

Institut für Parallele und Verteilte Systeme

Universität Stuttgart  
Universitätsstraße 38  
70569 Stuttgart

Diplomarbeit Nr. 3738

# Ein sicherer Datencontainer für die Cloud

Fritjof A. Mayer

**Studiengang:** Informatik

**Prüfer:** PD Dr. Holger Schwarz

**Betreuer:** Dipl.-Inf. Christoph Stach

**Beginn am:** 18. Juni 2015

**Beendet am:** 18. Dezember 2015

**CR-Nummer:** C.2.4, E.3, H.2.8, H.3.5, K.6.5



## Kurzfassung

Die zunehmende Verbreitung von mobilen Endgeräten und Cloud-Diensten führt auch zur Erfassung großer Mengen privater Daten. Um die Gefahr eines Missbrauchs dieser Daten zu verringern, können Zugriffsschutzsysteme und Verschlüsselung eingesetzt werden. Neben dem Schutz der Vertraulichkeit der gespeicherten Daten besteht aber auch das Interesse, die Ressourcen der Cloud zur Datenanalyse zu nutzen.

Mit der *Privacy Management Platform* [SM14], einem alternativen Berechtigungssystem für Android, und dem *Secure Data Container* [SM15] existiert bereits ein ganzheitliches Datensicherheitssystem für mobile Endgeräte. Diese Arbeit überträgt den Ansatz des *Secure Data Containers* auf die Datenspeicherung in der Cloud, wobei vor allem untersucht wird, wie umfangreiche Analysemöglichkeiten mit dem Schutz der Vertraulichkeit der Daten kombiniert werden können.

Dazu werden verschiedene Ansätze zur Durchführung von Analysen auf verschlüsselten Daten sowie verwandte Arbeiten vorgestellt. Der entwickelte Datencontainer wurde als Prototyp implementiert, was in dieser Arbeit erläutert wird. Inwieweit der Datencontainer verschiedene Angriffsszenarien abwehren kann, wird im Rahmen einer Evaluation untersucht.



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>9</b>
1.1	Motivation . . . . .	9
1.2	Aufgabenstellung . . . . .	11
<b>2</b>	<b>Grundlagen</b>	<b>13</b>
2.1	Zugriffsschutz . . . . .	13
2.2	Verschlüsselung . . . . .	15
2.2.1	Sicherheitskriterien . . . . .	15
2.2.2	Ende-zu-Ende-Verschlüsselung . . . . .	16
2.2.3	Homomorphe Verschlüsselung . . . . .	17
2.2.4	Deterministische Verschlüsselung . . . . .	17
2.2.5	Datenträgerverschlüsselung . . . . .	17
2.2.6	Datenbank-Verschlüsselung . . . . .	18
2.3	Sicheres Löschen . . . . .	18
<b>3</b>	<b>Verwandte Arbeiten</b>	<b>21</b>
3.1	Privacy Management Platform . . . . .	21
3.2	Secure Data Container . . . . .	21
3.3	VeraCrypt . . . . .	22
3.4	Tresorit . . . . .	23
3.5	CryptDB . . . . .	23
3.6	Cipherbase . . . . .	24
3.7	Übersicht . . . . .	24
<b>4</b>	<b>Lösungskonzept</b>	<b>25</b>
4.1	Anforderungen . . . . .	25
4.1.1	Funktionale Anforderungen . . . . .	25
4.1.2	Nicht-funktionale Anforderung . . . . .	26
4.2	Architektur . . . . .	26
4.3	Datenverwaltung . . . . .	27
4.4	Zugriffsschutz . . . . .	28
4.5	Verschlüsselung . . . . .	28
4.5.1	Temporäre Entschlüsselung in der Cloud . . . . .	29
4.5.2	Ende-zu-Ende-Verschlüsselung . . . . .	30

4.6	Datenaustausch . . . . .	31
4.7	Sicheres Löschen . . . . .	31
<b>5</b>	<b>Implementierung</b>	<b>33</b>
5.1	Cloud-SDC . . . . .	33
5.1.1	Google App Engine . . . . .	33
5.2	Benutzerverwaltung . . . . .	34
5.3	DBMS . . . . .	34
5.4	Schnittstellen . . . . .	36
5.5	PMP-App . . . . .	37
<b>6</b>	<b>Evaluation</b>	<b>41</b>
6.1	Angreifer . . . . .	41
6.1.1	Administratoren . . . . .	41
6.1.2	Cloud-SDC-Nutzer . . . . .	42
6.1.3	Sonstige Angreifer . . . . .	42
6.2	Angriffsszenarien . . . . .	42
6.2.1	Angriffe auf Clients . . . . .	42
6.2.2	Angriffe auf den Cloud-SDC . . . . .	43
6.2.3	Angriffe auf das DBMS . . . . .	44
6.2.4	Übersicht . . . . .	45
<b>7</b>	<b>Zusammenfassung &amp; Ausblick</b>	<b>47</b>
	<b>Literatur</b>	<b>49</b>

# Abbildungsverzeichnis

2.1	Zugriffsschutz . . . . .	14
4.1	Architektur . . . . .	27
4.2	Schlüsselverwaltung . . . . .	32
5.1	Models im NDB Storage . . . . .	35
5.2	Netzwerkanalyse in Firefox. Per Webformular wird die Methode <code>list</code> aufgerufen. In den Kopfzeilen ist der Cookie zur Authentisierung zu sehen, sowie Content-Type von Anfrage und Antwort. . . . .	38
5.3	Übergabe der Parameter der Methode <code>list</code> . . . . .	38
5.4	Antwort des DBMS: Alle gespeicherten Strings, die mit „ba“ beginnen. . . .	39
6.1	Angriffsszenarien . . . . .	43





# 1 Einleitung

## 1.1 Motivation

Im März 2015 wurde bekannt, dass Unberechtigte Zugriff auf Kundendaten von Premera Blue Cross erlangten, einem Krankenversicherungsunternehmen in Washington, USA [Pre15]. Es handelt sich bei den Daten um Namen, Geburtsdaten, Sozialversicherungsnummern, Kontonummern, Adressen, E-Mail-Adressen, Telefonnummern, Kundennummern sowie Versicherungs- und Gesundheitsdaten bis zurück ins Jahr 2002 von bis zu 11 Millionen Kunden [Pfa15]. Ähnlich gelagerte Vorfälle wurden im August 2015 bei Excellus BlueCross BlueShield (10 Mio. betroffene Kunden) [Kov15] und im Februar 2015 bei Anthem Inc. (78,8 Mio. betroffene Kunden) [Kir15] bekannt, beides ebenfalls US-amerikanische Krankenversicherungen. Im August 2014 wurden private Nacktfotos von verschiedenen Prominenten unerlaubt veröffentlicht. Die Fotos waren in Apples Online-Speicherdienst iCloud<sup>1</sup> abgelegt [mmq14]. Im Rahmen der 2013 von Edward Snowden enthüllten Geheimdienst Dokumente wurde bekannt, dass beispielsweise Microsoft dem amerikanischen Geheimdienst NSA den Zugriff auf ihren Cloudspeicher OneDrive [Mic15a] (früher SkyDrive) ermöglichte, der damals 250 Millionen Nutzer hatte [GMP+13]. Wie kommen all diese Daten ins Internet und warum erlangen unberechtigte Person Zugriff darauf?

Die Verbreitung von Smart Devices wie Smartphones und Tablets hat in den letzten Jahren immer weiter zugenommen: Im Jahre 2011 hatten in den USA schätzungsweise 44% der Bevölkerung ein Smartphone - 2013 waren es bereits 65% [Fin14]. Auch die Zahl der weltweit genutzten Geräte sowie die weltweiten jährlichen Verkaufszahlen sind gestiegen. Die Anzahl der weltweit genutzten Smartphones betrug 708 Millionen im dritten Quartal 2011. Im dritten Quartal 2012 waren 1,04 Milliarden Geräte weltweit in Benutzung [Bus12]. 2013 wurden bereits 970 Millionen Smartphones weltweit verkauft, 2014 waren es 1,24 Milliarden [Gar15]. Um sowohl vom Smart Device als auch vom Laptop oder Desktoprechner auf die eigenen Daten zugreifen zu können und um auch Freunden, Kollegen oder allen Internetteilnehmern Zugriff auf einen Teil dieser Daten zu ermöglichen, speichern immer mehr Privatpersonen und Unternehmen ihre Daten nicht nur auf den eigenen Geräten, sondern auch in der Cloud - auf Servern fremder Unternehmen. Beispielsweise kann man sowohl vom Smartphone als auch dem Laptop auf die selben E-Mails zugreifen, wenn diese an zentraler Stelle gespeichert

---

<sup>1</sup><https://www.apple.com/de/icloud/>

werden, was sich zunehmend großer Beliebtheit erfreut: Googles E-Mail-Dienst GMail<sup>2</sup> hatte 2012 nach eigenen Angaben 425 Millionen Nutzer [Lud12]. Im Mai 2015 waren es 900 Millionen, von denen 75% auch mit mobilen Endgeräten auf ihre Mails zugriffen [Lar15]. Auch Apples Online-Speicherdienst iCloud hatte nach eigenen Angaben 320 Millionen registrierte Nutzer im dritten Quartal 2013 [Kah13], die den Dienst nutzen, um Fotos oder ihren Kalender in Apples Cloud zu speichern, oder sogar ein vollständiges Backup ihrer Geräte. Auch das gemeinsame Arbeiten an Dokumenten wird durch die zentrale Speicherung ermöglicht, ohne dass diese beispielsweise per E-Mail hin- und hergeschickt werden müssen. Daher wird OneDrive von Microsoft auch gezielt für Unternehmen angeboten, als „zentraler Ort für die Unternehmens-IT“ und „alle wichtigen geschäftlichen Daten“ [Mic15a]. Auch Fitness- und Medizin-Apps, die auf Smart Devices laufen und zur Selbstüberwachung unterschiedlicher Kennzahlen im Gesundheitsbereich dienen, speichern häufig ihre Daten in der Cloud [LBBK15]. Beispiele sind Apps zur Überwachung der Ernährungsgewohnheiten, zur Analyse von Jogging-Strecken mit Verknüpfung von Zeit, Positionsdaten und Pulsfrequenz, oder Apps zur Unterstützung der korrekten Einnahme von Medikamenten. Sogar einige medizinische Implantate wie Herzschrittmacher oder Defibrillatoren übertragen ihre Messdaten per Telefoneinwahl über Festnetz oder Mobilfunk zu Servern von Unternehmen, die diese Daten dann Ärzten zur Ferndiagnose über das Internet bereitstellen [Med14].

Diese Beispiele zeigen, dass hunderte Millionen von Menschen Smartphones nutzen und unter anderem mit diesen Geräten erhobene und potentiell private Daten bei fremden Unternehmen speichern. Wann und wie diese Speicherung genau stattfindet, ob und wann die gespeicherten Daten wieder gelöscht werden und wer die Daten zu welchen Zwecken verarbeitet ist meist unklar. Teilweise ist dem Nutzer einer App nicht einmal bewusst, dass überhaupt Daten von der App an den Hersteller oder dessen Geschäftspartner übertragen werden. Hätte der Nutzer genauere Kenntnis über die Datenerfassung und -verarbeitung sowie Möglichkeiten, darauf Einfluss zu nehmen, ließe sich bereits die Menge der erfassten Daten verringern und somit auch das Missbrauchspotential. Die datenerfassenden Firmen haben oft kein großes Interesse daran, die persönlichen Daten ihrer Kunden zu schützen, weil damit nur ein geringes finanzielles Risiko verbunden ist. Teilweise basiert deren Geschäftsmodell sogar auf der Verarbeitung dieser Daten für eigene Zwecke. Google wertet beispielsweise die Daten von GMail aus, um Werbung zu verkaufen. Zusätzlich macht die zentrale Speicherung bei wenigen großen Anbietern diese Datensammlungen zu einem lohnenswerten Ziel für Kriminelle und Geheimdienste, da sie auf einen Schlag Zugriff auf eine große Menge Daten erhalten können.

Um dieser Problematik technisch zu begegnen, kann Verschlüsselung eingesetzt werden (siehe Abschnitt 2.2). Die Daten liegen in unverschlüsselter Form dann nur auf den Endgeräten wie Smart Devices, Laptops oder Desktoprechnern vor und werden verschlüsselt, bevor sie zu anderen Systemen übertragen oder beispielsweise in der Cloud gespeichert werden. Erst wenn die Daten wieder auf ein Endgerät übertragen wurden, werden sie wie-

---

<sup>2</sup><https://www.google.com/intl/de/mail/help/about.html>

der entschlüsselt. Dadurch sind die Daten unlesbar, wenn unberechtigte Personen auf dem Übertragungsweg oder in der Cloud Zugriff auf diese Daten erlangen. Dieses Verfahren wird als Ende-zu-Ende-Verschlüsselung bezeichnet (siehe Abschnitt 2.2.2). Praxisbeispiele sind die E-Mail-Verschlüsselungssysteme PGP<sup>3</sup> und S/MIME<sup>4</sup>, die Verschlüsselungsprotokolle Axolotl<sup>5</sup> und OTR [BGB04], der Cloudspeicher-Dienst Tresorit [Tre15] oder die Festplattenverschlüsselungs-Software VeraCrypt [IDR15] (siehe Kapitel 3).

Die Cloud wird nicht nur zum Speichern von Daten, sondern auch zum Durchführen von Berechnungen genutzt, wie z. B. für Analysen auf in der Cloud gespeicherten Daten. Die Analysen lassen sich mit interaktiven Kuchengrafiken, Balkendiagrammen, sich dynamisch verändernden Tabellen und vielen anderen Darstellungsformen visualisieren und im Browser anzeigen. Zur Analyse und Visualisierung von Unternehmenskennzahlen existieren beispielsweise verschiedene Plattformen, wie Microsofts Power BI<sup>6</sup> oder Oracles Analytics Cloud<sup>7</sup>. Diese Plattformen benötigen jedoch Daten in unverschlüsselter Form, wodurch kritische Unternehmensdaten gefährdet sein können. Inwiefern Daten verschlüsselt in der Cloud gespeichert und trotzdem analysiert werden können, ist daher Thema dieser Arbeit.

## 1.2 Aufgabenstellung

Im Rahmen dieser Diplomarbeit soll ein Lösungskonzept für einen Datencontainer entwickelt werden, der Daten verschlüsselt in der Cloud speichert und mit dem Analysen auf diesen verschlüsselten Daten in der Cloud durchgeführt werden können (siehe Kapitel 4). Der Datencontainer soll die Form einer Webanwendung haben und sowohl Webservices als auch Smart Devices als Clients unterstützen. Als Grundlage soll das Konzept des Secure Data Containers (SDC, Abschnitt 3.2) dienen, das die verschlüsselte Speicherung von Daten auf den Smart Devices selbst ermöglicht. Der SDC integriert sich in die Privacy Management Platform (PMP, Abschnitt 3.1), ein alternatives Berechtigungssystem für Android. Daher soll auch der zu entwickelnde Datencontainer in der Cloud zusammen mit der PMP genutzt werden können. Der Datencontainer sowie eine PMP-App, die diesen nutzt, sollen prototypisch implementiert werden.

