

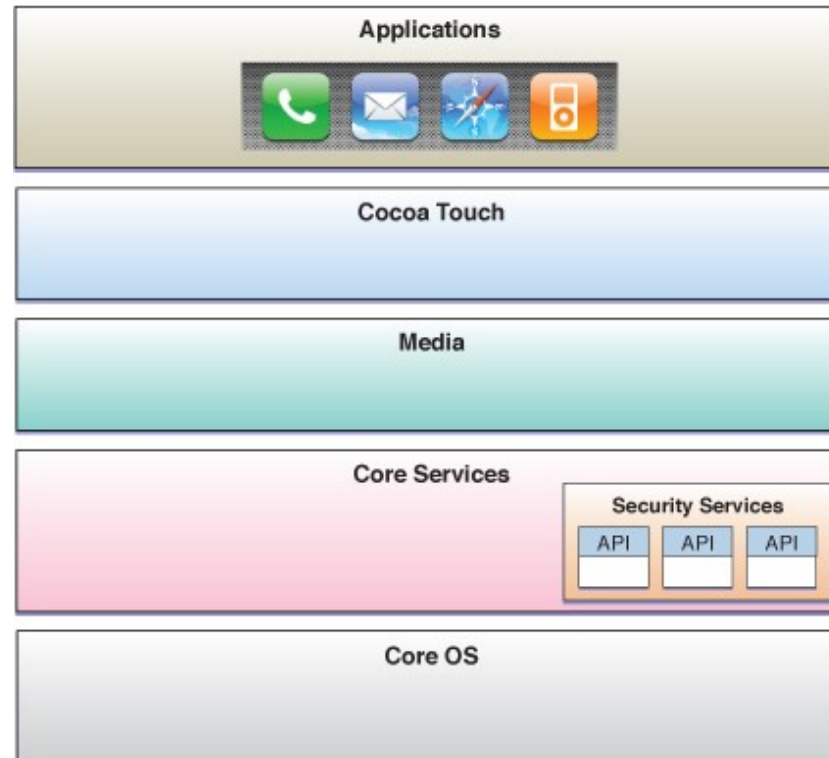
# Sicherheit von Smartphone-Betriebssystemen im Vergleich

Andreas Jansche  
Gerhard Klostermeier

# Inhalt

- iOS
  - Sicherheitsmechanismen allgemein
  - Sicherheits-APIs
  - weitere Features
  - Probleme
- Android
  - Architektur und Sandboxing
  - Dateisystemverschlüsselung
  - Device Administration API
  - weitere Features
- Vergleich

