Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorarbeit

Dennis Alexander Süwolto

Entwicklung eines Konzepts zur Virtualisierung von Applikationen auf mobilen Geräten

*Fakultät Technik und Informatik*
*Department Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

**Dennis Alexander Süwolto**

# Entwicklung eines Konzepts zur Virtualisierung von Applikationen auf mobilen Geräten

Bachelorarbeit eingereicht im Rahmen Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Abgegeben am 23.06.2016

**Dennis Alexander Süwolto**

**Thema der Bachelorarbeit**
Entwicklung eines Konzepts zur Virtualisierung von Applikationen auf mobilen Geräten

**Stichworte**
Virtualisierung, Hypervisor, Microkernel, Container, Applikationssicherheit, BYOD

**Kurzzusammenfassung**
Diese Arbeit beschäftigt sich mit der Erstellung eines Konzepts zum Thema Virtualisierung von Anwendungen auf mobilen Geräten, um damit das Sicherheitsniveau dieser Anwendungen zu erhöhen. Für die Ausarbeitung werden die Grundlagen der Virtualisierung vorgestellt, eine Marktanalyse zu aktuellen Lösungen durchgeführt, sowie ein fiktives Unternehmensszenario erstellt. Auf Basis dieses Wissens wird eine Strategie entwickelt, welche die Grundlage für die Realisierung des Vorhabens darstellt. Abgeschlossen wird das Konzept durch die Realisierung der Strategie in Form eines Entwurfs für eine Container-Virtualisierung-Lösung.

**Dennis Alexander Süwolto**

**Title of the paper**
Development of a concept for virtualization of applications on mobile devices

**Keywords**
Virtualization, Hypervisor, Microkernel, Container, Security of applications, BYOD

**Abstract**
This work focuses on the development of a concept for virtualization of applications on mobile devices in order to increase the security level of these applications. For this elaboration the basics of virtualization are presented, a market analysis to current solutions is carried out, and a fictitious company scenario is created. Based on this knowledge, a strategy is being developed, which is the basis for the realization of the project. The concept is concluded by the realization of the strategy in form of a draft for a container virtualization solution.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ABI | Application binary interface |
| ARM | Advanced RISC Machines |
| BDSG | Bundesdatenschutzgesetz |
| BYOD | Bring Your Own Device |
| COPE | Corporate-Owned, Personally Enabled |
| DBT | Dynamic Binary Translation |
| HAL | Hardware Abstraction Layer |
| ICC | Inter-Component Communication |
| iOS | Mobiles Betriebssystem der Firma Apple |
| IPC | Inter-Process Communication |
| ISA | Instruction Set Architecture |
| KVM | Kernel-based Virtual Machine |
| LOC | Lines Of Code |
| LXC | Linux Container |
| MDM | Mobile Device Management |
| MVP | Mobile Virtualization Plattform |
| mvpd | Mobile Virtualization Plattform Deamon |
| mvpkm | Mobile Virtualization Plattform Kernel Modul |
| OKL4 | Open Kernel Labs L4 microkernel |
| OS | Operating System |
| RISC | Reduced Instruction Set Computer |
| SDK | Software Development Kit |
| SEL4 | Secure Embedded L4 |
| TCB | Trusted Computing Base |
| TMG | Telemediengesetz |
| VLAN | Virtual Local Area Network |
| VM | Virtual Machine |
| VMM | Virtual Machine Monitor |
| VPN | Virtual Private Network |
| WLAN | Wireless Local Area Network |

# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1 Einleitung

Innerhalb kurzer Zeit hat sich der Funktionsumfang der Mobiltelefone vom reinen Kommunikationsmittel zu leistungsfähigen tragbaren Computern, auch Smartphones genannt, entwickelt. Diese leistungsstarken Geräte sind in der Lage Software auszuführen, welche bisher nur Notebooks oder Desktop-Rechnern vorbehalten war. Dadurch ist in den letzten Jahren ein gigantischer Markt für Anwendungen für diese mobilen Geräte entstanden. Allein der „Play Store" vom Unternehmen Google bietet für sein mobiles Betriebssystem Android zurzeit mehr als 186 Millionen Applikationen an. Ob Spiele, Multimedia oder Büroanwendungen, jeglicher Themenbereich ist meist mehrfach mit Anwendungen abgedeckt. Viele dieser Anwendungen wie zum Beispiel E-Mail-, Onlinebanking- oder Verkehrsinformationsprogramme sind aus dem Alltag gar nicht mehr weg zu denken.

Gleichzeitig zur weltweit rasanten Verbreitung der Smartphones haben sehr schnell viele Unternehmen die Vorteile dieser Technologie für sich entdeckt. Erfolg im Business ist heutzutage mit der Geschwindigkeit verknüpft. E-Mails, Termine und Dokumente müssen überall und zu jeder Zeit zugänglich sein. Durch ihren großen Funktionsumfang und immer besser werdenden Vernetzungsmöglichkeiten haben mobile Geräte, wie Smartphones und Tablets, schnell ihren Einzug in geschäftskritische Infrastrukturen gefunden.

Dies stellt die Unternehmen vor die Herausforderung die Sicherheitsbedürfnisse ihrer Organisation auf mobilen Geräten sicherzustellen. Beispielsweise sollten aktuelle Version der Betriebssysteme verwendet werden, um Risiken durch bekannte Sicherheitslücken zu vermeiden.

Zudem sollten die Daten und Anwendungen des Unternehmens vor unautorisierten Zugriff, welche durch den Verlust oder Diebstahl der Geräte entstehen können, geschützt werden.

Neben entsprechender IT-Infrastruktur kommt es dabei natürlich auch auf das entsprechende Know-how des verantwortlichen IT-Personals an. In der Praxis ist es jedoch oft schwierig, ein angemessenes IT-Sicherheitsniveau zu erreichen. Besonders schwer wird es dabei, wenn es sich um die Geräte im Besitz der Angestellten handelt.

Modelle wie „Bring your own device" (BYOD) werden von immer mehr Unternehmen angewendet. Laut einer Studie von Duo Labs von Anfang dieses Jahres geschieht jede fünfte Anmeldung an einem Unternehmensnetzwerk inzwischen mit mobilen Geräten, Tendenz steigend. Leider schafft diese Entwicklung auch neue Sicherheitsrisiken für die Unternehmen.

Weiter geht aus der Studie hervor, dass 92% der Android-Geräte nicht verschlüsselt sind, 34% keine Authentifizierungsabfrage beim Sperrbildschirm aktiviert haben und 5% der Geräte „jailbroken" (engl. Gefängnisausbruch) sind, also die unautorisierte Entfernung von Sicherheitsbeschränkungen der jeweiligen Geräte zum Beispiel durch die Freischaltung der Root-Zugriffsrechte. Ein zusätzliches Sicherheitsrisiko der Android-Geräte ist die Update-Politik des Betriebssystems. So verwenden immer noch 32% der Geräte die Android Version 4.0 oder niedriger, wodurch die Schließung von sicherheitskritischen Lücken auf den betroffenen Geräten verhindert wird (vgl. [HANL16]).
Der Einsatz von mobilen Geräte stellt die Unternehmen vor großen Herausforderungen, da die Verantwortung und der Besitz des Gerätes bei den Angestellten liegen. In öffentlichen Verwaltungen ist daher der Einsatz privater Datenverarbeitungsgeräte bisher prinzipiell untersagt. Trotzdem erscheint das Modell BYOD für viele Unternehmen als attraktiv. Die Angestellten können berufliche und private Aufgaben kombinieren, was zu mehr Motivation und einer deutlichen Steigerung an Effizienz und Produktivität beitragen kann. Das Unternehmen erscheint so als flexibler und attraktiver Arbeitgeber und spart selber Geld für die Anschaffung solcher Geräte. Eine hinreichende Möglichkeit diese Probleme zu lösen bietet die Virtualisierungstechnologie.
Durch Virtualisierung wird es möglich, die privaten Daten und Anwendungen des Angestellten durch die Erzeugung eines parallel laufenden Systems, welches für die Unternehmensseite genutzt wird, zu isolieren und so voreinander zu schützen.
Durch die Vielzahl der Anwendungen und der unterschiedlichen Bezugsquellen wird es den Betreibern erschwert, die Verbreitung von versteckter Schadsoftware zu verhindern. Zudem benötigen Virenscanner Muster zur Erkennung der Schadsoftware, welche dann oftmals durch die Erzeuger dieser Schadsoftware so verändert werden, dass diese von den Virenscannern nicht mehr erkannt werden können.
Virtualisierungstechniken können auch hier weiterhelfen. Denn durch die Isolierung einer Anwendung und durch Verwendung weiterer intelligenter Mechanismen, können die Auswirkungen durch Schadsoftware für das restliche System minimiert werden.

## 1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung eines Konzepts zur Virtualisierung von Anwendungen auf mobilen Geräten. Deshalb soll es in dieser Arbeit darum gehen, die unterschiedlichen Virtualisierungsvarianten auf Basis ihrer Eigenschaften und Besonderheiten für die Einsatzfähigkeit auf mobilen Geräten zu analysieren und zu bewerten. Die anschließend im Konzept entwickelte Lösung für das Vorhaben soll ein konzeptioneller Entwurf einer Anwendung zur Erfüllung des Vorhabens darstellen. Dabei wird nur auf die Architektur und auf die Funktionalität ihrer Komponenten Wert gelegt. Der Einsatz dieser

Virtualisierungstechniken soll hierbei auf die logische Isolierung der jeweiligen Anwendungen im Rahmen eines BYOD-Modells, innerhalb der Unternehmenswelt abzielen, um so eine geschützte Unternehmensumgebung auf den privaten Geräten der Angestellten herzustellen. Dazu wird neben einer Analyse der bestehenden Virtualisierungsvarianten eine Marktanalyse durchgeführt, welche die fortschrittlichsten Lösungen der einzelnen Virtualisierungsvarianten untersucht.

## 1.2 Themenabgrenzung

Die Virtualisierungstechnologie ist ein sehr breitgefächertes Themengebiet zu welchem eine große Zahl an Publikationen verfügbar stehen. Diese Arbeit ist keine vollständige Einführung in das Thema der Virtualisierung, sondern beschäftigt sich ausschließlich mit den Varianten und Verfahren die für den Einsatz auf mobile Geräte verfügbar beziehungsweise einsetzbar wären. Da es nicht ein mobiles Gerät und das Betriebssystem gibt, wird nicht auf die speziellen technischen Einzelheiten der Virtualisierung eingegangen. Der Fokus liegt auf den unterschiedlichen Prinzipien und dadurch entstehenden Vorteilen für den Einsatz auf diesen Geräten. Ebenfalls wird bei dem konzeptionellen Entwurf eine Lösung vorgestellt wird, aber nicht auf die technischen Details des folgenden Entwicklungsprozesses dieser Lösung eingegangen.

## 1.3 Struktur der Arbeit

Diese Arbeit ist in sechs aufeinander aufbauenden Kapitel gegliedert. Von Kapitel zu Kapitel wird eine tiefergehende Spezialisierung der Themen vorgenommen. Begonnen wird mit der Einführung in das Thema. Dabei wird aufgezeigt, warum die Virtualisierungstechnologie eine vielversprechende Lösung für den Einsatz von BYOD-Modellen sein kann.
Im zweiten Kapitel werden die allgemeinen Grundlagen der Virtualisierung mit ihren unterschiedlichen Formen und Prinzipien vorgestellt. Zudem werden die Unterschiede zwischen herkömmlicher Server- und Desktop-Virtualisierung zur Virtualisierung auf mobilen Geräten deutlich gemacht.
Das dritte Kapitel umfasst eine Marktanalyse aktueller Virtualisierungslösungen für den Einsatz auf mobilen Geräten. Damit soll ein Überblick über die aktuelle Lage am Markt gegeben. Außerdem dient die Marktanalyse als Basis für das zu entwickelnde Konzept.
Das im vierten Kapitel erstelle fiktive Szenario fungiert neben der Marktanalyse als Grundlage für die Auswahl fundamentaler Entscheidungen innerhalb des Konzepts.

Als Hauptteil dieser Arbeit folgt im fünften Kapitel das Konzept zur Virtualisierung von Anwendungen auf mobilen Geräten. Dabei wird die optimale Lösung für das dort vorgestellte Vorhaben ausgewählt und vorgestellt.
Abschließend wird der Inhalt, aufbauend auf den gewonnenen Erkenntnissen der Arbeit und ihrer Ergebnisse zusammengefasst und ein Ausblick gegeben, an welchen Stellen Weiterentwicklungen der Virtualisierungstechnologie vorgenommen werden sollte.

# 2 Grundlagen der Virtualisierung

*„If it's there and you can see it - it's REAL*
*If it's there and you can't see it - it's TRANSPARENT*
*If it's not there and you can see it - it's VIRTUAL*
*If it's not there and you can't see it - it's GONE!“*
Roy Wilks, 1983

Im folgenden Kapitel werden neben der Herkunft und der Bedeutung der Virtualisierung, auch der heutige Stand und die unterschiedlichen Konzepte dieser Technologie vorgestellt.

## 2.1 Was ist Virtualisierung?

Es gibt keine eindeutige Definition des Begriffs „Virtualisierung”, aber er wird oftmals als Synonym für die Emulation oder Simulation von Hardware-Ressourcen verwendet. Das Konzept der Virtualisierung ermöglicht daher die Entkopplung der Hardware von der darauf laufenden Software beziehungsweise des Betriebssystems und impliziert so einem höheren Grad an Flexibilität. Dieses Konzept der Entkopplung zweier Schichten beziehungsweise Systeme wird in der folgenden Arbeit als Definition für die Virtualisierung verwendet.
Dass es bis heute keine einheitliche Definition für den Begriff Virtualisierung in der Informatik gibt, kann an der Tatsache liegen, dass Virtualisierung eine enorme Vielzahl an unterschiedlichen technologischen Möglichkeiten bietet.

Obwohl diese Technologie in der neueren Zeit sehr populär wurde, handelt es sich dabei nicht um eine neu entwickelte Technologie. Die Wurzeln der Virtualisierung finden sich in den späten fünfziger Jahren des letzten Jahrhunderts mit Ursprung im Mainframe-Umfeld.
Im Jahre 1959 wurde das Konzept des Multitasking in der Abhandlung „Time Sharing In Large Fast Computers” von Christopher Strachey vorgestellt. Auf Basis dieses Konzeptes konnten die Auslastung der teuren Mainframes verbessert werden. Hierdurch wurde nicht nur die Auslastung der kostenintensiven Prozessoren verbessert, es konnte dadurch auch dem

Benutzer vermittelt werden, auf einer eigenen dedizierten Maschine zu arbeiten, obwohl er sich tatsächlich die Hardware mit anderen Nutzern teilte (vgl. [PICHT09]).

In den siebziger Jahren beschäftigten sich Gerald Popek und Robert Golfberg mit der Frage, welche Anforderungen notwendig wären, um effektiv zu virtualisieren. In ihrer Abhandlung „Formal Requirements for Virtualizable Third Generation Architectures“ aus dem Jahr 1974 beschreiben sie drei elementare Eigenschaften, die für eine virtuelle Maschine entscheidend sind, damit ein beliebiges Programm ausgeführt werden kann während das Steuerungsprogramm, auch VMM (Virtual Machine Monitor) genannt resident bleibt: Effektivität, Ressourcenkontrolle und Gleichwertigkeit. Diese grundlegenden Forderungen an eine Virtualisierung, welche bis heute ihre Gültigkeit behalten haben, werden im Unterkapitel 2.5 detaillierter vorgestellt.

Die Virtualisierungstechnologie wurde noch bis in die achtziger Jahre, insbesondere durch den damaligen Marktführer im Mainframe-Bereich IBM, weiterentwickelt. Jedoch wurde die weitere Entwicklung in den achtziger und neunziger Jahren dann durch das Aufkommen kostengünstiger x86-Hardware unwirtschaftlich. Günstige x86-basierte Workstations und Server Hardware verbreitete sich rasant und sorgte so für die Ablösung des Terminalcomputings.
Im Laufe der Zeit hat sich auch im x86-Hardware Bereich der Ansatz verbreitet, die ungenutzten Ressourcen der unterschiedlichen Systeme zu optimieren, anstatt die IT-Umgebung weiter aufzurüsten.
Ergänzend zu den Kostenaspekten im Bereich der Großrechner, wurde beim Ansatz der Virtualisierung auf der x86-Architektur auch die Erhöhung der Verfügbarkeit immer wichtiger. (vgl. [BALM14])

Heute wird das Virtualisierungskonzept in vielen unterschiedlichen Bereichen, wie u.a. beim „Timesharing” der CPU, der virtuellen Speicherverwaltung der Betriebssysteme, Erzeugung logischer Teilnetze beim VLAN (Virtual Local Area Network) oder bei der Virtualisierung ganzer Maschinen genutzt und trägt dort zur Optimierung der jeweiligen Technologien entscheidend bei.
Äquivalent zur Definition gibt es keine einheitliche Bezeichnung für die unterschiedlichen Virtualisierungsformen, die im Laufe der Zeit entstanden sind. In der folgenden Auflistung werden die Bezeichnungen, die in dieser Arbeit verwendet werden, mit ihren in der Literatur verbreiteten Alternativbezeichnungen vorgestellt. Bei den Begriffen innerhalb der Klammern handelt es sich um die Alternativbezeichnungen.