Weitere Anforderungen an den Datencontainer sind die Nutzung durch mehrere Anwendungen mit jeweils mehreren Nutzern sowie der nutzer- und anwendungsübergreifende Datenaustausch. Es sollen also unterschiedliche Webservices und PMP-Apps, die jeweils von mehreren Anwendern genutzt werden, auf den Datencontainer zugreifen können, und es sollen sowohl zwischen verschiedenen Nutzern einer Anwendung als auch zwischen

---

<sup>3</sup><https://tools.ietf.org/html/rfc4880>

<sup>4</sup><https://tools.ietf.org/html/rfc5751>

<sup>5</sup><https://github.com/trevp/axolotl/wiki>

<sup>6</sup><https://powerbi.microsoft.com/de-de/features>

<sup>7</sup>[https://cloud.oracle.com/de\\_DE/analytics-cloud](https://cloud.oracle.com/de_DE/analytics-cloud)

verschiedenen Nutzern unterschiedlicher Anwendungen sowie zwischen verschiedenen Anwendungen eines Nutzers Daten ausgetauscht werden können. Bezüglich der Verschlüsselung ist gefordert, dass aus Sicherheitsgründen die gespeicherten Daten nur bei berechtigten Anfragen an den Datencontainer vorübergehend entschlüsselt werden sollen. Aus Performancegründen soll bei Lese-/Schreibvorgängen nicht der gesamte Datenbestand ent- bzw. verschlüsselt werden. Außerdem sollen die gespeicherten Daten unter Ausnutzung der Verschlüsselung sicher gelöscht werden können (siehe Abschnitt 2.3).

Dazu sollen bestehende Lösungen zum Schutz sensibler Daten für Smart Devices und Cloud Services analysiert werden (Kapitel 3) und abschließend soll der Datencontainer bezüglich verschiedener Angriffsszenarien evaluiert werden (Kapitel 6). Das folgende Kapitel gibt zunächst eine Einführung in die notwendigen Grundlagen.

## 2 Grundlagen

Die vom Datencontainer verwalteten Informationen sollen vor unberechtigten Zugriffen geschützt werden. Der Schutz von Informationen ist das Kernthema des Gebiets der Informationssicherheit. Um diesen Schutz genauer zu definieren, unterscheidet man unterschiedliche Schutzziele. In den letzten Jahrzehnten hat eine Erweiterung und immer feinere Unterteilung dieser Schutzziele stattgefunden. Bedner und Ackermann geben eine Übersicht dazu [BA10].

Historisch war Vertraulichkeit schon immer ein wichtiges Schutzziel. Sie gehört zusammen mit Integrität und Verfügbarkeit zur sogenannten „CIA-Triade“, bestehend aus *Confidentiality*, *Integrity* und *Availability*, welche als wesentliche Schutzziele der Informationssicherheit angesehen werden [Per08][SM15, S. 3]. Das Schutzziel der Vertraulichkeit (*Confidentiality*) ist erreicht, wenn die zu schützenden Informationen nur Befugten zugänglich sind. Das Schutzziel der Integrität (*Integrity*) kann folgendermaßen unterteilt werden: *Systemintegrität* bezeichnet die korrekte Funktionsweise des Systems. *Datenintegrität* ist erreicht, wenn die Daten vollständig und korrekt sind, sie also weder auf dem Übertragungsweg noch bei der Verarbeitung oder Speicherung beschädigt oder manipuliert wurden, ohne dass diese Veränderungen erkannt werden können. Die Verfügbarkeit (*Availability*) ist erreicht, wenn das informationstechnische System und die Daten jederzeit verwendet werden können. Da es auch bei hochverfügbaren Systemen gelegentlich zu Ausfällen kommt, wird die Verfügbarkeit häufig als Verhältnis von Ausfallzeit zu Gesamtzeit angegeben. In dieser Arbeit wird vor allem untersucht, wie die Vertraulichkeit der Daten geschützt werden kann. Zentrale Prinzipien zur Sicherstellung der Vertraulichkeit sind Verschlüsselung und Zugriffsschutz, welche im weiteren Verlauf dieses Kapitels genauer betrachtet werden.

### 2.1 Zugriffsschutz

Der Zugriffsschutz dient dazu, ausschließlich berechtigte Zugriffe auf Daten zu erlauben und vor unberechtigten Zugriffen zu schützen [SS94]. Dazu werden für verschiedene Entitäten wie z. B. Benutzer jeweils Zugriffsrechte festgelegt. Bei der Berechtigungsprüfung muss eine anfragende Entität zunächst ihre Echtheit nachweisen - sie muss sich gegenüber der prüfenden Entität *authentisieren*. War die Prüfung des Echtheitsnachweises erfolgreich, wird geprüft ob die Entität die benötigten Zugriffsrechte besitzt. Besitzt die Entität die benötigten

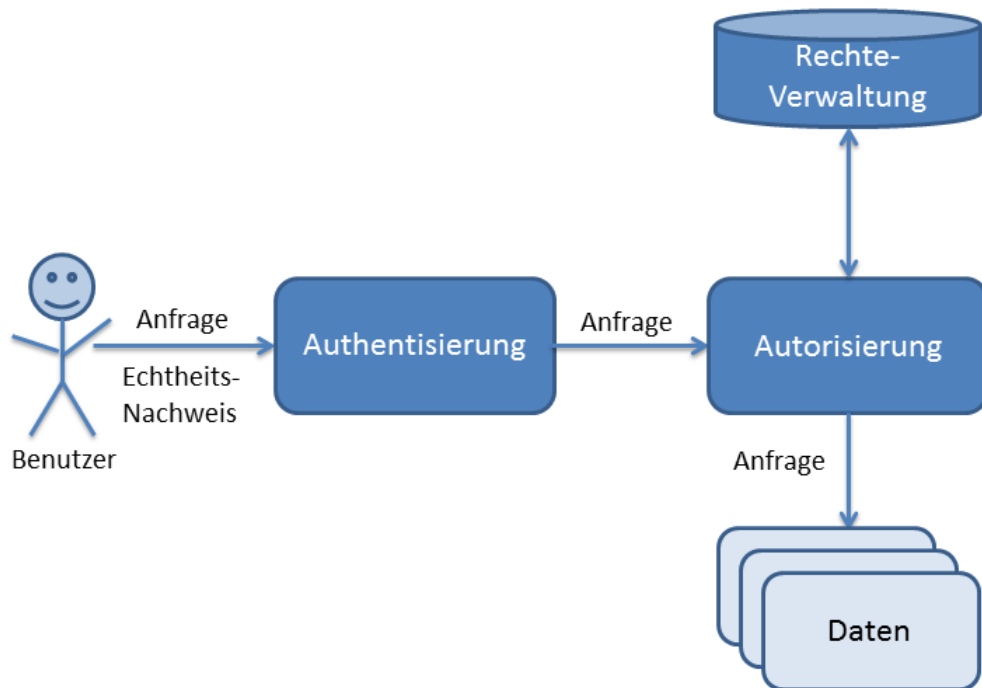


Abbildung 2.1: Zugriffsschutz

Zugriffsrechte so ist sie *autorisiert* - der Zugriff wird als berechtigt eingestuft und die Anfrage wird bearbeitet. Kann die anfragende Entität ihre Echtheit nicht nachweisen oder fehlen der Entität die erforderlichen Rechte, wird der Zugriff als unberechtigt eingestuft und die Anfrage abgelehnt.

*Beispiel:* Nehmen wir an, ein Benutzer möchte zugriffsgeschützte Daten lesen. Als Nachweis seiner Echtheit gibt der Benutzer am Zugriffsschutzsystem ein Passwort ein. Ist das Passwort korrekt, prüft das Zugriffsschutzsystem, ob der Benutzer ein Leserecht für die angefragten Daten besitzt. Ist dies der Fall, wird der Zugriff auf die Daten freigegeben. Bei falschem Passwort oder fehlendem Leserecht wird der Zugriff auf die Daten verweigert.

Für die genaue Definition der Rechte, deren Zuordnung zu den Entitäten sowie für Authentisierung und Autorisierung existieren viele verschiedene Ansätze, deren Details aber für diese Arbeit nicht weiter von Bedeutung sind. Im Lösungskonzept unterliegen Anfragen an den Datencontainer der Prüfung durch ein Zugriffsschutzsystem (siehe Kapitel 4) und im Prototyp wurde ein solches System implementiert (siehe Kapitel 5).

## 2.2 Verschlüsselung

Neben dem Zugriffsschutz ist die Verschlüsselung ein zentrales Prinzip zum Schutz der Vertraulichkeit von Daten. Sie ist Hauptthema der *Kryptographie*, welche als „die Lehre der Absicherung von Nachrichten durch Verschlüsselung“ verstanden wird [Ert12]. Als Verschlüsselung werden Verfahren bezeichnet, die Nachrichten unverständlich machen. Nur berechnete Empfänger sollen die unverständliche Nachricht wieder verständlich machen können, was als Entschlüsselung bezeichnet wird. Bei der Verschlüsselung wird eine verständliche Nachricht (der Klartext) mithilfe eines Schlüssels und einer Verschlüsselungsfunktion auf eine unverständliche Nachricht (den Geheimtext) abgebildet. Bei der Entschlüsselung wird der Geheimtext mithilfe eines Schlüssel und einer Entschlüsselungsfunktion wieder auf den ursprünglichen Klartext abgebildet. Wird zum Ver- und Entschlüsseln einer Nachricht derselbe Schlüssel verwendet, spricht man von einem symmetrischen Verschlüsselungsverfahren. Kommt bei der Verschlüsselung einer Nachricht ein anderer Schlüssel zum Einsatz, als bei ihrer Entschlüsselung, handelt es sich um ein asymmetrisches Verschlüsselungsverfahren. Ein Schlüssel kann ein Passwort, eine Zahl oder irgendeine andere Form von Information sein, die vom Verschlüsselungsverfahren unterstützt wird. Bei den heute eingesetzten Verfahren werden meistens Bitfolgen als Schlüssel verwendet.

Im Folgenden werden zunächst Kriterien zur Beurteilung der Sicherheit von kryptographischen Verfahren betrachtet, die für Designentscheidungen im Lösungskonzept (Kapitel 4) eine Rolle spielen. Das Prinzip der Ende-zu-Ende-Verschlüsselung, dessen Beachtung bzw. Vernachlässigung einen entscheidenden Einfluss auf den Schutz der Vertraulichkeit und die Analysemöglichkeiten hat, wird in Abschnitt 2.2.2 erläutert. Anschließend werden in Abschnitt 2.2.3 und Abschnitt 2.2.4 zwei Gruppen kryptographischer Verfahren kurz beschrieben, die für die Durchführung von Analysen bei der Nutzung von Ende-zu-Ende-Verschlüsselung zum Einsatz kommen können. Schließlich folgt eine Vorstellung zweier typischer Einsatzgebiete von Kryptographie, die im Lösungskonzept und teilweise in der Implementierung (Kapitel 5) Anwendung finden.

### 2.2.1 Sicherheitskriterien

Die Geheimhaltung des Schlüssels für die Entschlüsselung ist entscheidend für die Sicherheit eines Verschlüsselungsverfahrens. Dieser Schlüssel wird vor allem im Kontext asymmetrischer Verschlüsselungsverfahren auch als *privater Schlüssel* bezeichnet. Nach dem Kerckhoffs'schen Prinzip[Ker83] sollte die Sicherheit eines Verschlüsselungsverfahrens nicht von der Geheimhaltung des Algorithmus abhängen. Daher sind zentrale Fragen für den Schutz der Vertraulichkeit, wer Zugriff auf den Schlüssel hat und ob dessen Geheimhaltung sichergestellt werden kann.

Neben der Geheimhaltung des Schlüssels basiert die Sicherheit eines Verschlüsselungsverfahrens außerdem auf der Sicherheit der Ver- und Entschlüsselungsfunktionen. Mithilfe von Angreifermodellen und Sicherheitszielen kann die Sicherheit von Verschlüsselungsverfahren beurteilt werden. Werden Angreifermodell und Sicherheitsziel formal definiert, kann für einige Verschlüsselungsverfahren gezeigt oder widerlegt werden, ob ein Angreifer ein Sicherheitsziel verletzen kann.

Claude Shannon definierte 1949 das Sicherheitsziel der perfekten Geheimhaltung (*perfect secrecy*) [Sha49]. Erreicht ist dieses Ziel, wenn das Abhören des Geheimtextes dem Angreifer außer einer ungefähren Information über die Länge der Nachricht keinerlei Vorteil zur Berechnung des Klartextes bietet, sofern der private Schlüssel nicht bekannt ist. Shannon zeigte, dass das *One-Time Pad* Verschlüsselungssystem das Ziel der perfekten Geheimhaltung erfüllt. Wird das Ziel der perfekten Geheimhaltung dahingehend eingeschränkt, dass zur Berechnung des Klartextes nur polynomiell begrenzte Ressourcen zur Verfügung stehen, so spricht man von semantischer Sicherheit (*semantic security*) [GM84]. Die Ununterscheidbarkeit von Geheimtexten (*ciphertext indistinguishability*) ist erreicht, wenn ein Angreifer nicht entscheiden kann, welcher von zwei gleichlangen Geheimtexten einem von ihm bestimmten Klartext entspricht. Perfekte Geheimhaltung bzw. semantische Sicherheit und die Ununterscheidbarkeit von Geheimtexten sind äquivalent, je nach dem ob die Ressourcen des Angreifers polynomiell beschränkt sind oder nicht. Ciphertext indistinguishability kann daher als Kriterium herangezogen werden, um den Schutz der Vertraulichkeit durch ein kryptographisches System zu bewerten.

Zur Bewertung des Schutzes der Datenintegrität kann ein kryptographisches System bezüglich der Nichtverformbarkeit von Geheimtexten (*non-malleability*) untersucht werden. Wenn ein Angreifer einen Geheimtext nicht so verändern kann, dass die Klartextentsprechung des veränderten Geheimtextes in einer bestimmten Relation zum ursprünglichen Klartext steht, ist dieses Sicherheitsziel erreicht [DDN00].