# Position im Schichtenmodell

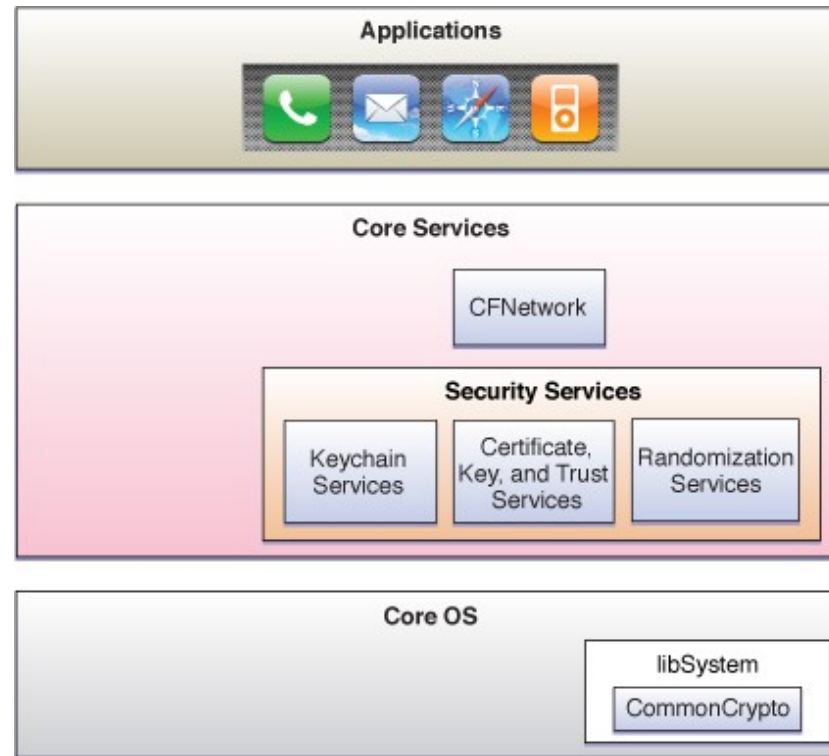


Quelle: <https://developer.apple.com>

## Security Server

- Implementiert Sicherheitsfunktionen
  - *Root certificate trust management*
  - *Keychain*
  - Sicherheitsprotokolle
- Regelt Autorisierung
- Hat keine eigene API → Zugriff über andere APIs

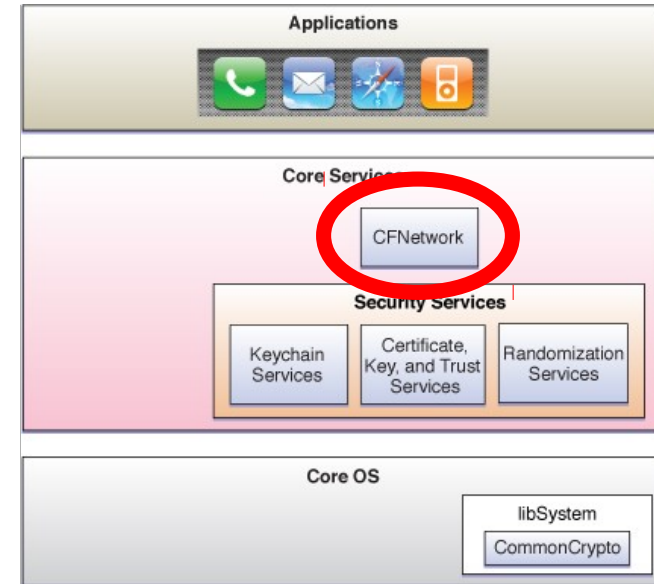
# iOS Security APIs



Quelle: <https://developer.apple.com>

## APIs – CFNetwork

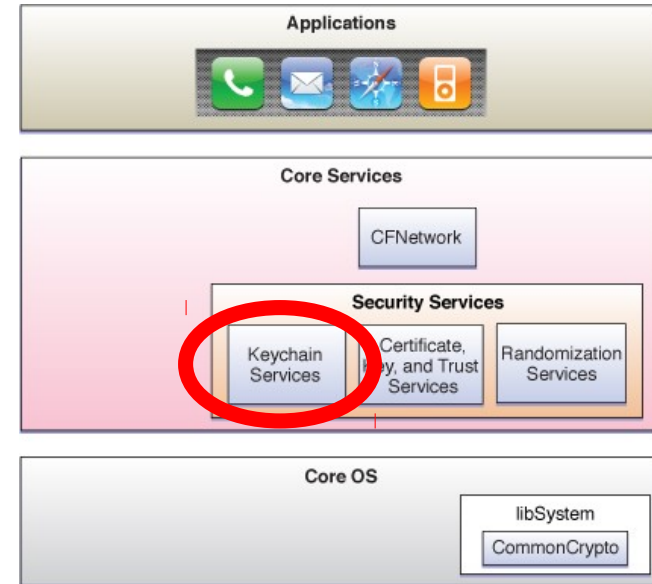
- High-level API
- Sichere Datenströme verwalten
- Authentisierungsinformationen an Nachrichten anhängen



Quelle: <https://developer.apple.com>

# APIs – Keychain Services

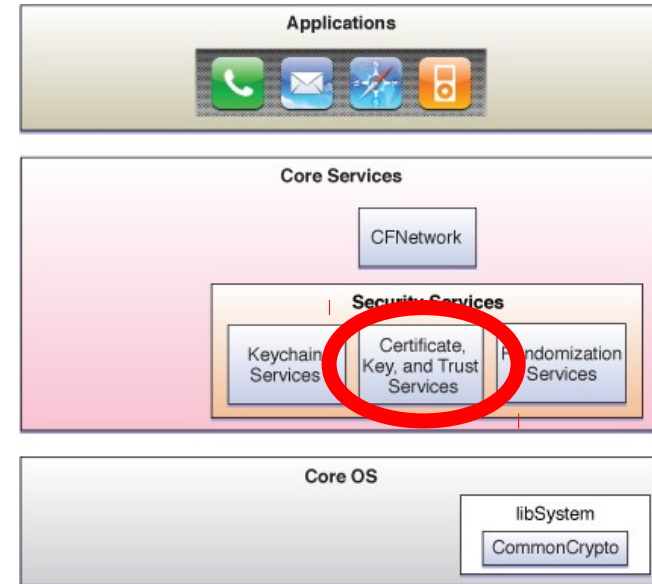
- „Schlüsselbund“
- Geheimnisverwaltung von Apple
- Geheimnisse sind z.B. Passwörter, Schlüssel, Zertifikate, etc.



