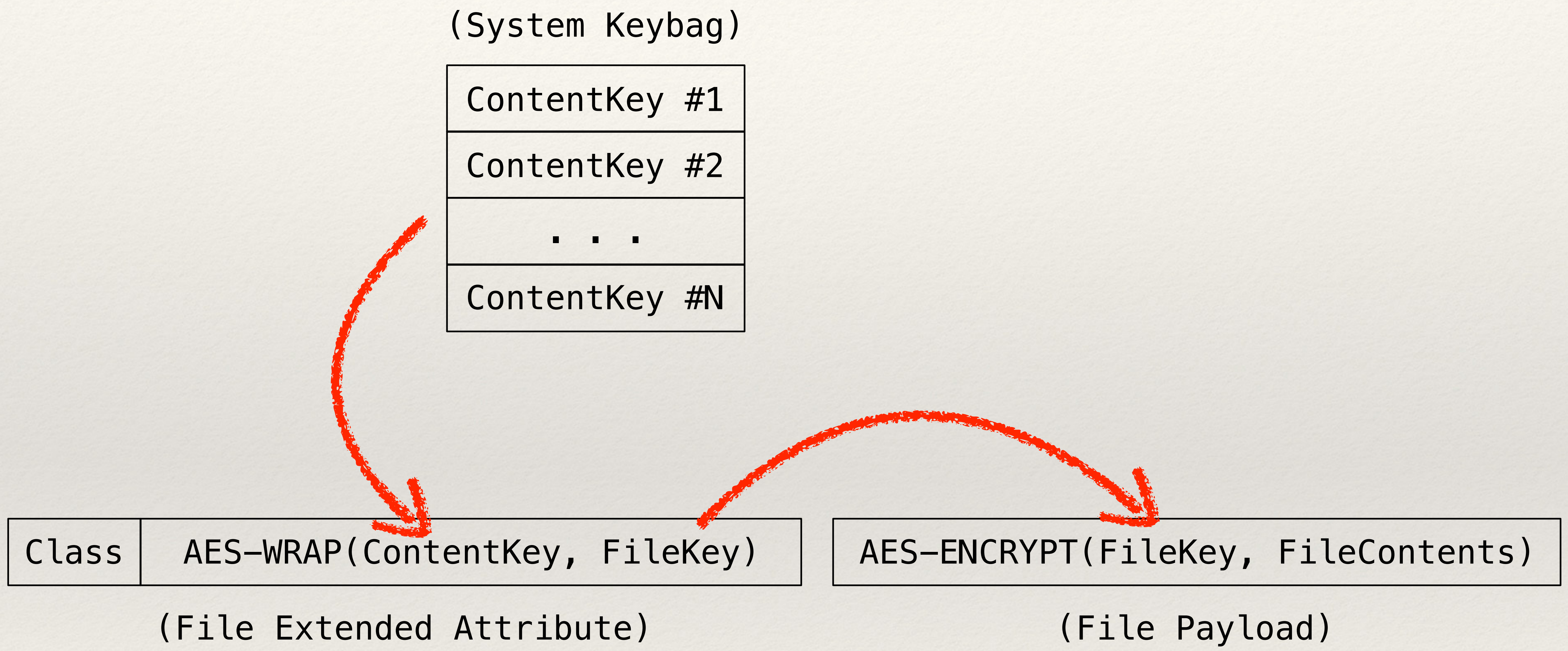
NSFileProtectionComplete…
…UntilFirstUserAuthentication

Accessible only after first unlock
Key depends on passcode
Key stored in memory until shutdown