**Übersicht der Virtualisierungsvarianten:**

Systemvirtualisierung

- Hypervisor (*Virtual Machine Monitor*)
  - Typ 1 (*native*, *bare-metal*)
    - Ausprägung als Voll- und Paravirtualisierung
  - Typ 2 (*hosted*)
    - Ausprägung als Voll- und Paravirtualisierung
- Microkernel
- Microvisor

Anwendungsvirtualisierung

- Container-Virtualisierung (*Betriebssystem-Virtualisierung*, *OS-level virtualization)*
- Remote-Desktop- Dienst (*Terminal*)

## 2.2 Ziele der Virtualisierung

Der Einsatz einer Virtualisierungstechnologie ist mit unterschiedlichen Absichten verbunden. Als primäres Ziel wird beim Konzept der Virtualisierung die logische Trennung zweier aufeinander folgenden Schichten wie zum Beispiel die Trennung der Software von der Hardware angestrebt. Heutzutage gibt es zudem Virtualisierungsformen, welche rein im Softwarebereich diese Art der Abstraktion durchführt.
Das Ziel der Entkopplung des logischen Systems von der physikalischen Hardware ermöglicht zahlreiche weitere Möglichkeiten, welche als sekundäre Ziele der Virtualisierung angesehen werden.
Neben der Optimierung der Ressourcenausnutzung durch die Konsolidierung der physikalischen Maschinen, sodass eine größere Maschine viele kleinere Maschinen ersetzt und so das Kosten-Leistungsverhältnis verbessert, bietet die Virtualisierungstechnologie zudem die Möglichkeit, die Verfügbarkeit und Sicherheit der Systeme zu erhöhen.
Verfügbarkeit und Ausfallsicherheit können durch den Einsatz der Virtualisierungstechnologie verbessert werden, indem mehrere physikalische Maschinen durch Virtualisierung zu einem logischen System zusammengefasst werden, wie es zum Beispiel bei Cloud-Diensten geschieht. Die IT-Sicherheit kann wiederum durch die logische Isolierung der einzelnen virtuellen Maschinen erhöht werden, da hierdurch der Zugriff auf andere VMs innerhalb derselben physikalischen Maschine unterbunden wird und Schadsoftware im „worstcase“ nur eine VM kompromittieren kann. Beim Entwurf einer Virtualisierungsarchitektur für mobile Geräte müssen zudem auch die verschiedenen Kommunikationskanäle dieser Geräte berücksichtigt werden.

## 2.3 Prinzipien der Virtualisierung

Bei einem System ohne den Einsatz von Virtualisierung greift das Betriebssystem als einziges laufendes System direkt auf die Hardware der Maschine zu und verwaltet diese. Vom Betriebssystem wird eine Anwendungsumgebung bereitgestellt, in welcher eine beliebige Anzahl an Anwendungen im selben Systemkontext ausgeführt werden kann.

Im Gegensatz dazu wird bei der Verwendung der Virtualisierung eine zusätzliche Schicht zwischen der Betriebssystem- und der Hardwareschicht eingefügt. Die Hardware wird somit nur noch von dieser Virtualisierungsschicht verwaltet. Das Betriebssystem wird bei dieser Variante der Virtualisierung in eine virtuelle Maschine verschoben, welche ein eigenständiges logisches System darstellt. Das Betriebssystem verhält sich weiterhin, als ob es alleine auf einer physikalischen Maschine ausgeführt wird (vgl. [EISE11]). Dieses Prinzip ist in Abbildung 1 als Schichtenmodell grafisch dargestellt.

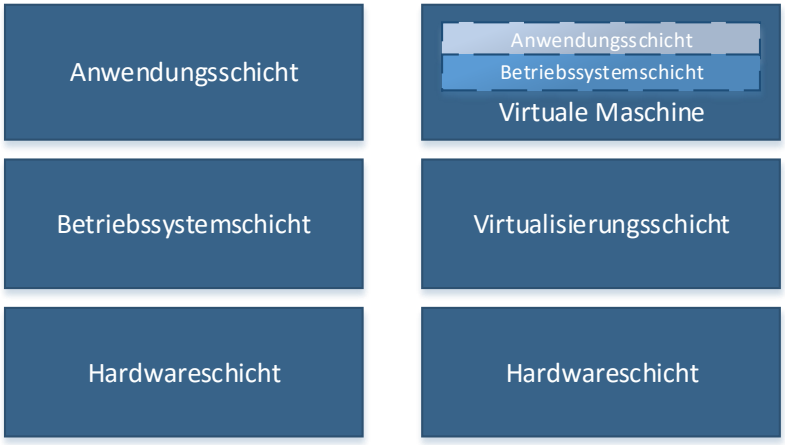


**Abbildung 1: Vergleich Schichtenmodell mit und ohne Virtualisierung**

Zu beachten ist, dass die Virtualisierungsschicht sich je nach Virtualisierungsvariante an unterschiedlichen Ebenen befinden kann. Bei der vorgestellten Darstellung handelt es sich um eine Vollvirtualisierungsvariante wie sie durch einen Typ-1- Hypervisor Virtualisierung nach Robert P. Goldberg beschrieben wird (vgl. [GOLD72]). Die unterschiedlichen Virtualisierungsvarianten werden in den folgenden Unterkapiteln vorgestellt.

Die Virtualisierungsschicht besteht aus einem Hypervisor, welcher die gesamten Zugriffe auf die Hardware verwaltet. Oberhalb des Hypervisor, aber innerhalb der Virtualisierungsschicht befinden sich eine beliebige Anzahl von „Virtual Machine Monitor“ (VMM). Jeder einzelne dieser VMMs hat die Aufgabe, eine Systemumgebung für jeweils genau eine virtuelle

Maschine bereitzustellen und diese zu verwalten. Diese Architektur wird in der Abbildung 2 grafisch dargestellt.

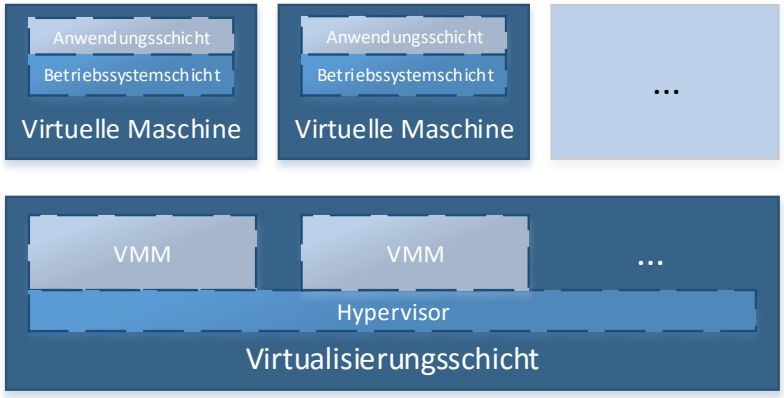


**Abbildung 2: Detaillierte Virtualisierungsschicht mit virtuellen Maschinen**

In Abbildung 3 werden die elementaren Unterschiede der Architektur mit und ohne Nutzung des Virtualisierungsprinzips dargestellt. Dabei verschiebt sich die Betriebssystemschicht und die ehemalige Anwendungsschicht in eine jeweilige virtuelle Maschine, welche parallel und isoliert zu einander mit beliebig vielen anderen VMs ausgeführt werden kann.

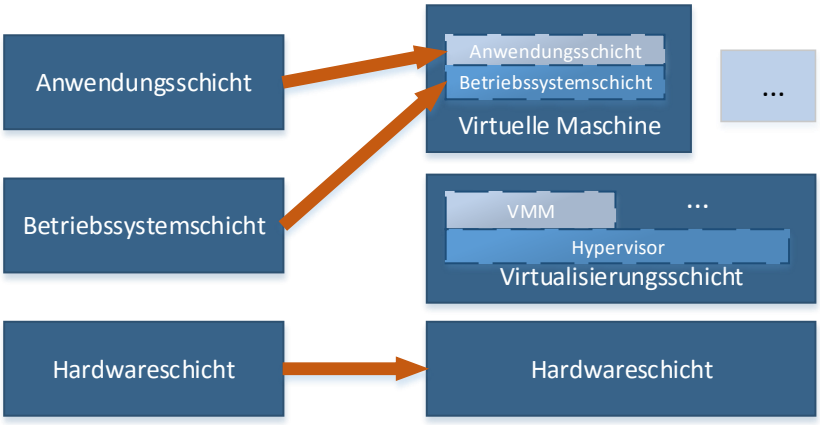


**Abbildung 3: Vergleichsdarstellung der Architekturen**

## 2.4 Unterschied Mobile und Desktop Virtualisierung?

Die Anforderungen an eine Hypervisor-Architektur für mobile Geräte sind anders als die für herkömmliche Desktop Systeme. Heutige Mobile Geräte sind Mehrzwecksysteme und ihre Funktionalität wächst immer weiter. Daher steigt auch die Komplexität ihrer Software (vgl.[HEIS08]). Diese Komplexität führt im Gegenzug zu Sicherheitslücken und dadurch entsteht eine immer größere Nachfrage nach Leistungsstarken Sicherheitslösungen wie die Virtualisierung.
Mobile Geräte werden jedoch von Hardwarebeschränkungen dominiert. Obwohl die Leistung ihrer CPUs und Arbeitsspeicher inzwischen genauso stark ist wie bei Desktop Systemen, sollte die Komplexität der Anwendungen geringgehalten werden, um möglichst wenig Energie zu benötigen. Darüber hinaus unterliegen Mobile Geräte Echtzeitbedingungen, was bedeutet, dass Ressourcenbeschränkungen im vollen Umfang bei Virtualisierung auf mobilen Geräten berücksichtigt werden müssen.

Ein weiterer Unterschied betrifft die in mobilen Geräten verwendete Prozessor-Architektur. Mit einem Marktanteil von über 95% bei Smartphones und über 85% bei Tablets beherrschen ARM-Prozessoren (Advanced RISC Machines) den Markt für mobile Geräte (vgl. [ARM15]). Die x86-Prozessor-Architektur unterstützt vier verschiedene Berechtigungsstufen, bekannt als Ring-Modell (vgl. [ADAM06]). Der Vorteil der unterschiedlichen Privilegierungsstufen ist, dass keine Komponente mit gleichen Rechten auf dieselben Ressourcen zugreifen kann. In einer virtuellen Umgebung läuft der VMM in Ring 0, Betriebssystemkernel in Ring 1 und Anwendungen laufen in Ring 3, während bei der ARM-Architektur sieben Privilegierungsstufen verwendet, wovon sechs Stufen im nicht-privilegiert Modus arbeiten. Hierdurch gibt es ein paar neue Herausforderungen bei der Virtualisierung des Prozessors auf mobilen Geräten.
Ein Vergleich der unterschiedlichen Eigenschaften zwischen der Server Virtualisierung und der Virtualisierung auf mobilen Geräten ist in Tabelle 1 angegeben.

| ***Eigenschaft*** | ***Server Virtualisierung*** | ***Mobile Virtualisierung*** |
|---|---|---|
| Energieversorgung | Hoch | Niedrig |
| Ressourcen Einschränkung | Nein | Ja |
| Echtzeitfähigkeit | Nein | Ja |
| Hardware Unterstützung | Seit 2005 | Seit 2012 |
| Führende Hardware-Befehlssatz-Architektur (ISA) | Intel | ARM |

**Tabelle 1: Unterschiede zwischen Server Virtualisierung und Mobile Virtualisierung**

Durch die Einführung der Hardware-Unterstützung für Virtualisierung bei ARM kam es zu einer Aktualisierung der Virtualisierung für mobile Geräte und ermöglicht seitdem effiziente Echtzeit Virtualisierungslösungen (vgl. [DALL14]).

Die Anforderungen an die Virtualisierung für mobile Geräte sind im Folgenden zusammengefasst:

1. Leichtgewichtige Virtualisierungsschicht mit Unterstützung für mehrere separate virtuelle Umgebungen;
2. Kommunikation zwischen den Systemkomponenten müssen eine geringe Latenz und eine hohe Bandbreite bieten;
3. Systemweite Umsetzung der Sicherheitspolitik;
4. Minimale Auswirkung auf Systemressourcen und der Echtzeitverarbeitung
5. Auf ARM-Architekturen Umsetzbar sein;
6. Hohe Benutzerfreundlichkeit.

In Kongruenz dazu können Anwendungsfälle für Virtualisierung auf mobilen Geräten wie folgt kategorisiert werden:

1. **Mehrere Heterogene Betriebssysteme:** Mobile Virtualisierung ermöglicht es auf einem physikalischen Gerät gleichzeitig mehrere heterogene Betriebssysteme auszuführen. Zum Beispiel bei BYOD-Modelle in Unternehmen kann ein Nutzer ein privates und ein Unternehmenssystem auf seinem Gerät verwenden.
2. **Sicherheit:** Virtualisierung ermöglicht die gleichzeitige Ausführung von sicherheitskritischen Anwendungen und interaktive Anwendungen des Benutzers auf demselben mobilen Gerät durch logische Isolation.
3. **Ressourcen-Management:** Virtualisierung ermöglicht zudem eine flexible Zuweisung der Hardware-Ressourcen zu den einzelnen Gastbetiebssystemen. Dieser Bereich wird nur zur Vollständigkeit hier erwähnt und ist nicht Teil dieser Arbeit.

## 2.5 Charakteristik einer Virtuellen Maschine

Es gibt drei substanzielle Eigenschaften an der Virtualisierung, während das kontrollierende Programm (Virtual Machine Monitor) eines beliebigen, in Ausführung befindenden Programms resident ist: Effektivität, Ressourcenkontrolle und Gleichwertigkeit (vgl. [POPE74]), welche im Folgenden vorgestellt werden:

**Effektivität:**

Der Großteil aller Instruktionen des virtuellen Prozessors muss direkt vom physikalischen Prozessor ausgeführt werden können (vgl. [PICHT09]).

**Ressourcenkontrolle:**
Es muss ausgeschlossen sein, dass ein beliebiges Programm Systemressourcen beeinflussen kann, ohne dass diese Ressource explizit vom verantwortlichen Virtual Machine Monitor verfügbar gemacht wurde.

**Gleichwertigkeit:**
Das Laufzeitverhalten eines beliebigen Programms in einer virtuellen Maschine sollte identisch sein mit dem Laufzeitverhalten des gleichen Programms ohne Verwendung von Virtualisierung.

Heutige Virtualisierungslösungen erfüllen die Eigenschaften der Ressourcenkontrolle und der Gleichwertigkeit. Die Eigenschaft der Effizienz ist jedoch nicht voll umsetzbar, wird aber durch die Verwendung spezieller Technologien bei der Implementierung von aktuellen Virtualisierungslösungen umgesetzt (vgl. [BALM14]).

## 2.6 Varianten der Systemvirtualisierung

In diesem Abschnitt werden die unterschiedlichen Ansätze zur Virtualisierung auf mobilen Geräten vorgestellt. Neben den verschiedenen Ansätzen der Systemvirtualisierung werden die unterschiedlichen Ansätze der Anwendungsvirtualisierung betrachtet. Der Unterschied zwischen den beiden erwähnten Gruppen besteht in der zu virtualisierenden Umgebung. Bei Systemvirtualisierung handelt es sich um das Erzeugen virtueller Maschinen durch Virtualisierung der gesamten Hardware-Ressourcen, inklusive Prozessor, Arbeitsspeicher, Festplattenspeicher und Peripheriegeräten (vgl. [HWAN08]).
Im Gegensatz dazu wird bei der Anwendungsvirtualisierung eine virtuelle Umgebung für die jeweilige Anwendung erzeugt, welche eine separate Kopie der originalen Betriebssystemumgebung darstellt.

Bei der Systemvirtualisierung werden die Ansätze des Hypervisor, des Microkernels, sowie des Microvisor, welcher eine Kombination beider Ansätze darstellt, vorgestellt. Als Ansätze der Anwendungsvirtualisierung wird neben der Container-Virtualisierung das Anwendungsstreaming und der Remote-Desktop-Dienst vorgestellt.

### 2.6.1 Hypervisor

Virtualisierung mit Hilfe eines Hypervisor ermöglicht es mehrere Instanzen eines Betriebssystems auf einer gemeinsamen Hardware auszuführen. Dabei gibt es zwei Typen von Hypervisor:

**Typ 1 – Hypervisor (native oder bare metal):**
Dieser Typ des Hypervisor wird direkt auf der Hardware ausgeführt und hat direkten Zugriff auf die Hardware-Ressourcen. Hypervisor vom Typ 1 können mehrere Betriebssysteme parallel betreiben. Durch den direkten Hardwarezugriff kann die Leistung jedes einzelnen Betriebssystems bzw. die umgebenden virtuellen Maschinen individuell optimiert werden. Außerdem sind alle virtuellen Maschinen komplett voneinander isoliert, wodurch die Sicherheit untereinander erhöht wird.

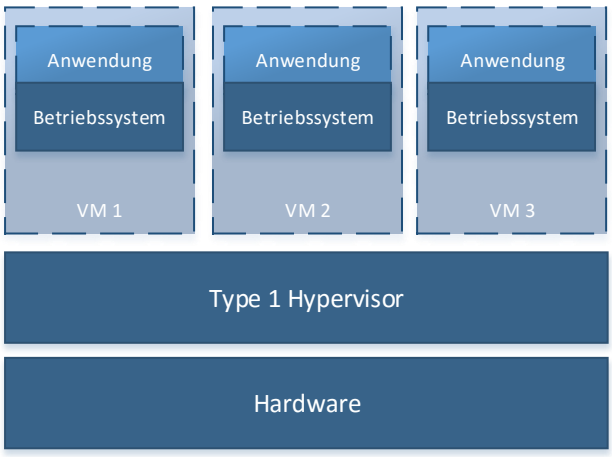


**Abbildung 4: Schichtenmodell des Hypervisor Typ 1**

**Typ 2 – Hypervisor (hosted):**
Der Hypervisor Typ zwei befindet sich oberhalb des Betriebssystems und wird dort in dessen Anwendungsschicht ausgeführt. Dabei erfolgt der Zugriff auf die Hardware nur über das zugrundeliegende Betriebssystem. Hierdurch ist die Leistung der einzelnen virtuellen Maschinen schlechter als bei denen der Typ 1-Virtualisierung. Unter einander sind die einzelnen virtuellen Maschinen isoliert, allerdings kann eine Kompromittierung oder Fehler des zugrundeliegenden Betriebssystems Auswirkung auf jede einzelne laufende virtuelle Maschine haben.