Quelle: <https://developer.apple.com>

# APIs – Certificate, Key, and Trust Services

- Stellt allgemein Sicherheitsfunktionen bereit:
  - Erstellen, lesen und verwalten von Zertifikaten
  - Zertifikate zu Geheimnisverwaltung (Keychain) hinzufügen
  - Verschlüsselungsschlüssel erstellen
  - Daten verschlüsseln und entschlüsseln
  - Daten signieren und Signaturen verifizieren
  - Vertrauensrichtlinien (trust policies) verwalten

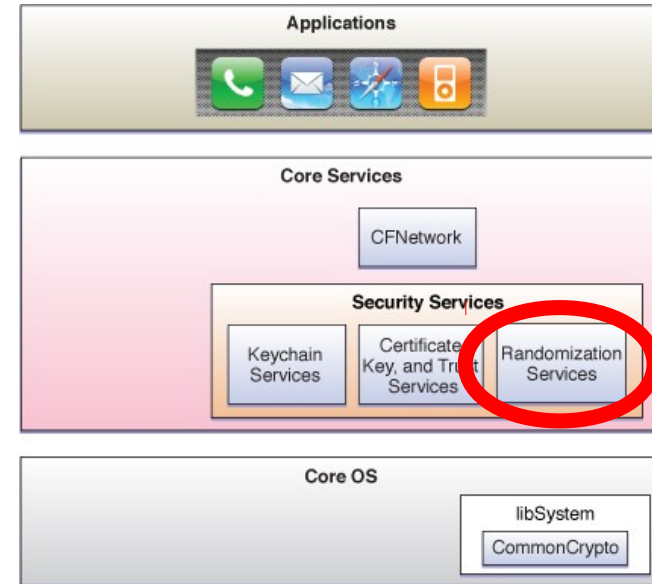


Quelle: <https://developer.apple.com>



# APIs – Randomization Services

- Generiert kryptographisch sichere Zufallszahlen
- Pseudo-Zufallszahlen
- Algorithmus kann nicht aus Zahlenfolge rekonstruiert werden



