HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Entwicklung eines WebDAV Proxy zum sicheren Speichern von Daten in der Cloud

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2016

Autoren:	Adrian Anthamatten Peter Frick Jeyanthan Ravindran
Betreuer:	Prof. Oliver Augenstein
Experte:	Dipl. El.-Ing. ETH Reto Bättig
Gegenleser:	Prof. Dr. Peter Heinzmann

Inhaltsverzeichnis

1. Abstract

Ausgangslage
Durch die stetig steigende Internet-Bandbreite und Verfügbarkeit mobiler Netze, stehen neue technologische Möglichkeiten wie Cloud-Dienste zum Speichern von Daten zur Verfügung. Diese werden immer günstiger und damit auch attraktiver für die breite Masse. Solche Dienste haben aber oft auch gravierende Nachteile: In den letzten Jahren gab es immer wieder Berichte über Datendiebstähle und es ist kein Geheimnis, dass vor allem preisgünstige Cloud Anbieter die gespeicherten Daten für Auswertungen und Analysezwecke verwenden. Ausserdem garantieren die Anbieter bei kostenlosen Angeboten keinen Schutz vor Datenverlust.

Vorgehen
Das Ziel dieses Projekts ist eine WebDAV basierte Anwendung, welche die Daten über mehrere Cloud Anbieter redundant verteilt. Dadurch ist sichergestellt, dass Daten nicht verloren gehen, wenn ein Anbieter die Daten temporär oder dauerhaft nicht mehr zur Verfügung stellen kann. Des Weiteren sollen die Daten vor unerwünschten Zugriffen der Provider so geschützt werden, dass gängigen Cloud-Funktionen, wie z. B. das Teilen von Dokumenten, weiterhin genutzt werden können.

Ergebnis
Das Resultat dieser Arbeit ist ein WebDAV Server, welcher als Proxy zwischen Client und Cloud Anbieter fungiert. Sobald der Proxy-Server mit mehreren Cloud Anbietern verbunden ist, kann sich der Benutzer über einen beliebigen WebDAV Client mit dem Proxy-Server verbinden und wie gewohnt seine Daten manipulieren. Die Daten werden für den Benutzer transparent nach dem RAID5 Verfahren in Blöcke aufgeteilt, welche so auf die Provider verteilt werden, dass bei einem Ausfall eines Cloud Anbieters weiterhin auf die Daten zugegriffen werden kann.

Um den Zugriff der Provider auf die Daten zu verhindern, werden die bei den einzelnen Providern abgespeicherten Datenblöcke mit automatisch generierten Keys verschlüsselt. Diese Keys werden dann bei einem anderen der beteiligten Provider abgelegt, was gewährleistet, dass ein einzelner Provider die bei ihm gespeicherten Daten nicht entschlüsseln kann. Beim Lesen einer Datei sammelt sich der Proxy-Server alle relevanten Blöcke und entschlüsselt diese mit den dazugehörigen Keys.

Um Daten mit anderen Benutzern zu teilen, können Links generiert und verschickt werden. Mithilfe dieser Links und einem eigenen Proxy-Server können die anderen Benutzer auf die freigegebenen Daten zugreifen.

2. Management Summary

2.1. Ausgangslage

Seit dem NSA-Skandal (Snowden Enthüllungen, 2013) haben viele Nutzer das Vertrauen in die Cloud Anbieter verloren. Es ist kein Geheimnis, dass diese die gespeicherten Daten von Benutzern analysieren und auswerten. Ein Benutzer, welcher nicht viele IT-Kentnisse besitzt, kann sich kaum dagegen schützen. Da die Benutzer auf ihre Daten mit verschiedenen Clients (Smartphone, Notebook, Desktop-PC, etc.) zugreifen möchten oder Cloud Angebote als Backup benutzen, ist der vollständige Verzicht der Cloud keine Option. Wenn ein Cloud Anbieter zeitweise nicht verfügbar ist oder gar ganz den Betrieb einstellt, hat der Benutzer keinen Zugriff auf seine Daten mehr.

In diesem Projekt wird eine Anwendung umgesetzt, welche Daten mithilfe des WebDAV-Protokolls auf verschiedenen Cloud Anbieter verteilt speichert. WebDAV ist eine Technologie, mit der man Dateien und ganze Ordnerstrukturen im Internet bereitstellen kann.

Damit die Daten auch erreichbar sind, wenn ein Cloud Anbieter temporär oder dauerhaft ausfällt, werden die Daten mehrfach (redundant) bei den Cloud Anbieter gespeichert. Des Weiteren werden die Daten so abgelegt, dass die Cloud Anbieter keinen Zugriff auf die Daten haben und sie somit keine nützlichen Analysen und Auswertungen mit den Daten anstellen können.

2.2. Vorgehen

2.2.1. Evaluation von WebDAV-Technologien

Diese Arbeit baut auf einem bereits existierenden WebDAV Server auf. Der Server soll dahingehend erweitert werden, dass er als Proxy Server fungieren kann. Da die Auswahl eines geeigneten Servers direkten Einfluss auf den Erfolg der Arbeit hat, wurde in einem ersten Schritt eine Evaluation möglicher WebDAV Frameworks durchgeführt. Eines der wichtigsten Kriterien war die Anpassungsfähigkeit des WebDAV Servers. Am besten abgeschnitten hat das Milton Annotation Framework. Dabei handelt es sich um eine in Java geschriebene Library, welche das WebDAV Protokoll implementiert.

2.2.2. WebDAV Proxy Architektur

Der WebDAV Proxy wurde in mehrere Komponenten unterteilt: Dateisystem, RAID5 Komponente und Verschlüsselungskomponente. Die drei Komponenten sind für die Verwaltung von Dateien und deren Metainformationen, die Ausfallsicherheit und für den Schutz vor den unerwünschten Datenzugriffen der Cloud Anbieter zuständig.

2.3. Ergebnis

2.3.1. Dateisystem

Das Dateisystem ist das Bindeglied zwischen der WebDAV-Schnittstelle und der RAID5 Komponente. Hier wird festelegt, wie die Datenstruktur bei den Cloud Anbietern aussieht. Ausserdem werden die Dateieigenschaften (sog. Metainformationen) wie z.B. das Erstellungsdatum oder der Name der Datei verwaltet.

2.3.2. RAID5 Komponente

Die Daten werden auf den verschiedenen Cloud Anbietern redundant gespeichert, um die Ausfallsicherheit zu gewährleisten. Das redundante Speichern wurde mit dem RAID5 Verfahren umgesetzt. Mit dem RAID5 Verfahren werden die Dateien in kleine Teile aufgeteilt und auf den Cloud Anbietern gleichmässig verteilt. Bei einem Ausfall eines Cloud Anbieters können dank diesem Verfahren die Daten, welche sich auf dem ausgefallenen Cloud Anbieter befinden, vollständig wiederhergestellt werden.

2.3.3. Verschlüsselung

Damit die Cloud Anbieter nicht in der Lage sind die Daten des Benutzers einzusehen, werden diese mit dem AES Verfahren verschlüsselt. Die dabei verwendeten Keys befinden sich auf einem anderen beteiligten Cloud Anbieter. Dadurch wird gewährleistet, dass ein einzelner Cloud Anbieter die bei ihm gespeicherten Daten nicht entschlüsseln kann.

2.4. Ausbilck

Bei der Architektur wurde Wert auf die Erweiterbarkeit des WebDAV Proxys gelegt. Dank dieser Vorarbeit besteht die Möglichkeit das WebDAV Proxy mit weiteren Features wie z. B. mit Historisierung zu erweitern.

2.4.1. Konfigurationsinterface

Zurzeit werden die Einstellungen (Default Blocksize, Liste von verwendeten Cloud Anbietern, Logindaten für die einzelnen Cloud Providert, etc.) über die Umgebungsvariablen vorgenommen. Der WebDAV Proxy könnte um ein Konfigurationsinterface erweitert werden, mit welchem der Benutzer in der Lage ist, die Einstellungen über ein GUI zu editieren.

2.4.2. Historisierung

Historisierung ist eine Funktionalität, mit welcher ein Benutzer auf ältere Versionen von seinen Dateien zuzugreifen kann. Diese Funktionalität ist von den meisten Cloud Anbietern gegeben. Der WebDAV Proxy unterstützt diese Funktionalität zurzeit jedoch nicht und könnte mit der Historisierung erweitert werden.

3. Aufgabenstellung

Entwicklung eines WebDAV Proxy zum sicheren Speichern von Daten in der Cloud

Studiengang:	Informatik (I)
Semester:	FS 2016 (22.02.2016-18.09.2016)
Durchführung:	Bachelorarbeit, Studienarbeit
Fachrichtung:	Internet-Technologien und -Anwendungen
Institut:	Diverses
Gruppengrösse:	2-3 Studierende
Status:	zugewiesen
Verantwortlicher:	Augenstein, Oliver
Betreuer:	Augenstein, Oliver
Gegenleser:	Peter Heinzmann
Experte:	Reto Bättig, m&f engineering
Industriepartner:	nein

Ausschreibung:

In dieser Arbeit geht es um die Entwicklung eines WebDAV Servers, der lokal oder als Proxyserver installiert werden kann.

In seiner Grundfunktion soll der Proxy-Server als Fileserver dienen, der die empfangenen Daten in einem RAID 5 ähnlichen Verfahren auf 3 Cloudstoragemedien verteilt. Die Daten sollen im Zuge des Splits ausserdem mit automatisch erzeugten Keys so verschlüsselt werden, dass jeder Storageprovider effektiv nur einen zufälligen Datenstrom erhält.

Die Datensicherheit wird weiter dadurch erhöht, dass jeder Storageprovider durch das RAID 5 Protokoll nur auf die Hälfte aller gesendeten Daten physikalischen Zugriff hat.

Das RAID 5 Verfahren bietet ausserdem den Vorteil von Datenredundanz: Für den Zugriff auf die Daten soll es ausreichen, dass nur 2 der 3 Cloudstoragemedien verfügbar sind. Die Verschlüsselung der Daten schützt also vor dem unautorisierten Zugriff durch einen Storageprovider aber nicht vor einem unautorisierten Zugriff von zwei kollaborierenden Providern.

Erweiterungen der Grundfunktionalität sind in viele Richtungen denkbar. Z.B.
1) Unterstützung von distributed File-Sharing (ähnlich wie bei Drop-Box)
2) Erweiterungen der Verschlüsselungskomponente um die Datensicherheit auch vor unauthorisiertem Zugriff von mehreren Providern zu schützen
3) Proxy zu anderen WebDAV Servern ohne intermediären Filesystem-Layer
4) Unterstützung weiterer WebDAV ähnlicher Protokolle (z.B: calDAV)
5) Erhöhung des Redundanzlevels durch Unterstützung von RAID 6 (mathematisch anspruchsvoll)
6) Flexible Konfigurierbarkeit des Redundanzlevels und der bei einem einzelnen Provider im Zugriff stehenden Datenmenge durch ein eigenes Protokoll

Voraussetzungen:	Interesse an Algorithmen und an Cloud-Speichersystemen Selbständiges Arbeiten

Bewerbungen:	Gruppe:	Anthamatten/Frick/RAVINDRAN ✉
	Einschreibung:	Bachelorarbeit
	Status:	Arbeit zugewiesen (Priorität Student: 1)

O. Augstein

4. Einleitung

Dieses Projekt ist eine Bachelorarbeit an der Hochschule für Technik in Rapperswil (HSR). Bei der Bachelorarbeit handelt es sich um eine wissenschaftliche Arbeit, welche von Studenten zum Abschluss eines Bachelorstudiengangs durchgeführt wird. Der Themenvorschlag für dieses Projekt stammt vom Prof. Oliver Augenstein und trägt den Titel "Entwicklung eines WebDAV Proxy zum sicheren Speichern von Daten in der Cloud". In dieser Einleitung wird erläutert, was die Ziele dieses Projekts sind und welchen Nutzen das Endprodukt (ein WebDAV Proxy) für dessen Benutzer hat.

4.1. Aktuelle Situtation im Cloud Computing Bereich

Abbildung 1: NSA Skandal, Quelle: Truth In Media [1]

Seit dem NSA-Skandal (Snowden Enthüllungen, 2013) sind die meisten grossen Onlinespeicherdienste in Verruf geraten. In einigen Ländern sind diese Anbieter dazu gezwungen, gespeicherte Daten an die Regierung weiterzureichen, wenn es von ihnen gefordert wird. Es ist auch kein Geheimnis mehr, dass die online gespeicherten Daten von den Cloud Anbietern ausgewertet werden, um zum Beispiel massgeschneiderte Werbungen einzublenden. Dieses Verhalten seitens der Behörden und der Cloud Anbieter ist laut Statistik die grösste Sorge der Cloud Dienst Nutzer in der Schweiz.

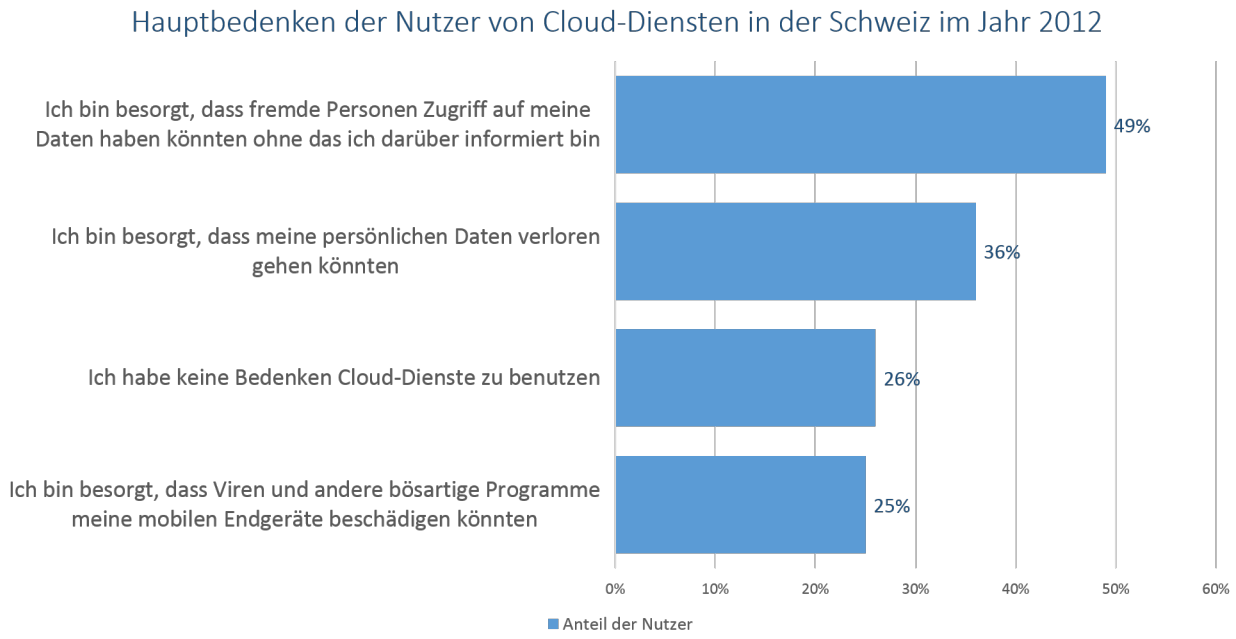

Abbildung 2: Hauptbedenken der Nutzer von Cloud-Diensten in der Schweiz im Jahr 2012, Quelle: Statista [2]

Obwohl seit dem NSA-Skandal 2013 schon Jahre vergangen sind, ist Datenschutz und Privatsphäre ein sehr aktuelles Thema. Das Vertrauen der Nutzer in die Cloud Anbieter fehlt weiterhin. Infolgedessen sind in den letzten Jahren viele neue Cloud Anbieter in den Markt eingedrungen, welche einen guten Datenschutzstandard versprechen. Es konnte sich jedoch keiner gross durchsetzten, da diese im Gegensatz zu den beliebten Cloud Anbietern kostenpflichtig sind. Aus diesen Gründen kommen die Nutzer weiterhin nicht von den grossen Cloud Anbietern weg.

4.2. Vision

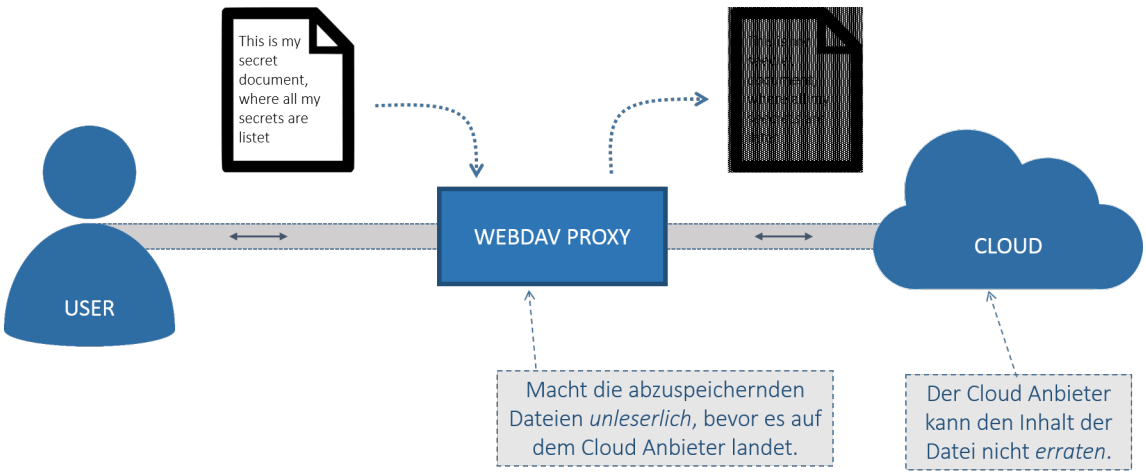

Abbildung 3: Daten für Cloud Anbieter unleserlich machen

Das Ziel dieses Projekts ist es, den Kunden Datensicherheit [3] unabhängig von verwendeten Cloud Anbietern zu gewährleisten. Hierbei wird davon ausgegangen, dass jeder Cloud Anbieter in der Lage ist, auf die Daten zuzugreifen, welche bei ihm abgelegt sind. Deshalb ist es wichtig die eigentlichen Daten “unleserlich“ auf der Cloud abzulegen und nur dem Kunden beim Auslesen der Daten diese wieder “leserlich“ darzustellen.

Der einzige Mehraufwand, welcher der Kunde haben sollte, ist die Installation und Konfiguration der Software (WebDAV Proxy). Danach kann der Kunde über einen WebDAV Client seine Daten in die Cloud hinauf- und herunterladen. Der Benutzer soll nicht mitbekommen, dass im Hintergrund die Daten unleserlich gemacht werden, bevor diese bei den Cloud Anbietern landen.

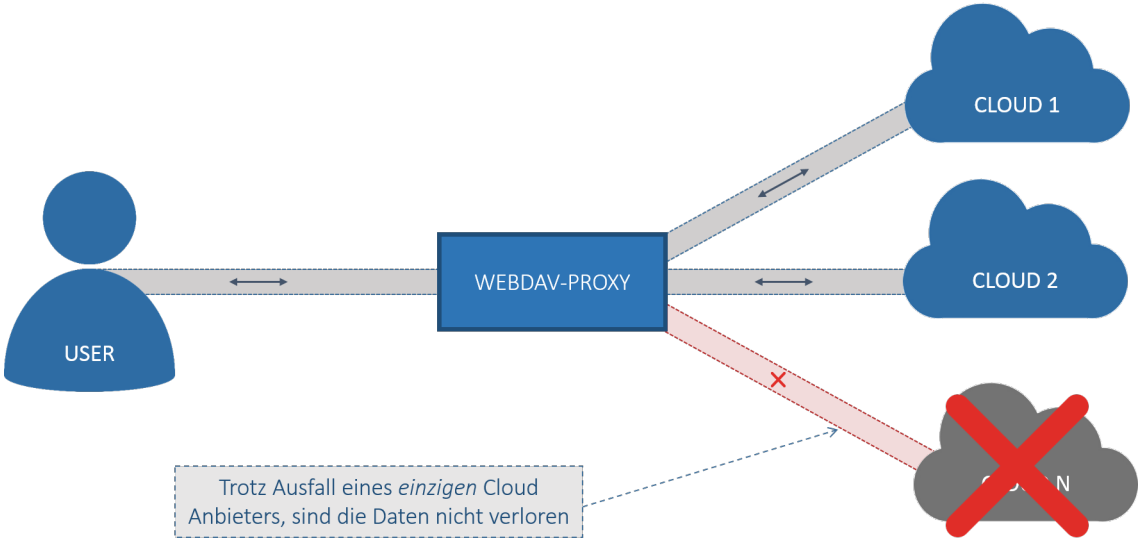

Abbildung 4: Ausfall eines Cloud Anbieters

Zusätzlich zum Datenschutz (Schutz vor unerlaubtem Zugriff) ist auch die Datensicherheit (Schutz vor Datenverlust) für die Nutzer von grosser Bedeutung. Damit sich der Benutzer bezüglich der Datensicherheit keine Sorgen machen muss, sollen die Daten auf verschiedene Cloud Anbieter verteilt werden. Dabei werden die Daten so verteilt, dass bei einem Ausfall eines einzigen Cloud Anbieters trotzdem auf die Daten zugegriffen werden kann. Anhand der Daten bei den restlichen Cloud Anbietern, sollen die verlorenen Teildaten wiederhergestellt werden können.

Dem Benutzer soll ausserdem die Möglichkeit geboten werden, seine Daten mit anderen zu Teilen. Die Sicherheit der Daten darf dadurch jedoch nicht beeinträchtigt werden.

5. Grundlagen

Die in diesem Kapitel beschriebenen Grundlagen sollen das Verständnis dieser Arbeit erleichtern. Dazu gehört das WebDAV Protokoll, welches im Zentrum dieser Arbeit steht. Da ein RAID5 ähnliches System sowie die Verschlüsselung von Daten ebenfalls wesentliche Bestandteile dieser Arbeit sind, werden auch diese Themen erläutert.

5.1. WebDAV

WebDAV [4] steht für Web-based Distributed Authoring and Versioning. Das Protokoll ist ein offener Standard [5] und dient dazu, Verzeichnisse und Dateien im Internet bereit zu stellen. WebDAV erweitert das HTTP/1.1 Protokoll, um dessen Einschränkungen in Bezug auf Datenmanipulationen aufzuheben. Dazu werden folgende zusätzliche Funktionen bereitgestellt:

- PROPFIND
  Wird verwendet, um Eigenschaften von Ressourcen oder ganze Verzeichnisstrukturen abzufragen.
- PROPPATCH
  Wird verwendet, um Eigenschaften von Ressourcen zu ändern.
- MKCOL
  Damit können Verzeichnisse (bei WebDAV „Collections“ genannt) erstellt wreden.
- COPY
  Hiermit können Ressourcen kopiert werden.
- MOVE
  Mit dieser Funktion kann eine Ressource in ein anderes Verzeichnis verschoben werden.
- DELETE
  Wird verwendet, um eine Ressource zu löschen.
- LOCK
  Damit kann eine Ressource gesperrt werden. Dadurch wird gewährleistet, dass sich zwei gleichzeitige Zugriffe nicht gegenseitig behindern oder gar Änderungen überschrieben werden.
- UNLOCK
  Wird verwendet, um gesperrte Ressource wieder frei zu gegeben.

Aufgrund der vollständigen Abwärtskompatibilität von HTTP/2 zu HTTP/1.1, sollte WebDAV auch mit HTTP/2 funktionieren.

Der Vorteil von WebDAV gegenüber Protokollen mit ähnlichen Funktionen wie z.B. FTP liegt darin, dass der Aufwand für den Benutzer geringer ist. Heutzutage stellt beinahe jedes Betriebssystem einen WebDAV Client zur Verfügung oder kann einfach nachinstalliert werden. Mit einem solchen Client ist es möglich Dateien direkt zu öffnen, zu verändern und anschliessend zu speichern wie man es von lokal gespeicherten Dateien gewohnt ist. Der Benutzer bekommt nicht mit, was im Hintergrund alles über WebDAV erledigt wird. Beim FTP Protokoll hingegen muss der Benutzer zuerst die Datei herunterladen, anschliessend kann er sie bearbeiten und muss sie dann wieder von Hand hochladen. Dadurch dass WebDAV eine Erweiterung vom HTTP Protokoll ist, kann der Port 80 verwendet werden. Dadurch sind keine weiteren Portfreigaben nötig, welche eine potentielle Sicherheitslücke darstellen könnte. Ausserdem kann man sich Verwaltungsaufwände sparen.

Folgende Grafiken zeigen ein Minimalbeispiel, wie die Ressource

`http://www.example.com/index.html`

nach

`http://www.example.com/moved/index.html`

mit der MOVE Funktion verschoben wird.

Request:

```
MOVE /index.html HTTP/1.1
Host: www.example.com
Destination: http://www.example.com/moved/index.html
```

Response:

```
HTTP/1.1 201 Created
Location: http://www.example.com/moved/index.html
```

5.2. RAID5

RAID steht für «Redundant Array of Independent Disks» [6] und dient dazu, Datenverluste zu vermeiden, indem die Daten redundant abgespeichert werden. RAID wird in verschiedene Levels unterteilt. Da in dieser Arbeit mit RAID5 gearbeitet wird, beschränkt sich dieses Kapitel auf dieses Level.

Bei einem RAID5 werden mindestens drei Festplatten benötigt. Die Daten werden in diesem Verfahren in sogenannte Datenblöcke (auch Block bzw. Blöcke genannt) aufgeteilt. Diese werden dann auf die verschiedenen Festplatten verteilt. Wenn eine Festplatte ausfällt, können die darauf gespeicherten Daten anhand einer sogenannten Parität wiederhergestellt werden. Dazu werden die Datenblöcke in Gruppen unterteilt, welche jeweils Blöcke mit Nutzdaten und einen einzigen Paritätsblock enthalten. Eine solche Gruppe wird von nun an als Slice bezeichnet. Die Blöcke mit Nutzdaten beinhalten die effektiven Daten der Datei und deren Anzahl pro Slice entspricht der Anzahl verwendeter Festplatten minus eins. Die Parität von einem Slice wird mittels XOR-Operation über die Blöcke mit Nutzdaten berechnet. Dabei werden die einzelnen Bits von zwei Blöcken miteinander

verglichen. Wenn beide Bits gleich sind, wird das Paritätsbit 0 und wenn die beiden unterschiedlich sind, wird das Paritätsbit 1. Abbildung 5 soll dies veranschaulichen. Wenn ein Slice mehr als zwei Blöcke mit Nutzdaten beinhaltet, wird die XOR-Operation zuerst für die ersten beiden Blöcke durchgeführt. Anschliessend wird mit dem Zwischenresultat und dem nächsten Block weitergemacht, solange bis alle Blöcke verrechnet wurden. Das Endresultat entspricht dann der Parity, welche abgespeichert werden kann.

a	b	a XOR b
0	0	0
1	1	0
1	0	1
0	1	1

Abbildung 5: XOR Operation

Bei einem Ausfall einer Festplatte müssen die verlorenen Datenblöcke wiederhergestellt werden, insofern es sich nicht um Paritätsblöcke handelt. Dazu werden genau gleich wie beim Berechnen der Parität gruppenweise XOR-Operationen durchgeführt. Das Resultat entspricht dann dem verlorenen Block.

Damit bei einem Ausfall nicht zu viele Daten wiederhergestellt werden müssen, ist es von Vorteil die Paritäten wie in Abbildung 6 gezeigt, gleichmässig auf den zur Verfügung stehenden Festplatten zu verteilen. Somit muss nur ein Bruchteil der Daten wiederhergestellt werden.

Der Vorteil vom RAID5 liegt darin, dass bei einem Ausfall einer Festplatte weiterhin auf die Daten zugegriffen werden kann, auch wenn die Festplatte noch nicht ausgetauscht wurde. Ausserdem wird im Vergleich zu anderen Verfahren wesentlich weniger Speicherplatz benötigt. Der zur Verfügung stehende Platz berechnet sich anhand folgender Formel:

(Anzahl der Festplatten - 1) * Kapazität der kleinsten Festplatte

Je mehr Festplatten verwendet werden, desto geringer fällt der prozentual benötigte Platz für die Paritäten im Verhältnis zu den Nutzdaten aus. Zudem können die Daten schneller ausgelesen werden, da von mehreren Festplatten gleichzeitig gelesen werden kann.

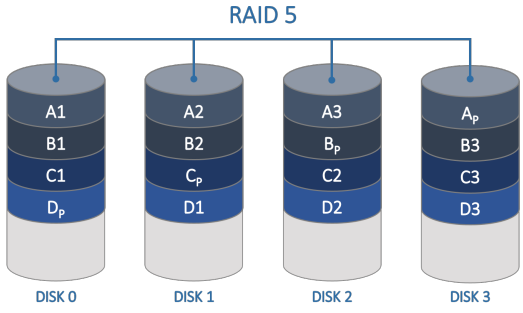

Abbildung 6: Daten- und Paritätsverteilung RAID5

5.3. Verschlüsselung

Die Verschlüsselung ist ein wesentlicher Teil der Arbeit, um den Zugriff der Cloud Anbieter auf die Daten zu begrenzen.

5.3.1. Algorithmus

Die Verschlüsselung der Daten erfolgt mit dem Advanced Encryption Standard (AES). Dieser ist der Nachfolger des Data Encryption Standard (DES) und wurde im Oktober 2000 vom National Institute of Standards and Technology (NIST) als Standard veröffentlicht [7].

5.3.2. Betriebsart

Der AES ist eine Blockchiffre. Dies bedeutet, er verschlüsselt Daten in Blöcken gewisser Länge, im Falle des AES 128 Bit. Um Daten zu verschlüsseln die mehr als 128 Bit Inhalt haben, existieren verschiedene Betriebsarten um die Blöcke aneinander zu hängen. In dieser Arbeit wurde dafür der so genannte Counter Mode (CTR Mode) verwendet. Dieser entfernt die Limitierung der Blockchiffre, dass die Länge der Daten immer ein Vielfaches der Blocklänge der Chiffre entsprechen muss [8]. Dem Verschlüsslungsalgorithmus wird noch ein Initialization Vector (IV) mitgegeben. Dieser wird für die Verschlüsselung benötigt, damit gleiche Blöcke nicht gleich verschlüsselt werden. Dies muss verhindert werden, da das Chiffrat sonst anfällig für Kryptoanalyse wird. Wie der IV gebildet wird, hängt von der Betriebsart ab [9].

5.3.3. Keys

Der AES bietet verschiedene Keylängen an: 128, 192 und 256 Bit. Für das Projekt wurden 128 Bit starke Keys verwendet. Obwohl dies die schwächste Option von AES ist bietet diese genügend Sicherheit, so dass die Daten über das Jahr 2030 noch sicher genug sind [10]. Grössere Keylängen sind also nicht nötig und haben gleich zweifach einen schlechten Einfluss auf die Performance. Zunächst wächst der Aufwand für den Verschlüsselungsvorgang mit der Keylänge mit. Ausserdem muss mehr Schlüsselmaterial generiert werden, um kryptographisch sichere Keys zu erhalten. Für die Keys werden pseudozufällige Bytes benötigt. Diese in genügender Qualität zu erhalten, ist aufwändig.

6. Requirement Analyse

Für die Requirement Analyse werden Userstories erstellt und priorisiert. Die Prioritäten werden in drei Kategorieen unterteilt: Must (höchste Priorität), Should (mittlere Priorität) und Could (tiefste Priorität). Ziel ist es alle Userstories aus der Kategorie Must umzusetzen und so viele wie möglich aus der Kategorie Should. Wenn am Schluss noch Zeit bleibt, werden die verbleibenden Userstories umgesetzt.

Userstory	Priorität	Umgesetzt
Simple WebDAV Proxy	Must	Ja
WebDAV Proxy mit Cloudanbieter	Must	Ja
Authentifizierung	Must	Ja
Datenverschlüsselung	Must	Ja
Dateiproperties	Must	Ja
Lesezugriff auf Daten bei Ausfall	Should	Ja
Link-Sharing für Dateien	Should	Ja
Konfigurationsinterface	Should	Nein
Vollzugriff auf Daten bei Ausfall	Could	Nein
Datenwiederherstellung	Could	Nein
Link-Sharing für Verzeichnisse	Could	Nein
Datei Historisierung	Could	Nein
Multi-User Support	Could	Nein
Data-Sharing	Could	Nein
CalDAV	Could	Nein
CardDAV	Could	Nein

Tabelle 1: Userstory Übersicht

Simple WebDAV Proxy

Kurzbeschreibung:	Als Benutzer möchte ich die Daten, welche auf dem lokalen Filesystem gespeichert sind, über den WebDAV Proxy manipulieren können.
Story:	Der User verbindet sich über einen WebDAV Client auf den WebDAV Proxy (ohne Login). Danach kann der User Dateien manipulieren (CRUD, mit Ausnahme von Benutzerrechten und Dateiproperties).
Akzeptanzkriterien:	- Der WebDAV Proxy muss errichbar sein. - Wenn der Benutzer CRUD Operationen auf dem Dateisystem (im WebDAV Proxy) macht, werden die Dateien entsprechend manipuliert.
Priorität:	Must

WebDAV Proxy mit Cloud Anbieter

Kurzbeschreibung:	Als Benutzer möchte ich die Daten, welche sich auf einem Cloud Anbieter befinden, über den WebDAV Proxy manipulieren können.
Story:	Der User verbindet sich über einen WebDAV Client auf den WebDAV Proxy (ohne Login). Danach kann der User Dateien manipulieren (CRUD, mit Ausnahme von Benutzerrechten und Dateiproperties). Der WebDAV Proxy verteilt die Datei auf mindestens drei Cloud Anbieter. Für den User ist dieser Vorgang transparent.
Akzeptanzkriterien:	- Der WebDAV Proxy ist mit allen Cloud Anbieter verbunden - Wenn der Benutzer CRUD Operationen auf dem Dateisystem (im WebDAV Proxy) macht, dann werden die Teile der Datei auf den Cloud Anbietern automatisch entsprechend manipuliert.
Priorität:	Must

Authentifizierung

Kurzbeschreibung:	Als Benutzer möchte ich mich nur mit einem gültigen Login anmelden können, um meine Daten vor fremden Zugriffen schützen zu können.
Story:	Der User verbindet sich über einen WebDAV Client mit dem korrekten Benutzernamen und Passwort auf dem WebDAV Proxy. Ohne gültige Benutzernamen und Passwort kann sich der Benutzer nicht anmelden.
Akzeptanzkriterien:	- Wenn sich der Benutzer mit seinem korrekten Benutzernamen und Passwort anmeldet, dann kann er auf die entschlüsselten Daten aus der Cloud zugreifen. - Wenn sich der Benutzer mit falschem Benutzernamen und Passwort anmeldet, dann kann er nicht auf die entschlüsselten Daten aus der Cloud zugreifen.
Priorität:	Must

Datenverschlüsselung

Kurzbeschreibung:	Als Benutzer möchte ich nicht, dass ein Cloud Anbieter meine Daten lesen kann.
Story:	Daten, welche bei Cloud Anbietern hinterlegt werden, sind mit einem gängigen Verschlüsselungsalgorithmus verschlüsselt, sodass diese nicht in der Lage sind die Daten zu lesen. Neben dem Inhalt der Daten, sind auch die Dateinamen und Ordnernamen nur verschlüsselt bei den Cloud Anbietern hinterlegt.
Akzeptanzkriterien:	- Dateiinhalte sind bei den Cloud Anbieter verschlüsselt hinterlegt - Datei- und Ordnernamen sind bei den Cloud Anbietern nicht in Klartext hinterlegt
Priorität:	Must

Dateiproperties

Kurzbeschreibung:	Als Benutzer möchte ich Properties von Dateien angezeigt bekommen.
Story:	Der User lässt sich ein Verzeichnis mit Dateien anzeigen. Neben den Dateinamen werden auch noch Properties (z.B. Dateigrösse, Erstellungsdatum, ...) angezeigt.
Akzeptanzkriterien:	- Wenn sich der User ein Verzeichnis anzeigen lässt, werden nicht nur die Dateinamen aufgelistet, sondern auch die Dateiproperties. - Bei einem Download kann aufgrund der Dateigrösse die noch verbleibende Zeit berechnet und angezeigt werden.
Priorität:	Must

Lesezugriff auf Daten bei Ausfall

Kurzbeschreibung:	Als Benutzer möchte ich auf Daten lesend zugreifen können, auch wenn ein Storage (Cloud Anbieter oder ein lokaler Ordner) vollständig ausfällt.
Story:	Wenn ein Storage vollständig ausfällt, hat der Benutzer trotzdem die Möglichkeit aufgrund eines RAID5 Systemes mittels Wiederherstellung der verbleibenden Teildaten, auf die gespeicherten Dateien zuzugreifen.
Akzeptanzkriterien:	- Es darf nur ein Storage ausfallen - Es sind nur lesende Zugriffe möglich (neue Dateien erstellen, löschen, verschieben und umbenennen ist nicht mehr möglich)
Priorität:	Should

Link-Sharing für Dateien

Kurzbeschreibung:	Als Benutzer möchte ich einzelne Dateien via Link für andere freigeben.
Story:	Der Benutzer kann einzelne Dateien für Benutzer mit einem eigenen WebDAV Proxy via Links freigeben. Zusätzlich hat der Benutzer Zugriff auf Daten, welche für ihm freigegeben wurden, indem er den Link von einem anderen Benutzer erhalten hat.
Akzeptanzkriterien:	- Wenn ein Benutzer A den Link einer Datei an Benutzer B sendet, dann kann Benutzer B auf die freigegebenen Daten von Benutzer A zugreifen und umgekehrt.
Priorität:	Should

Konfigurationsinterface

Kurzbeschreibung:	Als Benutzer möchte ich Konfigurationen für den WebDAV Proxy über ein User Interface verwalten.
Story:	Über ein User Interface werden dem Benutzer Konfigurationsmöglichkeiten (z.B. Verwalten von Cloud Anbieter Accounts, Dateifreigaben, ...) für den WebDAV Proxy zur Verfügung gestellt.
Akzeptanzkriterien:	- Wenn im User Interface die Konfigurationen für den Cloud Anbieter mit den korrekten Anmeldeinformationen hinterlegt sind, dann werden die Daten (bzw. Teile von Daten) auf die Cloud Anbieter hochgeladen - Wenn im User Interface bestimmte Dateien/Verzeichnisse für andere Benutzer freigegeben werden, dann können diese auf die freigegebenen Daten zugreifen
Priorität:	Should

Vollzugriff auf Daten bei Ausfall

Kurzbeschreibung:	Als Benutzer möchte ich auf meine Daten vollen Zugriff haben (CRUD), auch wenn ein Storage (Cloud Anbieter oder ein einfacher Ordner) vollständig ausfällt.
Story:	Wenn ein Storage vollständig ausfällt, hat der Benutzer trotzdem die Möglichkeit aufgrund eines RAID5 Systemes mittels Wiederherstellung der verbleibenden Teildaten, vollständig auf die gespeicherten Dateien zuzugriefen (CRUD).
Akzeptanzkriterien:	- Es darf nur ein Storage ausfallen
Priorität:	Could

Datenwiederherstellung

Kurzbeschreibung:	Nachdem ein ausgefallener Storage (Cloud Anbieter oder ein einfacher Ordner) wieder verfügbar ist, möchte ich als Benutzer nichts tun müssen um die Daten wieder in einen konsistenten Zustand zu bringen.
Story:	Nachdem ein ausgefallener Storage wieder verfügbar ist, soll automatisch ein Dienst gestartet werden, der alle in der Zwischenzeit des Ausfalls veränderten Daten rekonstruiert und die Daten wieder in einen konsistenten Zustand bringt.
Akzeptanzkriterien:	- Der Benutzer soll davon nichts mitbekommen
Priorität:	Could

Link-Sharing für Verzeichnisse

Kurzbeschreibung:	Als Benutzer möchte ich ganze Verzeichnisse (Ordner) via Link für andere freigeben.
Story:	Der Benutzer kann ganze Verzeichnisse für Benutzer mit einem eigenen WebDAV Proxy via Links freigeben. Zusätzlich hat der Benutzer Zugriff auf Daten, welche für ihm freigegeben wurden, indem er den Link von einem anderen Benutzer erhalten hat.
Akzeptanzkriterien:	- Wenn ein Benutzer A den Link eines Verzeichnis an Benutzer B sendet, dann kann Benutzer B auf die freigegebenen Daten von Benutzer A zugreifen und umgekehrt.
Priorität:	Could

Datei Historisierung

Kurzbeschreibung:	Als Benutzer möchte ich die Versionen der gespeicherten Dateien verwalten können.
Story:	Der Benutzer kann einzelne Dateien verändern, ohne diese zu überschreiben. Es werden verschiedene Versionen angelegt, auf welche der Benutzer zugreifen kann.
Akzeptanzkriterien:	- Wenn ein Benutzer eine bereits bestehende Datei verändert, wird diese nicht überschrieben, sondern eine neue Version angelegt. - Wenn der Benutzer eine ältere Version der Datei verwenden möchte, dann kann er in der Historie der Datei die gewünschte Version auswählen, welche die aktuelle Version ersetzt - Wenn der Benutzer die aktuelle Version einer Datei löscht, wird mit dieser auch dessen Historie unwiderruflich gelöscht
Priorität:	Could

Multi-User Support

Kurzbeschreibung:	Als Administrator möchte ich auf den WebDAV Proxy mehrere Benutzerkonten anlegen können.
Story:	Der Administrator hat die Möglichkeit mehrere Benutzerkonten anzulegen, wobei jeder Account seine eigenen Datenablage hat.
Akzeptanzkriterien:	- Wenn der Administrator einen neuen Benutzer mit Benutzernamen und Passwort anlegt, dann wird ihm eine eigene Datenablage zur Verfügung gestellt, welcher ab sofort benutzt werden kann
Priorität:	Could

Data-Sharing

Kurzbeschreibung:	Als Benutzer möchte ich bestimmte Daten mit anderen registrierten Benutzern teilen.
Story:	Der Benutzer kann einzelne Dateien oder Verzeichnisse für registrierte Benutzer freigeben. Dieser wiederum kann die freigegebenen Dateien und Verzeichnisse über seinen Account einsehen.
Akzeptanzkriterien:	- Wenn ein Benutzer A eine Dateien oder Verzeichnis an Benutzer B freigibt, dann kann Benutzer B auf die freigegebenen Daten von Benutzer A zugreifen und umgekehrt. - Der Benutzer, für den die Dateifreigabe eingerichtet werden soll, muss einen eigenen Account beim WebDAV Proxy besitzen (Multi-User Support).
Priorität:	Could

CalDAV

Kurzbeschreibung:	Als Benutzer möchte ich meine Termine über den WebDAV Proxy sichern.
Story:	Der User kann in seinem Kalender Termine verwalten, welche automatisch mithilfe von CalDAV über den WebDAV Proxy in die Cloud synchronisiert werden.
Akzeptanzkriterien:	- Wenn ein Benutzer einen neuen Termin anlegt, löscht oder einen bestehenden Termin verändert, wird dieser automatisch in der Cloud synchronisiert.
Priorität:	Could

CardDAV

Kurzbeschreibung:	Als Benutzer möchte ich meine Kontakte über den WebDAV Proxy sichern.
Story:	Der User kann in seinem Kontaktverwaltungsprogramm Kontakte verwalten, welche automatisch mithilfe von CardDAV über den WebDAV Proxy in die Cloud synchronisiert werden.
Akzeptanzkriterien:	- Wenn der Benutzer einen neuen Kontakt anlegt, löscht oder einen bestehenden Kontakt verändert, wird dieser automatisch in der Cloud synchronisiert.
Priorität:	Could

7. Risikomanagement

In dieser Tabelle wurden die Risiken im Projekt erfasst:

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrsch.	Gewichteter Schaden	Vorbeugung	Verhalten beim Eintreten
R1	Lokaler Datenverlust	Schaden an der Arbeitsstation (Notebook) eines Projektmitglieds führt zu Datenverlust	8	1%	0.08	Lokal gespeicherte Daten mindestens alle 4 Stunden einchecken	noch nicht eingecheckte Arbeiten gehen verloren und müssen nochmals erledigt werden
R2	Temporärer Ausfall eines Projektmitglieds	Ein Projektmitglied fällt für mehr als 1 Tag aus und erzielt nicht die gewünschten Fortschritte im Projekt	20	5%	1	Aufgaben priorisieren, damit die verbliebenen Projektmitglieder an den Aufgaben mit hoher Priorität weiterarbeiten können	Primär an Aufgaben mit hoher Priorität arbeiten
R3	Der WebDAV Server erfüllt nicht alle Anforderungen	Der ausgesuchte WebDAV Server erfüllt nicht alle Anforderungen, welche für das Projekt wichtig sind (z.B. Möglichkeit den bestehenden WebDAV Server zu erweitern)	80	2%	1.6	Bei der Auswahl des WebDAV Servers (Evaluation), muss genau überprüft werden, ob die Technologie alle Anforderungen erfüllt.	Entweder die fehlende Funktion selbst implementieren und diese in die bereits existierende Lösung integrieren oder den ausgewählten WebDAV Server vollständig durch ein anderes Produkt ersetzen.
R4	Änderung an der API der Cloud Anbieter	Die API von OwnCloud in Bezug auf das Teilen von Dateien ist nicht spezifiziert. Deshalb besteht die Möglichkeit, dass OwnCloud diese Schnittstelle ändern könnte.	10	10%	1	Die Sharing-Klassen unabhängig implementieren, sodass bei einer API Änderung beim Cloud Anbieter nur diese Klasse angepasst werden muss.	Sharing-Klasse anpassen
R5	Link Sharing zu komplex	Das Link Sharing könnte aufgrund der Verschlüsselung der Daten zu komplex für den Rahmen dieser Arbeit werden.	40	20%	8	Zuerst ein Konzept überlegen, welches funktionieren könnte. Wenn dies zu lange dauert und keine gute Lösung in einem angemessenen Zeitraum entsteht, die gewonnenen Erkenntnisse dokumentieren und mit einem anderen Feature anfangen.	Es wurde viel Zeit "verschwendet", welche für andere Features besser eingesetzt werden hätte können.

Tabelle 2: Risiken

8. WebDAV Server Evaluation

Der Projekterfolg steht und fällt mit der Wahl eines geeigneten WebDAV Servers. Aus diesem Grund wird eine Evaluation verschiedener möglicher Server durchgeführ. Ziel der Evaluation ist es den am besten geeigneten Server für das Projekt zu finden.

Als erstes wurde recherchiert und eine Auflistung von WebDAV Servern erstellt, welche in Frage kommen könnten:

- Java (WebdavServlet) [11]
  Ein Servlet, welches für Apache-Tomcat-Server entwickelt wurde
- NodeJS (jsDAV) [12]
  Erlaubt das einfache Hinzufügen von WebDAV-Support zu einer NodeJS-Applikation
- PHP (SabreDAV) [13]
  SabreDAV ist das populärste WebDAV-Framework für PHP
- Java (Milton) [14]
  Eine Java-Library für die Entwicklung eines benutzerdefinierten WebDAV Servers
- Python (PyWebDAV) [15]
  Eine WebDAV-Library mit einem standalone Server für Python

Für die Evaluation wurde die Anzahl der zu prüfenden Technologien auf drei begrenzt, somit konnte jedem Projektmitglied eine Technologie zugewiesen werden. Ausgesucht wurde das WebdavServlet und Milton, da die Projektmitglieder mit Java am besten vertraut waren. Als dritte Technologie wurde SabreDAV ausgewählt, da dies ein weitverbreiteter WebDAV Server ist und von einzelnen Cloud Anbietern wie z.B. OwnCloud [16] eingesetzt wird. Die ausgesuchten Technologien wurden jeweils vollständig mithilfe eines Prototyps untersucht.

8.1. Milton

Milton [14] ist eine Library der McEvoy Software Ltd. Sie bietet in einer Community Edition alle erforderlichen Funktionen für eine einfache WebDAV Anbindung unter der Apache Lizenz Version 2.0 an.

8.1.1. Funktionsumfang

Milton ist in zwei Versionen erhältlich: Community Edition und Enterprise Edition. Die Community Edition enthält alle für eine WebDAV Anbindung wichtigen Funktionen. Dabei wird nativ nur WebDAV level 1 (ohne Locking) unterstützt. Dies bedeutet, dass Clients welche nur WebDAV level 2 (mit Locking) unterstützen, keine Files hochladen können. Im Test war dies mit dem Nautilus Filebrowser der Fall. Diese Limitation kann jedoch mit wenigen Konfigurationen und überschreiben zweier Klassen umgangen werden. Die Enterprise Edition bietet einen grössenren Umfang an und unterstützt neben dem normalen WebDAV auch noch CalDAV und CardDAV zur Kalender- und Kontaktverwaltung an.

8.1.2. Anpassungsmöglichkeit

Um die verschiedenen Operationen von WebDAV zu implementieren, bietet Milton zwei Möglichkeiten: Das Annotation-Framework oder das Schreiben eines eigenen Controllers. Das Annotation-Framework ist die empfohlene Variante und bietet für die Entwicklung eines eigenen WebDAV Servers den grössten Komfort. Wie der Name schon andeutet, funktioniert das Annotation-Framework mit Java-Annotations. Damit können die benötigten Methoden wie das Hochladen oder Verschieben eines Files mit den entsprechenden Annotations gekennzeichnet werden.

8.1.3. Lizenz

Die beiden Versionen von Milton, Community Edition und Enterprise Edition, sind unter unterschiedlichen Lizenzen veröffentlicht.

Community Edition Die Community Edition ist vollständig unter der Apache 2 Lizenz veröffentlicht. Die Apache 2 Lizenz ist eine Open Source Lizenz und ermöglicht es Anpassungen am Code vorzunehmen und diese wieder zu veröffentlichen [17].

Enterprise Edition Die Enterprise Edition ist kostenpflichtig und kostet 2'700$ für 3 Server und 4'500$ für 5 Server. Diese ist proprietär.

8.2. SabreDAV

SabreDAV [13] ist der am weitesten verbreitete WebDAV Server in der PHP Welt. Der Server wurde komplett in PHP implementiert und läuft auf einem Apache2 Server. Die aktuellste Version ist 3.1 (stand 08.03.2016) und setzt die PHP Version 5.5 voraus.

8.2.1. Funktionsumfang

Der Server kann sehr einfach aufgesetzt werden. Dazu muss lediglich SabreDAV heruntergeladen (zip Archiv ins gewünschte Verzeichnis entpacken) und ein paar Zeilen PHP Code geschrieben werden. Über Plugins können Erweiterungen wie Benutzer Authentifizierung, Access Control List für Datei Zugriffsrechte oder Locking der Dateien installiert werden. Der Hersteller von SabreDAV bietet noch viel mehr Plugins an.

8.2.2. Anpassungsmöglichkeiten

Um SabreDAV zu erweitern, müssen alle Klassen überschrieben werden, welche gegen das Dateisystem implementiert wurden. Das bedeutet, dass z.B. bei der Klasse „File“ die Schreib- und Lesemethode überschrieben werden müssen.

8.2.3. Lizenz

SabreDAV ist Opne Source und steht unter der BSD Lizenz [18]. Das bedeutet, dass man die Software verändern darf, jedoch nich verpflichtet ist, den Quellcode der erweiterten Software zu veröffentlichen.

8.2.4. Prototyp

In einem ersten Schritt wurde der Server mit den Minimalfunktionen zum laufen gebracht und über einen WebDAV Client zugegriffen. Dabei wurden Funktionen wie zum Beispiel Anlegen von Dateien oder Ordner, verändern der Dateien und das Löschen der Dateien getestet. Um zu beweisen dass der Server soweit verändert werden kann, wie dies im Rahmen dieser Arbeit nötig ist, soll das physische Dateisystem abgekoppelt und durch ein virtuelles ersetzt werden. Die Dateien würden dann nicht mehr auf der Festplatte, sondern in den Arbeitsspeicher geschrieben. Dies soll den Zugriff auf einen Cloud Anbieter simulieren.

Für den Prototyp wurde die Klasse 'Directory.php' und 'File.php' komplett überschrieben. Die neue Implementation gibt den Inhalt eines Arrays zurück und nicht den Inhalt eines Verzeichnisses. Mit dem Test konnte gezeigt werden, dass es ohne grossen Aufwand möglich ist, den Server anzupassen.

```php
<?php

use Sabre\DAV;
require 'SabreDAV/vendor/autoload.php';

// Pfad der Datenablage
$rootDirectory = new DAV\FS\Directory('public');

// WebDAV Server-Object erstellen
$server = new DAV\Server($rootDirectory);

// Server URI ausgehend vom Webroot
$server->setBaseUri('/server.php');

// Server starten
$server->exec();

?>
```

Minimaler PHP Code

```php
<?php

// Locking Plugin
$backend = new DAV\Locks\Backend\File('locks');
$plugin = new DAV\Locks\Plugin($lockBackend);
$server->addPlugin($lockPlugin);

// Browser Plugin
$server->addPlugin(new DAV\Browser\Plugin());

// Benutzer Authentifizierung
use Sabre\DAV\Auth;
$authBackend = new Auth\Backend\File('passlist');
$authBackend->setRealm('SabreDAV');
$authPlugin = new Auth\Plugin($authBackend);
$server->addPlugin($authPlugin);

?>
```

Verwendung von Plugins in SabreDAV

8.3. Tomcat - WebDAVServlet

Apache Tomcat bietet ein vollständig implementierten WebDAV Servlet an. Dieses kann mit einer einfachen Konfiguration laufen gelassen werden. Konfiguriert wird das WebDAV Servlet in einem „Web Application Project“. Zuerst wird diesem Projekt ein Tomcat-Server zugewiesen. Danach ist eine Konfiguration in der 'web.xml' notwendig. Die entsprechende Standardkonfiguration ist im Javadoc von der Klasse „WebDAVServlet“ [11] beschrieben. Die Klasse WebDAVServlet ist der WebDAV Server welcher von Apache Tomcat standardmässig zur Verfügung gestellt wird. Nachdem die entsprechende Konfigurationen im „web.xml“ gemacht wurden, kann der Tomcat-Server gestartet werden und der WebDav Server von Tomcat läuft (standardmässig auf „localhost:8080“).

8.3.1. Funktionsumfang

Das WebDAV Servlet bietet alle Funktionalitäten (PROPFIND, MKCOL, LOCK, etc.) eines WebDAV Servers uneingeschränkt. Mit zusätzlichen Konfigurationen im „web.xml“ können z. B. Benutzer-Rollen erstellt werden, welche den Zugriff auf den WebDAV URLs regelt.

8.3.2. Anpassungsmöglichkeit

Es besteht die Möglichkeit die WebDAVServlet-Klasse zu erweitern und nur mit den nötige Funktionalitäten zu ergänzent. Die WebDAVServlet-Klasse ist jedoch riesig (über 3000 Code-Zeilen) und unübersichtlich. Dies erschwert das nötige Verständnis zu sammeln, welche für das erweitern der Klasse von wichtiger Bedeutung ist.

8.3.3. Lizenz

Der WebDAVServlet ist vollständig unter der Open-Source Lizent „Apache 2“ [17] veröffentlicht. Dies ermöglicht Anpassungen am Code vorzunehmen und diese wieder zu veröffentlichen.

8.3.4. Prototyp

Der WebDAV Proxys (Prototyp) soll nun auf dem WebDAV Server (WebDAVServlet-Klasse) basieren. Hierbei erweitert eine Java-Klasse „WebDAVServlet“ und erweitert die einzelne WebDAV-Funktionalitäten mit zusätzlichen Features.
Als Prototyp wurde eine Klasse namens „WebDAVProxy“ entwickelt, welche die WebDAVServlet-Klasse erweitert. Um die Machbarkeit eines WebDAV Proxys zu testen und zu demonstrieren wurden folgende Funktionen erfolgreich eingebaut:

- Erweiterung mit Logging von allen WebDAV-Methoden (GET, PUT, MOVE, DELETE, PROPFIND, COPY etc.) bei einem entsprechenden Request. Hiermit kann sichergestellt werden, dass der WebDAV Proxy Zugriff auf alle Reqests hat, um diese zu manipulieren.
- Logging des Dateiinhalts beim Hochladen (PUT) einer Datei. Dies ist notwendig für die spätere Anwendung des RAID5-Verfahrens.
- Manipulation einer Datei durch den WebDAV Proxy beim hochgeladen (PUT). Dies diente zur Demonstration, dass eine Datei beim Hochladen in mehrere Datenblöcke aufgeteilt werden kann. Dies ist notwendig für die Anwendung des RAID5-Verfahrens und für die Verschlüsselung.

8.4. Vergleich

Nach der Evaluation der drei ausgewählten Technologien, musste eine davon ausgesucht werden. Für die Entscheidung wurde eine Vergleichstabelle erstellt, bei welcher die drei Technologien miteinander verglichen und bewertet werden. Für die Bewertung wurde eine Skala von 1 bis 10 verwendet. 1 ist die schlechteste und 10 die beste Bewertung. Damit die Priorität der Evaluationskriterien berücksichtigt werden konnte, wurde eine Gewichtung miteinbezogen. Hierfür wurden pro Tabelle 100 Punkte vergeben. Für die Zwischenresultate wurde jeweils die Gewichtung der Kriteren mit den vergebenen Punkte multipliziert und pro Technologie aufsummiert. Manche Anforderungen haben keinen Einfluss auf die Wahl des WebDAV Servers. Diese wurden mit „technologieunabhängig“ gekennzeichnet.

Für den Vergleich wurden zwei verschiedene Tabellen erstellt. In der Ersten werden die WebDAV-Frameworks auf allgemeine Kriterien miteinander verglichen und in der Zweiten werden diese basierend den Projektanforderungen bewertet.

8.4.1. Allgemeiner Vergleich

Kriterium	Gewichtung	sabreDAV	WebDAVServlet	Milton CE
Funktionsumfang	5	10	8	5
Technologiekenntnis	20	3	9	9
Anpassungsfähigkeit	30	10	10	10
Einfachheit/Verständlichkeit	15	6	4	9
Unterstützung/Community	15	8	7	7
Dokumentation	15	8	8	7
Zwischenergebnis		**740**	**805**	**850**

Tabelle 3: Evaluation - Allgemeiner Vergleich

8.4.2. Projektspezifischer Vergleich

Kriterium	Gewichtung	sabreDAV	WebDAVServlet	Milton CE
WebDAV Proxy	15	6	6	9
Ausfallsicherheit	15	*technologieunabhängig*		
Authentifizierung	13	10	10	10
Datenverschlüsselung	13	*technologieunabhängig*		
Dateiproperties	11	10	7	10
Link-Sharing	11	*technologieunabhängig*		
Konfigurationsinterface	8	*technologieunabhängig*		
Multi-User Support	4	8	4	10
Data-Sharing	4	*technologieunabhängig*		
Datei-Historisierung	4	*technologieunabhängig*		
CalDAV	1	10	7	1
CardDAV	1	10	7	1
Zwischenergebnis		**382**	**327**	**409**

Tabelle 4: Evaluation - Projektspezifischer Vergleich

8.5. Bewertung

Dieser Kapitel bezieht sich auf die Vergleichtabelle aus dem vorherigen Kapitel. Hier wird auf manche Bewertungen noch genauer eingegangen.

8.5.1. Funktionsumfang

Beim Funktionsumfang geht es darum, welche Funktionen von den jeweiligen WebDAV Frameworks angeboten werden (CalDAV, CardDAV, Mulit-User Support, etc.):

SabreDAV SabreDAV bietet sehr viele Funktionen, welche über ein Plugin-System einfach hinzugefügt werden können. Aus diesem Grund wurde die maximale Anzahl Punkte vergeben.

WebDAVServlet Auch die WebDAVServlet-Klasse bietet viele Funktionalitäten wie Authentisierung, CalDAV und CardDAV. Das hinzufügen dieser Funktionalitäten ist über die richtige Konfiguration im „web.xml“ möglich. Dieses Hinzufügen von Funktionen ist schwieriger als bei SabreDAV. Deshalb hat das WebDAVServlet nur 8 von 10 Punkte erhalten.

Milton CE In der Milton CE (Community Edition) werden einzelne Funktionen nicht angeboten (z.B. Locking, CalDAV und CardDAV). Hierfür wäre die kostenpflichtige Edition (Enterprise Edition) von Milton notwendig.

8.5.2. Einfachheit/Verständlichkeit

Einfachheit und Verständlichkeit wird bewertet, um die Zeit für die Einarbeitung in die Technologie so gering wie möglich zu halten.

8.5.3. Unterstützung/Community

Damit ist gemient, wie gross die Community ist. Dieser Punkt ist wichtig, um genügend Hilfestellung bei Problemen zu bekommen. Alle drei untersuchten Technologien schneiden in diesem Punkt in etwa änlich gut ab.

8.6. Entscheidung

In beiden Vergleichstabellen schneidet Milton am besten ab. Deshalb wird für die Implementierung des WebDAV Proxys die Milton Annotation Framework (Milton Library) verwendet.

9. Algorithmen

In diesem Kapitel wird auf die Implementierungsdetails des WebDAV Proxys eingegange. Es wird erklärt, wie die Verschlüsselung der Daten, der Lese- und Schreibprozess (inklusive Ausfallsicherheit), die Berechnung der Blocklänge sowie das Dateisystem funktionieren.

9.1. Daten Verschlüsselung und Entschlüsselung

Für die Verschlüsselung der Datenblöcke wird der Advanced Encryption Standard (AES) verwendet. Als Betriebsmodus wurde der Counter Mode (CTR Mode) ausgewählt. Dadurch müssen die Datenblöcke nicht gleich gross oder ein vielfaches der Key-Länge sein. Der Verschlüsselungsalgorithmus wurde nicht selbst implementiert, da dies ein potentielles Sicherheitsrisiko darstellte. Statdessen wurden auf die Java Crypto Klassen zurückgegriffen.

Die Datenblöcke eines Slices werden jeweils einzeln verschlüsselt oder entschlüsselt. Durch das Parallelisieren der einzelnen Datenblöcken kann besonders bei grossen Blocklängen eine Performancesteigerung erzielt werden. Dadurch dass die Datenblöcke unabhängig voneinander verarbeitet werden können, besteht keine Gefahr durch Nebenläufigkeit.

Um unnötige Kopieroperationen während der Verschlüsselung und Entschlüsselung der Daten zu vermeiden, werden diese Operationen in-place durchgeführt. Das bedeutet, dass die einzelnen Bytes direkt im bereits existierenden Buffer manipuliert werden.

9.1.1. Verschlüsselung

Bei der Verschlüsselung wird zuerst ein neuer paralleler Tasks pro Datenblock gestartet. Jeder Task generiert mithilfe der Java KeyGenerator-Klasse einen eigenen Key, mit welchem dann der entsprechende Datenblock verschlüsselt wird. Damit ein Datenblock wieder entschlüssel werden kann, muss der verwendete Key und der Initialisierungsvektor (IV) im Datenblock Objekt abgespeichert werden. Der IV wird automatisch von der Java Crypto Klasse berechnet.

Damit der Benutzer die verwendeten Keys nicht selbst verwalten muss und diese vor Datenverlust geschützt sind, werden sie zusammen mit den Daten auf die Cloud Anbietern geschriben. Aus diesem Grund muss der Key von jedem Datenblock in seinem benachbarten Datenblock abgespeichert werden, sobald alle Tasks ordnungsgemäss beendet wurden. Dieser Key-Austausch ist nötig, damit beim Speichern eines Datenblocks nicht sein eigener Key mitgeschrieben wird. Ein Cloud Anbieter ist somit nur im Besitzt von Fremdschlüsseln, die zu Datenblöcken bei anderen Anbietern gehören. Da die Sicherheit von AES nicht von der Geheimhaltung des IV abhängig ist, muss dieser nicht vertauscht werden. Ein einzelner Cloud Anbieter kann somit nichts mit den bei ihm gespeicherten Daten anfangen.

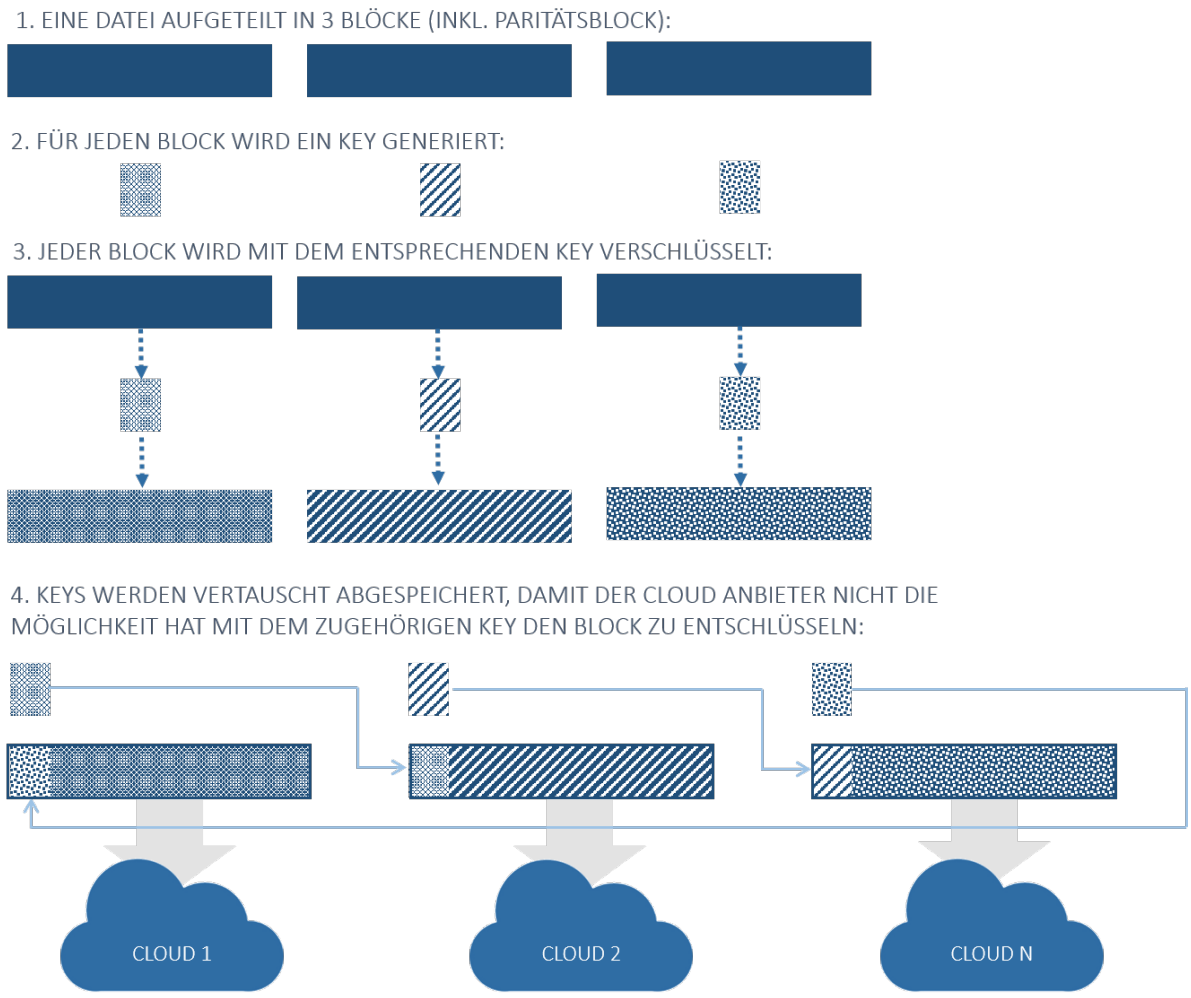

Abbildung 7: Key Austausch bei 3 Cloud Anbietern

9.1.2. Entschlüsselung

Die Entschlüsselung funktioniert genau umgekehrt. Zuerst müssen die Fremdschlüssel zurück in die dazugehörenden Datenblöcke kopiert werden. Anschliessend kann für jeden Datenblock im Slice ein eigener Task für die eigentliche Entschlüsselung gestartet werden. Die Daten können anhand des Keys und des IV wieder in ihre ursprüngliche Form gebracht werden. Zum Schluss muss nur noch auf die Beendigung aller Tasks gewartet werden.

9.2. Speicherung von Daten

In diesem Kapitel wird erläutert, wie der Schreibprozess der RAID5 Komponente im Detail funktioniert.

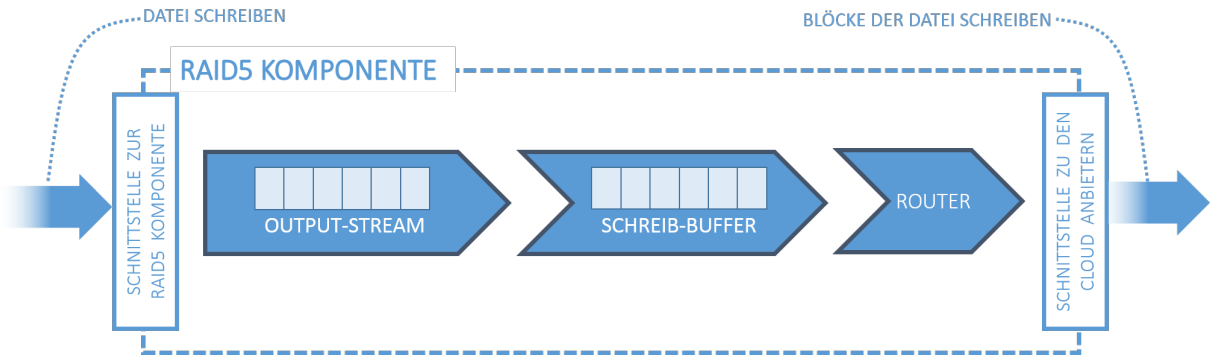

Abbildung 8: Schreibprozess

9.2.1. Output-Stream

Die RAID5-Komponente liefert einen Output-Stream, über welchen die Daten geschrieben werden können. Damit bei einem Ausfall eines Cloud Anbieters die Daten wiederhergestellt werden können, müssen diese immer als kompletter Slice von Datenblöcken abgespeichert werden. Das bedeutet, dass pro Slice bei jedem Cloud Anbieter ein Datenblock gespeichert wird. Einer dieser Blöcke ist die Parität im Slice.

Bevor die Parität von einem Slice berechnet werden kann, müssen alle Datenblöcke im Slice bekannt sein. Aus diesem Grund landen die zu schreibenden Daten zuerst in einem Buffer. Sobald dieser Buffer voll ist, sind die Daten vom Slice bereit zur Weiterverarbeitung. Am Ende des Schreibprozesses kann der Buffer für die Aufnahme der nächsten Daten zurückgesetzt werden. Die Länge des Buffers wird anhand folgender Formel berechnet:

```
DefaultBlockSize * (ProvidersCount - 1)
```

Über die DefaultBlockSize wird bestimmt, wie gross ein Datenblock standardmässig sein soll. Das -1 in der Formel stellt den Platz dar, welcher für die Parität reserviert wird. Somit ist der Buffer in der Lage, pro Cloud Anbieter einen Datenblock aufzunehmen. Dieser Buffer bedeutet zwar zusätzliche Kopieroperationen der einzelnen Bytes und somit eine kleine Verzögerung, dennoch ist dessen Verwendung und die Begrenzung der Länge sehr wichtig. Denn würden die Dateien auf einmal in das RAM geladen, könnte bei grossen Dateien der Speicherplatz ausgehen.

Wenn der Output-Stream geschlossen wird, obwohl es noch Daten im Buffer hat, müssen diese vor dem Schliessen noch auf die Cloud Anbieter geschrieben werden. Dieses Verhalten kann in den folgenden zwei Situationen eintreten. Einerseits wenn die zu schreibende Datei kleiner ist als die Grösse des Buffers und dieser somit nie voll wird. Andererseits wenn im letzten Slice einer Datei weniger Bytes vorhanden sind als in den vorangegangenen, wird der Buffer ebenfalls nicht voll. In diesen Situationen ist es nötig, die Blocklängen anzupassen und bei bedarf die Blöcke mit Padding Bytes aufzufüllen.

9.2.2. Splitten der Daten in Blöcke

Damit bei der Einteilung der Daten in Blöcke keine unnötigen Kopieroperationen der Bytes vorgenommen werden müssen, wird jedem Block-Objekt die selbe Refferenz auf den Buffer mit den Daten zugewiesen. Der Zugriff auf die Daten eines spezifischen Blocks erfolgt von nun an über einen Offset, mit welchem die Startposition im Buffer bekannt gegeben wird. Ausserdem wird die Anzahl der Nutzbytes und die Anzahl der Padding Bytes gespeichert, damit das Ende des Blocks im Buffer berechnet werden kann. Nachdem die Daten aus dem Buffer aufgeteilt sind, können die einzelnen Datenblöcke verschlüsselt werden. Wie das Block Shrinking- und Padding- im Detail funktioniert, wird zu einem späteren Zeitpunkt erläutert. Für das weitere Verständnis ist dieses Wissen nicht nötig.

9.2.3. Metainformationen

Damit die Daten bei den Cloud Anbietern auch wieder ausgelesen und korrekt zusammengefügt werden können, sind zusätzliche Informationen nötig. Diese Daten werden als Metainformationen bezeichnet. Abbildung 9 stellt die Struktur der Metadaten aus der RAID5 Komponente dar. Zu den Metadaten gehören die Anzahl verwendeter Padding Bytes sowie der Key und der Initialisierungsvektor, welche für die Entschlüsselung benötig werden. Damit die Metainformationen zusammen mit dem dazugehörigen Datenblock abgespeichert werden können, müssen sie zuerst zu einem Byte Array serialisiert werden. Danach können sie vor die Nutzdaten gesetzt werden. Das geschieht jedoch erst direkt vor dem Schreiben der Daten, damit weniger Kopieroperationen durchgeführt werden müssen.

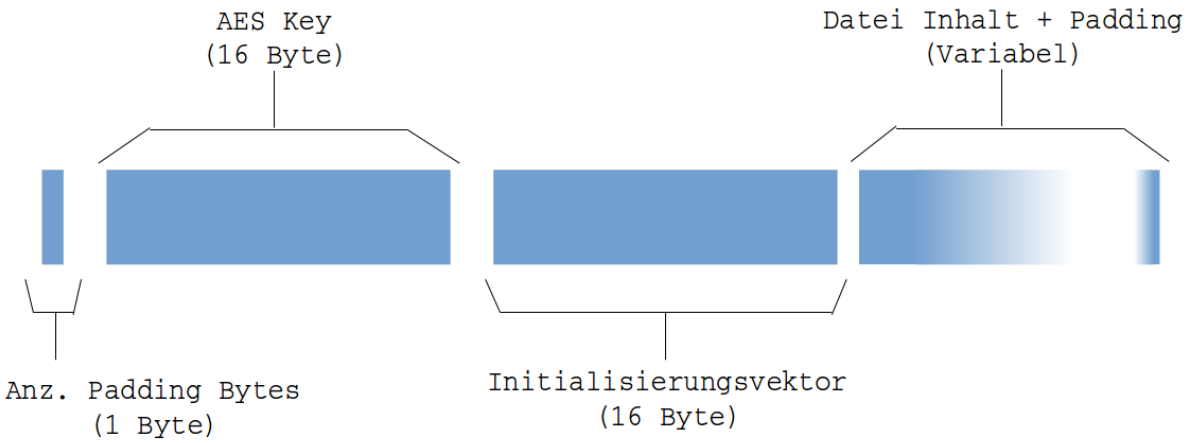

Abbildung 9: RAID5 Metadaten

9.2.4. Berechnung der Parität

Bevor die Parität geschrieben werden kann, muss sie zuerst berechnet werden. Hierfür wird ein Buffer mit der Länge der Metainformation + der Länge eines Blocks aus dem aktuellen Slices erstellt. Die Metainformationen der Datenblöcke müssen für die Parität berücksichtigt werden, ansonsten würden diese Informationen bei einem Cloud Anbieter Ausfall verloren gehen. Desshalb werden die Metainformationen der ersten beiden Blöcken

aus dem Slice serialisiert und zwischengespeichert. Über die Metainformationen wird eine XOR-Operation durchgeführt und das Resultat wird in den Buffer gespeichert. Der restiche Platz des Buffers wird für das Resultat einer weiteren XOR-Operation benötigt. Dieses Mal allerdings für die Nutzdaten der ersten beiden Blöcke. Für jeden weiteren Block im Slice müssen mit dem Zwischenresultat im Buffer ebenfalls die serialisierten Metainformationen und Nutzdaten mittels XOR-Operationen verrechnet werden. Das Resultat am Ende ist die Parität. Die Berechnung wird direkt vor dem Schreiben der Parität durchgeführt.

9.2.5. Verteilung und Speicherung der Daten

Nach diesen Prozessen ist der Slice mit seinen Blöcken für die Verteilung und Speicherung auf den Cloud Anbietern bereit. Das folgende Flussdiagramm stellt diesen Vorgang etwas vereinfacht dar.

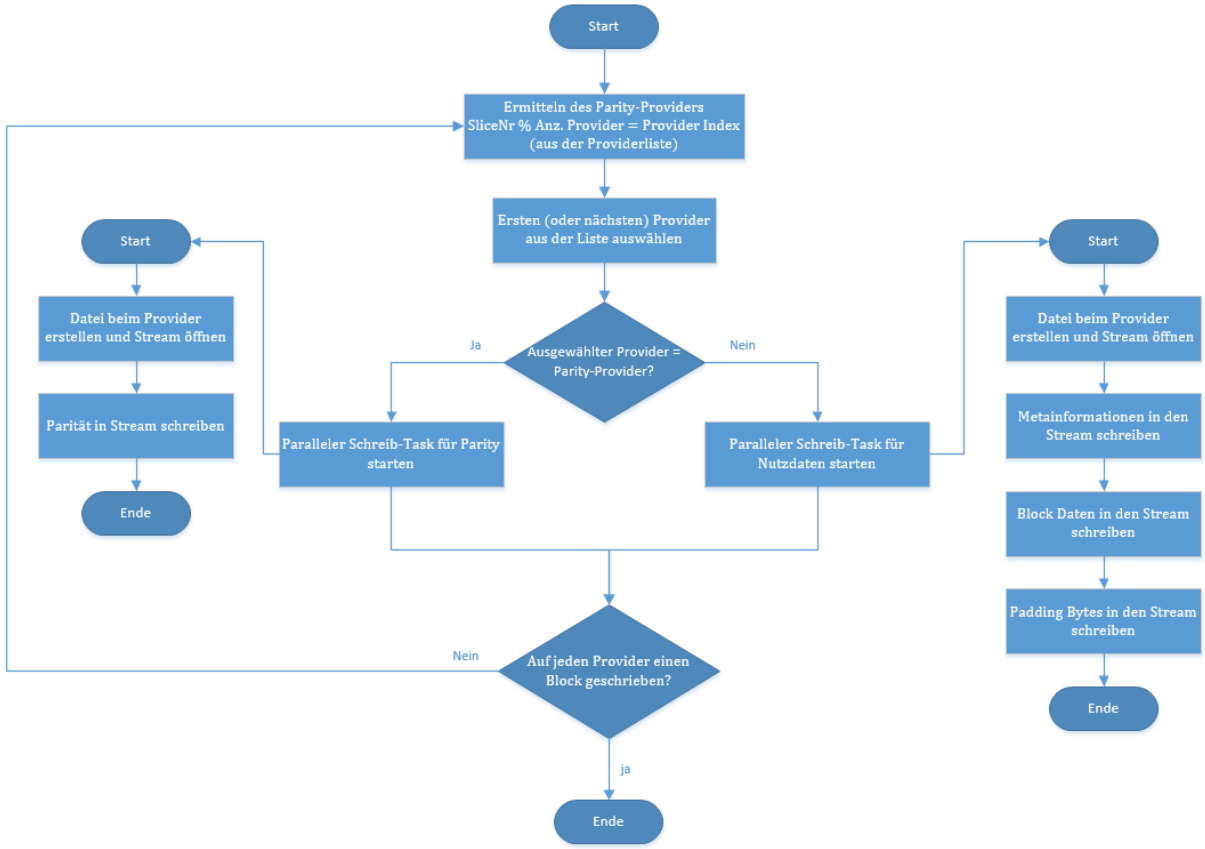

Abbildung 10: Algorithmus zur Verteilung und Speicherung der Daten

Der eigentliche Schreibprozess wird für jeden einzelnen Datenblock als paralleler Task gestartet. Hierfür wird ein ForkJoinPool mit so vielen Worker Threads gestartet, wie Prozessorkerne zur Verfügung stehen. Durch diese Parallelisierung kann der Umstand, dass jeder Cloud Anbieter unabhängig seine Daten schreiben kann, ausgenutzt und einen Performance Vorteil herausgeholt werden. Wenn der Output-Stream geschlossen wird, bedeutet dies, dass alle Bytes der zu schreibenden Datei verarbeitet wurden. Es kann aber durchaus passieren, dass noch nicht alle Schreib-Tasks beendet wurden. Aus diesem Grund muss noch gewartet werden, bevor der Schreibprozess als beendet betrachtet werden kann.

Ausserdem muss noch überprüft werden, ob bei einem Schreib-Task ein Fehler passiert ist und dementspechend muss anhand einer Exception der Schreibvorgang beendet werden. Wenn keine Fehler passiert sind, kann der Schreibvorgang erfolgrich beendet werden.

9.3. Block Shrinking

Dieses Kapitel beschreibt, wie die Daten in Blöcke aufgeteilt werden.

9.3.1. Block Size

In diesem Kapitel geht es um die Länge eines Datenblocks. Damit die Paritäten der Datenblöcke berechnet werden können, müssen alle Blöcke in einem Slice gleich lang sein. Wenn zu wenig Daten vorhanden sind, um einen ganzen Slice zu füllen, müssen die Datenblöcke mit Padding Bytes aufgefüllt werden.

Für kleine Dateien eignen sich kleine Blocklängen am besten, denn dadurch müssen die Datenblöcke mit weniger Padding Bytes aufgefüllt werden und es wird bei den Cloud Anbietern Platz gespart. Was für kleine Dateien gut ist, ist für grosse Dateien schlecht. Bei kleinen Blocklängen müsste eine grosse Datei in sehr viele Datenblöcke unterteilt werden. Durch die Verschlüsselung und das Transportieren vieler Datenblöcke über das WebDAV Protokoll entstehen Verzögerungen, welche mit grösseren Blocklängen verhindert werden könnten.

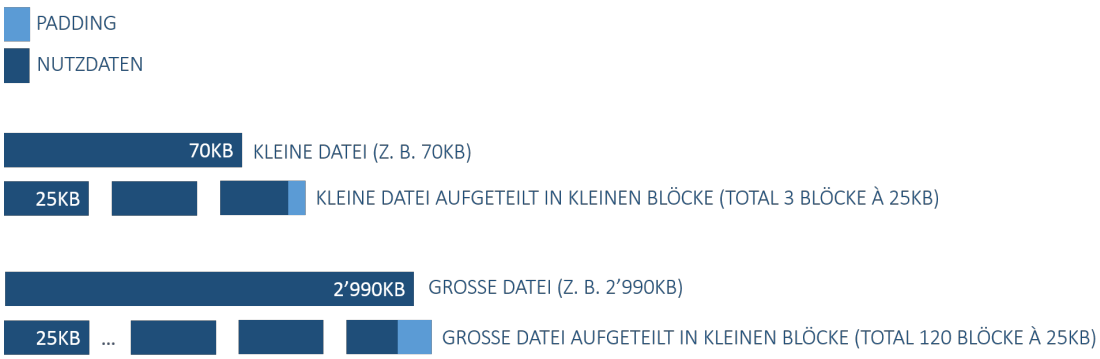

Abbildung 11: Effizienz bei kleine Blocklänge (3 Cloud Anbieter)

Für grosse Dateien sind grosse Blocklängen von Vorteil. Dadurch kann die zur Verfügung stehende Bandbreite effektiver genutzt werden. Für die kleine Dateien ist die Effizienz jedoch schlecht. Durch Padding wird unnötiger Speicher verbraucht, welcher bei kostenlosen Cloud Anbietern nur begrenzt vorhanden ist.

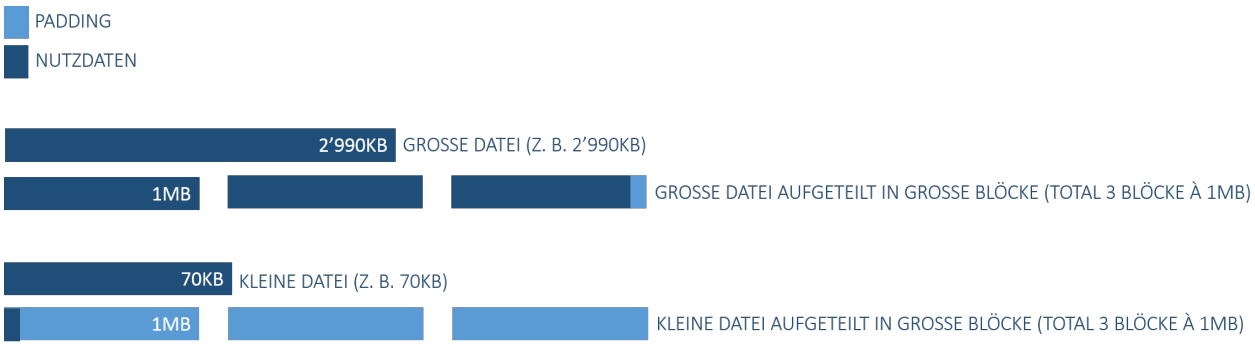

Abbildung 12: Effizienz bei grossen Blocklänge (3 Cloud Anbietern)

9.3.2. Dynamic Block Size

Damit man sich nicht für das kleinere Übel entscheiden muss, wird eine Kombination der beiden Optionen verwendet: Die Länge der Datenblöcke wird dynamisch gehalten. Der Benutzer legt eine „Default Blocksize“ fest. Wenn genügend Daten vorhanden sind, um die Datenblöcke dieser Länge zu füllen, wird dies gemacht. Wenn nicht, wird die Blocklänge optimiert, sodass möglichst wenig Padding nötig wird. Aus historischen Grüden wird auch eine „Minimum Blocksize“ verwendet, welche die minimale Blocklänge definiert. Dies kommt von der Verschlüsselung, als noch eine andere Betriebsart vom AES verwendet wurde. Dabei mussten die Blöcke genau so lang sein wie der verwendete Key oder ein vielfaches davon. Die Minimum Blocksize könnte somit komplett entfernt werden.

9.3.3. Block Shrinking Algorithmus

Um die optimale Blocklänge zu berechnen, wird immer die „Default Blocksize“ als Startgrösse genommen. Da nicht bekannt ist, wie gross die zu schreibende Datei ist, wird geprüft, ob ein Slice mit der „Default Blocksize“ gefüllt werden kann. Die Anzahl Daten-Blöcke pro Slice wird anhand Folgender Formel berechnet:

$$\boxed{StandardSliceSize = DefaultBlockSize * (ProvidersCount - 1)}$$

Wenn der Slice nicht mit der „Default Blocksize“ gefüllt werden kann, wird die Länge verkleinert. Im folgenden Flussdiagramm ist der Algorithmus beschrieben, welcher die optimale Blockgrösse für eine Datei berechnet.

Bei einer grossen Datei (grösser als die Standard-Slice-Size), werden zuerst so viele Slices wie möglich mit der „Default Blocksize“ gefüllt. Erst wenn der Rest der Datei keinen Slice mehr mit der „Default Blocksize“ füllen kann, wird die Blocklänge verkleinert. Bei einer kleinen Datei (kleiner als die Standard-Slice-Size) wird bereits am Anfang die optimale Blockgrösse berechnet.

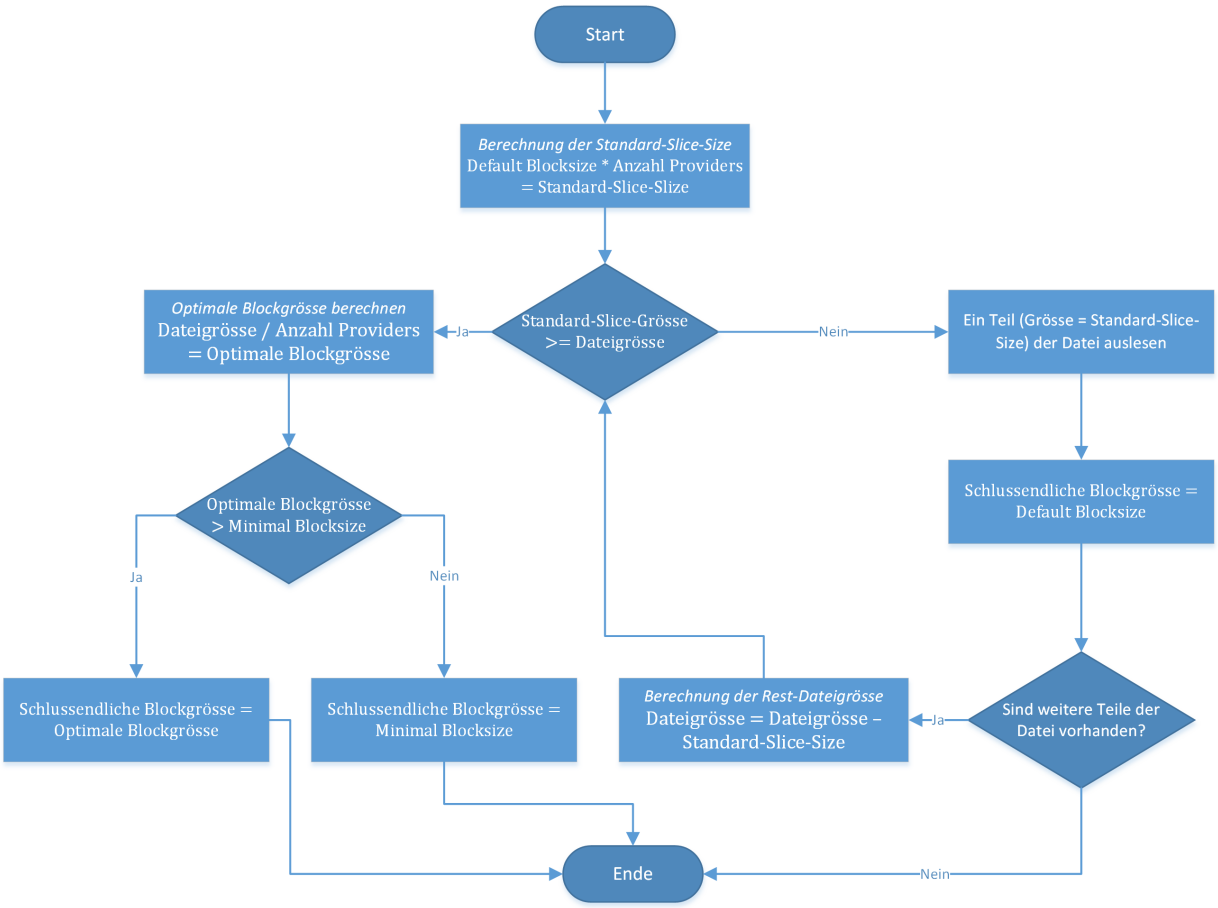

Abbildung 13: Block Shrinking Algorithmus

9.4. Daten Lesen

In diesem Kapitel wird erläutert, wie der Leseprozess der RAID5 Komponente im Detail funktioniert. Dazu gehöhrt auch das Verhalten bei einem Ausfall eines Cloud Anbieters.

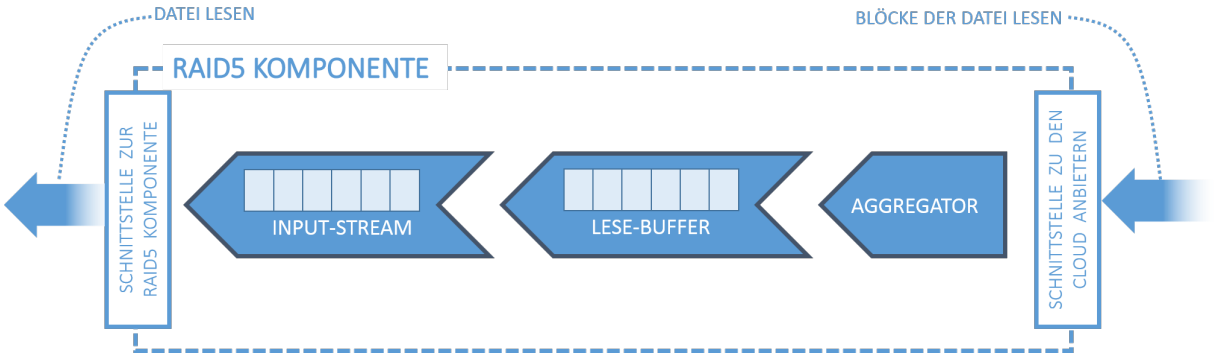

Abbildung 14: Leseprozess

9.4.1. Input-Stream

Die RAID5-Komponente liefert einen Input-Stream, über welchen die Daten gelesen werden können. Beim Aufrufen der Read Funktion des Streams, wird über die Hilfsklasse RaidAggregator ein kompletter Slice ausgelesen. Die Datenblöcke im Slice sind noch nicht in Klartext vorhanden und müssen daher entschlüsselt werden.

Nach der Entschlüsselung wird der erste Datenblock vom Slice in einen Buffer geladen. Anschliessend können die einzelnen Bytes des Buffers zurückgegeben werden, bis dieser leer ist. In diesem Fall wird der nächste Datenblock in den Buffer kopiert. Wenn alle Bytes in einem Slice verarbeitet wurden, wird über die Hilfsklasse RaidAggregator der nächste Slice ausgelesen. Sobald alle Slices verarbeitet wurden, ist die Datei komplett ausgelesen.

9.4.2. RaidAggregator

Der eigentliche Lese-Algorithmus wird in der Hilfsklasse RaidAggregator implementiert. Der Vorgang kann in zwei Teilprozesse unterteilt werden: Das Auslesen der einzelnen Datenblöcke bei den Cloud Anbietern und das Zusammenstellen der ausgelesenen Datenblöcke in der richtigen Reihenfolge. Die Ausfallsicherheit ist ein Bestandteil des Auslesens.

Da die Cloud Anbieter unabhängig voneinander ausgelesen werden können und dies je nach Anbieter durchaus etwas länger dauern kann, ist eine Parallelisierung dieses Vorgangs naheliegend. Mit dieser Parallelisierung kann eine spürbare Performancesteigerung erzielt werden. Die ausgelesenen Blöcke werden in Blockierenden Queues für due Weiterverarbeitung abgelegt. Mit blockierend ist gemeint, dass beim Auslesen der Queues auf neue Einträge gewartet wird, sollten diese leer sein. Das Blockieren ist wichtig, da das Zusammenfügen der einzelnen Datenblöcke zu Slices nicht parallelisiert wird. Dieser Vorgang ist deutlich schneller als das Auslesen der Blöcke und bedarf keiner Parallelisierung.

Auslesen der Datenblöcke

Abbildung 15 stellt das Auslesen der Datenblöcke von den Cloud Anbietern als Flussdiagramm dar.

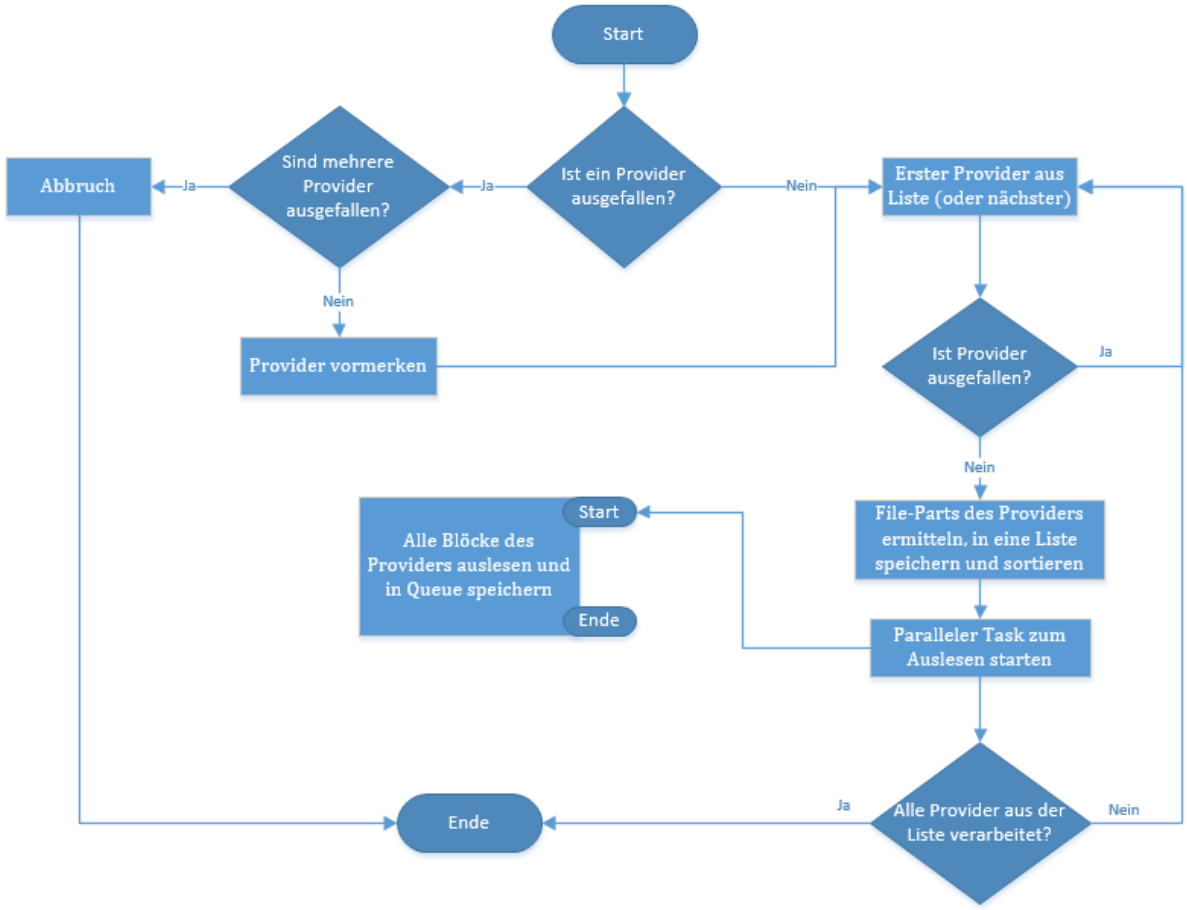

Abbildung 15: Algorithmus zum Auslesen der Datenblöcke

- Wenn ein Cloud Anbieter ausgefallen ist, kann dieser einfach übersprungen werden, da es nichts zum Auslesen gibt.
- Wenn kein Cloud Anbieter ausgefallen ist, können alle gefundenen Namen der Parityblöcke aus der Liste gelöscht werden. Da keine Wiederherstellung nötig ist, werden die Parityblöcke nicht benötigt und der Leseprozess kann gespart werden. Bei einem Ausfall eines Cloud Anbieters dürfen die Namen der Parityblöcke jedoch nicht glöscht werden.
- Damit die Datenblöcke bereits in der richtigen Reihenfolge ausgelesen werden, müssen die Namen der Blöcke bei den Cloud Anbietern in der Liste sortiert werden. Hierfür werden die Namen aufsteigend anhand ihrer Blocknummern sortiert. Die Parityblöcke kommen am Schluss.
- Damit der Sortiervorgang einfacher wird, erhält jeder Cloud Anbieter zwei blockierende Queues. Die eine Queue beinhaltet alle Datenblöcke, die andere alle Paritäten des Anbieters.

- Durch die Trennung der Datenblöcke und der Parityblöcke in unterschiedlichen Queues und die Sortierung der Blocknamen werden die ausgelesenen Datenblöcke automatisch in der richtigen Reihenfolge in die Queues geschrieben.

Zusammenfügen der Datenblöcke

Abbildung 16 stellt das Zusammenfügen der Datenblöcke zu Slices als Flussdiagramm dar.

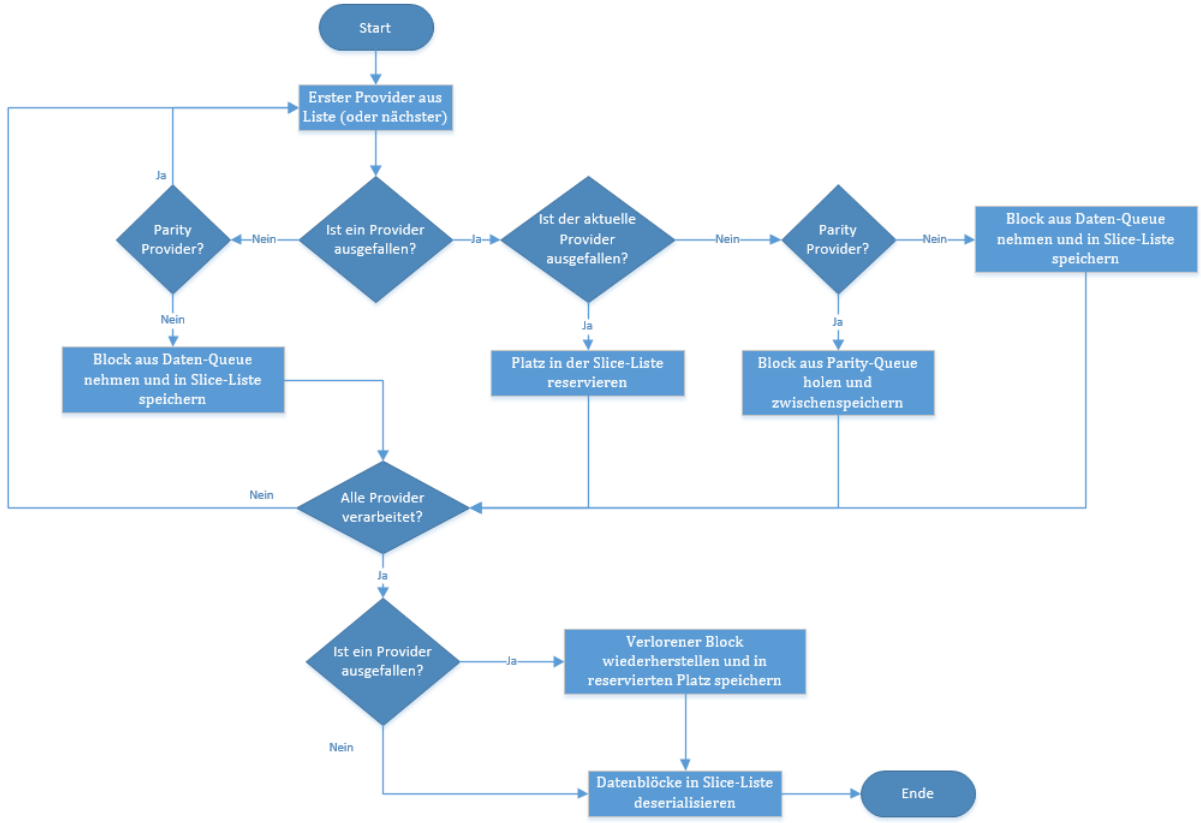

Abbildung 16: Algorithmus der Datenblock-Zusammensetzung

- Wenn kein Cloud Anbieter ausgefallen ist, müssen die Parityblöcke ignoriert werden, da diese nicht ausgelesen wurden und die Queues entsprechend leer sind.
- Wenn der aktuelle Cloud Anbieter ausgefallen ist und es sich nicht um den Parity Cloud Anbieter des Slices handelt, muss ein Platz in der Slice-Liste reserviert (null-Objekt speichern) und die Positon gemerkt werden. An dieser Stelle wird später der wiederhergestellte Block eingefügt. Dies wird so gemacht, damit die Reihenfolge der Datenblöcke bei einem Ausfall eines Cloud Anbieters nicht durcheinander gebracht wird.

Datenblock Wiederherstellung

Die Wiederherstellung eines verlorenen Datenblocks funktioniert gleich wie die Erstellung der Parität. Es werden einfach XOR-Operationen über alle Datenblöcke und die Parität durchgeführt. Das Resultat am Ende ist der verlorene Datenblock.

9.5. Dateisystem

Die RAID5 Komponente Speichert die Daten in einer flachen Hierarchie ab, das heisst es unterstützt keine Ordnerstruktur. Weiter sollen über die Dateien Metainformationen abgespeichert werden, wie Erstellungsdatum oder Dateigrösse. Um dem User über die WebDAV Schnittstelle trotzdem eine Ordnerstruktur anzubieten und die Metainformationen verwalten zu können, wurde ein Dateisystem entwickelt. Dieses verwaltet sowohl die Dateistrukturen, wie auch die dazugehörigen Metainformationen.

9.5.1. Dateien und Ordner

Um aus der flachen Hierarchie der RAID5 Komponente eine normale Ordnerstruktur bereitzustellen, wurden die Dateien in zwei logische Typen unterteilt: Dateien und Ordner. Die Dateien und Ordner werden mit Hilfe einer Universally Unique Identifier (UUID) eindeutig identifiziert. Die beiden Typen besitzen eine Type-ID um sie unterscheiden zu können. Die UUID wird als 36 Byte Zeichenkette gespeichert und die Type-ID als ein Byte.

Dateien Dateien speichern Daten und verwalten ihre Metainformationen. Sie können erstellt, überschrieben, umbenannt und gelöscht werden.

Ordner Ordner speichern Einträge über die in ihnen vorhandenen Dateien und Ordner und verwalten ihre eigenen Metainformationen. Ein Ordnereintrag besteht aus einer Type-ID und einer UUID. Weder zwischen Type-ID und UUID noch zwischen den einzelnen Einträgen gibt es Trennzeichen. Diese sind aufgrund der festen Grössen der Type-ID und UUID nicht nötig. Typisch sieht ein Eintrag wie folgt aus (hexadezimal):

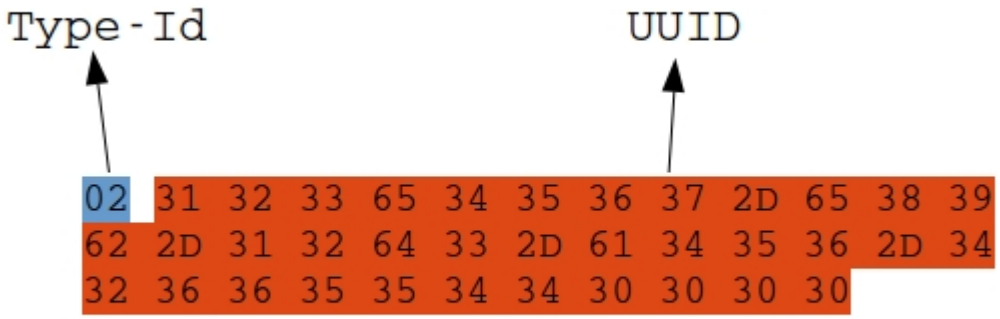

Abbildung 17: Ordnereintrag

Spezielle Strukturen im Dateisystem Der Einstiegspunkt für das Dateisystem ist ein Ordner mit der speziellen UUID 00000000-0000-0000-0000-000000000000. Weiter existiert ein Ordner in dem sich Dateien befinden, die entweder vom oder mit dem Benutzer geteilt

wurden. Dieser hat die UUID 00000000-0000-0000-0000-000000000001. Für Dateien die mit dem Benutzer geteilt wurden, gibt es einen speziellen Datentyp. Dieser Datentyp hat eine eigene Type-ID und ist eine Art Referenz auf ein anderes Dateisystem. Er speichert Informationen über die geteilte Datei und wie darauf zugegriffen werden kann und enthält selbst keine Metainformationen.

9.5.2. Metainformationen

Über die Dateien werden folgende Metainformationen gespeichert:

- Erstellungsdatum
- Veränderungsdatum
- Dateigrösse
- Dateiname
- UUID der Nutzdaten

Speicherung der Metainformationen Die Metainformationen und Nutzdaten der Dateien und Ordner werden getrennt gespeichert. Für jede Datei oder jeden Ordner existieren zwei Dateien auf der RAID5 Komponente. Die Datei mit den Metainformationen speichert dabei die UUID der Nutzdaten.

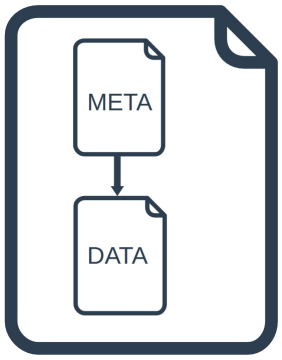

Abbildung 18: Struktur eines Files

Dies ist nötig, damit die Metainformationen - speziell der Dateiname - geändert werden kann, ohne dass die ganze Datei nochmals eingelesen und geschrieben werden muss.

Struktur der Metainformationen Die Metainformationen werden in folgender Struktur gespeichert:

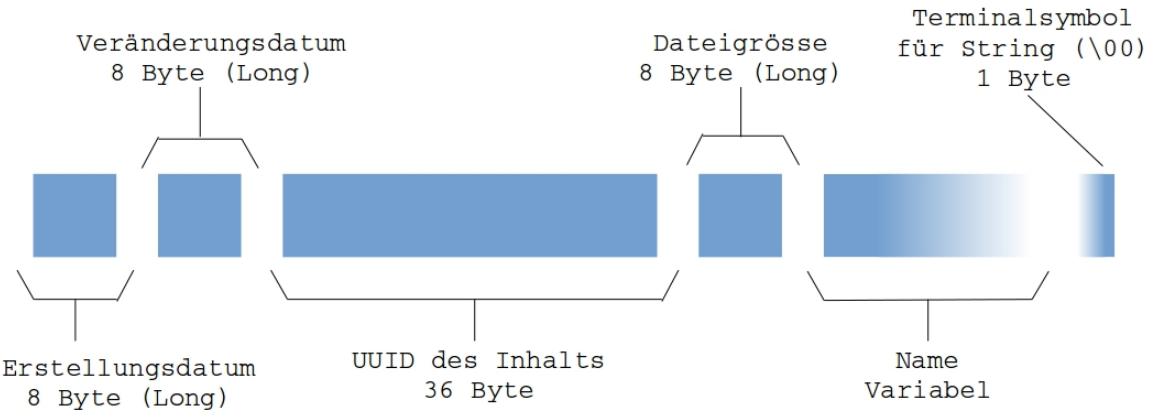

Abbildung 19: Struktur der Metainformationen

Meta-Attribut	**Beschreibung**
Erstellungsdatum	Das Erstellungsdatum wird bei der Erstellung der Datei oder des Ordners gesetzt. Es ändert sich danach nicht mehr.
Veränderungsdatum	Das Veränderungsdatum wird bei der Erstellung der Datei oder des Ordners gesetzt. Wird die Datei oder der Ordner danach umbenannt oder verändert (Datei wird überschrieben/-Ordner wird ein Eintrag geändert) wird das Veränderungsdatum aktualisiert.
UUID des Inhalts	Die UUID des Inhalts ist der Verweis auf den Speicherort der Nutzdaten der Datei oder des Ordners.
Dateigrösse	Die Dateigrösse wird für Dateien gespeichert und umfasst nur die Grösse der Nutzdaten. Bei Ordnern existiert dieses Feld auch, wird jedoch nicht verwendet.
Name	Der Name wird für Dateien und Ordner gespeichert. Dieser wird mit einem Terminalsymbol beendet (hexadezimal: `#00`).

9.5.3. Verschlüsselung

Diese Idee wurde für eine Verschlüsselung auf Ebene der RAID5 Komponente verworfen. Jedoch wurde ein Konzept erarbeitet, dass in diesem Abschnitt kurz erläutert wird.

Konzept Bei der Verschlüsselung auf der Dateisystemebene werden die Ordnereintäge so erweitert, dass sie zusätzlich zu jedem Eintrag einen Key speichern. Die dazugehörende Datei oder der Ordner wird mit diesem Key verschlüsselt.

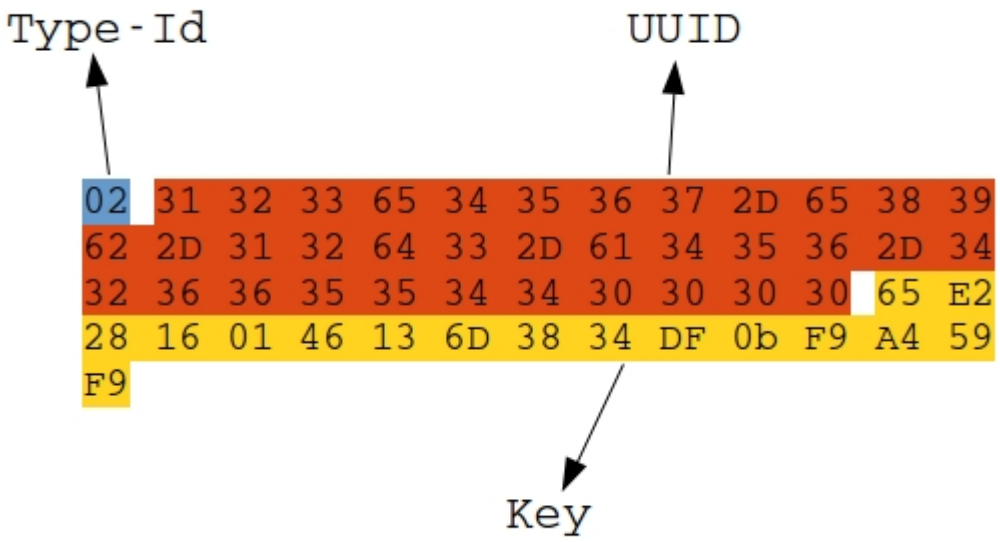

Abbildung 20: Struktur eines Files mit Anpassungen für Verschlüsselung

Das Root-Directory wird dann mit einem vom Benutzer gewählten Passwort verschlüsselt. So entsteht eine Baumstruktur, bei der jeder Ordner den Key für seinen Inhalt bereitstellt.

Sicherheit Bei diesem Verfahren ist die Sicherheit abhängig von der Verschlüsslungsmethode und dem gewählten Benutzerpasswort. Selbst wenn alle Dateien für einen Angreifer zugänglich wären, müsste dieser die Verschlüsselung des Root-Directories brechen.

9.6. Link Sharing

Der WebDAV Proxy bietet auch die Funktionalität, Dateien mit anderen WebDAV Proxy Nutzern zu teilen.

9.6.1. Features

Das Link-Sharing bietet folgende Features an:

- Datei teilen
  Indem man im WebDAV Proxy eine Datei in den Ordner „shared“ verschiebt wird diese Datei öffentlich gemacht. Die öffentliche Datei ist für jeden Benutzer (mit WebDAV Proxy und dem entsprechenden Link) zugreifbar.

- Link
  Damit ein Benutzer einem anderen den Zugriff auf eine freigegebene Datei ermöglichen kann, muss er sich über die URL „http://[WebDAV-Host]:8080/sharing“ den Link der freigegebenen Datei holen. Diesen Link kann er dann einem anderen Benutzer schicken. Der andere Benutzer kann dann über seinen eigenen WebDAV Proxy und dem Link auf die Datei zugreifen.

9.6.2. Konzept

Bei der Umsetzung des Link-Sharings musste weiterhin die Ausfallsicherheit und der Schutz der Daten vor Fremden zugriffen gewährleistet werden. Deshalb werden auch die geteilten Dateien in Blöcke geteilt, verschlüsselt und danach nach dem RAID5 Verfahren auf die verschiedenen Cloud Anbieter verteilt.

Die Basis für die Umsetzung des Link-Sharings bieten die Cloud Anbieter. Viele Cloud Anbieter stellen bereits eine Link Sharing Funktionalität zur Verfügung. Von diesen soll nun gebrauch gemacht werden.

Wenn eine Datei geteilt werden soll, dann muss jeder Block der Datei über einen öffentlichen Link geteilt werden. Der WebDAV Proxy kann über die jeweiligen öffentlichen Links auf alle Blöcke zugreifen und diese wieder zusammenführen. Dies bringt jedoch den Nachteil, dass es gleichviele Links wie Anzahl Blöcke von geteilten Dateien gibt. Ist zum Beispiel eine veröffentlichte Datei in 30 Blöcke aufgeteilt worden, dann werden 30 Links benötigt, um diese Datei mit einem anderen Benutzer zu teilen. Deshalb wurde ein Konzept entwickelt, welche pro geteilte Datei weniger Links benötigt, unabhängig von der Anzahl Blöcke.

9.6.3. Sharing-Konzept

Das Sharing-Konzept nutzt weiterhin die von den Cloud Anbietern zur Verfügung gestellte Sharing Funktionalität. Statt bei einem Cloud Anbieter jeden Block (von der geteilten Datei) einzeln zu Teilen, wird ein ganzer Ordner geteilt, welche alle Blöcke einer geteilten Datei enthält. Dieser geteilte Ordner stellt eine „Freigabe“ bzw. „Sharing“ dar.

9.6.4. Sharing-Konzept hinter den Kulissen

Anhand eines Beispiel wird ersichtlich, was im Hintergrund geschieht, wenn eine Datei geteilt wird:

1. Beim Initialen Zustand eines WebDAV Proxys existiert bereits ein Ordner Namens „shared“, welcher in diesem Dokument als shared-Ordner bezeichnet wird. In diesem Ordner befinden sich die Dateien, welche mit einem Link geteilt werden sollen (hier noch leer).

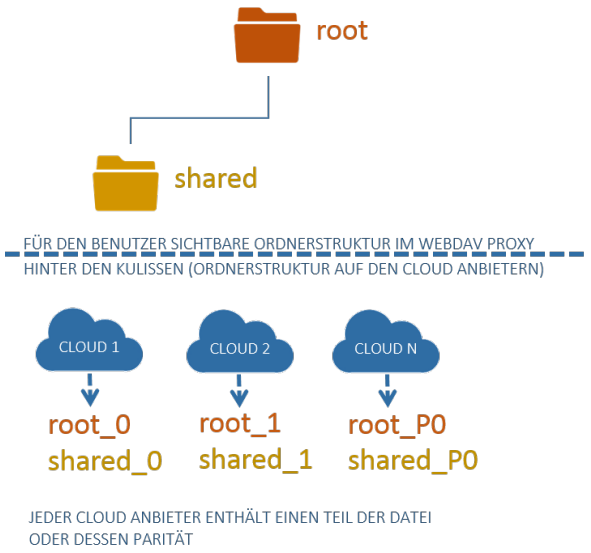

Abbildung 21: Initiale Ordnerstruktur vom WebDAV Proxy

2. Der Benutzer erstellt eine Datei, welche zum aktuellen Zeitpunkt noch nicht geteilt ist.

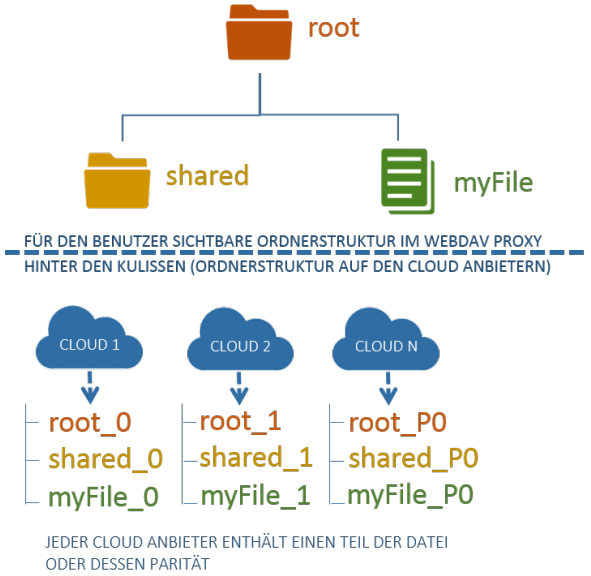

Abbildung 22: Ordnerstruktur vom WebDAV Proxy mit einer Datei

3. Der Benutzer teilt die Datei, indem er die Datei in den Ordner „shared“ verschiebt. Auf den jeweiligen Cloud Anbietern wurde nun eine neue Freigabe (geteilter Ordner) erstellt, in welche die Blöcke der Datei verschoben wurden. Dadurch spielt die Anzahl der Blöcke beim Teilen einer Datei keine Rolle mehr, da schlussendlich nur der Sharing-Link vom geteilten Ordner „sharing43“ nötig ist.

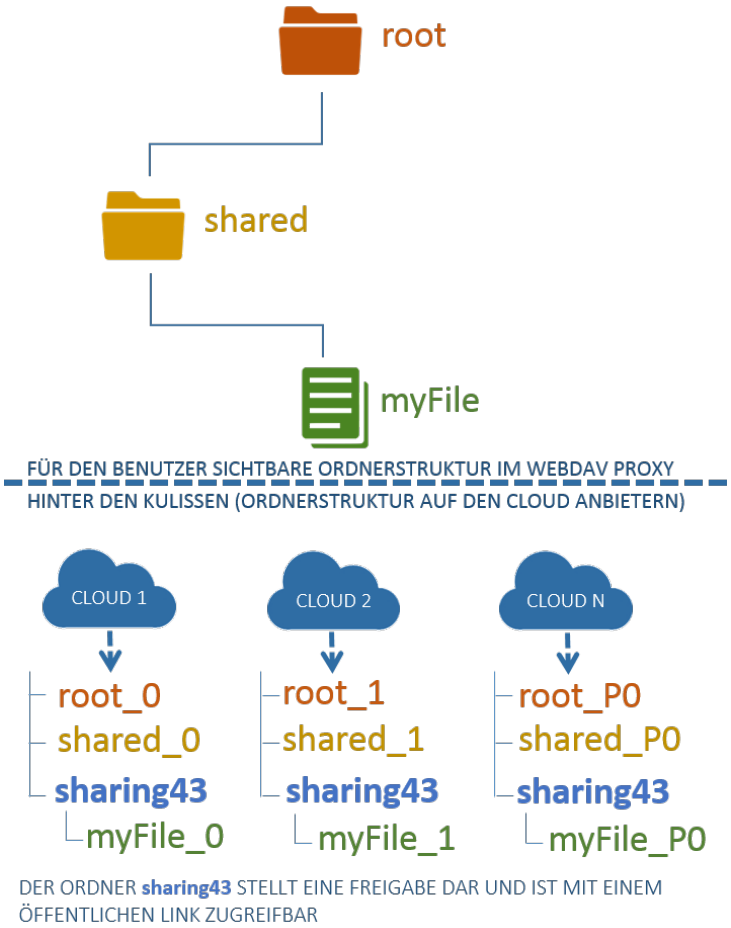

Abbildung 23: Ordnerstruktur vom WebDAV Proxy mit einer geteilten Datei

9.6.5. Links der geteilten Dateien

Gibt man in einem Internet Browser den Host oder die IP-Adresse und Port des WebDAV Proxys mit dem Pfad „/sharing“ an, werden alle vom Benutzer geteilten Dateien aufgelistet. Für jede geteilte Datei ist auch ein spezieller Link sichtbar, welcher der Benutzer kopieren und einem anderen Benutzer geben kann. Der andere Benutzer kann diesen Link seinem WebDAV Proxy mitgeben, welcher dank diesem Link in der Lage ist, auf die jeweilige Freigabe mit den entsprechenden Blöcken zuzugreifen.

10. Architektur

In diesem Kapitel wird für das bessere Verständnis zuerst ein grober Überblick der erarbeiteten Architektur gegeben. Anschliessend werden die einzelnen Komponenten genauer betrachtet.

10.1. Ideenfindung

Beim Konzipieren einer Architektur für den WebDAV Proxy wurde berücksichtigt, dass ein gewisses Risiko bezüglich des ausgewählten WebDAV Servers besteht. Im schlimmsten Fall wird eine Kernanforderung nicht erfüllt und es muss durch ein anderes WebDAV Server ersetzt werden. Deshalb wurde die WebDAV Erweiterung unabhängig vom WebDAV Server realisiert. Der WebDAV Server ist demzufolge eine Eigenständige Komponente im WebDAV Proxy.

Der WebDAV Server muss nun mit Funktionalitäten wie das Verteilen nach dem RAID5 Verfahren (für die Ausfallsicherheit) und Verschlüsselung (als Schutz vor dem Cloud Anbieter) erweitert werden. Für das Verteilen der Daten und für die Verschlüsselung waren zudem zwei verschiedene Komponenten notwendig. Aus folgenden Gründen besteht jedoch nicht die Möglichkeit, diese zwei Komponenten unabhängig voneinander zu modellieren:

- Die Keys müssen bei den Cloud Anbietern gespeichert werden, damit bei einem lokalen Datenverlust die Daten trozdem noch entschlüsselt werden können.
- Für die Kommunikation mit dem Cloud Anbieter ist die RAID5 Komponente zuständig, da nur die RAID5 Komponente darüber entscheiden kann, welche Datenblöcke bei welchem Cloud Anbieter landen.
- Die Dateien können nicht vor der RAID5 Komponente verschlüsselt werden, denn zu diesem Zeitpunkt existieren noch keine Datenblöcke. Man könnte höchstens eine Datei auf ein Mal verschlüsseln. Dazu müsste man jedoch die gesammte Datei in das RAM laden. Bei grossen Dateien würde das nicht funktionieren.
- Die Verteilung der Daten ist eine Aufgabe der RAID5 Komponente und kann von keiner anderen Komponente übernommen werden. Wenn die Verschlüsselungskomponente unabhängig implementiert werden würde, könnte diese die Daten nach dem Verschlüsseln nicht verteilen.

Deshalb wurde beschlossen, dass die Verschlüsselungskomponente als Teil der RAID5 Komponente zu implmentieren ist. Im nächsten Kapitel wird eine grobe Übersicht der gesammten Architektur gegeben.

10.2. Grobkonzept

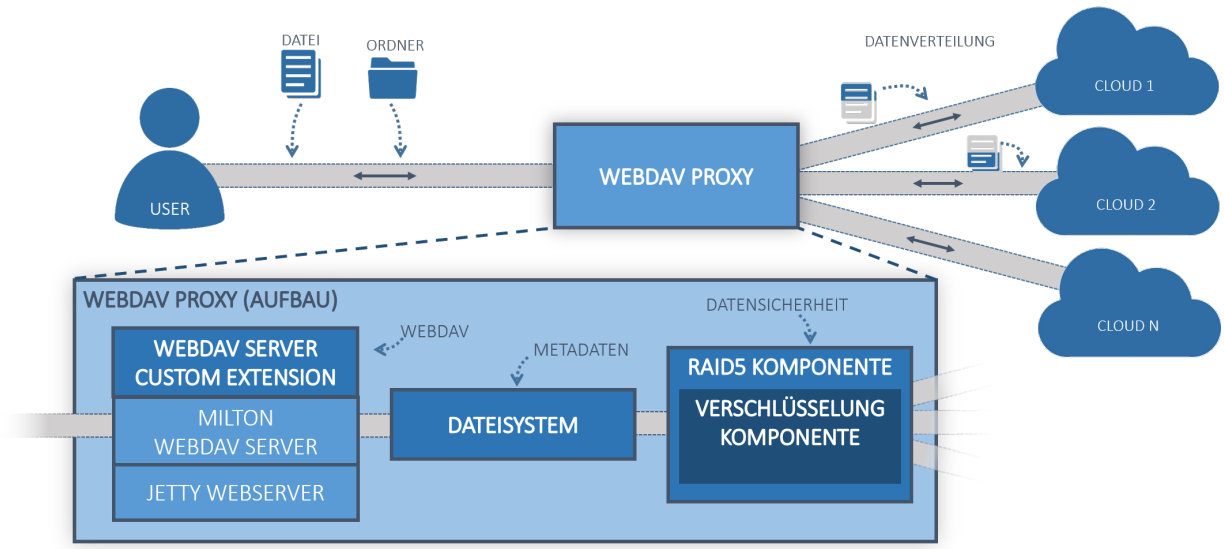

Abbildung 24: Architekturübersicht des WebDAV Proxys

Der WebDAV Server kann grundsätzlich in vier Komponenten unterteilt werden. Abbildung 24 soll dies veranschaulichen.

1. Milton WebDAV Server
   Da die Implementierung der grundlegenden Funktionen eines WebDAV Servers nicht Bestandteil der Aufgabenstellung war, wurde beschlossen hierfür den Milton WebDAV Server zu verwenden. Dieser läuft auf einem Jetty Webserver und dient als Schnittstelle zum Benutzer. Mithilfe von Java-Annotations konnte der Milton Server sehr einfach erweitert werden. Dies wird in Abbildung 24 als „WebDAV Server Custom Extension“ dargestellt.

2. Dateisystem
   Für die Verwaltung der logischen Struktur der Daten, wurde ein eigenes Dateisystem implementiert. Zu den Aufgaben dieses Dateisystems gehören Funktionen wie das Verwalten der Metainformationen (Dateiname, Erstellungsdatum, Dateigrösse, . . .) oder Generierung der Dateinamen, welche für die Speicherung der Datenblöcke bei den Cloud Anbietern verwendet werden.

3. RAID5 Komponente
   Die RAID5 Komponente ist für die Aufteilung der Dateien in Blöcke, für die Verteilung der Blöcke auf die Cloud Anbieter sowie für die redundante Speicherung zuständig. Ausserdem müssen die Datenblöcke beim Lesen einer Datei wieder in der richtigen Reihnefolge ausgelesen und zusammengefügt werden.

4. Verschlüsselungskomponente
   Die Verschlüsselungskomponente ist in die RAID5 Komponente integriert. Sie dient dazu, die einzelnen Datenblöcke zu verschlüsseln, bevor diese auf die Cloud Anbieter geschrieben werden und diese beim Lesen wieder zu entschlüsseln.

10.3. Dateisystem

Das Dateisystem ist das Bindeglied zwischen der WebDAV-Schnittstelle und der RAID5 Komponente.

10.3.1. Übersicht

Eine Übersicht der involvierten Klassen:

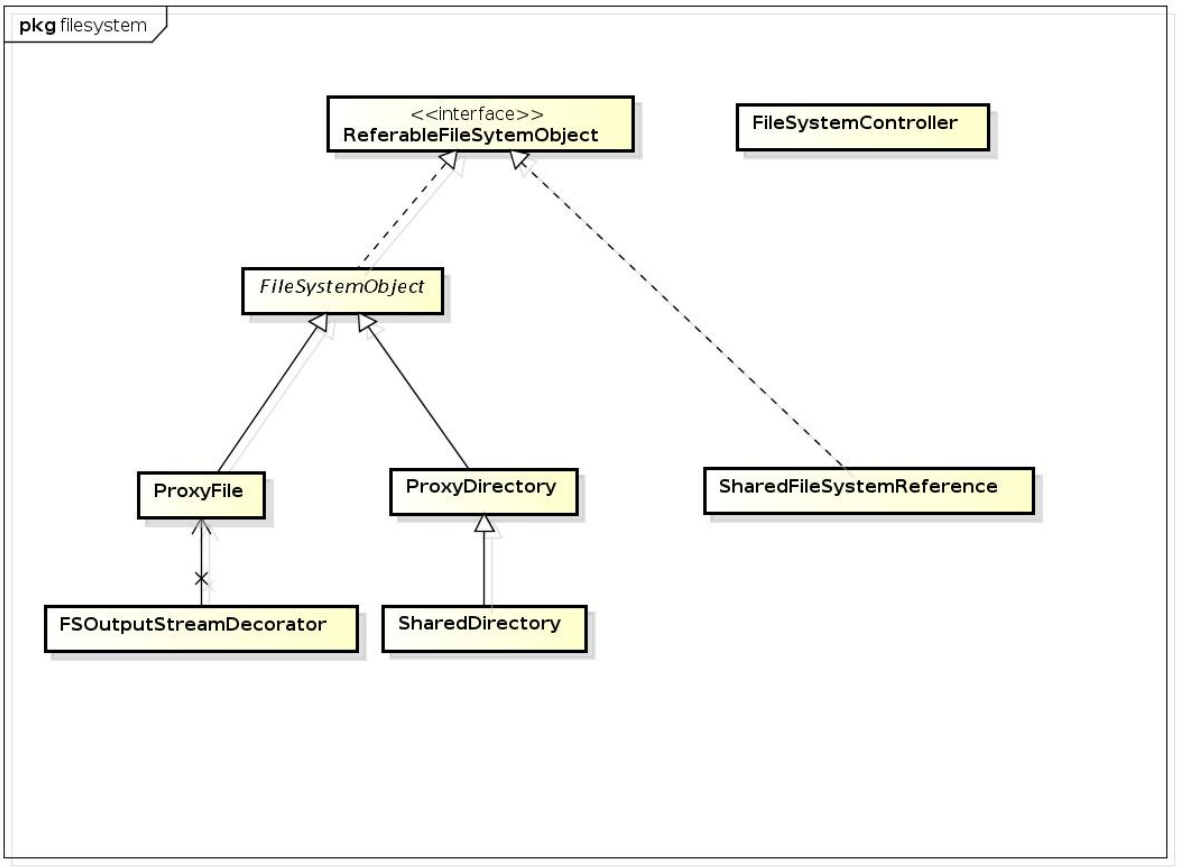

Abbildung 25: Klassendiagramm des Dateisystems

10.3.2. Klassen

Eine Beschreibung der Klassen und deren Aufgabe im Dateisystem.

ReferableFileSystemObject Ist ein Interface, dass alle Daten auf dem Dateisystem erfüllen müssen, um in einem ProxyDirectory gelistet werden zu können.

FileSystemObject Das FileSystemObject ist eine abstrakte Klasse, welche von den Klassen ProxyFile und ProxyDirectory erweitert werden. Sie enthält die meisten Methoden, welche für das Verwalten der Metainformationen nötig sind.

ProxyFile Das ProxyFile repräsentiert eine Datei, welche Daten speichern kann. Die Daten können mit getInputStream und getOutputStream gelesen beziehungsweise geschrieben werden.

FSOutputStreamDecorator Der FSOutputStreamDecorator implementiert das Decorator-Pattern und ein Callback ähnliches Verhalten. Bei der Instanzierung der Klasse durch das ProxyFile wird dem FSOutputStreamDecorator der OutputStream der RAID5 Komponente und das ProxyFile selbst als Parameter mitgegeben. Bei jedem Aufruf einer der write-Methoden des FSOutputStreamDecorator werden die geschriebenen Bytes gezählt und die Argumente dem OutputStream der RAID5 Komponente weitergereicht. Beim Aufruf der close-Methode ruft der FSOutputStreamDecorator die setContentLength-Methode des ProxyFile-Objekts auf und setzt so die Dateigrösse.

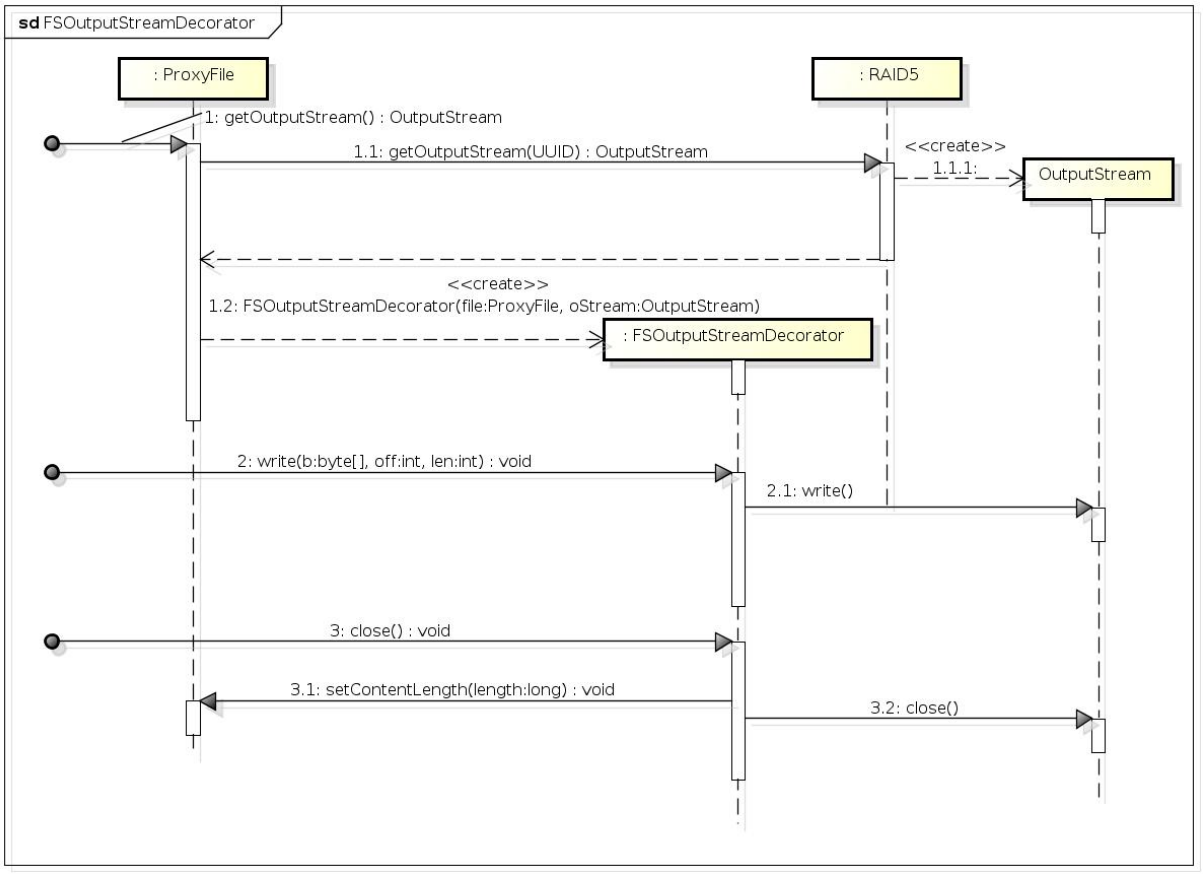

Abbildung 26: Sequenzdiagramm des FSOutputStreamDecorator

ProxyDirectory Das ProxyDirectory speichert die Einträge der ProxyFiles und der Sub-ProxyDirectories. Es bietet die Methoden getSubDirectories und getFiles an, welche eine Liste von ProxyDirectories und beziehungsweise ProxyFiles zurückgeben.

SharedDirectory Das SharedDirectory erweitert das ProxyDirectory und implementert die Methode getSharedReferneces.

SharedFileSystemReference Die SharedFileSystemReference ist ein Spezialfall. Es speichert nicht die üblichen Metainformationen, sondern eine Referenz auf ein externes FileSytemObject mit den Informationen, wie darauf zugegriffen wird. Die Informationen für den Zugriff auf ein externes File werden als JSON gespeichert, das der StreamProviderFactory mitgegeben wird. Diese erstellt dan einen StreamProvider für den Zugriff auf ein externes Dateisystem.

FileSystemController Der FileSystemController stellt allgemeine Methoden für das Dateisystem zur Verfügung wie zum Beispiel das Erstellen des Root-Directories.

10.4. RAID5 Komponente

Die RAID5 Komponente hat die folgenden Hauptaufgaben:

- Aufteilung der Daten in Blöcke:
  Die zu speichernden Dateien werden als Vorbereitung für die Verteilung in Datenblöcke unterteilt.
- Berechnung der Paritätsblöcke:
  Aus den unterteilten Datenblöcken, welche Nutzdaten enthalten, werden Paritätsblöcke für die redundatne Speicherung erstellt.
- Verteilung der Datenblöcke auf verschiedene Cloud Anbieter:
  Datenblöcke und Paritätsblöcke werden nach einem RAID5 Verfahren auf die verschiedenen Cloud Anbieter verteilt.
- Zusammensetzung der Datenblöcke zu Dateien
  Beim Lesen der Dateien werden die Datenblöcke von den Cloud Anbietern ausgelesen und in der richtigen Reihenfolge zusammengesetzt.
- Wiederherstellung von Daten
  Bei einem Ausfall eines Cloud Anbieters, müssen anhand der Paritäten die verlorenen Daten wiederhergestellt werden. Von diesem Vorgang bekommt der Benuzter jedoch nichts mit. Erst wenn mehr als ein Cloud Anbietern ausgefallen ist, können die Daten nicht mehr rekonstruiert werden.

Folgendes Klassendiagramm soll die Struktur der RAID5 Komponente verdeutlichen:

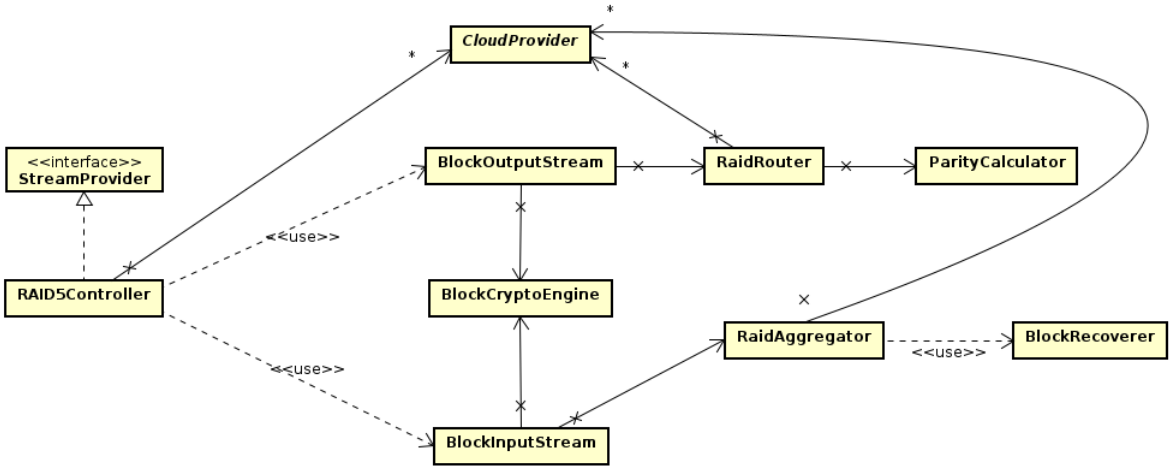

Abbildung 27: RAID5 Klassendiagramm

Als Schnittstelle der RAID5 Komponente dient die RAID5Controller-Klasse. Diese implementiert das StreamProvider Interface.

Um Dateien auf die Cloud Anbieter schreiben zu können, bietet die RAID5 Komponente einen Output-Stream an. In diesen Stream können dann Daten in Form von Bytes geschrieben werden. Der Stream kümmert sich darum, dass die Daten in kleinere Blöcke unterteilt werden. Über die RaidRouter-Hilfsklasse werden die Datenblöcke schlussendlich auf die Provider geschrieben.

Um die Dateien wieder lesen zu können, bietet die RAID5 Komponente einen Input-Stream an. Aus diesem Stream können die Daten in Form von Bytes ausgelesen werden. Über die RaidAggregater-Hilfsklasse werden die einzelnen Datenblöcke von den Providern gelesen und in der richtige Reihenfolge zusammengefügt. Die Ausfallsicherheit ist ebenfalls Bestandteil der RaidAggregator-Hilfsklasse.

10.5. Drivers - Cloud Anbieter Klassen

Für die Kommunikation mit den Cloud Anbietern wurde das Interface „CloudProvider“ definiert. Dieses Interface wird für die Implementation der Drivers für die verschiedenen Cloud Anbieter verwendet. Die Drivers sind für die Kommunikation (z. B. Daten lesen) mit bestimmten Cloud Anbieter zuständig. Im Rahmen dieser Arbeit wurde mehrere Drivers implementiert.

Zurzeit gibt es folgende Implementationen an CloudProvider-Klassen (Drivers):

- **LocalProvider**
  Die LocalProvider-Klasse stellt eine lokale CloudProvider-Implementation dar. Dieser Provider wurde zu Beginn entwickelt, um einen Cloud Anbieter zu simulieren. Die Daten werden nicht in eine Cloud, sondern in ein lokales Verzeichnis geschrieben. Die Struktur der gespeicherten Daten unterscheidet sich nicht von einem richtigen Cloud Anbieter. Ausserdem könnte z.B. ein Verzeichnis ausgewählt werden, welches über Dropbox synchronisiert wird und somit landen die Daten genauso bei einem Cloud Anbieter.

- **WebDAVProvider**
  Die WebDAVProvider-Klasse dient als Grundlage für alle Cloud Anbieter, welche nativ eine WebDAV Schnittstelle anbieten. Damit auch Dateien geteilt werden können, muss für jeden Cloud Anbieter eine separate CloudProvider-Klasse implementiert werden, welche nur Sharing Funktionalität anbietet. Das ist nötig, da der Algorithmus zum Teilen von Daten auf der Sharing Funktion der jeweiligen Cloud Anbieter aufbaut. Die Schnittstelle hierfür sieht bei jedem Anbieter anders aus. In dieser Arbeit wurde die Sharing Funktionalität für OwnCloud implementiert. Somit werden alle OwnCloud Anbieter unterstützt.

11. Resultat

Das Resultat dieser Arbeit ist nun ein erweiterbarer WebDAV Proxy, welcher die Daten auf verschiedenen Cloud Anbietern speichert.

Erstellt man auf dem WebDAV Proxy eine Datei wird diese von der RAID5 Komponente in Blöcke aufgeteilt. Diese Blöcke werden verschlüsselt bevor sie auf einem Cloud Anbieter landen.
Beim Lesen der Datei, werden die benötigten Blöcke von den Cloud Anbietern gelesen, entschlüsselt und in richtiger Reihenfolge zusammengefügt. Neben dem Schreiben und Lesen einer Datei, besteht auch die Möglichkeit Dateien zu kopieren, zu löschen und zu verschieben.

Zurzeit wurden als Cloud Anbieter das Lokale Dateisystem (ein bestimmter lokaler Ordner) und ein Cloud Anbieter mit WebDAV Unterstützung (OwnCloud) implementiert. Beide Cloud Anbieter implementieren dasselbe Interface. Dadurch besteht die Möglichkeit, weitere Cloud Anbieter zu implementieren.

Mit dem Link-Sharing ist der Benutzer in der Lage, Dateien mit anderen WebDAV Proxy Nutzer zu teilen. Für das Teilen einer Datei, kann der Benutzer diese in den shared-Ordner verschieben. Danach kann der Benutzer über die URL „[WebDAV-Host]:8080/sharing“ auf die Liste mit allen geteilten Dateien zugreifen. Dort kopiert er den Link und kann diesen einem beliebigen anderen WebDAV Proxy Nutzer senden.

Der WebDAV Proxy ist noch nicht ausgereift und stellt lediglich einen Proof of Concept dar. Für den produktiven Einsatz sind noch Optimierungen und Erweiterungen nötig.

12. Ausblick

In diesem Projekt wurden Entscheidungen getroffen, welche das Erweitern dieser Applikation ermöglichen und vereinfachen soll. Des Weiteren werden folglich auch Optimierungsvorschläge erwähnt, welche die Effizienz vom WebDAV Proxy steigern würden, jedoch im Rahmen der Bachelorarbeit nicht umgesetzt werden konnten.

12.1. Fertigstellen vom WebDAVProvider

Die WebDAVProvider-Klasse ist zur Zeit nicht in der Lage, herauszufinden ob der Cloud Anbieter online ist. Hierfür wurde noch kein Konzept entwickelt. Deshalb funktioniert die Ausfallsicherheit für den WebDAVProvider noch nicht, sondern nur für den LocalProvider. Damit die Ausfallsicherheit auch beim Ausfall eines WebDAVProviders gewährleistet ist, ist die Implementation der entsprechenden Funktion erforderlich. Ausserdem hat der WebDAVProvider noch Perfomanceprobleme. Hier ist noch nich ganz klar, wo die Ursache liegt. Möglich ist, dass die eingesetzte WebDAV-Library zu langsam arbeitet. Es könnte auch am WebDAV-Protokoll selbst liegen, dass dieses für eine Übertragung von vielen kleinen Dateien (Blöcke) ungeeignet ist.

12.2. Optimierung des Schreibprozesses

- **Verwenden von „write(byte[ ] b)“ statt „write(int b)“**
  Der Schreibprozess wird duch einen Output-Stream ermöglicht. Hierfür wurde die abstrakte standard Java Output-Stream Klasse erweitert. Da es nur nötig war, die standard write-Funktion mit nur einem Parameter (write(int b)) zu überschreiben, wurde dies in dieser Arbeit so gemacht. Die Standardimplementierung bietet jedoch mehrere Überladungen dieser write-Funktion an. Somit kann man ganze Arrays auf einmal schreiben. Wenn man diese jedoch nicht überschreibt, wird bei dessen Aufruf einfach immer wieder die Funktion mit nur einem Parameter aufgerufen. Diese schreibt jedoch nur Byte für Byte und nicht ein ganzes Byte-Array auf einmal. Um die Schreibgeschwindigeit zu erhöhen, ist es sinnvoll auch die anderen Überladungen zu überschreiben und ganze Byte-Arrays auf einmal zu schreiben anstatt jedes Byte einzeln.

- **Datei editieren ohne dessen Blöcke zu löschen**
  Bei der Editierung einer Datei oder eines Ordners, werden zuerst alle zugehörigen Blöcke gelöscht. Danach werden neue Blöcke erstellt, welche den neuen Inhalt enthalten. Der Grund für diese Implementierung ist die variierende Menge an Blöcke je nach Inhalt. Hat zum Beispiel eine Datei mit grossem Inhalt nach der Editierung nur noch einen kleinen Inhalt, werden viele Blöcke nicht mehr benötigt. Dafür müsste ausgerechnet werden wie viel und welche Blöcke gelöscht werden müssen und welche Blöcke nur editiert werden müssen. Da dies die Komplexität erhöhen würde, wurde in dieser Version darauf verzichtet. Um den Schreibprozess bei existierenden Datei zu optimieren kann man nun das Löschen aller Blöcke und das Erstellen der neuen Blöcke durch das Editieren der bestehenden Blöcke ersetzen. Die Anzahl der benötigten Blöcke wird dann anhand des neuen Inhalts berechnet und die überflüssigen Blöcke werden gelöscht (wenn der neue Inhalt kleiner ist als der alte Inhalt) bzw. wenn nötig werden zusätzliche Blöcke erstellt.

12.3. Optimierung des Leseprozess

Der Leseprozess wird durch einen Input-Stream ermöglicht. Genau gleich wie beim Output-Stream wurde auch hier die read-Funktion mit nur einem Parameter überschrieben. Somit werden die Daten Byte für Byte ausgelesen. Durch die Überschreibung der restlichen Überladungen könnte auch der Lesevorgang beschleunigt werden. Der Leseprozess hat noch ein weiteres Problem, welches für die Zukunft gelöst werden muss.

Das Auslesen der einzelnen Datenblöcke von den Cloud Anbietern läuft Parallel ab, um eine Performance Optimierung erzielen zu können. Die ausgelesenen Datenblöcke werden in blockierenden Queues für die Sortierung und korrekten Zusammensetzung zwischengespeichert. Die Weiterverarbeitung wird nicht parallel durchgeführt. Die Datenblöcke werden fortlaufend aus den Queues genommen und weiterverarbeitet. Wenn die Queues leer sind, weil das Lesen länger dauert wird gewartet. So weit so gut. In der jetzigen Implementierung könne es passieren, dass der Leseprozess schneller als der verarbeitende Prozess ist. Dies ist zwar eher unwahrscheinlich, da das Auslesen in der Regel mehr Zeit in anspruch nimmt, als die Weiterverarbeitung. Wenn der Fall troztdem eintritt, würde

dies bedeuten, dass das RAM über die Zeit immer voller wird.

Um diesem Problem entgegen zu wirken, könnte die Kapazität der Queues auf eine bestimmte Anzahl Datenblöcke begrenzt werden. Sobald die Queues voll sind, müssen die Leseprozesse blockieren und warten, bis wieder Platz frei wird. Somit kann der benötigte Speicherplatz im RAM nicht mehr unkontrolliert anwachsen.

12.4. Optimierung der Verschlüsselung

In der aktuellen Version des WebDAV Servers wird für jeden einzelnen Datenblock ein eigener Key generiert. Die Key Generierung ist jedoch relativ teuer und fällt besonders bei grossen Dateien ins Gewicht. Bei einer 300MB Datei mit einer Default Blockgrösse von 32kB müssen 9600 verschiedene Keys generiert werden. Durch die Parallelisierung kann zwar eine Performance Optimierung erzielt werden, aber der Kopiervorgang ist trotzdem noch langsam.

Um diesen Vorgang zu beschleunigen, könnte man die Keys öffter benutzen. Damit die Daten trotzdem noch sicher vor unerwünschten Zugriffen seitens des Cloud Anbiters sind, müsste jedoch dafür gesorgt werden, dass ein mehrfach verwendeter Key nur bei ein und dem Selben Cloud Anbieter zum Einsatz kommt. Ansonsten könnte ein Cloud Anbieter alle Keys der bei ihm gespeicherten Datenblöcke auslesen und mit diesen Keys versuchen die Datenblöcke zu entschlüsseln. Wenn er Glück hat, könnten somit einige Datenblöcke entschlüsselt werden.

Wenn jedoch pro Cloud Anbieter ein Pool von unterschiedlichen Keys angelegt wird, und beim Verschlüsseln der Datenblöcke zufälligerweise ein Key aus dem Pool verwendet wird, müssen nicht immer neue Keys generiert werden. Es muss allerdings darauf geachtet werden, dass ein Key nicht zu oft verwendet wird, um zu gewährleisten, dass die Wiederholung des Keys nicht die Sicherheit der Verschlüsselung schwächt.

12.5. Erweiterung mit weiteren Cloud Anbieter

Für den Benutzer ist zurzeit die Auswahl der Cloud Anbieter begrenzt. Es können nur Cloud Anbieter verwendet werden, welche WebDAV unterstützen. Das Link-Sharing ist zurzeit nur für OwnCloud implementiert. Hier wurde durch Abstraktion (Interfaces) darauf geachtet, dass in Zukunft weitere Cloud Anbieter Klassen implementiert werden können, welche eine bestimmtes Java-Klasse Namens „CloudProvider“ erweitern müssen.

12.6. Erweiterung des Link-Sharing

- **Schreiben einer geteilten Datei**
  Zurzeit kann eine geteilte Datei über den öffentlichen Link nur gelesen werden. Das Editieren ist nur dem Besitzer der Datei erlaubt. Dies könnte Erweitert werden: Man könnte die Möglichkeit bieten auch über den geteilten Link die Dateien zu editieren.

- **Teilen von Ordnern**
  Das Link-Sharing erlaubt zurzeit nur das Teilen von Dateien. Hier besteht eine Erweiterungsmöglichkeit sodass auch Ordnerstrukturen geteilt werden können.

- **Teilen von mehreren Dateien über eine einzige Freigabe**
  Aktuell enthält jede Freigabe nur die Blöcke einer einzigen geteilten Datei. Dies könnte so erweitert werden, dass der Benutzer über eine einzige Freigabe mehrere Datei (bzw. Blöcke von Dateien) teilen könnte. Dadurch kann man z.B. mehrere Dateien, welche mit einem bestimmten Benutzer geteilt werden sollen, in die Selbe Freigabe verschieben, statt für jede Datei eine neue Freigabe zu erstellen.

12.7. Erweiterung der Ausfallsicherheit

Die implementierte Ausfallsicherheit beschränkt sich im Moment auf ein Minimum. Es wird nur geprüft, ob ein Cloud Anbieter komplett ausgefallen ist und in diesem Fall werden die verlorenen Daten beim Lesen wiederhergestellt. Alle Manipulationen der Daten werden vom WebDAV Server verhindert, um nicht in einen inkonsistenten Zustand zu kommen. Die Ausfallsicherheit lässt sich in mehreren Schritten weiter ausbauen.

- **Verlust von einzelnen Datenblöcken**
  Wenn geprüft wird, ob alle Datenblöcke bei einem Cloud Anbieter vorhanden sind und nicht nur ob der Cloud Anbieter komplett ausgefallen ist, könnte man auch Dateien wiederherstellen, bei denen nur ein Teil verloren gieng.

- **Checksumme**
  Um die Korrektheit der Daten zu gewährleisten, wären auch Checksummen der Datenblöcke denkbar.

- **Wiedereintritt eines ausgefallenen Cloud Anbieters**
  Damit das Manipulieren der Daten auch während einem Ausfall eines Cloud Anbieters möglicht ist, müsste das Verhalten des WebDAV Servers definiert werden, nachdem ein ausgefallener Cloud Anbieter wieder verfügbar ist. Hierfür gibt es mehrere mögliche Ansätze.

  Zum Beispiel könnten bei einem Ausfall alle veränderten Datenblöcke, welche beim ausgefallenen Cloud Anbieter gespeichert werden sollten, in ein lokales Verzeichnis auf dem WebDAV Proxy geschrieben werden. Wenn der ausgefallene Cloud Anbieter wieder verfügbar wird, könnte ein Hintergrundprozess gestartet wreden, welcher anfängt die veralteten Datenblöcke beim Cloud Anbieter durch die neuen im lokalen Verzeichnis auszutauschen. Der Hintergrundprozess müsste dann mittels Locking den Zugriff auf die Datei verhindern, welche gerade verarbeitet wird. Wenn der Benutzer auf eine Datei zugreifen will, welche noch nicht synchronisiert wurde, könnte der Hintergrundprozess diese priorisieren und sofort synchronisieren. Für den Benutzer würde sich in diesem Fall bei einem Ausfall nichts ändern.

12.8. Erweiterung mit CalDAV und CardDAV

Im Verlauf des Projekts wurde die beiden Userstories CalDAV und CardDAV wegen der tiefen Priorität nicht weiter berücksichtigt. Es besteht jedoch die Möglichtkeit den Milton

WebDAV Server um diese zwei Funktionen zu erweitern. Hierfür ist jedoch die kostenpflichtige Milton Enterprise Edition [14] notwendig (aktuell wird nur die Milton Community Edition [14] eingesetzt).

12.9. Erweiterung mit einem Konfigurationsinterface

Zurzeit werden die Einstellungen (verwendete Cloud Anbieter, Logindaten für die einzelnen Cloud Anbieter, etc.) über die Umgebungsvariablen vorgenommen. Der WebDAV Proxy könnte um ein Konfigurationsinterface erweitert werden, mit welchem der Benutzer in der Lage ist, die Einstellungen über ein GUI zu editieren.

12.10. Erweiterung mit Historisierung

Historisierung ist eine Funktionalität, mit welcher ein Benutzer auf ältere Versionen von seinen Dateien zuzugreifen kann. Diese Funktionalität ist von den meisten Cloud Anbietern gegeben. Der WebDAV Proxy unterstützt diese Funktionalität zurzeit nicht. Bei der Architektur wurde jedoch diese Funktionalität schon berücksichtigt: In den Metainformationen ist zurzeit die Änderungsdatum eingefügt, welcher für die Historisierungsfunktion eine wichtige Grundlage ist.

12.11. Erweiterung mit Mulit-User Support

Der WebDAV Proxy kann mit einem Mulit-User Support erweitert werden. Dieser ermöglicht, dass mehrere mitdemselben WebDAV Proxy arbeiten können. Jeder Support hat auf dem WebDAV Proxy einen eigenen geschützten bereich (z.B. Ordner), auf welchen nur er Zugriff hat. Die Zugriffsrechte können mithilfe von Metainformationen von Dateien und Ordner umgesetzt werden.

12.12. Erweiterung mit Datei-Sharing

Diese Erweiterung setzt das Multi-User Support voraus. Benutzer welche denselben WebDAV Proxy verwenden kann die Funktion angeboten werden eigene Dateien untereinander zu teilen.

13. Glossar

Dateiproperty Dateiproperties sind Zusatzinformationen einer Datei. Dazu geöhren z.B. Erstellungsdatum, Änderungsdatum, Dateigrösse, ...

Datenblock Ein Datenblock (auch Block genannt) ist ein Teil einer Datei oder eines Ordners, welcher auf einem einzelnen Cloud Anbieter landet. Ein Datenblock kann Inhalte der Datei / des Ordners, Metainformationen, Padding und / oder die Parität enthalten.

Datenschutz Der Datenschutz bezieht sich nicht auf die vorhanden Daten, sondern auf deren Ursprung. Es geht im wesentlichen um das Recht, selbst zu bestimmen, wie mit den eigenen, persönlichen Daten umgegangen werden soll [3].

Datensicherheit Als Datensicherheit bezeichnet man die Eigenschaften von IT-Systemen, die die Vertraulichkeit, Verfügbarkeit und die Integrität der Informationen sicherstellen. Die Informationen werden dabei eingeschränkt (Vertraulichkeit), überprüfbar gemacht (Integrität) und mehrfach abgespeichert (Verfügbarkeit) [3].

DefaultBlockSize Über die DefaultBlockSize wird bestimmt, wie gross ein Datenblock standardmässig sein soll. Beim splitten der Daten in Blöcke wird immer versucht diese Grösse zu erreichen. Wenn zu wenig Bytes vorhanden sind, wird die Blockgrösse verkleinert und wenn nötig kommt ein Padding zum Einsatz.

Metainformation Als Metainformationen werden Daten bezeichnet, welche für das Schreiben und Lesen der Datenblöcke benötigt werden. Dazu gehören z.B. Informationen wie die Anzahl Padding Bytes im Block. Die Dateiproperties gehören auch zu den Metainformationen.

NSA-Skandal Die globale Überwachungs- und Spionageaffäre entstand aus Enthüllungen von als Top Secret gekennzeichneten Dokumenten der National Security Agency (NSA) und auf den darauf folgenden Veröffentlichungen sowie auf den internationalen Reaktionen [19].

Parität Das Resultat einer XOR-Operation über Datenblöcke mit Nutzdaten nennt man Parität. Die Parität ist im Grunde nichts anderes als ein Datenblock, welcher für die Wiederherstellung der Daten benötigt wird, nachdem ein Cloud Anbieter ausgefallen ist.

Slice In dieser Arbeit wird der Begriff Slice im Zusammenhang mit dem RAID5 System verwendet. Gemeint ist dabei eine Gruppe von Datenblöcken, welche bei den Cloud Anbietern gespeichert wir. Ein solcher Slice beinhaltet pro Cloud Anbieter einen Datenblock, wobei einer dieser Datenblöcke die Parität ist.

WebDAV Proxy Das Produkt dieser Arbeit wird als WebDAV Proxy bezeichnet.

WebDAV Server Als WebDAV Server werden alle Technologien bezeichnet, welche die WebDAV Grundfuntklionalitäten zur Verfügung stellen. Dazu gehört das Milton Anntotation Framework, SabreDAV, das Tomcat WebDAV Servlet und viele weitere.

Literatur

[1] Truth in Media. *NSA Scandal*, 06 2014. [Online; Stand 15. April 2016]; Link: http://truthinmedia.com/one-year-later-nsa-scandal-has-cost-tech-companies-billions/.

[2] Statista. *Hauptbedenken der Nutzer von Cloud-Diensten*, 04 2016. [Online; Stand 15. April 2016]; Link: http://de.statista.com/statistik/daten/studie/297127/umfrage/bedenken-bei-der-nutzung-von-cloud-diensten-in-der-schweiz/.

[3] mastertools. *Unterschied Datenschutz und Datensicherheit*, 09 2013. [Online; Stand 14. April 2016]; Link: http://www.webdav.org/neon/litmus/.

[4] Wikipedia. *WebDAV — Wikipedia, Die freie Enzyklopädie*, 04 2016. [Online; Stand 28. April 2016]; Link: https://de.wikipedia.org/w/index.php?title=WebDAV&oldid=152918343.

[5] *WebDAV Spezifikation*, 04 2010. [Online; Stand 26. Februar 2016]; Link: http://www.webdav.org/.

[6] Wikipedia. *RAID — Wikipedia, Die freie Enzyklopädie*, 04 2016. [Online; Stand 28. April 2016]; Link: https://de.wikipedia.org/w/index.php?title=RAID&oldid=154919261.

[7] Wikipedia. *Advanced Encryption Standard — Wikipedia, Die freie Enzyklopädie*, 05 2016. [Online; Stand 13. Juni 2016]; Link: https://de.wikipedia.org/w/index.php?title=Advanced_Encryption_Standard&oldid=155047763.

[8] Wikipedia. *Counter Mode — Wikipedia, Die freie Enzyklopädie*, 05 2016. [Online; Stand 14. Juni 2016]; Link: https://de.wikipedia.org/w/index.php?title=Counter_Mode&oldid=154608702.

[9] Wikipedia. *Block cipher mode of operation — Wikipedia, The Free Encyclopedia*, 2016. [Online; Stand 15. Juni 2016]; Link: https://en.wikipedia.org/w/index.php?title=Block_cipher_mode_of_operation&oldid=724670885.

[10] Elaine Barker. *Recommendation for Key Management, Part 1: General (Revision 4)*, 01 2016. [Online; Stand 13. Juni 2016]; This publication is available free of charge from: http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4.

[11] Apache. *tomcatWebDAV*, 03 2016. [Online; Stand 10. März 2016]; Link: https://tomcat.apache.org/tomcat-7.0-doc/api/org/apache/catalina/servlets/WebdavServlet.html.

[12] Mike de Boer. *jsDAV*, 01 2016. [Online; Stand 01. April 2016]; Link: https://github.com/mikedeboer/jsDAV.

[13] fruux GmbH. *sabreDAV*, 03 2016. [Online; Stand 3. März 2016]; Link: http://sabre.io/dav/.

[14] McEvoy Software Ltd. *Milton*, 03 2016. [Online; Stand 9. März 2016]; Link: http://milton.io.

[15] Python Software Foundation. *PyWebDAV*, 03 2012. [Online; Stand 01. April 2016]; Link: https://pypi.python.org/pypi/PyWebDAV/.

[16] OwnCloud. *sabreDAV and OwnCloud*, 04 2015. [Online; Stand 02. April 2016]; Link: https://owncloud.org/blog/owncloud-and-sabredav/.

[17] The Apache Software Foundation. *Apache License Version 2.0*, 01 2004. [Online; Stand 10. März 2016]; Link: https://www.apache.org/licenses/LICENSE-2.0.html.

[18] Wikipedia. *BSD-Lizenz — Wikipedia, Die freie Enzyklopädie*, 2015. [Online; Stand 3. März 2016]; Link: https://de.wikipedia.org/w/index.php?title=BSD-Lizenz&oldid=148956662.

[19] Wikipedia. *Globale Überwachungs- und Spionageaffäre — Wikipedia, Die freie Enzyklopädie*, 2016. [Online; Stand 16. Juni 2016]; Link: https://de.wikipedia.org/w/index.php?title=Globale_%C3%9Cberwachungs-_und_Spionageaff%C3%A4re&oldid=154460419.

[20] *litmus*, 12 2011. [Online; Stand 01. Juni 2016]; Link: http://www.webdav.org/neon/litmus/.

[21] *Taige Website*, 2016. Link: https://taiga.io/.

[22] Wikipedia. *Burn-Down-Chart — Wikipedia, Die freie Enzyklopädie*, 2015. [Online; Stand 15. Juni 2016]; Link: https://de.wikipedia.org/w/index.php?title=Burn-Down-Chart&oldid=146548681.

[23] Sun Microsystems/Oracle. *Java Coding Conventions: Oracle*, 1999. [Online; Stand 15. April 2016]; Link: http://www.oracle.com/technetwork/java/codeconvtoc-136057.html.

Tabellenverzeichnis

Abbildungsverzeichnis