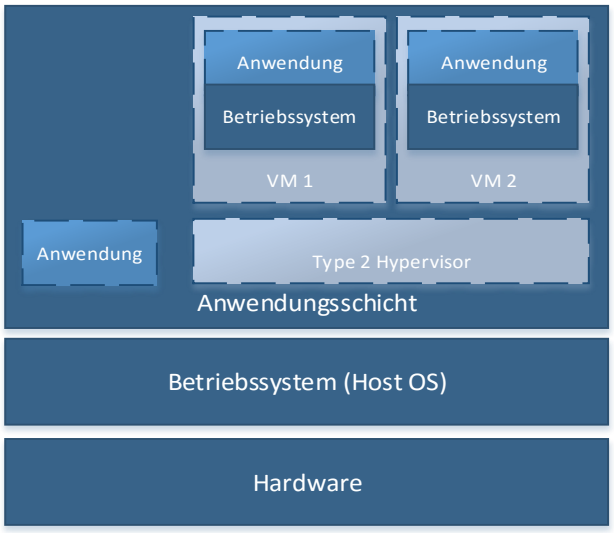


**Abbildung 5: Schichtenmodell des Hypervisor Typ 2**

Die Systemvirtualisierung des Hypervisor Ansatz kann weiter in Voll- und Paravirtualisierung Klassifiziert werden.

**Vollvirtualisierung:**
Bei der Vollvirtualisierung, welche auch als Komplettvirtualisierung bezeichnet wird, präsentiert der Hypervisor dem Gastbetriebssystem eine exakte Kopie von der Hardware (vgl. [VAHI13]). Das Gastbetriebssystem ist hierbei unmodifiziert und es erkennt nicht, dass es in einer virtuellen Maschine ausgeführt wird.
Die Vollvirtualisierung kann mit Hardware-Unterstützung oder mit einer dynamischen Binärübersetzung (engl. Dynamic Binary Translation, DBT) umgesetzt werden. Bei der dynamischen Binärübersetzung wird zur Laufzeit der Binärcode des Gast-Betriebssystems in dynamische Basisblöcke zerlegt, welche analysiert und modifiziert werden. Ein solcher dynamische Basisblock besteht stets aus der Menge der Anweisungen zwischen zwei Verzweigungen oder Sprüngen (vgl. [BREG15]).

**Paravirtualisierung:**
Bei der Paravirtualisierung wird der Kernel des Gastbetriebssystems modifiziert, damit dieser direkt mit einer bereitgestellten Schnittstelle kommunizieren kann, anstatt auf die physikalische Hardware zuzugreifen. Dabei werden sensible nicht-privilegierte Anweisungen im Gastbetriebssystem durch explizite Hypervisor Aufrufe, sogenannte „Hypercalls“ ersetzt. Durch die vom Hypervisor bereitgestellten Binärschnittstellen (engl. Application Binary Interface, ABI) wird den Gastbetriebssystemen die Kommunikation von kritischen Anweisungen ermöglicht. Hierdurch übertreffen paravirtualisierte Systeme in der Regel

vollständig virtualisierte Systeme, welche auf der gleichen Hardware betrieben werden (vgl. [VARA10]). In Tabelle 2 werden die beiden Virtualisierungstechniken des Typ 1 Hypervisor verglichen (vgl.[SHUJ16]).

| ***Virtualsierungs-techniken*** | ***Kategorie*** | ***Prinzip*** | ***Vorteile*** | ***Nachteile*** |
|---|---|---|---|---|
| *Voll-virtualisierung* | Hardware-Unterstützt | Nutzt Hypervisor Modus und Hardwareunterstützung | Geringe Komplexität des Hypervisor; Unmodifizierte Gastbetriebs-systeme | Sensitive nicht-privilegierte Anweisungen in ISA; Häufige Traps führen zu zeitintensiven Kontextwechsel |
| | Dynamische Binärüber-setzung | Dynamische Übersetzung der sensitive nicht-privilegiert Anweisungen | Anwendbar auf nicht virtualisierbaren Plattformen (ohne Hardware-Unterstützung); Keine Modifizierung des Betriebssystems erforderlich | Hohe Hypervisor Komplexität durch Analyse aller Anweisungen |
| *Para-virtualisierung* | | Austausch der sensitiven nicht-privilegierten Anweisung im Kernes des Gastsystems durch Hypercalls | Hohe Skalierbarkeit auf mehrere Hardware und Betriebssystemen Plattformen (teilweise auch während der Laufzeit) | Hardware muss unterstützt werden; Modifizierung des Kernels des Gastbetriebssystems notwendig |

**Tabelle 2: Vergleich der Hypervisor Typ 1 Virtualisierungstechniken**

### 2.6.2 Microkernel

Einer der Hauptansätze zur Lösung von Sicherheitsproblemen, ist die Verwendung eines möglichst minimalen Kernels, sowie minimale Komponenten und Interfaces, welche als Microkernel-Ansatz bezeichnet wird (vgl. [IQBA09]).
Dieser Ansatz nutzt das „Prinzip minimaler Rechte" (engl. Principle of least privilege), zu Deutsch das Prinzip der minimalen Privilegien, was bedeutet das jedes Modul (sei es ein Prozess, ein Benutzer oder weiteres) nur Zugriff auf Information und Ressourcen bekommen soll, welche notwendig für das Modul sind. Das zweite Prinzip, welches vom Microkernel-Ansatz befolgt wird, ist das „Prinzip einfacher Sicherheitsmechanismen" (engl. Principle of economy of mechanism). Dieser Ansatz bedeutet, dass das System und seine Sicherheitsmechanismen zwar funktionieren sollen, aber ihre Konstruktion so einfach wie möglich sein soll, um Lücken und Fehlerquellen durch zu hohe Komplexität zu vermeiden (vgl. [BERN06]).
Die oben genannten Prinzipien werden in dem Microkernel angewendet, damit dieser so minimalistisch wie möglich gehalten wird, aber trotzdem die Bereitstellung grundlegender Mechanismen, wie Speicherverwaltung, Thread-Verwaltung, Interprozess-Kommunikation et cetera bereitgestellt werden.
Andere wichtige Software, wie Gerätetreiber, Protokoll Stacks, Dateisysteme und Benutzerschnittstellen-Programmcode usw. sind im Benutzerbereich, ohne direkten Zugriff auf die zur Verfügung gestellte Hardware, untergebracht.
Solch ein minimaler Kernel mit den richtigen Sicherheitsmechanismen ermöglicht es, eine System-Basis bereitzustellen, welche im Betrieb ein hohes Maß an Vertrauen garantieren kann (vgl. [IQBA09]).

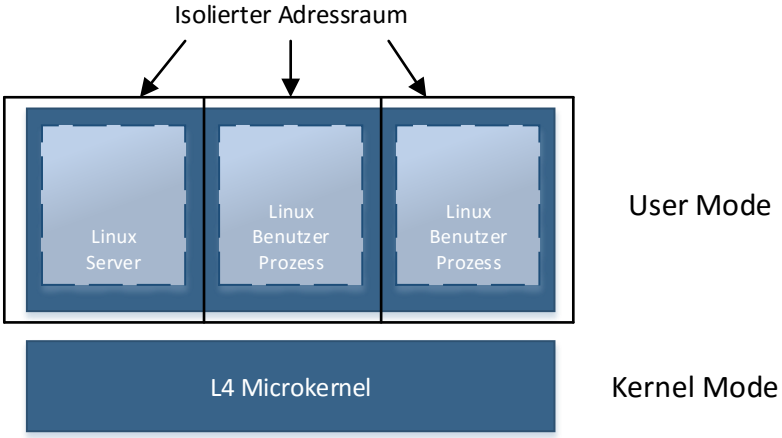


**Abbildung 6: Microkernel Schichtenmodell an Beispiel der L4Linux Architektur**

Die Nutzung eines minimalistischen Kernels impliziert nicht zwingend ein sicheres System, allerdings kann die Sicherheit durch Isolierung der einzelnen Systemdienste in einzelne Zellen erhöht werden. Bis auf regulierte Nachrichtenaustausch per IPC wird jegliche Kommunikation unter den einzelnen Zellen durch die Isolierung unterbunden. Dadurch kann sicherheitskritische Software in einer Zelle so platziert werden, dass der Zugriff auf den Adressraum dieser Zelle von „Außen“ verhindert wird und so zu einem sicheren System führt. Um ein anderes Betriebssystem auf einem Microkernel zu platzieren, sind am Kernel des Gastbetriebssystems viele Modifizierungen zur Deprivilegierung vorzunehmen. Diese Veränderungen umfassen die Modifikation von Systemaufrufen und –Schnittstellen, sowie der Speicherverwaltung und des Interrupt-Handling. Beispiele in der Praxis, welche diesen Virtualisierungsansatz umsetzten, sind Xen und die auf dem L4-Microkernel basierende Virtualisierung L4Linux (vgl. [BRAK08]).
Als Architektur verwendet L4Linux eine Client-Server-Architektur. Wie oben in Abbildung 6 zu sehen ist, werden die einzelnen Dienste in jeweils voneinander isoliert einzelnen Zellen verpackt. Benutzeranwendungen und Prozesse können nur mit den Serverdiensten per IPC kommunizieren. Die Konsequenz daraus ist, dass die Serverdienste komplett von den nicht vertrauenswürdigen Benutzerprozessen isoliert sind. Dadurch führt eine kompromittierte Anwendung oder VM zu keinen negativen Auswirkungen am restlichen, geschützten System. Dies ist ein wichtiger Aspekt der Isolierung (vgl. [IQBA09]).

### 2.6.3 Microvisor

In dem Beitrag „The OKL4 microvisor: convergence point of microkernels and hypervisors” von Gernot Heiser und Ben Lesie aus dem Jahr 2010 wird ein Ansatz beschrieben, welcher die Merkmale des Microkernel- und des Hypervisor-Ansatzes miteinander verbindet. Eine der wichtigsten Eigenschaften dieses Ansatzes ist, dass er auf vielen verschiedenen Prozessorarchitekturen, einschließlich der ARM-Prozessoren, eingesetzt werden kann. Dies ermöglicht den Einsatz auf Mobile Geräten. Bevor dieser Ansatz genauer vorgestellt wird, werden erst die Unterschiede der beiden zugrundeliegenden Ansätze beschrieben (vgl. [HEIS10]).
Microkernel Virtualisierung werden sowohl native, also Typ 1 als auch als „hosted”, Typ 2 angewendet. Allerdings hängt der Änderungsaufwand im Gastbetriebssystem davon ab, wie eng die Abstraktion der Hardware und der Virtualisierungssoftware ist. Eine hohe Ebene der Abstraktion erfordert einen höheren Modifizierungsgrad am Kernel des Gastbetriebssystems.
In diesem Sinne haben Microkernel- und Hypervisor-Ansätze ganz unterschiedliche Auswirkungen auf die Höhe und des Schwierigkeitsgrades der Modifikationen, um Gastbetriebssysteme in virtuelle Maschinen zu betreiben.

Hypervisor kennen typischerweise nur die virtuelle Maschine und das darin laufende Gast-Betriebssystem und ignorieren alle internen Details der laufenden virtuellen Maschinen. Zudem kennt der Hypervisor nur die Eigenschaften der virtuellen Maschinen, welche von ihm selbst verwaltet werden (Speicher, Geräte, Prozessorzuteilung etc.). Diese Sicht ähnelt sehr der Sicht eines Betriebssystems, welches direkt auf der Hardware betrieben wird.
Auf der anderen Seite, bieten Microkernel durch ihre oben beschriebenen Mechanismen eine höhere Ebene der Abstraktion. Zudem führen diese Mechanismen, welche den Mechanismen ähneln, die in Betriebssystemen implementiert sind, zu einem komplexeren Mapping führen. Tabelle 3 beschreibt die Unterschiede zwischen virtualisierte Betriebssysteme auf einem Microkernel und zu einem Hypervisor (vgl. [AMAN09]).

| ***Virtualisierungsvariante*** / ***Dienst*** | *Microkernel* | *Hypervisor* |
|---|---|---|
| *Threads* | Kennt alle Threads von jedem Betriebssystem und verwaltet global die CPU-Zuteilung | Kennt nur die Gast-Betriebssysteme und verwaltet deren CPU-Zuteilung. |
| *Prozesse* | Kennt einzelne Tasks und Prozesse des Gastbetriebssystems | Kennt nur die Gast-Betriebssystem-Partition |
| *Arbeitsspeicher* | Kennt Speicherkontext von Tasks und ist involviert in Kontextwechsel, Seitenfehlern etc. | Kennt nur die Speicherpartition des gesamten Gastbetriebs-systems. Inhalt dieser Partition wird vom jeweiligen Betriebssystem verwaltet. |
| *Kommunikation* | IPC imponiert API und Semantik | Per IPC Aufruf aller Systemdienste je nach Grad der Isolierung möglich. Kann zu hohen Latenzzeiten führen. |
| *Interrupts und Exceptions* | Abfangen und Weiterleiten per IPC | Virtuelle Interrupts werden zum Gastbetriebssystem weitergeleitet. Exceptions werden vom Gast-Betriebssystem verwaltet |
| *Peripherie* | Geräte Treiber als Microkernel-Prozesse | Kerntreiber im Hypervisor und Gerätetreiber im Gast-Betriebssystem |

**Tabelle 3: Unterschied Microkernel- zu Hypervisor-Ansatz**

Ziel des Microvisor ist es, als einzelner Kernel eine Plattform-Virtualisierung zu ermöglichen und dabei die Vorteile beider Ansätze vorweisen zu können. Die Effizienz eines Hypervisor kombiniert mit den Sicherheitsmechanismen und der Minimalität des Microkernel.

Eine der ersten Umsetzungen dieses Ansatzes ist der OKL4 von Open Kernel Labs. Inspiriert durch den SEL4 (secure embedded L4), dessen Spezifikationsimplementierung formal verifiziert ist, was bedeutet das der Kernel unter anderem nachweislich keine bekannten Entwurfsfehler wie z.B. Speicherüberläufe, Zeigerfehler und Speicherlecks enthält (vgl. [PICH09]). Im Gegensatz zu Xen, hat der OKL4 ein viel kleinere TCB (Trusted Computing Base) und nur ein Zehntel der Anzahl der LOC (Lines of code) des Xen-Hypervisor.

### 2.6.4 Zusammenfassung der Systemvirtualisierungsvarianten

Andere Techniken als die vollständige Virtualisierung sind mit hohen Kosten im Bezug des Software-Engineering-Prozesses verbunden (vgl. [HEIS11]). Die Vollvirtualisierung mit binärer Übersetzung erhöht die Komplexität durch Analyse und Modifizierung sämtlicher Anweisungen. DBT-basierte Vollvirtualisierung ermöglicht die Virtualisierung von unmodifizierten Gastbetriebssystemen. Allerdings entsteht ein signifikanter Leistungsabfall durch das „Trap-and-emulate-Verfahren“ mit DBT.
Im Gegenteil vermeiden paravirtualisierte Lösungen den Leistungsabfall durch geringe Nutzung des „Trap-and-emulate-Verfahren“, allerdings führt der Paravirtualisierungsansatzes zu einer hohen Komplexität bei der Gestaltung des Hypervisor und der Gastbetriebssysteme. Bei der paravirtualisierte Lösung müssen die Gastbetiebssysteme modifiziert werden und die Hardware durch den Hypervisor unterstützt werden.
Paravirtualisierung und DBT-basierte Vollvirtualisierung ermöglichen die Virtualisierung von Systemen, welche nicht klassisch zu virtualisieren sind.

## 2.7 Varianten der Anwendungsvirtualisierung

Im Gegensatz zur Systemvirtualisierung, bei welcher die vollständige Hardware virtualisiert bereitgestellt wird, wird bei der Anwendungsvirtualisierung nur die direkte Ausführungsumgebung der Anwendung virtualisiert. In der Literatur sind drei Varianten der Container-Virtualisierung, das Application-Streaming sowie die Remote-Desktop-Dienste verbreitet (vgl.[XU09]).

### 2.7.1 Container-Virtualisierung

Bei der Container-Virtualisierung wird eine logische Trennung der Anwendung vom unterliegenden Betriebssystem durchgeführt. Bei diesem Ansatz wird die Virtualisierungsschicht als eine Anwendung innerhalb des Betriebssystems platziert und wird

deshalb auch als Betriebssystemvirtualisierung (engl. OS-level virtualization) bezeichnet. Diese Variante wird in Abbildung 7 dargestellt.