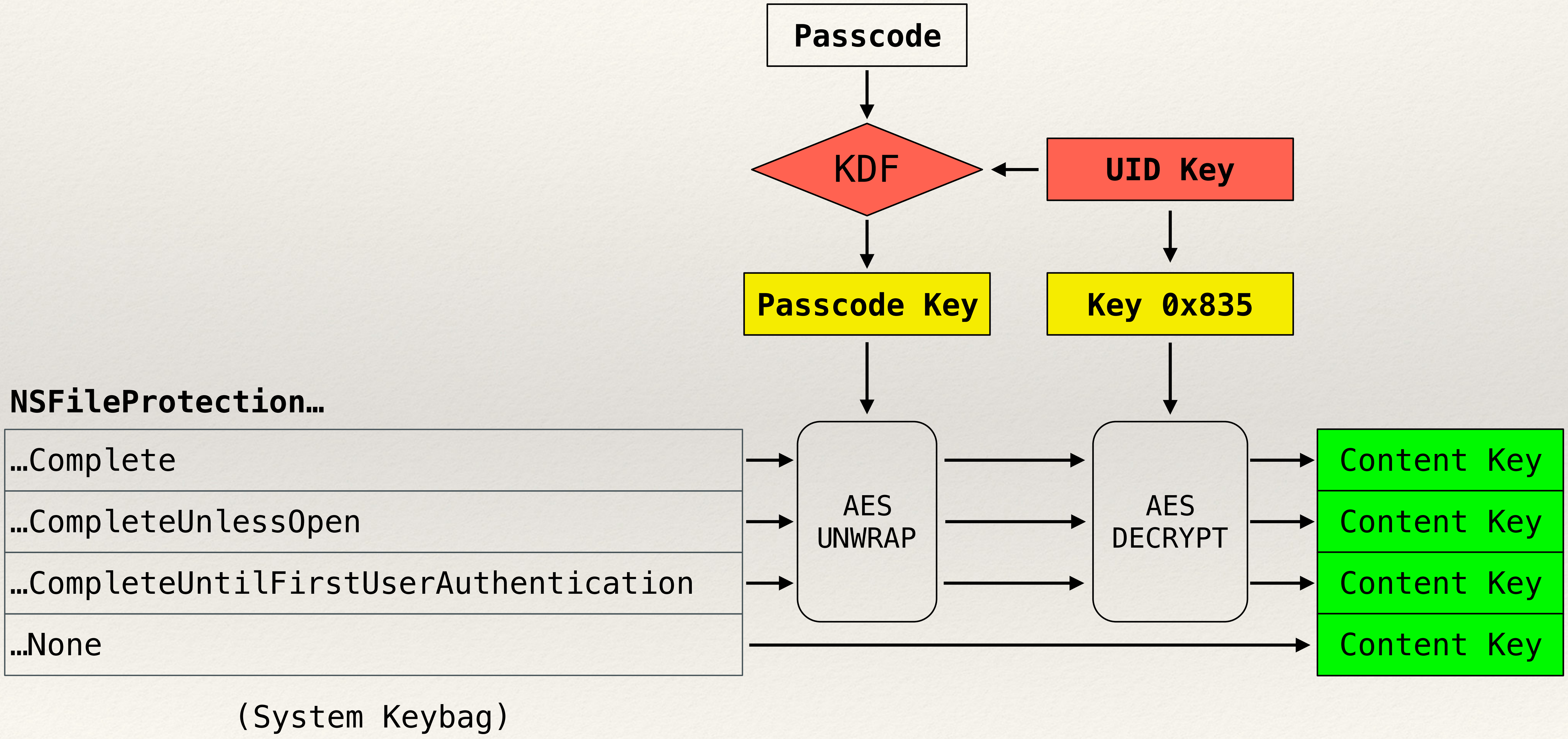
NSFileProtectionComplete…
…UnlessOpen

Key depends on passcode
Accessible only when unlocked, but can create files even while locked

Data Protection: Files


(System Keybag)
ContentKey #1
ContentKey #2
. . .
ContentKey #N
Class
AES-WRAP(ContentKey, FileKey)
AES-ENCRYPT(FileKey, FileContents)
(File Extended Attribute)
(File Payload)

