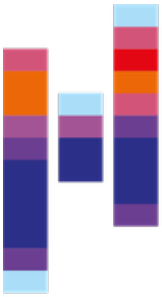

Hochschule
Flensburg
University of
Applied Sciences

Wissenschaftliche Ausarbeitung

Mobile Security: Verschlüsselung und Datensicherheit unter iOS und iPadOS

Florian Hansen, Michael Frank, Martin Hermannsen,

Jan-Erik Wieczorek, Tom Hartelt

8. November 2020

Inhaltsverzeichnis

1 Sicherheit im Alltag

Für die Sicherheit im Alltag hat Apple eine Vielzahl von Systemmechanismen implementiert, die den Schutz der Benutzerdaten sicherstellen. Die Dateiverschlüsselung bei Endgeräten mit iOS- und iPadOS als Betriebssysteme wird von Apple als Datensicherheit bezeichnet und ist das Pendant zum FileVault unter macOS [9].

Der Ausgangspunkt für die Hierarchien der Schlüsselverwaltung befindet sich bei Geräten mit SEP (Secure Enclave Processor, ab A7 und neuer) im dedizierten Silizium der Secure Enclave. Hier werden die Schlüssel und biometrische Daten für die Verwendung von Touch ID und Face ID hinterlegt. Dieser isolierte Bereich ist für den AP (Application Processor) nicht zugänglich, wenngleich das RAM von beiden Prozessoren geteilt wird, denn der Speicherbereich des SEP ist verschlüsselt [9].

Als weiteren Sicherheitsmechanismus werden Zugriffssteuerungen im Kernel erzwungen. Dadurch wird der unbefugte Zugriff auf Daten verhindert. Der Datenzugriff von Apps wird in der Regel durch Sandboxing realisiert und durch die Forcierung von Data Vaults ergänzt. Simplifiziert sind Data Vaults invertierte Sandboxes, die nicht die Aufrufe einer App, sondern den Zugriff auf geschützte Daten einschränken [9].

Die offensichtlichen Sicherheitsmechanismen, die der Benutzer im Alltag erfährt, werden im Folgenden vorgestellt.

1.1 Gerätecodes

Ein Gerätecode wird von dem Benutzer des Geräts eingegeben, um sicherzustellen, dass dieser die erforderlichen Zugriffsrechte besitzt. Ist ein Gerätecode eingerichtet, so wird gleichzeitig automatisch die Datensicherheit aktiviert. Das bedeutet, dass die Dateien des Systems verschlüsselt vorliegen und je nach Datensicherheitsklasse beim Entsperren des Geräts entschlüsselt werden. Ein Gerätecode besteht aus vier oder sechs Ziffern, jedoch können auch alphanumerische Zeichenfolgen (`a-z, A-Z, 0-9`) mit beliebiger Länge als

Gerätecode eingerichtet werden. Natürlich haben diese Codes dann eine höhere Sicherheit als die Varianten, die nur aus Ziffern bestehen. Wichtig zu erwähnen ist außerdem, dass die Gerätecodes mit der UID des Geräts verbunden sind. Eine Authentifizierung ist deshalb nur auf dem Zielgerät möglich und stellt eine Sicherheitsvorkehrung dar, da das Gerät im physischen Besitz des Angreifer sein muss. Selbst wenn das der Fall ist, existiert ein Mechanismus, der den Authentifizierungsprozess auf 80 Millisekunden festsetzt. Damit sind Brute-Force-Angriffe schon bei vierstelligen Gerätecodes, bestehend aus Ziffern, sehr unattraktiv für Angreifer. Zusätzlich können Touch ID oder Face ID als Gerätecodes verwendet werden. Dies bietet dem Benutzer eine bessere User-Experience und eine stärkere Sicherheit beim Verschlüsseln von Daten [9].

1.2 Sicherheit vor unautorisierten Datenverbindungen

Physisch angeschlossene Geräte, z.B. Mac, PC oder Zubehör, werden über die Lightning-, USB- oder Smart Connector-Schnittstelle mit dem Apple-Gerät verbunden. Aufgrund der Tatsache, dass die Ökosysteme der angeschlossenen Geräte vielfältig sein können und es keine kryptographisch zuverlässige Möglichkeit gibt, die Geräte vor einer Datenverbindung zu identifizieren, verlangen iPhones bzw. iPads die Eingabe des Gerätecodes, bevor eine Datenverbindung aufgebaut werden kann [9, S. 65]. Dies verhindert den unautorisierten Zugriff auf die Daten durch ein Fremdsystem. Nachdem die Datenverbindung zu einem Fremdsystem autorisiert wurde, muss der Gerätecode, für eine bestimmte Zeitspanne, nicht mehr für den Verbindungsaufbau zu dem System eingegeben werden. Dadurch können Nutzer, die regelmäßige Verbindungen zu bestimmten Geräten herstellen möchten, für eine Zeitspanne auf eine erneute Authentifizierung verzichten [9].