**Abbildung 7: Schichtenmodell Container-Virtualisierung**

Die Container-Technologie ermöglicht es mehrere isolierte Instanzen von Benutzerumgebungen auf einem einzigen Betriebssystem auszuführen. Auf Grund ihrer Ausführung in der Anwendungsschicht, sind die jeweiligen Container-Anwendungen in der Regel an die jeweiligen Betriebssystem Typen (z.B. Linux oder Microsoft Windows), für welche sie erzeugt wurden, gebunden. Falls es einem Angreifer gelingt aus dem laufenden Prozess „auszubrechen", findet er sich in der virtualisierten und isolierten Benutzerumgebung wieder, von welcher er aus keinen Schaden an dem zugrundeliegenden Betriebssystem ausüben kann.
Neuere Container-Technologien wie „Linux Containers" (kurz LXC) stellen mehrere Mechanismen wie Ressourcen-Management, Prozess-Management und Dateisystem-Isolierung zur Systemkonsolidierung zur Verfügung. Dabei enthält jeder Container eine reduzierte Ansicht desselben Kernels, wie der des zugrundeliegenden Betriebssystems.
Als weiterer Vorteil der Container-Virtualisierung zählt, dass das Betriebssystem bei diesem Ansatz direkt auf die Hardware zugreifen kann und die zusätzliche Abstraktionsschicht nur in der Anwendungsschicht hinzugefügt wird. Dadurch befindet sich das Betriebssystem weiterhin im privilegierten Kernel Mode, wodurch zeitaufwendige Kontextwechsel reduziert werden (vgl. [MACA13]).

### 2.7.2 Anwendungsstreaming

Bei der Streaming-Technologie werden Teile oder der komplette Programmcode und die zugehörigen Daten erst zur Laufzeit der Anwendung über eine Netzwerkverbindung auf das Gerät übertragen und existiert dort während der Nutzung im Arbeitsspeicher. Getätigte Änderungen werden im Anschluss wieder über die Netzwerkverbindung an einen zentralen Server übertragen. Dieser Ablauf wird in Abbildung 8 dargestellt.

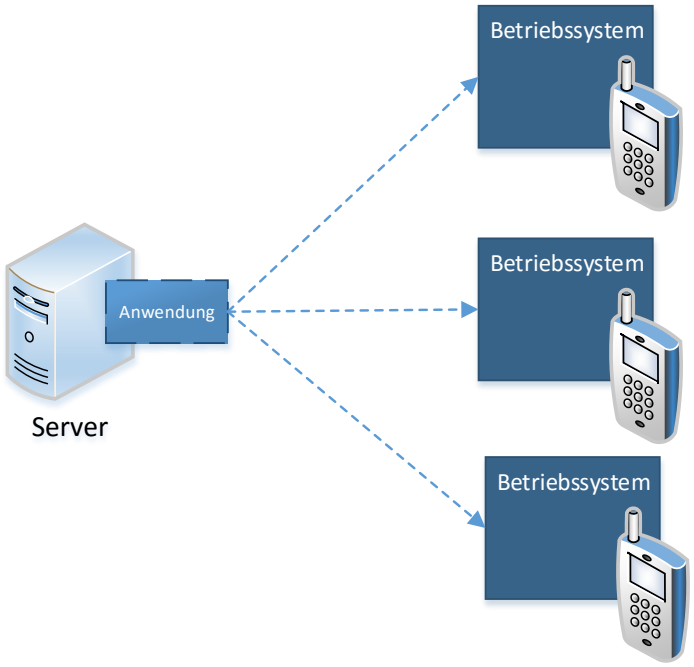


**Abbildung 8: Anwendungsstreaming**

Zur Laufzeit befindet sich die Anwendung vollständig im Speicher des Gerätes, wodurch zu diesem Zeitpunkt keine logische Trennung mehr vorliegt und sich die Virtualisierung, also die logische Trennung zweier Schichten sich nur auf die Zeit beschränkt, in der die Anwendung nicht ausgeführt wird. Deshalb ist die Streaming-Technologie keine Virtualisierungslösung, sondern nur ein Mechanismus, um zum Beispiel die Aktualität des Programmcodes und der Daten sicher zu stellen.

### 2.7.3 Anwendungsvirtualisierung per Remote-Desktop-Dienste

Mit Remote-Desktop-Diensten, auch als Terminal-Dienste bekannt, wird der Zugriff auf Anwendungen auf einen zentralen Terminal-Server ermöglicht. Die Anwendungen und zugehörigen Daten befinden sich dabei ausschließlich auf dem Terminal Server und werden

so von den Endgeräten isoliert. Lediglich die Bildschirmausgabe sowie Eingabebefehle werden zwischen dem Endgerät und dem Server ausgetauscht.

Wie im Abbildung 9 zu sehen ist, besteht die Virtualisierung der Anwendung im Gegensatz zum Streaming ganzzeitlich. Allerding eignet sich die Lösung nicht für den Einsatz auf mobilen Geräten in Rahmen eines BYOD-Modells, da diese Technologie eine Leistungsstarke Netzwerkverbindung sowie kostenintensive Terminalserver benötigt und nicht für eine parallele Nutzung durch eine hohe Anzahl an Benutzer ausgelegt ist (vgl. [TRIT06]).

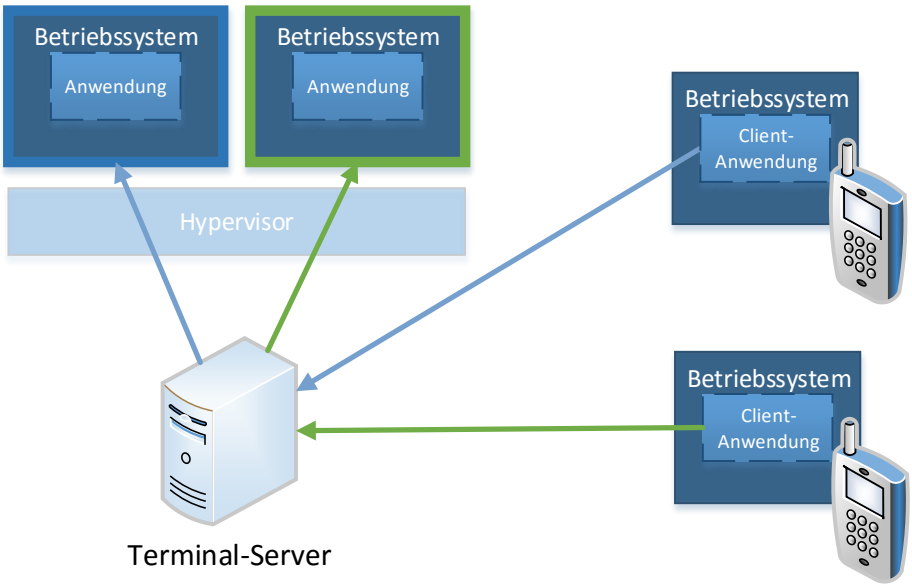


**Abbildung 9: Anwendungsvirtualisierung per Terminal-Dienste**

### 2.7.4 Zusammenfassung der Anwendungsvirtualisierungsvarianten

Die Container-Virtualisierung erzeugt um die einzelnen Anwendungen einen Container herum, damit sie von ihrer Umgebung zu isolieren werden. Außerdem wird damit eine separate Ausführungsumgebung für jede Container-Anwendung geschaffen. Anders als bei der Systemvirtualisierung handelt es sich hierbei um eine sehr leichtgewichtige Virtualisierungsvariante und benötig daher kaum zusätzliche Ressourcen.
Das Anwendungsstreaming trennt Teile des Programmcodes bzw. der Programmdaten nur während der ausführungsfreien Zeit von der eigentlichen Anwendung. Zur Laufzeit werden die benötigten Teile über die Netzwerkverbindung von einem entfernten Server „nachgeladen“ und nutzten zu diesem Zeitpunkt keine Vorteile der Virtualisierungstechnologie.
Bei der Anwendungsvirtualisierung via Remote-Desktop-Diensten wird die jeweilige Anwendung komplett vom physikalischen Gerät getrennt und auf einem entfernten Server ausgeführt. Lediglich die Ein- und Ausgabe der Anwendung erfolgt auf den Client-Geräten. Diese Technologie hat jedoch hohen Ansprüche an die Server- und Netzwerkinfrastruktur.

# 3 Marktanalyse

Auf dem Markt existieren mehrere kommerzielle und Open-Source Virtualisierungslösungen für mobile Geräte. Das Problem bei kommerziellen Lösungen liegt oftmals in ihrer begrenzten Dokumentation, was einen parametrischen Vergleich mit Open-Source Lösungen erschwert. Im folgenden Kapitel wird zuerst der Markt der mobilen Virtualisierungslösungen vorgestellt. Die Produktauswahl wurde auf Grund der Aktualität der Lösung, Umfang und Transparenz der Dokumentation, sowie der Popularität in anderen wissenschaftlichen Arbeiten und Fachzeitschriften getroffen.
Im Anschluss der Marktübersicht wird ein Vergleich auf Basis bekannter Online-Datenbanken für Exploits aufgestellt. Dazu werden die Anzahl der bekannten Sicherheitslücken der mobilen Betriebssysteme Android und iOS mit den Ergebnissen aus eine Auswahl an verschiedenen Suchwörtern der Virtualisierung vergleichen.

## 3.1 Lösungen der Systemvirtualisierung

Mit Virtualisierungsvarianten auf Basis eines Hypervisor oder per Microkernel Ansatz ist es möglich, zwei oder mehr Instanzen eines Betriebssystems parallel auf einem mobilen Gerät auszuführen. Diese Fähigkeit machen eine komplette logische Trennung der privaten und betrieblichen Anwendungsumgebung möglich.

**KVM on ARM (Hypervisor - Typ 1):**
Kernel based Virtual Machine (KVM) ist eine Virtualisierungslösung für den Linux Kernel. Sie unterstützt native Virtualisierung für Prozessoren mit Virtualisierungsunterstützung. Die KVM Technologie ist seit der Version 2.6.20 fest im Linux Kernel integriert.
KVM/ARM ist einer der ersten Hypervisor Lösungen welche die Hardware Virtualisierungsunterstützung der ARM-Prozessoren verwendet, um mehrere Systeme parallel mit nahe nativer Leistungsfähigkeit zu ermöglichen (vgl. [HWAN08]). Es wurde ursprünglich von dem Software Systems Lab der Columbia University entwickelt und nun in Zusammenarbeit mit Virtual Open Systems weitergeführt.
Das primäre Unterscheidungsmerkmal zwischen KVM/ARM und anderen Virtualisierungstechniken ist der einfach gehaltene Implementierungsansatz von KVM/ARM. Die meisten Hypervisor-Lösungen implementieren weitgehend alle wichtigen Dienste, wie

z.B. den Scheduler, Speichermanager und Timer. Dies führt zu einem relativ großen und komplizierten Programmcode. KVM/ARM ist eine Paravirtualisierungslösung und nutzt daher die Funktionalität im Linux Kernel und ist somit vergleichsweise kleiner und viel weniger komplex.
Der Hypervisor ist als Kernel-Modul implementiert und greift somit auf die Hardware über die Linux-Kernel-Schnittstellen zu. Das Gastbetriebssystem läuft als Prozess auf dem Host-Kernel. Verschiedene Gastbetriebssystem-Instanzen werden als separate Prozesse auf dem Host-Kernel betrachtet.

Obwohl KVM ein Modul des Linux Kernel ist und daher die meisten Kernel-Funktionalitäten nutzt, besitzt KVM eine schlechte Kompatibilität zum Android Betriebssystem, welches auch auf einen Linux Kernel entworfen ist (vgl. [XU15]). Zusätzlich müssen die Gastbetriebssysteme für die Nutzung der Funktion des gemeinsamen Kernels modifiziert werden.

**VMware Mobile Virtualization Platform (Hypervisor - Typ 2):**
Die mobile Virtualisierungsplattform von VMware (MVP) ist eine Typ 2 - Hypervisor Virtualisierungslösung, welche für die Anwendung im Rahmen eines BYOD-Modells entworfen wurde. MVP ermöglicht die Nutzung eines Gastsystems innerhalb des laufendes Systems. Zur Verwaltung des MVP wird ein Dienst („mvpd") mit „Super-User"-Rechten auf dem Host-Betriebssystem ausgeführt. Hinzu kommt ein spezielles Kernel-Modul („mvpkm"), welches die Verwaltung der Authentifizierung und Ausführung des Gastsystems, sowie das Wechseln zwischen dem nativen und virtuellen Betriebssystem übernimmt.
Der Zugriff auf die Hardware geschieht durch paravirtualisierte Treiber. Systemaufrufe und sensitive Instruktionen des Gastbetriebssystems werden vom MVP-Hypervisor abgefangen und durch sogenannte Hypercalls ersetzt oder emuliert. Als Gastbetiebssysteme können nur angepasste Android-Kernel verwendet werden (vgl. [BARR10]).
Wie schon im Grundlagenkapitel erläutert, fordert die Paravirtualisierung die Modifizierung des Kernels des Gastbetriebssystems. Hierdurch muss der Quellcode des Kernels zur Modifikation dem Hersteller einer Paravirtualisierungslösung bereitgestellt werden. Da dies bei kommerziellen Betriebssystemen wie Windows oder iOS nicht möglich ist, kann Paravirtualisierung als Lösung für Heterogene Systeme wie in einem BYOD-Modell nicht genutzt werden.

**Cells & Cellrox (Microkernel):**
Cells ist eines der ersten Open-Source Projekte im Bereich der Virtualisierung auf mobilen Geräten. Dabei können auf einem physikalischen Gerät mehrere parallel laufende virtuelle Maschinen mit einem gemeinsamen modifizierten Linux-Kernel betrieben werden. Als Betriebssystem kommt ein modifiziertes Android zum Einsatz.
Cells nutzt LXC (Linux Container) um Prozesse voneinander zu isolieren, dabei werden Kernel-Ressourcen virtualisiert und gegeneinander abgeschottet. Durch einen besonderen Namensraum Mechanismus wird ein effektives Multiplex Zugriffsverfahren auf die Hardware-Treiber des Kernel und in der Benutzerumgebung ermöglicht. Hierdurch ist es für Cells möglich jede laufende virtuelle Maschine eine Leistungsfähigkeit wie auf nativer Hardware anzubieten. Dazu gehört vollständige 3D-Grafik-Beschleunigung zu ermöglichen, welche vom Anwender von seinem mobilen Gerät erwartet wird. Der Overhead, also der Leistungsunterschied zur nativen Ausführung, betrug in Laufzeittests weniger als 2% (vgl. [ANDR11]).

Während Cells als Open Source Projekt durch die Columbia Universität geführt wird, gibt es auch eine kommerzielle Version unter dem Namen Cellrox Thinvisor mit gleichen Eigenschaften wie das Open Source Projekt. Zusätzlich wurde Cellrox jedoch um weitere Mobile Device Management (MDM) Funktionen erweitert. In Abbildung 10 wird der Namensraum-Mechanismus des Cellrox Thinvisor dargestellt (vgl. [CELL15]). Anders als bei Container Lösungen, welche sich meist einen gemeinsamen Namensraum teilen, wird bei dem Ansatz von Cells bzw. Cellrox Thinvisor für jede einzelne virtuelle Maschine ein eigner Namensraum erzeugt. Wenn eine virtuelle Maschine kompromittiert wird, kann es zu keinem Sicherheitsproblem für die anderen laufenden virtuellen Maschinen führen.


**Abbildung 10: Cellrox Thinvisor Namespace Architektur**

Eines der besonderen Merkmale von Cellrox ist die Möglichkeit, Verknüpfungen zu Anwendungen auf den anderen laufenden virtuellen Maschinen zu erstellen. Dadurch muss der Anwender nicht umständlich zwischen den virtuellen Maschinen wechseln.

Eine vielversprechende Lösung mit nahezu nativer Leistungsfähigkeit. Durch die Quellenoffenheit des Cells-Projekt zudem gut dokumentiert. Ist jedoch bisher nur als Android Betriebssystem verfügbar und fordert den Austausch des gesamten Betriebssystems durch ein von der Universität Columbia bzw. Cellrox modifiziertes Android-basierendes Betriebssystem.

**OKL4 Microvisor (Microkernel):**
Beim OKL4 handelt es sich um einen auf dem L4 basierenden Microkernel Ansatz. Hierbei wird der Kernelgröße durch Auslagerung einiger grundlegenden Dienste, aus dem Kernel als Serverdienst in der Benutzerumgebung reduziert. Diese Dienste kommunizieren über IPC mit dem Microkernel. Das grundlegende Ziel des Microkernel Ansatzes ist es die Trusted Computing Base soweit zu reduzieren, um ihre Sicherheit formal verifizieren zu können. Ein weiteres Ziel ist es einen generellen Kernel-Raum zu erzeugen, um die Kernel Policy von seinen Mechanismen zu trennen (vgl. [HEIS10]). Die Policy geht hierbei auf die Frage ein „wie die Funktionalität verwendet werden kann", während es beim Mechanismus darum geht „welche Funktionalität zur Verfügung gestellt werden muss". Durch eine Trennung dieser beiden Bereiche, kann das System deutlich einfacher entwickelt und an verschiedene Bedürfnisse angepasst werden (vgl. [RUBI02]).

Das Prinzip der Separation hilft dabei die Hardware für den privilegierten Microkernel zu virtualisieren, welcher dann als Hypervisor auftritt und mit den Gastbetriebssystemen per IPC kommuniziert.
Der OKL4 Microkernel stellt den Linux-Gastbetriebssystemen eine paravirtualisierte HAL (Hardware Abstraction Layer) bereit, was den Gästen ermöglicht, Informationen über die verfügbare Hardware abzurufen und mit dieser zu kommunizieren.
System Calls und Interrupts werden vom Microkernel abgefangen und in IPC-Nachrichten umgewandelt an die Gastbetriebssysteme weitergeleitet. Gerätetreiber sind als separate Benutzerumgebungsprozesse implementiert, mit welchen ebenso ausschließlich per IPC kommuniziert werden kann. Um die Sicherheit des Systems zu gewährleisten, wurde im OKL4 ein Policy-Modul implementiert, welche die Kommunikation der Gastsysteme mit den einzelnen separaten Treibern regelt. Die Effizienz der Microkernel Virtualisierungslösung wird durch die Leistung des IPC-Mechanismus definiert, welcher sich im Kern der Microkernel-Architektur befindet (vgl. [HEIS10]).