### 2.2.2 Ende-zu-Ende-Verschlüsselung

Wenn Daten über das Internet übertragen und in der Cloud gespeichert werden, kann Verschlüsselung auf unterschiedliche Arten zum Einsatz kommen. Wir betrachten hier zwei Arten, die sich darin unterscheiden, ob Klartexte bzw. private Schlüssel in der Cloud vorliegen oder nicht: nämlich Verfahren die Ende-zu-Ende-Verschlüsselung einsetzen und solche, bei denen eine temporäre Entschlüsselung in der Cloud stattfindet. Bei Ende-zu-Ende-Verschlüsselung liegt der Klartext nur auf dem Endgerät, wie z. B. einem Smartphone, vor. Auf dem Übertragungsweg und bei Speicherung in der Cloud hingegen liegt ausschließlich der Geheimtext vor. Der private Schlüssel kommt lediglich auf dem Endgerät zum Einsatz und wird nicht in die Cloud übertragen. Bei anderen Ansätzen sind die Daten lediglich auf dem Transportweg zwischen Endgerät und Cloud verschlüsselt, oder werden z. B. in der Cloud auf verschlüsselten Festplatten gespeichert. Klartext und teilweise auch Schlüssel



liegen dabei aber in der Cloud vor, was bei Ende-zu-Ende-Verschlüsselung nicht der Fall ist.

### 2.2.3 Homomorphe Verschlüsselung

Lässt sich das verschlüsselte Ergebnis einer Funktion auf Klartexten berechnen, wenn diese ebenfalls nur in verschlüsselter Form vorliegen, so handelt es sich um ein homomorphes Verschlüsselungssystem. Sei beispielsweise die Funktion  $f(a, b) = c$  eine Funktion, die zwei Klartexte  $a$  und  $b$  auf den Klartext  $c$  abbildet. Sei außerdem  $g(x, y) = z$  ein Funktion, die zwei Geheimtexte  $x$  und  $y$  auf den Geheimtext  $z$  abbildet. Sei  $Encrypt$  die Verschlüsselungsfunktion und  $Encrypt(a) = x$ ,  $Encrypt(b) = y$  und  $Encrypt(c) = z$ . Dann kann  $c$  berechnet werden, indem  $a$  und  $b$  verschlüsselt und  $z$  entschlüsselt wird.

Existieren solche Homomorphieeigenschaften für alle  $n$ -stelligen Funktionen auf Klartexten, spricht man von einem voll-homomorphen Verschlüsselungssystem (*fully homomorphic encryption*). Craig Gentry beschrieb 2009 erstmalig die Konstruktion eines solchen Systems [Gen09]. Ist das System bezüglich der Funktionen eingeschränkt, handelt es sich um ein teil-homomorphes Verschlüsselungssystem (*partially homomorphic encryption*). Das Pallier-Kryptosystem [Pai99] ist beispielsweise teil-homomorph bezüglich der Addition. Homomorphe Verschlüsselungssysteme müssen das Sicherheitskriterium der non-malleability verletzen.

### 2.2.4 Deterministische Verschlüsselung

Bei deterministischen Verschlüsselungssystemen wird ein bestimmter Klartext immer auf denselben Geheimtext abgebildet. Dadurch kann entschieden werden, ob zwei Geheimtexte dem gleichen Klartext entsprechen, ohne Klartext oder privaten Schlüssel zu kennen. Kann dies analog für eine Ordnungsrelation entschieden werden, spricht man von einem ordnungserhaltenden Verschlüsselungssystem (*order-preserving encryption*). Deterministische und ordnungserhaltende Systeme müssen das Sicherheitskriterium der ciphertext indistinguishability verletzen und damit auch das der perfekten bzw. semantischen Sicherheit.

Blockchiffren im *Electronic Codebook Mode* sind z. B. deterministisch in diesem Sinne [Dwo01, S. 6.1]. Ein in CryptDB (Abschnitt 3.5) eingesetztes, ordnungserhaltendes System wurde von Boldyreva et al. entworfen [BCLO09].

### 2.2.5 Datenträgerverschlüsselung

Bei der Verschlüsselung von Datenträgern, Partitionen oder Imagedateien werden Daten vor dem Schreiben auf das Speichermedium verschlüsselt, und nach dem Lesen vom Me-

dium wieder entschlüsselt. Der Ver- bzw. Entschlüsselungsvorgang ist dabei transparent: Verschlüsselte Imagedateien, Partitionen oder ganze Datenträger werden vom Betriebssystem eingebunden und können genauso verwendet werden wie wenn sie nicht verschlüsselt wären. Der Schlüssel wird beim Einbinden des Mediums zur Verfügung gestellt und während der Benutzung im Arbeitsspeicher gehalten. Bei vollständig verschlüsselten Datenträgern wie Festplatten oder Solid State Drives kann die Ver- bzw. Entschlüsselung auch vom Datenträger-Controller durchgeführt werden. Dies wird auch als *hardware-based full disk encryption* oder *self-encrypting drive* bezeichnet [Cou11]. Der Schlüssel wird dabei vom Datenträger-Controller verwaltet und muss beim Start des Datenträgers freigeschaltet werden.

### 2.2.6 Datenbank-Verschlüsselung

Einige Datenbankmanagementsysteme (DBMS) bieten die Möglichkeit an, die zu verwaltenen Daten in verschlüsselter Form zu speichern [Mic15b] [Ora15]. Dazu können komplette Datenbanken oder nur einzelne Tabellen, Spalten oder Zellen verschlüsselt werden. Um Zugriff auf die verschlüsselten Daten zu erhalten, muss dem DBMS der Schlüssel zur Verfügung gestellt werden, oder die Ver- bzw. Entschlüsselungsoperation muss von einem getrennten Hardware-Sicherheitsmodul durchgeführt werden. Solche Hardware-Sicherheitsmodule ermöglichen Ver- und Entschlüsselung von Daten, ohne dass der Schlüssel im Arbeitsspeicher des Systems vorliegen muss, das die Daten verarbeitet. Hardware-Sicherheitsmodule sind in der Regel mit Maßnahmen zum Schutz und zur Erkennung von Manipulation ausgestattet, um den Aufwand und das Risiko für einen Angreifer erhöhen.

Ende-zu-Ende-Verschlüsselung ist grundsätzlich auch ohne Unterstützung von Seiten des DBMS möglich. Dazu werden die verschlüsselten Daten z. B. als uninterpretierte Bytes oder Strings gespeichert. Da das DBMS die Daten nicht entschlüsseln kann, sind keine sinnvollen Analysen durch das DBMS möglich. Bietet das DBMS oder eine DBMS-Erweiterung eine Unterstützung für entsprechende kryptographische Systeme, ist eine Analyse trotz Ende-zu-Ende-Verschlüsselung zu einem gewissen Grad möglich. So ermöglicht deterministische Verschlüsselung (Abschnitt 2.2.4) z. B. JOINS oder WHERE-Clauses mit Gleichheitskriterien. Mit einem additiv-homomorphen System lassen sich z. B. Summen über verschlüsselte Werte berechnen.

## 2.3 Sicheres Löschen

Unter Ausnutzung von Verschlüsselung lassen sich Daten *sicher löschen* [CHHS13]. Damit ist gemeint, dass die Daten nach dem Löschen auch mit forensischen Methoden nicht wiederhergestellt werden können. Sofern das eingesetzte kryptographische System bezüglich eines geeigneten Angreifermodells das Sicherheitsziel der semantischen Sicherheit erreicht, sind

die Daten ohne Schlüssel vollständig unlesbar. Daher lassen sie sich indirekt sicher löschen, indem der Schlüssel sicher gelöscht wird. Diese Indirektion lässt sich über Verschlüsseln des Schlüssels beliebig fortsetzen, bis schließlich der letzte Schlüssel in der Kette unter Ausnutzung der physikalischen Eigenschaften eines Speichermediums gelöscht werden muss. Bei beispielsweise magnetischen Speichermedien kann dies durch mehrfaches Überschreiben mit Zufallsdaten geschehen, bis die ursprüngliche Magnetisierung des Speicherbereichs, in dem der Schlüssel abgelegt war, nicht mehr rekonstruiert werden kann.

Da in der Cloud vor allem virtuelle Datenträger zum Einsatz kommen, deren physische Repräsentation unbekannt ist, kann sicheres Löschen auf Basis physikalischer Eigenschaften des Speichermediums meistens nicht durchgeführt werden. Daher muss unter Ausnutzung der Verschlüsselung sicher gelöscht werden.



## 3 Verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über Arbeiten, die sich mit Fragestellungen auseinandersetzen, die auch für diese Arbeit eine Rolle spielen. Die einzelnen Arbeiten und ihre Relevanz für die Aufgabenstellung werden kurz vorgestellt.

### 3.1 Privacy Management Platform

Die *Privacy Management Platform (PMP)* [SM14] [Sta13] ist ein alternatives Berechtigungssystem für Android, mit dem Zugriffsrechte für Apps einzeln zur Laufzeit verändert werden können und das Apps mit veränderten oder randomisierten Daten anstatt der Echtdaten versorgen kann. Die Zugriffsrechte für Apps können mit der PMP kontextsensitiv vergeben werden, gelten also bspw. nur an bestimmten Orten oder zu bestimmten Zeiten. Mit dem Standardberechtigungs-system von Android hingegen kann eine App nur genutzt werden, wenn ihr alle angeforderten Zugriffsrechte erteilt werden und es werden immer die Echtdaten an die App übermittelt. Sollen die Zugriffsrechte einer App über die PMP verwaltet werden, so muss dies bei der Entwicklung dieser App berücksichtigt werden. Dadurch kann der Nutzer von der PMP stets darüber informiert werden, welche Auswirkungen die Einstellungen der Zugriffsrechte auf den Funktionsumfang einer App haben und die App kann entsprechend auf eingeschränkte Rechte reagieren.

Das im Rahmen dieser Arbeit entwickelte Konzept eines *sicheren Datencontainers für die Cloud* berücksichtigt die Nutzung des Datencontainers durch Apps, die sich in die PMP integrieren. Im Rahmen des Prototyps wurde eine solche App auch implementiert.

### 3.2 Secure Data Container

Um die *Privacy Management Platform* zu einem „ganzheitlichen Datensicherheitssystem“ zu erweitern, haben Christoph Stach und Bernhard Mitschang den *Secure Data Container (SDC)* eingeführt[SM15]. Der SDC speichert Daten verschlüsselt auf dem Smart Device und integriert sich in die PMP. Für die Verschlüsselung wird das symmetrische AES-Verfahren und Schlüssel mit einer Länge von 256 Bit verwendet. Es können mehrere SDC-Instanzen

auf einem Smart Device existieren, wobei pro SDC-Instanz ein Schlüssel verwendet wird. Die Schlüssel werden auf dem Smart Device gespeichert und vom SDC verwaltet.

Neben der verschlüsselten Speicherung von Daten ermöglicht der SDC außerdem Datenaustausch zwischen Apps. Für jeden Datensatz ist im SDC eine Besitzer-App festgelegt, sowie optional weitere Apps, die ebenfalls Zugriff auf diesen Datensatz erhalten. Welche Apps Zugriff erhalten und welche nicht, kann von der Besitzer-App festgelegt werden.

Unter Ausnutzung der Verschlüsselung können vollständige SDC-Instanzen sicher gelöscht werden, indem der zugehörige Schlüssel sicher gelöscht wird. Das sichere Löschen des Schlüssels muss über ein Verfahren sichergestellt werden, das auf den physikalischen Eigenschaften des Speichermediums basiert (siehe Abschnitt 2.3).

Das in dieser Arbeit entwickelte Lösungskonzept unterstützt ebenfalls verschlüsselte Speicherung von Daten, Datenaustausch und sicheres Löschen. Da die Speicherung der Daten statt auf dem Smart Device aber in der Cloud erfolgt, unterscheidet sich das Lösungskonzept dieser Arbeit in einigen Punkten wesentlich von dem des SDC.

## 3.3 VeraCrypt

VeraCrypt [IDR15] ist eine Software zur Datenträgerverschlüsselung. Es werden Imagedateien, Partitionen, vollständige Datenträger wie Festplatten und Solid State Drives unterstützt, inklusive der Verschlüsselung von Datenträgern, auf denen ein Betriebssystem installiert ist. VeraCrypt unterstützt verschiedene symmetrische Verschlüsselungsverfahren (AES, Serpent, Twofish) sowie Keyfiles, Security Tokens und Smart Cards für erhöhte Sicherheit bei der Schlüsselverwaltung. In einem verschlüsselten Datenträger können mehrere Klartextdatenträger existieren, auf die mit jeweils einem eigenen Schlüssel zugegriffen wird. Abhängig vom eingesetzten Schlüssel wird jeweils ein anderer Bereich entschlüsselt, wobei es ohne Kenntnis aller Schlüssel unmöglich ist zu bestimmen, wie viele solcher Bereiche innerhalb eines verschlüsselten Datenträgers existieren. Diese Eigenschaft kann dazu genutzt werden, die Existenz gewisser Datenbereiche zu bestreiten.

VeraCrypt basiert auf dem TrueCrypt-Projekt, das nicht mehr weiterentwickelt wird. Der Quellcode beider Projekte ist öffentlich verfügbar und kann für Security Audits verwendet werden, wie dies beispielsweise im Rahmen des öffentlichen TrueCrypt Audits<sup>1</sup> bereits stattgefunden hat.

Mithilfe von VeraCrypt können Daten per Ende-zu-Ende-Verschlüsselung in der Cloud gespeichert werden. Dazu werden die Daten zuerst auf dem Endgerät in einer verschlüsselten Imagedatei gespeichert und anschließend wird die Imagedatei in die Cloud übertragen. Nun kann die Imagedatei wieder auf ein Endgerät übertragen und dort entschlüsselt werden.

---

<sup>1</sup><http://istruecryptauditedyet.com/>

VeraCrypt stellt eine Möglichkeit zur Verschlüsselung der Daten in der Cloud im Lösungskonzept dieser Arbeit dar.

### 3.4 Tresorit

Tresorit [Tre15] ist ein Cloud-Speicherdienst, der Ende-zu-Ende-Verschlüsselung verwendet. Dazu bietet Tresorit entsprechende Apps für unterschiedliche Desktop- und Mobilplattformen an. Neben dem verschlüsselten Speichern der Daten in der Cloud bietet Tresorit die Möglichkeit des Datenaustauschs mit anderen Nutzern inklusive feingranularer Zugriffsrechteverwaltung. Analysemöglichkeiten auf den gespeicherten Daten in der Cloud sind nicht vorgesehen. Es wird sogar explizit erwähnt, dass bei der verwendeten Verschlüsselung die Gleichheit von verschlüsselten Inhalten nicht festgestellt werden könne, das Sicherheitskriterium der ciphertext indistinguishability also erfüllt sei.

Tresorit hat mit dem Lösungskonzept dieser Arbeit das verschlüsselte Speichern von Daten in der Cloud, die Möglichkeit des Datenaustausches zwischen den Nutzern, die Unterstützung von Smart Devices als Nutzer und die Implementierung als Webanwendung gemeinsam.

### 3.5 CryptDB

Den Analysemöglichkeiten auf verschlüsselten Daten widmet sich das CryptDB-Projekt vom Massachusetts Institute of Technology [PRZB11]. Die Daten werden mit Ende-zu-Ende-Verschlüsselung in einer Datenbank gespeichert, die auch in der Cloud liegen kann. Die Daten werden dabei redundant mit unterschiedlichen Verschlüsselungsverfahren wie teilhomomorphen oder ordnungserhaltenden Verfahren verschlüsselt, die jeweils unterschiedliche Operationen auf den verschlüsselten Daten erlauben (siehe Abschnitt 2.2).