1.3 Schutz persönlicher Daten

Mobile Anwendungen benötigen oftmals den Zugriff auf bestimmte Funktionaliten oder Daten des Smartphones. Diese stehen den Anwendungen aber

nicht einfach zur Verfügung, Nutzer können für jede App festlegen, auf welche personenbezogene Informationen zugegriffen werden darf. Realisiert wird dies mit verschiedenen Technologien wie bspw. dem Data Vault. In den Einstellungen des Betriebssystems kann der Nutzer einer App die Rechte für den Zugriff auf individuelle Informationen geben. In einigen systemrelevanten Apps von iOS und iPadOS wird der Zugriff jedoch erzwungen, dazu zählen bspw. Kalender, Kamera, Bluetooth und Fitness (Health) [9].

Seit der iOS bzw. iPadOS Version 13.4 werden die Daten der Apps von externen Anbieter (nicht Apple) in einem Data Vault geschützt, um den Schutz vor unbefugten Datenzugriffen zu gewährleisten. Die Apps unter iOS und iPadOS erhalten bei der Nutzung von iCloud standardmäßig Zugriff auf die iCloud Drive. Der Nutzer kann den iCloud-Zugriff jedoch in den Einstellungen ändern und einschränken [9].

1.4 Klassifizierung der Sicherheitschipes

Im Rahmen dieser Ausarbeitung werden oftmals die Sicherheitschips (SoCs - System on Chip) der mobilen Endgeräte thematisiert. Die meisten dieser Sicherheitschips basieren auf einer ARM-Architektur und werden in die meisten der Apple-Produkte wie bspw. iPhone, iPad, Apple Watch, etc. verbaut. Die Chips unterscheiden sich dabei u.a. durch unterschiedliche Anwendungszwecke, Fertigungstechnologien, integrierte Co-Prozessoren, AI-Beschleuniger und der Anzahl der integrierten Transistoren. Für eine bessere Übersicht werden folgend die verschiedenen SoC-Serien, und die Produkte in denen sie verbaut werden, aufgelistet [9].

- *A series*: iPhone und iPad
- *S series*: Apple Watch
- *T series* series: macOS-Produkte (MacBook, Mac-Computer)
- *W series*: AirPods und HomePod

2 Verschlüsselung

Kryptographische Schlüssel werden von dem Betriebsystem für die Ver- und Entschlüsselung von Informationen verwendet. Daten werden verschlüsselt, um die darin enthaltenden Informationen vor potentielle Angreifer geheimzuhalten. Die Verschlüsselung von Daten kann vielfältig eingesetzt werden, typische Anwendungszwecke können bspw. die Verschlüsselung einzelner Dateien oder gar des gesamten internen Dateisystems sein [13].

2.1 Verschlüsselungalgorithmus

Für die Verschlüsselung der Datenaustausches zwischen Komponenten, Systemen oder Apps werden unter iOS unterschiedliche Varianten des etablierten Verschlüsselungsverfahrens AES (Advanced Encryption Standard) eingesetzt. AES ist ein symmetrisches Verschlüsselungsverfahren, die miteinander interagierenden Parteien müssen also einen gemeinsamen Schlüssel austauschen, welcher daraufhin für das ver- und entschlüsseln von Informationen verwendet wird [13, 9].

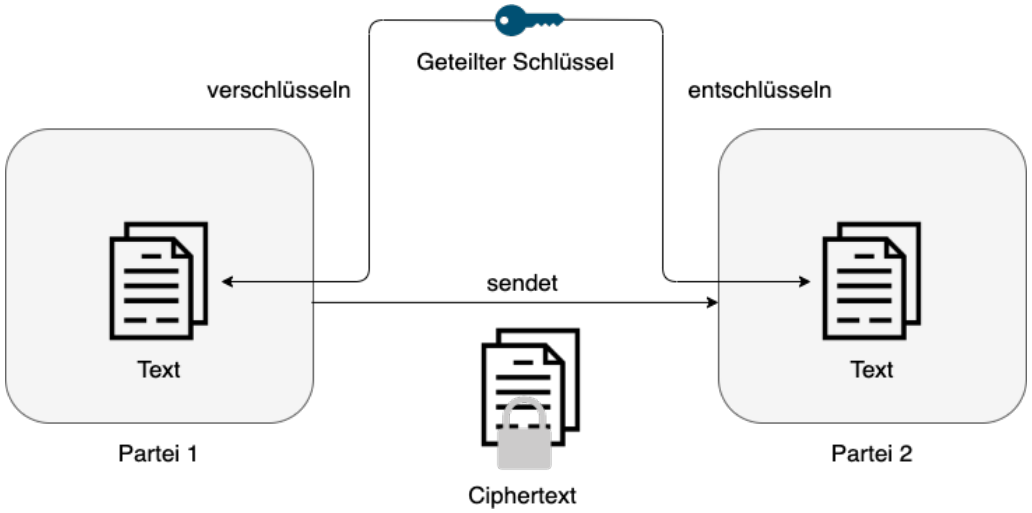

Abbildung 1: Symmetrische Verschlüsselung [13]