Der OKL4-Microkernel bietet neben einer hohen Leistung einen formal verifizierten Kernel und Hypervisor und sollte sich hierdurch für den Einsatz auf unternehmenskritischen Geräten eignen. Allerdings müssen auch hier wiederum die von den Herstellern bereitgestellten Betriebssysteme durch das OKL4-Betriebssystem ersetzt werden.

## 3.2 Lösungen der Anwendungsvirtualisierung

Mit Hilfe der Container-Virtualisierung wird die Ausführung betrieblicher Anwendungen in isolierten Umgebungen ermöglicht. Die Isolierung reguliert jegliche Kommunikation der Container-Anwendungen.

**BizzTrust for Android (Container):**
BizzTrust for Android ist eine Container-Lösung, welche vom Fraunhofer SIT und der Sirrix AG zusammen entwickelt wurde. Ziel dieser Lösung ist es die Sicherheit von Unternehmensdaten und der zugehörigen Anwendungen zu ermöglichen, ohne dabei die Funktionalität oder die Nutzer einzuschränken. BizzTrust ermöglicht eine sichere Teilung der Anwendungsbereiche in Unternehmens- und Privatumgebung. Anwendungen aus beiden Umgebungen können so parallel und isoliert voneinander ausgeführt werden. Jeder Bereich verfügt dabei über seine eigenen Zugriffsberechtigungen. Zudem erleichtert diese Lösung die Integration der mobilen Geräte in das unternehmenseigene Netzwerk. Mit Hilfe von VPN kann auf das Intranet des Unternehmens zugegriffen werden. Weiterhin kann der geschäftlich genutzte Bereich auf den mobilen Geräten ferngewartet werden und so u.a. der Gerätezustand festgestellt werden oder automatische Software-Updates eingespielt werden. Die Kernkomponente der

Architektur heißt TrustDroid. Dabei werden sämtliche Kanäle auf dem mobilen Gerät, über welchen ein unautorisierter Informationsfluss stattfinden kann, durch zusätzliche Erweiterungen der Android Architektur verbessert (vgl. [BUGI11]). Als grundlegende Technik wird dabei ein sogenanntes Application Coloring angewendet. Hierbei wird jeder Anwendung eine Vertrauensstufe in Form einer Farbe zugewiesen. Es gibt die Vertrauensstufen vorinstallierten Systemanwendungen, vertrauenswürdige Drittanbieter-Anwendungen sowie nicht vertrauenswürdigen Anwendungen von Drittanbietern.
Während vertrauenswürdige Anwendungen und nicht vertrauenswürdige Anwendungen voneinander isoliert werden können, müssen Systemanwendungen in der Regel allen installierten Anwendungen zur Verfügung stehen, damit die Funktionalität dieser nicht eingeschränkt wird.

Die Hauptidee ist es, sämtliche Anwendungen in isolierten Domänen zu gruppieren. Isolierung bedeutet hierbei das die Kommunikation von Anwendungen zu Ressourcen oder Anwendungen in anderen Domänen via ICC (engl. Inter-Component Communication), IPC, dem Dateisystem oder über lokale Netzwerkverbindungen unterbunden wird.

**Containerlösungen in kommerziellen in MDM-System:**
Um mobile Geräte in die Unternehmens-IT-Infrastruktur einzubinden werden sogenannte Mobile-Device-Management (MDM) Systeme verwendet. Diese Systeme ermöglichen der Unternehmensseite eine zentrale Verwaltung der mobilen Geräte wie Smartphones, Tablets und Notebooks. Zudem sollen MDMs die Sicherheit und Funktionalität am Arbeitsplatz optimieren. Dies geschieht durch Funktionen zur automatischen Verteilung von Anwendungen, Daten sowie Konfigurationseinstellung. Ziel dieser MDM-Systeme ist es, dass Administratoren des Unternehmens die mobilen Geräte genauso schnell und automatisiert überwachen und verwalten können, wie es mit Desktop-Rechnern schon seit Jahren möglich ist (vgl. [ROUS13]).
Viele Anbieter von MDM-Lösungen setzten zur Steigerung der Sicherheit Container-Funktionen in ihren MDM-Systeme ein. Dazu werden im Folgenden die drei MDM-Lösungen mit ihren jeweiligen Container-Funktionen vorgestellt. Auf Basis des „Magic Quadrant for Enterprise Mobility Management Suites“ von Gartner wurden dazu die drei Marktführenden Lösungen im Bereich der MDM-Systeme ausgewählt (vgl. [GART14]).


**Abbildung 11: Gartners Magic Quadrant zum Thema MDM-Anbieter Analyse**

**AirWatch by VMware (Container):**

Der Marktführer im MDM-Sektor ist AirWatch, welches ein Tochterunternehmen des erfahrenen Virtualisierungslösungen Hersteller VMware ist. Dieser setzt mit Workspace Management, wie auch einige weitere Marktteilnehmer, auf Containiersierung zur Steigerung der Sicherheit seines MDM-Systems. Workspace Management bietet laute AirWatch vollständige Trennung der Unternehmensdaten vom Rest des persönlichen Systems der Anwender in einem BYOD-Modell. Dabei bietet Workspace Management vier unterschiedliche Arten für Container-Typen an. Neben den „App“ und „Content“ Containern gibt es noch einen speziellen „Email“ und einen „Browsing“ Container Typ, welche hauseigene sichere Lösungen für die Bereiche E-Mail und Browser enthalten. Der Content-Container dient zur sicheren Verteilung sensibler Informationen und ermöglicht einfachen Zugriff auf Inhalte, Dateisynchronisation und Kollaboration.

Beim App Container Typ kann mit der Einbindung AirWatchs Hauseignen Software Development Kit (SDK) im Entwicklungsprozess einer Anwendung die Sicherheitsfunktionen der Container-Lösung in die Anwendung eingebunden werden. Alternative bietet AirWatch eine Wrapping-Funktion an, welche bereits entwickelte Anwendungen in einen sichereren Container umwandeln soll. Im Zuge der Wrapping-Funktion wird die Anwendung um verschiedene Sicherheitsfunktionen, wie zum Beispiel zusätzliche Chiffrierungsmechanismen zur Verschlüsselung der lokalen Daten sowie der Netzwerkkommunikation generisch

erweitert. Dabei werden die im Programmcode verwendeten Standartklassen der Anwendung auf neue Klassen des MDM-Systems umgebogen. Zu dem Wrapping-Verfahren sind nicht alle Anwendungen kompatibel (vgl. [MEUS15]).

**MobileIron (Container):**
Als weiterer Marktführer im Bereich der MDM-Systeme bietet MobileIron ebenso eine Container-Lösung zur Trennung der Unternehmensdaten vom persönlichen Bereich des Nutzers. Die Architektur der Container-Funktion von MobileIrons MDM-Lösung besteht aus zwei Komponenten: „AppConnect“ und „AppTunnel“.
Die Container-Erstellung bei „AppConnect“ funktioniert Analog zu „AirWatchs Workspace Management“ der Einbindung einer speziellen SDK oder durch Umwandlung mit Hilfe einer Wrapping-Funktion. „AppTunnel“ ermöglicht Mechanismen zur sicheren Kommunikation mit dem Unternehmen.

**XenMobile (Container):**
Als drittes Beispiel von Container-Funktionen in MDM-Systemen wurde XenMobile vom Citrix Systems analysiert. Ebenso wie bei den anderen MDM-Anbietern werden Anwendungen bei XenMobile nicht in Container verpackt, sondern muss durch ein Wrapping-Verfahren in Container-Anwendungen umgewandelt werden. Dieses „Applikation wrappen“ werden zusätzliche Funktionen wie zum Beispiel lokale Verschlüsselungsfunktionen und Mechanismen zur sichere Kommunikation durch Modifizierung der Anwendung hinzugefügt (vgl. [WANN15]).

## 3.3 Sicherheitssanalyse

Die Annahme, dass der Einsatz einer Virtualisierungstechnologie ein System vollkommen vor Kompromittierung schützt ist falsch. Allerdings kann der Einsatz der Virtualisierungstechnologie die Wahrscheinlichkeit, dass es zu einer Kompromittierung kommt, stark verringern. Am Beispiel von veröffentlichen Sicherheitslücken wird in Tabelle 9 dargestellt, wie das Verhältnis von bekannten Sicherheitslücken in mobilen Betriebssystemen zu bekannten Sicherheitslücken in der Virtualisierungstechnologie ist. Dazu wurden die Suchergebnisse für die mobilen Betriebssysteme Android und iOS mit den Suchergebnissen für die Suchwörter „virtualization“, „virtual machinie“ und „container“ in vier Online-Exploit-Datenbanken ausgewertet. Dabei wird die durchschnittliche Anzahl der Datenbankeinträge für die Suchwörter der Virtualisierung mit den Einträgen für die Betriebssysteme in Verhältnis gesetzt. Als Ergebnis ist Tabelle 4 zu entnehmen, dass es im Durchschnitt 891% mehr Sicherheitslücken für die Suchanfragen „Android“ und „iOS“ gab, als für die Suchanfragen der Virtualisierung. Dabei ist zu beachten, dass die Ergebnisse keine Hinweise auf die Auswirkung der Sicherheitslücken geben, aber ein allgemeines Verhältnis der Sicherheitslücken von mobilen Betriebssystemen zu der Virtualisierungstechnologie aufzeigen. Dementsprechend kann die Wahrscheinlichkeit der Kompromittierung durch den Einsatz einer Virtualisierungstechnologie um ein Vielfaches gesenkt werden.

| ***Datenbank*** / ***Suchwort*** | *exploit-db.com* | *cxsecurity.com* | *0day.today* | *packetstormsecurity.com* | ***Durchschnitt Ø*** |
|---|---|---|---|---|---|
| *virtualization* | 1 | 48 | 1 | 8 | 15 |
| *virtual machine* | 10 | 0 | 2 | 70 | 21 |
| *container* | 3 | 92 | 2 | 26 | 31 |
| ***Durchschnitt Ø*** | | | | | 21,92 |
| *android* | 54 | 247 | 55 | 18 | 94 |
| *ios* | 250 | 806 | 74 | 59 | 297 |
| ***Durchschnitt Ø*** | | | | | 195,38 |
| ***Verhältnis in %*** | 3257 | 1128 | 3870 | 111 | **891** |

**Tabelle 4: Exploits-Analyse Stand Juni 2016**

# 4 Szenario

In diesem Kapitel wird das Szenario vorgestellt, auf dessen Basis das Konzept entwickelt wird. Dazu wird das fiktive Unternehmen „Jacobi GmbH“ vorgestellt und seine strategischen Ziele für die betriebsinterne Nutzung von mobilen Geräten erläutert. Im Anschluss werden die Ziele aus diesem Szenario modelliert, um so im Konzept eine Grundlage für Entscheidungen bereitzustellen.

### 4.1.1 Das Unternehmen

Die Jacobi GmbH ist ein fiktives, international tätiges Konsumgüterunternehmen mit Hauptsitz in Hamburg. Das Unternehmen wurde 1909 gegründet und beschäftigt ca. 1000 Mitarbeiter. Der Tätigkeitsbereich der Jacobi GmbH liegt im Bereich der Herstellung und Vermarktung von zahlreichen Hautpflegeprodukten.

### 4.1.2 Unternehmenseigene Anwendungen

Zur Produktivitätssteigerung werden Anwendungen für mobile Geräte angeboten, welche durch ein externes Entwicklerstudio entwickelt wurde und durch die interne IT-Abteilung verwaltet wird. Änderungen an den jeweiligen Anwendungen werden kostenpflichtig durch das externe Entwicklerstudio durchgeführt. Folgende Anwendungen werden von der Unternehmensseite angeboten:

| | |
|---|---|
| ***Bezeichnung*** | Jacobi Messager |
| ***Beschreibung*** | Bei dem Jacobi Messager handelt es sich um einem Instant-Messaging-Dienst mit welchem Benutzer Textnachrichten, Bilddateien und Dokumente zwischen zwei Personen oder in vordefinierten Arbeitsgruppen austauschen können. |
| ***Plattform*** | Android 4.0 und iOS 8.0 |
| ***Besondere Merkmale*** | Der Nachrichtenverlauf wird mit der Unternehmensinternen Own-Cloud regelmäßig synchronisiert und gesichert. |
| ***Größe*** | 36,44 MB |

**Tabelle 5: Szenario „Jacobi Messager“ Anwendung**

| | |
|---|---|
| ***Bezeichnung*** | Jacobi Kanban-Board |
| ***Beschreibung*** | Die Jacobi Kanban-Board App ist eine Projektmanagementsoftware, mit welcher die einzelnen Abteilungs- und Projektgruppen ihre Aufgaben anhand einer digitalen Kanban Tafel verwalten und planen können. Sämtliche Daten werden auf einem zentralen Server im Unternehmensnetz gespeichert und befindet sich nur zur Einsicht und zur Bearbeitung im Arbeitsspeicher der jeweiligen Geräte. Bei Änderungen wird der Nutzer per Kanban-Board App benachrichtigt. |
| ***Plattform*** | Android 4.0 und iOS 8.0 |
| ***Besondere Merkmale*** | keine |
| ***Größe*** | 15,19 MB |

**Tabelle 6: Szenario „Jacobi Kanban-Board“ Anwendung**

| | |
|---|---|
| ***Bezeichnung*** | Jacobi Arbeitszeitverwaltung |
| ***Beschreibung*** | Im Unternehmen gibt es keine einheitliche Arbeitszeitregelung für die gesamte Belegschaft. Ein Großteil der Angestellten arbeiten im Rahmen einer Gleitzeit und einige in einer Home-Office Regelung. Damit die Verwaltung der Arbeitszeit für beide Seiten so einfach wie möglich ist, wurde die Arbeitszeitverwaltungs-App bereitgestellt. Mit dieser können die Angestellten ihre Arbeitszeiten verwalten. Zudem können von Abteilungs- bzw. Projektleitern Kernarbeitszeiten und Termine festgelegt werden. |
| ***Plattform*** | Android 4.0 und iOS 8.0 |
| ***Besondere Merkmale*** | Daten werden mit einem zentralen Server synchronisiert und unverschlüsselt auf dem Gerät gespeichert. |
| ***Größe*** | 20,45 MB |

**Tabelle 7: Szenario „Jacobi Arbeitszeitverwaltung“ Anwendung**

**Drittanbieter Anwendungen im Unternehmen:**
Zusätzlich werden noch Anwendungen für Kalender-, E-Mail- und Clouddienste von Drittanbietern auf den mobilen Geräten der Angestellten genutzt. Alle drei Dienste laufen auf separate unternehmenseigene Server in marktüblichen Konfigurationen.

### 4.1.3 Verbundene Risiken

Durch die fehlende lokale Verschlüsselung entsteht ein hohes Sicherheitsrisiko für die Daten der verschiedenen Anwendungen. Als weiteres Sicherheitsrisiko wurde die mögliche Kompromittierung der unternehmensinternen Information durch Schadsoftware auf den Geräten der Angestellten identifiziert.

Um die Kosten zu minimieren wurde durch die Unternehmensführung entschieden eine entsprechende Mobilitätsstrategie auf Basis des BYOD-Modells für eine sichere Integration der privaten Geräte in das betriebliche Umfeld zu entwerfen.
Dazu wurden bei sämtlichen Mitarbeitern eine Befragung zu den ihren verwendeten mobilen Geräten durchgeführt. Dies führte zu folgendes Ergebnis:

| ***Betriebssystem*** | ***Verteilung auf Angestellte*** | ***Niedrigste genutzte Version*** |
|---|---|---|
| Android | 48,9 % | 2.3 |
| iOS | 32,4 % | 7.1 |
| Windowsphone | 9,5 % | Windows Phone 8 |
| Sonstige | 9,2 % | |

**Tabelle 8: Szenario Umfrageergebnisse der verwendeten Betriebssysteme**

Aufgrund der Umfrageergebnisse und um unkontrollierbaren „Wildwuchs“ zu vermeiden, soll die Mobilitätsstrategie nur auf mobile Geräte mit Android- und iOS-Betriebssystemen (bei Android ab >= Version 4.0, bei iOS ab Version >= 8.0) ausgerichtet werden.

# 5 Konzept

Als Hauptteil dieser Arbeit befasst sich fünfte Kapitel mit der Entwicklung des eigentlichen Konzepts zur Virtualisierung von Anwendungen auf mobilen Geräten.
Zu Beginn wird der Aufbau und die Struktur des Konzepts vorgestellt. Im Folgenden wird nach diesem Schema die bestmögliche, verfügbare Lösung am Markt für das durch das Szenario dargestellte Problem gesucht. Diese dient wiederum als Basis für den Entwurf einer optimalen Virtualisierungslösung.