CryptDB führt SQL Queries auf verschlüsselten Daten aus. Das Design basiert vor allem auf *User Defined Functions (UDFs)* und wurde für MySQL und Postgres implementiert. Bei einer UDF handelt es sich um eine Erweiterungsfunktion für ein Datenbankmanagementsystem (DBMS), die im Falle von CryptDB eine SQL-Operation auf verschlüsselten Daten implementiert. Die Autoren gehen davon aus, dass CryptDB grundsätzlich mit jedem SQL-fähigen DBMS verwendet werden kann, das UDFs unterstützt.

CryptDB stellt eine Möglichkeit dar im Rahmen des Lösungskonzeptes Analysen auf verschlüsselten Daten trotz Ende-zu-Ende-Verschlüsselung durchzuführen.

## 3.6 Cipherbase

Cipherbase<sup>2</sup> [AEJ+14] ist ein Forschungsprojekt von Microsoft, das deren DBMS *SQL Server* um externe Hardware für Verschlüsselung und Schlüsselverwaltung erweitert. Die Daten werden verschlüsselt gespeichert und nur in der externen Hardware während der Bearbeitung von Datenbankabfragen entschlüsselt. Die im 2.2.6 erwähnten Hardware-Sicherheitsmodule werden dagegen lediglich für die Schlüsselverwaltung eingesetzt. Bei Cipherbase werden alle Teile der Datenbankabfragen, die Klartext benötigen, von der externen Hardware durchgeführt. Zwischen- und Endergebnisse von Datenbankabfragen werden nur verschlüsselt von der externen Hardware zurückgegeben und schließlich das Endergebnis verschlüsselt an den Nutzer weitergereicht. Es existiert eine Implementierung von Microsoft, bei der die externe Hardware mittels *Field Programmable Gate Arrays (FPGAs)* umgesetzt wurde.

Cipherbase und ähnliche DBMS können im Rahmen des Lösungskonzeptes eingesetzt werden.

## 3.7 Übersicht

Name	Fokus	Ungelöst
SDC	Verschlüsselung, Datenaustausch, sicheres Löschen	Cloud
VeraCrypt	Verschlüsselung, Datenaustausch	Analyse
Tresorit	Verschlüsselung, Datenaustausch, Cloud	Analyse, sicheres Löschen
CryptDB	Verschlüsselung, Analyse (eingeschränkt)	Client-Schnittstelle, Analyse (uneingeschränkt)
Cipherbase	Verschlüsselung, Analyse	Client-Schnittstelle, spezielle Hardware

---

<sup>2</sup><http://research.microsoft.com/en-us/projects/cipherbase/>



# 4 Lösungskonzept

In Kapitel 2 wurden die wissenschaftlichen und technischen Grundlagen erläutert, die zur Entwicklung des Datencontainers notwendig sind. Die in Kapitel 3 vorgestellten Arbeiten lösen zwar ähnliche Probleme, jedoch deckt keine der Arbeiten alle Anforderungen ab. In diesem Kapitel werden daher zunächst die in Kapitel 1 motivierten und beschriebenen Anforderungen an den sicheren Datencontainer für die Cloud konkretisiert. Anschließend wird deren Realisierung durch das entwickelte Lösungskonzept beschrieben und die getroffenen Entscheidungen werden begründet.

## 4.1 Anforderungen

Die Anforderungen an den Datencontainer werden in funktionale und nicht-funktionale Anforderungen unterteilt: Funktionale Anforderungen beziehen sich auf eine konkrete Funktion, die im Datencontainer umgesetzt werden muss. Nicht-funktionale Anforderungen definieren Qualität und Randbedingungen der umzusetzenden Funktionen.

### 4.1.1 Funktionale Anforderungen

- Der Datencontainer muss Daten von PMP-Apps und Webservices (Clients) entgegennehmen können und diese speichern.
- Die gespeicherten Daten müssen einem Client und einem Benutzer zugeordnet werden.
- Der Datencontainer muss die gespeicherten Daten dem Client/Benutzer-Paar zur Verfügung stellen können, das sie zum Datencontainer übertragen hat. Dieses Client/Benutzer-Paar wird als Besitzer der Daten bezeichnet.
- Der Datencontainer muss Analysen auf im Datencontainer gespeicherten Daten nach Vorgabe durch den Besitzer durchführen können und die Ergebnisse dem Besitzer zur Verfügung stellen.
- Der Datencontainer muss die gespeicherten Daten anderen Client/Benutzer-Paaren als dem Besitzer zur Verfügung stellen sowie Analysen nach deren Vorgabe durchführen und ihnen die Ergebnisse zur Verfügung stellen können.

- Besitzer müssen ihre gespeicherten Daten für andere Client/Benutzer-Paare freigeben und diese Freigabe widerrufen können.
- Besitzer müssen ihre im Datencontainer gespeicherten Daten vollständig löschen können.

### 4.1.2 Nicht-funktionale Anforderung

- Die Echtheit von Benutzern und Clients muss sichergestellt werden.
- Die Daten müssen im Datencontainer verschlüsselt gespeichert werden.
- Die gespeicherten Daten dürfen nur bei berechtigten Anfragen an den Datencontainer vorübergehend entschlüsselt werden.
- Bei Zugriffen auf die gespeicherten Daten darf nicht der gesamte Datenbestand aller Client/Benutzer-Paare ent- bzw. verschlüsselt werden.
- Die Kommunikation zwischen Clients und dem Datencontainer muss über einen sicheren Kanal erfolgen.
- Daten und Analyseergebnisse dürfen einem anderen Client/Benutzer-Paar als dem Besitzer nur zur Verfügung gestellt werden, wenn der Besitzer für dieses Client/Benutzer-Paar eine entsprechende Freigabe erteilt hat.
- Das vollständige Löschen der Daten durch den Besitzer muss durch sicheres Löschen unter Ausnutzung der Verschlüsselung erfolgen.

## 4.2 Architektur

Eine wesentliche Anforderung stellt das verschlüsselte Speichern von Daten sowie das Durchführen von Analysen auf diesen Daten dar. Daher kommt zur Verwaltung der Daten ein Datenbankmanagementsystem (DBMS) zum Einsatz. Sowohl Webservices als auch PMP-Apps (Clients) können den sicheren Datencontainer (Cloud-SDC) verwenden, der als Webanwendung realisiert ist. Die Kommunikation zwischen Clients und Cloud-SDC findet über einen sicheren Kanal statt, beispielsweise HTTPS. Der Cloud-SDC nimmt Anfragen von PMP-Apps und Webservices an und übergibt diese an das DBMS, welches die angefragten Daten ausliest, Analysen durchführt oder Änderungen an den Daten vornimmt. Das Ergebnis der Anfrage oder eine Fehlermeldung wird vom DBMS an den Cloud-SDC übermittelt, welcher dies wiederum an den Client weitergibt.

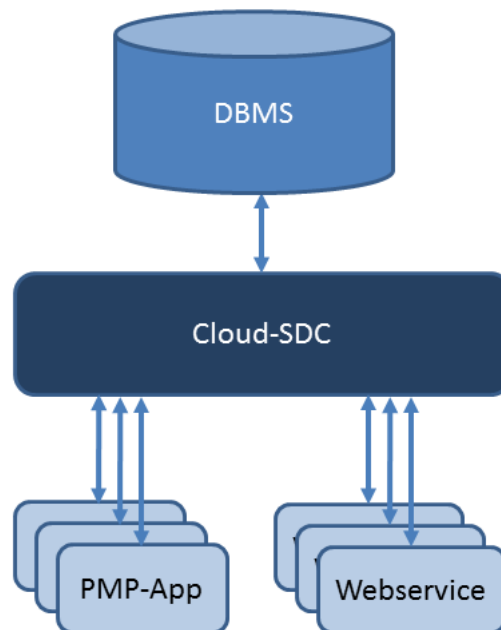


Abbildung 4.1: Architektur

## 4.3 Datenverwaltung

Jede PMP-App und jeder Webservice erhält pro Benutzer eine Datenbank, in der dieser Client die Daten des Benutzers speichern kann. Dieses Client/Benutzer-Paar ist der Besitzer der Datenbank. Die Wahl dieser Granularität wird im Abschnitt 4.5 erläutert und begründet.

Eine Anfrage an den Cloud-SDC enthält ein Client/Benutzer-Paar, dessen Datenbank verwendet werden soll. Die Syntax und Semantik der Anfragen zum Speichern oder Bereitstellen von Daten sowie zur Durchführung von Analysen ist durch das DBMS festgelegt. Der Cloud-SDC hat keine Kenntnis über Syntax oder Semantik der Anfragen oder der Ergebnisse. Die Anfrage zu generieren ist Aufgabe des Clients, diese zu interpretieren ist Aufgabe des DBMS. Das Ergebnis zu berechnen ist Aufgabe des DBMS, dieses zu interpretieren ist Aufgabe des Clients. Der Cloud-SDC weist das DBMS lediglich an, welche Datenbank für die Anfrage zu verwenden ist und eventuell welche Zugriffsrechte für die Anfrage zu berücksichtigen sind (siehe Abschnitt 4.4).

Dadurch können unterschiedliche DBMS zusammen mit dem Cloud-SDC eingesetzt werden, wobei keine Logik zur Interpretation der Anfragen oder Ergebnisse implementiert werden muss. Der Cloud-SDC muss das DBMS lediglich mit einer Eingabe in serialisierter Form versorgen, eine Ausgabe in serialisierter Form entgegennehmen, sowie die Datenbank auswählen und eventuelle Zugriffsrechte übergeben können. Auch der parallele Einsatz verschiedener DBMS mit einer Cloud-SDC-Instanz ist denkbar. Hätte der Cloud-SDC dagegen

eine eigene Abfragesprache oder eine anderweitig fest definierte Schnittstelle zum Lesen und Schreiben der Daten gegenüber den Clients, müsste diese Schnittstelle im Zweifelsfall für jedes DBMS eigens implementiert werden. Eine feste Schnittstelle könnte nur gemeinsame Basisfunktionalitäten unterschiedlicher DBMS unterstützen oder müsste erweiterbar sein. Die Erweiterungen wiederum müssten für jedes DBMS zusätzlich implementiert werden. Ist eine DBMS-unabhängige Schnittstelle gewünscht, so kann diese von einem Webservice oder über die PMP bereitgestellt werden.

### 4.4 Zugriffsschutz

Bei Anfragen an den Cloud-SDC muss zunächst die PMP-App bzw. der Webservice sowie der Benutzer authentisiert werden. Dazu müssen sowohl der Client als auch der Benutzer einen Echtheitsnachweis erbringen, der durch den Cloud-SDC geprüft wird (siehe Abschnitt 2.1). Bei erfolgreicher Authentisierung prüft der Cloud-SDC, ob die Zugriffsrechte zur Bearbeitung der Anfrage vorliegen. Die Zugriffsrechte werden vom Cloud-SDC pro Datenbank (siehe Abschnitt 4.3) verwaltet. Soll für eine Anfrage die Datenbank verwendet werden, dessen Besitzer das anfragende und authentifizierte Client/Benutzer-Paar ist, wird die Anfrage an das DBMS weitergeleitet, da der Besitzer uneingeschränkte Rechte auf seiner Datenbank hat. Soll die Datenbank eines anderen Client/Benutzer-Paars verwendet werden, prüft der Cloud-SDC, ob die Datenbank vom Besitzer für das anfragende Client/Benutzer-Paar freigegeben wurde. Ist dies der Fall, wird die Anfrage an das DBMS weitergeleitet, ansonsten wird die Anfrage vom Cloud-SDC verweigert. Zusätzlich zum grundlegenden Freigaberecht, kann der Cloud-SDC noch weitere Rechte pro Client/Nutzer-Paar bezüglich einer Datenbank speichern. Diese Rechte sind DBMS-spezifisch und werden vom Cloud-SDC nicht geprüft, sondern dem DBMS zur Überprüfung zusammen mit der Anfrage übergeben.

Der Besitzer verwaltet die Freigaben und eventuellen weiteren Rechte über Administrations-Anfragen an den Cloud-SDC. Aufgrund der Trennung von Benutzer und Client kann ein Besitzer eine Datenbank beispielsweise für all seine Apps oder die Datenbank eines von ihm genutzten Webservices für alle Benutzer freigeben. Gruppen, Hierarchien und ähnliche Strukturen von Benutzern und Clients zur Zugriffsrechteverwaltung können bei Bedarf implementiert werden.

### 4.5 Verschlüsselung

Auf den gespeicherten Daten sollen Analysen in der Cloud möglich sein. Mit Ende-zu-Ende-Verschlüsselung müssen Verschlüsselungssysteme zum Einsatz kommen, die Operationen auf Geheimtexten ohne Kenntnis des Schlüssels ermöglichen. Ohne Ende-zu-Ende-Verschlüsselung können die Daten zur Durchführung der Analyse in der Cloud entschlüsselt

werden, allerdings besteht dadurch die Gefahr des unberechtigten Zugriffs auf den Schlüssel in der Cloud. Da existierende Implementierungen vollhomomorpher Verschlüsselungsverfahren nicht performant genug für ein solches Einsatzszenario sind, kommen für Ende-zu-Ende-Verschlüsselung nur teilhomomorphe, deterministische und ordnungserhaltende Verfahren in Frage. Diese Verfahren schränken allerdings die Möglichkeiten bei der Analyse ein, da nur bestimmte Funktionen berechnet werden können. Vor allem deterministische und ordnungserhaltende Verfahren schwächen die Gewährleistung der Vertraulichkeit, da sie das Sicherheitskriterium der semantischen Sicherheit nicht erfüllen können.

Aufgrund dieses Dilemmas sieht das Lösungskonzept grundsätzlich eine Entschlüsselung in der Cloud vor, die aber durch Ende-zu-Ende-Verschlüsselung ergänzt werden kann, sofern nur eingeschränkte oder gar keine Analysen der Daten in der Cloud benötigt werden bzw. die Sicherheit der Daten im Vergleich zu potentiellen Analysemöglichkeiten überwiegt.

### 4.5.1 Temporäre Entschlüsselung in der Cloud

Zur Entschlüsselung der Daten zwecks Analyse in der Cloud muss der Schlüssel in der Cloud vorliegen. Die Daten werden verschlüsselt in der Cloud gespeichert, um sie vor unberechtigtem Zugriff zu schützen. Befindet sich nun der Schlüssel in der Cloud, ist der Schutz der Vertraulichkeit der verschlüsselten Daten gefährdet, da der Schlüssel wie die Daten selbst durch unberechtigte Zugriffe bedroht ist und die Sicherheit der Verschlüsselung vor allem auf der Geheimhaltung des Schlüssels beruht.