Quelle: <https://developer.apple.com>

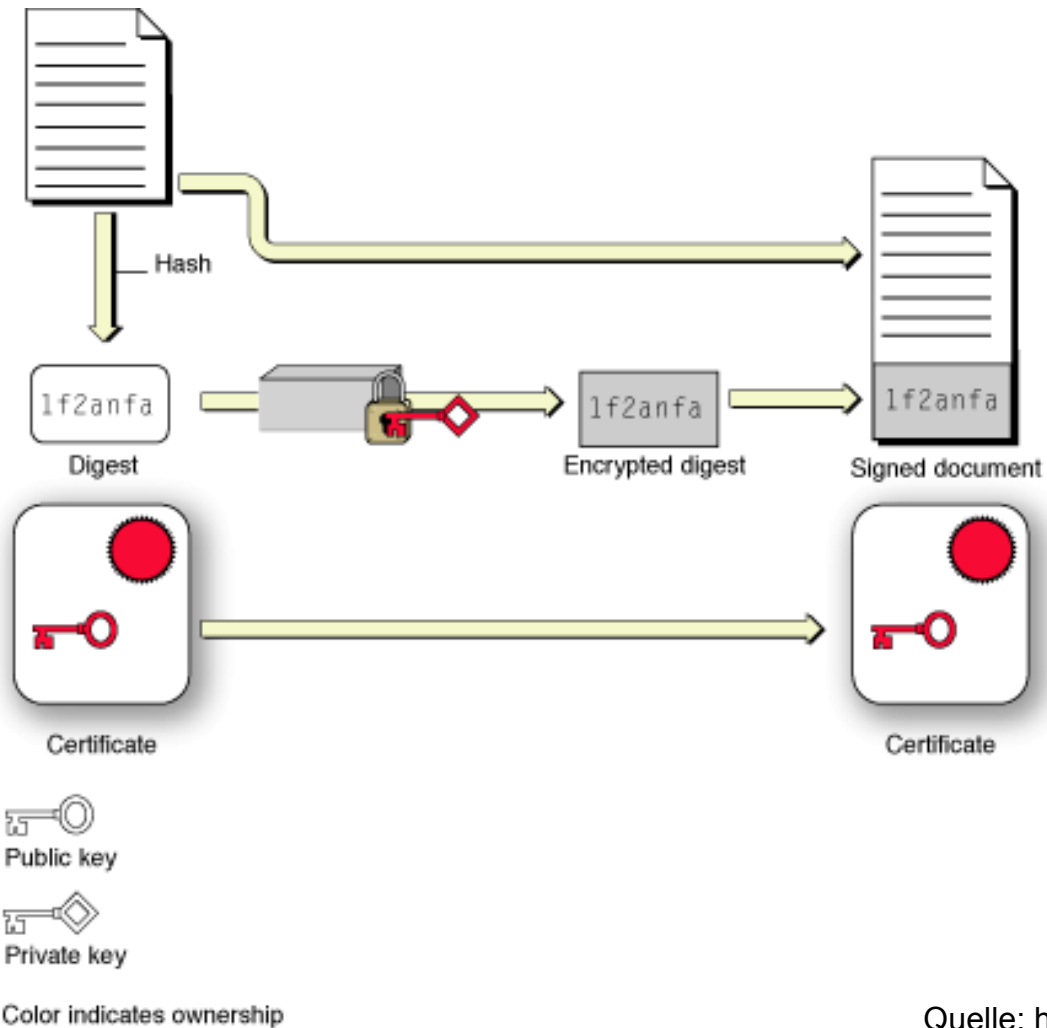
## Sandboxing

- Anwendung wird von anderen isoliert
- Jede App wird in eine Sandbox installiert
- Anwendung kann nur auf eigene Dateien / Einstellungen zugreifen
- Kann nur seine eigenen Keychain-Geheimnisse sehen
- Kann nicht auf Hardware zugreifen