## 5.1 Methodik

Ein Konzept ist eine Methode, um ein Vorhaben klar zu definieren. Ein klar abgrenzendes Konzept sorgt dafür, dass eine Idee für die Lösung eines Problems allen Projektbeteiligten zur Verfügung gestellt wird. Deshalb dient das Konzept allgemein als wichtiger Faktor für den Erfolg eines Projekts und als Basis für erste Schätzung des Aufwandes, der Zeit und schließlich der Kosten zur Umsetzung des Projekts (vgl. [BOJA15]).
Der Umfang der einzelnen Kategorien eines Konzepts variiert dabei stark nach der Art des Vorhabens und den entsprechenden Stakeholdern. Strukturiert ist das hier entwickelte Konzept wie folgt:

1. **Kurzzusammenfassung des Vorhabens**
Eine Zusammenfassung des Vorhabens macht die wesentlichen Elemente des Konzepts transparent und stellt diese für eine Gesamtansicht dar.

2. **Ausgangslage**
Die Ausgangslage beschreibt den Ist-Zustand und soll allen Projektbeteiligten deutlich machen, warum das Vorhaben relevant und notwendig ist.

3. **Strategie**
Die Strategie ist in die Abschnitte Zielsetzung und Wirkung gegliedert. Um die Wirkung des Vorhabens möglichst genau zu prognostizieren, sollten sämtliche Fragen beantwortet werden, welche eine Auswirkung auf die Durchführung des Vorhabens haben.

4. **Realisierung**

In der Realisierung wird die unmittelbare Umsetzung des Vorhabens vorgestellt. Dazu wird aus den verfügbaren Lösungen, nach den in der Strategie vorgestellten Kriterien, die bestmögliche Lösung zum Realisieren des Vorhabens ausgewählt und vorgestellt.

**5. Rahmenbedingungen**
Die Rahmenbedingungen geben zusätzliche technische, sowie betrieblichen Voraussetzungen und Rahmenbedingungen an, welche für die Umsetzung des Vorhabens beachtet werden müssen, aber keinen direkten Einfluss auf die Realisierung besitzen.

## 5.2 Kurzzusammenfassung des Vorhabens

Der Einsatz von Virtualisierungstechnologien auf mobilen Geräten wie Smartphones, Tablets, Netbooks usw. bietet ein erhebliches Potential für Verwaltung, Sicherheit sowie Kosten für den dualen Einsatz mobiler Geräte im privaten und betrieblichen Umfeld.

Dieses Konzept soll dabei aufzeigen, was mit dem heutigen Stand der unterschiedlichen Virtualisierungslösungen und –varianten bereits möglich ist und dabei elementare Fragestellungen beantworten. Dabei wird auf Basis der bereits vorgestellten Grundlagen, der Marktanalyse und des vorgestellten Unternehmensszenarios der Entscheidungsprozess für die beste verfügbare Technologie vorgestellt. Im Anschluss an die Entscheidungsphase wird ein Konzept für die optimale Virtualisierungslösung für die Problembeschreibungen aus dem Szenario vorgestellt. Da es sich dabei um ein Konzept handelt, wird die Lösung nicht auf technische Details und Besonderheiten aller einzelner Betriebssysteme oder mobilen Geräten eingehen, sondern am Beispiel von Android aufzeigen, welche Funktionalität und Konzepte eine sichere Virtualisierungslösung aufweisen sollte. So kann diese im Rahmen eines BYOD-Modells effektiv zur Steigerung der IT-Sicherheit eines Unternehmens eingesetzt werden kann.

## 5.3 Ausgangslage

Aktuell verwenden die Angestellten des Unternehmens mehrere unterschiedliche Anwendungen um damit unternehmensinterne Dienste wie gemeinsame Kalender, Kommunikationssysteme, Arbeitszeitverwaltung, Projektplanung sowie Dokumentenzugriff durch eine unternehmenseigene Cloud zu nutzen.
Alle Anwendungen können sowohl über Anwendungen auf den unternehmenseigenen Arbeitsplatzrechnern als auch auf den mobilen Geräten der Angestellten genutzt werden. Bei den mobilen Anwendungen gibt es eine Beschränkung auf Android- (ab Version 4.0) und iOS-

Betriebssystemen (ab Version 8.0). Beim Start jeder Anwendung auf den mobilen Geräten muss der Angestellte jeweils separat für jede Anwendung seine Benutzerkennung zur Authentifizierung eingeben.
Weiterhin melden sich die Angestellten mit denselben Benutzerkennungen, welche sie zudem auch zur Anmeldung an den Personal Computern im Unternehmen verwenden, an das im Unternehmen verfügbare WLAN an.
Es gibt keine Übersicht oder Beschränkungen mit wie vielen Geräten ein einzelner Nutzer auf unternehmensinterne Dienste wie das Netzwerk, Anwendungen oder Dokumente zugreift.

## 5.4 Strategie

Im Rahmen der Entwicklung einer neuen Mobilitätsstrategie für das Unternehmen wurde sich aus Kostengründen gegen ein COPE-Modell (Corporate Owned Personally Enabled) und für das BYOD-Modell entschieden.
Das primäre Ziel der neuen Strategie soll es sein, sämtliche Unternehmensdaten des Angestellten auf seinem mobilen Gerät höchstmöglich vor Manipulation, Sabotage oder Spionage zu schützen.
Als sekundäres Ziel wird nach einer Lösung gesucht, mit welcher die bisher unverschlüsselten lokalen Daten der einzelnen mobilen Anwendungen sicher auf den mobilen Geräten der Angestellten geschützt werden können. Zudem soll es nicht mehr möglich sein, auf die Daten zuzugreifen sobald ein Angestellter das Unternehmen verlassen hat. Die Lösung soll für die Betriebssysteme Android (ab Version 4.0) und iOS (ab Version 8.0) umgesetzt werden.
Mit Hilfe der Virtualisierungstechnologie soll auf den mobilen Geräten der Angestellten eine isolierte Unternehmensumgebung erzeugt werden, welche die Umsetzung der Unternehmensinteressen ermöglichen kann.

## 5.5 Realisierung

Die Realisierung ist unterteilt in drei Abschnitte. Zu Beginn der Realisierung muss die Virtualisierungsvariante, welche sich am besten zur Umsetzung des Vorhabens eignet, aus den unterschiedlichen Virtualisierungsvarianten ausgewählt werden. Als nächstes wird das gleiche Verfahren angewendet, um aus den verfügbaren Lösungen die optimale auszuwählen. Im letzten Abschnitt der Realisierung wird ein Konzept für eine optimale Lösung vorgestellt. Dabei wird die Architektur und alle elementaren Komponenten vorgestellt, welche die Lösung zu einer sicheren Lösung des Vorhabens machen würde.

### 5.5.1 Auswahl der Virtualisierungsvariante

Bei der Auswahl der sich am besten eignenden Virtualisierungsvariante werden die einzelnen Varianten mit den primären Zielen, welche im Abschnitt 5.4 Strategie definiert wurden, in einer Kontingenztafel gegenübergestellt. Die Variante, die sämtliche Bedingungen erfüllt, wird ausgewählt und im nächsten Schritt weiter untersucht.

| **Varianten** / **Primären Ziele** | *Hypervisor* | | *Microkernel* | *Container* |
|---|---|---|---|---|
| | *Typ 1* | *Typ 2* | | |
| *BYOD* | Nein | Ja | Nein | Ja |
| *Lokale Verschlüsselung* | Ja | Ja | Ja | Ja |
| *Effizienz* | Ja | Nein | Ja | Ja |
| *Isolierung* | Ja | Ja | Ja | Ja |

**Tabelle 9: Kontigenztafel der Virtualisierungsvarianten und Primärzielen**

Wie aus der Tabelle 9 zu entnehmen ist, eignet sich für die Umsetzung des Vorhabens am besten eine Containerlösung. Die Varianten Microkernel und Hypervisor Typ 1 sind nicht in einem BYOD-Modell einsetzbar, da bei beiden Varianten die vorinstallierten Betriebssysteme gegen modifizierte Versionen ersetzt werden müssen. Dadurch kann zum einen ein Verlust der Herstellergarantie eintreten und zum anderen kann ein Unternehmen einen solchen Eingriff nicht von seinen Angestellten verlangen, da die vollständige Funktionalität und Hardware Kompatibilität nicht allgemein für mehrere unterschiedliche Geräte mit einer Lösung sichergestellt werden kann.
Die Hypervisor-Variante des Typ 2 erfüllt die Bedingung der Effizienz nicht, da es dabei zu einem deutlichen Leistungsabfall kommt. Des Weiteren wird bei dieser Virtualisierungsvariante ein komplettes zweites Betriebssystem virtualisiert und benötigt dafür dementsprechend mehr Speicherplatz im Arbeitsspeicher und auf der Festplatte des Gerätes. Aufgrund dessen kommt diese Virtualisierungsvariante wegen der marktüblichen Ressourcenknappheit vieler mobiler Geräte für das Vorhaben nicht in Frage.
Durch den höchsten Grad der Kontingenz kommt die Container-Virtualisierung als einzige Variante zur Lösung des Vorhabens in Frage.

### 5.5.2 Virtualisierungslösungen am Markt

Im nächsten Schritt werden die in der Marktanalyse vorgestellten Lösungen der Container-Virtualisierungsvariante für die Umsetzung des Vorhabens miteinander verglichen. Hierbei werden wiederum in einer Kontingenztafel die einzelnen Virtualisierungslösungen den unterschiedlichen Sekundärzielen gegenübergestellt.

| ***Lösungen*** / ***Sekundäre Ziele*** | *Bizztrust for Android* | *Airwatch by VMware* | *Mobil Iron* | *XenMobile* |
|---|---|---|---|---|
| *Verfügbarkeit für Android und iOS* | Nein | Ja | Ja | Ja |
| *Anwendungen ohne Änderungen am Quellcode zu virtualisieren* | Ja | Nein | Nein | Nein |
| *Verschlüsselung Container Daten* | Ja | Ja | Ja | Ja |
| *Verschlüsselung Kommunikation (z.B. VPN)* | Ja | Ja | Ja | Ja |
| *Verwaltung durch Unternehmen* | Ja | Ja | Ja | Ja |
| *Single-Sign-On* | Ja | Ja | Ja | Ja |

**Tabelle 10: Kontigenztafel der Container Lösungen und Sekundärziele**

Wie aus Tabelle 10 zu entnehmen ist, erfüllt keine der drei vorgestellten Lösungen sämtliche Anforderungen zur Umsetzung des Vorhabens. Die quellenoffene Lösung Bizztrust ist aktuell nur für Android verfügbar und die Lösung verlangt die Installation einer modifizierten Android-Firmware. Bei den kommerziellen Lösungen AirWatch und Mobil Iron ist die Container-Virtualisierungsfunktion nur durch Änderungen am Quellcode der zu virtualisierenden Anwendung umsetzbar. Dabei kann nicht sichergestellt werden, dass der Umwandlungsmechanismus für jede Anwendung funktioniert. Außerdem werden die einzelnen Anwendungen hierbei nicht virtualisiert, sondern die Isolierung der Anwendung wird nur durch Modifizierung des Programmcodes ermöglicht. Es wird somit für die Anwendung keine eigenständige Umgebung erzeugt, wodurch eine logische Trennung der Anwendung vom Betriebssystem umgesetzt werden.
Es wurde in der Marktanalyse keine Lösung gefunden, welche auf herkömmlichen Weg auf mobilen Geräten installiert werden können, mit welcher sämtliche Anwendungen in einer geschützten Umgebung ausgeführt werden können. Deshalb wird im Folgenden ein

konzeptioneller Entwurf für eine Container-Virtualisierungslösung vorgestellt, mit dieser das Vorhaben erfolgreich umgesetzt werden kann.

### 5.5.3 Konzeptioneller Entwurf der Containerlösung

Das Ziel einer Containerlösung ist es, eine einzelne Anwendung oder eine Gruppe von Anwendungen vom Rest des Systems logisch zu isolieren, um damit sicherzustellen, dass beide Seiten auf einander keine Auswirkung ausüben können.

Um eine effektive Containerlösung zu entwerfen müssen alle möglichen Kommunikationswege zwischen den Anwendungen identifiziert werden und durch Schutzmechanismen reguliert werden.

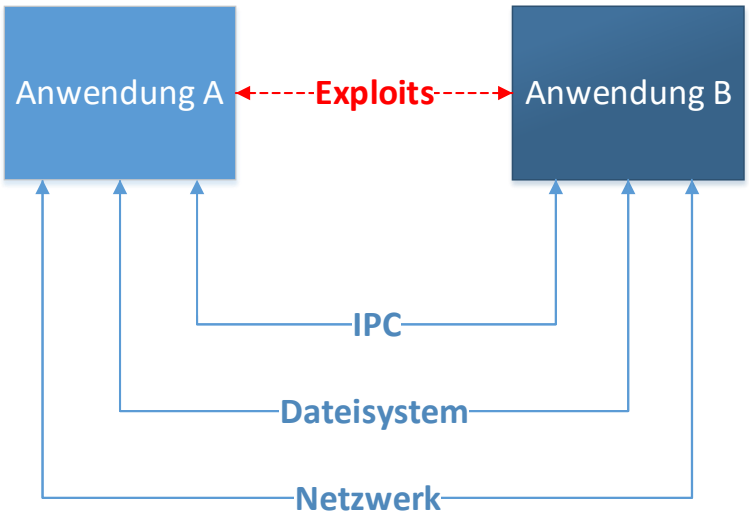


**Abbildung 12: Kommunikationswege zwischen Anwendungen**

Wie in Abbildung 12 zu entnehmen ist, gibt es in modernen Betriebssystemen drei reguläre Wege für Anwendungen mit einander zu kommunizieren. Neben der Möglichkeit des Nachrichtenaustauschs durch IPC, können Anwendungen durch das modifizieren von Daten im Dateisystem, sowie durch Nachrichtenaustausch über anwendungsinterne Netzwerk-Sockets kommunizieren. Ein zusätzlicher vierter Weg ist die Möglichkeit des Angriffs auf eine Anwendung durch Nutzung von Sicherheitslücken, sogenannter Exploits. Dabei geschieht die Kommunikation üblicher Weise über die drei Vorgestellten Kommunikationskanäle, jedoch wird dabei versucht böswilligen Programmcode durch die jeweilige Anwendung ausführen zu lassen um diese zu kompromittieren. Ein genereller Schutz vor Exploits gibt es nicht, da es sich bei den Sicherheitslücken um ungewollte Fehler im Entwurf bzw. der Implementierung

der Anwendung handelt. Allerdings gibt es unterschiedliche Mechanismen um solche Angriffe zu erkennen.
Das Ziel einer Container-Lösung ist es diese identifizierten Kommunikationswege vollständig zu regulieren und unerwünschte Kommunikationsversuche direkt zu unterbinden. Da die Containerlösung selbst kein Teil des Betriebssystems sein kann, da hierzu das vom Hersteller vorinstallierte Betriebssystem modifiziert werden müsste, bietet es sich an die Container-Lösung als Middleware zwischen dem Betriebssystem und den zu schützenden Anwendungen zu platzieren. Dadurch kann die Container-Lösung als logische Trennung fungieren und sämtliche Kommunikation in und aus den zu schützenden Bereichen regulieren und gegebenenfalls blocken. Diese Architektur ist in Abbildung 13 dargestellt.

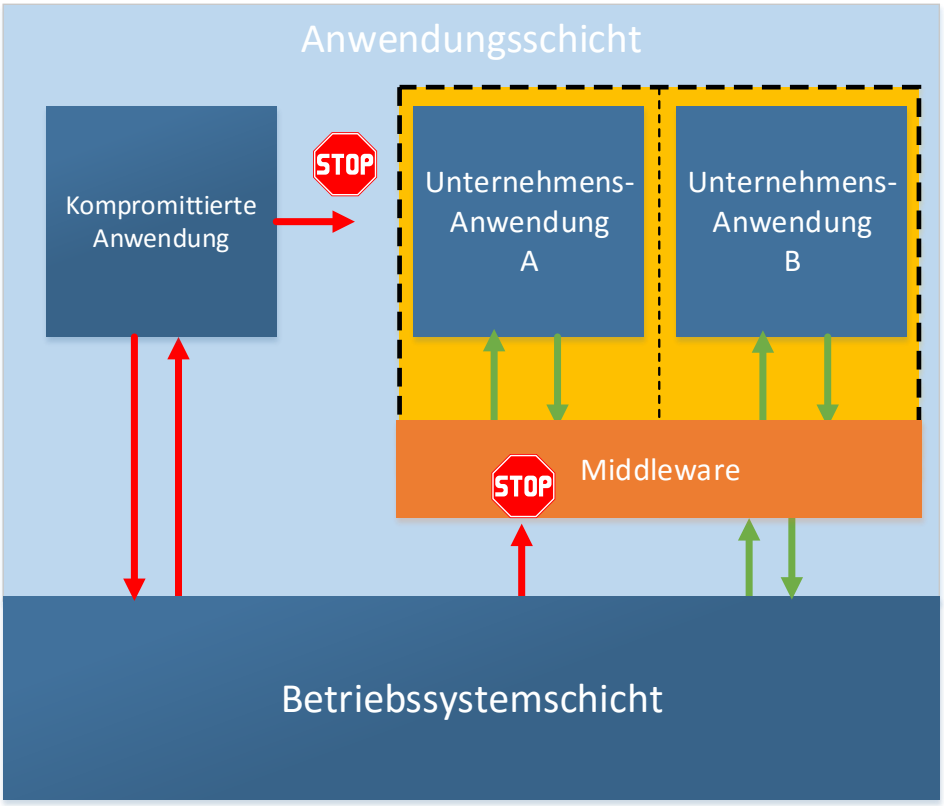


**Abbildung 13: Container-Lösung mit Middleware-Schicht**

Diese Container-Lösung benötigt unterschiedliche Komponenten um eine hohe Sicherheit für die Daten und Anwendungen im Container sicherzustellen. Dazu werden im Folgenden die für eine vollständige Lösung benötigten Komponenten vorgestellt.