Protection Classes: Files



Protection Classes: Keychain

kSecAttrAccessibleAlways

Accessible at any time
Key does not depend on passcode

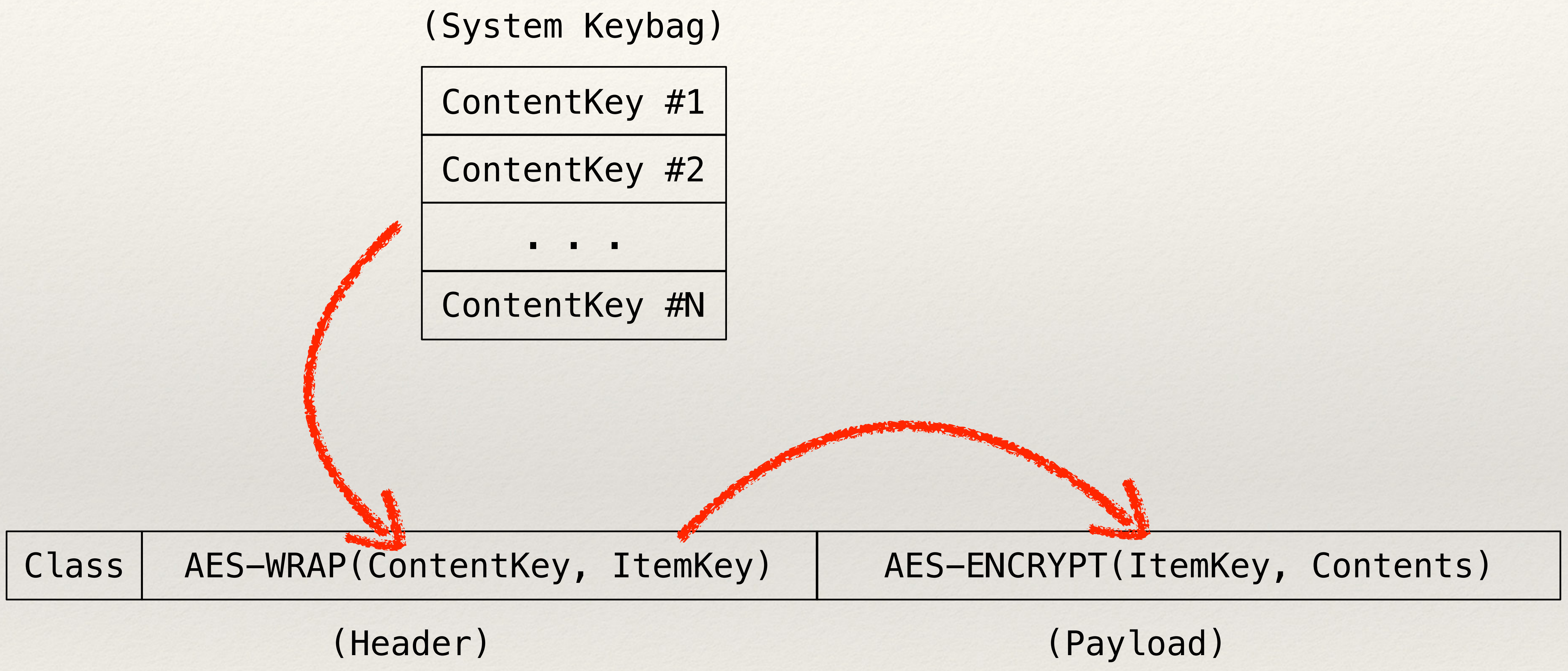
kSecAttrAccessibleWhenUnlocked

Accessible only when unlocked
Key depends on passcode
Key purged from memory on lock