Um diese Gefahr zu verringern, wird in diesem Lösungskonzept der Schlüssel nicht in der Cloud persistiert, sondern bei jeder Anfrage vom Client an den Cloud-SDC übermittelt.

Unterstützt das eingesetzte DBMS die Verschlüsselung ganzer Datenbanken, übermittelt der Cloud-SDC den Schlüssel zusammen mit der zu verwendenden Datenbank, eventuellen Zugriffsrechten und der Abfrage an das DBMS. Das DBMS führt mithilfe des Schlüssels die Abfrage auf der verschlüsselten Datenbank durch. Jede Datenbank wird mit einem anderen Schlüssel verschlüsselt. Erhalten Unberechtigte Zugriff auf einen dieser Schlüssel, können damit nur die Daten eines Client/Benutzer-Paars entschlüsselt werden.

Bei einer größeren Granularität würde der Schutz vor unberechtigten Zugriffen auf die Daten anderer Nutzer/Apps/Webservices alleine auf den im Cloud-SDC vergebenen Zugriffsrechten beruhen. Kann ein Angreifer den Zugriffsschutz durch den Cloud-SDC umgehen, erhält er Zugriff auf alle Daten, die mit dem ihm bekannten Schlüssel verschlüsselt sind. Wären Daten anderer Nutzer/Apps/Webservices ebenfalls mit diesem Schlüssel verschlüsselt, wäre die Vertraulichkeit dieser Daten nicht mehr sichergestellt.

Eine feinere Granularität würde entweder eine entsprechende Unterstützung vom DBMS erfordern oder eine Unterstützung von Abfragen über mehrere Datenbanken, oder Einschränkungen bezüglich der unterstützten Abfragen bedeuten und nur einen vergleichsweise

geringen Zuwachs an Sicherheit mit sich bringen. DBMS wie *Microsoft SQL Server* [Mic15b] oder *Oracle Database* [Ora15] unterstützen die Verschlüsselung von vollständigen Datenbanken. Daher wurde aufgrund der Unterstützung durch existierende DBMS zugunsten der Analysemöglichkeiten eine Granularität von einem Schlüssel pro Datenbank gewählt.

Unterstützt das eingesetzte DBMS keine Verschlüsselung, werden aber Datenbanken durch disjunkte Mengen von Dateien repräsentiert, so können verschlüsselte Imagedateien verwendet werden. Dazu kommt eine Software zur Datenträgerverschlüsselung zum Einsatz. Der Cloud-SDC bindet mit dem vom Client übermittelten Schlüssel eine verschlüsselte Imagedatei ein, in der die Dateien der Datenbank liegen. Das DBMS verwendet die durch diese Dateien repräsentierte Datenbank zur Verarbeitung der Abfrage, genau so wie eine unverschlüsselte Datenbank. Die Verschlüsselung findet transparent auf Dateisystemebene statt. Durch Verwendung von Imagedateien besteht kein Unterschied der Granularität der Schlüssel zu verschlüsselungsfähigen DBMS, da ein Schlüssel pro Imagedatei verwendet wird und eine Imagedatei die Dateien einer Datenbank enthält.

Unterstützt das DBMS externe Hardware zur Durchführung von Datenbankabfragen auf verschlüsselten Daten (siehe 3.6), kann diese zum Einsatz kommen, sofern für jede Datenbank ein eigener Schlüssel im Hardwaremodul hinterlegt ist und dieser durch den vom Client übermittelten Schlüssel freigeschaltet werden muss. Dadurch werden keine Schlüssel im eigentlichen Sinne in der Cloud persistiert und die Granularität der Schlüssel bleibt erhalten.

### 4.5.2 Ende-zu-Ende-Verschlüsselung

Daten, deren Inhalt nicht für die Durchführung von Analysen benötigt wird, können auf dem Endgerät verschlüsselt und anschließend nach dem in 4.5.1 beschriebenen Verfahren in der Cloud gespeichert werden. Diese Daten werden dann letztendlich zweifach verschlüsselt: zuerst durch den Client auf dem Endgerät und anschließend durch das DBMS bzw. durch Datenträgerverschlüsselung in der Cloud, wobei eine Entschlüsselung der ersten Stufe erst wieder auf einem Endgerät stattfindet und der dazugehörige Schlüssel ebenfalls nur auf dem Endgerät vorliegt. Daher ist das Kriterium der Ende-zu-Ende-Verschlüsselung für diese Daten erfüllt. Dem DBMS liegen aber nur die verschlüsselten Inhalte vor, die ihm Rahmen von Abfragen nicht sinnvoll verwendet werden können.

Um trotz Ende-zu-Ende-Verschlüsselung noch Analysen auf den Daten durchführen zu können, muss das Verschlüsselungssystem die Berechnung der für diese Analysen benötigten Funktionen ohne Kenntnis des Schlüssels erlauben. Dazu können Systeme wie CryptDB eingesetzt werden - je nach Sicherheits- und Performance-Anforderungen sowie Unterstützung durch das DBMS entweder zusätzlich zum in 4.5.1 beschriebenen Verfahren oder alternativ dazu. Bei alternativem Einsatz würde die Ver- bzw. Entschlüsselung durch das DBMS oder die Datenträgerverschlüsselungssoftware entfallen, was Performancegewinne

bedeuten könnte. Allerdings würde dies beispielsweise bei Einsatz von deterministischen oder ordnungserhaltenden Verschlüsselungsverfahren auch eine Schwächung der Sicherheit bedeuten.

## 4.6 Datenaustausch

Um Daten mit anderen Client/Benutzer-Paaren austauschen zu können, müssen entsprechende Zugriffsrechte vergeben werden (siehe Abschnitt 4.4). Da jede Datenbank mit einem eigenen Schlüssel verschlüsselt ist und der Schlüssel dem Cloud-SDC jeweils nur für die Dauer einer Anfrage bekannt ist, muss der Schlüssel an die berechtigten Client/Benutzer-Paare übermittelt werden. Hierzu ist ein Ende-zu-Ende-verschlüsselter Übertragungskanal notwendig. Dieser kann entweder vom Cloud-SDC bereitgestellt werden oder der Schlüsselaustausch kann über Ende-zu-Ende-verschlüsselte E-Mail, Instant Messaging etc. durchgeführt werden.

Sollen alle Zugriffsrechte auf eine Datenbank einem Client/Benutzer-Paar wieder entzogen werden, so wird dies zuerst durch eine entsprechende Administrationsanfrage an den Cloud-SDC durchgeführt, der die Zugriffsrechte entsprechend setzt. Ab diesem Zeitpunkt werden zukünftige Anfragen von dem gesperrten Client/Benutzer-Paar für diese Datenbank vom Cloud-SDC abgelehnt. Zusätzlich muss die Datenbank mit einem neuen Schlüssel verschlüsselt, die bisherige verschlüsselte Datenbank sicher gelöscht (siehe Abschnitt 4.7) und der neue Schlüssel wieder an alle berechtigten Client/Benutzer-Paare verteilt werden. Dieser Schritt dient vor allem dem Schutz neuer/veränderter Daten, falls das gesperrte Client/Benutzer-Paar den Zugriffsschutz des Cloud-SDC umgehen kann. Ohne Neuverschlüsselung wäre sonst ein Zugriff auf die Daten weiterhin möglich, obwohl alle Zugriffsrechte entzogen wurden, da das gesperrte Client/Benutzer-Paar weiterhin im Besitz des Schlüssels ist.

## 4.7 Sicheres Löschen

Jede vom Cloud-SDC gespeicherte Datenbank kann sicher gelöscht werden, indem alle Kopien des zugehörigen Schlüssels sicher gelöscht werden und die Datenbank normal gelöscht wird. Dazu könnten alle Clients aufgefordert werden, einen bestimmten Schlüssel sicher zu löschen. Ist ein Client nicht erreichbar oder folgt er der Aufforderung nicht, könnte die Datenbank nicht sicher gelöscht werden. Dies gälte auch, wenn bspw. im Rahmen von Datensicherungen weitere Kopien der Schlüssel erstellt werden. Das sichere Löschen der Kopien müsste dann manuell durchgeführt werden, was voraussetzt, dass all diese Kopien auffindbar sind und nicht nur ein normales sondern ein sicheres Löschen überhaupt möglich ist. All diese Anforderungen in der Praxis umzusetzen erscheint schwierig, vor allem wenn man Beteiligte in Betracht zieht, die den Löschaufforderungen absichtlich nicht nachkommen.

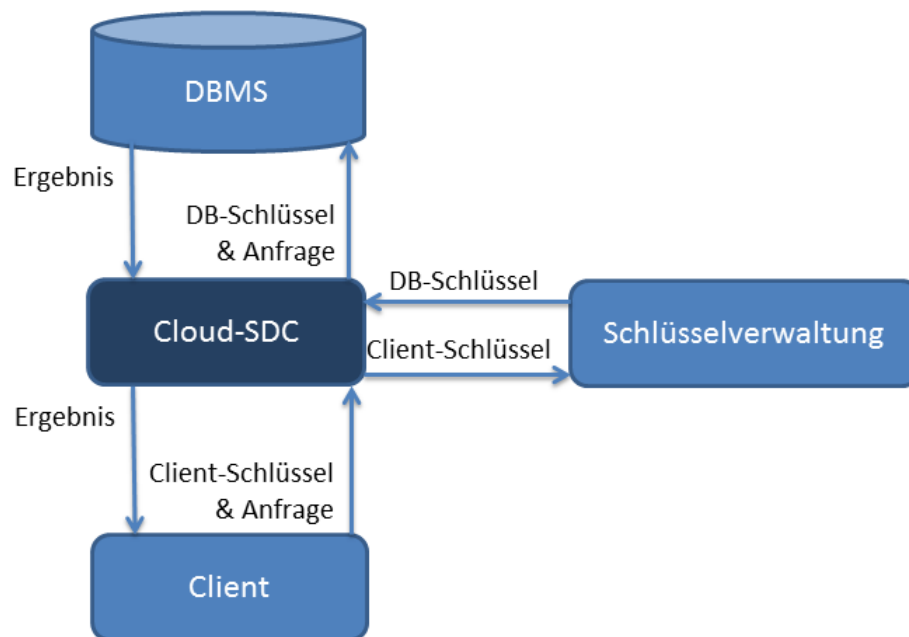


Abbildung 4.2: Schlüsselverwaltung

Daher sieht das Lösungskonzept eine zusätzliche Schlüsselverwaltung im Cloud-SDC vor. Der vom Client übermittelte Schlüssel (Client-Schlüssel) wird dazu genutzt, einen im Cloud-SDC verschlüsselt gespeicherten Schlüssel (DB-Schlüssel) zu entschlüsseln. Der DB-Schlüssel wird schließlich vom Cloud-SDC an das DBMS bzw. die Datenträgerverschlüsselungssoftware weitergereicht. Die verschlüsselten DB-Schlüssel müssen vom Cloud-SDC unter Ausnutzung des Speichermediums sicher gelöscht werden können, beispielsweise indem sie mit einem Hardware Sicherheitsmodul verwaltet werden. Ohne den Client-Schlüssel kann der DB-Schlüssel nicht zur Entschlüsselung der Datenbank genutzt werden, daher werden keine Schlüssel im eigentlichen Sinne in der Cloud persistiert. Soll nun eine Datenbank sicher gelöscht werden, wird der verschlüsselte DB-Schlüssel im Cloud-SDC sicher gelöscht. Die Client-Schlüssel werden dadurch nutzlos und sind für das sichere Löschen der Datenbank ohne Bedeutung.



# 5 Implementierung

Das in Kapitel 4 beschriebene Lösungskonzept wurde im Rahmen dieser Arbeit prototypisch implementiert. Die Implementierung besteht aus dem Datencontainer selbst und einer PMP-App. Im Folgenden wird die Implementierung genauer erläutert und Entscheidungen begründet.

## 5.1 Cloud-SDC

Eine wesentliche Entscheidung bei der Implementierung des Datencontainers stellte die Wahl der Cloud-Infrastruktur dar. Dabei fiel die Wahl auf das Platform-as-a-Service-Modell (PaaS), bei dem Laufzeitumgebung und API für Webanwendungen vom Cloud-Anbieter zur Verfügung gestellt wird. Beim Infrastructure-as-a-Service-Modell (IaaS) stellt der Cloud-Anbieter stattdessen virtuelle Maschinen bereit. Daher hätte bei IaaS zunächst ein Betriebssystem, ein Webserver und eine Laufzeitumgebung ausgewählt werden und anschließend installiert und konfiguriert werden müssen. Bei PaaS werden all diese administrativen Aufgaben vom Cloud-Anbieter übernommen, wodurch Ressourcen für die tatsächliche Implementierung frei werden.

### 5.1.1 Google App Engine

Neben dem Cloud-SDC wurde auch eine PMP-App unter Android prototypisch implementiert, was die Auswahl des PaaS-Anbieters wesentlich beeinflusst hat: Das Betriebssystem Android wird von *Google* und der *Open Handset Alliance* entwickelt [Goo15c]. *Google Play*<sup>1</sup>, die unter Android meistgenutzte Vertriebsplattform für Apps, wird von Google betrieben. Für die Nutzung von *Google Play* ist ein *Google-Konto*, erforderlich [Goo15b]. Google bietet eine PaaS-Umgebung für Webanwendungen an, die *Google App Engine* (GAE), welche kostenlos genutzt werden kann, solange die tatsächlich in Anspruch genommenen Ressourcen unterhalb einer bestimmten Grenze bleiben [Goo15d]. Sowohl Android als auch die GAE bieten eine API zur Benutzerverwaltung und Authentisierung über Google-Konten an. Aufgrund dieser engen Verbindung von Google und Android und der direkten Unterstützung von Google-Konten

---

<sup>1</sup><https://play.google.com/store>

für Authentisierung und Benutzerverwaltung fiel die Wahl des konkreten PaaS-Anbieters auf die GAE.

Die GAE bietet eine Laufzeitumgebung für die Programmiersprachen *Python*, *Java*, *PHP* und *Go* [Goo15a]. Für Entwicklungszwecke existiert eine lokal installierbare Version der GAE, welche von Google zur Verfügung gestellt wird. So können Programme zuerst lokal auf der Entwicklungsmaschine getestet und anschließend in die Cloud hochgeladen werden. Für die Implementierung des Cloud-SDC wurde die Sprache Python gewählt, da mit *PyCrypto*<sup>2</sup> eine umfangreiche Kryptographie-Bibliothek zur Verfügung steht, die offiziell von der GAE unterstützt wird.

## 5.2 Benutzerverwaltung

Da im Prototyp nur eine einzige App als Client existiert, wurde jedem Benutzer direkt eine Datenbank zugeordnet anstatt einer Datenbank pro Client/Benutzer-Paar, wie im Konzept beschrieben. Für die Unterstützung unterschiedlicher Clients müsste der Prototyp entsprechend angepasst werden, darauf wurde aufgrund mangelnder Testmöglichkeit jedoch verzichtet.