2.2 Hardwaresicherheit

Anhand der Beschreibung des symmetrischen Verschlüsselungsverfahren wird deutlich, dass die Schlüssel in gesicherten Umgebung verwaltet und verwendet werden müssen. Anderenfalls kann nicht von der Vertraulichkeit der Daten ausgegangen werden, weil ein Angreifer möglicherweise durch den Diebstahl der Schlüssel den Datenverkehr mitlesen kann [9].

Aus diesem Grund werden in modern iOS-, iPadOS- und watchOS-Geräten Sicherheitschips verbaut, die einen sicheren Coprozessor enhalten. Dieser Coprozessor wird auch *Secure Enclave* genannt und ist ein hardwarebasierter Schlüsselmanager, der von dem Hauptprozessor isoliert ist. Dadurch wird eine weitere Sicherheitsebene in die Verwaltung der sicherheitskritischen Schlüssel umgesetzt. Die Verschlüsselungsschlüssel werden nicht mal der CPU oder dem Kernel des Betriebsystemes offengelegt, weil diese potenziell von einem Angreifer manipuliert werden können. Außerdem enthält der Sicherheitschip eine Hardware-AES-Engine, die für die Ver- und Entschlüsselung von Dateien mithilfe des AES-Verschlüsselungsalgorithmus eingesetzt wird. Während des Start des Systemes tauschen die Secure Enclave und die AES-Engine einen temporären Schlüssel miteinander aus. Mithilfe des temporären Schlüssels können die beiden Komponenten sicher innerhalb des Sicherheitschips kommunizieren [4, 9].

2.3 Schlüssel- und Passwörterverwaltung

Sicherheitskritische Daten wie kryptographische Schlüssel oder Passwörter müssen in einer gesicherten Umgebung verwaltet werden können. Mit der Secure Enclave wurde bereits eine Methode zum Speichern dieser Daten erläutert, in den folgenden Abschnitten werden weitere Methoden, die iOS und iPadOS verwenden, vorgestellt [9].

2.3.1 Schlüsselbund

Mobile Anwendungen müssen oftmals vertrauliche Daten, wie bspw. Passwörter oder Tokens, auf dem Gerät hinterlegen. Das Betriebssystem bietet dafür den sogenannten Schlüsselbund an. Diese Komponente ist eine SQLite-Datenbank, welche die sensiblen Daten in verschlüsselter Form speichert. Der Schlüsselbund befindet sich dabei **nicht** in der Secure Enclave, sondern im Gerätespeicher. Dies ist aber dennoch sicher, weil der Verschlüsselungsschlüssel, welcher für die Entschlüsselung der Daten benötigt wird, sich in der Secure Enclave befindet [9].

2.3.2 Zugriff auf den Schlüsselbund durch Apps

Wie bereits erläutert, ist der Schlüsselbund durch eine verschlüsselte SQLite-Datenbank realisiert, die im Dateisystem hinterlegt ist. Eine App kann den Schlüsselbund verwenden um Passwörter, Kreditkarteninformationen, Zertifikate, Identitäten oder auch kurze Notizen verschlüsselt zu hinterlegen [8]. Diesen Daten können Attribute zugeordnet werden, welche nicht verschlüsselt sind, sodass die Daten auch nach der Verschlüsselung im Schlüsselbund gefunden werden können [7].