- Kann keine OS Ressourcen abgreifen
- Wenn eine App weitere Rechte haben will (z.B. Kontaktdaten-Zugriff), muss sie den Security Server fragen

## Code Signing

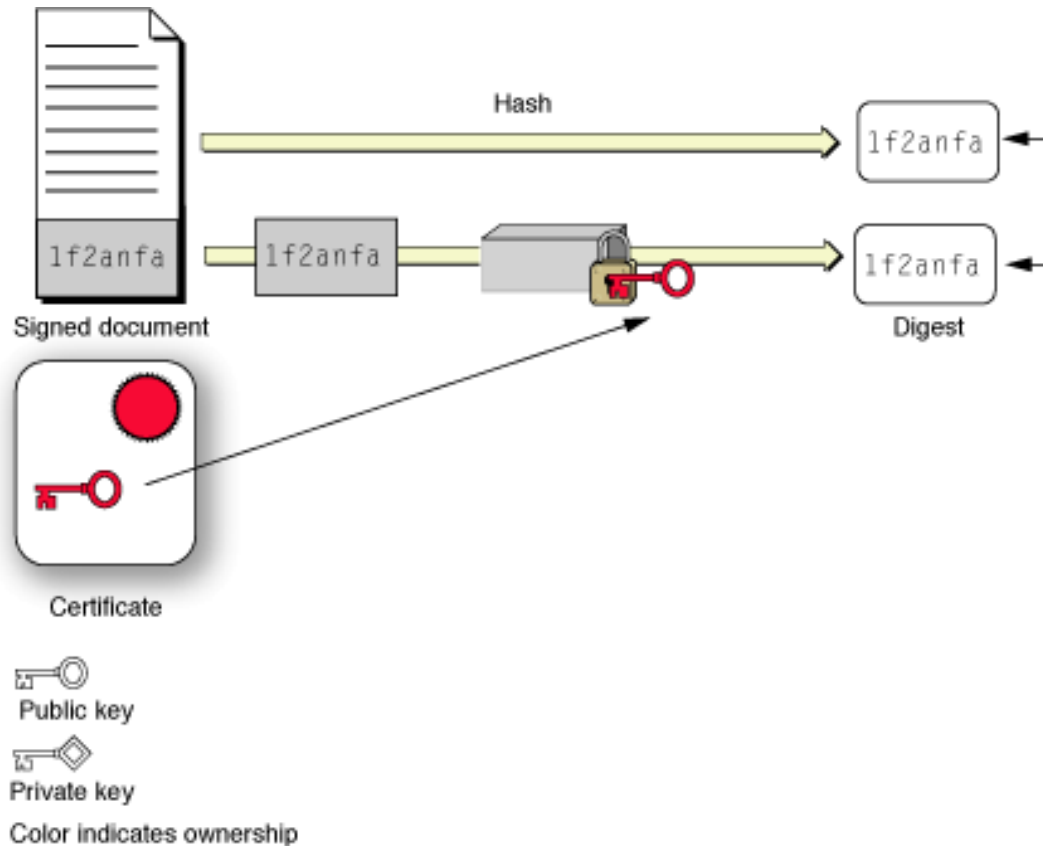
- Anwendungen werden digital signiert
- iPhone führt nur von Apple signierte Apps aus
- Manipulierte Apps werden ebenfalls nicht ausgeführt