Die Authentisierung der Benutzer erfolgt über Google-Konten und die entsprechende API der GAE. Ein Benutzer wird über die Id des Google-Kontos eindeutig identifiziert. Jeder Id kann eine Datenbank zugeordnet werden. Diese Id identifiziert den Besitzer der Datenbank. Jeder Datenbank können wiederum 0 bis  $n$  Google-Konten-Ids zugeordnet werden, für die jeweils hinterlegt ist, ob ihnen der Zugriff auf die Datenbank vom Besitzer eingeräumt wurde.

## 5.3 DBMS

Bei Google App Engine und anderen PaaS-Anbietern werden keine verschlüsselungsfähigen DBMS angeboten. Es besteht außerdem keine Möglichkeit zur Verwendung einer Software zur Datenträgerverschlüsselung. Direkter Zugriff auf Dateien ist ebenfalls nicht möglich. Daher wurde ein eigenes DBMS entwickelt, das Verschlüsselung unterstützt.

Das DBMS nimmt einen kryptographischen Schlüssel sowie eine Abfrage entgegen und führt anschließend diese Abfrage auf verschlüsselten Daten durch. Da der Fokus des Lösungskonzeptes auf dem verschlüsselten Speichern der Daten liegt und die Verwaltung komplexer Datenstrukturen und umfangreiche Analysefunktionen für den Prototyp eine untergeordnete Rolle spielen, wurde im DBMS zugunsten einer einfacheren Implementierung die Komplexität der verwaltbaren Datenstrukturen sowie der Umfang der Analysemöglichkeit stark eingeschränkt. Eine von diesem DBMS verwaltete Datenbank besteht daher aus

---

<sup>2</sup><https://www.dlitz.net/software/pycrypto/>

einer Liste von Strings, dies stellt jedoch keine Einschränkung des Lösungskonzeptes dar, welches grundsätzlich unabhängig vom eingesetzten DBMS ist. Das implementierte DBMS erlaubt das Anhängen eines Strings an das Ende der Liste sowie das Entfernen eines Strings per Index. Außerdem können die Elemente der Stringliste abgerufen sowie die komplette Stringliste gelöscht werden. Das DBMS unterstützt auch eine Analysefunktion: Aus der Stringliste können alle Elemente herausgefiltert werden, die mit einem bestimmten Präfix beginnen.

Als Verschlüsselungsverfahren wird die AES-Implementierung der Bibliothek PyCrypto verwendet. Das DBMS entschlüsselt zuerst die gesamte Stringliste, führt anschließend die Abfrage aus und verschlüsselt die Stringliste anschließend wieder, sofern Änderungen durchgeführt wurden. Die verschlüsselte Stringliste wird im NDB Storage der Google App Engine gespeichert.

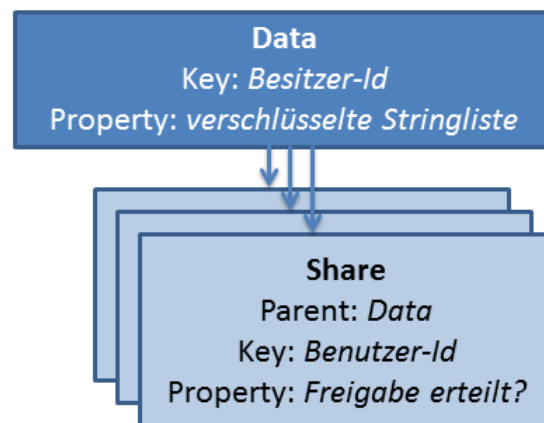


Abbildung 5.1: Models im NDB Storage

In NDB werden *Entities* gespeichert. Eine Entity kann mehrere *Properties* haben, die einen Datentyp und einen Wert haben. Entities werden über *Models* definiert, eine Art Schema. Jede Entity wird über einen eindeutigen *Key* identifiziert. Jeder Key kann einem übergeordneten Key zugeordnet werden. Dadurch lassen sich hierarchische Strukturen für Entities definieren.

Es wurde jeweils ein Model definiert, um eine Stringliste und eine Freigabe zu speichern (siehe Abschnitt 5.2). Der Key einer Stringlisten-Entity besteht aus der Id des Google-Kontos des Besitzers. Der Key einer Freigaben-Entity besteht aus der Id des Besitzers und aus der Id des Benutzers, für den die Freigabe gilt.

## 5.4 Schnittstellen

Das DBMS ist im Prototyp nicht als externes System realisiert sondern in den Cloud-SDC integriert. Die Kommunikation findet daher direkt über Funktions-Aufrufe in Python statt.

Die Anfragen der Clients werden per HTTPS an den Cloud-SDC übermittelt. Zur Bearbeitung der Anfragen wird die API *webapp2* der GAE verwendet. Als Format für die Anfragen an den Cloud-SDC wurde die HTTP POST Methode<sup>3</sup> und *URL-encoded form data* gewählt [Wor14]. Auf die gleiche Weise werden auch Inhalte von Webformularen vom Browser an den Webserver übertragen. Dadurch können die Methoden der Cloud-SDC-Schnittstelle mit einem Webformular über den Browser getestet werden, ohne dass die Formulardaten beispielsweise per JavaScript<sup>4</sup> vor dem Versenden an den Cloud-SDC konvertiert werden müssen. So konnte die Cloud-SDC-Schnittstelle bereits vor Implementierung der PMP-App getestet werden. Anpassungen an der Schnittstelle konnten ebenfalls zuerst über den Browser getestet und anschließend in der PMP-App umgesetzt werden. Der Cloud-SDC antwortet mit einem passenden HTTP-Statuscode<sup>5</sup>, um Erfolg oder Fehler an den Client zu melden. Es wurden folgende Methoden implementiert:

- `list`: Stellt dem Client die im Datencontainer gespeicherten Daten zur Verfügung.

Mit dem Parameter `crypto_key` wird der kryptographische Schlüssel in Form eines base64-encodierten Strings übergeben<sup>6</sup>.

Mit dem Parameter `user_id` wird als String die Id des Google-Kontos des Benutzers übergeben. Wird der Parameter leer gelassen, geht der Cloud-SDC davon aus, dass die Anfrage auf den Daten des Besitzers ausgeführt werden soll.

Mit dem Parameter `prefix` kann ein Filterpräfix für die Anfrage definiert werden. Dieser Parameter wird vom Cloud-SDC an das DBMS weitergereicht, welches die gespeicherten Daten entsprechend filtert.

Der Cloud-SDC reicht das Ergebnis des DBMS in Form einer Liste von Strings in Plaintext an den Client weiter. Die Elemente der Liste werden durch die Steuerzeichen *Carriage-Return* gefolgt von *Linefeed* getrennt.

- `add`: Fügt neue Daten hinzu.

Neben dem Parameter `crypto_key` (analog zu `list`) existiert für diese Methode der Parameter `item`, der die neuen Daten in Form eines Strings enthält. Es können nur Elemente zu den Daten des Besitzers hinzugefügt werden. Um auch Elemente

---

<sup>3</sup><https://tools.ietf.org/html/rfc7231#section-4.3.3>

<sup>4</sup><https://javascript.spec.whatwg.org/>

<sup>5</sup><https://tools.ietf.org/html/rfc7231#section-6>

<sup>6</sup><https://tools.ietf.org/html/rfc4648>

zu Daten anderer Benutzer hinzufügen zu können, müsste diese Methode analog zu `list` um den Parameter `user_id` erweitert werden.

- `remove`: Entfernt ein bestimmtes Element aus den im Datencontainer gespeicherten Daten.

Dazu wird neben `crypto_key` der Parameter `index` übergeben, der den null-basierten Index des zu entfernenden Elements enthält. Um Elemente aus Daten anderer Benutzer als des Besitzers entfernen zu können, müsste diese Methode wie `add` um den Parameter `user_id` erweitert werden.

- `delete`: Löscht alle im Datencontainer gespeicherten Daten.

Diese Methode hat keine Parameter, könnte aber zum Löschen anderer Daten als denen des Besitzers um den Parameter `user_id` erweitert werden.

- `share`: Gibt die Daten des Besitzers für einen anderen Benutzer frei oder hebt eine Freigabe auf.

Der Parameter `user_id` enthält die Id des Google-Kontos für das die Freigabe geändert werden soll.

Der Parameter `list_access` gibt an, ob das Konto die Methode `list` aufrufen darf (`True`) oder nicht (`False`). Für eine Erweiterung der Methoden `add`, `remove` und `delete` auf andere Benutzer als den Besitzer könnten analog jeweils eigene Zugriffsrechte eingeführt werden.

## 5.5 PMP-App

Als Client wurde eine Android App implementiert, die sich in die PMP integriert. Die App bietet eine graphische Oberfläche zum Anzeigen und Verändern einer Stringliste. Der Zugriff auf den Cloud-SDC findet über eine *PMP-Ressource* statt: Die App meldet sich bei der PMP an und fragt nach Zugriff auf die Cloud-SDC-Ressource. Wurde der App über die PMP das Recht eingeräumt, diese Ressource zu benutzen, kann ein Zugriff auf den Cloud-SDC erfolgen, ansonsten nicht. Außerdem müssen auf dem Android-Gerät die Zugangsdaten eines Google Accounts hinterlegt sein, mit dem sich die App gegenüber dem Cloud-SDC authentisieren kann.

Für die Kommunikation mit der GAE von Android-Apps aus werden von Google verschiedene Frameworks angeboten. Diese erschienen für den Prototyp jedoch überdimensioniert. Ein Testen der Cloud-SDC-Schnittstelle über Webformulare wäre mit diesen Frameworks nicht möglich gewesen (siehe Abschnitt 5.4). Die Kommunikation der PMP-Ressource mit der GAE wurde daher direkt über HTTP-Anfragen implementiert. Zur direkten Authentisierung gegenüber der GAE mit Google-Konten auf Basis von HTTP-Anfragen wird von Google

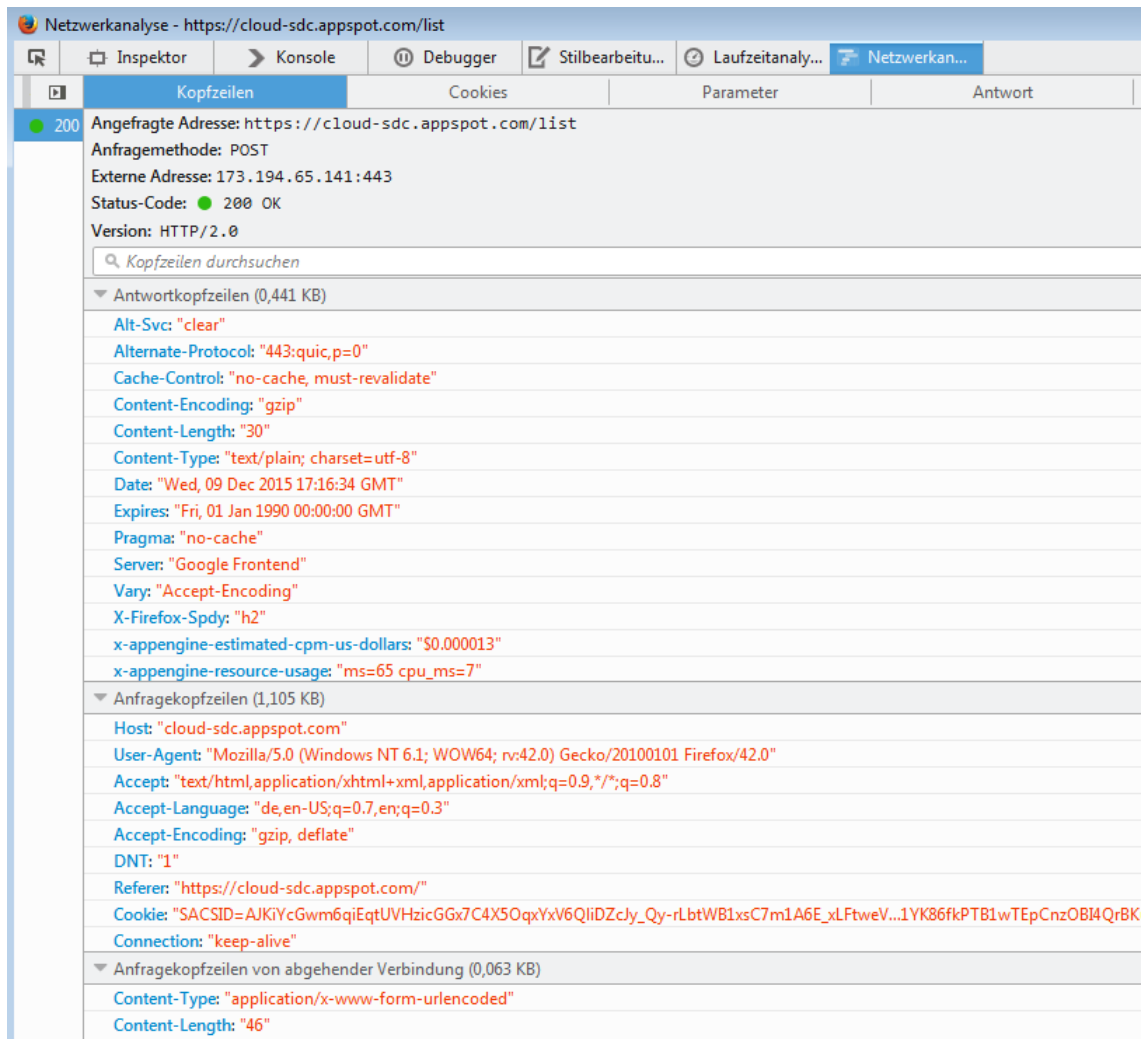


Abbildung 5.2: Netzwerkanalyse in Firefox. Per Webformular wird die Methode `list` aufgerufen. In den Kopfzeilen ist der Cookie zur Authentisierung zu sehen, sowie Content-Type von Anfrage und Antwort.

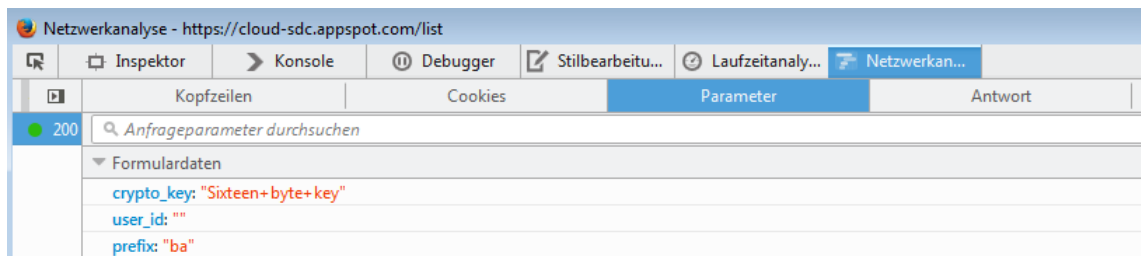
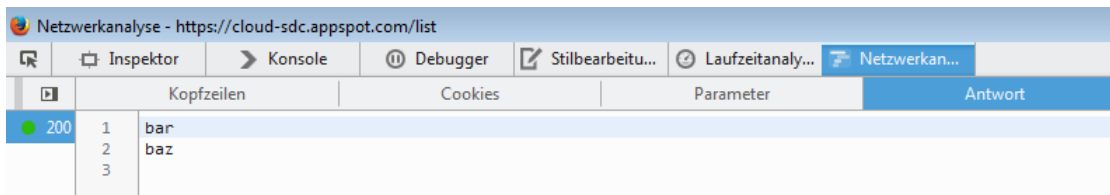


Abbildung 5.3: Übergabe der Parameter der Methode `list`.