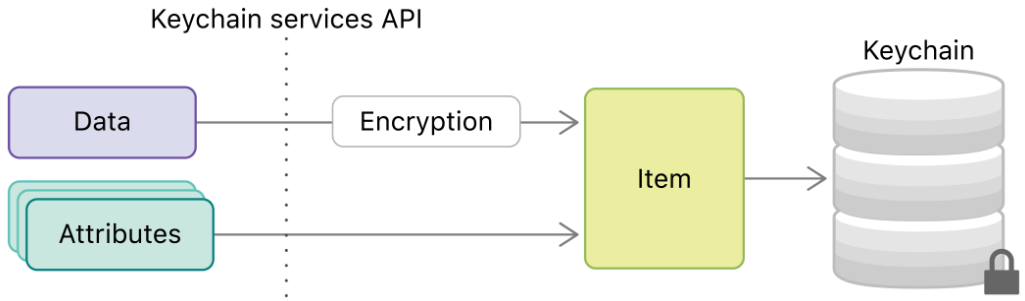

Abbildung 2: Hinzufügen eines Eintrags zu dem Schlüsselbund durch App [7]

Der Zugriff durch eine App auf den Schlüsselbund geschieht durch den *securityd*-Daemon (Abb. 2). Dieser bietet eine API zum Speichern beziehungsweise Verschlüsseln und Auslesen von Schlüsselbundeinträgen. Zudem entscheidet der Deamon auf Basis der Berechtigung *keychain-access-groups*, welche App

auf welchen Eintrag im Schlüsselbund zugreifen darf. Durch die Klassifizierung der Apps in Gruppen, können Informationen innerhalb dieser Gruppen ausgetauscht werden [6]. Somit muss ein*e Nutzer*in sich nur in einer von zwei oder mehr zusammenhängenden Apps authentifzieren und ist unmittelbar auch in den anderen der Gruppe zugehörigen Apps angemeldet. Als Voraussetzung gilt auch hier, dass die Apps von dem gleichen Entwickler*innenteam veröffentlicht wurden.

Apps die neben dem Schlüsselbund auch weitere Daten austauschen möchten, sind einer *application-group* zugehörig. Somit können gemeinsame Container verwendet oder durch eine Interprozesskommunikation Informationen ausgetauscht werden [5].

2.3.3 Keybags

Keybags sammeln und verwalten die Schlüssel der verschiedenen Datensicherheitsklassen. Jeder Keybag verwaltet aber nur die Klassenschlüssel seines Types, im dem folgenden Kapitel werden diese vorgestellt und ihre Funktionsweise erklärt [9]. Für die verschiedenen kryptografischen Verfahren, die bei der Gewährleistung von Sicherheit in iOS zur Anwendung kommen, werden Schlüssel benötigt. Diese werden in so genannten *Keybags* abgelegt und können mit Hilfe des *keybagd* Deamon ausgelesen und gespeichert werden. Es gibt in iOS die fünf *Keybags*, *User*, *Device*, *Backup*, *Escrow* und *iCloud Backup*, die unter anderem auch bei den Datei- beziehungsweise Schlüsselbund-Datensicherheitsklassen zum Einsatz kommen [9].

User Enthält alle normalen, für den Betrieb notwendigen, Klassenschlüssel und wird durch die *Secure Enclave* verwaltet. Ein Zugriff ist erst nach erfolgreichem Entsperren des Gerätes möglich.

Device Wird für das Speichern von Schlüsseln verwendet, die unabhängig von Benutzerkonten benötitgt werden.

Backup Enthält die Schlüssel, die für das Erstellen eines verschlüsselten Backups durch iTunes auf einem Computer, notwendig sind. Dieser *Keybag* wird durch ein Passwort gesichert, welches in iTunes festgelegt wird.

Escrow Dieser *Keybag* enthält Schlüssel für die Verbesserung der Benutzerfreundlichkeit beim Synchronisieren von Geräten. Dafür muss ein Gerät in einem ungesperrten Zustand mit iTunes verbunden werden, wodurch eine Paarung stattfindet, sodass der Gerätecode nicht erneut eingegeben werden muss.

iCloud Backup Ermöglicht die gleiche Funktionsweise zum *Keybag Backup*, für die Realisierung eines Backups durch die iCloud.

3 Zugriffskontrolle

Innerhalb einer App werden Informationen auf unterschiedlichste Art und Weise verwaltet. Beispielsweise sind Anwendungen, die keinerlei Daten ablegen bzw. auf die Datenträger der Plattform speichern, ziemlich langweilig. Häufig möchte man Konfigurationen oder Teile der Geschäftslogik irgendwo auf dem Gerät ablegen, um auch nach Neustart der App die jeweiligen Datensätze vorliegen zu haben und nicht zu verlieren.

Das Speichern von Informationen in Dateien ist also ein sehr beliebtes Mittel, um Daten persistent abrufbar zu gestalten – auch über die Terminierung einer App hinaus. Nur leider ist es auch möglich, dass Apps sensible Daten in Dateien ablegen. Es muss daher einen Mechanismus geben, welcher den unautorisierten Zugriff auf sensible Daten innerhalb einer Datei verbietet bzw. einschränkt. Vor allem soll der Zugriff von anderen (unbekannten) Apps kontrolliert werden. Hierfür existieren auf Apple-Plattformen sogenannte *Datensicherheitsklassen*, die von Apps verwendet werden, um den Zugriff auf die von der App erstellten Dateien zu kontrollieren [9, S. 50].