# Code Signing



Quelle: <https://developer.apple.com>

# Code Signing



Quelle: <https://developer.apple.com>

## App Store / Code Signing

- Harmlos aussehende App einreichen
- App wird nicht enttarnt und im App Store verteilt
- App kann heruntergeladen und installiert werden
- Eine Sicherheitslücke wird ausgenutzt → Unsignierter Code wird nachgeladen und ausgeführt

## Jailbreak / Sandbox

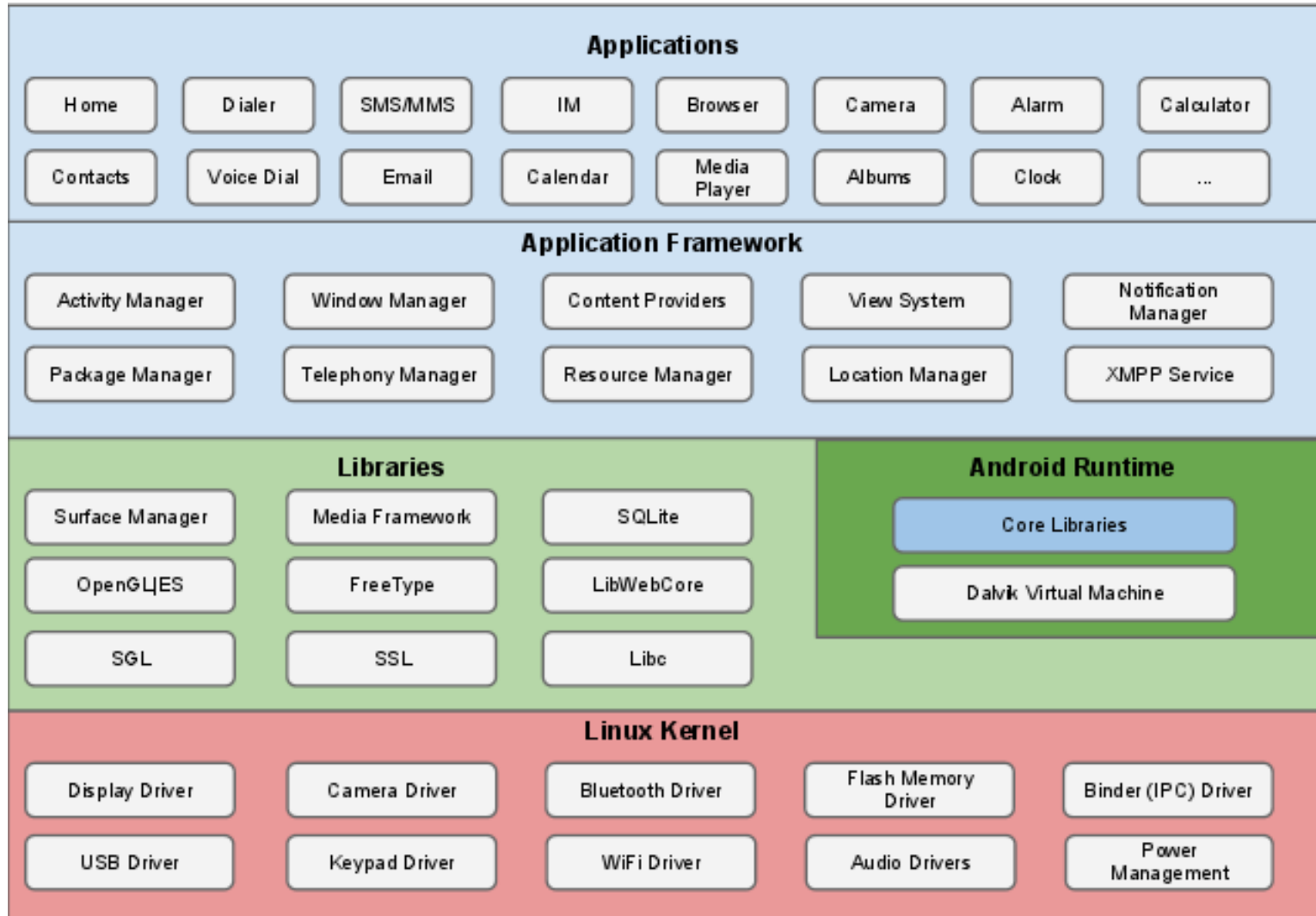
- Überwinden von Nutzungseinschränkungen
- Möglich für fast jedes iOS Gerät  
(außer iPhone 4S, iOS 5.0.1, bis jetzt...)
- Von Apple nicht signierte Apps können ausgeführt werden
- Anwendungen können auch von anderen Quellen installiert werden
- Sicherheitsrisiko liegt beim Benutzer

# Android





# Architektur



Quelle: <http://source.android.com/tech/security/index.html>

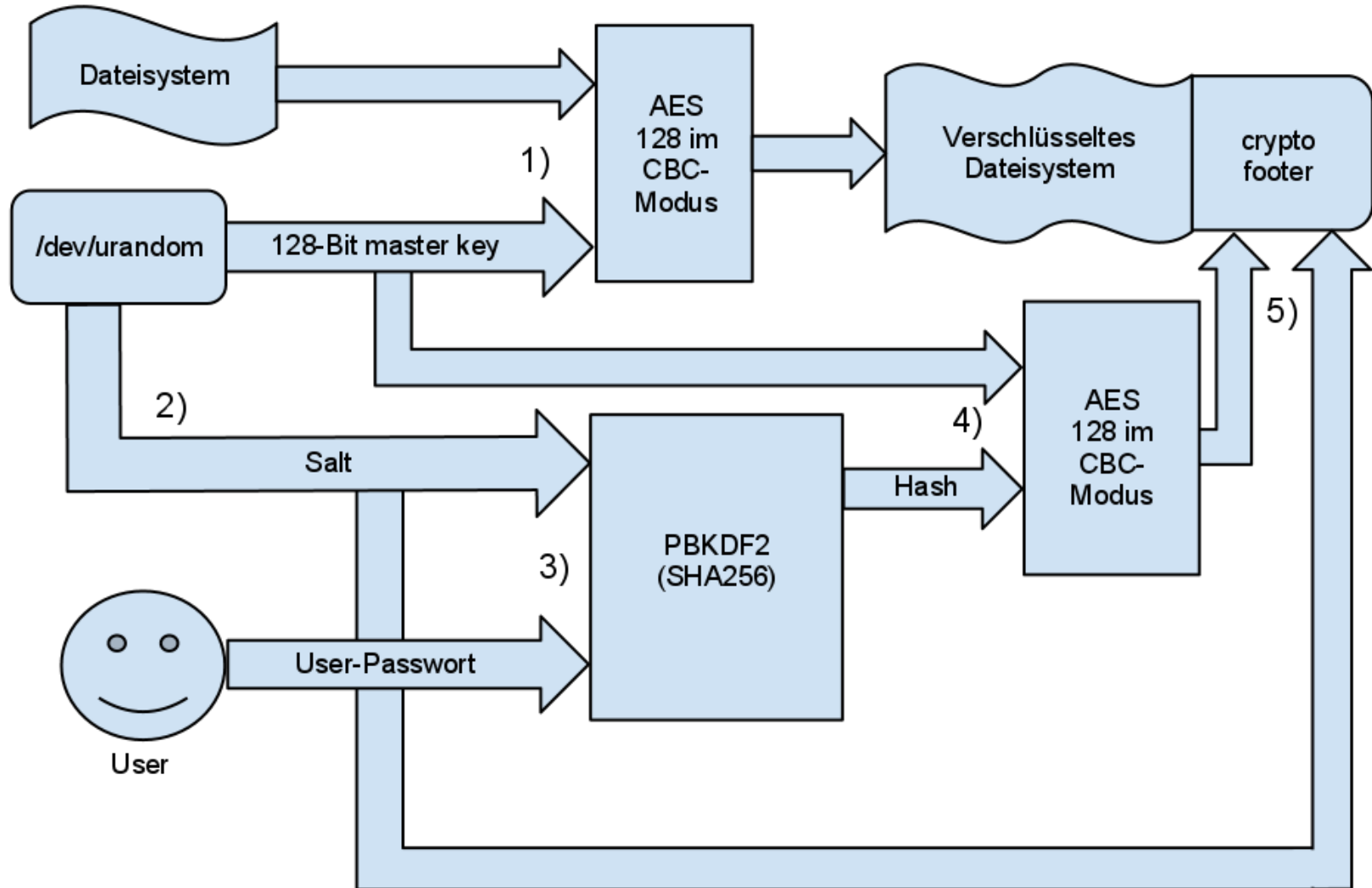
# Sandboxing

- Grundlage: Benutzerkonzept von Linux (nicht VM mit eingeschränkten Rechten)
  - Jeder Benutzer hat UID
  - Prozesse eines Benutzer laufen unter dessen UID
  - Zugriffsrechte über diese UID geregelt
- Unter Android:
  - Jeder Prozess bekommt eine UID
  - Ressourcen des Prozesses vor Zugriff durch andere Prozesse geschützt
- Realisierung auf Kernel-Ebene  
→ Gilt für die oberen 3 Schichten

# Dateisystemverschlüsselung

- Ab Android 3.0
- AES128 und SHA256
- Nutzung eines User-Passwortes nötig

# Dateisystemverschlüsselung



















## Device Administration API

- Konzept um BYOD sicherer zu gestalten
- Ab Android 2.2, danach erweitert
  
- Admin entwickelt App, welche Security Policies erzwingt
- App muss auf Gerät des Mitarbeiters installiert und aktiv sein
- Sind die Policies eingehalten: Zugang zu Systemen / Daten des Unternehmens
  
- Mögliche Policies:
  - Erzwingen eines User-Passwortes samt Passwortrichtlinien
  - Erzwingen einer Dateisystemverschlüsselung (ab 3.0)
  - Deaktivierte Kamera (ab 4.0)

## weitere Features

- Android Security Program
- Maßnahmen gegen memory corruption
  - im System: ASLR, NX-Bit, ...
  - im SDK/Compiler: ProPolice, ...
- Rooten durch Installation eines modifizierten OS  
→ Löschen der Nutzerdaten
- read-only Systempartition
  - Kernel, Libraries, Laufzeitumgebung, ...
- Safe Mode
  - Booten ohne Drittanbietersoftware

## Vergleich

	Android	iOS
Sandboxing	 Alles im Userspace, also auch für nativen Code	
Verschlüsselung des Dateisystems	 ab v3.0 AES128,SHA256	
read-only Systempartition		 (siehe: [Mor10])
Sicheres Booten (ohne Drittanbieter- Software)		 Vermutlich nicht, da andere Konzepte dies überflüssig machen
Sicherheitskonzept für BYOD	 Dev. Admin. API ab v2.2	 Kaum von Apple, hauptsächlich über Drittanbieter möglich [Heil1b]
Nur signierter Code ausführbar	 Code Signing (nur durch Entwickler)	 Code Signing (durch Entwickler und Apple)
Kontrollierte App-Distribution	 Keine Kontrolle durch Google	 App Store
Maßnahmen gegen <i>memory corruption</i>		 (siehe: [Con11])

# Fragen?