The screenshot shows a network analysis tool interface with a status bar at the top displaying '200'. Below the status bar, there are four tabs: 'Kopfzeilen', 'Cookies', 'Parameter', and 'Antwort'. The 'Antwort' tab is selected and shows a response body with three lines of text: 'bar', 'baz', and 'baz'.

Status	Line	Content
200	1	bar
	2	baz
	3	baz

Abbildung 5.4: Antwort des DBMS: Alle gespeicherten Strings, die mit „ba“ beginnen.

jedoch keine Dokumentation angeboten. Der Google-Entwickler Nick Johnson weist jedoch auf ein Python-Skript hin, welches im Rahmen der lokalen GAE-Version ausgeliefert wird und sich zum Hochladen von GAE-Programmen in die Cloud per Google-Konto gegenüber der GAE authentisiert [Joh08]. Mithilfe des Quellcodes dieses Python-Skripts wurde die Authentisierung gegenüber der GAE in der PMP-Ressource implementiert: Zuerst wird über `android.accounts.AccountManager` das `access token` für das Google-Konto abgefragt, welches auf dem Android-Gerät hinterlegt ist. Dieses Token hat die Form eines Strings und wird per HTTP an die GAE übertragen, welche als Antwort ein Session-Cookie generiert. Durch Übergeben dieses Cookies authentisiert sich die PMP-Ressource gegenüber der GAE bei Anfragen an den Cloud-SDC. Ein ähnlicher Authentisierungsmechanismus kommt zum Einsatz, wenn Anfragen an ein GAE-Programm über den Browser stattfinden, beispielsweise beim Test der Cloud-SDC-Schnittstelle über ein Webformular. Hier wird der Cookie durch Übertragen von Benutzername und Passwort anstatt des Tokens generiert. Im Gegensatz zur PMP-Ressource muss jedoch kein Code geschrieben werden, da die entsprechenden Webseiten von der GAE generiert werden.





# 6 Evaluation

Dieses Kapitel beschreibt verschiedene Angriffsszenarien und beurteilt, inwieweit das Lösungskonzept diesen Angriffen standhält.

## 6.1 Angreifer

Angreifer sind Personen die das Ziel haben, im Datencontainer gespeicherte Daten unberechtigt zu lesen, zu verändern oder weiterzugeben, oder Infrastruktur des Datencontainers für ihre Zwecke zu missbrauchen. Zur Erreichung ihrer Ziele nutzen sie eine bestimmte Vorgehensweise, die als Angriff bezeichnet wird. Angreifer lassen sich in drei Gruppen unterteilen:

- Administratoren, die ihre Privilegien missbrauchen
- Cloud-SDC-Nutzer, Apps und Webservices, die den eingeschränkten Zugang zum Cloud-SDC missbrauchen
- Sonstige Internetteilnehmer

### 6.1.1 Administratoren

Um IT-Systeme zu betreiben, werden Administratoren benötigt, die privilegierten Zugang zu diesen Systemen erhalten, um sie initial einzurichten und zu warten. Im Rahmen des Lösungskonzepts werden verschiedene Administratoren benötigt, die für die Evaluation in vier Bereiche eingeteilt werden. Administratoren für den Cloud-SDC haben Zugriff auf den Webserver und können die Benutzer des Cloud-SDC und deren Rechte verwalten. Datenbank-Administratoren haben Zugriff auf das DBMS und dessen Rechteverwaltung sowie die Datenbanken. Betriebssystem-Administratoren haben Zugriff auf die Betriebssysteme, auf denen der Cloud-SDC oder das DBMS laufen. Hardware-Administratoren haben physischen Zugriff auf Rechner und Festplatten.

### **6.1.2 Cloud-SDC-Nutzer**

Cloud-SDC-Nutzer haben mittels Apps und Webservices eingeschränkten Zugriff auf den Cloud-SDC im Rahmen der vom Zugriffsschutzsystem gewährten Rechte. Sie sind als einzige im Besitz von kryptographischen Schlüsseln zur Entschlüsselung der im Datencontainer gespeicherten Daten.

### **6.1.3 Sonstige Angreifer**

Sonstige Internetteilnehmer können sich zwar mit dem Cloud-SDC verbinden, sofern dieser öffentlich im Internet erreichbar ist, haben aber kein Benutzerkonto und können den Cloud-SDC somit nicht verwenden. Einige von ihnen können aber unter Umständen die Kommunikation von Cloud-SDC-Nutzern oder Administratoren mitlesen oder verändern, wenn sie entsprechenden Zugriff auf beteiligte Netzwerkkomponenten haben.

## **6.2 Angriffsszenarien**

Unberechtigter Zugriff auf im Datencontainer gespeicherte Inhalte kann durch die Weitergabe von Daten und kryptographischen Schlüsseln durch berechtigte Personen stattfinden, sowie durch die Manipulation von Software und Hardware. Im Folgenden werden einige Angriffsszenarien genauer untersucht.

### **6.2.1 Angriffe auf Clients**

Die Besitzer der im Datencontainer gespeicherten Daten können diese an andere Personen weitergeben. Das gilt auch für die Zugangsdaten ihres Benutzerkontos und die kryptographischen Schlüssel. Geschieht dies absichtlich, handelt es sich um einen berechtigten Zugriff. Findet die Weitergabe jedoch ohne Einverständnis des Benutzers durch die App oder den Webservice statt, stellt dies einen unberechtigten Zugriff dar. Werden nur kryptographische Schlüssel weitergegeben, verhindert das Zugriffsschutzsystem des Cloud-SDC, dass diese zur Entschlüsselung der Daten eingesetzt werden können. Werden auch die Zugangsdaten zum Cloud-SDC von App oder Webservice unerlaubt weitergegeben, oder kann das Zugriffsschutzsystem umgangen werden, so kann dadurch auch ein unberechtigter Zugriff auf die verschlüsselten Daten stattfinden und diese können mit dem passenden Schlüssel lesbar gemacht werden. Da jedoch pro App/Webservice für jeden Nutzer ein eigener Schlüssel eingesetzt wird, kann auf diesem Weg kein Zugriff auf Daten anderer Apps/Webservices stattfinden.

Bei PMP-Apps werden die Schlüssel von der PMP verwaltet und können daher nicht unerlaubt durch Apps weitergegeben werden. Eine unerlaubte Weitergabe von Daten ist jedoch möglich. Schadsoftware kann durch Umgehung der Sicherheitsmechanismen des Betriebssystems oder durch Ausnutzung von Programmierfehlern in den Apps Zugriff auf Daten und Schlüssel erhalten. Auch durch Hardwaremanipulation ist ein solcher Zugriff denkbar.

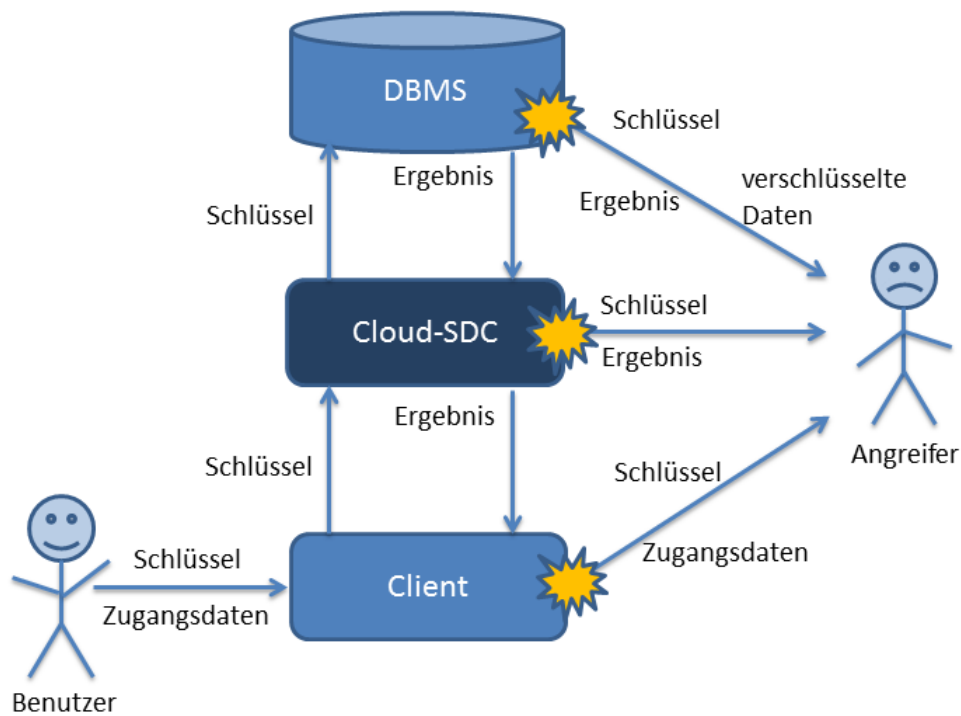


Abbildung 6.1: Angriffsszenarien

### 6.2.2 Angriffe auf den Cloud-SDC

Wie die Clients können auch die Komponenten des Datencontainers selbst kompromittiert werden. Durch Ausnutzung von Programmierfehlern oder Manipulation der zuständigen Administratoren kann das Zugriffsschutzsystem des Cloud-SDC umgangen werden. Dadurch kann z. B. eine Weiterleitung von Anfragen an das DBMS stattfinden, die Daten betreffen, zu denen der anfragende Benutzer nicht die entsprechenden Rechte besitzt. Auch Anfragen ohne vorherige Authentisierung können dann weitergeleitet werden. Solange nicht der passende kryptographische Schlüssel übermittelt wird, führen diese Anfragen jedoch zu Fehlern im DBMS. Der Schlüssel kann allerdings durch Kompromittierung des Cloud-SDC mitgelesen werden, wenn er bei berechtigten Anfragen übertragen wird. Ein Mitlesen der übertragenen

Schlüssel kann auch durch detaillierte Protokolle des Webservers oder dessen Kompromittierung über Programmierfehler erreicht werden. Ein Cloud-SDC-Administrator kann die Protokoll-Einstellungen des Webservers anpassen oder den Cloud-SDC selbst verändern. Ein Betriebssystem-Administrator kann dies ebenfalls tun, da er über den uneingeschränkten Zugriff auf das Betriebssystem auch Zugriff auf den Webserver und den Cloud-SDC hat. Aufgrund der Granularität muss der Schlüssel allerdings für jeden App/Webservice/Nutzer bei einer berechtigten Anfrage kopiert werden, um anschließend eine erfolgreiche unberechtigte Anfrage an das DBMS schicken zu können. Bei Ende-zu-Ende-Verschlüsselung ist ein Zugriff auf die im Datencontainer gespeicherten Inhalte auf diesem Weg nicht möglich, da das DBMS nur verschlüsselte Daten an den Cloud-SDC weitergibt und die Schlüssel ausschließlich auf den Clients vorliegen.

### 6.2.3 Angriffe auf das DBMS

Da das DBMS nicht direkt mit dem Internet verbunden ist, muss zuerst ein anderes System kompromittiert werden, das eine Verbindung zum DBMS hat, um unerlaubten Zugriff vom Internet aus zu erlangen. Es wäre z. B. eine Kompromittierung des DBMS über vorherige Kompromittierung des Cloud-SDC denkbar. Durch Manipulation des DBMS können die Schlüssel mitgelesen werden, wenn sie vom Cloud-SDC übergeben werden. Die verschlüsselten Daten können ebenfalls durch Manipulation des DBMS oder über Dateizugriff kopiert werden. Beides kann durch Ausnutzung von Programmierfehlern oder durch den Betriebssystem-Administrator stattfinden. Der Datenbank-Administrator sollte dazu nicht in der Lage sein, sofern er keine Schwachstellen ausnutzt. Neben den Schlüsseln können Daten im Klartext mitgelesen werden, wenn sie vom DBMS an den Cloud-SDC übergeben werden. Zusammen mit dem passenden kryptographischen Schlüssel können beliebige Abfragen im DBMS durchgeführt werden. Der Umfang der Datenzugriffe ist auch hier durch die Granularität der Schlüssel beschränkt. Bei Ende-zu-Ende-Verschlüsselung ist kein Zugriff auf Inhalte im Klartext durch eine Kompromittierung des DBMS möglich, da die Schlüssel nur auf den Clients vorliegen.

### 6.2.4 Übersicht

Angriffsszenario	Gegenmaßnahme	Problem
Client gibt Zugangsdaten weiter	Android verwaltet Zugangsdaten	Android kompromittiert
Client gibt Schlüssel weiter	PMP verwaltet Schlüssel	PMP kompromittiert
Cloud-SDC/DBMS kompromittiert	Schlüssel nicht persistiert	Angreifer liest Anfragen mit
Cloud-SDC/DBMS kompromittiert	Feine Schlüsselgranularität	Angreifer liest viele Anfragen mit
Cloud-SDC/DBMS kompromittiert	Ende-zu-Ende-Verschlüsselung	-



## 7 Zusammenfassung & Ausblick

Mit zunehmender Erfassung von Daten und deren Speicherung in der Cloud steigt auch das Missbrauchspotential dieser Daten. Um dieses Missbrauchspotential zu verringern, können die Daten verschlüsselt werden. Dadurch sind die Inhalte nur noch zusammen mit dem passenden kryptographischen Schlüssel verständlich. Die Sicherheit der Verschlüsselung hängt wesentlich von der Geheimhaltung der Schlüssel ab. Bei der zentralen Speicherung von kryptographischen Schlüsseln in der Cloud kann diese Geheimhaltung kaum sichergestellt werden. Bei Einsatz von Ende-zu-Ende-Verschlüsselung liegen die Schlüssel dezentral auf den Endgeräten vor, was deren Geheimhaltung vereinfacht. Allerdings ist das Durchführen von Analysen auf den gespeicherten Daten in der Cloud nur noch mithilfe spezieller Verschlüsselungssysteme möglich, was die Analysemöglichkeit stark reduziert und teilweise die Sicherheit der Verschlüsselung schwächt. Daher setzen bestehende Systeme zum Speichern von Daten in der Cloud entweder keine Verschlüsselung ein oder speichern ihre Schlüssel ebenfalls in der Cloud. Einige setzen Ende-zu-Ende-Verschlüsselung ein, eine Analyse der Daten ist dann allerdings nicht möglich.