Neben dem Speichern von Informationen im Allgemeinen benötigt man weitere, sicherheitsrelevante Steuermechanismen für das Ablegen von sensiblen Daten. Vor allem bei sehr sensiblen Informationen, wie beispielsweise Passwörter, kryptographische Schlüssel und Anmelde-Tokens, wünscht man eine sichere Speicherung, damit nicht jeder Benutzer bzw. jede App Zugriff erhält. Hierfür bieten die Apple-Plattformen Schutzmechanismen wie *Access-Control-Lists* (ACL), um Schlüsselbunde abzusichern und den Zugriff auf Ihnen einzuschränken. [9, S. 55].

3.1 Informationsflusskontrolle von Dateien

Bevor der Kontrollmechanismus für das Anlegen und Verwalten von Dateien auf Apple-Plattformen diskutiert wird, schauen wir uns ein Beispiel an. Stellen wir uns vor, es sollen militärische Dokumente verwaltet werden. Sie sollen gelesen und verändert werden können. Da es sich meist um Dokumen-

te handelt, die besondere Diskretion erfordern, sollte klar sein, dass einfache Soldaten nicht dieselben Rechte zum Lesen und Bearbeiten von solchen Dokumenten besitzen wie Generäle. Um den Informationsfluss zu kontrollieren, klassifizieren wir die Dokumente, sodass nur Entitäten mit dem entsprechenden militärischen Rang Lese- und Schreibrechte besitzen. Das Einteilen und fortlaufende Kontrollieren von Informationen und deren Fluss durch Anwendungen wird durch eine *Informationsflusskontrolle*, englisch *Information Flow Control (IFC)*, beschrieben. Kurz zusammengefasst erstellen wir *Gitter* aus Klassen, die letztenendes den Informationsfluss durch die Anwendung beschreiben [12].

Beim nächsten Beispiel handelt es sich um eine App, die beispielsweise Kontaktdaten in eine Datei auf das Gerät (iOS) speichert. Auch hier möchten wir sicherstellen, dass die Informationen nur unter bestimmten Umständen ausgelesen werden können. Genau wie bei der Informationsflusskontrolle. Wenn eine App auf iOS eine Datei schreibt, wird dieser Datei laut [2, 9] zusätzlich eine Datensicherheitsklasse zugewiesen, um dieses Ziel zu erreichen. Eine Sicherheitsklasse gibt an, wann auf die Informationen zugegriffen bzw. wann die Datei gelesen werden darf. Hierfür wurden vier Klassen implementiert, die im Folgenden kurz erläutert werden.

Vollständiger Schutz. Die Verschlüsselung des Klassenschlüssels wird mithilfe des Benutzercodes und der eindeutigen UID *(unique identifier)* des Geräts realisiert. Nachdem das Gerät gesperrt wird, wird der entschlüsselte Klassenschlüssel verworfen und erst bei erneuter Entsperrung entschlüsselt. Dateien mit dieser Sicherheitsklasse können also nicht gelesen oder geschrieben werden, solange das Gerät gesperrt ist.

Geschützt, außer wenn offen. Wenn diese Sicherheitsklasse für eine Datei verwendet wird, kann diese erstellt werden, auch wenn das Gerät gesperrt wird. Ist der Erstellungsvorgang abgeschlossen, kann diese Datei erst wieder gelesen werden, wenn das Gerät entsperrt wird.

Geschützt bis zur ersten Authentifizierung. Genau wie die Klasse *Vollständiger Schutz* wird mithilfe des Benutzercodes und der UID des Geräts ein Klassenschlüssel entschlüsselt. Der Unterschied liegt nun darin, dass dieser entschlüsselte Klassenschlüssel beim Sperren des Geräts nicht verworfen wird. Erst bei Neustart muss dieser erneut entschlüsselt werden. Dieser Schutz ist vergleichbar mit der Verschlüsselung der Festplatten bei Desktoprechnern.

Kein Schutz. Der Klassenschlüssel wird hier nur mithilfe der UID des Geräts entschlüsselt bzw. geschützt. Auch wenn einer Datei keine Sicherheitsklasse zugewiesen wurde, wird diese vom Betriebssystem verschlüsselt gespeichert.

3.2 Kontrolle von Schlüsselbundzugriffen

Damit nicht jeder auf die Schlüsselbunde eines Geräts zugreifen kann, werden *Access-Control-Lists* (ACLs) verwendet, die Richtlinien für den Zugriff und Authentifizierung bereitstellt. Das Ziel ist unter anderem, dass der Benutzer des Geräts anwesend ist und sich mittels Touch ID oder anderen Authentifizierungsmethoden ausweist. ACLs werden in der *Secure-Enclave* geprüft und erst dann an den Kernel weitergegeben, falls alle Bedingungen erfüllt wurden [9].

Dies allein ist jedoch nicht Schutz genug gegen einen Angreifer. Betrachten wir folgendes Beispiel, wird die Problematik klarer. Falls das Gerät in die Hände eines Angreifers gerät, wird dieser versuchen, Zugriff auf Schlüsselbunde zu bekommen. Hierfür legt sich der Angreifer einfach einen neuen Fingerabdruck an und authentifiziert sich mit diesem. Falls nur die Anwesenheit eines authentifizierten Benutzers geprüft werden würde, würde also nicht sichergestellt werden, dass nur die bis zum Angriff bekannten Benutzer anwesend sind. Hierfür kann zusätzlich kontrolliert werden, ob eine Touch ID verändert bzw. wann sie erstellt wurde, um den Zugriff zu kontrollieren [9].

4 Apple File System

Das *Apple File System* (*APFS*) ist das seit iOS 10.3 auf Mobilgeräten und seit macOS High Sierra 10.13 auf Computern verwendete Dateisystem von Apple. Es ist der Nachfolger von *HFS+* und ähnelt den Dateisystemen BRTFS und ZFS in einigen Punkten, hat jedoch vor allem im Bereich der Sicherheit und Verschlüsselung einige zusätzliche Features [10, 3].

4.1 Volumes und Container

Unter *APFS* ist es möglich mehrere Volumes auf demselben physischen Speicher zu verwenden [1]. Ein Volume befindet sich dabei immer in einem Container. Wird also ein Speichermedium mit einer Speicherkapazität von 500 GiB verwendet, ist es also möglich einen zum Beispiel 400 GiB grossen *APFS*-Container zu erstellen. In diesem Container können dann beliebig viele Volumes mit einer Größe von 400 GiB erstellt werden.

Wird dies getan, so benutzen alle Volumes dieselben 400 GiB des Speichermediums, konkurrieren andererseits damit aber auch um denselben Speicherplatz. Werden zum Beispiel 3 400 GiB grosse Volumes auf dem Container erstellt, wobei die einzelnen Volumes jeweils tatsächlich 100 GiB, 30 GiB und 20 GiB verwenden, so ist der freie Platz eines jeden Volumes 250 GiB [10].

Sollte dabei der Speicherplatz des Containers nicht mehr ausreichen, so ist es möglich den Container zu erweitern. Auf diese Weise werden logische Volumes von physischen Speichergegebenheiten abstrahiert und ermöglichen eine variable und dynamische Nutzung des verfügbaren Speichers.

4.2 Datensicherheit

4.2.1 Daten

Zentraler Punkt der Verschlüsselung unter iOS mit *APFS* ist eine von Apple als *Datensicherheit* betitelte Technologie [9, S. 48 ff]. Diese beinhaltet vor allem zwei wesentliche Merkmale.

Das erste ist, dass jede Datei auf dem Dateisystem mit einem eigenem Schlüssel verschlüsselt wird. Dies gilt auch, wenn eine Datei geklont wird [9, S. 49].

Das zweite Merkmal sind die *Klassenschlüssel*, wobei jede Datei einer spezifischen Klasse angehört. Mit diesem Schlüssel wird jede Datei, deren Metadaten sie der entsprechenden Klasse zuordnen nochmals verschlüsselt [9, S. 49].

Auf diese Weise wird gewährleistet, dass nur Apps, welche die Zugriffsrechte für die entsprechende Klasse besitzen auch wirklich auf die Dateien dieser Klasse zugreifen können.

4.2.2 Metadaten

Im Gegensatz zu den eigentlichen Dateien, teilen sich alle Metadaten einen Schluessel [9, S. 50]. Dieser wird einmalig bei der Einrichtung des iOS-Gerätes erstellt und in der *Secure Enclave* gespeichert. Weitergehend wird dieser Schlüssel mit einem weiteren Schlüssel verschlüsselt, der allerdings im *Effaceable Storage* gespeichert ist.

Zweck dieses Schlüssels ist es über die Einstellungen oder per Fernzugriff schnell gelöscht werden zu können. Auf diese Weise geht der eigentliche Schlüssel für die Metadaten verloren und sämtliche Daten auf dem Gerät sind unwiderruflich verschlüsselt und somit vor dem Zugriff Dritter geschützt.

4.2.3 Hardware

Die Verschlüsselung sowohl der eigentlichen Daten, als auch der Metadaten wird weitergehend dadurch unterstützt, dass die eigentlichen Schlüssel nie an den Hauptprozessor weitergegeben werden [9, S. 50]. Stattdessen werden alle Verschlüsselungen und Entschlüsselungen nur in der *Secure Enclave* durchgeführt.

4.2.4 Appzugriff

Der Zugriff von Apps auf Daten des Nutzers wird vor allem durch einen sogenannten *Data Vault* geregelt [9, S. 47]. Dieser verwaltet die Zugriffsrechte auf verschiedene Arten von nutzerbezogenen Informationen. Dabei kann der Nutzer jederzeit einsehen, welche Apps welche dieser Zugriffsrechte haben und diese entziehen oder neue gewähren.

Einzige Ausnahme bilden dabei einige Systemapplikationen wie Kalender, Kamera, Medien oder aehnliches. Diese Systemapplikationen erfordern auf Grund ihrer Funktionalität einen Zugriff auf die persönlichen Daten und setzen diesen deshalb voraus.

Auf diese Weise hat der Nutzer zu jedem Zeitpunkt volle Information und Kontrolle ueber den Verbleib seiner Daten.

4.3 Copy-On-Write

Um unnötige Kopien oder korrupte Dateien zu vermeiden, also die Integrität der gespeicherten Daten sicher zu stellen, setzt *APFS* auf einen Copy-On-Write Ansatz fuer die Metadaten von Dateien [10]. Diese Metadaten werden also erst geupdatet, sobald eine Datei beziehungsweise ein Block des Speichermediums wirklich verändert werden soll.

Durch die Realisierung dieser Veränderung mittels einer atomaren Transaktion *Atomic Safe-Save* wird das Dateisystem weitergehend vor korrupten Daten geschützt, da auf diese Weise sichergestellt wird, dass eine Transaktion entweder vollständig oder gar nicht durchgeführt wird.

4.4 Klonen

Der Copy-On-Write Ansatz vom *APFS* wird allerdings nicht nur bei den Metadaten eingesetzt, sondern auch bei dem tatsächlichen Inhalt von Dateien. Wird eine Datei unter APFS geklont, so wird diese nicht komplett physisch auf eine andere Stelle des Speichermediums kopiert. Stattdessen werden le-

diglich Verweise auf die ursprüngliche Datei erstellt. Werden zu dem Klon weitere Blöcke hinzugefügt oder verändert, so werden für diese Neuerungen weitere Verweise erstellt, während die bestehenden Verweise auf die Ur-Datei erhalten bleiben.

Dieser Ansatz beschleunigt das Kopieren von Dateien enorm, da nicht die gesamte Datei kopiert wird, sondern lediglich ein Link – beziehungsweise eine neue Version der Datei – erzeugt wird [1]. Da somit dieselbe Datei nicht zweimal vorhanden sein muss, wird damit außerdem Speicherplatz gespart.

Ein weiterer Vorteil dieses Vorgehens ist es weiterhin, dass es damit einfach ist eine Art Versionshistorie von Dateien zu erstellen und damit gemachte Änderungen besser nachvollziehen zu können oder einfacherer zu vorherigen Versionen zurück zu springen.

Die Nachvollziehbarkeit von Transaktionen wird außerdem durch eine bessere Auflösung der Zeitstempel im System (1 ns) verglichen mit *HFS+* (1 s) gefördert [11].

4.5 Backups

Das *APFS* unterstützt mit *Snapshots* eine Funktion zum Erstellen von Backups des Dateisystems. Dieses funktioniert ähnlich wie das Klonen von Dateien, allerdings mit der Einschränkung, dass ein Snapshot nicht veränderbar, also read only ist [11].

Anders formuliert, erstellt ein Snapshot also einen Klon aller Dateien des Dateisystems zum aktuellen Zeitpunkt. Soll das System also mittels Snapshot auf einen vorherigen Zustand zurückgesetzt werden, so müssen keine Dateien kopiert werden, sondern lediglich die Verweise auf bestehende Dateien verändert, beziehungsweise auf den Stand des Snapshots zurückgesetzt werden.

Die einzige Ausnahme bilden dabei Bloecke, die verändert oder gelöscht wurden. Diese werden dem Snapshot hinzugefügt und müssen im Falle einer Wiederherstellung tatsächlich kopiert werden.

Durch die standardmäßige Verschlüsselung von Daten mittels Klassen- und Dateischluesseln, welche beim klonen von Dateien bestehen bleibt sind Backups automatisch verschlüsselt [9, S. 50]. Da die UID des Gerätes Teil der Klassenschlüssel ist [9, S. 50], wird außerdem gewährleistet, dass Backups nur auf dem Gerät auf dem sie erstellt wurden wiederhergestellt werden können. Bei Verlust des Speichermediums ist ein Auslesen des Snapshots auf anderen Geräten also nicht möglich.