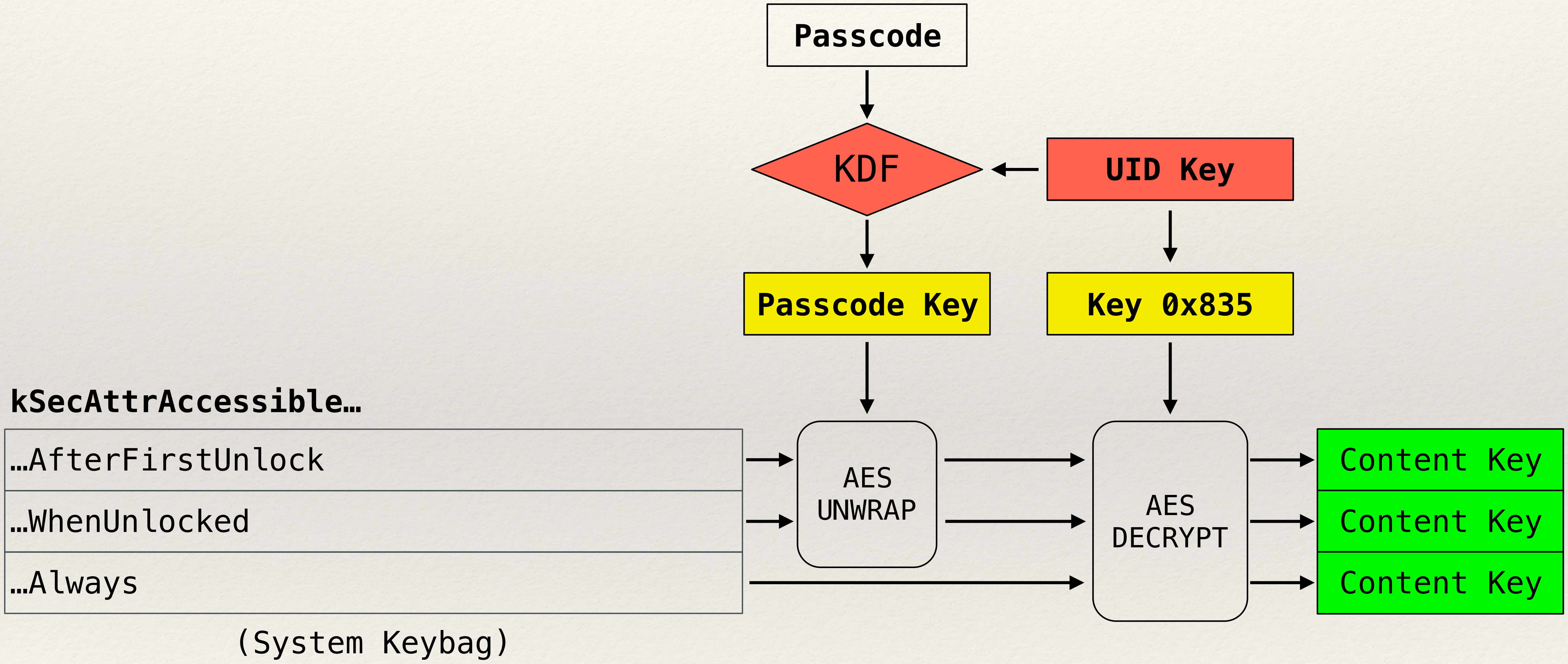
kSecAttrAccessibleAfterFirstUnlock

Accessible only after first unlock
Key depends on passcode
Key stored in memory until shutdown

Data Protection: Keychain



Protection Classes: Keychain



Protection Classes: Keychain

kSecAttrAccessibleAlways…
…ThisDeviceOnly

Accessible at any time
Key does not depend on passcode
Does not migrate to new device

kSecAttrAccessibleWhenUnlocked…
…ThisDeviceOnly

Accessible only when unlocked
Key depends on passcode
Key purged from memory on lock
Does not migrate to new device

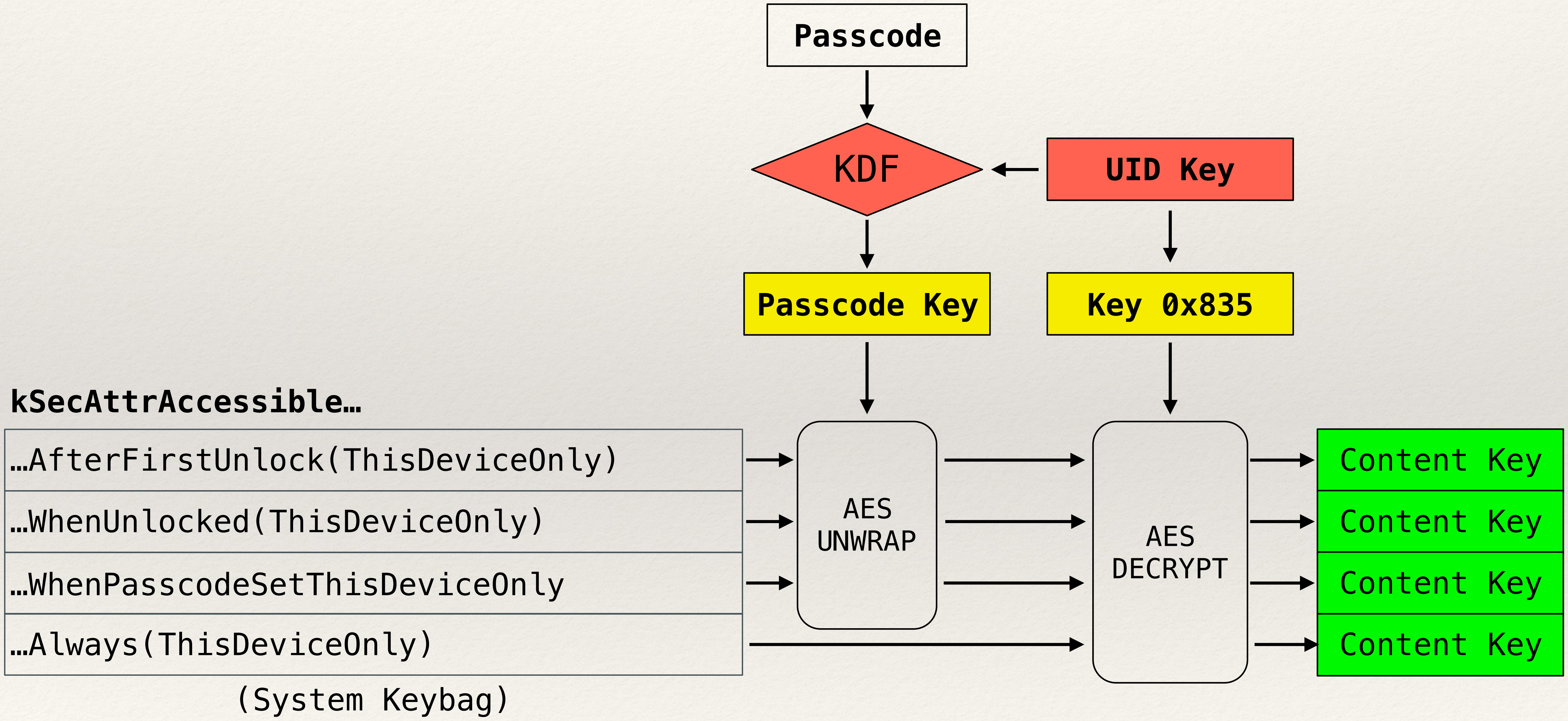
kSecAttrAccessibleAfterFirstUnlock…
…ThisDeviceOnly