**Regulierung der Interprozesskommunikation:**

Interprozesskommunikation dient Prozessen untereinander zur Kommunikation. Die Middleware muss eine zentrale Komponente besitzen, welche die ein- und ausgehende IPC Nachrichten reguliert. Damit soll der Aufbau einer unautorisierten Kommunikation unterbunden werden.

**Regulierung der Netzwerkkommunikation:**
Als Schutz vor unautorisierter Kommunikation über Netzwerkkarten sollte eine Firewall-Komponente integriert werden. Mit Hilfe vom Administrator verwaltete Regeln kann die Sicherheit über diesen Kanal verbessert werden. Zudem sollten in der Komponente Regeln definiert werden können. Mit Hilfe dieser Regeln kann anschließend ungewöhnliches Verhalten, zum Beispiel durch Angriffe, erkannt und der Benutzer darüber informiert werden.

**Verschlüsselung-Komponente:**
Sämtliche Dateien müssen, sobald sie den geschützten und isolierten Bereich der Container verlassen mit kryptografischen Verfahren chiffriert werden. Ebenso sollte die Dechiffrierung nur innerhalb der Middleware geschehen, damit es zum Beispiel durch eine Man-In-The-Middle-Attacke nicht zu einem unerlaubten Datenabfluss in Klartext kommen kann. Neben den Daten, welche persistent auf dem mobilen Gerät gespeichert werden sollen, muss auch sämtlicher ausgehende Netzwerk-Datenverkehr der einzelnen Containeranwendungen zum Beispiel mit Virtual Private Network (VPN) Verfahren verschlüsselt werden. Hierbei muss beachtet werden, dass diese Komponente auf der Serverseite zur De- bzw. Chiffrierung der Anwendungsdaten integriert werden müsste.
Bei der Auswahl der jeweiligen kryptografischen Verfahren muss die schwächere Leistungsfähigkeit der verbauten Prozessoren in den meisten mobilen Geräten beachtet werden und deshalb besonders bei der Netzwerkkommunikation ein Verfahren gewählt werden, welches die Benutzbarkeit der Anwendungen nicht unnötig stark beeinträchtigt. Da viele Daten wie zum Beispiel die Arbeitszeitdaten oder Kalenderdaten oftmals nur für eine gewisse Zeit sicher geschützt sein müssen. Die abgespeicherten Daten und Container-Anwendungen anderseits können deutlich stärker chiffriert werden, da hierbei die Dechiffrierung das Gefühl der Echtzeitverarbeitung nicht beeinträchtigen würde, da es nur bei dem Start einer Anwendung passieren würde.
Zusätzlich sollten sämtliche Daten nach dem dechiffrieren durch ein Hash-Verfahren auf Kompromittierung überprüft werden. Die berechneten Hashwerte können mit vorherigen Werten überprüft werden.

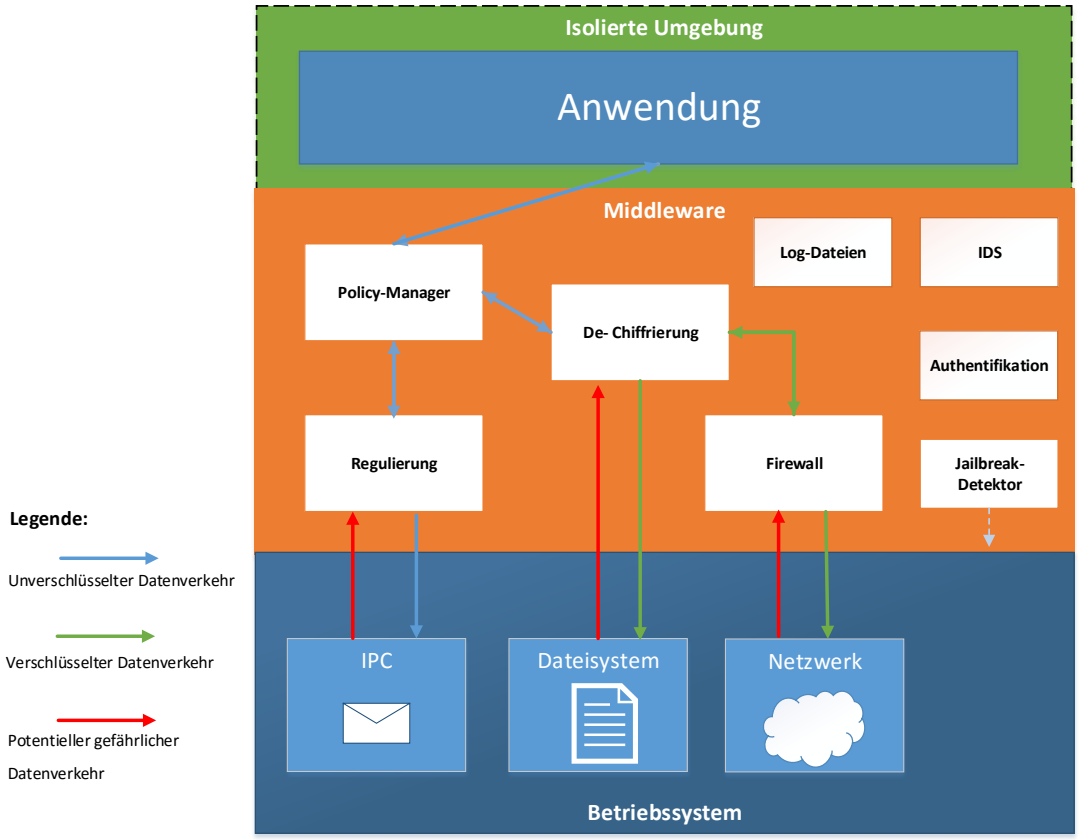


**Abbildung 14: Ein- und Ausgehender Datenverkehr durch die Middleware**

**Jailbreak-Detektor:**

In der Container-Middleware muss es eine Komponente geben, welche direkt zu Beginn die mobilen Geräte auf „Jailbreaks“ überprüft. Sobald die Sicherheitsmechanismen der Gerätehersteller einmal ausgehebelt sind, kann die Sicherheit einer Containerlösung nicht mehr sichergestellt werden. Das bedeutet, dass Anwendungen nicht mehr an die vorgegebenen Einschränkungen des Betriebssystems gebunden sind. So könnten Schadprogramme zum Beispiel unbemerkt das Mikrofon aktivieren, Daten der Speicherkarte auslesen oder sämtlichen Datenverkehr über fremde Server leiten. Ein „Jailbreak“- Detektor kann das zwar nicht verhindern, aber dafür ermöglichen, dass die entsprechenden ungesicherten mobilen Geräte frühzeitig identifiziert werden.

**Authentifikation:**

Beim Start der Containerumgebung sollte der Benutzer sich authentifizieren. Dazu sollten die Anmeldedaten des Benutzers verschlüsselt an eine Authentifizierungsstelle des Unternehmens zur Verifikation übermittelt werden. So kann sichergestellt werden das Änderungen an den Zugriffsberechtigungen für einzelne Anwendungen oder für den

gesamten Container weiterhin bestehen. Hierbei zu beachten wäre die optimale Einbringung einer „Offline“-Übergangszeit, da es vorkommen kann das ein Benutzer keine Datenverbindungen zur Verfügung stehen und trotzdem Zugriff auf gespeicherte Daten, zum Beispiel Kundendaten, zugreifen muss. Der Anmeldeprozess ist in Abbildung 15 dargestellt.

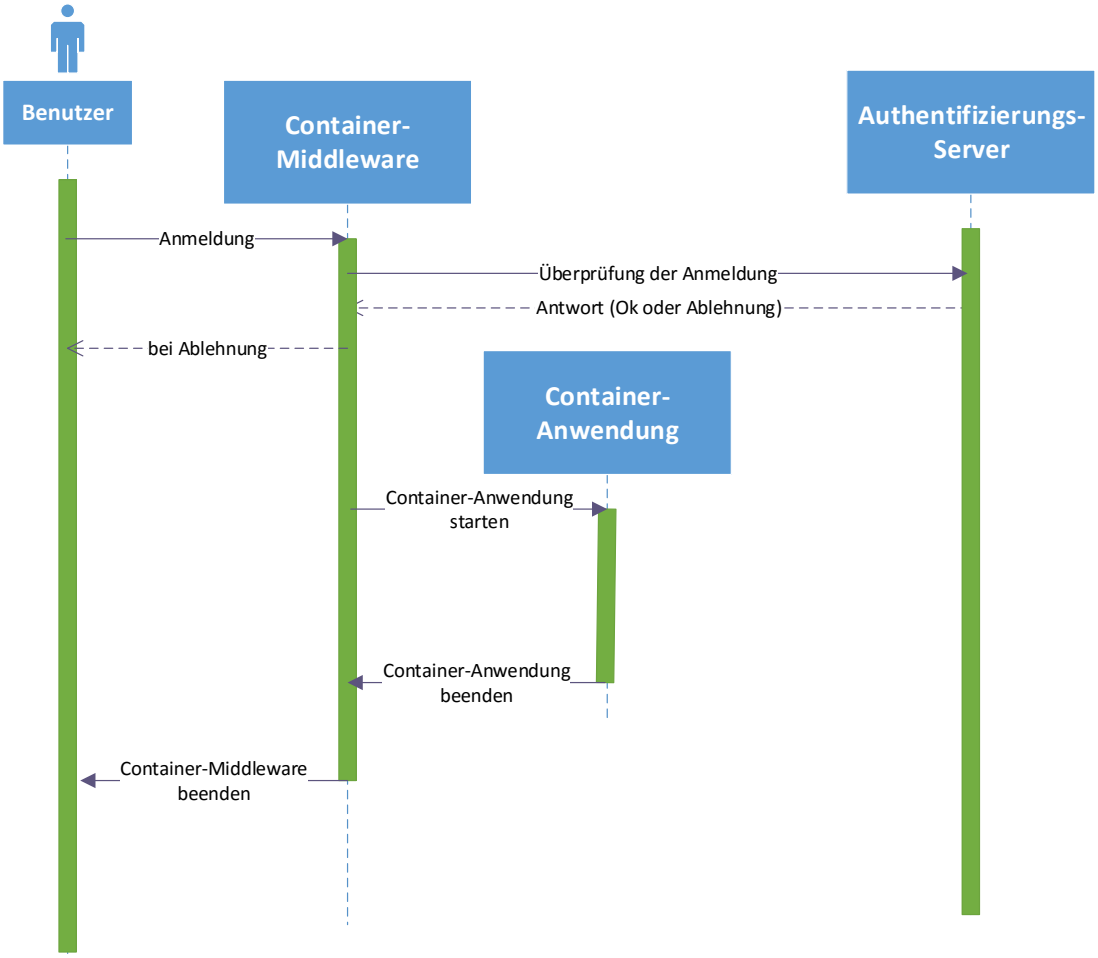


**Abbildung 15: Authentifikation beim Start der Containeranwendung**

**Policy-Manager:**

Als zentraler Kontrollmechanismus wird jegliche Kommunikation durch den Policy-Manager reguliert. Dazu benötigt der Policy-Manager vordefinierte Regeln welche die Einstellungen und Beschränkungen für die Kommunikation mit den externen Quellen wie IPC, Dateisystem sowie den unterschiedlichen Netzwerkverbindungen sicherstellt.

Für den Einsatz in einem BYOD-Modell innerhalb eines Unternehmens muss dieser Policy-Manager vom Unternehmen verwaltet werden können. Dies kann durch Push-Benachrichtigungen oder durch einen internen Pull-Mechanismus in Form eines Hintergrunddienstes umgesetzt werden.

**Angriffserkennungssystem:**
Die bisher vorgestellten Komponenten dienen hauptsächlich als allgemeine Schutzmechanismen für die verfügbaren Kommunikationskanäle. Deshalb muss es Schutzkomponenten geben, welche den Inhalt der Container vor Angriffsversuchen durch Exploits schützen.
Anti-Viren-Software ist so konzipiert, dass sie ausschließlich den Programmcode auf bereits bekannte Muster analysiert und den schädlichen Code dann blockiert. Ein Intrusion-Detection-System (IDS) hingegen verfolgt bei der Abwehr von böswilligen Programmcode eine andere Strategie. Es durchsucht sämtliche Kommunikation nach potenziell sicherheitsgefährdenden Exploits. Hierdurch bietet ein IDS nicht nur Schutz vor bekanntem Schadcode, sondern vor allem vor unbekanntem Schadcode. Um dies zu erreichen benötigt das IDS vordefinierte Regeln für die im Container ausgeführten Anwendungen, um ungewöhnliches Verhalten zu erkennen.
Das IDS entscheidet dabei jedoch nicht selbstständig, ob der bösartige Programmcode blockiert werden soll oder nicht. Es informiert lediglich den Benutzer beziehungsweise die Administratoren des Unternehmens, dass ein potenzielles Sicherheitsrisiko vorliegt, sobald festgestellt wird, dass eine Nachricht (z.B. über IPC oder das Netzwerk) einen Exploit verwendet. Auf diese Weise arbeitet die Komponente zugriffskontrollierend und muss sämtliche Kommunikationskanäle überwachen.

**Log-Dateien:**
Als zusätzliche Möglichkeit Fehlverhalten oder potentielle Kompromittierung zu erkennen, sollten Log-Dateien angelegt und mit entsprechenden Einträgen gefüllt werden.
Log-Dateien sind eine chronologische Auflistung der durchgeführten Aktivitäten und sollten überall dort verwendet werden, wo verschiedene Vorgänge nachvollziehbar dokumentiert werden sollen.

**Weitere Schutzmechanismen:**
Besonders für den beruflichen Einsatz wäre zusätzliche Funktionen wie die Unterbindung der Screenshot- und Kopier-Funktionen ratsam. Jedoch sind sie für die allgemeine Funktion und Sicherheit des Entwurfes unwichtig und werden deshalb nicht detaillierter beschrieben. Eine Übersicht der unterschiedlichen Komponenten wird oben in Abbildung 14 dargestellt.

**Vergleich der Anforderungen:**
Der konzeptionelle Entwurf erfüllt die in Kapitel 2.4 vorgestellten Anforderungen an eine Virtualisierungslösung für mobile Geräte wie folgt:

**1. Leichtgewichtige Virtualisierungsschicht mit Unterstützung für mehrere separate virtuelle Umgebungen:**
Die Vorgestellte Lösung kann mehrere separate Containeranwendungen unterstützen. Da es keine Implementierung dieses Entwurfs gibt, kann keine Beurteilung zu den benötigten Ressourcen erfolgen.

**2. Kommunikation zwischen den Systemkomponenten müssen eine geringe Latenz und eine hohe Bandbreite bieten:**
Da der Entwurf so konzeptioniert wurde, dass sich alle Komponente innerhalb der Middleware befinden und nur die unterschiedlichen Kommunikationskanäle überwacht werden, sollte es zu keiner hohen Latenz kommen. Die Bandbreite der gesamten Kommunikation stimmt mit der nativen der einzelnen Geräte überein, da nach der Prüfung auf die Kommunikationsmechanismen des unterliegenden Betriebssystems zugegriffen wird. – den Satz verstehe ich nicht

**3. Systemweite Umsetzung der Sicherheitspolitik:**
Die Sicherheitspolitik erfolgt auf alle Containeranwendungen und wird hierdurch für das gesamte virtuelle System umgesetzt.

**4. Minimale Auswirkung auf Systemressourcen und der Echtzeitverarbeitung:**
Analog zu der zweiten Anforderung sollten die Systemressourcen durch diese Virtualisierungslösung keine bemerkbaren Auswirkungen auf die Echtzeitverarbeitung ausweisen. Der einzige zusätzliche Zeitaufwand könnte durch die Überprüfung der Kommunikationskanäle zustande kommen.

**5. Auf ARM-Architekturen Umsetzbar sein:**
Da die komplette Virtualisierungsschicht als einzelne Anwendung für die jeweiligen Betriebssysteme wie Android und iOS entworfen wurde, ist die Lösung auf der ARM-Architektur einsatzfähig.

**6. Hohe Benutzerfreundlichkeit:**
Die Benutzerfreundlichkeit hat durch zusätzliche Eingabe eines Passworts für die Containeranwendungen einen minimalen Rückgang. Dies steht aber im Vergleich zu der Nutzung eines zusätzlichen mobilen Geräts und ist annehmbar. Die Benutzerfreundlichkeit der einzelnen Anwendungen ist durch die Virtualisierung in Containern unverändert.

## 5.6 Rahmenbedingungen

Damit das Vorhaben erfolgreich Umgesetzt werden kann, müssen gesetzliche Regelungen zum Thema Datenschutz beachtet werden.

### 5.6.1 Datenschutz

Damit der im Abschnitt 5.5 vorgestellte Entwurf nach seiner Implementierung innerhalb eines Unternehmens eingesetzt werden kann, müssen unterschiedliche geltende Gesetzte des jeweiligen Landes beachtet werden. Im weiteren Verlauf wird hierbei der Fokus auf die geltenden Gesetze der Bundesrepublik Deutschland gelegt. So muss der Entwickler bzw. Vertreiber solch einer Virtualisierungslösung darlegen, dass in seiner Anwendung hinreichende technische und organisatorische Sicherheitsmaßnahmen zum Schutz personenbezogener Daten implementiert worden sind. Nachfolgend wird ein Auszug der allgemeinen Grundsätze des Datenschutzes aufgeführt und mit der Containerlösung in Verbindung gesetzt.