4.6 Vergleich zu MacOS

Das *APFS* wird mittlerweile sowohl auf iOS als auch auf MacOS Geräten eingesetzt. Allerdings sind die Features von APFS auf iOS und MacOS jedoch unterschiedlich. Der Hauptunterschied ist dabei die Verwendung von *FileVault* unter MacOS [9, S. 56 ff], im Gegensatz zur *Datensicherheit* unter iOS [9, S. 48 ff].

FileVault ist dabei dafuer zustaendig, die Volumes des *APFS* zu verschlüsseln. Dieser Schlüssel muss bei jedem Startvorgang vom Nutzer unter Eingabe seiner Anmeldeinformationen freigegeben werden.

Ein weiterer Unterschied zwischen der Verwendung vom *APFS* unter iOS und MacOS sind außerdem die benötigten Volumes eines Containers. Während iOS mit einem Systemvolume zur Speicherung der Systemdaten und einem Datenvolume fuer die Nutzerdaten auskommt [9, S. 49], benötigt MacOS 3 weitere Volumes [9, S. 48]. Diese sind zum einen das Preboot-Volume, welches Informationen über alle Volumes des Containers enthält und vor allem zum Booten benötigt wird. Zum anderen das VM-Volume, welches dem Austausch von Datenspeicher dient. Und zum letzten dem Wiederherstellungsvolume, welches *recoveryOS* beinhaltet und damit für Systemwiederherstellungen verantwortlich ist.

5 Senden und empfangen von Nachrichten

iOS unterstützt das Senden und Empfangen von verschlüsselten Nachrichten durch *S/MIME. Secure/Multipurpose Internet Mail Extensions* ist ein Standard zum Signieren und Verschlüsseln von Nachrichten und wird unter anderem bei dem Versenden von verschlüsselten E-Mails verwendet.

Für das Versenden einer verschlüsselten E-Mail wird das Zertifikat des/der Empfänger*in benötigt. Dieses wird zum Signieren verwendet und fungiert beim Verschlüsseln als öffentlicher Schlüssel. Der Austausch des Zertifikats kann über den Empfang einer E-Mail oder auch über einen direkten, manuellen Austausch der Zertifikatsdatei stattfinden.

Literatur

[1] Apple. About apple file system. https://developer.apple.com/documentation/foundation/file_system/about_apple_file_system. [Online, Aufgerufen 08.11.2020].

[2] Apple. Apple Developer Documentation. https://developer.apple.com/documentation/foundation/nsfileprotectiontype. [Online Aufgerufen 06.11.2020].

[3] Apple. Apple file system reference. https://developer.apple.com/support/downloads/Apple-File-System-Reference.pdf. [Online, Aufgerufen 08.11.2020].

[4] Apple. Hardwaresicherheit – Übersicht. https://support.apple.com/de-de/guide/security/secf020d1074/web. [Online; Aufgerufen 05.11.2020].

[5] Apple. Keychain application groups. https://developer.apple.com/documentation/bundleresources/entitlements/com_apple_security_application-groups. [Online; Aufgerufen 05.11.2020].

[6] Apple. Keychain item groups. https://developer.apple.com/documentation/security/keychain_services/keychain_items/sharing_access_to_keychain_items_among_a_collection_of_apps. [Online; Aufgerufen 05.11.2020].

[7] Apple. Keychain items. https://developer.apple.com/documentation/security/keychain_services/keychain_items. [Online; Aufgerufen 05.11.2020].

[8] Apple. Keychain services. https://developer.apple.com/documentation/security/keychain_services. [Online; Aufgerufen 05.11.2020].

[9] Apple. Sicherheit der Apple-Plattformen. https://manuals.info.apple.com/MANUALS/1000/MA1902/de_DE/

apple-platform-security-guide-d.pdf. [Online; Aufgerufen 27.10.2020].

[10] S. Grüner. Apple erstellt eigenes modernes dateisystem. https://www.golem.de/news/apfs-apple-erstellt-eigenes-modernes-dateisystem-1606-121496.html. [Online, Aufgerufen 08.11.2020].

[11] L. Hutchinson. Digging into the dev documentation for apfs, apple's new file system. https://arstechnica.com/gadgets/2016/06/digging-into-the-dev-documentation-for-apfs-apples-new-file-system/. [Online, Aufgerufen 08.11.2020].

[12] G. Smith. Principles of Secure Information Flow Analysis. *Springer US*, 2007.

[13] Özlem Sönmez. Symmetric key management: Key derivation and key wrap. https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/03/soenmez.pdf. [Online; Aufgerufen 05.11.2020].