Accessible only after first unlock
Key depends on passcode
Key stored in memory until shutdown
Does not migrate to new device

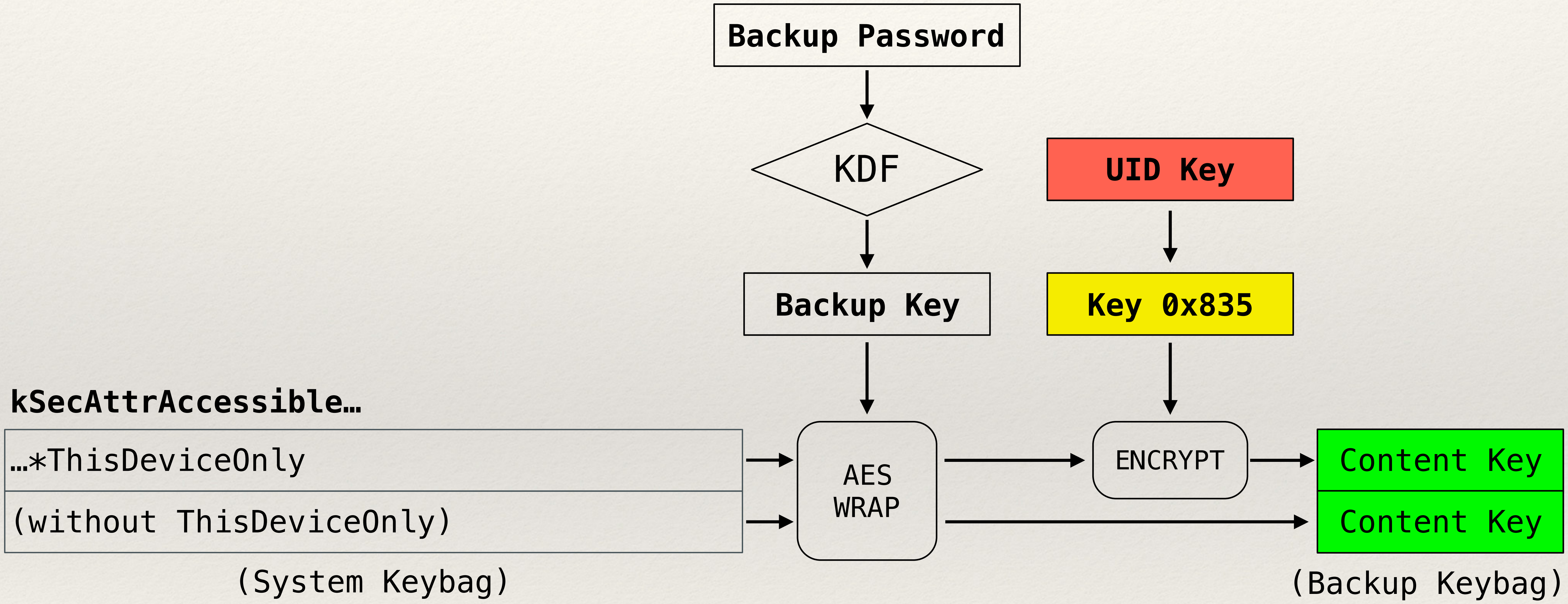
kSecAttrAccessibleWhenPasscodeSet
ThisDeviceOnly

Similar to …WhenUnlocked
Key is destroyed when
Does not migrate to new device

Protection Classes: Keychain



Protection Classes: Keychain Backup



iOS Data Protection (2)

- Secure Enclave
- Touch ID
- Keychain ACLs
- iPhone 5s and iOS 7+

Secure Enclave

- Embedded secure coprocessor
- Own OS, own secure boot
- A7 and newer CPU (iPhone 5s and newer)
- Handles Touch ID, passcode verification, content keys, Keychain ACLs

Keychain ACLs

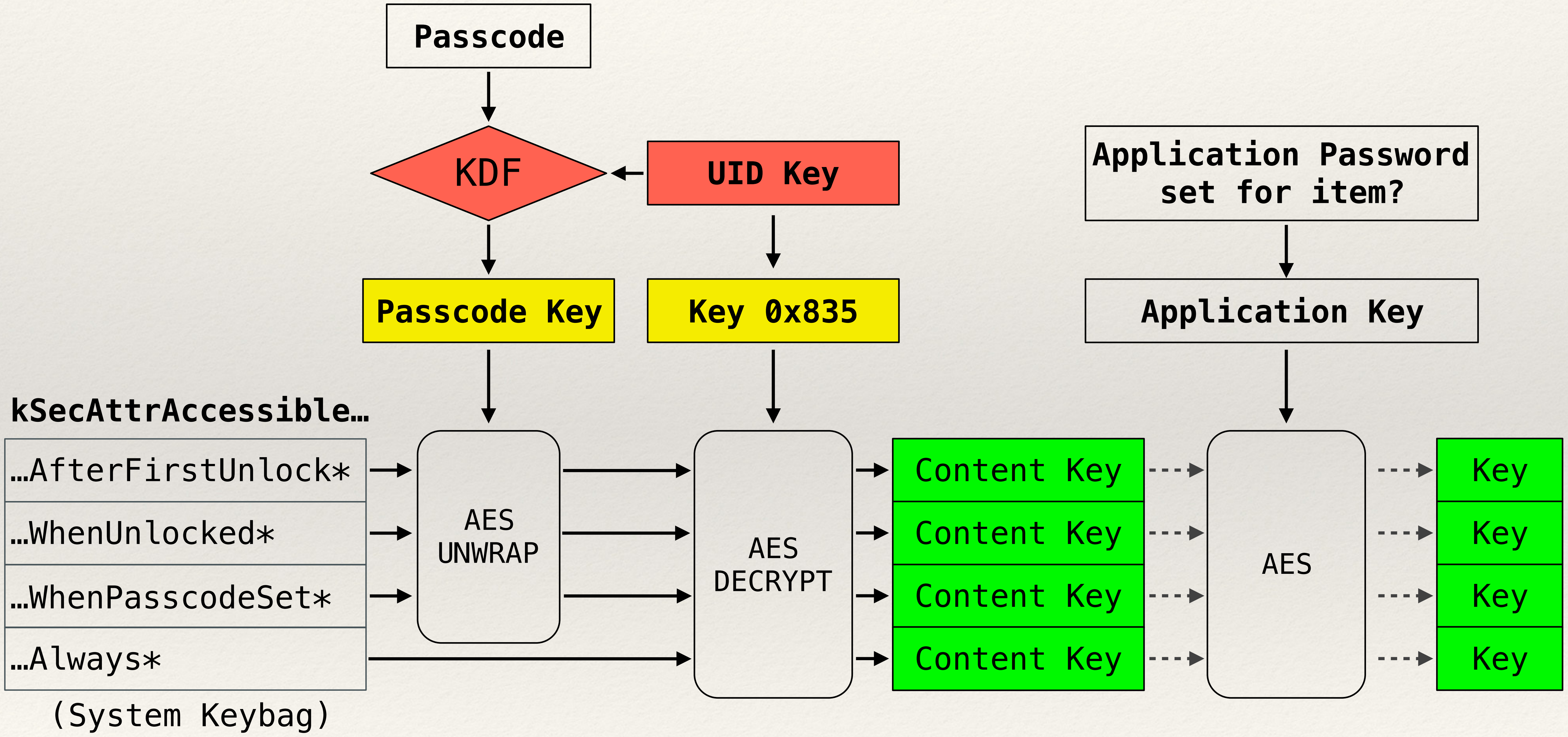
Control when Keychain item is released:

- `kSecAccessControlUserPresence`
- `kSecAccessControlTouchIDAny`
- `kSecAccessControlTouchIDCurrentSet`
- `kSecAccessControlDevicePasscode`

Mix application-managed secret into encryption:

- `kSecAccessControlApplicationPassword`

Application Password

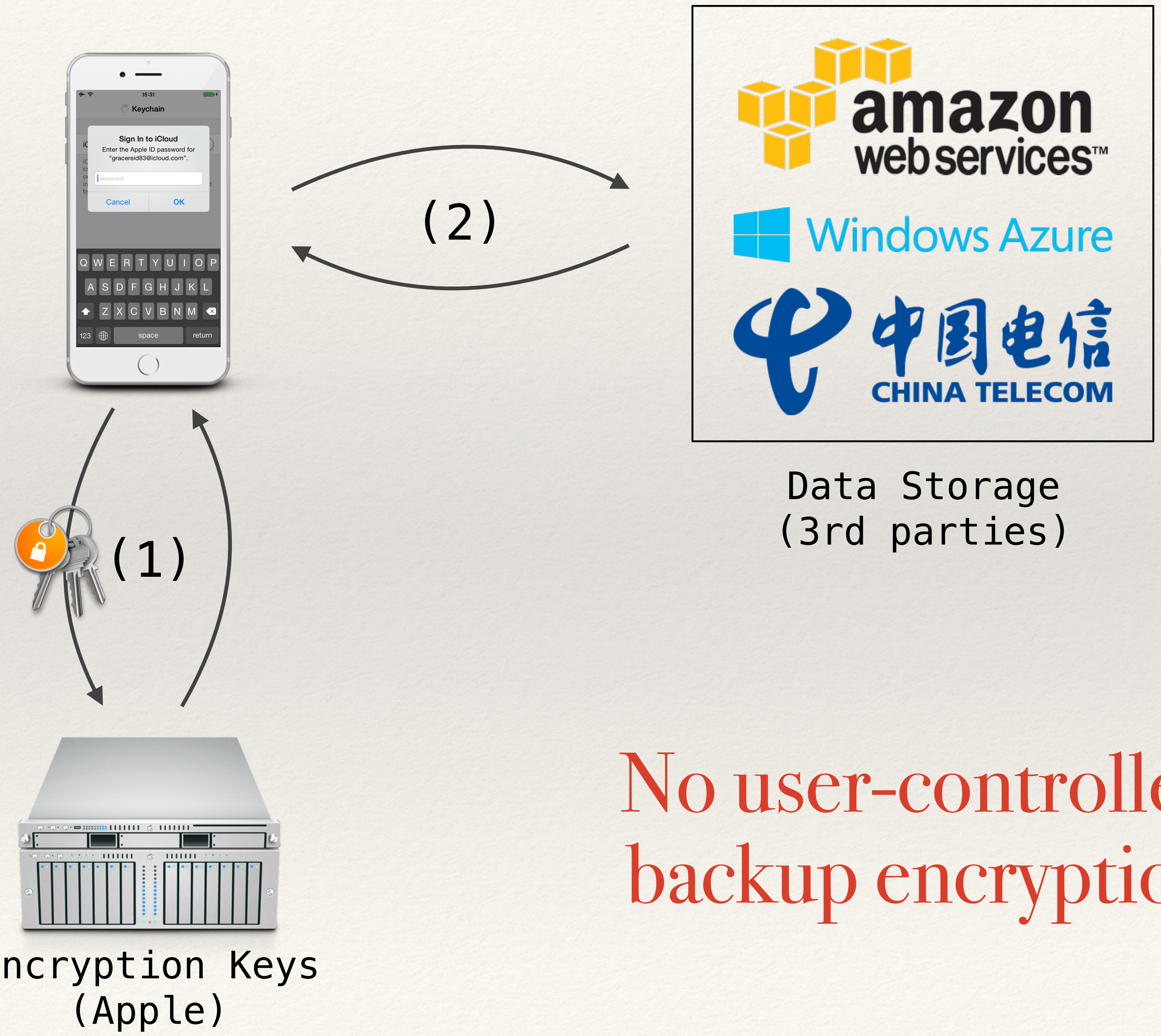


iOS Data Protection reasonably protects data at rest <u>on the device</u>

If application stores data on device, data is likely to be stored off device too

(device backups, settings sync, etc)

iCloud Backup



No user-controlled backup encryption

Once data has left the device, user has no control over it

Limit Data Exposure: Backups

- `<app>/Documents` **is** backed up
- `<app>/Library/Caches` and `<app>/tmp` **are not** backed up
- `NSURLIsExcludedFromBackupKey` excludes file from backup
- Keychain items without …`ThisDeviceOnly` can be recovered from encrypted backup

Limit Data Exposure: File Sharing

- Application sandbox accessible via `house_arrest` service
- Was enabled for all apps before iOS 8.3
- Still enabled on all iOS beta builds
- In iOS 8.4+ enabled for apps with `UIFileSharingEnabled`
- Do not set it unless really needed!

Application-Level Encryption

- Encrypt data *before* writing to file
- Provides defence-in-depth, e.g. if data becomes available off device and outside of Data Protection
- Idea similar to Keychain application passwords
- Application needs to manage keys, encryption, etc…
- Encryption is easy to get wrong!

Application-Level Encryption

- SQLCipher (open source)
- project-imas/encrypted-core-data (open source)
- SQLite Encryption Extensions ($ 2’000)

Application-Level Encryption

- Application still have to manage keys
- Still easy to get wrong :(
- Handle with care!

Applications need to transmit data

Transport Layer Security

- TLS (for TCP) and DTLS (for UDP) are industry standards for securing data in transit
- Problem 1: depends on certificate ecosystem
- Problem 2: fairly difficult to get right

Certificates

- Certificate is deemed trusted if its trust anchor is trusted by OS (i.e. root certificate is in Trusted CA List)
- iOS 9 contains **187** trusted root CA
  - Governments, telecom providers and manufacturers
  - `https://support.apple.com/en-us/HT205205`
- This may or may not be OK for your application

NetLock Kozjegyzoi (Class A) Tanusitvanykiado	NetLock Kozjegyzoi (Class A) Tanusitvanykiado	RSA	2048 bits	MD5	01 03

DigiNotar Services 1024 CA	DigiNotar Root CA	RSA	1024 bits	SHA-1	36 16 71 55 43 42 1B 9D E6 CB A3 64 41 DF 24 38	13:27:58 Mar 29, 2025

Certificate Pinning

- Pinning restricts allowed trust anchors, e.g.:
  - app can accept certificate with a particular public key
  - app can accept certificates issued by a particular CA
- Harder to manage: pins must be kept up to date
- Possibility of self-inflicted denial of service
- AppStore and iTunes Store apps use pinning; iCloud does not

Certificate Pinning

- Error prone: implement with great care
- Can inadvertently disable certificate validation
    - See recent AFNetworking bug
- `https://datatheorem.github.io/TrustKit/`

But Certificate Trust is not the Only Problem…

Weak Cryptography

- Certificate signatures: RSA < 2048 bits, MD5, **SHA-1**
- All versions of SSL and TLS 1.0
- Ciphersuites:
  - Weak key exchange: DH (Logjam)
  - Weak integrity: MD5
  - Weak confidentiality: RC4
  - Export

App Transport Security

- **Enforces** secure communications
- TLS 1.2
- Strong certificates signatures
  - SHA-256 with RSA-2048 or EC-256
- Strong ciphersuites with perfect forward secrecy
  - `ECDHE_{ECDSA|RSA}_WITH_AES_{GCM|CBC}_{SHA2|SHA}`
- Connections with weaker security will fail

App Transport Security

- iOS 9 and OS X 10.11
- Can specify exceptions in `Info.plist`
- `https://developer.apple.com/library/prerelease/ios/technotes/App-Transport-Security-Technote/`

ATS makes TLS misconfigurations easy to notice:

your app just stops working

ATS is great for security

Absolutely use it if you care about your users' data

Data protection doesn’t end here

HackInBo

Thank You!

Andrey Belenko

andrey.belenko@gmail.com
@abelenko