**1. Grundsätze der Datenvermeidung und Datensparsamkeit**
Die Gebote zur Datenvermeidung und Datensparsamkeit müssen berücksichtigt werden (§ 3a BDSG). Bei der Auswahl und Gestaltung der Containerlösung ist daher der Grundsatz, nur so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten und zu nutzen, zu berücksichtigen, ohne jedoch die Sicherheit der Unternehmensdaten zu gefährden.

**2. Transparenz**
Dem Nutzer muss eine klar verständliche Beschreibung der Datenverarbeitung sowie etwaige Datenübermittlungen bzw. Zugriffsrechte zur Verfügung gestellt werden.

**3. Informationspflichten der §§ 13 ff. TMG**

Ebenfalls muss die Lösung ein den Anforderungen von § 5 TMG entsprechendes Impressum enthalten und der Nutzer sollte in klar verständlichen Worten im Rahmen einer Datenschutzerklärung über die Datenverarbeitung, insbesondere Art, Umfang und Zweck der Erhebung und Verwendung personenbezogener Daten hinreichend aufgeklärt werden.

**4. Zweckbindung und Zweckänderung**
Der Anbieter der Virtualisierungslösung hat bei der Datenspeicherung, -verarbeitung und –Nutzung sicherzustellen, dass die erhobenen Daten nur gemäß ihrer Zweckbestimmung verarbeitet werden oder dass eine gesetzlich zulässige Zweckänderung gegeben ist.

**5. Zulässigkeit der Datenverarbeitung**
Auch muss sichergestellt werden, dass personenbezogene Daten nur erhoben, verarbeitet oder genutzt werde, weil entweder ein Gesetz dieses zulässt oder der Betroffene eingewilligt hat.

### 5.6.2 Benötigte Berechtigungen der Virtualisierungslösung

Da sämtliche Container-Anwendungen durch die Virtualisierungslösung nach außen hin präsentiert werden, müssen aufgrund der Informationspflicht die Datenschutzerklärungen der verwendeten Anwendungen mit der der Virtualisierungslösung kombiniert werden.
Wenn eine Anwendung Zugriff auf die Kamera benötigt, muss die Virtualisierungslösung dieses Zugriffsrecht ermöglichen. Dementsprechend muss die Virtualisierungslösung sämtliche Rechte bekommen beziehungsweise anfordern können, welche die einzelnen Container-Anwendungen zur Ausführung benötigen.

# 6 Schluss

Als Abschluss dieser Arbeit werden die gewonnenen Erkenntnisse zusammengefasst. Hierzu wird ein Resümee des Konzepts erfolgen und ein Ausblick auf den aktuellen Stand und zukünftige Entwicklungen der Virtualisierungstechnologie auf mobile Geräte gegeben.

## 6.1 Zusammenfassung

Das Ziel dieser Bachelorarbeit, ein Konzept zur Virtualisierung von Anwendungen auf mobilen Geräten auszuarbeiten, wurde erreicht. Zudem wurde ein konzeptioneller Entwurf für eine Virtualisierungslösung angefertigt, der die Umsetzung des Vorhabens aus dem vorgestellten Szenarios ermöglichen würde.
Zu Beginn der Arbeit wurden die Grundlagen der Virtualisierung analysiert und die unterschiedlichen Virtualisierungsvarianten vorgestellt. Zudem wurde der Markt der Virtualisierungslösungen für mobile Geräte betrachtet und zu jeder vorgestellten Virtualisierungsvariante die führenden Lösungen analysiert. Die Grundlagen und die Marktanalyse bilden die Basis für das Konzept und dienen dort als Entscheidungsgrundlage zur Auswahl der besten Lösung.
Als Grundlage für das Konzept wurde ein Szenario entworfen und vorgestellt. Dabei war das Ziel eine möglichst realistische und marktübliche Unternehmenssituation darzustellen, damit die im Konzept entworfene Lösung eine hohe Portierbarkeit auf ähnliche Szenarien besitzt.
Das Konzept besteht aus den vier Abschnitten Ausgangslage, Strategie, Realisierung und Rahmenbedingungen. In der Ausgangslage wurde die Ist-Situation des Unternehmens aus dem Szenario analysieret. Der Abschnitt Strategie definiert die primären und sekundären Ziele und stellt den Rahmen des Vorhabens auf. Diese Lösung des Vorhabens wird im dritten Abschnitt Realisierung ermittelt. Dazu werden als erstes die unterschiedlichen Virtualisierungsvarianten mit den primären Zielen gegenübergestellt. Da bis auf die Container- und die Hypervisor Typ 2-Virtualisierung sämtliche anderen vorgestellten Virtualisierungsvarianten den Austausch des ursprünglichen Betriebssystems erfordern, sind diese Varianten innerhalb eines BYOD-Modells nicht einsetzbar. Die Variante des Hypervisor-Typ 2 scheidet wegen hoher Ressourcenanforderungen und wegen mangelnder Effizienz als Lösung des Vorhabens aus. Danach wurden im nächsten Schritt der Realisierung die in der Marktanalyse vorgestellten Lösungen der Container-Virtualisierung untersucht. Da keine der

vorgestellten Lösungen die Anforderungen zur Umsetzung des Vorhabens erfüllen wird im dritten Abschnitt der Realisierung ein konzeptioneller Entwurf, eine passenden Container-Virtualisierungslösung, entworfen. Die Architektur dieses Entwurfs besteht aus der Virtualisierungsschicht, welche eine Middleware darstellt, sowie aus den einzelnen Container-Anwendungen. Die Middleware-Anwendung verwaltet und reguliert sämtliche Kommunikation zwischen den Container-Anwendungen und dem restlichen Betriebssystem. Zudem besitzt die Virtualisierungsschicht mehrere Sicherheitsmechanismen um die Daten und Anwendungen vor unbefugter Manipulation und Spionage zu schützen. Der letzte Abschnitt befasst sich mit den Rahmenbedingungen des Konzepts und beantwortet Fragen wie zum Beispiel zum Datenschutz, welche nicht während der Realisierung beantwortet wurde.

## 6.2 Ausblick

Weltweit steigen die Verkaufszahlen von mobilen Geräten wie Smartphones und Tablets immer weiter, während die Verkaufszahlen von stationären Arbeitsplatzrechnern abnimmt (vgl. [STAT16a], [STAT16b]). Dieser Entwicklung können sich Unternehmen nicht entziehen, so wird es wichtiger eigene Strategie darauf auszurichten.
Der Trend geht immer mehr in Richtung BYOD-Modelle, da der Fortschritt ermöglicht, dass flexible Mitarbeiter ebenso effizient von zu Hause oder auf Reisen arbeiten können. Laut einer von Gartner weltweit durchgeführten Umfrage zum Thema „Bring Your Own Device“, planen knapp zwei Fünftel aller befragten Unternehmen schon ab 2016 keine Smartphones für ihre Angestellten mehr zu kaufen. Denn diese sollen dann ihre eigenen Geräte auch im beruflichen Umfeld einsetzten. Weiter geht aus der Umfrage hervor, dass schon ab 2017 die Hälfte der Unternehmen voraussetzen, dass ihre Angestellten eigene Smartphones für berufliche Zwecke nutzen (vgl. [MEUL13]). Diese Entwicklung birgt für die Unternehmen Chancen und Risiken. Für Mitarbeiter kann es angenehmer sein, ihre privaten Geräte auch für berufliche Zwecke zu verwenden, anstatt zwei Geräte mit sich führen zu müssen. Damit es aber nicht zu einem „Bring Your Own Malware“-Modell kommt, muss eine Mobilitätsstrategie entworfen werden, welche eine effiziente Nutzung diese mobilen Geräte ermöglicht und parallel die Sicherheit der IT-Unternehmensinfrastruktur unterstützt.
Die Virtualisierungstechnologie kann dazu eingesetzt werden, um eine logische Trennung zu erzeugen, welche die Unternehmensdaten von den privaten Daten der Mitarbeiter isoliert. Wie in der Marktanalyse vorgestellt, gibt es zu jeder Virtualisierungsvariante vielversprechende Lösungsansätze. Jedoch ist deren Einsatzfähigkeit innerhalb von BYOD-Modellen nur gering und wird sich mittelfristig nicht ändern, sofern die Betriebssystem Hersteller in diesen Bereich nicht selbst Lösungen präsentieren werden. Für aktuelle Betriebssysteme besitzt die Container-Virtualisierung die größten Chancen in den

kommenden Jahren zu einer produktiven Lösungsmöglichkeit entwickeln. An Container-Lösungen wie Docker im Serverbereich kann man die Vorteile und Leistungsfähigkeit der Container-Virtualisierung schon heute beobachten. Docker steht dort selbst noch am Anfang auf den Weg zum „Plateau der Produktivität“ (vgl. [DAWS15]).

# Literaturverzeichnis

**[ADAM06]**
Adams, Keith; Agesen, Ole (2006): A comparison of software and hardware techniques for x86 virtualization. In: *ACM Sigplan Notices* 41 (11), S. 2–13.

**[ANDR11]**
Andrus, Jeremy; Dall, Christoffer; Van't Hof, Alexander; Laadan, Oren; Nieh, Jason (2011): Cells: a virtual smartphone architecture. 23rd ACM Symposium on Operating Systems Principles. Cascais, Protugal, Oktober 2011. Online verfügbar unter http://sigops.org/sosp/sosp11/current/2011-Cascais/13-andrus-slides.pdf, zuletzt geprüft am 25.04.2016.

**[ARM15]**
ARM Holdings (2015): ARM Holdings plc 2015 Analyst and Investor Day. Online verfügbar unter http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MzA0NTAyfENoaWxkSUQ9LTF8VHlwZT0z&t=1&cb=635778262245579577, zuletzt geprüft am 25.04.2016.

**[ARMA09]**
Armand, François; Gien, Michel (2009): A practical look at micro-kernels and virtual machine monitors. In: Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE. IEEE, S. 1–7.

**[BALM14]**
Balmes, F. (2014): Virtualisierungstechniken. Grundlagen und Anwendung im Serverbetrieb: Diplomica Verlag. Online verfügbar unter https://books.google.de/books?id=o12wBQAAQBAJ.

**[BREG15]**
Bregenzer, J. (2015): Effizienter Einsatz von Multicore-Architekturen in der Steuerungstechnik: Würzburg University Press. Online verfügbar unter https://books.google.de/books?id=yAk0CgAAQBAJ.

**[CELL15]**
Cellrox Ltd. (Hg.) (2015): Cellrox Thinvisor. Online verfügbar unter http://www.cellrox.com/product/, zuletzt geprüft am 22.06.2016.

**[DALL14]**
Dall, Christoffer; Nieh, Jason (2014): KVM/ARM: The design and implementation of the Linux ARM hypervisor. In: ASPLOS '14: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems. New York, NY, USA: ACM, S. 333–348.

**[DAWS15]**
Dawson, Philip; Hill, Nathan (2015): Hype Cycle for Virtualization. Hg. v. Inc. Gartner. Online verfügbar unter https://www.gartner.com/doc/3089927/hype-cycle-virtualization-, zuletzt geprüft am 22.06.2016.

**[EISE11]**
Eisen, Morty (2011): Introduction to Virtualization. IEEE Circuits and Systems (CAS) Society, 28.04.2011. Online verfügbar unter https://www.ieee.li/pdf/viewgraphs/introduction_to_virtualization.pdf, zuletzt geprüft am 20.03.2016.

**[GOLD72]**
Goldberg, Robert P. (1972): Principles for Virtual Computer Systems. Dissertation. Harvard University, Cambridge, MA, USA.

**[HANL16]**
Hanley, Mike (2016): Duo Analytics: Android Device Security. Hg. v. Duo Labs. Online verfügbar unter https://duo.com/blog/duo-analytics-android-device-security-article, zuletzt geprüft am 04.04.2016.

**[HEIS08]**
Heiser, Gernot (2008): The role of virtualization in embedded systems. In: Proceedings of the 1st workshop on Isolation and integration in embedded systems, S. 11–16.

**[HEIS10]**
Heiser, Gernot; Leslie, Ben (2010): The OKL4 microvisor: convergence point of microkernels and hypervisors. In: Proceedings of the first ACM asia-pacific workshop on Workshop on systems, S. 19–24.

**[HWAN08]**
Hwang, J. Y. (2008): 5th IEEE Consumer Communications and Networking Conference, 2008. CCNC 2008. Xen on arm: System virtualization using xen hypervisor for arm-based secure mobile phones: IEEE / Institute of Electrical and Electronics Engineers Incorporated.

**[IQBA09]**
Iqbal, Asif; Sadeque, Nayeema; Mutia, Rafika Ida (2009): An overview of microkernel, hypervisor and microvisor virtualization approaches for embedded systems. In: *Report, Department of Electrical and Information Technology, Lund University, Sweden* 2110.

**[JARA14]**
Jaramillo, David; Furht, Borivoje; Agarwal, Ankur (2014): Virtualization techniques for mobile systems. Cham: Springer (Multimedia Systems and Applications).

**[KLEI09]**
Klein, Gerwin (2009): The L4.verified project. A Formally Correct Operating System Kernel. Hg. v. NICTA. Online verfügbar unter http://ssrg.nicta.com.au/projects/TS/l4.verified/, zuletzt geprüft am 31.03.2016.

**[MACA13]**
Macario, Gianpaolo (2013): LXCBENCH. Understanding the capabilities of Linux Containers in IVI applications through benchmarkingbenchmarking. Automotive Linux Summit Fall. Edinburgh, UK, 10.2013.

**[MEUS15]**
Meuser, Peter (2015): Datenlecks in iOS-8-Container festgestellt. Hg. v. IDG Business Media GmbH. Online verfügbar unter http://www.computerwoche.de/a/datenlecks-in-ios-8-container-festgestellt,3091751, zuletzt geprüft am 22.06.2016.

**[PICH09]**
Pichler, Thomas (2009): Sicherheits-Beweis für Betriebssystem-Kernel. Forscher melden mathematischen Nachweis für fehlerfreien Code. Hg. v. pressetext.austria. Online verfügbar unter http://www.pressetext.com/news/20090817022, zuletzt geprüft am 31.03.2016.

**[PICHT09]**
Picht, Hans-Joachim (2009): Xen-Kochbuch: O'Reilly Media, Inc.

**[POPE74]**
Popek, Gerald J.; Goldberg, Robert P. (1974): Formal Requirements for Virtualizable Third Generation Architectures. In: *Commun. ACM* 17 (7), S. 412–421. DOI: 10.1145/361011.361073.

**[ROUS13]**
Rouse, Margaret (2013): Mobile Device Management (MDM). Online verfügbar unter http://www.searchnetworking.de/definition/Mobile-Device-Management-MDM, zuletzt geprüft am 22.06.2016.

**[SMIT05]**
Smith, J. E.; Nair, R. (2005): Virtual Machines: Versatile Platforms for Systems and Processes: Morgan Kaufmann Publishers. Online verfügbar unter https://books.google.de/books?id=LCtNaLNOa4gC.

**[STAT16a]**
Statista GmbH (Hg.) (2016): Absatz von Smartphones weltweit vom 1. Quartal 2009 bis zum 1. Quartal 2016. Online verfügbar unter http://de.statista.com/statistik/daten/studie/246300/umfrage/weltweiter-absatz-von-smartphone-nach-quartalen/.

**[STAT16b]**
Statista GmbH (Hg.) (2016): Prognose zum Absatz von Tablets, Laptops und Desktop-PCs weltweit von 2010 bis 2020. Online verfügbar unter http://de.statista.com/statistik/daten/studie/183419/umfrage/prognose-zum-weltweiten-absatz-von-pcs-nach-kategorie/.

**[TRIT06]**
Tritsch, Bernhard; Anderson, Christa (2006): Whitepaper Skalierbarkeit von 64-Bit Terminalserver-Plattformen. Hg. v. visionapp GmbH. Online verfügbar unter http://www.visionapp.com/uploads/secure/mit_download/WP_64-bit-Scalability_DE.pdf, zuletzt geprüft am 22.06.2016.

**[VAHI13]**
Vahidi, Arash; Ekdahl, Patrik (2013): VETE: Virtualizing the Trusted Execution Environment.

**[MEUL13]**
Meulen, Rob van der; Rivera, Janessa (2013): Gartner Predicts by 2017, Half of Employers will Require Employees to Supply Their Own Device for Work Purposes. Enterprises That Offer Only Corporate-Liable Programs Will Soon Be the Exception. Hg. v. Inc. Gartner. Online verfügbar unter http://www.gartner.com/newsroom/id/2466615.

**[VARA10]**
Varanasi, Prashant (2010): Implementing Hardware-supported Virtualization in OKL4 on ARM. UNIVERSITY OF NEW SOUTH WALES.

**[WANN15]**
Wannagat, Danny (2015): Citrix Techtalk: XenMobile MDX Container Security - Wie funktionieren der MDX Container und die Wrapping Technology? Hg. v. Inc Citrix Systems. Online verfügbar unter https://www.citrix.de/events/citrix-techtalk-xenmobile-mdx-sontainer-security-de.html, zuletzt geprüft am 22.06.2016.

**[WITT06]**
Witt, Bernhard C. (2006): IT-Sicherheit kompakt und verständlich. In: *Eine praxisorientierte Einführung, Wiesbaden*.

**[XU09]**
Xu, Jingli (2009): Virtualisierung als Möglichkeit der Optimierung des IT-Managements: Igel Verlag.

**[XU15]**
Xu, Lei; Wang, Zonghui; Chen, Wenzhi (2015): The study and evaluation of ARM-Based mobile virtualization. In: *International Journal of Distributed Sensor Networks* 2015, S. 1.

# Versicherung über Selbstständigkeit

*Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.*

*Hamburg, den* _____ _____