Aufgrund der Konkurrenz von Analysemöglichkeiten und Datensicherheit führt das entwickelte Lösungskonzept eine temporäre Entschlüsselung in der Cloud durch: die Schlüssel werden bei jeder Anfrage über einen sicheren Kanal an den Datencontainer übertragen und nicht dauerhaft in der Cloud gespeichert. So stehen alle Analysemöglichkeiten zur Verfügung und die Gefahr des unberechtigten Schlüsselzugriffs wird verringert. Für Daten mit höherem Schutzbedarf und niedrigeren Anforderungen an die Analysemöglichkeiten kann zusätzlich Ende-zu-Ende-Verschlüsselung eingesetzt werden.

Je mehr Daten durch einen Schlüssel entschlüsselt werden können, umso entscheidender ist dessen Geheimhaltung. Daher wurde eine möglichst feine Schlüsselgranularität gewählt. Um das Missbrauchspotential weiter zu reduzieren, wird die Verschlüsselung durch ein Zugriffsschutzsystem ergänzt. Der Datencontainer erlaubt einen gezielten Datenaustausch, der durch das Zugriffsschutzsystem und die Verschlüsselung kontrolliert wird. Unter Ausnutzung der Verschlüsselung können die gespeicherten Daten sicher gelöscht werden.

Bei der prototypischen Implementierung des Lösungskonzepts kam auf Clientseite mit der Privacy Management Platform ein alternatives Berechtigungssystem für Android zum Einsatz, um das Missbrauchspotential auch auf den Clients zu verringern. Eine Implementierung für den Einsatz in der Praxis sowie Untersuchungen zu Performance und Skalierbarkeit könnten Inhalt zukünftiger Arbeiten sein.

Die Evaluation verschiedener Angriffsszenarien zeigt die Vorteile der feinen Schlüsselgranularität und des Verzichts auf dauerhafte Speicherung der Schlüssel in der Cloud. Dennoch wird deutlich, dass nur Ende-zu-Ende-Verschlüsselung die Geheimhaltung der Schlüssel bei Angriffen in der Cloud garantieren kann. Mit performanten, vollhomomorphen Verschlüsselungssystemen könnte trotz Ende-zu-Ende-Verschlüsselung auch eine uneingeschränkte Analyse in der Cloud durchgeführt werden. Solche Systeme zu entwickeln, könnte Ziel zukünftiger Forschung sein.

Neben der Verschlüsselung von Daten und der Weiterentwicklung von kryptographischen Systemen stellt die Datensparsamkeit einen entscheidenden Ansatz zur Verhinderung von Datenmissbrauch dar. Wenn nur benötigte Daten erfasst, soweit wie möglich anonymisiert und so früh wie möglich wieder gelöscht werden, sinkt dadurch auch das Missbrauchspotential. Dieser Grundsatz findet sich auch in §3a des Bundesdatenschutzgesetzes [Bun09]. Die Datensparsamkeit weltweit durch Gesetze und Abkommen zu stärken, sowie Verstöße zu sanktionieren, ist Aufgabe von Politik und Gesellschaft.



# Literatur

- [AEJ+14] A. Arasu, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann und R. Ramamurthy. *Transaction Processing on Confidential Data using Cipherbase*. Techn. Ber. MSR-TR-2014-106. Aug. 2014 (siehe S. 24).
- [BA10] M. Bedner und T. Ackermann. „Schutzziele der IT-Sicherheit“. In: *Datenschutz und Datensicherheit - DuD* 34.5 (2010), S. 323–328. ISSN: 1614-0702. DOI: 10.1007/s11623-010-0096-1 (siehe S. 13).
- [BCLO09] A. Boldyreva, N. Chenette, Y. Lee und A. O’Neill. „Order-Preserving Symmetric Encryption“. In: *Advances in Cryptology - EUROCRYPT 2009*. Hrsg. von A. Joux. Bd. 5479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, S. 224–241. ISBN: 978-3-642-01000-2. DOI: 10.1007/978-3-642-01001-9\_13 (siehe S. 17).
- [BGB04] N. Borisov, I. Goldberg und E. Brewer. „Off-the-record Communication, or, Why Not to Use PGP“. In: *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. WPES ’04. Washington DC, USA: ACM, 2004, S. 77–84. ISBN: 1-58113-968-3. DOI: 10.1145/1029179.1029200 (siehe S. 11).
- [Bun09] Bundesanzeiger. „Gesetz zur Änderung datenschutzrechtlicher Vorschriften“. In: *Bundesgesetzblatt* I.54 (Aug. 2009), S. 2814 (siehe S. 48).
- [Bus12] Business Wire. *Strategy Analytics: Worldwide Smartphone Population Tops 1 Billion in Q3 2012*. Okt. 2012. URL: <http://www.businesswire.com/news/home/20121017005479/en/Strategy-Analytics-Worldwide-Smartphone-Population-Tops-1> (siehe S. 9).
- [CHHS13] C. Cachin, K. Haralambiev, H.-C. Hsiao und A. Sorniotti. „Policy-based Secure Deletion“. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS ’13. Berlin, Germany: ACM, 2013, S. 259–270. ISBN: 978-1-4503-2477-9. DOI: 10.1145/2508859.2516690 (siehe S. 18).
- [Cou11] T. M. Coughlin. *Self-Encrypting Drive. Market & Technology Report*. Techn. Ber. Coughlin Associates, Sep. 2011. URL: [http://www.tomcoughlin.com/Techpapers/2011%20Self-Encrypting\\_Drive\\_Market\\_and\\_Technology\\_Analysis%20Brochure,\\_092011.pdf](http://www.tomcoughlin.com/Techpapers/2011%20Self-Encrypting_Drive_Market_and_Technology_Analysis%20Brochure,_092011.pdf) (siehe S. 18).

- [DDN00] D. Dolev, C. Dwork und M. Naor. „Non-Malleable Cryptography“. In: *SIAM Journal on Computing*. 2000, S. 542–552 (siehe S. 16).
- [Dwo01] M. Dworkin. *Recommendation for Block Cipher Modes of Operation*. Techn. Ber. National Institute of Standards und Technology, 2001. URL: <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf> (siehe S. 17).
- [Ert12] W. Ertel. *Angewandte Kryptographie*. 4. Aufl. Carl Hanser Verlag GmbH & Co. KG, 2012. ISBN: 978-3446427563 (siehe S. 15).
- [Fin14] J. Fingas. „Two-thirds of Americans now have smartphones“. In: *Engadget* (Feb. 2014). URL: <http://www.engadget.com/2014/02/11/two-thirds-of-americans-now-have-smartphones/> (siehe S. 9).
- [Gar15] Gartner. *Gartner Says Smartphone Sales Surpassed One Billion Units in 2014*. März 2015. URL: <https://www.gartner.com/newsroom/id/2996817> (siehe S. 9).
- [Gen09] C. Gentry. „Fully Homomorphic Encryption Using Ideal Lattices“. In: *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*. STOC '09. Bethesda, MD, USA: ACM, 2009, S. 169–178. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536440 (siehe S. 17).
- [GM84] S. Goldwasser und S. Micali. „Probabilistic encryption“. In: *Journal of computer and system sciences* 28.2 (1984), S. 270–299. DOI: 10.1016/0022-0000(84)90070-9 (siehe S. 16).
- [GMP+13] G. Greenwald, E. MacAskill, L. Poitras, S. Ackerman und D. Rushe. „Microsoft handed the NSA access to encrypted messages“. In: *The Guardian* (Juli 2013). URL: <http://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data> (siehe S. 9).
- [Goo15a] Google. *Google App Engine - Platform as a Service*. Nov. 2015. URL: <https://cloud.google.com/appengine/docs> (siehe S. 34).
- [Goo15b] Google. *Konten auf Ihrem Gerät hinzufügen und verwenden*. 2015. URL: [https://support.google.com/googleplay/answer/2521798?hl=de&ref\\_topic=3364260](https://support.google.com/googleplay/answer/2521798?hl=de&ref_topic=3364260) (siehe S. 33).
- [Goo15c] Google. *Legal Notice. Android Developers*. 2015. URL: <https://developer.android.com/legal.html> (siehe S. 33).
- [Goo15d] Google. *Quotas - App Engine - Google Cloud Platform*. Nov. 2015. URL: <https://cloud.google.com/appengine/docs/quotas> (siehe S. 33).
- [IDR15] IDRIX. *VeraCrypt*. 2015. URL: <https://veracrypt.codeplex.com/documentation> (siehe S. 11, 22).

- [Joh08] N. Johnson. „How do you access an authenticated Google App Engine service from a (non-web) python client?“ In: *Stackoverflow* (Sep. 2008). URL: <https://stackoverflow.com/questions/101742/how-do-you-access-an-authenticated-google-app-engine-service-from-a-non-web-py#102509> (siehe S. 39).
- [Kah13] J. Kahn. „Apple at Q3 call: iCloud reaches 320M accounts, 900B iMessages, 125B photo uploads“. In: *9to5 Mac* (Juli 2013). URL: <http://9to5mac.com/2013/07/23/apple-talks-numbers-at-q2-earnings-itunes-sales-retail-stores-320m-icloud-accounts/> (siehe S. 10).
- [Ker83] A. Kerckhoffs. „La Cryptographie Militaire“. In: *Journal des Sciences Militaires* (1883) (siehe S. 15).
- [Kir15] J. Kirk. „Anthem now says 78.8M were affected by breach“. In: *Computerworld* (Feb. 2015). URL: <http://www.computerworld.com/article/2888267/anthems-now-says-788m-were-affected-by-breach.html> (siehe S. 9).
- [Kov15] E. Kovacs. „Excellus Data Breach Impacts 10 Million“. In: *SecurityWeek* (Sep. 2015). URL: <http://www.securityweek.com/excellus-data-breach-impacts-10-million> (siehe S. 9).
- [Lar15] F. Lardinois. „Gmail Now Has 900M Active Users, 75% On Mobile“. In: *TechCrunch* (Mai 2015). URL: <http://techcrunch.com/2015/05/28/gmail-now-has-900m-active-users-75-on-mobile/?ncid=rss> (siehe S. 10).
- [LBBK15] M. Lucht, R. Bredenkamp, M. Boeker und U. Kramer. *Gesundheits- und Versorgungs-Apps. Hintegründe zu deren Entwicklung und Einsatz*. Universitätsklinikum Freiburg. 2015. URL: <https://www.tk.de/tk/themen/digitale-gesundheit/studie-gesundheits-apps-freiburg/744480> (siehe S. 10).
- [Lud12] S. Ludwig. „Gmail finally blows past Hotmail to become the world’s largest email service“. In: *VentureBeat* (Juni 2012). URL: <http://venturebeat.com/2012/06/28/gmail-hotmail-yahoo-email-users/> (siehe S. 10).
- [Med14] Medtronic. *Medtronic CareLink Network for Cardiac Device Patients*. Dez. 2014. URL: <http://www.medtronic.com/for-healthcare-professionals/products-therapies/cardiac-rhythm/patient-management-carelink/medtronic-carelink-network-for-cardiac-device-patients/index.htm> (siehe S. 10).
- [Mic15a] Microsoft. *OneDrive for Business*. 2015. URL: <https://onedrive.live.com/about/de-de/business/> (siehe S. 9, 10).

- [Mic15b] Microsoft. „Transparent Data Encryption (TDE)“. In: *SQL Server 2016* (Nov. 2015) (siehe S. 18, 30).
- [mmq14] mmq/dpa/AFP. „Nacktfotos von Lawrence und Co.: Apple weist Mitschuld an Hackerangriff zurück“. In: *Spiegel Online* (Sep. 2014). URL: <http://www.spiegel.de/netzwelt/web/hacker-klaunen-nacktfotos-von-promis-laut-apple-keine-luecke-bei-icloud-a-989545.html> (siehe S. 9).
- [Ora15] Oracle. „Transparent Data Encryption“. In: *Database 12c* (2015). URL: <http://www.oracle.com/technetwork/database/options/advanced-security/index-099011.html> (siehe S. 18, 30).
- [Pai99] P. Paillier. „Public-Key Cryptosystems Based on Composite Degree Residuosity Classes“. In: *Advances in Cryptology - EUROCRYPT '99*. Hrsg. von J. Stern. Bd. 1592. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, S. 223–238. ISBN: 978-3-540-65889-4. DOI: 10.1007/3-540-48910-X\_16 (siehe S. 17).
- [Per08] C. Perrin. „The CIA Triad“. In: *TechRepublic* (Juni 2008). URL: <http://www.techrepublic.com/blog/it-security/the-cia-triad/> (siehe S. 13).
- [Pfa15] Pfau Cochran Vertetis Amala PLLC. *Premera Data Breach Class Action Lawsuit*. 2015. URL: <https://www.premeralawsuit.com/> (siehe S. 9).
- [Pre15] Premera Blue Cross. *Premera Update - FAQ's*. 2015. URL: <http://premeraupdate.com/faqs/> (siehe S. 9).
- [PRZB11] R. A. Popa, C. M. S. Redfield, N. Zeldovich und H. Balakrishnan. „CryptDB: Protecting Confidentiality with Encrypted Query Processing“. In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Cascais, Portugal, 2011, S. 85–100. ISBN: 978-1-4503-0977-6. DOI: 10.1145/2043556.2043566 (siehe S. 23).
- [Sha49] C. E. Shannon. „Communication theory of secrecy systems“. In: *Bell system technical journal* 28.4 (Okt. 1949), S. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x (siehe S. 16).
- [SM14] C. Stach und B. Mitschang. „Design and Implementation of the Privacy Management Platform“. In: *Mobile Data Management (MDM), 2014 IEEE 15th International Conference on*. Bd. 1. Juli 2014, S. 69–72. DOI: 10.1109/MDM.2014.14 (siehe S. 3, 21).
- [SM15] C. Stach und B. Mitschang. „Der Secure Data Container (SDC)“. In: *Datenbank-Spektrum* 15.2 (2015), S. 109–118. ISSN: 1618-2162. DOI: 10.1007/s13222-015-0189-y (siehe S. 3, 13, 21).

- [SS94] R. S. Sandhu und P. Samarati. „Access control: principle and practice“. In: *Communications Magazine, IEEE* 32.9 (Sep. 1994), S. 40–48. ISSN: 0163-6804. DOI: 10.1109/35.312842 (siehe S. 13).
- [Sta13] C. Stach. „Wie funktioniert Datenschutz auf Mobilplattformen?“ In: *43. Jahrestagung der Gesellschaft für Informatik - Informatik 2013*. Sep. 2013 (siehe S. 21).
- [Tre15] Tresorit. *Tresorit White Paper*. 2015. URL: <https://tresorit.com/files/tresoritwhitepaper.pdf> (siehe S. 11, 23).
- [Wor14] World Wide Web Consortium. *Forms - HTML5*. Okt. 2014. URL: <http://www.w3.org/TR/html5/forms.html> (siehe S. 36).

Alle URLs wurden zuletzt am 14.12.2015 geprüft.



## **Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift