FERDINAND PORSCHE
FERN FH

Verteilte Datenspeicherung in der Cloud als Bestandteil eines modernen Backup-Konzepts

Masterarbeit

eingereicht von: **Stefan Lindorfer, BSc**
Matrikelnummer: 51807110

im Fachhochschul-Masterstudiengang Wirtschaftsinformatik

der Ferdinand Porsche FernFH GmbH

zur Erlangung des akademischen Grades

Master of Arts in Business

Betreuung und Beurteilung: DI. Dr. Igor Miladinovic

Zweitgutachten: Thomas Krabina, MSc

Wien, Mai 2019

Ehrenwörtliche Erklärung

Ich versichere hiermit,

1. dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Inhalte, die direkt oder indirekt aus fremden Quellen entnommen sind, sind durch entsprechende Quellenangaben gekennzeichnet.

2. dass ich diese Masterarbeit bisher weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit zur Beurteilung vorgelegt oder veröffentlicht habe.

3. dass die vorliegende Fassung der Arbeit mit der eingereichten elektronischen Version in allen Teilen übereinstimmt.

Putzleinsdorf, 08.05.2019 _____

Unterschrift

Kurzzusammenfassung: Verteilte Datenspeicherung in der Cloud als Bestandteil eines modernen Backup-Konzepts

Neue Technologien erlauben es, immer mehr Geschäftsprozesse elektronisch zu unterstützen. Aus Sicht einer Backup-Strategie bedeutet das, dass aufgrund der steigenden Menge und Wichtigkeit immer mehr Daten immer sicherer und in immer kürzeren Intervallen zu möglichst niedrigen Kosten abgelegt werden müssen. Einen potentiell kostengünstigen Speicherplatz für die Ablage von Datensicherungen in beliebig kurzen Intervallen stellen Cloud Storage Services zur Verfügung. Ohne weitere Sicherheitsvorkehrungen können allerdings viele Cloud Storage Provider oder potentielle Angreifer die abgelegten Daten direkt einsehen. Es besteht also eine wesentliche Frage darin, unter welchen Voraussetzungen sich kritische Daten sicher und sinnvoll in der Cloud ablegen lassen.

Anhand einer Literaturrecherche werden dazu im Rahmen dieser Arbeit wesentliche Aspekte sowohl für den Bereich der Cloud Storage Services als auch für den Themenbereich der Datensicherungen erörtert. Daraus abgeleitet ergibt sich ein konkretes Konzept für eine verteilte Speicherung von Daten in der Cloud, auf dessen Basis ein Prototyp implementiert wird.

Der Prototyp deckt schlussendlich nicht nur die Anforderungen von im Rahmen der Arbeit konstruierten Anwendungsfällen vollständig ab, sondern er stellt auch gleichzeitig die Antwort auf die Forschungsfrage dar. Er ist in der Lage, Änderungen von Dateien in definierten Verzeichnissen zu erkennen und den Unterschied zu Vorgängerversionen zu berechnen. Die dargestellte Strategie für eine verschlüsselte und verteilte Ablage von Daten auf mehreren Cloud Storage Services stellt sicher, dass keiner der verwendeten Provider im Besitz der vollständigen Daten ist. Für einen Cloud Storage Provider oder für potentielle Angreifer ist es deshalb selbst bei veralteten und unsicheren Authentifizierungs- und Verschlüsselungsalgorithmen nahezu unmöglich, die Daten einzusehen. Durch die gewählte Form der Verteilung der Daten ist außerdem sichergestellt, dass ein Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht einschränkt.

Schlagwörter:

Cloud Storage Service, Datensicherung, verteilte Speicherung, Datenschutz, Datensicherheit, Amazon S3, Backblaze, Google Drive

Abstract: Distributed data storage in the cloud as a component of modern backup strategies

New technologies make it possible to support more and more business processes computer-aided. Through that, it becomes increasingly important to protect business data from data loss. This can be achieved by backing up the data on secure data storages in shortest possible intervals. In the future, potential cheap storage space with the opportunity to save data in short intervals, could be provided by cloud storage services. But without further security measures, many cloud storage providers or potential attackers are able to view the stored data. Therefore, the question arises which steps are required to be able to save critical data in the cloud in a secure way.

Based on literature research, the essential aspects concerning cloud storage services as well as data backups are shown. The most important facts became part of a concept for saving data in the cloud in a secure way. In a further step, a prototype was developed.

Finally, the created prototype does not only meet the requirements of some use cases constructed in the document. The prototype also represents the answer to the research question. As well as it is possible to detect changes of files in defined directories, the prototype is able to calculate differences to previous versions. The described strategy for encrypted and distributed storage of data across multiple cloud storage services has the consequence, that no provider has access to the entire data. For this reason, it is almost impossible for a cloud storage provider or a potential attacker to view the stored data, even if the authentication or encryption algorithm are outdated. The described solution of distributing data to different cloud storage services also ensures that the failure of a single provider does not reduce the availability of the stored data.

Keywords:

cloud storage service, data backup, distributed storage, data protection, data security, Amazon S3, Backblaze, Google Drive

Inhaltsverzeichnis

1. EINLEITUNG - 1 -

1.1 Problemstellung - 1 -
1.2 Zielsetzung - 2 -
1.3 Aufbau der Arbeit - 3 -

2. GRUNDLAGEN CLOUD COMPUTING - 4 -

2.1 Entstehung und Definition von Cloud Computing - 4 -
2.2 Bereitstellungsmodelle und Serviceebenen der Cloud - 7 -
2.3 Cloud Storage Services - 10 -
2.3.1 Grundlagen - 10 -
2.3.2 Nutzen und Risiken im Vergleich zu On-Premises Lösungen - 14 -
2.3.3 Cloud Storage Security - 16 -
2.3.4 Welcher Anbieter eignet sich zur sicheren Ablage von schützenswerten Daten in der Cloud? - 18 -
2.3.5 Bestehende Services zur verteilten Speicherung in der Cloud - 21 -

3. GRUNDLAGEN DATENSICHERUNG - 24 -

3.1 Business Continuity Management und Disaster Recovery - 24 -
3.2 Hochverfügbarkeit versus Disaster Recovery - 27 -
3.3 Sicherungsarten - 29 -
3.4 Sicherungshäufigkeit und Anzahl der Sicherungskopien - 31 -
3.5 Sicherungsmedien - 32 -
3.6 Sicherungsstrategien und Kosten - 39 -
3.7 Anbieter von Sicherungssoftware - 42 -

4. KONZEPT UND IMPLEMENTIERUNG - 44 -

4.1 Beschreibung und Zielsetzung - 44 -
4.1.1 Darstellung des gewünschten Funktionsumfangs - 44 -
4.1.2 Auswahl der zu nutzenden Plattform - 46 -
4.2 Technische Konzeptionierung der einzelnen Teilschritte - 48 -
4.2.1 Änderung von Dateien erkennen - 48 -
4.2.2 Berechnung der Unterschiede zwischen zwei Dateien - 49 -
4.2.3 Komprimierung - 51 -
4.2.4 Dateipfad und Dateiname verbergen - 52 -
4.2.5 Verschlüsselung - 53 -
4.2.6 Aufteilung einer Datei in mehrere Teildateien - 54 -
4.2.7 Kommunikation mit den Cloud Speicherdiensten - 54 -
4.3 Struktur der Implementierung - 59 -

5. VALIDIERUNG DER FUNKTIONEN DES PROTOTYPS - 64 -

5.1 Unternehmenstyp 1 - 65 -
5.2 Unternehmenstyp 2 - 68 -

6. DISKUSSION DER ERGEBNISSE - 72 -

6.1 Notwendigkeit von Datensicherungen - 72 -
6.2 Zeitgemäße Sicherungsmedien - 73 -
6.3 Sichere Ablage von Daten in der Cloud - 74 -
6.4 Begrenzte Ressource „Internet“ - 76 -
6.5 Bewertung des Prototyps - 77 -

7. FAZIT UND AUSBLICK - 79 -

LITERATURVERZEICHNIS - 81 -

ABBILDUNGSVERZEICHNIS - 89 -

TABELLENVERZEICHNIS - 91 -

ABKÜRZUNGSVERZEICHNIS - 92 -

ANHANG A: DATEI „README.TXT“ - 93 -

ANHANG B: SOURCECODE „PROTOTYP.SH“ - 96 -

ANHANG C: SOURCECODE „AMAZON_S3_GLACIER.SH“ - 106 -

ANHANG D: SOURCECODE „BACKBLAZE.SH“ - 109 -

ANHANG E: SOURCECODE „GOOGLE_DRIVE.SH“ - 112 -

1. Einleitung

1.1 Problemstellung

Für viele Unternehmen kann der Verlust von Geschäftsdaten katastrophale Folgen haben. Mögliche Konsequenzen sind Reputationsschäden, regulatorische Strafen, Verlust von Wettbewerbsvorteilen oder negative Auswirkungen auf den Kundenservice. Je nach Ausmaß des Datenverlusts kann auch die Existenz eines Unternehmens bedroht sein. Entsprechend wichtig ist daher eine adäquate Sicherungsstrategie für alle relevanten Daten eines Unternehmens.

Neue Technologien erlauben es, immer mehr Geschäftsprozesse elektronisch zu unterstützen. Aus Sicht einer Backup-Strategie bedeutet das, dass aufgrund der steigenden Menge und Wichtigkeit immer mehr Daten immer sicherer und in immer kürzeren Intervallen zu möglichst niedrigen Kosten abgelegt werden müssen.

Neue Technologien liefern möglicherweise eine Antwort darauf, wie das auch in Zukunft gelingen kann. Ein zentraler Begriff an dieser Stelle ist der des „Cloud Computing“. Neben Privatpersonen profitieren immer mehr Unternehmen von der Hinzuziehung virtueller Ressourcen aus der Cloud. Eine weitreichende Skalierbarkeit sowie eine nutzungsabhängige Verrechnung sind wesentliche Vorteile dieser modernen Technologien [Re18, S. 4]. Cloud Speicherdienste wie Google Drive [Go19a], Dropbox [Dr18], Microsoft OneDrive [Mi19a] oder Amazon S3 [Am19a] stellen einen potentiell kostengünstigen Speicherplatz für die Ablage von Datensicherungen zur Verfügung. Dieser kann permanent, und somit in beliebig kurzen Intervallen, über das Internet beschrieben werden. In vielen Berichten taucht aber immer wieder auf, dass nur sichere und vertrauensvolle Cloud Storage Provider für die Ablage von unternehmenskritischen Daten verwendet werden sollen. Doch welcher Cloud Storage Provider ist sicher und vertrauensvoll? Es besteht also eine offene Frage darin, unter welchen Voraussetzungen sich Datensicherungen sicher und sinnvoll in der Cloud ablegen lassen. Zu beachten ist dabei auch die zur Verfügung stehende Internet-Bandbreite.

1.2 Zielsetzung

Das Ziel der vorliegenden Arbeit ist es, einen Prototyp für die Sicherung von Daten in der Cloud zu entwickeln. Die Sicherung sollte ein adäquates Niveau sowohl im Hinblick auf den Datenschutz als auch im Hinblick auf die Datensicherheit aufweisen, sowie eine hohe Aktualität der Daten sicherstellen.

Daten sind dann geschützt abgelegt, wenn keine Zerstörung, Manipulation oder unzulässige Weitergabe der Daten möglich ist. Im Falle der Nutzung von Cloud Services wird die Verantwortung ein Stück weit an den Cloud Storage Provider abgegeben. Es besteht die Möglichkeit, dass der Anbieter selbst oder ein erfolgreicher Angreifer Daten löscht, manipuliert oder entwendet. Auch eine Verschlüsselung der Daten bringt keine hundertprozentige Sicherheit, dass die Daten nie eingesehen werden. Speziell bei längerer Aufbewahrung der Daten kann sich ein unzureichendes Schutzniveau dadurch ergeben, dass Schwachstellen an verwendeten Verschlüsselungsalgorithmen entdeckt und neue Algorithmen entwickelt werden. Ein ausreichendes Datenschutzniveau ist deshalb erst dann erreicht, wenn ein einzelner Service Provider nicht im Besitz der vollständigen Daten ist.

Ausreichende Datensicherheit ist dann gewährleistet, wenn der Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht einschränkt. Die Aktualität einer Datensicherung ist dann in einem ausreichenden Maß gegeben, wenn die Daten in der Sicherung so aktuell sind, dass ein Datenverlust nach einer Rücksicherung keine nennenswerten negativen wirtschaftlichen Folgen nach sich zieht. Dies kann bedeuten, dass ausgewählte Daten in einem Intervall von wenigen Minuten gesichert werden müssen. Es gilt dabei, die vorhandene Bandbreite so effizient wie möglich zu nutzen. Die Berechnung einer Dateiänderung im Vergleich zu einer Ursprungsdatei sollte im Rahmen der vorliegenden Arbeit ebenfalls untersucht und in Form eines Prototyps implementiert werden.

Die Forschungsfrage für die geplante Arbeit lautet wie folgt:

„Welche Funktionen muss eine Anwendung beinhalten, um in der Lage zu sein, Änderungen von Dateien in definierten Verzeichnissen zu erkennen, den Unterschied zu einer Ursprungsversion

zu berechnen und diesen in verschlüsselter Form so auf Cloudspeichern abzulegen, dass kein Provider im Vollbesitz der Daten ist und der Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht einschränkt?“

1.3 Aufbau der Arbeit

Der erste Abschnitt der Arbeit behandelt auf Basis einer Literaturrecherche die Grundlagen des Cloud Computing. Darauf aufbauend werden die Cloud Storage Services als eine spezielle Form des Cloud Computing beschrieben. Auf den möglichen Nutzen, die Risiken sowie die Sicherheit wird in diesem Abschnitt näher eingegangen.

Im Anschluss daran werden die Grundlagen der Datensicherung aufgegriffen. Eine Darstellung der wichtigsten Sicherungsarten und -medien sind ebenfalls Bestandteil der Literaturrecherche dieses Kapitels. Weiters wird das Thema Datensicherung von Hochverfügbarkeit abgegrenzt und die Notwendigkeit von Datensicherungen erörtert.

Auf Basis der gewonnenen Erkenntnisse wird im anschließenden Abschnitt ein konkretes Konzept für einen Prototyp entworfen. Eine Beschreibung der Umsetzung in Form einer Software ist ebenso Inhalt dieses Abschnitts.

Die Validierung der Funktionen des Prototyps, die Beantwortung der Forschungsfrage sowie ein Ausblick auf mögliche zukünftige Entwicklungen bilden den Abschluss der Arbeit.

2. Grundlagen Cloud Computing

2.1 Entstehung und Definition von Cloud Computing

Das Cloud Computing ist die fünfte Generation des Computing. Am Beginn der Entwicklung stand das Mainframe Computing. Es handelte sich dabei um zentral bereitgestellte Großrechner, welche in etwa ab 1950 in der Lage waren, sogenannte Batch-Aufträge zu verarbeiten. Die Batchprogramme wurde in Form von Lochkarten zur Verfügung gestellt. Später gab es auch die Möglichkeit, Befehle über Terminal-Anbindungen zu übermitteln. [PP12, S. 182]

Aus dem Mainframe Computing entwickelte sich das Personal Computing. Es wurden im Vergleich zum Mainframe Computing wesentlich kleinere und günstigere Systeme eingesetzt. Die 1980er waren die Blütezeit der sogenannten Personal Computer (PC). Sie wurden in dieser Zeit sukzessive in den Büroalltag integriert. Das Personal Computing konzentriert sich darauf, den einzelnen Benutzer anzusprechen, um mit ihm in einen Dialog zu treten. Mittels eines PCs war es möglich, verschiedene Tasks dezentral abzuarbeiten. [PP12, S. 184]

Die Vernetzung von zwei oder mehreren Personal Computing Systemen war Gegenstand des Network Computing. Die durch das Network Computing entstandenen Netzwerke lassen sich unterschiedlich kategorisieren. Einer der Ansätze referenziert die geografische Distanz welche überbrückt werden soll. So handelt es sich beispielsweise bei Local Area Networks (LAN) um Netzwerke innerhalb einzelner Gebäude. Wide Area Networks (WAN) bezeichnen Netzwerke, welche größere Distanzen überbrücken. Das World Wide Web (WWW) ist das größte öffentliche WAN. Ein anderer Ansatz unterscheidet die Netzwerke hinsichtlich ihrem Design. Neben Peer-to-Peer Netzwerken entstanden in der Zeit um 1990 auch die für das geschäftliche Umfeld bedeutsame Client-Server-Architektur. Die Clients sind dabei häufig kostengünstige Geräte, welche nach Bedarf Dienste von Servern anfordern. Ein Server arbeitet üblicherweise für mehrere Clients gleichzeitig und ermöglicht es für die Benutzerin und Benutzer, komplexe Aufgaben abzuarbeiten. [PP12, S. 185]

Um das Jahr 2000 entwickelte sich aufbauend auf dem Network Computing das Grid Computing. Beim Grid Computing stellen viele lose über Netzwerke miteinander gekoppelten Rechner ihre ungenutzten Ressourcen zur Verfügung. Durch diese Art von Kopplung wird ein virtueller Hochleistungsrechner erzeugt, der von Anwendungen mit hohem Ressourcenbedarf genutzt werden kann. Die einzelnen Knoten können dabei sehr heterogen sein. [PP12, S. 189] Heute wird Grid-Computing beispielsweise noch in der Pharmaforschung oder in den Wirtschaftswissenschaften zur Lösung von komplexen Problemen eingesetzt.

Aus dem Grid Computing entwickelte sich in etwa ab 2010 das Cloud Computing. Cloud Computing beinhaltet Technologien, welche es erlauben, IT-Ressourcen dynamisch zur Verfügung zu stellen und ihre Nutzung nach flexiblen Bezahlmodellen abzurechnen. Anstelle IT-Ressourcen wie Server oder Anwendungen in unternehmenseigenen Rechenzentren zu betreiben, können diese bedarfsorientiert und flexibel in Form eines dienstleistungsbasierten Geschäftsmodells über das Internet bezogen werden [Sp18]. Während es beim Grid Computing um die Nutzung von verteilten Ressourcen geht, gibt es beim Cloud Computing eine zentrale Steuerung. Im Fall des Cloud Computing gibt es genau einen Anbieter. Die angebotenen Ressourcen können von mehreren Nutzerinnen und Nutzern unabhängig voneinander in Anspruch genommen werden.

Die Schritte der Evolution des Computing sind in Abbildung 1 grafisch dargestellt.

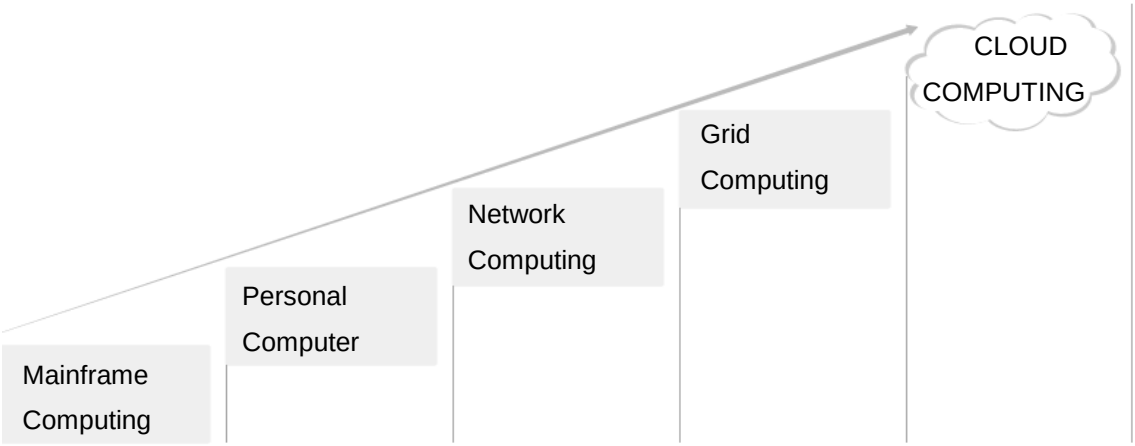

Abbildung 1: Evolution des Computing (Quelle: in Anlehnung an [PP12, S. 183])

Eine allgemeingültige Definition für den Begriff *Cloud Computing* konnte sich bis dato nicht durchsetzen. Viele der verwendeten Definitionen sind sich aber sehr ähnlich. Eine Definition, die häufig herangezogen wird, ist die der US-amerikanischen Standardisierungsstelle NIST (National Institute of Standards and Technology) [Na19], welche auch von der ENISA (European Network and Information Security Agency) [Eu19] genutzt wird. Die Definition lautet wie folgt [MG11, S. 2]:

"Cloud Computing ist ein Modell, das es erlaubt, bei Bedarf jederzeit und überall bequem über ein Netz auf einen geteilten Pool von konfigurierbaren Rechnerressourcen (z.B. Netze, Server, Speichersysteme, Anwendungen und Dienste) zuzugreifen, die schnell und mit minimalem Managementaufwand oder geringer Service Provider-Interaktion zur Verfügung gestellt werden können."

Die folgenden fünf Eigenschaften charakterisieren gemäß der NIST-Definition einen Cloud Service [MG11, S. 2]:

- **On-demand Self Service:** Die Provisionierung der Ressourcen (Rechenleistung, Storage, …) läuft automatisch und ohne Interaktion mit dem Service Provider ab.
- **Broad Network Access:** Die Services sind mit Standard-Mechanismen über das Netz verfügbar und nicht an einen bestimmten Client gebunden.
- **Resource Pooling:** Die Ressourcen des Anbieters liegen in einem Pool vor, aus dem sich viele Anwender bedienen können (Multi-Tenant Modell). Dabei wissen die Anwender nicht, wo sich die Ressourcen befinden. Mittlerweile können die Nutzerinnen und Nutzer bei ausgewählten Services auch den Speicherort vertraglich einschränken.
- **Rapid Elasticity:** Die Services können schnell und elastisch zur Verfügung gestellt werden, in manchen Fällen auch automatisch. Aus Anwendersicht scheinen die Ressourcen daher unendlich zu sein.
- **Measured Services:** Die Ressourcennutzung kann gemessen, überwacht und dokumentiert werden. Darauf aufbauend kann beispielsweise die Abrechnung oder auch eine automatische Skalierung erfolgen.

2.2 Bereitstellungsmodelle und Serviceebenen der Cloud

Im Zusammenhang mit der Cloud wird zwischen vier Bereitstellungsmodellen unterschieden [Eg15, S. 33]:

- In einer **Private Cloud** wird die Cloud-Infrastruktur nur für eine einzige Institution betrieben. Sie kann von der Institution selbst oder einem Dritten organisiert und geführt werden. Die Infrastruktur kann in einem Rechenzentrum der eigenen Institution oder in Räumlichkeiten einer fremden Institution stehen.
- Von einer **Public Cloud** wird dann gesprochen, wenn die Services von der Allgemeinheit oder einer großen Gruppe genutzt werden können. Bereitgestellt werden die Services von einem einzigen Anbieter.
- In einer **Community Cloud** wird die Infrastruktur von mehreren Institutionen geteilt, die ähnliche Interessen haben. Eine solche Cloud kann von einer dieser Institutionen oder von einem Dritten betrieben werden.
- Werden mehrere Cloud Infrastrukturen, die für sich selbst eigenständig sind, über standardisierte Schnittstellen miteinander verbunden und gemeinsam genutzt, wird dies als **Hybrid Cloud** bezeichnet.

In Abbildung 2 sind die beschriebenen Bereitstellungsmodelle schematisch dargestellt.

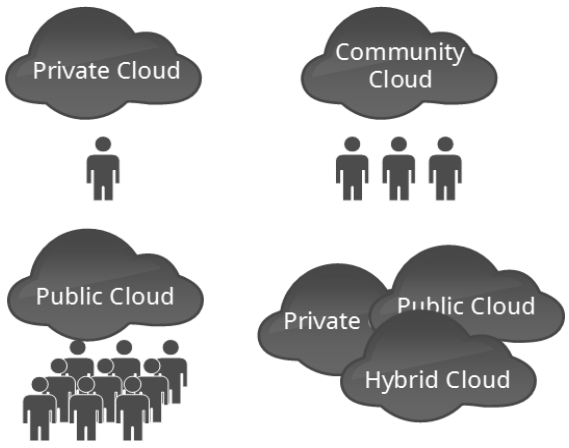

Abbildung 2: Übersicht Cloud-Deployment-Varianten (Quelle: entnommen aus [Me16])

Darüber hinaus lassen sich die Serviceebenen, auf denen Cloud-Dienstleistungen für die Anwender („Cloud Consumer“) zur Verfügung gestellt werden, durch ein weitgehend akzeptiertes 3-Ebenenmodell beschreiben:

1. **Infrastructure as a Service (IaaS):** Auf der Infrastrukturebene werden durch die Anbieter grundlegende Dienste wie Rechenleistung (Prozessorleistung), Datenspeicher (Storage) oder zum Teil auch Kommunikationsverbindungen bereitgestellt. Der Benutzer baut auf den bezogenen Services seine eigene Betriebssystemlandschaft auf (virtuelle Server), ohne selbst Arbeitsspeicher, Datenspeicher oder Prozessoren zu besitzen. Der Vorteil im Vergleich zu einem dezidierten eigenen Rechenzentrum liegt in der Skalierbarkeit. Die Recheninstanzen können je nach Anforderungen erweitert oder reduziert werden. Zusätzlich genießt man auch für kleinste Systemlandschaften die Vorteile moderner Rechenzentren. Dies können weitreichende Hardwareredundanzen oder Zutritts-, Leckage- sowie Brandmanagementsysteme sein. [MPR15, S. 10]

   Die Benutzer haben vollen Zugriff auf die virtuellen Objekte und können selbst Anwendungen installieren. Im Gegenzug dazu müssen sie die Server aber auch selbst administrieren und die nötigen Sicherheitspatches für die von ihnen eingesetzte Software einspielen. Der Cloud Provider haftet dementsprechend für keine Schäden, die auf die Fehlkonfiguration durch den Kunden zurückzuführen sind. [Be13, S. 29]

   Bekannte Beispiele für IaaS-Services sind Amazon mit EC2 (Rechenleistung) und S3 (Speicher), Rackspace, Eucalyptus oder auch GoGrid [MPR15, S. 10].
2. **Platform as a Service (PaaS):** Platform as a Service beschreibt zum einen die Bereitstellung von Softwareentwicklungsplattformen (Programming Environment) und zum anderen die Möglichkeit, entwickelte Software auf den Rechnern des Providers auszuführen (Execution Environment). PaaS richtet sich demzufolge an selbständige Softwareentwickler, Softwareentwicklungsunternehmen oder IT-Abteilungen von sonstigen Unternehmen. [MPR15, S. 10]

Der Vorteil der zentralen Bereitstellung in der Cloud liegt darin, dass der Aufwand für die Erstellung einer Softwareentwicklungsumgebung für die Entwickler wegfällt. Außerdem müssen sie sich nicht mehr um die bisher notwendige Serveradministration kümmern, sondern können sich vollständig auf ihre Programmier- und Softwareentwicklungsaufgaben konzentrieren. Im Gegensatz zu IaaS können die Nutzer die zugrundeliegenden Server nicht administrieren. Diese Aufgabe wird vom Plattformanbieter übernommen. [Be13, S. 30]

Beispiele für bekannte PaaS-Services sind Azure von Microsoft, App Engine von Google, force.com von salesforce.com, BusinessByDesign Studio von SAP oder auch Beanstalk von Amazon [MPR15, S. 11].

3. **Software as a Service (SaaS):** Im Fall von Software as a Service wird Anwendungssoftware von einem Provider online bereitgestellt. Die Abrechnung erfolgt üblicherweise abhängig von der tatsächlichen Nutzung. Dieses Modell stellt demnach eine Form der Softwaremiete dar. Softwareanwendungen werden entweder gar nicht mehr lokal installiert und nur noch über den Webbrowser genutzt oder nur in Teilen lokal installiert, während die Hauptbestandteile über das Netz bezogen werden. [Be13, S. 30] Der Anwender erwirbt kein individuelles Nutzungsrecht an der Anwendung. Er bezieht die Ressourcen vielmehr parallel zu vielen anderen Nutzern aus einem Multi-Mandanten-System („multi-tenancy“), wobei alle Anwender auf die gleiche Version einer Software zugreifen. Diese unterliegt der Verantwortung des Cloud Service Providers. Modifikationen am Kern der Applikation im Sinne einer individuellen Anpassung sind für die Anwenderin oder den Anwender nicht möglich. Beispielhafte SaaS-Produkte sind das CRM-System von salesforce.com, das ERP-System Business ByDesign aus dem Hause SAP, die Kommunikationslösung WebEx von Cisco, die Office-Lösung Office 365 von Microsoft oder Apps for Business von Google. Auch Cloud Storage Services wie Dropbox, Google Drive oder Microsoft OneDrive fallen in diese Kategorie. [MPR15, S. 12]

Die Serviceebenen der Cloud sind in Abbildung 3 überblicksartig dargestellt.

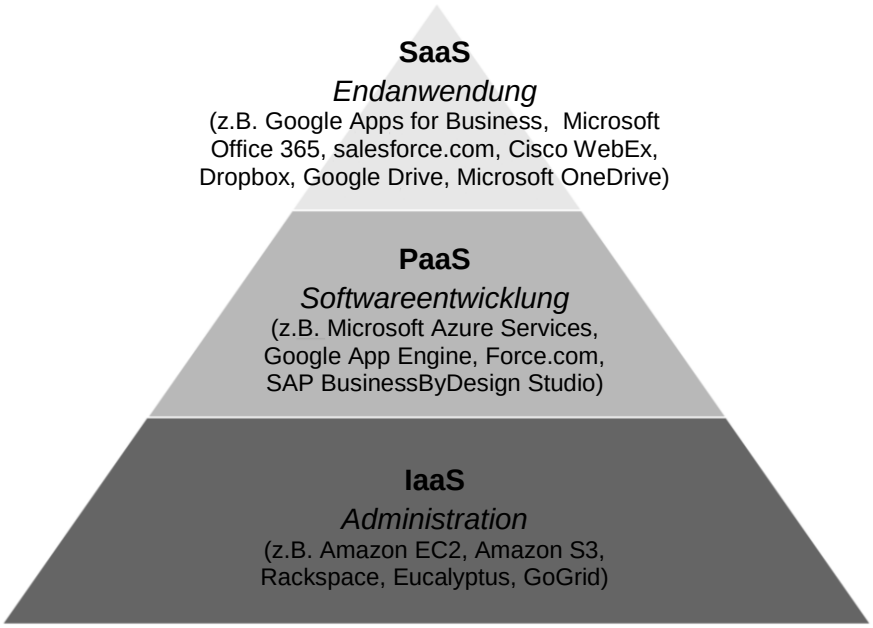

Abbildung 3: Serviceebenen der Cloud (Quelle: angelehnt an [MPR15, S. 9])

2.3 Cloud Storage Services

2.3.1 Grundlagen

Cloud Storage Services stellen ein spezielle, mittlerweile sehr bedeutsame Form des Cloud Computing dar. Es werden dabei virtuelle Festplatte zur Verfügung gestellt, auf die die Nutzerinnen und Nutzer jederzeit über das Internet zugreifen können. Somit lassen sich Daten zentral speichern und an unterschiedlichen Orten aufrufen. Auch möglich ist die Synchronisierung des Cloud Speichers mit lokalen Ordnern eines Desktop-PCs. Werden auf dem PC neue Daten hinzugefügt, so werden diese direkt auf die Online-Festplatte übertragen. Je nach Anbieter werden zusätzlich zum Cloud Speicher auch damit verwandte Funktionen, wie beispielsweise ein Weboffice, angeboten. Es lassen sich damit Textdokumente, Tabellen oder Präsentationen direkt im Webbrowser erstellen und auf dem Cloud Speicher ablegen.

Hinsichtlich der Einordnung in die Serviceebenen der Cloud (siehe Kapitel 2.2) gilt Folgendes. Angebote, bei denen ausschließlich Speicherplatz zur Verfügung gestellt wird, um ihn als Teil einer IT-Infrastruktur zu nutzen, sind der Serviceebene IaaS zuzuordnen. Für den Up- und Download von Daten stehen dabei APIs (Application

Programming Interfaces) zur Verfügung. Ein Zugriff per Webbrowser existiert meist nur zur Verwaltung oder auch zum Einsehen von Statistiken. Alle Angebote mit höherem Funktionsumfang sind der Serviceebene SaaS zuzuordnen.

Cloud Storage Services kommen in Form von folgenden Bereitstellungsmodellen vor:

- Private Cloud Storage
- Public Cloud Storage
- Hybrid Cloud Storage

Die in Kapitel 2.2 getätigten Definitionen gelten in vollem Umfang. Ein Public Cloud Storage stellt die flexibelste Form der Cloud Storage Services dar. Der Betrieb der Systemlandschaft erfolgt aus Sicht der Nutzerinnen und Nutzer durch einen externen Partner. Die Daten sind über das öffentliche Internet erreichbar. Für die Nutzerinnen und Nutzer entstehen keinerlei Investitionskosten für Hardware. Die Abrechnung der in Anspruch genommenen Services erfolgt nach tatsächlicher Nutzung. Bei einem Private Cloud Storage handelt es sich um einen Cloud Storage Service, der exklusiv für eine auftraggebende Institution betrieben wird. Datenschutzkritische Geschäftsprozesse können die Notwendigkeit eines Private Cloud Storage nach sich ziehen. Beim Hybrid Cloud Storage handelt es sich um eine Mischform aus Private Cloud Storage und Public Cloud Storage. Üblicherweise werden die Geschäftsprozesse dabei in datenschutzkritische und -unkritische Prozesse unterteilt und der jeweils passenden Cloud-Struktur zugeordnet.

Die Funktionen der einzelnen Cloud Storage Services unterscheiden sich je nach Anbieter und gewähltem Nutzungspaket. Folgende Funktionen sind mit Cloud Storage Services prinzipiell möglich [Tr19c]:

Speichern von Dateien

- *Mobile App:* Das Angebot einiger Cloud Storage Provider umfasst Apps für Smartphones und Tablets. Damit ist es möglich, Daten auf dem Online-Speicher einzusehen oder neue Daten zu ergänzen.

- *PC-Client:* Viele Cloud Storage Provider bieten einen eigenen Client für die Betriebssysteme Windows, MacOS und in einigen Fällen auch für Linux an. Damit lassen sich lokale Ordner direkt mit der virtuellen Festplatte in der Cloud synchronisieren.
- *Web App:* Sehr häufig kann der Zugang zu den Daten auch über den Webbrowser erfolgen. Es muss somit weder eine App noch ein Client zwingend installiert.
- *Application Programming Interface (API):* Während bei IaaS-Angeboten der Zugriff auf die Daten praktisch immer über Programmierschnittstellen erfolgt, handelt es sich bei SaaS-Angeboten um gern gesehene Zusatzfunktionalitäten.

Betrachten von Dateien

- *Bildergalerie:* Mit dieser Funktion lassen sich Fotos direkt im Webbrowser in Form einer Vorschau oder einer Diashow betrachten.
- *PDF/Textdatei-Viewer:* Ist dieses Feature im Cloud Storage Service integriert, so können Text- oder PDF-Dateien direkt im Webbrowser oder innerhalb einer App betrachtet werden.
- *Videostreaming:* Wurden Videos in einem Cloud Speicher abgelegt der Streaming unterstützt, so können die Videos direkt im Webbrowser betrachtet werden. Ein vorheriges Herunterladen ist nicht notwendig.

Teilen von Dateien

- *Teilen per Link:* Dritten wird der Download einer bestimmten Datei per Link ermöglicht. Jede Person, die über diesen Link verfügt, kann die Datei herunterladen. Auch können ganze Ordner direkt per Link freigegeben werden.
- *Passwortschutz:* Bei manchen Anbietern lassen sich die Freigabelinks zusätzlich mit einem Passwort koppeln. Herunterladen können demnach nur jene Personen, die über den Link und das Passwort verfügen.
- *Beschränkung der Gültigkeit:* Zusätzlich kann häufig auch die Gültigkeit der Links beschränkt werden. Der Download ist dann nur bis zu einem gewissen Datum möglich.

Kollaborative Funktionen

- *Gemeinsames Bearbeiten von Dokumenten:* Hat der Cloud Speicher kollaborative Funktionen im Repertoire, so können mehrere Kolleginnen und Kollegen in Echtzeit an einem Dokument arbeiten.
- *Live-Kommentare:* Zum gegenseitigen Austausch unter Mitarbeiterinnen und Mitarbeitern dient die Live-Kommentarfunktion. Wichtige Anmerkungen können damit direkt im jeweiligen Dokument an der passenden Stelle hinterlegt werden.
- *Dateiversionierung:* Änderungen können im Nachhinein überprüft oder bei Bedarf rückgängig gemacht werden.

Cloud Speicher eignen sich sowohl für die Verwendung durch Privatpersonen als auch durch Unternehmen. Die Schwerpunkte sind dabei verschieden [Tr19c]:

- **Privatpersonen:** Für Privatpersonen steht meist der Speicherplatz im Vordergrund, der dazu genutzt wird, private Dateien wie Bilder oder Videos mit der Familie oder Freunden zu teilen. Ein großer Vorteil ist die weltweite Erreichbarkeit des Speichers. Während Reisen erstellte Fotos können direkt abgelegt werden, um sie später vom PC aus zu weiterzuverarbeiten.
- **Unternehmen:** Unternehmen nutzen einerseits IaaS-Angebote als Teil ihrer IT-Infrastruktur. Andererseits helfen Cloud Speicher den Unternehmen dann weiter, wenn Sie über kollaborative Funktionen verfügen und so das Arbeiten an gemeinsamen Projekten vereinfachen. Am besten eignen sich dafür Produkte, die auch Weboffice-Funktionen beinhalten und eine Dateiversionierung unterstützen.
- **Beide Gruppen:** Die Festplatte in der Cloud lässt sich grundsätzlich auch als Speicherplatz für Backups nutzen. Für den Fall, dass lokale Speichermedien wie Festplatten oder USB-Geräte kaputt gehen, stehen die Daten so möglicherweise noch zur Verfügung. Diese Möglichkeit eignet sich für Privatpersonen und Unternehmen gleichermaßen. Ob das gebotene Ausmaß an Datensicherheit und Datenschutz ausreicht, ist für jeden Einzelfall zu überprüfen.

2.3.2 Nutzen und Risiken im Vergleich zu On-Premises Lösungen

On-Premises beschreibt den Betrieb von IT-Infrastruktur in eigenen Räumlichkeiten. Ein Beispiel dafür sind herkömmliche Speichersysteme, die von der Nutzerin oder dem Nutzer beschafft und innerhalb der eigenen Rechenräume betrieben werden.

Der resultierende Nutzen und die Vorteile, welche sich durch die Verwendung von Cloud Storage Services im Vergleich zu herkömmlichen On-Premises-Lösungen ergeben, lassen sich durch folgende Punkten beschreiben:

- **Skalierbarkeit:** Ein Vorteil von Cloud Services ist die hervorragende Skalierbarkeit. Steigen die Anforderungen, so kann auch die Leistung dazu passend verändert werden. Wird mehr Speicherplatz benötigt, so kann dieser gegen angepasste laufende Kosten einfach durch den Provider zur Verfügung gestellt werden.
- **Gesteigerte organisatorische Flexibilität:** Die Anwenderinnen und Anwender erhöhen durch die Verwendung von Cloud Storage Services ihre Flexibilität. Auch bei schwer kalkulierbarem oder schwankendem Nutzungsverhalten lassen sich die bereitgestellten Ressourcen ohne viel Vorlaufzeit automatisiert an den Ressourcenbedarf anpassen. Es stehen aus Sicht der Anwenderinnen und Anwender nie überflüssige Ressourcen bereit. Sicherheits-, Auslastungs-, Know-how- und Technologierisiken werden an den Dienstleister ausgelagert.
- **Kosteneffizienz:** Aus fixen Investitionskosten und -risiken werden variable Kosten, wobei die Kosten mit den genutzten Ressourcen aufgrund einer verbrauchsabhängigen Verrechnung übereinstimmen. Investitionen in Überkapazitäten um Lastspitzen abdecken zu können müssen nicht mehr getätigt werden. Das nicht gebundene Kapital kann anderswo investiert werden. Ist die benötigte Leistung bekannt, so lassen sich auch die Kosten sehr gut vorausbestimmen. Unerwartete Kosten, wie beispielsweise für den Austausch von Festplatten in On-Premises-Speichersystemen, können ausgeschlossen werden. Um die Instandhaltung der Hardware kümmert sich ausschließlich der Anbieter.

- **Keine Pflege der Software notwendig:** Verfügt der Cloud Speicher über Zusatzfunktionen wie beispielsweise ein Weboffice, so kümmert sich der Anbieter um die einwandfreie Funktionalität der Software.

Folgende Risiken gehen mit der Nutzung von Cloud Speichern einher: [Bu12, S. 2f]

- **Nicht-Verfügbarkeit / Dienstausfall:** Ist der Zugriff auf in der Cloud abgelegte Daten nicht möglich, so kann dies Geschäftsprozesse stören oder ganz zum Stillstand bringen. Abhängig davon um welche Prozesse es sich handelt, drohen bei längeren Ausfallzeiten finanzielle Verluste oder Imageschäden.
- **Datenverlust / Stilllegung eines Cloud Storage Services:** In vielen Bereichen gibt es Dokumente und Daten, die von entscheidender Bedeutung sind. Werden diese ausschließlich in der Cloud abgelegt, so besteht die Eventualität, dass der Provider selbst oder erfolgreiche Angreifer Daten löschen oder verändern. Ebenso kann ein Dienst vom Netz genommen werden. In beiden Fällen gehen dabei Informationen verloren, wenn diese nicht vorher auf andere Datenlager kopiert werden oder die Vorlaufzeit dazu nicht ausreicht. Speziell im geschäftlichen Umfeld können neben erheblichen finanziellen Belastungen auch rechtliche Konsequenzen folgen, wenn dadurch beispielsweise gesetzliche Aufbewahrungsfristen nicht eingehalten werden können.
- **Verlust der Vertraulichkeit der Daten:** Neben der Verfügbarkeit des Online-Speichers und der darin abgelegten Daten hat vor allem auch die Vertraulichkeit der Informationen einen hohen Stellenwert. Gelingt es Angreifern, Zugang zu sensiblen Daten zu erlangen und diese unerlaubt einem breiteren Personenkreis zugänglich zu machen, so drohen neben einem erheblichen Imageverlust auch rechtliche Konsequenzen und finanzielle Einbußen.
- **Verlust der Integrität der Daten:** Bei der Übertragung von Daten, deren Bearbeitung über das Netz oder deren abschließender Speicherung können Integritätsprobleme auftreten.
- **Verstoß gegen Datenschutzbestimmungen:** Handelt es sich bei den an den Cloud Service Provider übermittelten Daten um personenbezogene Daten im Sinne der Datenschutz-Grundverordnung (DSGVO), so ist auf den physischen

Speicherort der Daten zu achten. Laut DSGVO dürfen personenbezogene Daten von EU-Bürgern nicht in einem Rechtsstaat aufbewahrt werden, dessen geltenden Regelungen nicht den EU-Mindeststandards entsprechen. Eine Ausnahme bilden dabei die USA. US-Unternehmen dürfen auch dann Daten empfangen, wenn sie nach dem Privacy-Shield-Programm [US19] zertifiziert sind. Bei einer Zuwiderhandlung besteht das Risiko, gegen bestehendes Recht zu verstoßen. Neben einer möglichen Schädigung des eigenen Rufs kann das auch hohe Bußgelder nach sich ziehen.

- **Unsichere Client-Software:** Sofern die Client-Software des Cloud Service Providers Schwachstellen aufweist, ist auf diesem Weg ebenfalls der Zugriff Unbefugter auf die Daten der Anwenderin oder des Anwenders möglich.

2.3.3 Cloud Storage Security

Basis einer sicheren Kommunikation mit einem Cloud Storage Provider sind geeignete Zugangskontrollmechanismen. Neben eigenen Implementierungen kommen hier auch standardisierte Methoden zum Einsatz. Eine weitverbreitete Möglichkeit für die Umsetzung einer sicheren API-Autorisierung bietet das offene Protokoll OAuth (Open Authorization) in der aktuellen Version 2.0. Namhafte Unternehmen nutzen es, um Services von Drittanbietern auf die eigenen Ressourcen zugreifen zu lassen [Bi15, S. 4]. Benutzername und Passwort werden dabei nicht offen gelegt. Anstelle dessen wird mit einem Access Token gearbeitet, das von einem Autorisierungs-Server bereitgestellt und an den Drittanbieter übergeben wird. Das Access Token ist zeitlich begrenzt und erlaubt den Zugriff auf die dafür zugelassenen Ressourcen.

Neben geeigneten Zugangskontrollmechanismen ist auch die sichere Übertragung der Daten entscheidend. Viele Cloud Storage Services werben damit, die Daten zu verschlüsseln. Doch Verschlüsselung ist nicht gleich Verschlüsselung. Bei aktuellen Services wird zwischen drei Arten von Verschlüsselung unterschieden.

Die Verschlüsselung der Daten kann dabei

- ausschließlich auf dem Transportweg (Data-in-Motion),
- am schlussendlichen Speicherort der Daten (Data-at-Rest) oder
- bereits auf dem Client des jeweiligen Benutzers (End-to-End)

erfolgen. Die Verschlüsselung der Daten auf dem Transportweg (Data-in-Motion Verschlüsselung) wird in der Gegenwart von einer überwiegenden Anzahl der Anbieter standardmäßig zur Verfügung gestellt. Es entstehen dabei weder zusätzliche Kosten noch zusätzlicher Einrichtungsaufwand für den Kunden. Zu beachten ist dabei, dass damit lediglich ein Schutz vor unerwünschtem Zugriff auf dem Weg zwischen Quelle und Ziel gewährleistet ist. [Bu12, S. 5]

Eine Steigerung des Sicherheitsniveaus ist durch zusätzliche Verschlüsselung der Daten am schlussendlichen Speicherort (Data-at-Rest) gegeben. Der Anbieter ist dabei allerdings für das Schlüsselmanagement verantwortlich und dadurch potentiell in der Lage, die Daten zu entschlüsseln und einzusehen. Der Schutz entfach seinen Nutzen lediglich dann, wenn Daten entwendet werden. In verschlüsselter Form sind diese für den Angreifer in der Regel nicht direkt weiterverwendbar. [Bu12, S. 5]

Die Verschlüsselung der extern gespeicherten Daten bereits auf dem Client der Nutzerinnen und Nutzer vorzunehmen (End-to-End) stellt die sicherste Methode zur Gewährleistung der Vertraulichkeit dar. Dem Cloud Storage Provider werden dabei ausschließlich verschlüsselte Daten übergeben. Manche Cloud Storage Provider haben eine solche Verschlüsselungsfunktionalität bereits in ihrer Client-Software integriert. [Bu12, S. 5] Es handelt sich dabei allerdings um weniger namhafte Anbieter. Mögliche Gründe dafür sind Nachteile in Bezug auf die Deduplizierung oder den blockweisen Upload von Daten. Datendeduplizierung bedeutet, dass zwei gleiche Dateien in unterschiedlichen Verzeichnissen nur einmal beim Cloud Storage Provider abgelegt werden. Dadurch kann Speicherplatz eingespart werden. Blockweiser Upload bedeutet, dass durch einen Abgleich von lokal liegenden Daten mit den Daten, welche online abgespeichert sind, jene Blöcke identifiziert werden, welche verändert wurden. Um die

Internet-Bandbreite zu schonen werden nicht die vollständigen Dateien geladen, sondern lediglich die geänderten Blöcke online aktualisiert [Mu11, S. 3].

2.3.4 Welcher Anbieter eignet sich zur sicheren Ablage von schützenswerten Daten in der Cloud?

Entscheidet man sich, Cloud Storage Services zu nutzen, so empfiehlt es sich, die Gefahren und Risiken genau unter die Lupe zu nehmen, um sie für den konkreten Anwendungsfall zu bewerten.

Die in Kapitel 2.3.2 genannten Risiken lassen sich in Gruppen zusammenfassen. Für jede dieser Gruppen wurde in Tabelle 1 eine Maßnahme festgelegt, mittels der sich die Eintrittswahrscheinlichkeiten oder das Schadensausmaß reduzieren lässt.

Tabelle 1: Risiken und mögliche Maßnahmen (Cloud Storage Services)

Gruppe #	**Mögliches Risiko**	**Minimierung Eintrittswahrscheinlichkeit / Schadensausmaß durch …**
1	• Verstoß gegen Datenschutzbestimmungen	Ablage an geografisch zulässigen Orten
2	• Verlust der Vertraulichkeit der Daten • Verlust der Integrität	End-to-End Verschlüsselung
3	• Nicht-Verfügbarkeit / Dienstausfall • Datenverlust / Stilllegung eines Cloud Storage Services	Redundante Ablage der Daten
4	• Unsichere Client-Software	Keine Verwendung eines bereitgestellten Clients

Um nicht gegen Datenschutzbestimmungen zu verstoßen, empfiehlt es sich, lediglich Provider zu verwenden, die Daten an geografisch zulässigen Orten ablegen. Tabelle 2 beinhaltet eine Aufteilung der bekanntesten Cloud Storage Services in IaaS- und SaaS-

Angebote (siehe Kapitel 2.2). Personenbezogene Daten dürfen auf jedem der Services abgelegt werden. Dies wird entweder durch den Standort der Server oder durch die Mitgliedschaft am Privacy-Shield-Programm [US19] ermöglicht.

Tabelle 2: Beispiele für Cloud Storage Services (Quelle: angelehnt an [Cl19a])

Cloud Storage Services (IaaS)	Cloud Storage Services (SaaS)
• Amazon S3 • Google Cloud Storage • Microsoft Azure • Backblaze B2 • Rackspace Cloud Files	• Dropbox • Google Drive • Microsoft OneDrive • Amazon Cloud Drive • Apple iCloud • Box • pCloud • SugarSync • Telekom MagentaCloud • Tresorit

Eine End-to-End Verschlüsselung (siehe Kapitel 2.3.3) trägt dazu bei, sowohl die Vertraulichkeit als auch die Integrität der Daten zu wahren. Da bei IaaS-Angeboten keine Client-Software existiert, können diese naturgemäß nicht mit direkter End-to-End Verschlüsselung angeboten werden. Von den angeführten SaaS-Angeboten unterstützt lediglich das Produkt Tresorit [Tr19a] die Möglichkeit einer End-to-End Verschlüsselung.

Bei Ausfall oder Stilllegung eines Services kann die Gesamtverfügbarkeit nur dann aufrecht erhalten werden, wenn die Daten zuvor auf einem zweiten Provider abgelegt wurden.

Inwieweit angebotene Client-Softwarepakete sicher sind lässt sich nicht messen. Es ist vielmehr Vertrauenssache. Programmierfehler oder Backdoors können auch bei implementierter End-to-End Verschlüsselung die Sicherheit beeinträchtigen. Es empfiehlt sich, bei kritischen Anwendungen nicht auf bereitgestellte Clients zurückzugreifen.

Um alle Maßnahmen zu berücksichtigen, müssen die Daten somit auf mehreren Providern abgelegt werden, ohne Client-Software zu verwenden, die von den Cloud Storage Providern zur Verfügung gestellt wird. Up- und Download der Dateien sind in Folge per API vorzunehmen. Die Verschlüsselung ist im Sinne einer End-to-End Verschlüsselung vor dem Upload durchzuführen.

Werden die Dateien zusätzlich so zerteilt, dass kein Provider im Besitz der vollständigen Daten ist, so erreicht man zusätzliche eine Unabhängigkeit von veralteten Authentifizierungs- und Verschlüsselungsalgorithmen.

Verwendet man beispielsweise drei Cloud Service Provider zur Ablage von Daten und beschickt jeden davon nur mit zwei von drei Teilen der Daten, so benötigt die erreichte verteilte Speicherung nicht mehr Speicherplatz, als die vollständige Ablage der Daten auf zwei Cloud Storages. Die Sicherheit ist allerdings unter Verwendung dieses Ansatzes um ein Vielfaches höher als bei vollständiger Übertragung der Daten zu einem einzigen Provider. Abbildung 4 zeigt eine schematische Darstellung der anzuwendenden verteilten Speicherung in der Cloud.

Abbildung 4: Verteilte Speicherung in der Cloud

Durch die beschriebenen Maßnahmen können prinzipiell alle Services aus Tabelle 2 für die sichere Ablage von Daten verwendet werden, sofern sie APIs zur Verfügung stellen. Zu beachten gilt es, dass manche SaaS-Angebote auf IaaS-Angeboten aufbauen. So wird beispielsweise der Service Dropbox [Dr18] auf Amazon S3 [Am19e] betrieben [Mu11, S. 3]. Ein weiteres Kriterium für die Auswahl können die Kosten der einzelnen Services sein. Darauf wird in einem späteren Abschnitt näher eingegangen.

Die Antwort auf die Frage aus der Überschrift dieses Kapitels lässt sich folglich damit beantworten, dass sich alle der in Tabelle 2 dargestellten Services für die sichere Ablage von schützenswerten Daten in der Cloud eignen. Voraussetzung ist, dass mehrere Services gleichzeitig verwendet werden, um darauf Daten verteilt abzulegen. Außerdem sollten die einzelnen Services APIs zur Verfügung stellen und zu keinem der anderen verwendeten Services in Abhängigkeit stehen.

2.3.5 Bestehende Services zur verteilten Speicherung in der Cloud

2.3.5.1 SecureBeam

Das Konzept von SecureBeam [Be19] berücksichtigt, die Daten der Nutzerinnen und Nutzer vor dem Upload zu zerteilen. Erst danach werden die entstandenen Datenfragmente verschlüsselt auf verschiedene Cloud Speicher verschoben. SecureBeam basiert auf einem Freemium-Modell. Die kostenlose Version unterstützt dabei die Kombination der Cloud Services Dropbox [Dr18] und Google Drive [Go19a]. Nach dem Google Drive 15 Gigabyte und Dropbox 2 Gigabyte kostenlos zur Verfügung stellen, können durch die Kombination bereits 17 Gigabyte kostenloser Speicherplatz erschlossen werden. Durch In-App-Käufe ist es möglich, weitere Provider anzubinden.

Abbildung 5 zeigt SecureBeam in Form einer mobilen Android-Anwendung. Weitere unterstützte Plattformen sind iOS und Windows. Konkrete Beschreibungen zu den Funktionen des Windows-Clients gibt es keine. Die Entwicklung dürfte mittlerweile eingestellt worden sein. Die aktuellste Version im Google Play Store stammt aus dem Jahr 2015 [Se19]. Auch durchgeführte Tests zeigten, dass die Funktion auf aktuellen Betriebssystemversionen nicht mehr gegeben ist.

Abbildung 5: SecureBeam (Quelle: entnommen aus [Se19])

Zusammengefasst ist SecureBeam somit ein interessanter Ansatz für die verteilte Speicherung von Daten, der leider nicht mehr weiterentwickelt wird.

2.3.5.2 MultCloud

Bei Multcloud [Mu19] ist es möglich, über eine Weboberfläche verschiedene Cloud Storage Services zu verwalten. Es können Dateien direkt über eine Weboberfläche von einem Cloud Storage Provider zum anderen kopiert oder verschoben werden. Zusätzlich ist es möglich, ausgewählte Daten mittels Share-Funktion zu teilen. Außerdem kann eine Synchronisierung eingerichtet werden, die Daten von einem Service im Hintergrund mit einem anderen Service abgleicht. Abbildung 6 beinhaltet eine Darstellung einer beispielhaften Übersicht, die nach dem Login auf der Weboberfläche von MultCloud erscheint.

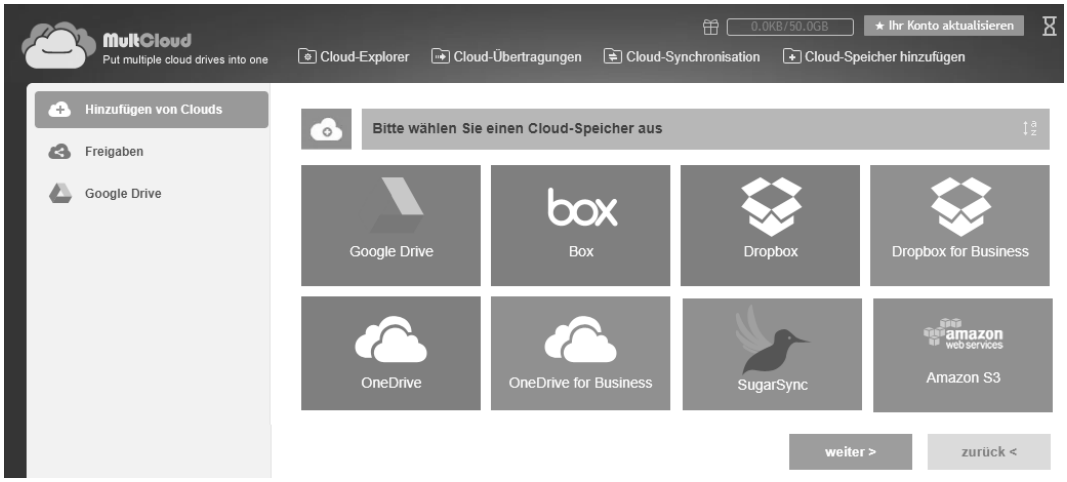

Abbildung 6: MultCloud (Quelle: entnommen aus [Da19d])

MultCloud ist somit eine Applikation, mit der verschiedene Cloud Storages bequem zentral verwaltet werden können. Der Ansatz einer verteilten Speicherung von Daten in der Cloud wird allerdings nicht unterstützt.

2.3.5.3 Weitere Services

Die Services Otixo [Ot19], odrive [Od19], CloudFuze [Cl19b] und cloudHQ [Cl19c] sind sehr ähnlich zu MultCloud. Zentrale Motivation ist jeweils die Vereinigung von mehreren Cloud Storage Services in einer Applikation. Der zentrale Zugriff erfolgt entweder über Webinterface oder über einen offline installierten Client. Von dort aus können Dateien kopiert, verschoben, synchronisiert oder geteilt werden.

Eine wie in Kapitel 2.3.4 dargestellte redundante, verteilte und verschlüsselte Speicherung von Daten auf bestehenden Public Cloud Services ist mit den genannten Produkten somit nicht möglich.

3. Grundlagen Datensicherung

Eine Datensicherung (Backup) bezeichnet das Kopieren von Daten mit der Absicht, diese im Fall eines Datenverlustes wiederherstellen zu können [Pa13, S. 52].

Bevor nun diskutiert wird, wie die in Kapitel 2.3.4 dargestellte sichere Ablage von Daten in der Cloud im Rahmen von Datensicherungskonzepten verwendet werden kann, gilt es, die Notwendigkeit von Datensicherungen zu erörtern.

3.1 Business Continuity Management und Disaster Recovery

Der zunehmende Einfluss der Informationstechnologie auf sämtliche Geschäftsprozesse erhöht das Ertragspotential von Unternehmen. Gleichzeitig steigert diese Entwicklung jedoch auch die Verwundbarkeit. Deshalb ist es für Unternehmen essentiell, entsprechende Notfallpläne zu entwickeln. Sowohl der Begriff des Business Continuity Management (BCM) als auch die untergeordneten Teilaufgaben Business Continuity Planning (BCP), Incident Response (IR) und Disaster Recovery (DR) spielen dabei eine wesentliche Rolle.

Das Business Continuity Management (BCM) ist ein ganzheitlicher Managementprozess, welcher durch Planung präventiver Maßnahmen, gezielte Vorbereitung eines Notfall- und Krisenmanagements sowie unverzüglicher Wiederherstellung unterbrochener Prozesse die Stabilität einer Organisation in Notlagen gewährleisten und eine Unterbrechung des Geschäftsbetriebs trotz widriger Umstände vermeiden soll [Bo14, S. 14]. BCM soll demnach also verhindern, dass es zu keiner Betriebsunterbrechung kommt. Falls dieses Szenario doch eintritt, soll das BCM helfen, durch systematisch durchzuführende Schritte die Betriebsfähigkeit in einer angemessenen Zeit wieder zu erlangen. Zu potentiellen Business Continuity Problemen zählen neben Geschehnissen im Bereich der IT auch Krankheit oder Weggang von wichtigen Team-Mitgliedern, Ausfälle von Zulieferketten oder Ausfälle aufgrund von Katastrophen [Te14].

Für BCP, IR und DR sei Folgendes definiert:

- Business Continuity Planning: *„The process of developing prior arrangements and procedures that enable an organization to respond to an event in such a manner that critical business functions can continue within planned levels of disruption.“ [Di18, S. 8]*
- *Incident Response (IR): „The response of an organization to a disaster or other significant event that may significantly impact the organization, its people, or its ability to function productively“ [Di18, S. 44]*
- Disaster Recovery (DR): *„The process, policies and procedures related to preparing for recovery or continuation of technology infrastructure, systems and applications which are vital to an organization after a disaster or outage.“ [Di18, S. 26]*

Abbildung 7 zeigt die zeitliche Anordnung dieser Aufgabenbereiche, die sich direkt aus den Definitionen ableiten lässt.

Abbildung 7: Zeitliche Abfolge BCP, IR und DR (Quelle: entnommen aus [Ru08, S. 10])

Desaster unterbrechen die Notfallvorsorge und müssen durch Sofort- (IR) und Wiederherstellungsmaßnahmen (DR) in den Ausgangszustand gebracht werden. In der Praxis sind BCP und DR eng miteinander verbunden, liegen aber in der Abbildung auf verschiedenen Seiten einer Desaster-Situation. Als Desaster gilt jedes plötzlich auftretende Ereignis, dass die Fähigkeit einer Organisation zur Aufrechterhaltung von kritischen Funktionen, Prozessen oder Diensten über einen inakzeptabel langen Zeitraum beeinträchtigt. BCP und DR teilen das gemeinsame Ziel, durch die Aufrechterhaltung solcher kritischen Geschäftsprozesse wirtschaftliche Schäden auf ein Minimum zu reduzieren. BCP beschäftigt sich dabei mit der Notfallvorsorge und ist proaktiv. Incident Response beschreibt Sofortmaßnahmen, die direkt nach dem Eintreten eines Desasters zu treffen sind. Das Disaster Recovery ist gleich wie die

Sofortmaßnahmen reaktiv und folgt unter optimalen Bedingungen nur den Instruktionen eines vorbereiteten Plans. [Ru08, S. 10]

Die darauf folgende Phase wird als Disaster Recovery bezeichnet. Diese beschäftigt sich unter Zuhilfenahme des DR Plans mit dem Wiederanlauf der Geschäftsaktivitäten. Im Rahmen der Informationstechnik umfasst DR sowohl die Datenwiederherstellung als auch das Ersetzen von nicht mehr benutzbarer Hardware. Es werden Server oder Mainframes mithilfe von Backups wiederherstellt, PBX-Geräte (Private Branch Exchange) wieder einsatzfähig gemacht oder LANs (Local Area Networks) in Stand gesetzt. Spätestens mit der Aufnahme des Normalbetriebs ist die Disaster Recovery Phase beendet. [Te14]

Bei der Erstellung einer Disaster Recovery Lösung sind folgende Punkte zu berücksichtigen, welche aus einer Business Impact Analyse (BIA) resultieren: [Mo14, S. 6]

- **Recovery Time Objective (RTO):** Wie lange darf ein Geschäftsprozess ausfallen? Bei der Recovery Time Objective handelt es sich um die Zeit, die vom Zeitpunkt des Schadens bis zur vollständigen Wiederherstellung der Geschäftsprozesse (Wiederherstellung von Infrastruktur, Daten, Nachbearbeitung von Daten) vergehen darf. Der Zeitraum kann hier von 0 Minuten (Systeme müssen sofort verfügbar sein) bis mehrere Tage oder in Einzelfällen Wochen betragen.
- **Recovery Point Objective (RPO):** Wie viel Datenverlust kann in Kauf genommen werden? Bei der Recovery Point Objective handelt es sich um den Zeitraum, der zwischen zwei Datensicherungen liegen darf, das heißt, wie viele Daten oder Transaktionen dürfen zwischen der letzten Sicherung und dem Systemausfall höchstens verloren gehen. Wenn kein Datenverlust hinnehmbar ist, beträgt die RPO 0 Minuten.

Die Business Impact Analysis (BIA) ist ein wichtiger Bestandteil eines Business Continuity Plans. Die BIA wird durchführt, um die für die Organisation wesentlichen Kernprozesse zu erkennen und ihre Kritikalität zu bewerten. [KSK11, S. 18]

Es gibt keine formellen Standards und die Methodologie kann je nach Unternehmen variieren. Der BIA-Report umfasst meist eine kurze Zusammenfassung, Informationen zur Methodologie, detaillierte Ergebnisse verschiedener Geschäftsabteilungen sowie Charts und Diagramme, um potentielle Verluste zu illustrieren und Empfehlungen für das Desaster Recovery darzustellen. Der Report priorisiert die wichtigsten Geschäftsfunktionen und überprüft die Auswirkungen bei Ausfällen. Darüber hinaus legt er tolerierbare Ausfallzeiten und Verluste fest und listet diese in Form von RTOs und RPOs auf.

3.2 Hochverfügbarkeit versus Disaster Recovery

In vielen Unternehmen, welche gegenwärtig Informationstechnologie einsetzen, lassen sich Geschäftsprozesse finden, welche zumindest innerhalb der gewöhnlichen Geschäftszeiten de facto ausfallfrei und jedenfalls ohne Datenverlust funktionieren müssen. Beispiele sind ERP-Systeme, ohne die keine Produktion möglich ist oder Webshops, die in der Zeit des Ausfalls keinen Umsatz generieren. Typischerweise werden solche Prozesse im Rahmen einer BIA (vergleiche Kapitel 3.1) wie folgt bewertet [St19a, S. 1]:

- RTO < 5 Minuten pro Monat
- RPO = 0

Dies entspricht einer Verfügbarkeit von 99,99%. Die Frage, ob geplante Unterbrechungen wie beispielsweise für Wartungsarbeiten als Ausfallzeit zählen oder nicht, soll an dieser Stelle nicht weiter diskutiert werden. Diese oder vergleichbare Werte sind nur mit hochverfügbaren Clustersystemen zu erreichen. Solche Cluster-Systeme bestehen aus mehreren eigenständigen Serverknoten, welche in verschiedenen Rechenräumen betrieben werden. Fällt aufgrund von Hard- oder Softwarefehlern ein Ressourcenknoten aus, so können hochverfügbare Dienste automatisch auf andere Serverknoten geschoben werden. In vielen Fälle sind solche Failover möglich, ohne dass die aktuellen Verbindungen der Nutzerinnen und Nutzer abbrechen.

Die Frage ist, ob eine Hochverfügbarkeitslösung eine Sicherung der Daten überflüssig macht. Betrachten wir dazu die Definition für Disaster Recovery (DR) in Kapitel 3.1. Entsprechend dieser verhilft ein Disaster Recovery dazu, Geschäftsprozesse, nach Eintreten einer natürlichen oder durch den Menschen verursachten Katastrophe, wiederherzustellen. Als Katastrophe zu werten sind aus Sicht der IT jedenfalls Ereignisse wie Feuer, Erdbeben oder Hochwasser, sofern IT-Ressourcen von den Auswirkungen betroffen sind. Doch es gibt auch viele kleinere Fehler, die als Katastrophe betrachtet werden müssen. Aus Sicht von Streppel [St19a] ist es sinnvoll, mögliche Fehlersituationen, die im Rahmen einer Business Impact Analysis (BIA) als Bedrohung für den IT-Betrieb gesehen werden, in Klassen einzuteilen. Darauf aufbauend kann entschieden werden, wo eine Katastrophe beginnt, und wie der Schutz gegen Fehler dieser Klassen aussehen kann.

Im Folgenden werden die vier Fehlerklassen definiert [St19a, S. 2]:

- **Klasse K1:** Einfache Fehler, die vor allem die Hardware betreffen. Dies sind beispielsweise Ausfälle von Festplatten oder Netzteilen und einfache Ausfälle aufgrund von Softwarefehlern.
- **Klasse K2:** Schwerwiegende Softwarefehler. Sowohl Betriebssystemfehler als auch komplexere Hardwarefehler, wie beispielsweise der Ausfall von kompletten Servern.
- **Klasse K3:** Mehrfachfehler wie zum Beispiel gleichzeitiger Ausfall von zwei Servern, aber auch der gleichzeitige Ausfall von unterschiedlichen Komponenten. Dazu kommen Fehler in kritischen Komponenten wie zum Beispiel in einer Cluster-Software, die ja gerade Redundanz über mehrere Systeme hinweg koordinieren soll. In Summe enthält diese Fehlerklasse komplexe Fehlersituationen, die in ihrer Art nicht vorhersehbar sind.
- **Klasse K4:** Datenkorruption und Datenverluste aller Art. Vor allem aber verursacht durch administrative Fehler, Ransomware oder Feuer, Erdbeben und Hochwasser.

Fehler der Fehlerklasse K1 werden gegenwärtig durch redundante Hardwarekomponenten abgefedert. Fehler der Klasse K2 werden durch Redundanz auf

Systemebene oder durch darauf aufbauende Cluster-Systeme abgesichert. Damit können komplette Systemausfälle sowohl auf Hardware- als auch auf Softwareebene kompensiert werden. Bei Fehlern der Klasse K3 helfen Redundanzen nicht mehr. Hier handelt es sich in der Regel um eine Katastrophe, die nur mit einer getrennten Umgebung und mit geeigneten Umschaltprozeduren bewältigt werden kann [St19a, S. 2]. Fehler der Klasse K4 sind ebenfalls als Katastrophe zu werten. Teilweise wird ein Datenstand benötigt, der nicht innerhalb von Sekundenbruchteilen von neuen Daten überschrieben wird. Bei anderen Szenarien muss auf Datensicherungen zurückgegriffen werden, welche an einem geografisch unterschiedlichen Ort abgelegt sind.

Die wesentliche Eigenschaft einer Katastrophe ist, dass ihr Ablauf nicht vorhersehbar und damit die Abwehrmaßnahmen nicht oder nur sehr schwer planbar sind. Daraus ergeben sich nun Anforderungen an eine Disaster Recovery Lösung, mit der die Folgen einer Katastrophe minimiert werden können. Aus der Definition der Fehlerklassen wird klar, dass eine wesentliche Anforderung an eine Datensicherung darin besteht, dass mindestens zwei weitgehend voneinander unabhängige Datenbestände zur Verfügung stehen. Die Idee ist, dass ein Fehler, so komplex und katastrophal er auch sein mag, sich nicht auf alle Datenbestände ausbreiten kann.

Die Anforderungen an RPO und RTO im Falle einer Katastrophe sind üblicherweise erheblich geringer. Das ergibt sich daraus, dass die Eintrittswahrscheinlichkeiten relativ niedrig sind und die technischen Lösungen vergleichsweise sehr teuer. Typische einzuhaltende Werte sind [St19a, S. 2]:

- RTO < 24 Stunden
- RPO < 1 Stunde

3.3 Sicherungsarten

Im Zuge der Erstellung von Datensicherungen werden folgende Sicherungsarten unterschieden [LC03, S. 19]:

- **Komplett-/Vollsicherung:** Bei dieser Sicherungsart werden bei jedem Sicherungslauf sämtliche zur Sicherung vorgesehenen Dateien vollständig

gesichert. Vollsicherungen sind einfach durchzuführen, und die Wiederherstellung der Daten ist ebenso mit wenig Aufwand möglich. Allerdings ist der Speicherbedarf sowie die Durchlaufzeit aufgrund der jeweils zu sichernden Datenmenge vergleichsweise hoch.

- **Inkrementelle Sicherung:** Bei der inkrementellen Sicherung werden immer nur die Dateien gespeichert, die seit der letzten inkrementellen Sicherung, oder im Falle der ersten inkrementellen Sicherung seit der letzten Komplettsicherung, geändert wurden oder neu hinzugekommen sind. Es wird also immer auf der letzten inkrementellen Sicherung aufgesetzt. Dieses Verfahren hat den Nachteil, dass bei einer Wiederherstellung die Daten aus mehreren Sicherungen zusammengesucht werden müssen. Vorteil der inkrementellen Sicherung ist der sehr geringe Speicherbedarf. Das Verfahren eignet sich daher für die Datensicherung in Netzwerken oder in der Cloud. Andererseits sind prinzipbedingt alle Inkremente miteinander verkettet, weshalb es nur mit großem Rechenaufwand möglich ist, ein Inkrement zwischen zwei anderen Inkrementen zu entfernen, etwa um Speicherplatz zu sparen.
- **Differenzielle Sicherung**: Bei der sogenannten differenziellen Sicherung werden alle Dateien, die seit der letzten Komplettsicherung geändert wurden oder neu hinzugekommen sind, gesichert. Es wird also immer wieder auf der letzten Komplettsicherung aufgesetzt, wobei gegenüber einer neuen Vollsicherung Speicherplatz und Zeit gespart werden kann. Wenn eine Datei verändert wurde, wird die jeweilige Version der Datei bei jedem differenziellen Lauf gesichert. Vorteilhaft ist der deutlich reduzierte Speicherbedarf im Vergleich zu einer Vollsicherung und eine überschaubare Komplexität, da die aktuellste Datensicherung immer nur einen Schritt von der letzten Vollsicherung entfernt ist. Ebenfalls von Vorteil ist, dass nicht mehr benötigte Sicherungsstände unabhängig voneinander gelöscht werden können, während im Vergleich dazu inkrementelle Sicherungen zwangsläufig miteinander verkettet sind. Bei sehr großen Dateien, die sich häufig ändern, ist die differenzielle Sicherung nachteilig. Trotz nur kleiner Änderungen beinhaltet jede differenzielle Sicherung die seit der letzten Vollsicherung geänderten oder hinzugefügten Daten in vollständiger Form.

- **Speicherabbildsicherung:** Bei der Speicherabbildsicherung (Image-Sicherung) wird ein kompletter Datenträger (Festplatte, USB-Massenspeicher, optische Medien) oder eine einzelne Partitionen in Form eines 1-zu-1-Abbilds gesichert. Es kann so neben den Nutzdaten auch das gesamte Dateisystem inklusive Betriebssystem gesichert werden. Der Vorteil dieser Sicherung besteht darin, dass bei einem Totalausfall eines Systems das Speicherabbild auf einen Datenträger zurückgeschrieben werden kann. Dadurch wird der Zustand der Datenträger wieder vollständig auf den Sicherungszeitpunkt hergestellt.

3.4 Sicherungshäufigkeit und Anzahl der Sicherungskopien

Die Häufigkeit einer Datensicherung orientiert sich daran, wie wichtig beziehungsweise wie geschäftskritisch die zu sichernden Daten sind. Die Recovery Point Objectives (RPOs siehe Kapitel 3.1) beschreiben, wie viel Datenverlust in Kauf genommen werden kann. So ist es vorstellbar, dass ein einzelner Client-PC nie gesichert wird. Wenn die wichtigen Daten der Benutzer auf zentralen Netzlaufwerken liegen würde eine regelmäßige Speicherabbildsicherung (siehe Kapitel 3.3) mehr Kosten verursachen als ein defekter PC, der im Schadenfall neu zu installieren ist. Im Gegensatz dazu kann es bei Datenbanken oder zentralen Dateiablagen notwendig sein, innerhalb eines einzelnen Tages mehrfache Datensicherungen durchzuführen. Das Intervall ist so zu wählen, dass die Kosten für die Datensicherung die durch den Datenverlust entstandenen Schäden nicht übersteigen.

Die Antwort auf die Frage nach der Anzahl an anzufertigenden Sicherungskopien lässt sich mit der weitreichend akzeptierten 3-2-1-Regel von Peter Krogh finden [Kr09, S. 207]:

- Es sollten mindestens **drei Kopien** der Daten vorhanden sein.
- Die Kopien sollten auf **zwei unterschiedlichen Medien** gespeichert werden.
- Eine Backup-Kopie sollte an einem **externen Speicherort** abgelegt werden.

Drei Kopien bedeutet, dass zusätzlich zu den primären Daten mindestens zwei weitere Sicherungen vorhanden sein sollten. Das Resultat ist eine stark reduzierte Wahrscheinlichkeit, dass das primäre System und die Sicherungsdatenträger

gleichzeitig ausfallen. Angenommen die Originaldaten liegen auf Medium 1 und das Backup wird auf Medium 2 aufbewahrt. Wenn die Ausfallwahrscheinlichkeit für Medium 1 und auch für Medium 2 bei 1/100 liegt, so ergibt sich die Wahrscheinlichkeit für den gleichzeitigen Ausfall beider Medium, unter der Annahme dass beide Geräte aus unterschiedlichen Gründen und damit unabhängig voneinander ausfallen, mit 1/100 * 1/100 = 1/10.000. Werden nun die primären Daten auf Medium 1 und zwei Backups auf den Medien 2 und 3 gespeichert, so beträgt die Wahrscheinlichkeit eines gleichzeitigen Ausfalls aller drei Medien bereits 1/100 * 1/100 * 1/100 = 1/1.000.000. Mit jeder zusätzlichen Sicherungskopie kann somit das Risiko von Datenverlusten in Folge eines gleichzeitigen Ausfalls aller Medien reduziert werden. [Ve16]

Die beschriebene Minimierung des Risikos lässt sich nur dann erreichen, wenn die verschiedenen Medien unabhängig voneinander ausfallen. Stimmen für die Daten des primären Systems und für die Daten der Datensicherung nicht nur Sicherungsmedium sondern auch der Speicherort überein, so kann das dazu führen, dass diese nicht mehr unabhängig voneinander ausfallen (Vibrationen, Temperatur, Feuer, …). Aus diesem Grund besagt die 3-2-1-Regel, dass die Kopien der Daten auf mindestens zwei unterschiedlichen Speichertypen aufbewahrt werden sollten. Beispielsweise auf einem internen Festplattenlaufwerk und einem Wechseldatenträger (Bandlaufwerk, externe Festplatte, USB-Laufwerk, SD-Karte, CD, DVD oder sogar Diskette [Ve16]. Von entscheidender Bedeutung ist es auch, die Kopien physisch voneinander getrennt aufzubewahren. Es ist nicht empfehlenswert, jede Sicherungskopie in dem Raum aufzubewahren, in dem sich auch das produktive Speichersystem befindet. Bei bestimmten Katastrophen (siehe Kapitel 3.2) wären alle Daten unwiederbringlich verloren. [Ve16]

3.5 Sicherungsmedien

Aus technischer Sicht sind alle Arten von Datenträgern als Sicherungsmedien geeignet. Im einfachsten Fall kann es ausreichen, wöchentlich den gesamten vorhandenen Datenbestand in Form eines manuellen Tasks auf eine CD-ROM, DVD oder Blu-ray zu brennen. Auch USB-Sticks, Flash Speicherkarten oder externe USB-Festplatten können verwendet werden.

Je kürzer die geforderten Zeitabstände sind, in denen Sicherungen angefertigt werden müssen, desto mehr Arbeits- und Zeitaufwand bedeutet allerdings das manuelle Kopieren der Daten. Deshalb ist es ab einer bestimmten Anzahl an Sicherungen in einem betrachteten Zeitraum sinnvoll, geeignete Sicherungssoftwareprodukte und spezielle Sicherungslaufwerke einzusetzen. Auf dem Markt erhältliche Sicherungssoftware ist auch für komplexere Sicherungsaufgaben, wie das Sichern von Datenbanken, geeignet und mit einer größeren Auswahl an Sicherungsmedien kompatibel. Übliche Ziele für Datensicherungen dieser Art sind USB-Festplatten, Network Attached Storages (NAS), Storage Area Network-Systeme (SAN) und Bandlaufwerke.

Für kleine bis mittlere Datenmengen lässt sich auch die Möglichkeit der Speicherung in der Cloud nutzen. Der Vorteil dieser Methode besteht darin, dass die Daten außer Haus abgelegt werden. Es wird so eine räumliche Trennung der Sicherungen von den Originaldaten erreicht, ohne dass ein Wechseldatenträger manuell von einem an einen anderen Ort transportiert werden muss. Ein großes Augenmerk ist dabei allerdings auf die Seriosität und Sicherheit der Cloud Service Provider zu legen. Beinhalten die Daten Geschäftsgeheimnisse oder hängen ganze Existenzen von Unternehmen im Extremfall von der Verfügbarkeit der gesicherten Daten ab, so empfiehlt es sich, die Daten nicht bei einem einzelnen Cloud Storage Provider abzulegen. Abhilfe schafft der im Rahmen dieser Arbeit vorgestellte Ansatz der verteilten Speicherung von Daten in der Cloud. Durch die verteilte Ablage der Daten ist keiner der eingebundenen Provider im Besitz der vollständigen Daten. Gleichzeitig sind die Daten bei Ausfall eines einzelnen Providers unverändert verfügbar. Details dazu sind Kapitel 2.3 zu entnehmen.

Die Verwendung eines Online-Speichers für die Speicherung von großen Datenmengen oder die Nutzung von Cloud Storage Services als zentrales Sicherungsarchiv eines Unternehmens ist nicht immer zu empfehlen. Neben den genannten Sicherheitsbedenken besteht ein weiterer Grund darin, dass sämtliche Daten über das Internet übertragen werden müssen. Bei direkter Sicherung auf einen Cloud Speicher können sich die Laufzeiten von Datensicherungen aufgrund einer begrenzten Internet-Bandbreite merklich erhöhen. Noch gravierender können die Auswirkungen sein, wenn die Daten im Desaster-Fall zur Rücksicherung benötigt werden. Laut einer Umfrage der Statistik Austria [St18a] verfügt die Mehrheit der österreichischen Unternehmen unter

249 Beschäftigten über eine Internetanbindung, welche eine Download-Geschwindigkeit von maximal 100 Mbit/s zulässt. Die Dauer für den Download eines einzelnen Terabytes an Daten würde mit einer für diese Umfrage durchschnittlichen Internetleitung mehr als 1,8 Tage benötigen. Geht man davon aus, dass eine Vollsicherung (siehe Kapitel 3.3) für ein bezogen auf diese Umfrage durchschnittliches Unternehmen mit rund 125 Beschäftigten mehrere Terabyte an Daten umfasst, so lässt sich die Cloud als möglicher Speicherort für die Ablage einer Vollsicherung ausschließen. Die Internet-Bandbreite alleinig für die Unterstützung einer Cloud-Sicherungsstrategie zu erhöhen ist in vielen Fällen kaufmännisch nicht sinnvoll. In manchen Regionen wäre es auch technisch gar nicht möglich, die Bandbreite ausreichend zu erhöhen.

Da das Verhältnis von vorhandener Internet-Bandbreite zur Größe einer Vollsicherung stark variieren kann, sollten für den weiteren Verlauf dieser Arbeit zwei Typen von Unternehmen betrachtet werden.

- **Unternehmenstyp 1:** Datenvolumen* / Internet-Bandbreite < 5 Stunden
- **Unternehmenstyp 2:** Datenvolumen* / Internet-Bandbreite > 48 Stunden

  * *Das im Zuge eines DR im schlechtesten Fall zu ladende Datenvolumen.*

Mit der Internet-Bandbreite ist sowohl die Bandbreite für den Upload, als auch für den Download gemeint. Im unternehmerischen Umfeld sind die beiden Werte häufig gleich. Diese Gegebenheit wird auch als „symmetrische Internetanbindung“ bezeichnet.

Ist wie in Kapitel 3.2 angeführt für ein Disaster Recovery 24 Stunden Zeit, so wäre es für Unternehmenstyp 1 möglich, sämtliche Daten in der Cloud abzulegen. Voraussetzung dafür ist, dass auch die genutzten Cloud Storage Provider eine ausreichende Transfergeschwindigkeit zulassen und die Zeit für eine Inbetriebnahme von Ersatz für zerstörte Hardware vor der Rücksicherung ausreicht [Di15, S. 44]. Dies gilt es im Anlassfall zu prüfen. Unternehmenstyp 2 benötigt jedenfalls ein Medium ungleich der Cloud, sofern Datensicherungen sinnvoll außer Haus abgelegt und RTO sowie RPO aus Kapitel 3.2 berücksichtigt werden sollten.

Tabelle 3 beinhaltet eine Aufstellung der Sicherungsmedien, welche aktuell von Bedeutung sind.

Tabelle 3: Sicherungsmedien

Medium	Beschreibung
Optische Medien	CD, CD-RW, DVD+-, DVD-RW, Blue-Ray
Flashspeicher	USB Sticks, Flash Speicherkarten, SSD-Festplatten
Hard Disks	Magnetische Festplatten - intern, extern, SATA, SAS, SCSI
Netzwerkspeichersysteme	NAS, SAN
Magnetbänder	
Cloud	IaaS, SaaS

Optische Medien wie CD, CD-RW, DVD+-, DVD-RW und Blu-Ray eignen sich nur für die Sicherung von kleineren Datenmengen. Für gewerbliche Zwecke ist das verfügbare Datenvolumen in vielen Fällen zu gering. Für das Beschreiben sind entsprechende Brenner notwendig. Die Lagerung und Aufbewahrung gestaltet sich aufwendig. Die optimale Umgebungstemperatur liegt bei +25 Grad Celsius und die optimale Luftfeuchtigkeit zwischen 40 und 60 Prozent. Kratzer, Aufkleber die chemische Lösungsmittel im Klebstoff enthalten, die Beschriftung mit einer zu harten Spitze sowie Sonneneinstrahlung reduzieren die Haltbarkeit dieser Medien. [Kr16] Die Speicherkapazität liegt bei CDs in etwa bei 700 Megabyte, bei DVDs bei 8,5 Gigabyte und bei Blu-ray-Discs bei 50 Gigabyte. Eine Blu-Ray-Disc kann von den genannten optischen Medien am schnellsten beschrieben sowie gelesen werden. Die Datenraten liegen bei 432 Mbit/s für Schreiben und bei 72 MByte/s für Lesen [Bl19]. Die Kosten für ein Gigabyte Speicherkapazität liegen aktuell bei 0,03 €.

USB Sticks, Flash Speicherkarten und SSD Festplatten eignen sich im Allgemeinen besser zur Sicherung von Daten. Hersteller garantieren eine verlässliche Funktionsfähigkeit bei bis zu 100.000 Schreibzyklen. Diese gilt es nicht zu überschreiten. Durch den hohen Preisdruck kommen oftmals Speicherchips von schlechter Qualität zum Einsatz. Flash-Speicher sind bereits mit sehr hohen Speicherkapazitäten erhältlich und im Gegensatz zu HDDs gegen Erschütterungen unempfindlich. Sehr schnelle Zugriffszeiten und Transferraten zeichnen diese Medien ebenfalls aus. [Kr16] Am Markt sind im Moment Flash-Speicher mit Kapazitäten bis zu 16 Terabyte erhältlich. Die

Transferraten liegen bei 490 MByte/s für Schreiben und 510 MByte/s für Lesen [So19]. Die Kosten für ein Gigabyte Speicherkapazität liegen aktuell bei 0,16 €.

Magnetische Festplatten können per USB- oder FireWire-Schnittstelle an einen PC oder Server angeschlossen sowie in Netzwerkspeichern (NAS- oder SAN-Systemen) verbaut werden. In Form von Netzwerkspeichern eignen sich magnetische Festplatten vorzüglich zur Speicherung von größeren Datenmengen. Sie sind preiswert und bieten große Kapazitäten bei geringem Platzbedarf. Im dauerhaften Betrieb überleben Festplatten nur wenige Jahre. Unter Verwendung von RAID-Systemen können Hardwareschäden allerdings weitgehend unter Kontrolle gebracht werden. Bei Verwendung von nicht fest verbauten Festplatten in Form von Wechseldatenträgern, beispielsweise zur externen Ablage von Sicherungen, ist zu beachten, dass Erschütterungen und Stöße zum Totalschaden der Datenträger führen können. [Kr16] Am Markt sind magnetische Festplatten im Moment mit Kapazitäten bis zu 14 Terabyte erhältlich [ID19]. Die maximalen Transferraten für Lesen und Schreiben liegen bei 120 MByte/s [Sp19]. In NAS- oder SAN-Systemen können abhängig von der Konfiguration mehrere Festplatten gleichzeitig beschrieben werden. Aus diesem Grund erreichen solche Systeme höhere Schreib- und Leseraten als einzelne Festplatten. Ein Gigabyte Speicherkapazität kostet aktuell 0,04 €.

Magnetbänder (Tapes) finden nahezu ausschließlich im Geschäftsumfeld und bei sehr großen Datenmengen Anwendung. Sie speichern große Kapazitäten zu einem sehr günstigen Medienpreis. Magnetbänder sind bei richtiger Lagerung sehr lange haltbar. Einmal beschrieben kann ein Tape bis zu 30 Jahre lang lesbar sein. [Kr16] Zum Lesen oder Schreiben der Tapes sind spezielle Laufwerke notwendig. Es existieren verschiedene Typen von Magnetbändern. Zu den bedeutendsten zählt der Typ LTO. Linear Tape Open (LTO) steht für eine Spezifikation von Magnetbändern und der entsprechenden Bandlaufwerke, welche von IBM, HP und Seagate in Form eines Gemeinschaftsprojekts erarbeitet wurde. Die aktuellsten Tapes aus der LTO-Reihe gehören der Generation LTO-7 an. Die maximale Speicherkapazität liegt bei 15 Terabyte. Die Transferraten für Lesen und Schreiben liegen bei 300 MByte/s [He15b]. Die Kosten für ein Gigabyte Speicherkapazität liegen aktuell bei 0,004 €.

Die Abrechnung für die Nutzung von Cloud Storage Services folgt in der Regel individuellen Preisstaffeln der verschiedenen Anbieter. Beispielsweise ist der monatliche Verrechnungspreis bei Nutzung des IaaS-Angebots Amazon S3 [Am19e] abhängig vom genutzten Speicherplatz, der Upload- und Download-Volumen sowie der Anzahl der Upload- und Download-Anfragen [Am19f]. Abhängig vom jeweiligen Anwendungsfall können neben dem S3 Standard-Speicher auch Produkte wie S3 Infrequent Access-Speicher oder Glacier-Speicher gewählt werden. Diese Produkte weisen erhöhte Zugriffszeiten im Bereich von Minuten bei niedrigeren Verrechnungspreisen auf. Die S3 Glacier-Speicherung eignet sich laut Angabe des Herstellers zur Ablage von Datensicherungen [Am19a]. Dropbox [Dr18] stellt für jeden Benutzer 2 Gigabyte Speicher kostenlos zur Verfügung. Obendrein gibt es Pakete mit einem oder zwei Terabyte, welche zu monatlichen Fixpreisen erhältlich sind. Ähnlich verhält es sich bei Google Drive [Go19a]. Hier stehen für jede Nutzerin und jeden Nutzer 15 Gigabyte Speicherplatz zur freien Verfügung. Darüber hinaus gibt es Pakete mit 100 Gigabyte und einem beziehungsweise zehn Terabyte. Microsoft OneDrive [Mi19a] stellt 5 Gigabyte Speicherplatz kostenfrei zur Verfügung. Zusätzlich sind Pakete mit einem Umfang von 50 Gigabyte beziehungsweise einem und sechs Terabyte erhältlich.

Aufgrund der beschriebenen Preisstaffelungen ist es notwendig, für die Darstellung konkreter Kosten zusätzliche Annahmen zu treffen. Aufbauend auf die zuvor in diesem Kapitel definierten Unternehmenstypen seien folgende beiden Anwendungsfälle definiert:

- **Anwendungsfall 1:** Unternehmenstyp 1 nutzt die Cloud zur vollständigen Ablage seiner Sicherungsdaten. Eine Vollsicherung benötigt ein Speichervolumen von 2 Terabyte. 10 Versionen sollten insgesamt vorgehalten werden. Es steht eine Internet-Bandbreite von 1 Gbit/s zur Verfügung, welche die genutzten Provider in vollem Umfang unterstützen. Der benötigte Speicherplatz ergibt sich somit mit 20 TB. Das monatliche Upload-Volumen wird mit 8,1 Terabyte angenommen. Dies entspricht einer wöchentlichen Vollsicherung und 100 Gigabyte an neu generierten Daten. 8 Terabyte an alten Sicherungsdaten werden im selben Zeitraum gelöscht. Die Anzahl der monatlichen Upload-Anfragen wird mit 700 angenommen. Dies berücksichtigt in etwa eine

Transferanfrage pro Stunde. Regelmäßiger Download findet keiner statt. Nachdem in dieser Arbeit vorgestellten Prinzip der verteilten Speicherung (siehe Kapitel 2.3.4) ergibt sich pro Cloud Storage Provider ein maximaler Speicherbedarf von 13,33 Terabyte.

- **Anwendungsfall 2:** Unternehmenstyp 2 nutzt die Cloud ergänzend zu einer externen Ablage der Vollsicherung auf einem Medium ungleich der Cloud. Die verfügbare Internet-Bandbreite von 50 Mbit/s ermöglicht lediglich die regelmäßige Speicherung der neu generierten Daten. Die Anfertigung einer Vollsicherung, welche außer Haus abgelegt werden soll, sollte wöchentlich erfolgen. Nach erfolgter Ablage der Sicherung an einem externen Ort werden die Daten am Cloud Speicher gelöscht. Pro Monat werden 100 Gigabyte an Daten neu generiert. Dieser Wert entspricht somit in etwa dem monatlichen Upload-Volumen. Ebenso viele veraltete Sicherungsdaten werden im selben Zeitraum gelöscht. Das maximal benötigte Speichervolumen am Cloud Speicher beträgt 25 Gigabyte. Die Anzahl der monatlichen Upload-Anfragen wird mit 700 angenommen. Dies berücksichtigt in etwa eine Transferanfrage pro Stunde. Regelmäßiger Download findet keiner statt. Nach dem in dieser Arbeit vorgestellten Prinzip der verteilten Speicherung (siehe Kapitel 2.3.4) ergibt sich pro Cloud Storage Provider ein maximaler Speicherbedarf von 16,67 Gigabyte.

Die Kosten für die in Kapitel 2.3.4 aufgeführten Cloud Storage Services sind in Tabelle 4 dargestellt.

Tabelle 4: Kosten der Cloud Storage Services (Stand März 2019)

Service-ebene	Servicename	Anwendungsfall 1: 13,33 Terabyte	Anwendungsfall 2: 16,67 Gigabyte
SaaS	Dropbox	64,80 € / Monat*	9,99 € / Monat [Dr19]
SaaS	Google Drive	199,99 € / Monat	1,99 € / Monat [Go19b]
SaaS	Microsoft OneDrive	Nicht möglich	2 € / Monat [Mi19b]
SaaS	Amazon Cloud Drive	699,93 € / Monat	1,67 € / Monat [Am19b]
SaaS	Apple iCloud	Nicht möglich	1,06 € / Monat [Ap19]
SaaS	Box**	40,50 € / Monat*	4,50 € / Monat [Bo19]

SaaS	pCloud	Nicht möglich	3,99 € / Monat [PC19]
SaaS	SugarSync	Nicht möglich	6,67 € / Monat [Su19]
SaaS	Telekom MagentaCloud	Nicht möglich	1,95 € / Monat [Te19b]
SaaS	Tresorit	Nicht möglich	16 € / Monat [Tr19b]
IaaS	Amazon S3 (Glacier)	48,60 € / Monat	0,09 € / Monat [Am19f]
IaaS	Google Cloud Storage (Coldline Storage)	83,06 € / Monat	0,10 € / Monat [Go19c]
IaaS	Microsoft Azure (ZRS COOL)	130,53 € / Monat	0,16 € / Monat [Mi19c]
IaaS	Backblaze B2	62,22 € / Monat	0,9 € / Monat [Cl19e]
IaaS	Rackspace Cloud	1.215,37€ / Monat	11 € / Monat [Ra19]

* *Es sind zumindest 3 Benutzer zu abonnieren.*

** *Maximale Dateigröße 5 Gigabyte*.

Für die vorliegende Arbeit werden die Cloud Storage Services Amazon S3 Glacier [Am19a], Backblaze B2 [Cl19e] sowie Google Drive [Go19b] berücksichtigt. Die ersten beiden Services sind die für den beschriebenen Anwendungsfall 1 kostengünstigsten Services. Das SaaS-Angebot Box bleibt deswegen unberücksichtigt, weil die Uploads einer Beschränkung hinsichtlich einer Dateigröße von 5 Gigabyte [Bo19] unterliegen. Dies kann im Rahmen von Datensicherungen rasch zu einer unerwünschten Einschränkung führen. Google Drive stellt den wichtigsten Vertreter aus der Gruppe der SaaS-Angebote gemessen an den Nutzerzahlen dar. Der Service hat im letzten Jahr die Grenze von einer Milliarde Nutzerinnen und Nutzern überschritten [Mi18]. Gründe dafür sind weitreichende Integration in vielen Applikationen und ein vergleichsweise hohes Datenvolumen, das kostenfrei zur Verfügung steht. Der erste Verfolger ist mit etwa der Hälfte an aktiven Nutzerinnen und Nutzern der Service Dropbox [Fo18].

3.6 Sicherungsstrategien und Kosten

Für den Anwendungsfall 1 können, wie in Kapitel 3.5 beschrieben, sämtliche Daten in der Cloud gesichert werden. Durch die verteilte Speicherung in der Cloud werden sowohl die 3-2-1-Regel (siehe Kapitel 3.4) als auch die geforderten RTOs und RPOs für das Disaster Recovery (siehe Kapitel 3.2) unterstützt. Wie in Abbildung 8 dargestellt werden

in Summe zwei Sicherungskopien außer Haus gespeichert. Die Wahrscheinlichkeit, dass keine Sicherungskopie zugreifbar ist, ist aufgrund der verteilten Speicherung äußerst gering.

Im Anwendungsfall 2 können aufgrund der zur Verfügung stehenden Internetanbindung nicht alle Daten in der Cloud abgelegt werden. Aus diesem Grund soll hier ein unternehmensinterner Netzwerkspeicher (NAS) zum Einsatz kommen, auf dem die Sicherungskopie 1 abgelegt wird. Als Medium für die externe Sicherungskopie 2 wird das Magnetband gewählt. Die Lagerungseigenschaften, die niedrigen Kosten pro Gigabyte Speichervolumen und die kostengünstige Möglichkeit zur Skalierung sind Gründe, die auch in der Gegenwart noch für die Verwendung von Magnetbändern sprechen. Die 3-2-1-Regel (siehe Kapitel 3.4) ist durch die Verwendung der zwei technisch voneinander unabhängigen Medien ebenfalls erfüllt. Da die externe Sicherungskopie manuell außer Haus abgelegt werden muss, ist nur eine begrenzte Backup-Häufigkeit möglich. Im vorliegenden Fall wird von einem in der Realität häufig vorkommenden Intervall von einer Woche ausgegangen. Um die Zwischenzeit zu überbrücken, sollten zwischen diesen Sicherungsläufen jegliche Änderungen der Daten verteilt in der Cloud gespeichert werden. Auch dieser Sachverhalt ist in Abbildung 8 schematisch dargestellt.

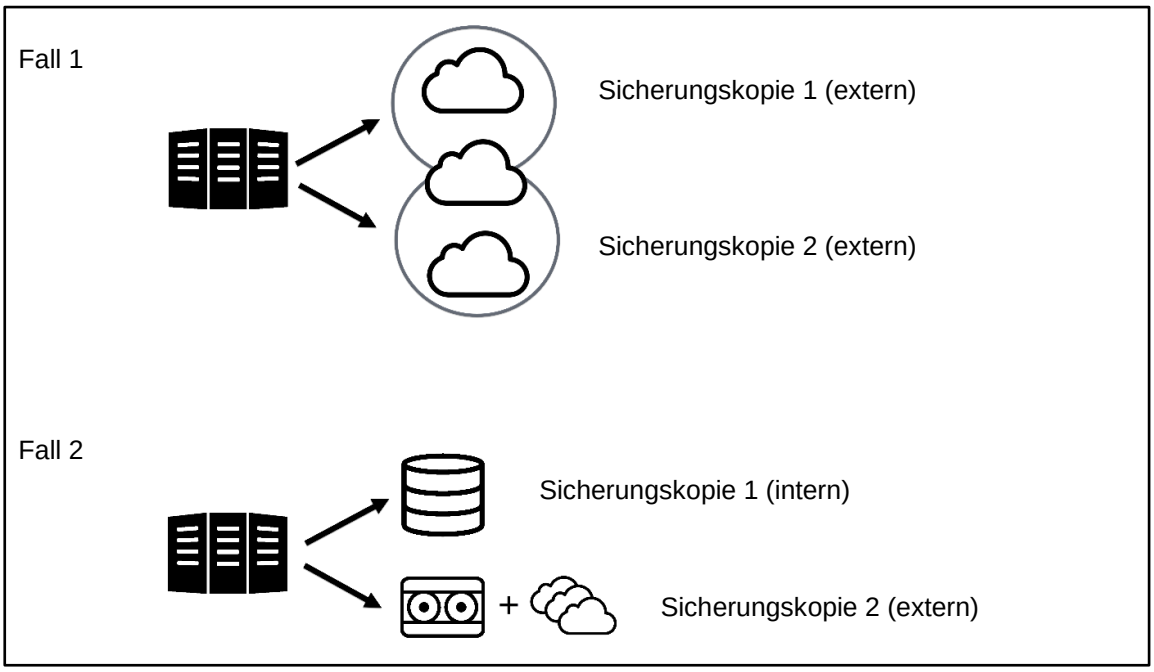

Abbildung 8: Sicherungsstrategien für Fall 1 und Fall 2

Zum Abschluss dieses Abschnitts erfolgt ein kaufmännischer Vergleich von Fall 1 und 2. Die Internetleitung wird als gegeben angenommen und verursacht aus Sicht der Datensicherung keine Kosten.

Für Fall 1 ergeben sich unter Verwendung der drei günstigsten IaaS-Angebote Amazon S3 [Am19a], Backblaze B2 [Cl19e] und Google Cloud Storage [Go19c] Kosten in Höhe von 193,88 € pro Monat. Die Werte pro Cloud Storage Service sind Tabelle 4 (Kapitel 3.5) zu entnehmen. Für Fall 2 entstehen zusätzlich Kosten durch die Anschaffung von Bandlaufwerk und NAS-System. Diese werden inklusive Inbetriebnahme mit 6000 € angenommen. Üblicherweise sind solche Systeme für einen Betrieb von 5 Jahren ausgelegt. Im Vergleich zu Fall 1 fallen außerdem sowohl Aufwände für den Betrieb der Systeme als auch für den Transport der Sicherungsbänder an. Wird wöchentlich eine halbe Stunde für den Transport der Bänder und zwei Stunden im Monat für den Betrieb der im Vergleich zu Fall 1 zusätzlich notwendigen Systeme angesetzt, so ergeben sich unter Berücksichtigung einer durchschnittlichen Arbeitskraft in der Systembetreuung mit 3 bis 5 Jahren Berufserfahrung (Monatsbrutto 3.100 € [Ba18]) Kosten von rund 100 € pro Monat. Die lohnabhängigen Abgaben für den Arbeitgeber wurden mit 30% vom

Bruttogehalt und die wöchentliche Normalarbeitszeit mit 38,5 Stunden angenommen. Ein durchschnittliches Monat umfasst 4,33 Wochen.

Im Anwendungsfall 1 ergeben sich auf 60 Monate gerechnet Kosten von rund 11.630 € während im Anwendungsfall 2 Kosten von rund 12.040 € entstehen. Angemerkt sei, dass für den Anwendungsfall 2 finanzielle Aufwände für die benötigte elektrische Energie, die anteilige Infrastruktur (Netzwerk, Serverraumequipment, …) sowie für die Nutzung der Cloud Storage Services vernachlässigt wurden.

Für die zwei genannten Anwendungsfälle sind die Kosten somit nicht das Kriterium, nachdem entschieden wird, ob es besser ist, Datensicherungen in der Cloud oder On-Premises abzulegen. Allgemein kann festgehalten werden, dass die Kosten für jedes Unternehmen und jeden Anwendungsfall genau zu prüfen sind. Es ist allerdings nicht auszuschließen, dass die Public Cloud mittlerweile bereits für viele Backup-Konzepte eine attraktive Möglichkeit zur Ablage von Daten darstellt.

3.7 Anbieter von Sicherungssoftware

Kommerzielle Sicherungssoftware wird derzeit aus mehreren Gründen eingesetzt. Kurz gefasst ergibt sich mittels Einsatz einer Sicherungssoftware die Möglichkeit, die Sicherung von heterogenen IT-Landschaften zu homogenisieren. Es ist nicht nur möglich, Filesystem-Sicherungen innerhalb verschiedener Betriebssysteme oder ganze Speicherabbildsicherungen durchzuführen, sondern es können auch direkt Applikationen, Datenbanken oder Virtualisierungshosts angesprochen und die Datenbestände gesichert werden.

Neben einem zentralen Management ist auch ein zentrales Monitoring möglich. Der Aufbau eines optimalen Sicherungszeitplans wird stark vereinfacht und das Ansprechen und Verwalten von benötigten Sicherungsmedien ermöglicht.

Einige Anbieter von Backup-Software unterstützen bereits die Ablage von Daten in der Cloud. Zu den wichtigsten Backup-Anbietern, welche eine Sicherung in der Cloud ermöglichen, gehören Acronis, Arcserve, Backblaze, Carbonite, Commvault, Dell EMC, Druva, iDrive, Unitrends, Veeam Software und Veritas Technologies [Te19a]. Keines der

Produkte unterstützt allerdings eine mit den Darstellungen in Kapitel 2.3.4 vergleichbare Möglichkeit der verteilten Speicherung in der Cloud.

4. Konzept und Implementierung

4.1 Beschreibung und Zielsetzung

Im Verlauf der Arbeit wurde bisher sowohl der Bereich der Cloud Storage Services als auch das Thema Datensicherung eingehend aufgearbeitet. Auf Basis der gewonnenen Erkenntnisse ist es nun möglich, die Eckpunkte für den geplanten Prototyp zu definieren.

4.1.1 Darstellung des gewünschten Funktionsumfangs

Für die Ablage der Daten sollten, wie in Kapitel 3.6 dargestellt, die Services Amazon S3 Glacier [Am19a], Backblaze B2 [Cl19e] und Google Drive [Go19b] zum Einsatz kommen. Die Ablage von personenbezogenen Daten ist auf diesen Services wie beschrieben möglich, ohne gegen Datenschutzbestimmungen zu verstoßen. Aus Gründen des Datenschutzes sollten die Daten bereits zusätzlich vor dem Upload bereits verschlüsselt und außerdem verteilt auf den einzelnen Cloud Storage Services abgelegt werden. Keiner der Provider soll im Besitz der vollständigen Daten sein. Die Datensicherheit wird durch die zuletzt genannte Maßnahme ebenso erhöht, da durch Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht eingeschränkt wird. Anstatt potentiell unsicherer Client-Software soll der Transfer der Daten über API-Aufrufe erfolgen. Der Prototyp sollte schlussendlich für die zwei in Kapitel 3.5 beschriebenen Anwendungsfälle verwendbar sein. Um die Internet-Bandbreite zu schonen, ist die Änderungen von Dateien zu berechnen und in komprimierter Form hochzuladen. Als Datenquelle soll für den Prototyp eine einfache Dateiablage dienen, innerhalb der mehrere Änderungen pro Stunde vorgenommen werden.

In Tabelle 5 sind die notwendigen Teilschritte aufgelistet, welche durchlaufen werden müssen, um Änderungen von Dateien in der Cloud abzulegen.

Tabelle 5: Ablauf für Upload einer Datei

Schritt #	Beschreibung
Schritt 1	Änderung von Datei erkennen
Schritt 2	Berechnung der Unterschiede zwischen zwei Dateien
Schritt 3	Komprimierung
Schritt 4	Dateipfad und Dateiname verbergen
Schritt 5	Verschlüsselung
Schritt 6	Aufteilung einer Datei in mehrere Teildateien
Schritt 7	Kommunikation mit den Cloud Speicherdiensten

Es soll demnach ein Programm implementiert werden, das in der Lage ist, Änderungen von Dateien zu erkennen. Für jede erkannte Änderung sollte sodann der Unterschied zum Stand aus der letzten Vollsicherung errechnet werden. Wurde die Datei neu erstellt, so entspricht die Dateiänderung im Vergleich zur letzten Vollsicherung der kompletten Datei. Unmittelbar danach sollte eine Komprimierung der Dateninhalte stattfinden ehe sie verschlüsselt und in einzelne Teile aufgeteilt werden. Die so generierten Dateifragmente sollten anschließend auf die gewählten Cloud Speicherdienste geladen werden. Wie in Kapitel 2.3.4 dargestellt sollte jeder der Cloud Storage Provider lediglich zwei von drei Teilen der Dateien erhalten.

Das Programm ist transparent zu gestalten. Die Anwenderin oder der Anwender sollte zur Förderung des Vertrauens jeden einzelnen Teilschritt nachvollziehen können. Sind für einzelne Schritte, wie beispielsweise Verschlüsseln oder Komprimieren, eigene Algorithmen zum Einsatz zu bringen, so sollten die dahingehenden Veränderungen einfach umsetzbar sein.

Abbildung 9 zeigt die Systemlandschaft, in die der Prototyp eingebettet wird.

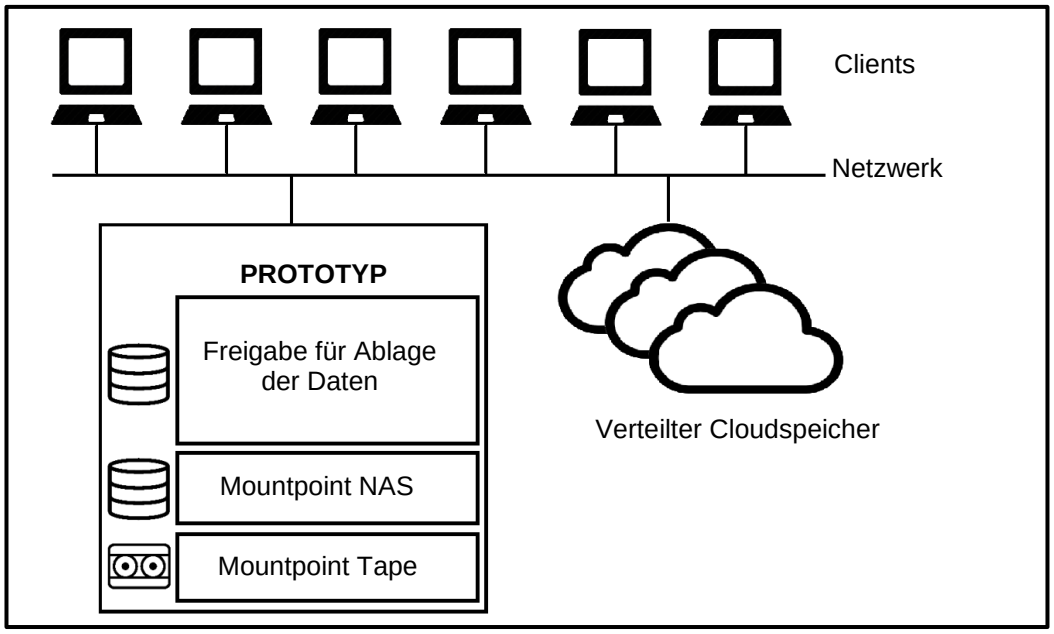

Abbildung 9: Systemlandschaft Prototyp

Sowohl NAS-System als auch Tape-Laufwerk werden im Prototyp als eigene Verzeichnisse im Dateisystem ausgeführt. In realen Systemlandschaften können an dieser Stelle NAS-System und Tape-Laufwerk gemountet werden.

4.1.2 Auswahl der zu nutzenden Plattform

Der geplante Prototyp soll innerhalb einer Linux-Umgebung entstehen. Das Betriebssystem Ubuntu [Ca19] soll dabei zum Einsatz kommen. Die Version 18.10 entspricht dem aktuellsten Release.

Es gibt mehrere Gründe die für diese Wahl sprechen. Zum einen gelten Linux-Betriebssysteme als vergleichsweise sicher. Die Gründe liegen hauptsächlich in der verhältnismäßig geringen Verbreitung dieser Systeme [St19b]. Die Attraktivität für Programmierer von Schadsoftware ist dadurch überschaubar. Außerdem erschwert Linux den Programmiererinnen und Programmierern von Schadsoftware zusätzlich die Arbeit durch Vielfalt und Heterogenität. Das Minderheitensystem spaltet sich weiter auf in diverse Distributionen, die sich technisch deutlicher unterscheiden als etwa ein altes Windows XP von einem aktuellen Windows 10. Eine Schadsoftware, die Ubuntu befallen kann, funktioniert wahrscheinlich unter einer anderen Distribution nicht. Die Windows-

Versionen sind hingegen so homogen, dass ein für Windows 98 geschriebener Virus theoretisch auch noch auf Windows 10 lauffähig sein kann. Darüber hinaus hat der Benutzer in Linux-Umgebungen nur Zugriff auf seine persönlichen Daten. Unter Windows ist der Benutzer standardmäßig ein Administrator und kann dadurch uneingeschränkt auf das gesamte System zugreifen. Jeder Virus, der unter Verwendung eines solchen Benutzers in ein System eindringt, hat die gleichen Rechte [Fi10, S. 695].

Weiters ist für die Umsetzung des Prototyps die Wahrung der Transparenz und der Erweiterbarkeit entscheidend. Hier eignet sich hervorragend die Nutzung der Shell. Die Shell wird von jedem Linux-Betriebssystem automatisch mitgeliefert. Es handelt sich dabei um einen Kommandozeileninterpreter (CLI - Command Line Interpreter), der von einer Benutzerin oder einem Benutzer eingegebene Kommandos liest und zur Ausführung bringt [He03, S. 1]. Neben der Aufgabe der Kommandointerpretation verfügt die Shell auch über Fähigkeiten, die von höheren Programmiersprachen bekannt sind. Es können beispielsweise Verzweigungen, Schleifen oder auch Funktionen gebildet werden. Dies macht die Shell zu einem mächtigen Werkzeug. Viele der im Rahmen dieser Arbeit benötigten Funktionen sind bereits Teil der Basisinstallation des Betriebssystems. Die Komplexität und der Installationsaufwand kann dadurch niedrig gehalten werden und die sicherheitskritischen Anwendungen stammen von einer öffentlichen Community.

Im Laufe der Zeit wurden unterschiedliche Shells entwickelt. Gelegentlich können besondere Anforderungen, wie beispielsweise besonders komfortable Programmierbarkeit, Unterstützung spezieller Programmierkonstrukte oder Kompatibilitätserwägungen, eine Rolle bei der Auswahl der Shell spielen. Sehr häufig kommt in der Linux-Welt aber die Bourne-Again-Shell (bash) zum Einsatz [SH18, S. 14]. Für den in dieser Arbeit umzusetzenden Prototyp soll eben diese Bourne-Again-Shell verwendet werden.

4.2 Technische Konzeptionierung der einzelnen Teilschritte

4.2.1 Änderung von Dateien erkennen

Es gibt verschiedene Strategien, um Änderungen von Dateien zu erkennen. Eine Möglichkeit ist es, Checksummen zu errechnen und mit denen des vorherigen Durchlaufs zu vergleichen. Im Allgemeinen ist aber die Berechnung von Checksummen mit sehr viel Rechen- und Zeitaufwand verbunden. Bei großen Dateien oder einem Verzeichnis mit vielen Dateien kann diese Operation rasch einen nicht vertretbaren Zeitaufwand verursachen.

Besser funktioniert die Verwendung von Zeitstempeln. In aktuellen Linux-Dateisystemen, wie beispielsweise dem Dateisystem EXT4, stehen drei für die Anwenderin oder den Anwender zugreifbare Zeitstempel zur Verfügung: [Mö13]

- **Access Time (atime):** Dieser Zeitstempel wird jedes Mal neu gesetzt, wenn der Inhalt der Datei ausgelesen wird. Es wird somit der letzte Zugriff auf den Inhalt der Datei festgehalten. Durch die Auswertung dieses Zeitstempels wäre es beispielsweise möglich, festzustellen, welche Dateien über einen längeren Zeitraum nicht mehr benutzt wurden.
- **Modify Time (mtime):** Dieser Zeitstempel wird gesetzt, wenn der Inhalt der Datei verändert wird.
- **Change Time (ctime):** Mit diesem Zeitstempel wird die Zeit gespeichert, zu der das letzte Mal einzelne Speicherblöcke am Dateisystem geändert wurden, welche im Zusammenhang mit der abgelegten Datei stehen. Bei einem Ändern des Dateiinhaltes werden sowohl die Dateigröße als auch der Dateiinhalt am Dateisystem aktualisiert. Beim Ändern von Zugriffsrechten wird dieser Zeitstempel ebenso aktualisiert. Aber auch bei einem Umbenennen der Datei, wird die ctime neu gesetzt. Es gibt eine einzige Ausnahme bei der die ctime nicht gesetzt wird. Wenn durch das Auslesen der Datei nur die Access Time neu gesetzt wird, dann wird die Change Time nicht neu gesetzt. Während sich die anderen beiden Zeitstempel durch Befehle auf eine bestimmte Zeit manipulieren lassen, ist es nicht möglich, die ctime auf diese Art zu verändern.

Die Change Time (ctime) eignet sich aufgrund der beschriebenen Eigenschaften wunderbar für die Identifikation von etwaigen Dateiänderungen. Abbildung 10 beinhaltet eine Darstellung des Befehls „stat“. Mittels dieses auf Ubuntu vorinstallierten Befehls können die genannten Zeitstempel ausgegeben werden.

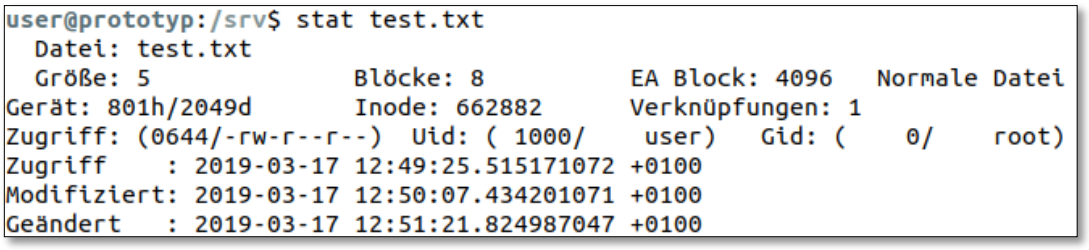

Abbildung 10: Ausgabe des Befehls "stat"

Innerhalb eines Ordners mit Metadaten soll für jede Datei der Zeitstempel in Form einer Datei abgelegt werden. Auf diese Weise steht diese Information beim nächsten Durchlauf für einen Vergleich zur Verfügung.

4.2.2 Berechnung der Unterschiede zwischen zwei Dateien

Um die Bandbreite zu schonen, sollte es möglich sein, nur die geänderten Teile einer Datei zu transferieren. Es gibt dazu einen in der Praxis gängigen Ansatz. Eine Datei wird dabei in Blöcke fester Größe unterteilt, um auf dieser Basis Vergleiche durchzuführen. Alle Blöcke die übereinstimmen werden nicht transferiert. Lediglich die geänderten Blöcke werden übertragen und mit der Zieldatei verschmolzen.

Ein Beispiel, bei dem nach diesem Prinzip gearbeitet wird, ist Dropbox [Dr18]. Die einzelnen Blöcke werden dabei auf Basis einer berechneten Prüfsumme verglichen. Die Wahrscheinlichkeit, dass die Hashwerte bei unterschiedlichen Inhalten der Blöcke übereinstimmen, wird als verschwindend gering bezeichnet [Mu11, S. 3].

Als weiteres Beispiel gilt das Programm „rsync“ in dem der gleichnamige Rsync-Algorithmus von Andrew Tridgell implementiert wurde [Rs19]. Das Programm „rsync“ kann dazu genutzt werden, um Daten über langsame Netzwerkverbindungen zu übertragen. Der Empfänger teilt dabei seine vorhandene Datei in Blöcke fester Größe auf, berechnet für jeden Block eine Prüfsumme und teilt nur diese Prüfsummen dem

Sender mit. Dieser durchforstet wiederum seine eigene Datei und ermittelt, welchen der Blöcke übereinstimmen. Am Ende übermittelt der Sender alle fehlenden Stellen zusammen mit Hinweisen, welche Blöcke der Empfänger an welcher Stelle wiederverwenden muss. Eine Kommunikation kann nur dann stattfinden, wenn sowohl Sender als auch Empfänger über das Programm „rsync“ verfügen. [Le07]

Die Aufgabe „Finde einen Abschnitt in der Datei, der eine bestimmte Prüfsumme besitzt, egal an welcher Stelle dieser Abschnitt liegt“ bedeutet auf den ersten Blick sehr viel Rechnerei. Andrew Tridgell fand allerdings eine Möglichkeit, nicht jedes Mal die komplette Prüfsumme zu ermitteln. Bei jeder neuen Position ändern sich schließlich nur zwei Stellen im Block. Ein Teil fällt hinten weg und ein neuer Teil kommt vorne hinzu. Tridgells Algorithmus braucht für die Berechnung der neuen Prüfsumme nur den herausfallenden und den neu hinzukommenden Teil. [Le07]

In Abbildung 11 ist dieser Sachverhalt grafisch dargestellt.

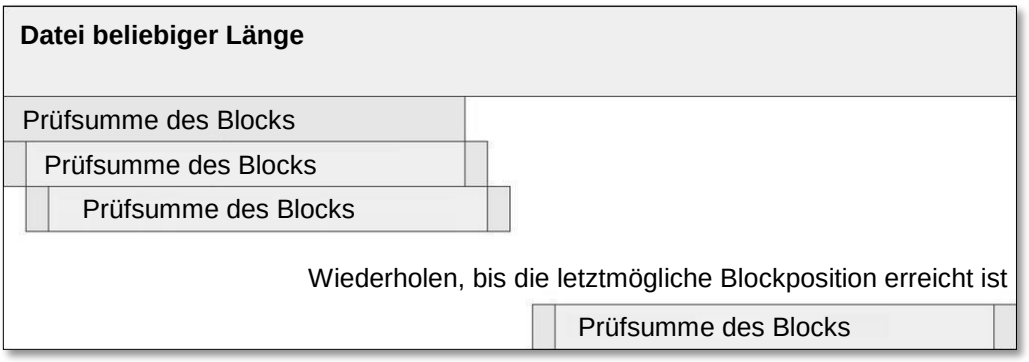

Abbildung 11: Rsync - Rollende Bildung der Prüfsumme (Quelle: entnommen aus [Le07])

Da diese Prüfsumme sehr anfällig für Kollisionen ist, wird bei ihrer Übereinstimmung noch eine zweite Prüfsumme gebildet. Bei der zweiten Prüfsumme ist im Gegensatz zur rollierenden Prüfsumme eine Kollision praktisch ausgeschlossen, allerdings ist ihre Berechnung sehr viel aufwendiger [TM96, S. 3]. Diese zweite Prüfsumme wird ab dem Release 3.0.0 unter Zuhilfenahme des MD5-Verfahrens gebildet [Ne19].

Für die vorliegende Arbeit kann das Programm „rsync“ nicht direkt verwendet werden. Grund dafür ist, dass die Vollsicherung auf einem anderen Medium liegen kann als die

Sicherung der geänderten Blöcke. An dieser Stelle ist das Programm „rdiff“ hilfreich. Dieses Programm arbeitet mit denselben Algorithmen wie „rsync“. Darüber hinaus bietet es aber auch die Möglichkeit, bestimmte Befehle explizit auszuführen. Folgende Operationen stehen zur Verfügung: [Rd19]

- **rdiff signature:** Erstellung einer Signaturdatei. Diese enthält pro Block die in den obigen Darstellungen beschriebenen Prüfsummen.
- **rdiff delta:** Erstellung einer Delta-Datei. Diese beinhaltet die inhaltlichen Unterschiede zwischen aktueller Datei und Signaturdatei.
- **rdiff patch:** Zum Verschmelzen von Delta-Datei mit der Originaldatei aus der Vollsicherung. Es kann dadurch ein Zustand zu einem Zeitpunkt hergestellt werden, der nach dem Zeitpunkt der Vollsicherung liegt.

Das Programm ist in den Standard-Paketquellen von Ubuntu enthalten.

4.2.3 Komprimierung

Um die nachfolgende Verschlüsselung der Daten zu beschleunigen, erfolgt die Komprimierung der Daten bereits einen Schritt davor. Im Themenkomplex der Komprimierung wird zwischen verlustbehafteter und verlustfreier Komprimierung unterschieden. Bei einer verlustfreien Kompression können die Originaldaten exakt aus den komprimierten Daten wiederhergestellt werden. Es geht keinerlei Information verloren. Im Gegensatz dazu steht die verlustbehaftete Kompression. Bei der verlustbehafteten Kompression können die Originaldaten aus den komprimierten Daten in der Regel nicht mehr exakt zurückgewonnen werden [Ki17, S. 1]. Das heißt, ein Teil der Information geht verloren. Solche Verfahren werden häufig zur Bild-, Video- oder Audiodatenkompression eingesetzt. Es geht dabei Qualität verloren, die aber möglicherweise für das menschliche Auge nicht sichtbar ist. Im Rahmen einer Datensicherung ist selbstverständlich von einer verlustfreien Kompression auszugehen.

Die Ziele eines jeden Kompressionsalgorithmus sind es, einerseits die Daten möglichst gut zu verdichten und andererseits, dabei sehr schnell und ressourcenschonend zu sein. Die genannten zwei Ziele stehen allerdings in der Regel in einem Widerspruch zueinander. Schnelle Kompression geht meist zu Lasten der Kompressionsrate und

umgekehrt. Während sich Textdateien sehr gut komprimieren lassen, gibt es Dateiformate, wie beispielsweise JPEG-Bilder, bei denen durch die Komprimierung keine große Speicherplatzersparnis mehr erzielbar ist.

Welcher Algorithmus das beste Ergebnis liefert hängt sehr stark vom zu komprimierenden Datenbestand ab. In der vorliegenden Arbeit wird von einem gemischten Datenbestand ausgegangen. Der Algorithmus LZ4 verspricht dazu einen guten Kompromiss zu liefern. Der Algorithmus ist primär auf hohe Kompressions- und Dekompressionsgeschwindigkeiten ausgelegt. Die durch die Komprimierung erreichte Reduzierung des Speicherplatzes ist im Verhältnis dazu respektabel [Se17] .

Das Programm „lz4“ ist in den Standard-Paketquellen von Ubuntu enthalten.

4.2.4 Dateipfad und Dateiname verbergen

Um die im Dateinamen oder in den Ordnerstrukturen enthaltene Information vor unbefugtem Zugriff zu schützen, sollten diese verborgen werden. Auf den Cloud Storage Services sollte jeweils lediglich eine flache Struktur mit den darin enthaltenen Teildateien entstehen. Um die Informationen über Dateipfad und Dateiname nicht zu verlieren, sind diese zuvor in das Archiv mit aufzunehmen, welches im Zuge der Komprimierung erstellt wird (siehe Abschnitt 4.2.3).

Basis für den Dateinamen am Online-Speicher sollte ein Hashwert berechnet aus dem vollständigen Dateipfad sein. Mit dem auf Ubuntu vorinstallierten Programm „sha256sum“ kann die gewünschte Zeichenkette mit einer fixen Länge von 256 Bit generiert werden.

Daran anknüpfen sollten ein Zeitstempel des Sicherungszeitpunktes sowie ein Indikator um zu erkennen, um welche Teildatei es sich handelt. Somit ist keine Komplexität einer Datenbank notwendig, um die Zuordnung eines kryptischen Dateinamen zum originalen Pfad abzuspeichern. Ein beispielhafter Dateiname ist in Tabelle 6 dargestellt.

Tabelle 6: Generierter Dateiname

SHA-256 Zeichenkette auf Basis des Dateipfads	Zeitstempel	*
689fd60d1debe854df6b1da3d05b1aa7de38c3bf9b7bcedf36480eefcf8c4e3d	20190318162320	1

** Teildatei (0-2)*

4.2.5 Verschlüsselung

Im Zuge der Verschlüsselung werden im Klartext vorliegende Ausgangsdaten mithilfe mathematischer Verfahren, sogenannten kryptographischen Algorithmen, in einen Geheimtext umgewandelt. Dies geschieht auf eine solche Weise, dass es nur dann möglich ist, aus den verschlüsselten Daten die Ausgangsdaten zu ermitteln, wenn man das Verfahren und den zugehörigen Schlüssel kennt. [He15a, S. 7]

Aktuelle Verfahren lassen sich aufteilen in symmetrische und asymmetrische Verfahren. Bei symmetrischen Verschlüsselungsverfahren gibt es nur einen Schlüssel, der zum Ver- und Entschlüsseln dient [Sc10, S. 7]. Diese Verfahren gelten als schnell, effizient und sicher. Wichtig dabei sind eine ausreichende Schlüssellänge und die sichere Aufbewahrung des Schlüssels.

Bei asymmetrischen Verschlüsselungsverfahren gibt es zwei Schlüssel: Den Public Key (Öffentlicher Schlüssel) und den Private Key (Privater Schlüssel). Wer im Besitz des Public Key ist, kann Nachrichten verschlüsselt weitergeben. Um die Daten zu entschlüsseln, wird der Private Key benötigt. Ein anschauliches Beispiel ist der Briefkasten. Jeder, der Zugang zum Briefkasten hat, kann Nachrichten einwerfen. Einmal im Briefkasten, kommt nur noch der Inhaber des Briefkastenschlüssels an die Nachricht heran. [Sc10, S. 13]

Da im Zuge von Datensicherungen keine außenstehenden Personen Daten verschlüsseln müssen, eignet sich für die vorliegende Arbeit ein symmetrisches Verfahren für die Verschlüsselung der Daten. Es soll der Advanced Encryption Standard (AES) zum Einsatz kommen. Dabei handelt es sich um ein symmetrisches Verschlüsselungsverfahren, das als Nachfolger für DES im Jahr 2001 im Auftrag des National Institute of Standards and Technology (NIST) als Standard bekannt gegeben wurde [Fe01]. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) empfiehlt,

dass die eingesetzten Schlüssel bei symmetrischen Verfahren mindestens 100 Bit lang sein sollten. [Bu19]

Entsprechend der Länge des verwendeten Schlüssels wird beim Advanced Encryption Standard (AES) zwischen AES-128, AES-192 und AES-256 unterschieden.

Im Prototyp für die vorliegende Arbeit sollten die Daten unter Verwendung des GNU Privacy Guard (GPG) [Th19] mit dem Verfahren AES-256 verschlüsselt werden. Der GPG hat das Ziel, einer möglichst großen Benutzergruppe die Verwendung von kryptographischen Methoden zu ermöglichen. In der aktuellsten Version von Ubuntu ist das Softwarepaket „gpg“ vorinstalliert.

Mit dem Programm „sha256sum“ soll aus einem Master-Passwort der Anwenderin oder des Anwenders ein Schlüssel mit ausreichender Länge generiert und für den automatisierten Zugriff in Form einer Datei abgelegt werden.

4.2.6 Aufteilung einer Datei in mehrere Teildateien

Die Aufteilung der Dateien in mehrere Teildateien ermöglicht das Programm „split“. Auch dieses Programm ist Teil des Auslieferungsumfangs von Ubuntu und bereits vorinstalliert. Unter Angabe der gewünschten Dateigröße für eine der Teildateien ist es möglich, diese generieren zu lassen. Im Rahmen der Rücksicherung ist es entscheidend, diese wieder in der richtigen Reihenfolge zusammenzufügen.

4.2.7 Kommunikation mit den Cloud Speicherdiensten

4.2.7.1 Representational State Transfer (REST)

Eine weitverbreitete Möglichkeit, mit Cloud Services in Form einer Maschine-zu-Maschine-Kommunikation zu kommunizieren, ist mittels Nutzung von REST APIs. Das Akronym REST steht für Representational State Transfer und bezeichnet ein Programmierparadigma. REST und HTTP werden häufig in einem Atemzug genannt. Das liegt daran, dass REST sehr häufig mit HTTP umgesetzt wird. In diesem Fall spricht man von RESTful HTTP [Ti15, S. 11]. HTTP basiert auf einem einfachen Request-Response-Modell, bei dem Nachrichten zwischen einem Client und einem entfernten

Server über ein Netzwerk ausgetauscht werden. Ein Server kann eine Response-Nachricht nur als Antwort auf eine Request-Nachricht versenden.

REST soll nachfolgend immer im Sinne von RESTful HTTP verstanden werden.

Die Grundprinzipien von REST lassen sich wie folgt zusammenfassen: [Ti15, S. 11]

- Eindeutige Identifikation von Ressourcen
- Verwendung von Hypermedia
- Verwendung von HTTP-Standardmethoden
- Unterschiedliche Repräsentationen von Ressourcen
- Statuslose Kommunikation

Jede Ressource braucht zur Identifikation einen eindeutigen Schlüssel. Für das Web eignen sich für diese Aufgabe Uniform Resource Identifiers (URIs). Diese einheitlichen Bezeichner für Ressourcen bilden einen globalen Namensraum. [Sp17, S. 144] In Abbildung 12 sind beispielhafte URIs dargestellt.

```
http://example.com/answers/42
http://example.com/agents/agent-007/cars
http://example.com/users?locked=true
```

Abbildung 12: Beispiele für URIs (Quelle: entnommen aus [Sp17, S. 144])

Der Begriff Hypermedia ist eine Mischung aus Hypertext und Multimedia. Während Hypertext ausschließlich Texte verknüpft, schließt Hypermedia auch Dokumente, Bilder, Töne, Videos und andere multimedialen Inhalte mit ein. Ein wesentliches Merkmal von Hypermedia stellen Links zur Verknüpfung verschiedener Medien dar. Links sind vor allem durch HTML sehr gut bekannt. Mit einem Browser kann man den Links leicht folgen. Sie dienen zur Ausführung von Aktionen und zur Navigation im Web. HTML ist daher ein klassisches Hypermediaformat. Die für den Client möglichen Aktionen und Navigationspfade werden vom Server über Hypermedia angeboten. Darüber hinaus können aber auch andere Formate mit Links verbunden werden. Das XML-Beispiel in Abbildung 13 zeigt eine Nachricht, deren Anhang nicht in die Nachricht eingebettet ist, sondern mittels einer URI referenziert wird.

```
<message self="http://example.com/messages/17">
  <body>...</body>
  <attachment ref="http://example.com/attachments/1701" />
</message>
```

Abbildung 13: XML-Nachricht mit URI-Referenz (Quelle: entnommen aus [Sp17, S. 146])

Der Empfänger dieser Nachricht kann einfach dem Link folgen um weitere Informationen zu erhalten. Der entscheidende Vorteil von URIs ist, dass auch Ressourcen in anderen Systemen referenziert werden können. [Sp17, S. 146]

Damit die Anwenderin oder der Anwender weiß, was mit der URI gemacht werden kann, braucht es eine einheitliche Schnittstelle. Die Schnittstelle der URIs besteht im Wesentlichen aus den HTTP-Methoden GET, HEAD, POST, PUT und DELETE. Ihre Bedeutung ist in der HTTP-Spezifikation definiert. Weil diese allgemeine Schnittstelle für jede Ressource verwendet wird, kann man ohne spezielles Vorwissen mit einem einfachen GET eine Repräsentation abrufen. GET ist beispielsweise sicher, sodass Clients keine Seiteneffekte zu erwarten haben, wenn sie diese HTTP-Methode wählen. Denn ein rein lesender Service ändert den Zustand der Daten nicht. [Sp17, S. 146]

Mit den bisher genannten Eigenschaften von REST können Clients eine Ressource über deren URI identifizieren und mit den bekannten HTTP-Methoden aufrufen. In welcher Form das Ergebnis repräsentiert wird, ist noch unklar. Die Lösung ist, dass die Clients das Format angeben, das sie bei der Kommunikation verwenden wollen. In Abbildung 14 ist ein Beispiel dargestellt.

```
GET /customers/1234 HTTP/1.1
Host: example.com
Accept: text/x-vcard
```

Abbildung 14: Definition der Repräsentationsform (Quelle: entnommen aus [Ti15, S. 17])

Der Client teilt die gewünschte Form der Repräsentation über eine Deklaration in den Header-Daten der Anfrage mit. In diesem Fall könnte eine Kundenadresse im vCard-Format zurückgeliefert werden. [Ti15, S. 18]

Das letzte Grundprinzip ist die statuslose Kommunikation. Konsequenterweise gibt es bei REST-konformen Anwendungen keinen Sitzungsstatus, der serverseitig über mehrere Client-Anfragen hinweg vorgehalten wird. Stattdessen muss der Kommunikationszustand im Client oder in der Repräsentation der Ressource gespeichert werden. Hierdurch wird die Kopplung zwischen Client und Server verringert. Diese Einschränkung bietet viele Vorteile. Zum Beispiel kann ein Server zwischen zwei Requests mit aktualisierter Software neu gestartet werden. Der Client würde es nicht merken. Genauso gut könnte zur Lastverteilung ein Load Balancer die Requests zu unterschiedlichen Serverinstanzen routen. Auch aufeinanderfolgende Requests eines Clients könnten von unterschiedlichen Serverinstanzen bearbeitet werden. [Sp17, S. 148]

4.2.7.2 APIs der gewählten Cloud Storage Services

Die drei für den Prototyp berücksichtigten Cloud Storage Services stellen allesamt REST APIs zur Verfügung.

Jeder der verwendeten Services muss folgende Operationen unterstützen:

- Download einer Dateiliste
- Download einer Datei
- Upload einer Datei
- Löschen einer Datei

Unter Amazon S3 steht eine GET-Methode [Am19c] bereit, welche sowohl den Download einer Datei als auch den Download einer Dateiliste erlaubt. Die Unterscheidung erfolgt über die in der Anfrage verwendete URI. Zum Download einer Datei ist darin der Dateiname zu spezifizieren. Für das Löschen einer Datei wird eine DELETE-Methode bereitgestellt. Der Upload einer Datei ist mittels Verwendung einer PUT-Methode möglich. Teil dieses Aufrufes ist auch die Spezifikation des Speichertyps. Für den vorhandenen Prototyp wird der Speichertyp „STANDARD“ verwendet. Dieser hat den Vorteil, dass Daten ohne Zeitverzögerung abgerufen werden können. Verwendet man, wie in Kapitel 3.5 angedacht, den Speichertyp „GLACIER“, so werden die Daten standardmäßig innerhalb von drei bis fünf Stunden zurückgegeben. Der beschleunigte

Datenabruf reduziert die Zeit bis zum Start des Downloads auf eine bis fünf Minuten [Am19a], diese Option ist aber kostenpflichtig. Dafür ist dieser Speichertyp aber der günstigste, den Amazon im Angebot hat.

Google Drive stellt ebenso eine GET-Methode [Go19d] zur Verfügung, welche den Download von speziellen Dateien sowie einer Dateiliste zulässt. Der Upload ist unter Verwendung einer POST-Methode möglich. Für das Löschen von Dateien steht eine DELETE-Methode zur Verfügung.

Unter Backblaze B2 werden die meisten Operationen jeweils rein über die URI referenziert. [Ba19b] enthält eine Auflistung der verfügbaren Funktionen. Für den Download einer Dateiliste kann die API-Operation „b2_list_file_names“ verwendet. Der Download einer Datei ist unter Verwendung der GET-Methode und Angabe eines Dateinamen möglich. Für den Upload einer Datei ist zuerst mit der API-Operation „b2_get_upload_url“ eine URI zu ermitteln, welche anschließend für den Upload einer Datei verwendet werden kann. Mit der API-Operation „b2_delete_file_version“ ist das Löschen einer Datei möglich.

Die jeweiligen Anfragen können mit dem Programm „curl“ gestellt werden. Das Programm ist in den Standard-Paketquellen von Ubuntu enthalten.

4.2.7.3 Authentifizierung und Autorisierung

Um Daten mittels der im vorigen Abschnitt beschriebenen Methoden auf den Cloud Storage Services ablegen zu können, ist es notwendig, die externe Anwendung gemäß den Vorgaben des jeweiligen Anbieters zu registrieren. Nach der Erstellung eines Benutzerkontos ist es bei jedem Anbieter notwendig, ein für den Prototyp spezifisches Schlüsselpaar bestehend aus Client-ID und geheimer Zeichenfolge zu generieren.

Bei Amazon S3 [Am19a] können direkt nach dem Login sogenannte „Buckets“ erstellt werden. Diese Buckets entsprechen Datenlagern, innerhalb der Daten abgelegt werden können. Darauf können Berechtigungen erteilt werden. Die Berechtigungen und Zugriffsschlüssel können im Identity and Access Management (IAM) von Amazon [Am19d] verwaltet werden. Jeder Zugriffsschlüssel besteht aus einer ID und einem

Secret, welche man durch das Hinzufügen eines neuen Benutzers erhält. Berechtigungen können vergeben werden, indem man einen zuvor erstellten Benutzer bearbeitet. Im einfachsten Fall können sogenannte „vorhandene Richtlinien“ verwendet werden, welche Amazon bereitstellt.

Nach dem erfolgreichen Login ist es bei Google Drive [Go19a] ebenso notwendig, die zugreifende Anwendung zu registrieren. Dies ist in der Google API Console [Go19e] möglich. Nach der Erstellung eines Projekts kann über die Schaltfläche „APIs und Dienste aktivieren“ die API für Google Drive aktiviert werden. Client-ID und Client-Secret erhält man, indem man anschließend am Ziel des Menüpunkts „Anmeldedaten“ einen neuen Eintrag für ein CLI-Tool anlegt.

Backblaze B2 [Ba19a] ermöglicht ähnlich wie Amazon S3 die Anlage von „Buckets« direkt nach dem Login. Nach der Anlage eines Buckets gelangt man in der Bucket-Übersicht über die Schaltfläche „Show Account ID and Application Key“ in einen Verwaltungsbereich, in dem sowohl Client-ID und Client-Secret generiert als auch Zugriffsberechtigungen verwaltet werden können.

4.3 Struktur der Implementierung

Im Folgenden wird die Struktur des Prototyps beschrieben.

Der Prototyp setzt auf einem Hauptverzeichnis auf, das folgende Ordner beinhaltet:

- 1_filestorage
- 2_nas
- 3_tape
- 4_script

Das Verzeichnis „1_filestorage“ stellt jenes Verzeichnis dar, in dem die zu sichernden Daten liegen. Diese Ressource könnte beispielsweise über das Netzwerk freigegeben und von verschiedenen Benutzerinnen und Benutzern beschrieben werden. Die Verzeichnisse „2_nas“ und „3_tape“ stellen die Mountpoints für das NAS-System und

das Tape-Laufwerk dar. Es können darin Vollsicherungen abgelegt werden. Im Verzeichnis „4_script“ sind die Programm- und Metadaten abgelegt.

Die Implementierung des Prototyps in Form einer Software umfasst folgende Programmdateien:

- 4_script/prototyp.sh
- 4_script/amazon_s3_glacier.sh
- 4_script/backblaze.sh
- 4_script/google_drive.sh
- 4_script/readme.txt

Die Programmdateien „amazon_s3_glacier.sh“, „backblaze.sh“ und „google_drive.sh“ sind ähnlich aufgebaut. Es sind darin die in Abschnitt 4.2.7.3 erwähnten Werte für „Client-ID“ und „Client-Secret“ einzutragen. Außerdem stellen sie die benötigten Funktionen für die Kommunikation mit den Cloud Storage Providern zur Verfügung. In Tabelle 7 sind die möglichen Programmaufrufe dargestellt.

Tabelle 7: Mögliche Programmaufrufe zur Interaktion mit den CSPs

Programmaufruf	**amazon_s3_glacier.sh**	**backblaze.sh**	**google_drive.sh**
./<cloud_storage_service>.sh get_onlinefile_list	X	X	X
./<cloud_storage_service>.sh download_file <filenname>	X	X	X
./<cloud_storage_service>.sh upload <filenname>	X	X	X
./<cloud_storage_service>.sh delete <filenname>	X	X	X
./<cloud_storage_service>.sh authorize			X

Mit dem Parameter „get_onlinefile_list“ kann eine Liste der Dateien abgerufen werden, welche am Cloud Speicherdienst zum Zeitpunkt des Aufrufs gespeichert sind. Mittels Verwendung der Parameter „download_file“, „delete“ sowie „upload“ und zusätzlicher Angabe des Dateinamens als zweiten Parameter können Dateien vom Online-Speicher heruntergeladen, gelöscht oder auf den Online-Speicher geladen werden. Der Parameter „authorize“ steht nur im Zusammenhang mit Google Drive zur Verfügung. Der Grund ist, dass bei diesem Service zur Autorisierung ein Login auf der Webseite des Anbieters notwendig ist. Der Programmaufruf mit diesem Parameter unterstützt die Nutzerinnen und Nutzer dabei und gibt entsprechende Anweisungen. Nähere Informationen beinhaltet die Datei „readme.txt“. Siehe dazu auch Anhang A.

Die Programmdatei „prototyp.sh“ beinhaltet alle zur Sicherung und Wiederherstellung der Daten notwendigen Funktionen. In Tabelle 8 sind die möglichen Aufrufe für dieses Programm dargestellt.

Tabelle 8: Mögliche Aufrufe für "prototyp.sh"

Programmaufruf	Beschreibung
./prototyp.sh set_password	Master-Passwort zur Verschlüsselung der Daten setzen.
./prototyp.sh fullbackup_nas_tape	Vollsicherung durchführen auf NAS-System und Tape-Laufwerk ablegen.
./prototyp.sh onlinebackup	Bei Veränderungen einer Datei die geänderten Teile online in verteilter Form ablegen. Neu erstellte Files vollständig online in verteilter Form ablegen.
./prototyp.sh restore	Rücksicherung unter Verwendungen von Tape- und Onlinedaten durchführen.
./prototyp.sh get_filelist	Dateiliste der online abgelegten Dateien erstellen.
./prototyp.sh download_data	Alle online abgelegte Daten herunterladen.
./prototyp.sh reconstruct_data	Dateien durch Zusammenfügen der einzelnen Fragmente rekonstruieren.

Mit einem Aufruf unter Verwendung des Parameters „set_passwort“ wird aus einem eingegebenen Master-Passwort des Benutzers ein Hashwert berechnet und in Form einer Datei abgelegt. Dieser Hashwert wird später zur Verschlüsselung der Daten verwendet. Das tatsächliche Passwort muss auf diese Weise nicht direkt abgelegt werden.

Ein Aufruf unter Verwendung des Parameters „fullbackup_nas_tape“ ermöglicht es, eine Vollsicherung der Daten in den Pfaden abzulegen, welche für NAS-System und Tape-Laufwerk vorgesehen sind. Zum Zeitpunkt der Sicherung wird im Hintergrund eine Signaturdatei (siehe Kapitel 4.2.2) erstellt, welche den Zustand einer Datei in Form von pro Block berechneten Prüfsummen beinhaltet. Auf diese Weise können später Dateiunterschiede berechnet werden, ohne direkt auf die Vollsicherung zugreifen zu müssen. Die Vollsicherung kann somit beispielsweise extern abgelegt werden.

Ruft man das Programm „prototyp.sh“ mit dem Parameter „onlinebackup“ auf, so wird jenes Verzeichnis permanent überwacht, das für die Ablage der Userdaten vorgesehen ist. Wird eine Veränderung festgestellt, so wird der Unterschied der Datei im Vergleich zur letzten Vollsicherung berechnet. Ist die Datei noch in keiner Vollsicherung vorhanden, so wird einmalig das komplette File hochgeladen. Vor dem Upload erfolgen Komprimierung, Verschlüsselung und die Aufteilung in Teildateien.

Mit dem Parameter „restore“ lassen sich die Daten wiederherstellen. Es werden dazu sowohl die Dateifragmente aus den Online-Speichern als auch eventuell in Vollsicherungen vorhandene Daten in ein temporäres Download-Verzeichnis geladen. Die Aufrufe mit den Parametern „get_filelist“, „download_data“ und „reconstruct_data“ werden im Zuge der Rücksicherung genutzt. Mit „get_filelist“ wird eine Dateiliste erstellt, welche die online abgelegten Daten beinhaltet. Auf dieser Basis lässt sich mit „download_data“ eine Datei nach der anderen herunterladen. Der Aufruf mit „reconstruct_data“ versucht, die geladenen Daten in der richtigen Reihenfolge zusammenzusetzen, sie zu entschlüsseln und zu dekomprimieren. Handelt es sich bei den entpackten Daten lediglich um abgespeicherte Dateiunterschiede, so wird diese mit der passenden Vorgängerversionen verschmolzen.

Die Sourcecodes der Dateien „prototyp.sh“ sowie „amazon_s3_glacier.sh“, „backblaze.sh“ und „google_drive.sh“ befinden sich in den Anhängen B bis E.

5. Validierung der Funktionen des Prototyps

Der gewünschte Funktionsumfang, welcher in Kapitel 4.1.1 beschrieben wurde, konnte im implementierten Prototyp vollständig berücksichtigt werden. Der Prototyp ist demnach in der Lage, Änderungen von Dateien zu erkennen. Es wird für jede Datei ein Zeitstempel ausgelesen, der Informationen darüber enthält, zu welchem Zeitpunkt Inhalt, Dateiname oder Berechtigung zum letzten Mal geändert wurden. Hat sich dieser Zeitpunkt im Vergleich zum Durchlauf davor verändert, so wird eine neue Sicherung angestoßen.

Im nächsten Schritt wird überprüft, ob für die Datei bereits eine ältere Sicherung existiert. Wenn ja, dann werden die Änderungen im Vergleich zum vorhandenen Sicherungsstand ermittelt. Eine Signaturdatei erlaubt es, die Unterschiede auch dann zu ermitteln, wenn die Originaldatei nicht für einen Abgleich zur Verfügung steht. Existiert keine Sicherung der Datei, so wird diese vollständig gesichert.

Um die zu transferierende Datenmenge zu reduzieren, erfolgt zusätzlich eine Komprimierung der Daten.

Um in Dateinamen und Ordnerstrukturen enthaltene Informationen zu verbergen, werden auf den Cloud Storage Services flache Strukturen mit kryptischen Dateinamen gebildet. Dateiname und -pfad der Originaldaten werden vor dem Upload in das bei der Komprimierung erstellte Datenarchiv aufgenommen. Diese Informationen stehen somit im Rahmen der Rücksicherung wieder zur Verfügung.

Ehe der zu sichernde Inhalt in drei Datenfragmente aufgeteilt wird, erfolgt eine Verschlüsselung der Daten.

Mittels REST-APIs werden die Dateien danach auf den einzelnen Online-Speichern platziert. Neben der Funktion für den Upload von Daten stehen auch Funktionen für den Download und für das Löschen von einzelnen Elementen zur Verfügung. Der Download der Datenfragmente ist zur Wiederherstellung notwendig. Das Löschen online abgelegter Daten kann dann notwendig sein, wenn die online abgelegten Daten ohnehin in neueren Sicherungen enthalten sind. Beispielsweise kann das bei einer Vollsicherung der Fall sein, die zuvor abgelegte Onlinesicherungen entbehrlich macht.

Zur Validierung der Ergebnisse sei an dieser Stelle auf die zwei in Kapitel 3.5 beschriebenen Anwendungsfälle Bezug genommen. Da das Verhältnis von vorhandener Internet-Bandbreite zum Volumen der Vollsicherung bei verschiedenen Unternehmen stark variieren kann, werden zwei Unternehmenstypen unterschieden:

- **Unternehmenstyp 1:** Hohe Internet-Bandbreite im Vergleich zum Volumen der zu sichernden Daten. Die gesamten Sicherungsdaten können in der Cloud abgelegt und ausreichend schnell wiederhergestellt werden.
- **Unternehmenstyp 2:** Geringe Internet-Bandbreite im Vergleich zum Volumen der zu sichernden Daten. Der Großteil der Sicherungsdaten kann nicht in der Cloud abgelegt werden. Ein zusätzliches Medium für die externe Lagerung von Sicherungsdaten ist notwendig. In der Cloud werden lediglich die Datenänderungen aus dem Zeitraum zwischen zwei Vollsicherungen abgelegt.

Jedes Unternehmen mit Zugang zum Internet lässt sich einem der beiden Fälle zuordnen. In Österreich sind das laut Statistik Austria bei Unternehmen mit mindestens 10 Beschäftigten mehr als 99 Prozent [St18b].

Im Folgenden soll für beide Fälle sowohl eine Datensicherung als auch eine Datenrücksicherung unter Verwendung des entwickelten Prototyps protokolliert werden.

5.1 Unternehmenstyp 1

Unternehmenstyp 1 nutzt die Cloud zur vollständigen Ablage seiner Sicherungsdaten. Die zu sichernden Daten entsprechen einem gemischten Datenbestand, wie er in Abbildung 15 dargestellt ist.

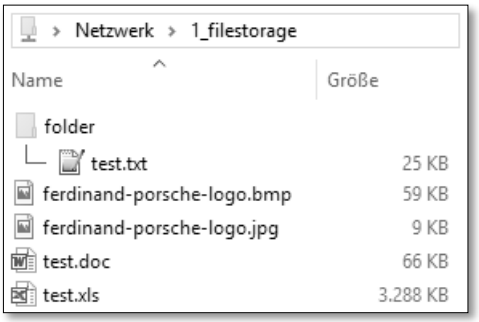

Abbildung 15: Datenbestand

Das Unternehmen 1 kann für die Sicherung der Daten die Funktion „onlinebackup“ nutzen. Der Prototyp erkennt bei der erstmaligen Ausführung, dass für jede Datei eine vollständige Übertragung notwendig ist. Eine beispielhafte Ausgabe des Prototyps für diesen Sachverhalt ist in Abbildung 16 dargestellt.

```
04-04-2019 16:39:45  -  File /srv/1_filestorage/test.xls verarbeiten
04-04-2019 16:39:45  -  Vollstaendige Datei muss geladen werden
04-04-2019 16:39:45  -  Komprimierung: 3366400 Byte zu 1561232 Byte
04-04-2019 16:39:46  -  Upload PART00 nach ...
04-04-2019 16:39:46  -  Google Drive
04-04-2019 16:39:48  -  Amazon S3 Glacier
04-04-2019 16:39:49  -  Upload PART01 nach ...
04-04-2019 16:39:49  -  Amazon S3 Glacier
04-04-2019 16:39:49  -  Backblaze
04-04-2019 16:39:53  -  Upload PART02 nach ...
04-04-2019 16:39:53  -  Backblaze
04-04-2019 16:39:58  -  Google Drive
04-04-2019 16:40:00  -  Upload erfolgreich!
```

Abbildung 16: Ausgabe des Prototyps bei initialer Onlinesicherung

Nach einer ersten vollständigen Sicherung der Daten werden bei jeder späteren Veränderung der Dateien lediglich die Blöcke übertragen, die auch verändert wurden. Eine beispielhafte Ausgabe dazu enthält Abbildung 17.

```
04-04-2019 16:46:22  -  File /srv/1_filestorage/test.xls verarbeiten
04-04-2019 16:46:22  -  Aktuelle Dateigroesse 3366400 Byte
04-04-2019 16:46:22  -  Dateiunterschied wird geladen
04-04-2019 16:46:22  -  Komprimierung: 12343 Byte zu 7163 Byte
04-04-2019 16:46:22  -  Upload PART00 nach ...
04-04-2019 16:46:22  -  Google Drive
04-04-2019 16:46:26  -  Amazon S3 Glacier
04-04-2019 16:46:26  -  Upload PART01 nach ...
04-04-2019 16:46:26  -  Amazon S3 Glacier
04-04-2019 16:46:26  -  Backblaze
04-04-2019 16:46:29  -  Upload PART02 nach ...
04-04-2019 16:46:29  -  Backblaze
04-04-2019 16:46:32  -  Google Drive
04-04-2019 16:46:34  -  Upload erfolgreich!
```

Abbildung 17: Ausgabe des Prototyps bei Veränderung einer Datei

Anstatt einer Dateigröße von rund 3,4 Megabyte wird eine Differenzdatei errechnet, welche lediglich 12,3 Kilobyte groß ist. Die Komprimierung der Daten führt dazu, dass überhaupt nur 7,2 Kilobyte an Daten an die Online-Speicher übertragen werden müssen.

Der Online-Speicher Amazon S3 enthält zu diesem Zeitpunkt die in Abbildung 18 dargestellten Datenfragmente.

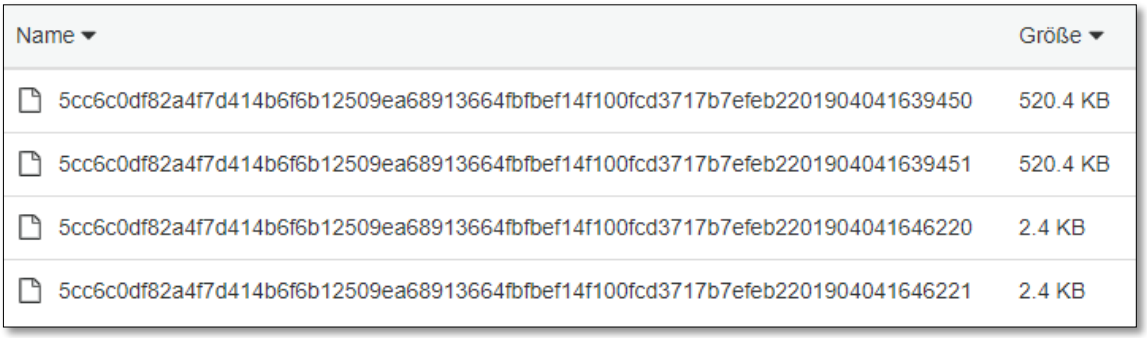

Abbildung 18: Beispielhafter Datenbestand von Amazon S3

Die ersten beiden Einträge entsprechen zwei von drei Teilen des Initial-Uploads. Die unteren beiden Zeilen stammen vom Upload der Dateiänderung.

Der Restore erfolgt mittels der Funktion „restore“. Sowohl für die Ablage der heruntergeladenen Daten als auch für die Ablage der rekonstruierten Daten werden

Unterverzeichnisse des Metadaten-Verzeichnisses verwendet. Diese können in der Programmdatei „prototyp.sh“ beliebig verändert werden. Siehe dazu auch Anhang B.

Abbildung 19 beinhaltet eine beispielhafte Ausgabe einer Datenrücksicherung. Es werden zuerst alle Daten von den Online-Speichern heruntergeladen, ehe sie in der richtigen Reihenfolge zusammengesetzt und entschlüsselt beziehungsweise dekomprimiert werden. Existiert die Zieldatei bereits, so wird jedes weitere Datenfragment mit der bestehenden Datei verschmolzen.

```
user@prototyp:/srv/4_script$ ./prototyp.sh restore
04-04-2019 17:35:02  -  Dateiliste erfolgreich erstellt!
04-04-2019 17:35:02  -  Lade 5cc6c0df82a4f7d414b6f6b12509ea68913664fb
fbef14f100fcd3717b7efeb220190404163945 ...
04-04-2019 17:35:02  -  Download von Google Drive
04-04-2019 17:35:03  -  Download von Amazon S3 Glacier
04-04-2019 17:35:04  -  Download von Backblaze
04-04-2019 17:35:06  -  Lade 5cc6c0df82a4f7d414b6f6b12509ea68913664fb
fbef14f100fcd3717b7efeb220190404164622 ...
04-04-2019 17:35:06  -  Download von Google Drive
04-04-2019 17:35:07  -  Download von Amazon S3 Glacier
04-04-2019 17:35:07  -  Download von Backblaze
04-04-2019 17:35:09  -  Rekonstruiere 5cc6c0df82a4f7d414b6f6b12509ea6
8913664fbfbef14f100fcd3717b7efeb220190404163945
04-04-2019 17:35:10  -  /srv/4_script/metadata/restore/srv/1_filestor
age/test.xls erstellt
04-04-2019 17:35:10  -  Rekonstruiere 5cc6c0df82a4f7d414b6f6b12509ea6
8913664fbfbef14f100fcd3717b7efeb220190404164622
04-04-2019 17:35:11  -  Verschmelzen ...
04-04-2019 17:35:11  -  /srv/4_script/metadata/restore/srv/1_filestor
age/test.xls aktualisiert
```

Abbildung 19: Ausgabe des Prototyps bei Rücksicherung einer Datei

Als Ergebnis ergibt sich im Verzeichnis für die Rücksicherung ein Datenbestand, der dem in Abbildung 15 exakt entspricht. Der Datenverlust ist somit gleich null.

5.2 Unternehmenstyp 2

Unternehmenstyp 2 nutzt die Cloud ergänzend zu einer externen Ablage der Vollsicherung auf einem Medium ungleich der Cloud. Um der 3-2-1-Regel zu genügen, soll die Vollsicherung, wie in Kapitel 3.6 dargestellt, sowohl auf einem unternehmensinternen NAS-System als auch auf einem externen Magnetband abgelegt

werden. Da die externe Sicherungskopie manuell außer Haus abgelegt werden muss, ist nur eine begrenzte Backup-Häufigkeit möglich. Um die Zwischenzeit zu überbrücken, sollten zwischen den Sicherungsläufen die Änderungen der Daten in der Cloud verteilt abgelegt werden.

Für die Vollsicherung kann die Funktion „fullbackup_nas_tape“ des Prototyps verwendet werden. Es werden dabei vollständige Sicherungskopien der Daten in jenen beiden Verzeichnissen abgelegt, welche als Mountpoints für NAS-System und Tape-Laufwerk definiert wurden. Um die Cloud Storage Services nicht unnötig zu belasten, wird zeitgleich für jede zu sichernde Datei die Löschung aller korrespondierenden Datenfragmente auf den Online-Speichern vorgenommen. Eine beispielhafte Ausgabe dazu beinhaltet Abbildung 20.

```
user@prototyp:/srv/4_script$ ./prototyp.sh fullbackup_nas_tape
04-04-2019 18:10:13  -  Dateiliste erfolgreich erstellt!
04-04-2019 18:10:13  -  Sichere /srv/1_filestorage/test.xls ...
04-04-2019 18:10:13  -  Vollstaendige Datei muss geladen werden
04-04-2019 18:10:13  -  File /srv/1_filestorage/test.xls
04-04-2019 18:10:14  -  Komprimierung: 3366400 Byte zu 1561227 Byte
04-04-2019 18:10:15  -  File online vorhanden 5cc6c0df82a4f7d414b6f6b
12509ea68913664fbfbef14f100fcd3717b7efeb220190404163945
04-04-2019 18:10:15  -  Loesche PART00 ...
04-04-2019 18:10:15  -  Google Drive
04-04-2019 18:10:16  -  Amazon S3 Glacier
04-04-2019 18:10:17  -  Loesche PART01 ...
04-04-2019 18:10:17  -  Amazon S3 Glacier
04-04-2019 18:10:18  -  Backblaze
04-04-2019 18:10:20  -  Loesche PART02 ...
04-04-2019 18:10:20  -  Backblaze
04-04-2019 18:10:22  -  Google Drive
04-04-2019 18:10:23  -  File online vorhanden 5cc6c0df82a4f7d414b6f6b
12509ea68913664fbfbef14f100fcd3717b7efeb220190404164622
04-04-2019 18:10:23  -  Loesche PART00 ...
04-04-2019 18:10:23  -  Google Drive
04-04-2019 18:10:25  -  Amazon S3 Glacier
04-04-2019 18:10:25  -  Loesche PART01 ...
04-04-2019 18:10:25  -  Amazon S3 Glacier
04-04-2019 18:10:25  -  Backblaze
04-04-2019 18:10:27  -  Loesche PART02 ...
04-04-2019 18:10:27  -  Backblaze
04-04-2019 18:10:29  -  Google Drive
04-04-2019 18:10:30  -  Synchronisation erfolgreich beendet.
```

Abbildung 20: Ausgabe des Prototyps bei Durchführung einer Vollsicherung

Um sicherzugehen, dass kein Zugriff auf die Daten in der Vollsicherung bestehen muss, um die Dateiänderungen für die Online-Ablage zu berechnen, wird an dieser Stelle die Sicherung im NAS-Verzeichnis manuell gelöscht und das Verzeichnis für die Magnetbandsicherung umbenannt. Danach wird der Prototyp mit dem Parameter „onlinebackup“ gestartet. Ab diesem Zeitpunkt werden alle Datenänderungen online abgelegt. Nach Änderung einer Datei enthält jeder Online-Speicher nicht wie in Anwendungsfall 1 gezeigt vier zu dieser Datei gehörenden Datenfragmente, sondern nur zwei.

Für die Rücksicherung kann ebenso wie im vorherigen Fall die Funktion „restore“ verwendet werden. Vor dem Ausführen wird das aus Testzwecken das zuvor umbenannte Verzeichnis für die Tape-Sicherung wieder mit dem originalen Namen versehen. Im Zuge der Rücksicherung werden zuerst die Onlineinhalte geladen. Danach erfolgt das Kopieren der Daten aus der Tape-Sicherung. Wie in Abbildung 21 ersichtlich erfolgt anschließend wieder eine chronologische Abarbeitung der einzelnen Datenfragmente und es wird daraus entweder eine Datei im Rücksicherungsverzeichnis erstellt oder eine bestehende Datei mit den neueren Dateiinhalten versorgt.

```
user@prototyp:/srv/4_script$ ./prototyp.sh restore
04-04-2019 18:59:15  -  Dateiliste erfolgreich erstellt!
04-04-2019 18:59:15  -  Lade 5cc6c0df82a4f7d414b6f6b12509ea68913664fb
fbef14f100fcd3717b7efeb220190404185748 ...
04-04-2019 18:59:15  -  Download von Google Drive
04-04-2019 18:59:16  -  Download von Amazon S3 Glacier
04-04-2019 18:59:17  -  Download von Backblaze
04-04-2019 18:59:19  -  Lade Tape-Sicherung ...
04-04-2019 18:59:19  -  Rekonstruiere 5cc6c0df82a4f7d414b6f6b12509ea6
8913664fbfbef14f100fcd3717b7efeb220190404185617
04-04-2019 18:59:20  -  /srv/4_script/metadata/restore/srv/1_filestor
age/test.xls erstellt
04-04-2019 18:59:20  -  Rekonstruiere 5cc6c0df82a4f7d414b6f6b12509ea6
8913664fbfbef14f100fcd3717b7efeb220190404185748
04-04-2019 18:59:20  -  Verschmelzen ...
04-04-2019 18:59:20  -  /srv/4_script/metadata/restore/srv/1_filestor
age/test.xls aktualisiert
```

Abbildung 21: Ausgabe des Prototyps bei Rücksicherung aus Band- und Onlinesicherung

Im Verzeichnis für die Rücksicherung befinden sich nach Abschluss der Rücksicherung wieder exakt die gleichen Daten wie in Abbildung 15. Der Datenverlust ist wiederum gleich null.

Die Größe der Dateisignaturen war für die getesteten Datenbestände in etwa um den Faktor 120 kleiner als die Größe der Nutzdaten.

6. Diskussion der Ergebnisse

6.1 Notwendigkeit von Datensicherungen

Der zunehmende Einfluss der Informationstechnologie auf sämtliche Geschäftsprozesse von Unternehmen erhöht deren Ertragspotential. Gleichzeitig steigert diese Entwicklung jedoch auch die Verwundbarkeit. Unternehmen etablieren deshalb Managementprozesse, welche durch Planung präventiver Maßnahmen versuchen, die Verfügbarkeit geschäftskritischer Prozesse ausreichend hoch zu halten. Kommt es trotz der getätigten Maßnahmen zu Ausfällen, so sollten im Vorhinein ausgearbeitete Notfallpläne eine rasche und koordinierte Wiederherstellung sicherstellen (siehe BCM in Kapitel 3.1). Bei der Wiederherstellung gilt es, die pro Geschäftsprozess definierten Zielwerte für RTO und RPO einzuhalten (siehe Kapitel 3.1). Die beiden Werte beschreiben, wie lange ein Prozess ausfallen darf beziehungsweise welcher Datenverlust in Kauf genommen werden kann, um die negativen wirtschaftlichen Folgen in einem vertretbaren Rahmen zu halten.

In vielen Unternehmen existieren mittlerweile Geschäftsprozesse, welche de facto ausfallfrei und jedenfalls ohne Datenverlust funktionieren müssen. Beispiele sind ERP-Systeme, ohne die keine Produktion möglich ist oder Webshops, die in der Zeit des Ausfalls keinen Umsatz generieren. Die technische Antwort auf Anforderungen dieser Art sind hochverfügbare Clustersysteme. Solche Systeme bestehen aus mehreren eigenständigen Serverknoten, welche in verschiedenen Rechenräumen betrieben werden. Fällt aufgrund von Hard- oder Softwarefehlern ein Ressourcenknoten aus, so können kritische Dienste automatisch von anderen Serverknoten übernommen werden. In vielen Fällen ist es bereits möglich, diese Übernahme durchzuführen, ohne die aktiven Verbindungen der Nutzerinnen und Nutzer zu unterbrechen.

Durch Hochverfügbarkeitslösungen (siehe Kapitel 3.2) lässt sich die Verfügbarkeit geschäftskritischer Systeme somit wesentlich erhöhen. Trotzdem existieren Bedrohungen, deren Auswirkungen im schlimmsten Fall die Existenz von Unternehmen bedrohen. Während Ausfälle von einzelnen Hardwarekomponenten bis hin zu

kompletten Serverknoten sehr gut kompensiert werden können, helfen Redundanzen bei folgenden Fehlern nicht mehr:

- Mehrfachfehler wie beispielsweise gleichzeitiger Ausfall von mehreren Serverknoten
- Fehler in der Cluster-Software
- Datenkorruption und Datenverluste aller Art (verursacht durch administrative Fehler, Ransomware, Feuer, Erdbeben oder Hochwasser)

Diese Fehler können nur bewältigt werden, indem regelmäßig ein Datenbestand in einer getrennten Umgebung abgelegt wird, der nicht innerhalb von Sekundenbruchteilen überschrieben wird. Weitreichend akzeptierten Regeln zu Folge sollten insgesamt zumindest drei Kopien der Daten zur Verfügung stehen. Um die Wahrscheinlichkeit für einen gleichzeitigen Ausfall von zwei Sicherungsständen zu minimieren, sollten zwei unterschiedliche Medien verwendet werden (siehe 3-2-1 Regel von Peter Krogh in Kapitel 3.4).

6.2 Zeitgemäße Sicherungsmedien

Aus technischer Sicht sind alle Arten von Datenträgern als Sicherungsmedien geeignet. Im einfachsten Fall kann es ausreichen, den vorhandenen Datenbestand in einer bestimmten Regelmäßigkeit manuell auf CD-ROM, DVD oder Blu-ray zu brennen. Auch USB-Sticks, Flash Speicherkarten oder externe USB-Festplatten können verwendet werden. Je kürzer allerdings die geforderten Zeitabstände sind, in denen Sicherungen anzufertigen sind, desto mehr Arbeits- und Zeitaufwand entsteht durch das manuelle Kopieren der Daten. Deshalb ist es ab einer bestimmten Sicherungshäufigkeit sinnvoll, geeignete Sicherungssoftwareprodukte einzusetzen. Auf dem Markt erhältliche Sicherungssoftware ist auch für komplexe Sicherungsaufgaben geeignet. Es können beispielsweise direkt Datenbanken gesichert werden, ohne einen manuellen Eingriff zu erfordern. Darüber hinaus ist eine Kompatibilität mit einer großen Auswahl an Sicherungsmedien gegeben. Übliche Ziele für Datensicherungen mit professionellen Sicherungssoftwareprodukten sind Network Attached Storages (NAS) oder Storage Area Network-Systeme (SAN). Für die externe Ablage von Sicherungsdaten erfreuen

sich neben Magnetbändern auch Cloud Storage Services immer größerer Beliebtheit. Im Gegensatz zu Sicherungen auf Magnetband ist bei Verwendung von Cloud Services kein manueller Transport von Sicherungsdatenträgern erforderlich. Cloud Storage Services haben nicht nur das Potential, Magnetbänder für die Ablage von externen Sicherungen abzulösen. Durch die permanente Verfügbarkeit erlauben Sie es in bestimmten Fällen auch, die oftmals in kürzeren Zeitabständen angefertigten und vor Ort abgelegten primären Sicherungen zu übernehmen.

Neben einer hervorragenden Skalierbarkeit wird durch die Nutzung von Cloud Services auch die organisatorische Flexibilität gesteigert, da Ressourcen ohne nennenswerte Investitionen und ohne größerer Vorlaufzeit an den jeweiligen Bedarf angepasst werden können. Teilweise ungenutzte Kapazitäten für das Abdecken von Lastspitzen können vermieden werden und das Technologierisiko wird an die Cloud Storage Provider abgewälzt. Es ergeben sich somit einige Vorteile, die schlussendlich Unternehmen dabei helfen können, Kosten einzusparen. Neben einer effizienten Nutzung der vorhandenen Internet-Bandbreite ist allerdings speziell im Zusammenhang mit geschäftskritischen Daten auch ein großes Augenmerk auf die Seriosität und Sicherheit der Cloud Service Provider zu legen.

6.3 Sichere Ablage von Daten in der Cloud

Beinhalten auf Cloud Storage Services abgelegte Daten Geschäftsgeheimnisse oder hängen ganze Existenzen von Unternehmen im Extremfall von der Verfügbarkeit der gesicherten Daten ab, so empfiehlt es sich, folgende Gefahren und Risiken bei der Ablage von Daten in der Cloud genau unter die Lupe zu nehmen, um sie für den konkreten Anwendungsfall zu bewerten:

- Verstoß gegen Datenschutzbestimmungen
- Verlust der Vertraulichkeit der Daten, Verlust der Integrität
- Dienstausfall, Stilllegung von Cloud Storage Services, Datenverlust
- Unsichere Client-Software

Während eine datenschutzkonforme Ablage von Daten praktisch bei jedem der gegenwärtig bekannten Cloud Storage Provider möglich ist (siehe Kapitel 2.3.4), existieren im Zusammenhang mit der Wahrung von Vertraulichkeit und Integrität erhebliche Einschränkungen. Durch eine Verschlüsselung der Daten noch vor der Übergabe an den Cloud Storage Provider wäre ein guter Schutz der Daten gewährleistet. Diese sogenannte End-to-End Verschlüsselung wird aber aktuell nur in seltenen Fällen angebotenen. Viel häufiger werden die Daten ausschließlich auf dem Transportweg verschlüsselt. Dies erlaubt es jedoch sowohl dem Cloud Storage Provider als auch potentiellen Angreifern, die abgelegten Daten direkt einzusehen.

Ein Ausfall oder eine Stilllegung eines Cloud Storage Services kann dazu führen, dass die Verfügbarkeit der Daten temporär oder permanent eingeschränkt ist. Die Verfügbarkeit kann nur dann aufrecht erhalten werden, wenn die Daten zuvor bei einem zweiten Provider abgelegt wurden.

Um sich gegen Programmierfehler oder Backdoors bei der Client-Software zu schützen, empfiehlt es sich, bei kritischen Anwendungen nicht auf die vom Service Provider bereitgestellte Client-Software zurückzugreifen.

Um alle Risiken zu minimieren, müssen die Daten somit auf mehreren Providern abgelegt werden, ohne bereitgestellte Client-Software zu nutzen. Up- und Download der Daten sind in Folge per API durchzuführen und die Verschlüsselung ist im Sinne einer End-to-End Verschlüsselung vorzunehmen.

Wird zu diesem Zeitpunkt die in Summe doppelte Speicherung der Daten in der Cloud als gegeben angenommen, so lässt sich das Schutzniveau der Daten mit einer weiteren Maßnahme noch einmal maßgeblich erhöhen, ohne andere Aspekte negativ zu beeinflussen oder zusätzliche Kosten zu generieren. Werden nämlich die Dateien dreigeteilt und in Form von jeweils zwei Teilen bei drei Cloud Storage Providern abgelegt, so ist plötzlich kein Provider mehr im Besitz der vollständigen Daten (siehe Kapitel 2.3.4). Für einen einzelnen Provider oder für einen potentiellen Angreifer ist es in Folge nahezu unmöglich, die Daten einzusehen, auch wenn diese vor längerer Zeit abgelegt wurden und Schwachstellen an verwendeten Verschlüsselungsalgorithmen

entdeckt werden. Der benötigte Online-Speicherplatz bleibt in Summe unverändert und entspricht dem doppelten des Sicherungsvolumens.

Da auf dem Markt kein Softwareprodukt existiert, welches in diesem Zusammenhang genutzt werden kann, wurde im Rahmen dieser Arbeit ein Prototyp entwickelt, der sämtliche Risiken wie beschrieben minimiert, ohne die Vorteile bei Ablage von Daten in der Cloud negativ zu beeinflussen.

Unter Berücksichtigung der beschriebenen Maßnahmen kann prinzipiell jeder Cloud Storage Service für die sichere Ablage von Sicherungsdaten zum Einsatz kommen. Die einzigen Voraussetzungen sind, dass APIs zur Verfügung stehen müssen und dass keine Abhängigkeiten zu anderen verwendeten Diensten existieren. Beispielsweise sollte der Service Dropbox [Dr18] nicht gemeinsam mit Amazon S3 [Am19e] zum Einsatz kommen. Der eine Service baut auf dem anderen auf.

6.4 Begrenzte Ressource „Internet“

Um Sicherungsdaten nicht nur sicher sondern auch sinnvoll in der Cloud abzulegen, gilt es, auch die vorhandene Internet-Bandbreite zu beachten. Nahezu jedes Unternehmen in Österreich verfügt mittlerweile über einen Zugang zum Internet [St18b]. Manche davon beziehen sehr hohe Bandbreiten. Bei wiederum anderen Unternehmen kann es sein, dass die Internet-Bandbreite im Verhältnis zur Datenmenge, welche im Desaster-Fall geladen werden muss, als niedrig zu bewerten ist. Trotz Komprimierung und Reduzierung der Datenmenge durch Berechnung und ausschließlichen Transfer von geänderten Blöcken (siehe Kapitel 4.2.2) kann es sich ergeben, dass die Rücksicherung nicht innerhalb einer geforderten Zeit (RTO siehe Kapitel 3.2) abgeschlossen werden kann.

Entsprechend ergeben sich zwei Typen von Unternehmen (siehe Kapitel 3.5):

- **Unternehmenstyp 1** steht für ein Unternehmen, bei dem die Internet-Bandbreite dafür ausreicht, das gesamte Datenvolumen im Fall einer Rücksicherung ausreichend schnell aus der Cloud zu laden. Da der Cloud Speicher permanent beschrieben werden kann, können Sicherungsintervalle von wenigen Minuten

erreicht werden. Je kürzer das Sicherungsintervall ist, desto näher liegt der Datenverlust bei Rücksicherung im Desaster-Fall bei der Marke von null Minuten.

- **Unternehmenstyp 2** beschreibt ein Unternehmen, welches die gesamten Sicherungsdaten aufgrund der verfügbaren Internet-Bandbreite nicht in der Cloud ablegen kann. Cloud Storage Services können bei diesen Unternehmen allerdings dazu genutzt werden, Datenänderungen, welche in der Zeit zwischen der Anfertigung von extern abgelegten (Magnetband-)Vollsicherungen generiert werden, online abzulegen. Der Datenverlust im Desaster-Fall kann dadurch ebenfalls in der Nähe der Marke von null Minuten gehalten werden. Im Unterschied zu Unternehmenstyp 1 müssen in diesem Fall aber Netzwerkspeicher und Bandlaufwerke angeschafft und betrieben werden. Außerdem wird für den Transport der Magnetbänder Arbeitszeit in Anspruch genommen.

6.5 Bewertung des Prototyps

Der im Rahmen dieser Arbeit entwickelte Prototyp ist nicht nur die Antwort auf die eingangs dargestellte Forschungsfrage:

„Welche Funktionen muss eine Anwendung beinhalten, um in der Lage zu sein, Änderungen von Dateien in definierten Verzeichnissen zu erkennen, den Unterschied zu einer Ursprungsversion zu berechnen und diesen in verschlüsselter Form so auf Cloudspeichern abzulegen, dass kein Provider im Vollbesitz der Daten ist und der Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht einschränkt?“

Er beinhaltet auch alle Funktionalitäten, die für die Datensicherung der beiden im vorigen Abschnitt beschriebenen Unternehmenstypen notwendig sind. Die Protokollierung einer erfolgreichen Sicherung sowie Rücksicherung ist in Kapitel 5 dargestellt. Es wird ersichtlich, dass zu keinem Zeitpunkt einer der verwendeten Provider im Besitz der vollständigen Daten ist. Eine Entschlüsselung der abgelegten Datenfragmente läuft selbst unter Kenntnis von Verschlüsselungsverfahren und Passphrase auf Fehler. Um die vorhandene Internet-Bandbreite effizient zu nutzen, wurden aus den abgelegten Dateien jene Teile errechnet, die auch tatsächlich verändert wurden. Lediglich diese

Teile der Dateien werden komprimiert und über das Internet übertragen. Ob eine Datei verändert wurde, lässt sich mittels Abfrage von speziellen Zeitstempeln performant feststellen (siehe Kapitel 4.2.1). Der Unterschied zu einer Ursprungsversion kann auf Basis eines blockweisen Abgleichs von Prüfsummen errechnet werden (siehe Kapitel 4.2.2). Durch die gewählte Form der Verteilung der Daten kann ein Provider ausfallen, ohne dadurch die Verfügbarkeit der Sicherungsdaten einzuschränken.

Im Rahmen dieser Arbeit wird somit eine Möglichkeit aufgezeigt, wie selbst kritische Daten sicher in der Cloud abgelegt werden können. Ein Vergleich der Kosten zeigt für ausgewählte Fälle, dass auch eine in Summe doppelte Ablage der Daten in der Cloud zu keiner Überschreitung der Kosten von On-Premises Lösungen führt (siehe Kapitel 3.6).

Wenn eine vorhandene Internet-Bandbreite für eine vollständige Auslagerung der Sicherungsdaten nicht ausreicht, kann die Cloud trotzdem eine sinnvolle Ergänzung zu einer manuell außer Haus abgelegten Datensicherung sein. Auch in diesem Fall ist es möglich, die veränderten Datenblöcke aus den abgelegten Dateien zu berechnen, um diese bis zur nächsten Vollsicherung in der Cloud abzulegen. Ein direkter Zugriff auf eine Vollsicherung ist dazu nicht notwendig.

Durch die Umsetzung des Prototyps in Form einer Shell-Anwendung wurde eine hohe Transparenz sichergestellt. Es ist einfach nachzuvollziehen, was in welchem Teilschritt passiert. Außerdem kann die Anwendung an gewünschten Stellen mit einfachen Mitteln angepasst werden. Die Einbindung von eigenen Programmen für die Bearbeitung einzelner Teilschritte ist mit einfachen Mitteln möglich.

In der Praxis ist vorstellbar, dass der entwickelte Prototyp zusätzlich zu einem bestehenden Sicherungssoftwareprodukt zum Einsatz kommt. Von der Sicherungssoftware abgelegte Datensicherungen können in einem zweiten Schritt in die Cloud synchronisiert werden. Als noch sinnvoller ist eine Integration der gezeigten Funktionalitäten in bestehende Sicherungssoftwareprodukte zu erachten.

7. Fazit und Ausblick

Diese Arbeit beschäftigt sich mit Cloud Speicherdiensten und deren Verwendung innerhalb von Backup-Konzepten. Im ersten Abschnitt wurde die Problemstellung und Zielsetzung definiert und der Aufbau der Arbeit festgelegt. Der nachfolgende Abschnitt beschäftigte sich mit den Grundlagen des Cloud Computing. Darauf aufbauend wurden die Cloud Storage Services als eine spezielle Form des Cloud Computing dargestellt. Auf den möglichen Nutzen, die Risiken sowie wichtige Sicherheitsaspekte wurde dabei näher eingegangen. Im Anschluss daran wurden die Grundlagen der Datensicherung aufgegriffen. Es wurde die Notwendigkeit für die Anfertigung von Datensicherungen erörtert und mögliche Sicherungsarten, -medien und -strategien dargestellt. Auf Basis der gewonnenen Erkenntnisse wurde im anschließenden Abschnitt ein konkretes Konzept entworfen, welches eine sichere Ablage von unternehmenskritischen Sicherungsdaten auf Cloud Speicherdiensten zulässt. Nach der Umsetzung des Konzepts in Form eines Prototyps wurde das Ergebnis, aufbauend auf im Rahmen der Arbeit konstruierten Anwendungsfällen, validiert.

Der entwickelte Prototyp deckt nicht nur die Anforderungen der konstruierten Anwendungsfälle vollständig ab, sondern ist auch gleichzeitig die Antwort auf die Forschungsfrage. Er ist in der Lage, Änderungen von Dateien in definierten Verzeichnissen zu erkennen und den Unterschied zu Vorgängerversionen zu berechnen. Zur Ermittlung dieses Unterschieds ist dabei kein direkter Zugriff auf die Vorgängerdatei erforderlich. Der Unterschied wird vielmehr auf Basis von Metadaten errechnet. Diese Tatsache erlaubt es, die vollständigen Vorgängerdateien außerhalb des primären Systems abzulegen. Im Rahmen von Datensicherungskonzepten kann dies einer extern vorgehaltenen Vollsicherung auf Magentband entsprechen. Die dargestellte Strategie zur verschlüsselten und verteilten Ablage von Daten in der Cloud stellt sicher, dass keiner der verwendeten Cloud Storage Provider im Besitz der vollständigen Daten ist. Für den Provider oder für potentielle Angreifer ist es selbst bei veralteten und unsicheren Authentifizierungs- und Verschlüsselungsalgorithmen nahezu unmöglich, die Daten einzusehen. Durch die gewählte Form der Verteilung der Daten ist außerdem sichergestellt, dass ein Ausfall eines einzelnen Providers die Verfügbarkeit der Daten nicht einschränkt.

Zukünftige Untersuchungen sollten die Performance des implementierten Prototyps betreffen. In der aktuellen Version wird beispielsweise für jedes Datenfragment eine Upload-Anfrage initiiert. Eine Zusammenlegung aller Datenfragmente aus einem Durchlauf zu einem einzigen Upload-Vorgang generiert das Potenzial, Geschwindigkeitsvorteile zu erzielen. Selbiges gilt für den Download und die Rekonstruktion der Daten. Während in der aktuellen Version sowohl Download als auch Rekonstruktion der Daten strikt nacheinander erfolgen, kann durch eine Parallelisierung die Rücksicherung von größeren Datenmengen möglicherweise beschleunigt werden. Vor einer ersten produktiven Verwendung der Anwendung ist diese ebenso um ein geeignetes Monitoring zu erweitern. Die schlussendlich ausgereifte Softwareanwendung kann ergänzend zu einer bestehenden Sicherungssoftware zum Einsatz kommen. Während die bestehende Sicherungssoftware einen Netzwerkspeicher mit Daten befüllt, könnte der Prototyp diese abgreifen und in die Cloud synchronisieren.

Eine weitere Untersuchung sollte die Frage behandeln, ob die beschriebene verteilte Speicherung in der Cloud nicht ohnehin einen Mehrwert für jede bestehende professionelle Sicherungssoftware darstellt. Eine vollständige Integration in bestehende Produkte ist anzudenken.

Außerdem wäre eine Diskussion darüber interessant, ob durch die Zerteilung der Daten noch personenbezogene Daten vorliegen, auch wenn die ursprünglichen Inhalte personenbezogen waren. Jede Datei wird zuerst komprimiert, dann verschlüsselt und erst dann zerteilt. Auch wenn man die verwendeten Verschlüsselungsverfahren und die zugehörigen Schlüssel kennt, laufen Entschlüsselung und Dekomprimierung von einzelnen Teildateien auf Fehler und liefern ein unbrauchbares Ergebnis.

Literaturverzeichnis

[Am19a] AMAZON WEB SERVICES, INC.: *AWS | Amazon Glacier – Online Backup Server in der Cloud*. URL https://aws.amazon.com/de/glacier/. - abgerufen am 2019-03-09

[Am19b] AMAZON WEB SERVICES, INC.: *Pricing FAQ @ Amazon.de*. URL https://www.amazon.de/b?ie=UTF8&node=14156437031. - abgerufen am 2019-03-09

[Am19c] AMAZON WEB SERVICES, INC.: *Erstellen einer REST-API als Amazon S3-Proxy in API Gateway - Amazon API Gateway*. URL https://docs.aws.amazon.com/de_de/apigateway/latest/developerguide/integrating-api-with-aws-services-s3.html. - abgerufen am 2019-04-02

[Am19d] AMAZON WEB SERVICES, INC.: *IAM Management Console*. URL https://console.aws.amazon.com/iam/home#/users. - abgerufen am 2019-04-02

[Am19e] AMAZON WEB SERVICES, INC.: *Amazon Simple Storage Service S3 – Cloud Online-Speicher*. URL https://aws.amazon.com/de/s3/. - abgerufen am 2019-02-12

[Am19f] AMAZON WEB SERVICES, INC.: *Amazon Web Services Simple Monthly Calculator*. URL https://calculator.s3.amazonaws.com/index.html. - abgerufen am 2019-03-08

[Ap19] APPLE INC.: *Speicherpläne und -preise iCloud*. URL https://support.apple.com/de-at/HT201238. - abgerufen am 2019-03-09

[Ba19a] BACKBLAZE: *Backblaze B2 Cloud Storage*. URL https://www.backblaze.com/b2/cloud-storage.html. - abgerufen am 2019-03-09

[Ba19b] BACKBLAZE: *Overview Backblaze API*. URL https://www.backblaze.com/b2/docs/. - abgerufen am 2019-04-02

[Ba18] BALDAUF, MARKUS: *IT Gehälter 2018 in Österreich*. URL https://www.mbmc.at/it-gehaelter-oesterreich/. - abgerufen am 2019-03-11

[Be19] *Beam your data - absolutely secure!* URL https://www.securebeam.com. - abgerufen am 2019-03-11

[Be13] BEDNER, MARK: *Cloud Computing: Technik, Sicherheit und rechtliche Gestaltung*, *Forum Wirtschaftsrecht*. Kassel : Kassel University Press, 2013 — ISBN 978-3-86219-080-5

[Bi15] BIHIS, CHARLES: *Mastering OAuth 2.0: Create powerful applications to interact with popular service providers such as Facebook, Google, Twitter, and more by leveraging the OAuth 2.0 Authorization Framework*. Birmingham : Pakt Publishing Ltd., 2015 — ISBN 978-1-78439-540-7

[Bl19] *Blu-ray Disc*. URL https://de.wikipedia.org/w/index.php?title=Blu-ray_Disc&oldid=186044691. - abgerufen am 2019-03-08

[Bo14] BOERSE, JAN-HENDRIK: *Unternehmen durch die Krise führen Business Continuity Management im Härtetest einer Pandemie*. Hamburg : Diplomica-Verlag, 2014 — ISBN 978-3-8428-7873-0

[Bo19] BOX, INC.: *Preisgestaltung von Box für Unternehmen und Einzelpersonen | Box Deutschland*. URL https://www.box.com/de-de/pricing. - abgerufen am 2019-03-09

[Bu19] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK, BSI: *Maßnahmenkatalog Organisation - IT-Grundschutz-Kataloge - M 2.164 Auswahl eines geeigneten kryptographischen Verfahrens*. URL https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02164.html. - abgerufen am 2019-03-18

[Bu12] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Überblickspapier Online-Speicher*. URL https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Download/Ueberblickspapier_Online-Speicher_pdf.pdf?__blob=publicationFile. - abgerufen am 2019-02-16

[Ca19] CANONICAL LTD.: *The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu*. URL https://www.ubuntu.com/. - abgerufen am 2019-03-11

[Cl19a] CLOUDWARDS.NET: *Best Cloud Storage Reviews 2019 – Top 50 Services Compared*. URL https://www.cloudwards.net/cloud-storage-reviews/. - abgerufen am 2019-02-13

[Cl19b] CLOUDFUZE, INC: *Easy File Transfers and Migration Between Cloud Storage Services*. URL https://www.cloudfuze.com/. - abgerufen am 2019-03-11

[Cl19c] CLOUDHQ LLC: *Gmail Productivity Tools | Sync, Migration and Back up - cloudHQ*. URL https://www.cloudhq.net/g_suite. - abgerufen am 2019-03-11

[Di15] DISARIO, DOMENIC PHILIP: *Backup Fanatic: How to Ensure Business Continuity by Delivering Continuous Protection, Secured Storage, Data Compliance, and Instant Data Recovery* : CreateSpace Independent Publishing Platform, 2015 — ISBN 978-1-5028-1695-5

[Di18] DISASTER RECOVERY JOURNAL: *Business Continuity Terms*. URL https://www.drj.com/downloads/drj_glossary.pdf. - abgerufen am 2018-12-02

[Dr18] DROPBOX INC.: *Dropbox*. URL https://www.dropbox.com/h. - abgerufen am 2018-06-24

[Dr19] DROPBOX INC.: *Choose the right Dropbox for you and your business*. URL https://www.dropbox.com/plans?trigger=nr. - abgerufen am 2019-03-09

[Eg15] EGGENBERGER, SANDRO: *Entwicklung des Cloud Computings in deutschschweizer KMUs*. Norderstedt : GRIN Publishing, 2015 — ISBN 978-3-668-01222-6

[Eu19] EUROPEAN UNION AGENCY FOR NETWORK AND INFORMATION SECURITY: *ENISA*. URL https://www.enisa.europa.eu/. - abgerufen am 2019-02-05

[Fe01] FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATIONS: Advanced Encryption Standard (AES). In: *Federal Information Processing Standards Publication* Bd. 197 (2001)

[Fi10] FISCHER, MARCUS: *Ubuntu GNU/Linux: Das umfassende Handbuch, aktuell zu Ubuntu 10.04 LTS »Lucid Lynx«*, *Galileo Computing*. 5. Auflage. Bonn : Galileo Press, 2010 — ISBN 978-3-8362-1654-8

[Fo18] FORBES MEDIA LLC: *Dropbox Is Doing Well, But Looks Rich In The Face Of Industry Headwinds*. URL https://www.forbes.com/sites/greatspeculations/2018/05/21/dropbox-is-doing-well-but-looks-rich-in-the-face-of-industry-headwinds/. - abgerufen am 2019-03-09

[Go19a] GOOGLE IRELAND LIMITED: *Google Drive*. URL https://www.google.com/drive/. - abgerufen am 2019-03-24

[Go19b] GOOGLE IRELAND LIMITED: *Google Drive – Mehr Speicherplatz und weitere Vorteile von Google*. URL https://one.google.com/about. - abgerufen am 2019-03-09

[Go19c] GOOGLE IRELAND LIMITED: *Cloud Storage – Preise | Cloud Storage | Google Cloud*. URL https://cloud.google.com/storage/pricing. - abgerufen am 2019-03-09

[Go19d] GOOGLE IRELAND LIMITED: *API Reference | Drive REST API v2*. URL https://developers.google.com/drive/api/v2/reference/. - abgerufen am 2019-04-02

[Go19e] GOOGLE IRELAND LIMITED: *APIs & Dienste – Google API Console*. URL https://console.developers.google.com/apis/dashboard. - abgerufen am 2019-04-02

[He15b] HEISE MEDIEN: *Bandspeicher: Erste LTO-7-Laufwerke für 6-TByte-Bänder*. URL https://www.heise.de/ix/meldung/Bandspeicher-Erste-LTO-7-Laufwerke-fuer-6-TByte-Baender-2840518.html. - abgerufen am 2019-04-10

[He03] HEROLD, HELMUT: *Linux-Unix-Shells: Bourne-Shell, Korn-Shell, C-Shell, bash, tcsh*, *Linux/Unix und seine Werkzeuge*. 3. Auflage. München : Addison-Wesley, 2003 — ISBN 978-3-8273-1511-3

[He15a] HESSISCHES MINISTERIUM FÜR WIRTSCHAFT, ENERGIE, VERKEHR UND LANDESENTWICKLUNG: *Vertraulichkeitsschutz durch Verschlüsselung*. URL https://wirtschaft.hessen.de/sites/default/files/media/hmwvl/leitfaden_vertraulichkeitsschutz_durch_verschluesselung.pdf. - abgerufen am 2018-06-24

[ID19] IDG TECH MEDIA GMBH: *Hardware-Trends 2019: Festplatten & SSDs*. URL https://www.pcwelt.de/ratgeber/Hardware-Trends-2019-Festplatten-SSDs-10085261.html. - abgerufen am 2019-04-10

[Ki17] KILIAN, TIM: *Datenkompression. Grundlagen und Kompressionsverfahren*. Norderstedt : GRIN Verlag, 2017 — ISBN 978-3-668-73274-2

[KSK11] KLETT, GERHARD ; SCHRÖDER, KLAUS-WERNER ; KERSTEN, HEINRICH: *IT-Notfallmanagement mit System: Notfälle bei der Informationsverarbeitung sicher beherrschen*, *Praxis*. 1. Auflage. Wiesbaden : Vieweg + Teubner, 2011 — ISBN 978-3-8348-1288-9

[Kr16] KRATOCHWILL - IT DIENSTLEISTUNGEN U. D. V.: *Datensicherung – Sicherungsmedien und Aufbewahrungsorte*. URL https://wdns.at/thema-datensicherung-sicherungsmedien-und-aufbewahrungsorte/. - abgerufen am 2019-03-08

[Kr09] KROGH, PETER: *The DAM book: digital asset management for photographers*. 2. Auflage. Sebastopol : O'Reilly, 2009 — ISBN 978-0-596-52357-2

[Le07] LEITNER, ACHIM: Der clevere Rsync-Algorithmus findet Gemeinsamkeiten in Dateien auf verschiedenen Rechnern. In: *Linux-Magazin* (2007), Nr. 08

[LC03] LITTLE, DAVID B. ; CHAPA, DAVID A.: *Implementing Backup and Recovery: The Readiness Guide for the Enterprise*, *Veritas series*. New York : Wiley, 2003 — ISBN 978-0-471-22714-4

[MG11] MELL, PETER ; GRANCE, TIMOTHY: *The NIST definition of cloud computing.* Gaithersburg : National Institute of Standards and Technology, 2011

[Me16] MEINHOLD, DANIEL: *Cloud Services: von IaaS zu Everything as a Service*. URL https://blog.orbit.de/2016/10/13/cloud-services-und-bereitstellungsmodelle/. - abgerufen am 2019-02-07

[Mi19a] MICROSOFT CORPORATION: *Microsoft OneDrive*. URL https://onedrive.live.com/about/de-at/. - abgerufen am 2019-03-24

[Mi19b] MICROSOFT CORPORATION: *Plans - Microsoft OneDrive*. URL https://onedrive.live.com/about/de-DE/plans/. - abgerufen am 2019-03-09

[Mi19c] MICROSOFT CORPORATION: *Preisübersicht – Azure-Preisgestaltung | Microsoft Azure*. URL https://azure.microsoft.com/de-de/pricing/. - abgerufen am 2019-03-09

[Mi18] MINOR, JENS: *Meilenstein: Als achtes Google-Produkt - Google Drive hat mehr als 1 Milliarde aktive Nutzer*. URL https://www.googlewatchblog.de/2018/07/meilenstein-google-drive-milliarde/. - abgerufen am 2019-03-09

[Mo14] MOHN, CHRISTIAN: *Learning Veeam Backup & Replication for VMware vSphere: Learn how to protect your data in your VMware vSphere infrastructure with Veeam Backup & Replication*. Birmingham, U.K : Packt Publishing, 2014 — ISBN 978-1-78217-418-9

[Mö13] MÖNKEMEIER, THOMAS: *Zeitstempel von Dateien – Linupedia*. URL http://linux-club.de/wiki/opensuse/Zeitstempel_von_Dateien#Welche_Zeit_wird_mit_welcher_Operation_neu_gesetzt. - abgerufen am 2019-03-17

[Mu11] MULAZZANI, MARTIN ; SCHRITTWIESER, SEBASTIAN ; LEITHNER, MANUEL ; HUBER, MARKUS ; WEIPPL, EDGAR: CLOUD SPEICHERDIENSTE ALS ANGRIFFSVEKTOREN. In: *Wien: 9th Information Security Konferenz in Krems* (2011)

[Mu19] MULTCLOUD: *Dateien in der Cloud mit MultCloud verwalten, verschieben, kopieren und migrieren*. URL https://www.multcloud.com/de/. - abgerufen am 2019-03-11

[MPR15] MÜNZL, GERALD ; PAULY, MICHAEL ; RETI, MARTIN: *Cloud Computing als neue Herausforderung für Management und IT*, *Essentials*. Berlin : Springer Vieweg, 2015 — ISBN 978-3-662-45831-0

[Na19] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *NIST*. URL https://www.nist.gov/. - abgerufen am 2019-02-05

[Ne19] *News for rsync 3.0.0*. URL https://download.samba.org/pub/rsync/src/rsync-3.0.0-NEWS. - abgerufen am 2019-03-18

[Od19] *Odrive - All cloud storage in one place*. URL https://www.odrive.com. - abgerufen am 2019-03-11

[Ot19] OTIXO, INC.: *WORK SMARTER. SHARE TOGETHER.* URL https://www.otixo.com/. - abgerufen am 2019-03-11

[PP12] PADHY, RABI PRASAD ; PATRA, MANAS RANJAN: Evolution of Cloud Computing and Enabling Technologies. In: *International Journal of Cloud Computing and Services Science (IJ-CLOSER)* Bd. 1 (2012), Nr. 4

[Pa13] PAUSTIAN, SASCHA: *Das Buch zum Wirtschaftsfachwirt IHK*. 1. Auflage. Heilbronn, Neckar : Lernstarter Bildungsmedien, 2013 — ISBN 978-3-9815428-6-8

[PC19] PCLOUD AG: *Die besten Cloud-Storage Preis- und Kostenpläne - pCloud*. URL https://www.pcloud.com/de/cloud-storage-pricing-plans.html. - abgerufen am 2019-03-09

[Ra19] RACKSPACE US, INC.: *Rackspace Public Cloud - Rechner*. URL https://www.rackspace.com/de-at/calculator. - abgerufen am 2019-03-09

[Rd19] *Rdiff - Linux man page*. URL https://linux.die.net/man/1/rdiff. - abgerufen am 2019-03-18

[Re18] REINHEIMER, S. (Hrsg.): *Cloud Computing: die Infrastruktur der Digitalisierung*, *Edition HMD*. Wiesbaden : Springer Fachmedien Wiesbaden GmbH, 2018 — ISBN 978-3-658-20966-7

[Rs19] *Rsync*. URL https://rsync.samba.org/. - abgerufen am 2019-03-18

[Ru08] RUTENBECK, SASCHA: *Business Continuity Planning*. Koblenz : Universität Koblenz, 2008

[Sc10] SCHWENK, JÖRG: *Sicherheit und Kryptographie im Internet: von sicherer E-Mail bis zu IP-Verschlüsselung*, *Studium*. 3. Auflage. Wiesbaden : Vieweg + Teubner, 2010 — ISBN 978-3-8348-0814-1

[Se19] *SecureBeam - control the cloud – Apps bei Google Play*. URL https://play.google.com/store/apps/details?id=com.securebeam&hl=de. - abgerufen am 2019-03-11

[Se17] SEIDEL, PHILIPP: *Vergleich der Kompression bei Borg Backup*. URL https://www.dinotools.de/2017/04/11/vergleich-der-kompression-bei-borg-backup/. - abgerufen am 2019-03-18

[Sp19] SPATH PRINTWARE + SERVICE GMBH & CO. KG: *Backup Sicherungsmedien | spath printware + service*. URL https://www.spath-printware.de/produkte/backup-sicherungsmedien/. - abgerufen am 2019-04-10

[SH18] SOBELL, MARK G. ; HELMKE, MATTHEW: *A practical guide to linux commands, editors, and shell programming*. 4. Auflage. Boston, MA : Addison-Wesley, 2018 — ISBN 978-0-13-477460-2

[So19] *Solid-State-Drive*. URL https://de.wikipedia.org/w/index.php?title=Solid-State-Drive&oldid=185212299. - abgerufen am 2019-03-08

[Sp17] SPICHALE, KAI: *API-Design: Praxishandbuch für Java- und Webservice-Entwickler*. Korrigierter Nachdruck. Heidelberg : dpunkt.verlag, 2017 — ISBN 978-3-86490-387-8

[Sp18] SPRINGER FACHMEDIEN WIESBADEN GMBH: *Definition: Cloud Computing*. URL https://wirtschaftslexikon.gabler.de/definition/cloud-computing-53360/version-276453. - abgerufen am 2019-02-07

[St18a] STATISTIK AUSTRIA: *Download-Geschwindigkeit in Unternehmen 2018*. URL https://www.statistik.at/wcm/idc/groups/b/documents/webobj/mdaw/mdqy/~edisp/042329.png. - abgerufen am 2019-03-06

[St19b] STATISTA GMBH: *Betriebssysteme - Marktanteile weltweit bis Januar 2019 | Statistik*. URL https://de.statista.com/statistik/daten/studie/157902/umfrage/marktanteil-der-genutzten-betriebssysteme-weltweit-seit-2009/. - abgerufen am 2019-03-16

[St18b] STATISTIK AUSTRIA: *IKT-Einsatz in Unternehmen*. URL http://www.statistik.at/web_de/statistiken/energie_umwelt_innovation_mobilitaet/informationsgesellschaft/ikt-einsatz_in_unternehmen/022195.html. - abgerufen am 2019-05-04

[St19a] STREPPEL, HARTMUT: *Hochverfügbarkeit versus Disaster Recovery*. URL https://www.doag.org/formes/pubfiles/5053585/2013-INF-Hartmut_Streppel-Hochverfuegbarkeit_versus_Disaster_Recovery-Manuskript.pdf. - abgerufen am 2019-02-16

[Su19] SUGARSYNC: *File Sync & Online Backup - Access and File Sharing from Any Device | SugarSync*. URL https://www2.sugarsync.com/pricing. - abgerufen am 2019-03-09

[Te14] TECHTARGET GERMANY GMBH: *Was ist Business Continuity und Disaster-Recovery (BC/DR)?* URL https://www.computerweekly.com/de/definition/Business-Continuity-und-Disaster-Recovery-BC-DR. - abgerufen am 2019-03-02

[Te19a] TECHTARGET GERMANY GMBH: *Vor- und Nachteile von Cloud-Backups*. URL https://www.computerweekly.com/de/tipp/Vor-und-Nachteile-von-Cloud-Backups. - abgerufen am 2019-03-14

[Te19b] TELEKOM DEUTSCHLAND GMBH: *MagentaCLOUD: Speicher erweitern | Telekom*. URL https://cloud.telekom-dienste.de/speicher-erweiterung. - abgerufen am 2019-03-09

[Ti15] TILKOV, STEFAN ; EIGENBRODT, MARTIN ; SCHREIER, SILVIA ; WOLF, OLIVER: *REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web*. 3. Auflage. Heidelberg : dpunkt.verlag, 2015 — ISBN 978-3-86491-644-1

[Th19] *The GNU Privacy Guard*. URL https://www.gnupg.org/. - abgerufen am 2019-03-18

[Tr19a] TRESORIT: *Cloud Speicher für Unternehmen - Business Cloud aus der Schweiz*. URL https://tresorit.com/de/. - abgerufen am 2019-02-17

[Tr19b] TRESORIT: *Sicherer Cloud Dienst - Tresorit Preise*. URL https://tresorit.com/de/preise. - abgerufen am 2019-03-09

[TM96] TRIDGELL, ANDREW ; MACKERRAS, PAUL: *The rsync algorithm*. Canberra : The Australian National University, 1996

[Tr19c] TRUSTED GMBH: *Cloud Speicher Dienste Vergleich: Die 38 besten Anbieter 2019 im Test*. URL https://trusted.de/cloud-speicher. - abgerufen am 2019-02-14

[US19] US DEPARTMENT OF COMMERCE: *Privacy Shield*. URL https://www.privacyshield.gov/list. - abgerufen am 2019-02-16

[Ve16] VEEAM SOFTWARE: *Befolgen der 3-2-1-Regel der Datensicherung mit Veeam Backup & Replication*. URL https://www.veeam.com/blog/de/how-to-follow-the-3-2-1-backup-rule-with-veeam-backup-replication.html. - abgerufen am 2019-03-11

Abbildungsverzeichnis

Abbildung 1: Evolution des Computing - 5 -

Abbildung 2: Übersicht Cloud-Deployment-Varianten - 7 -

Abbildung 3: Serviceebenen der Cloud - 10 -

Abbildung 4: Verteilte Speicherung in der Cloud - 20 -

Abbildung 5: SecureBeam - 22 -

Abbildung 6: MultCloud - 23 -

Abbildung 7: Zeitliche Abfolge BCP, IR und DR - 25 -

Abbildung 8: Sicherungsstrategien für Fall 1 und Fall 2 - 41 -

Abbildung 9: Systemlandschaft Prototyp - 46 -

Abbildung 10: Ausgabe des Befehls "stat" - 49 -

Abbildung 11: Rsync - Rollende Bildung der Prüfsumme - 50 -

Abbildung 12: Beispiele für URIs - 55 -

Abbildung 13: XML-Nachricht mit URI-Referenz - 56 -

Abbildung 14: Definition der Repräsentationsform - 56 -

Abbildung 15: Datenbestand - 66 -

Abbildung 16: Ausgabe des Prototyps bei initialer Onlinesicherung - 66 -

Abbildung 17: Ausgabe des Prototyps bei Veränderung einer Datei - 67 -

Abbildung 18: Beispielhafter Datenbestand von Amazon S3 - 67 -

Abbildung 19: Ausgabe des Prototyps bei Rücksicherung einer Datei - 68 -

Abbildung 20: Ausgabe des Prototyps bei Durchführung einer Vollsicherung - 69 -

Abbildung 21: Ausgabe des Prototyps bei Rücksicherung aus Band- und Onlinesicherung - 70 -

Tabellenverzeichnis

Tabelle 1: Risiken und mögliche Maßnahmen (Cloud Storage Services) - 18 -

Tabelle 2: Beispiele für Cloud Storage Services - 19 -

Tabelle 3: Sicherungsmedien - 35 -

Tabelle 4: Kosten der Cloud Storage Services (Stand März 2019) - 38 -

Tabelle 5: Ablauf für Upload einer Datei - 45 -

Tabelle 6: Generierter Dateiname - 53 -

Tabelle 7: Mögliche Programmaufrufe zur Interaktion mit den CSPs - 60 -

Tabelle 8: Mögliche Aufrufe für "prototyp.sh" - 61 -

Abkürzungsverzeichnis

AES Advanced Encryption Standard
API Application Programming Interface
BASH Bourne-Again-Shell
BC Business Continuity
BCM Business Continuity Management
BCP Business Continuity Planning
BIA Business Impact Analyse
BSI Sicherheit in der Informationstechnik
CLI Command Line Interpreter
DR Disaster Recovery
ENISA European Network and Information Security Agency
GPG GNU Privacy Guard
IAM Identity and Access Management
IR Incident Response
LAN Local Area Network
LTO Linear Tape Open
NAS Network Attached Storage
NIST National Institute of Standards and Technology
PBX Private Branch Exchange
REST Representational State Transfer
SAN Storage Area Networks
URI Uniform Resource Identifier
WAN Wide Area Networks
WWW World Wide Web

Anhang A: Datei „readme.txt“

```
########################################################################

Vorbereitungen

########################################################################

Amazon S3 (Glacier)

      -Benutzerkonto bei Amazon AWS anlegen
       (https://aws.amazon.com/de/s3/)

      -Unter "Services" den Dienst "S3" auswählen und "Bucket erstellen"
       wählen

      -Unter "Services" den Dienst "IAM" auswählen um einen Benutzer zu
       erstellen. Danach mit Doppelklick den Benutzer bearbeiten und
       über "Berechtigungen hinzufügen" beispielsweise die vorhandene
       Richtlinie "AmazonS3FullAccess" vergeben.

      -Angezeigte Werte für "Key" und "Secret" sowie den Namen des
       erstellten Buckets in die Datei "amazon_s3_glacier.sh" übertragen

Backblaze B2

      -Benutzerkonto bei Backblaze erstellen
       (https://www.backblaze.com/b2/cloud-storage.html)

      -Nach dem Login unter Menüpunkt "Buckets" die Anlage eines selbigen
       vornehmen

      -Die angezeigten Werte für "Bucket Name" und "Bucket ID" in die
       Datei "backblaze.sh" übertragen

      -Im selben Menü kann man durch Betätigung der Schaltfläche "Show
       Account ID and Application Key" Schlüsselpaare anlegen.

      -Nach der Anlage die Werte für "applicationKeyId" und
       "applicationKeySecret" in die Datei "backblaze.sh" übertragen

Google Drive

      -Benutzerkonto bei Google anlegen
       (https://accounts.google.com/signin)

      -Nach dem Login in die Google API Console wechseln
       (https://console.developers.google.com/apis/dashboard)

      -Projekt erstellen
```

-Über die Schaltfläche „APIs und Dienste aktivieren“ die API für Google Drive aktivieren

-Am Ziel des Menüpunkts „Anmeldedaten“ einen neuen Eintrag für ein CLI-Tool anlegen

-Client-ID und Client-Schlüssel in die Datei "google_drive.sh" übertragen

-"./google_drive.sh authorize" im Terminal ausführen

-Den angezeigten Link im Webbrowser aufrufen und den angezeigten Schlüssel eingeben

-Den Zugriff des Clients auf Google Drive über die Schaltfläche "Zulassen" gewähren

-Ausgabe im Terminal beachten. Nach Erscheinen des "Refresh-Tokens" diesen ebenso in die Datei "google_drive.sh" übertragen

Datei mit verschlüsseltem Master-Passwort erstellen

-"./prototyp set_password" im Terminal aufrufen und den Anweisung folgen

##

Hauptfunktionen des Prototyps

##

./prototyp.sh fullbackup_nas_tape Um eine Vollsicherung durchzuführen. Die Daten werden in den Mountpoints für NAS-System und Tape-Laufwerk abgelegt.

./prototyp.sh onlinebackup Onlinebackup durchführen. Bei Veränderungen einer Datei werden die geänderten Teile im Vergleich zur Version in der Vollsicherung online in verteilter Form abgelegt. Neu erstellte Files welche in keiner Vollsicherung vorkommen werden einmalig vollständig online in verteilter Form abgelegt.

./prototyp.sh restore Zur Rücksicherung der Daten unter Verwendungen der Tape-Daten sowie der Online abgelegten Fragmente.

```
#########################################################################

Hilfsfunktionen des Prototyps

#########################################################################

./prototyp.sh get_filelist    Hilfsfunktion zum Erstellen einer
                              Dateiliste der online abgelegten Daten. Der
                              Aufruf mit dem Parameter "restore" verwendet
                              diese Funktion.

./prototyp.sh download_data   Hilfsfunktion zum Herunterladen aller
                              online abgelegten Daten. Der Aufruf mit dem
                              Parameter "restore" verwendet diese Funktion.

./prototyp.sh reconstruct_data          Hilfsfunktion zum
                              Rekonstruieren der vollständigen Daten durch
                              Zusammenfügen der einzelnen Fragemente. Der
                              Aufruf mit dem Parameter "restore" verwendet
                              diese Funktion.
```

Anhang B: Sourcecode „prototyp.sh“

```
001 #!/bin/bash
002 ######################################################################
003
004 #Benutzerparameter
005 DIR_DATASTORE="/srv/1_filestorage/"
006 DIR_NAS="/srv/2_nas/"
007 DIR_TAPE="/srv/3_tape/"
008 DIR_SCRIPT="/srv/4_script/"
009
010 ######################################################################
011
012 #Variablen setzen
013 DIR_METADATA=$DIR_SCRIPT"metadata/"
014 DIR_METADATA_TSTAMP=$DIR_METADATA"timestamps/"
015 DIR_METADATA_SIGNATURE=$DIR_METADATA"signature/"
016 DIR_METADATA_UPLOAD=$DIR_METADATA"upload/"
017 DIR_METADATA_DOWNLOAD=$DIR_METADATA"download/"
018 DIR_METADATA_RESTORE=$DIR_METADATA"restore/"
019 FILE_MASTERPASSWORD=$DIR_METADATA/"masterpw"
020 SERVICE1_NAME="Google Drive"
021 SERVICE1_SCRIPT="google_drive.sh"
022 SERVICE2_NAME="Amazon S3 Glacier"
023 SERVICE2_SCRIPT="amazon_s3_glacier.sh"
024 SERVICE3_NAME="Backblaze"
025 SERVICE3_SCRIPT="backblaze.sh"
026 ANZAHL_FILEPARTS=3
027 SUFFIX_FILEPART1="0"
028 SUFFIX_FILEPART2="1"
029 SUFFIX_FILEPART3="2"
030
031 ######################################################################
032
033 #Log-Funktion
034 function log {
035     DEBUG=0
036     DISPLAY=0
037
038     #Wenn kein zweiter Parameter übergeben wird, dann übergebenen
        Text ausgeben
039     if [ ${#2} -eq 0 ]; then
040         DISPLAY=1
041     fi
042
043     #Wenn als zweiter Parameter "DEBUG" übergeben wurde und die
        Variable DEBUG auf 1 steht, dann übergebenen Text ausgeben
044     if [[ "$2" == "DEBUG" && $DEBUG -eq 1 ]]; then
045         DISPLAY=1
046     fi
047
048     #Übergebenen Text um Timestamp erweitern und ausgeben
049     if [ $DISPLAY -eq 1 ]; then
050         echo $(date +"%d-%m-%Y %T") " - " $1;
051     fi
052 }
053
054 ######################################################################
```

```
055
056 #Prüfe ob Datei geändert wurde
057 function check_change_timestamp {
058     #Variablen setzen
059     DIR_METADATA_TSTAMP=$1
060     FILE=$2
061     RETVAL="0"
062
063     #Prüfen ob Metadaten-Verzeichnis vorhanden
064     if [ -d $(dirname "$DIR_METADATA_TSTAMP$FILE") ]; then
065         log "Verzeichnis fuer Timestamp vorhanden "$(dirname
        "$DIR_METADATA_TSTAMP$FILE") "DEBUG"
066     else
067         log "Verzeichnis fuer Timestamp erstellen "$(dirname
        "$DIR_METADATA_TSTAMP$FILE") "DEBUG"
068         mkdir -p $(dirname "$DIR_METADATA_TSTAMP$FILE");
069     fi
070
071     #Prüfen ob Timestamp-Datei vorhanden
072     if [ -f $DIR_METADATA_TSTAMP/$FILE ]; then
073         log "Datei fuer Timestamp vorhanden "$FILE "DEBUG"
074         #Vergleich der Timestamps zwischen aktuellem Timestamp und
        #dem Timestamp der letzten Sicherung vergleichen
075         stat -c '%z' $FILE > $DIR_METADATA_TSTAMP.comparefile
076         if [[ $(cat $DIR_METADATA_TSTAMP/$FILE) = $(cat
        $DIR_METADATA_TSTAMP.comparefile) ]]; then
077             RETVAL="1"
078         fi
079     fi
080
081     #Timestamp aktualisieren
082     stat -c '%z' $FILE > $DIR_METADATA_TSTAMP$FILE
083
084     return $RETVAL
085 }
086
087 ########################################################################
088
089 #Hochzuladendes File berechenen
090 function calculate_uploadfile {
091     #Variablen setzen
092     DIR_METADATA_SIGNATURE=$1
093     FILE=$2
094     RETVAL=""
095
096     #Prüfe ob Verzeichnis für Dateisignatur vorhanden
097     if [ -d $(dirname "$DIR_METADATA_SIGNATURE$FILE") ]; then
098         log "Verzeichnis fuer Signatur vorhanden "$(dirname
        "$DIR_METADATA_SIGNATURE$FILE") "DEBUG"
099     else
100         log "Verzeichnis fuer Signatur erstellen "$(dirname
        "$DIR_METADATA_SIGNATURE$FILE") "DEBUG"
101         mkdir -p $(dirname "$DIR_METADATA_SIGNATURE$FILE");
102     fi
103
104     #Prüfen ob Dateisignatur vorhanden
105     if [ -f $DIR_METADATA_SIGNATURE/$FILE ]; then
106         log "Signaturdatei vorhanden "$FILE "DEBUG"
107         #Unterschied zwischen aktueller Datei und Datei in der
```

```
        Vollsicherung auf Basis der Dateisignatur berechnen
108     rdiff delta $DIR_METADATA_SIGNATURE/$FILE $FILE
        $DIR_METADATA_SIGNATURE/delta
109     log "Aktuelle Dateigroesse "$(stat -c%s $FILE)" Byte"
110     log "Dateiunterschied wird geladen"
111     RETVAL=$DIR_METADATA_SIGNATURE/delta
112 else
113     log "Vollstaendige Datei muss geladen werden"
114     RETVAL=$FILE
115 fi
116
117 #Dateisignatur aktualisieren
118 rdiff signature $FILE $DIR_METADATA_SIGNATURE/$FILE
119
120 RET=$RETVAL
121 }
122
123 ########################################################################
124
125 #Masterpasswort auf Basis eines Benutzerpassworts berechnen und in
    Datei ablegen
126 if [ $1 == "set_password" ]; then
127     echo -n Passwort:
128     read -s password
129     echo $password | sha256sum | awk '{print $1}' >
        $FILE_MASTERPASSWORD
130     clear
131     log "Passwort erfolgreich gesetzt!"
132 fi
133
134 ########################################################################
135
136 #Kontinuierliches Cloud-Backup durchführen
137 if [ $1 == "onlinebackup" ]; then
138     while true
139     do
140         #Quellverzeichnis lesen
141         FILELIST="$(find -L "$DIR_DATASTORE" -type f)"
142         echo "$FILELIST" | while read FILE; do
143             #Bestimmte Dateien nicht berücksichtigen
144             if [[ $FILE == *".tmp"* || $FILE == *".~lock."* ]]; then
145                 continue
146             fi
147
148             #Timestamp-Metadaten vergleichen
149             log "Pruefe $FILE ..." "DEBUG"
150             TIMESTAMP=$(date +"%Y%m%d%H%M%S")
151
152             #Prüfe ob Datei geändert wurde
153             check_change_timestamp $DIR_METADATA_TSTAMP $FILE
154             if [ $? -eq 1 ]; then
155                 log "File unveraendert" "DEBUG"
156             else
157                 log "File "$FILE" verarbeiten"
158                 #Hochzuladende Datei berechnen
159                 UPLOADFILE=""
160                 #Veränderte Inhalte berechnen
161                 calculate_uploadfile $DIR_METADATA_SIGNATURE $FILE
162                 UPLOADFILE=$RET
```

```
163                 log "File "$UPLOADFILE DEBUG
164
165                 #Datei komprimieren
166                 lz4 -f $UPLOADFILE $DIR_METADATA_UPLOAD
                    "uploadfile.lz4" >/dev/null 2>&1
167                 log "Komprimierung: "$(stat -c%s $UPLOADFILE)" Byte
                    zu "$(stat -c%s $DIR_METADATA_UPLOAD"uploadfile.lz4")
                    " Byte"
168                 #Originalen Dateinamen in Datei schreiben und
                    TAR-Archiv erstellen
169                 echo $FILE > $DIR_METADATA_UPLOAD"original_filename"
170                 cd $DIR_METADATA_UPLOAD
171                 tar -czf $DIR_METADATA"upload.tar" * > /dev/null 2>&1
172
173                 #Onlinedateiname generieren
174                 ONLINEFILEBASENAME=$(echo $FILE | sha256sum | awk
                    '{print $1}')
175
176                 #Verschlüsselung der Daten vornehmen
177                 gpg --batch --cipher-algo AES256 --symmetric --
                    passphrase-file $FILE_MASTERPASSWORD --output
                    $DIR_METADATA"upload_verschluesselt.tar"
                    $DIR_METADATA"upload.tar"
178
179                 #Aufteilen in Teildateien
180                 SIZE=$(ls -l
                    $DIR_METADATA"upload_verschluesselt.tar" | awk '{
                    print $5 }')
181                 cd $DIR_METADATA
182                 split -d --bytes $(($SIZE/$ANZAHL_FILEPARTS+1))
                    $DIR_METADATA"upload_verschluesselt.tar" "PART"
183
184                 #Upload der Teildateien
185                 log "Upload PART00 nach ..."
186                 ONLINEFILENAME=
                    $DIR_METADATA$ONLINEFILEBASENAME$TIMESTAMP
                    $SUFFIX_FILEPART1
187                 mv $DIR_METADATA"PART00" $ONLINEFILENAME
188                 log "$SERVICE1_NAME "
189                 $DIR_SCRIPT/$SERVICE1_SCRIPT "upload" $ONLINEFILENAME
190                 log "$SERVICE2_NAME "
191                 $DIR_SCRIPT/$SERVICE2_SCRIPT "upload" $ONLINEFILENAME
192                 rm $ONLINEFILENAME
193
194                 log "Upload PART01 nach ..."
195                 ONLINEFILENAME=
                    $DIR_METADATA$ONLINEFILEBASENAME$TIMESTAMP
                    $SUFFIX_FILEPART2
196                 mv $DIR_METADATA"PART01" $ONLINEFILENAME
197                 log "$SERVICE2_NAME "
198                 $DIR_SCRIPT/$SERVICE2_SCRIPT "upload" $ONLINEFILENAME
199                 log "$SERVICE3_NAME "
200                 $DIR_SCRIPT/$SERVICE3_SCRIPT "upload" $ONLINEFILENAME
201                 rm $ONLINEFILENAME
202
203                 log "Upload PART02 nach ..."
204                 ONLINEFILENAME=
                    $DIR_METADATA$ONLINEFILEBASENAME
                    $TIMESTAMP$SUFFIX_FILEPART3
```

```
205                 mv $DIR_METADATA"PART02" $ONLINEFILENAME
206                 log "$SERVICE3_NAME "
207                 $DIR_SCRIPT/$SERVICE3_SCRIPT "upload" $ONLINEFILENAME
208                 log "$SERVICE1_NAME "
209                 $DIR_SCRIPT/$SERVICE1_SCRIPT "upload" $ONLINEFILENAME
210                 rm $ONLINEFILENAME
211
212                 #Temporäre Dateien löschen
213                 if [ -f $DIR_METADATA"upload_verschluesselt.tar" ];
                    then
214                     rm $DIR_METADATA"upload_verschluesselt.tar"
215                 fi
216
217                 log "Upload erfolgreich!"
218             fi
219         done
220         sleep 3;
221     done
222 fi
223
224 ###########################################################################
225
226 #Dateiliste laden
227 if [ $1 == "get_filelist" ]; then
228     retVal=$($DIR_SCRIPT/$SERVICE1_SCRIPT "get_onlinefile_list")
229     if [ "$retVal" != "error" ]; then
230         echo $retVal > $DIR_METADATA"filelist"
231         log "Dateiliste erfolgreich erstellt!"
232     else
233         retVal=$($DIR_SCRIPT/$SERVICE2_SCRIPT "get_onlinefile_list" )
234         if [ "$retVal" != "error" ]; then
235             echo $retVal > $DIR_METADATA"filelist"
236             log "Dateiliste erfolgreich erstellt!"
237         else
238             retVal=$($DIR_SCRIPT/$SERVICE3_SCRIPT
                    "get_onlinefile_list" )
239             if [ "$retVal" != "error" ]; then
240                 echo $retVal > $DIR_METADATA"filelist"
241                 log "Dateiliste erfolgreich erstellt!"
242             else
243                 log "Dateiliste konnte nicht erstellt werden!"
244             fi
245         fi
246     fi
247 fi
248
249 ###########################################################################s
250
251 #Daten herunterladen
252 if [ $1 == "download_data" ]; then
253     #Temporäre Dateien löschen
254     if [ -f $DIR_METADATA"filelist_sorted" ]; then
255         rm $DIR_METADATA"filelist_sorted"
256     fi
257
258     #Letztes Zeichen der Dateinamen abschneiden (Teilindex) und in
        Datei schreiben
259     for ONLINEFILENAME in `cat $DIR_METADATA"filelist"`;do
260         echo "${ONLINEFILENAME::-1}">>$DIR_METADATA"filelist_sorted"
```

```
261     done
262
263     #Doppelte Einträge entfernen und chronologisch sortieren
264     echo `cat $DIR_METADATA"filelist_sorted" | sort | uniq` >
        $DIR_METADATA"filelist_sorted"
265
266     #Daten herunterladen
267     rm -r $DIR_METADATA_DOWNLOAD/*
268     for ONLINEFILENAME in `cat $DIR_METADATA"filelist_sorted"`;do
269         log "Lade "$ONLINEFILENAME" ..."
270         log "Download von $SERVICE1_NAME "
271         $DIR_SCRIPT/$SERVICE1_SCRIPT "download_file"
            $ONLINEFILENAME$SUFFIX_FILEPART1 $DIR_METADATA_DOWNLOAD
272         log "Download von $SERVICE2_NAME "
273         $DIR_SCRIPT/$SERVICE2_SCRIPT "download_file"
            $ONLINEFILENAME$SUFFIX_FILEPART2 $DIR_METADATA_DOWNLOAD
274         log "Download von $SERVICE3_NAME "
275         $DIR_SCRIPT/$SERVICE3_SCRIPT "download_file"
            $ONLINEFILENAME$SUFFIX_FILEPART3 $DIR_METADATA_DOWNLOAD
276     done
277 fi
278
279 ########################################################################
280
281 #Dateien aus den heruntergeladenen Daten rekonstruieren
282 if [ $1 == "reconstruct_data" ]; then
283     if [ -f $DIR_METADATA"filelist_sorted" ]; then
284         rm $DIR_METADATA"filelist_sorted"
285     fi
286
287     #Dateinamen der heruntergeladenen Dateinamen in Datei schreiben
        und sortieren
288     ls $DIR_METADATA_DOWNLOAD > $DIR_METADATA"filelist"
289     for FILENAME in `cat $DIR_METADATA"filelist"`;do
290         echo "${FILENAME::-1}">>$DIR_METADATA"filelist_sorted"
291     done
292     echo `cat $DIR_METADATA"filelist_sorted" | sort | uniq` >
        $DIR_METADATA"filelist_sorted"
293
294     #Rekonstruktion starten
295     FILENAMEOLD="_"
296     for FILENAME in `cat $DIR_METADATA"filelist_sorted"`;do
297         log "Rekonstruiere $FILENAME"
298
299         #Teile in der richtigen Reihenfolge zusammensetzen
300         if [ -f $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART1 ];
            then
301             cat $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART1 >
                $DIR_METADATA_DOWNLOAD/$FILENAME
302             rm $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART1
303         fi
304         if [ -f $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART2 ];
            then
305             cat $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART2 >>
                $DIR_METADATA_DOWNLOAD/$FILENAME
306             rm $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART2
307         fi
308         if [ -f $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART3 ];
            then
```

```
309             cat $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART3 >>
                $DIR_METADATA_DOWNLOAD/$FILENAME
310             rm $DIR_METADATA_DOWNLOAD/$FILENAME$SUFFIX_FILEPART3
311         fi
312         if [ -f $DIR_METADATA_DOWNLOAD/$FILENAME"_" ]; then
313             cat $DIR_METADATA_DOWNLOAD/$FILENAME"_" >>
                $DIR_METADATA_DOWNLOAD/$FILENAME
314             rm $DIR_METADATA_DOWNLOAD/$FILENAME"_"
315         fi
316
317         #Entschlüsseln
318         gpg --output $DIR_METADATA_DOWNLOAD/$FILENAME"_DECRYPT" --
            batch --passphrase-file  $FILE_MASTERPASSWORD --decrypt
            $DIR_METADATA_DOWNLOAD/$FILENAME > /dev/null 2>&1
319         if [ $? -ne 0 ]; then
320             log "Fehler beim Entschlüsseln"
321             continue
322         else
323             rm $DIR_METADATA_DOWNLOAD$FILENAME
324             mv $DIR_METADATA_DOWNLOAD$FILENAME"_DECRYPT"
                $DIR_METADATA_DOWNLOAD$FILENAME
325         fi
326
327         #Daten aus den komprimierten Archiven extrahieren
328         cd $DIR_METADATA_DOWNLOAD
329         tar -xvf $DIR_METADATA_DOWNLOAD$FILENAME > /dev/null
330         if [ $? -ne 0 ]; then
331             log "Fehler beim Entpacken"
332             continue
333         else
334             RESTOREFILENAME=$(cat
                $DIR_METADATA_DOWNLOAD"original_filename")
335
336             #Check ob Verzeichnis für Restore der Datei vorhanden
337             if [ -d $(dirname
                "$DIR_METADATA_RESTORE$RESTOREFILENAME") ]; then
338                 log "Verzeichnis fuer Restore vorhanden "$(dirname
                    "$DIR_METADATA_RESTORE$RESTOREFILENAME") "DEBUG"
339             else
340                 log "Verzeichnis fuer Restore erstellen "$(dirname
                    "$DIR_METADATA_RESTORE$RESTOREFILENAME") "DEBUG"
341                 mkdir -p $(dirname
                    "$DIR_METADATA_RESTORE$RESTOREFILENAME");
342             fi
343
344             #Existiert aus einem vorherigen Druchlauf bereits eine
                Datei, so ist die neue Datei damit zu verschmelzen
345             if [ $FILENAMEOLD == ${FILENAME::-14} ]; then
346                 log "Verschmelzen ..."
347                 #Entpacken
348                 lz4cat $DIR_METADATA_DOWNLOAD"uploadfile.lz4" >
                    $DIR_METADATA_DOWNLOAD"delta"
349                 rdiff patch $DIR_METADATA_RESTORE$RESTOREFILENAME
                    $DIR_METADATA_DOWNLOAD"delta"
                    $DIR_METADATA_RESTORE$RESTOREFILENAME"melted"
350                 rm $DIR_METADATA_RESTORE$RESTOREFILENAME && mv
                    $DIR_METADATA_RESTORE$RESTOREFILENAME"melted"
                    $DIR_METADATA_RESTORE$RESTOREFILENAME
351                 log ${DIR_METADATA_RESTORE::-1}$RESTOREFILENAME"
```

```
                aktualisiert"
352             else
353                 #Entpacken
354                 lz4cat $DIR_METADATA_DOWNLOAD"uploadfile.lz4" >
                    $DIR_METADATA_RESTORE$RESTOREFILENAME
355                 log ${DIR_METADATA_RESTORE::-1}$RESTOREFILENAME"
                    erstellt"
356             fi
357
358             #Dateiname als Merker für den nächsten Durchlauf setzen
359             FILENAMEOLD=${FILENAME::-14} #Filename ohne Timestamp
360         fi
361     done
362 fi
363
364 ########################################################################
365
366 #Restore starten
367 if [ $1 == "restore" ]; then
368     #Rekursiver Aufruf
369     $0 "get_filelist"
370     $0 "download_data"
371
372     #Daten aus der Tape-Sicherung kopieren
373     if [ -z "$(ls -A $DIR_TAPE)" ]; then
374         log "Tape-Verzeichnis ist leer" "DEBUG"
375     else
376         log "Lade Tape-Sicherung ..."
377         cp $DIR_TAPE/* $DIR_METADATA_DOWNLOAD
378     fi
379
380     #Daten rekonstruieren
381     $0 "reconstruct_data"
382 fi
383
384 ########################################################################
385
386 #Vollsicherung auf NAS-System und Tape-Laufwerk simulieren
387 if [ $1 == "fullbackup_nas_tape" ]; then
388     #Dateiliste der online abgelegten Files erstellen und sortieren
389     $0 "get_filelist"
390     #Liste sortieren
391     if [ -f $DIR_METADATA"filelist_sorted" ]; then
392         rm $DIR_METADATA"filelist_sorted"
393     fi
394     for FILENAME in `cat $DIR_METADATA"filelist"`;do
395         echo "${FILENAME::-1}">>$DIR_METADATA"filelist_sorted"
396     done
397     echo `cat $DIR_METADATA"filelist_sorted" | sort | uniq` >
    $DIR_METADATA"filelist_sorted"
398
399     #Quellverzeichnis lesen
400     FILELIST="$(find -L "$DIR_DATASTORE" -type f)"
401     echo "$FILELIST" | while read FILE; do
402         #Bestimmte Dateien nicht berücksichtigen
403         if [[ $FILE == *".tmp"* || $FILE == *".~lock."* ]]; then
404             continue
405         fi
406
```

```
407         log "Sichere $FILE ..."
408
409         #Timestamp-Metadaten aktualisieren
410         TIMESTAMP=$(date +"%Y%m%d%H%M%S")
411         check_change_timestamp $DIR_METADATA_TSTAMP $FILE
412
413         #Neuerstellung der Signatur erzwingen, welche die Datei in
            der Vollsicherung beschreibt
414         SAVEFILE=""
415         if [ -f $DIR_METADATA_SIGNATURE$FILE ]; then
416             rm  $DIR_METADATA_SIGNATURE$FILE
417         fi
418         calculate_uploadfile $DIR_METADATA_SIGNATURE $FILE
419         SAVEFILE=$RET
420         log "File "$SAVEFILE
421
422         #Datei komprimieren
423         lz4 -f $SAVEFILE $DIR_METADATA_UPLOAD"uploadfile.lz4" >/dev/
            null 2>&1
424         log "Komprimierung: "$(stat -c%s $SAVEFILE)" Byte zu "$(stat
            -c%s $DIR_METADATA_UPLOAD"uploadfile.lz4")" Byte"
425         #Originalen Dateinamen in Datei schreiben und TAR-Archiv
            erstellen
426         echo $FILE > $DIR_METADATA_UPLOAD"original_filename"
427         cd $DIR_METADATA_UPLOAD
428         tar -czf $DIR_METADATA"savefile.tar" * > /dev/null 2>&1
429
430         #Dateiname für Savefile generieren
431         SAVEFILEBASENAME=$(echo $FILE | sha256sum | awk '{print $1}')
432
433         #Verschlüsselung der Daten
434         gpg --batch --cipher-algo AES256 --symmetric --
            passphrase-file $FILE_MASTERPASSWORD --output  $DIR_METADATA
            "savefile_verschluesselt.tar" $DIR_METADATA"savefile.tar"
435
436         #Kopieren der Daten in die Verzeichnisse für die
            Vollsicherungen
437         cp $DIR_METADATA"savefile_verschluesselt.tar"
            $DIR_NAS$SAVEFILEBASENAME$TIMESTAMP"_"
438         if [ $? -ne 0 ]; then
439             log "Fehler bei Backup"
440         fi
441
442         #Temporäre Dateien löschen
443         if [ -f $DIR_METADATA"savefile_verschluesselt.tar" ]; then
444             rm $DIR_METADATA"savefile_verschluesselt.tar"
445         fi
446
447         #Alle zugehörigen Onlinefiles löschen
448         for FILENAME in `cat $DIR_METADATA"filelist_sorted"`;do
449             if [[ $FILENAME == *$SAVEFILEBASENAME* ]]; then
450                 log "File online vorhanden "$FILENAME
451                 log "Loesche PART00 ..."
452                 ONLINEFILENAME=$FILENAME$SUFFIX_FILEPART1
453                 log "$SERVICE1_NAME "
454                 retVal=$($DIR_SCRIPT/$SERVICE1_SCRIPT "delete"
                    $ONLINEFILENAME)
455                 log "$SERVICE2_NAME "
456                 retval=$($DIR_SCRIPT/$SERVICE2_SCRIPT "delete"
```

```
                $ONLINEFILENAME)
457
458                 log "Loesche PART01 ..."
459                 ONLINEFILENAME=$FILENAME$SUFFIX_FILEPART2
460                 log "$SERVICE2_NAME "
461                 retVal=$($DIR_SCRIPT/$SERVICE2_SCRIPT "delete"
                    $ONLINEFILENAME)
462                 log "$SERVICE3_NAME "
463                 retVal=$($DIR_SCRIPT/$SERVICE3_SCRIPT "delete"
                    $ONLINEFILENAME)
464
465                 log "Loesche PART02 ..."
466                 ONLINEFILENAME=$FILENAME$SUFFIX_FILEPART3
467                 log "$SERVICE3_NAME "
468                 retVal=$($DIR_SCRIPT/$SERVICE3_SCRIPT "delete"
                    $ONLINEFILENAME)
469                 log "$SERVICE1_NAME "
470                 retVal=$($DIR_SCRIPT/$SERVICE1_SCRIPT "delete"
                    $ONLINEFILENAME)
471             else
472                 log "Loeschen von Onlinedaten nicht erforderlich. "
                    $FILENAME
473             fi
474         done
475     done
476
477     #NAS-Verzeichnis und Tape-Verzeichnis synchronisieren
478     rsync -avP $DIR_NAS $DIR_TAPE > /dev/null 2>&1
479     if [ $? -ne 0 ]; then
480         log "Fehler bei Synchronisation."
481     else
482         log "Synchronisation erfolgreich beendet."
483     fi
484 fi
485
486 ######################################################################
```

Anhang C: Sourcecode „amazon_s3_glacier.sh“

```
001 #!/bin/bash
002 ########################################################################
003
004 #Benutzerparameter
005 S3KEY=          XXXXXXXXXXXXXXXXXXXXXXXXX
006 S3SECRET=       XXXXXXXXXXXXXXXXXXXXXXXXX
007 S3BUCKET=       XXXXXXXXXXXXXXXXXXXXXXXXX
008
009 ########################################################################
010
011 #Vorbelegungen
012 S3STORAGETYPE="STANDARD" #STANDARD | REDUCED_REDUNDANCY | GLACIER |
                             STANDARD_IA | ONEZONE_IA | INTELLIGENT_TIERING |
                             DEEP_ARCHIVE
013 aws_path="/"
014 date=$(date -R)
015 acl="x-amz-acl:private"
016 content_type="application/x-compressed-tar"
017 storage_type="x-amz-storage-class:${S3STORAGETYPE}"
018 AWSREGION="s3-eu-west-1"
019
020 ########################################################################
021
022 #Wert eines JSON-Elements ermitteln
023 function jsonValue() {
024     KEY=$1
025     num=$2
026     awk -F"[,:}][^://]" '{for(i=1;i<=NF;i++){if($i~/\042'$KEY'\042/){print
        $(i+1)}}}' | tr -d '"' | sed -n ${num}p | sed -e 's/[}]*$//' -e
        's/^[[:space:]]*//' -e 's/[[:space:]]*$//' -e 's/[,]*$//'
027 }
028
029 ########################################################################
030
031 #Upload einer Datei
032 if [ $1 == "upload" ]; then
033
034     #Variablen setzen
035     file_path=$(basename $2)
036     string="PUT\n\n$content_type\n$date\n$acl\n$storage_type\n/$S3BUCKET
        $aws_path${file_path##/*/}"
037     signature=$(echo -en "${string}" | openssl sha1 -hmac "${S3SECRET}" -
        binary | base64)
038
039     #Verzeichnis wechseln
040     cd $(dirname $2)
041
042     #Upload durchführen
043     curl --silent -s --retry 1 --retry-delay 1 -X PUT -T "$file_path" \
044           -H "Host: $S3BUCKET.${AWSREGION}.amazonaws.com" \
045           -H "Date: $date" \
046           -H "Content-Type: $content_type" \
047           -H "$storage_type" \
048           -H "$acl" \
049           -H "Authorization: AWS ${S3KEY}:$signature" \
```

```
050             "https://$S3BUCKET.${AWSREGION}.amazonaws.com$aws_path
                ${file_path##/*/} "
051 fi
052
053 ########################################################################
054
055 #Dateiliste laden
056 if [ $1 == "get_onlinefile_list" ]; then
057     #Variablen setzen
058     string="GET\n\n$content_type\n$date\n$storage_type\n/$S3BUCKET
        $aws_path${file_path##/*/}"
059     signature=$(echo -en "${string}" | openssl sha1 -hmac "${S3SECRET}" -
        binary | base64)
060
061     #Dateiliste abrufen
062     retVal=` curl  -s --retry 1 --retry-delay 1 -X GET  \
063        -H "Host: $S3BUCKET.${AWSREGION}.amazonaws.com" \
064        -H "Date: $date" \
065        -H "Content-Type: $content_type" \
066        -H "$storage_type" \
067        -H "Authorization: AWS ${S3KEY}:$signature" \
068         "https://$S3BUCKET.${AWSREGION}.amazonaws.com$aws_path" `
069
070     #Ausgabe aufbereiten
071     if [ $? -eq 0 ]; then
072         #Daten aus XML extrahieren
073         retVal=($(echo $retVal | grep -oP '(?<=Key>)[^<]+' ))
074         for i in ${!retVal[*]}
075         do
076           echo "${retVal[$i]}"
077         done
078     else
079         echo "error"
080     fi
081 fi
082
083 ########################################################################
084
085 #Datei herunterladen
086 if [ $1 == "download_file" ]; then
087     #Variablen setzen
088     download_dir=$3
089     file_path=$(basename $2)
090     string="GET\n\n$content_type\n$date\n$acl\n$storage_type\n/$S3BUCKET
        $aws_path${file_path##/*/}"
091     signature=$(echo -en "${string}" | openssl sha1 -hmac "${S3SECRET}" -
        binary | base64)
092
093     #Datei herunterladen
094     curl  -s --retry 1 --retry-delay 1 -X GET  \
095            -H "Host: $S3BUCKET.${AWSREGION}.amazonaws.com" \
096            -H "Date: $date" \
097            -H "Accept: application/json" \
098            -H "Content-Type: $content_type" \
099            -H "$storage_type" \
100            -H "$acl" \
101            -H "Authorization: AWS ${S3KEY}:$signature" \
102            "https://$S3BUCKET.${AWSREGION}.amazonaws.com$aws_path
               ${file_path##/*/}" > $download_dir/$file_path
```

```
103 fi
104
105 ######################################################################
106
107 #Datei löschen
108 if [ $1 == "delete" ]; then
109     #Variablen setzen
110     file=$(basename $2)
111     string="DELETE\n\n$content_type\n$date\n$acl\n$storage_type\n/
        $S3BUCKET$aws_path${file##/*/}"
112     signature=$(echo -en "${string}" | openssl sha1 -hmac "${S3SECRET}" -
        binary | base64)
113
114     #Datei löschen
115     retVal=`curl --silent -v -s --retry 1 --retry-delay 1 -X DELETE  \
116             -H "Host: $S3BUCKET.${AWSREGION}.amazonaws.com" \
117             -H "Date: $date" \
118             -H "Content-Type: $content_type" \
119             -H "$storage_type" \
120             -H "$acl" \
121             -H "Authorization: AWS ${S3KEY}:$signature" \
122             "https://$S3BUCKET.${AWSREGION}.amazonaws.com$aws_path
                ${file##/*/}" > /dev/null 2>&1`
123 fi
124
125 ######################################################################
```

Anhang D: Sourcecode „backblaze.sh“

```
001 #!/bin/bash
002 ######################################################################
003
004 #Benutzerparameter
005 ACCOUNT_ID=            XXXXXXXXXXXXXXXXXXXXXXXXX
006 APPLICATION_KEY=       XXXXXXXXXXXXXXXXXXXXXXXXX
007 BUCKET_ID=             XXXXXXXXXXXXXXXXXXXXXXXXX
008 BUCKET_NAME=           XXXXXXXXXXXXXXXXXXXXXXXXX
009
010 ######################################################################
011
012 #Wert eines JSON-Elements ermitteln
013 function jsonValue() {
014     KEY=$1
015     num=$2
016     awk -F"[,:}][^://]" '{for(i=1;i<=NF;i++){if($i~/\042'$KEY'\042/){print
        $(i+1)}}}' | tr -d '"' | sed -n ${num}p | sed -e 's/[}]*$//' -e
        's/^[[:space:]]*//' -e 's/[[:space:]]*$//' -e 's/[,]*$//'
017 }
018
019 ######################################################################
020
021 #Token und API-URL anfordern
022 retVal=$(curl --silent https://api.backblazeb2.com/b2api/v2/
    b2_authorize_account -u "$ACCOUNT_ID:$APPLICATION_KEY")
023 authorizationToken=`echo $retVal | jsonValue authorizationToken`
024 apiUrl=`echo $retVal | jsonValue apiUrl`
025
026 ######################################################################
027
028 #Upload einer Datei
029 if [ $1 == "upload" ]; then
030     #Variablen setzen
031     FILENAME=$(basename $2)
032     FILEPATH=$(dirname $2)
033     MIME_TYPE=text/plain
034     SHA1_OF_FILE=$(openssl dgst -sha1 $FILENAME | awk '{print $2;}')
035
036     #Verzeichnis wechseln
037     cd $FILEPATH
038
039     #Upload-URL und Upload-Token anfordern
040     retVal=$(curl --silent  \
041     -H 'Authorization: '$authorizationToken \
042     -d '{"bucketId": "'$BUCKET_ID'"}' \
043     $apiUrl/b2api/v2/b2_get_upload_url )
044     UPLOAD_URL=`echo $retVal | jsonValue uploadUrl`
045     UPLOAD_AUTHORIZATION_TOKEN=`echo $retVal | jsonValue
      authorizationToken`
046
047     #Upload durchführen
048     curl --silent -o /dev/null \
049         -H "Authorization: $UPLOAD_AUTHORIZATION_TOKEN" \
050         -H "X-Bz-File-Name: $FILENAME" \
051         -H "Content-Type: $MIME_TYPE" \
052         -H "X-Bz-Content-Sha1: $SHA1_OF_FILE" \
```

```
053         -H "X-Bz-Info-Author: unknown" \
054         --data-binary "@$FILENAME" \
055         $UPLOAD_URL
056 fi
057
058 ################################################################################
059
060 #Dateiliste laden
061 if [ $1 == "get_onlinefile_list" ]; then
062     #Dateiliste abrufen
063     retVal=`curl --silent \
064         -H 'Authorization: '$authorizationToken \
065         -d '{"bucketId": "'$BUCKET_ID'"}' \
066         $apiUrl/b2api/v2/b2_list_file_names`
067
068     #Ausgabe aufbereiten
069     if [ $? -eq 0 ]; then
070         echo $retVal | jsonValue fileName
071     else
072         echo "error"
073     fi
074 fi
075
076 ################################################################################
077
078 #Datei herunterladen
079 if [ $1 == "download_file" ]; then
080     #Variablen setzen
081     FILE=$(basename $2)
082     DOWNLOAD_DIR=$3
083
084     #Datei herunterladen
085     curl --silent -H "Authorization: $authorizationToken" \
086         "${apiUrl}/file/${BUCKET_NAME}/${FILE}" > $DOWNLOAD_DIR/$FILE
087 fi
088
089 ################################################################################
090
091 #Datei löschen
092 if [ $1 == "delete" ]; then
093     #Variablen setzen
094     FILE_NAME=$(basename $2)
095
096     #Liste der Dateien laden um daraus die File-ID zu ermitteln
097     retVal=`curl --silent \
098     -H 'Authorization: '$authorizationToken \
099     -d '{"bucketId": "'$BUCKET_ID'"}' \
100     $apiUrl/b2api/v2/b2_list_file_names`
101
102     #File-ID ermitteln
103     resfileId=""
104     echo "$retVal" | while read line; do
105         tmpVal=$(echo $line | jsonValue fileId)
106         if [ ${#tmpVal} -ne 0 ]; then
107             fileId=$tmpVal
108         fi
109         tmpVal=$(echo $line | jsonValue fileName)
110         if [ ${#tmpVal} -ne 0 ]; then
111             if [ "$tmpVal" == "$FILE_NAME" ]; then
```

```
112                 #Wenn File-ID gefunden wurde das File löschen
113                 curl --silent \
114                 -H "Authorization: "$authorizationToken \
115                 -d '{"fileName": "'$FILE_NAME'", "fileId": "'$fileId'"}' \
116                 $apiUrl/b2api/v2/b2_delete_file_version 2>/dev/null
117             fi
118         fi
119     done
120 fi
121
122 ##########################################################################
```

Anhang E: Sourcecode „google_drive.sh“

```
001 #!/bin/bash
002 ######################################################################
003
004 #Benutzerparameter
005 client_id=      XXXXXXXXXXXXXXXXXXXXXXXXX
006 client_secret=  XXXXXXXXXXXXXXXXXXXXXXXXX
007 #Generierung über ./google_drive.sh authorize
008 refresh_token=  XXXXXXXXXXXXXXXXXXXXXXXXX
009
010 ######################################################################
011
012 #Wert eines JSON-Elements ermitteln
013 function jsonValue() {
014     KEY=$1
015     num=$2
016     awk -F"[,:}][^://]" '{for(i=1;i<=NF;i++){if($i~/\042'$KEY'\042/){print
        $(i+1)}}}' | tr -d '"' | sed -n ${num}p | sed -e 's/[}]*$//' -e
        's/^[[:space:]]*//' -e 's/[[:space:]]*$//' -e 's/[,]*$//'
017 }
018
019 ######################################################################
020
021 #Token anfordern
022 retVal=$(curl -d "client_id=$client_id&client_secret=$client_secret
    &refresh_token=$refresh_token&grant_type=refresh_token"
    https://accounts.google.com/o/oauth2/token 2>/dev/null)
023 access_token=`echo $retVal | jsonValue access_token`
024
025 ######################################################################
026
027 #Anwendung authorisieren
028 if [ $1 == "authorize" ]; then
029     #Device-Code abrufen
030     SCOPE=${SCOPE:-"https://docs.google.com/feeds"}
031     retVal=$(curl --silent "https://accounts.google.com/o/oauth2/
        device/code" --data "client_id=$client_id&scope=$SCOPE")
032     device_code=`echo $retVal | jsonValue device_code`
033
034     #Benutzer zu Interaktion im Webbrowser aufrufen
035     echo "Rufen Sie im Webbrowser folgende URL auf: "`echo $retVal |
        jsonValue verification_url`
036     echo "Geben Sie darin folgenden Code ein: "`echo $retVal | jsonValue
        user_code`
037     echo "Lassen Sie den Zugriff zu!"
038
039     #Prüfen ob die Kopplung erfolgreich getätigt wurde
040     while true; do
041         retVal=$(curl --silent "https://accounts.google.com/o/oauth2/
            token" -data "client_id=$client_id&client_secret=$client_secret
            &code=$device_code&grant_type=http://oauth.net/grant_type/device/
            1.0")
042         sleep 2;
043         refresh_token=`echo $retVal | jsonValue refresh_token`
044
045         if [ ${#refresh_token} -ne 0 ]; then
046             echo "Kopplung erfolgreich! Refresh-Token: "$refresh_token
```

```
047             exit 0;
048         fi
049     done
050 fi
051
052 ########################################################################
053
054 #Upload einer Datei
055 if [ $1 == "upload" ]; then
056     #Variablen setzen
057     FILE=$2
058     FOLDER_ID="root"
059     ACCESS_TOKEN=$access_token
060     SLUG=`basename "$FILE"`
061     FILENAME="${SLUG%.*}"
062     EXTENSION="${SLUG##*.}"
063     FILESIZE=$(stat -c%s "$FILE")
064     if [ "$FILENAME" == "$EXTENSION" -o ! "$(command -v mimetype)" ]
065     then
066             MIME_TYPE=`file --brief --mime-type "$FILE"`
067     else
068             MIME_TYPE=`mimetype --output-format %m  "$FILE"`
069     fi
070     postData="{\"mimeType\": \"$MIME_TYPE\",\"title\":
        \"$SLUG\",\"parents\": [{\"id\": \"$FOLDER_ID\"}]}"
071     postDataSize=$(echo $postData | wc -c)
072
073     #Upload-Link abrufen
074     uploadlink=`/usr/bin/curl \
075                 --silent \
076                 -X POST \
077                 -H "Host: www.googleapis.com" \
078                 -H "Authorization: Bearer ${ACCESS_TOKEN}" \
079                 -H "Content-Type: application/json; charset=UTF-8" \
080                 -H "X-Upload-Content-Type: $MIME_TYPE" \
081                 -H "X-Upload-Content-Length: $FILESIZE" \
082                 -d "$postData" \
083                 "https://www.googleapis.com/upload/drive/v2/files
                     ?uploadType=resumable" \
084                 --dump-header - | sed -ne s/"Location: "//pi | tr -d
                    '\r\n'`
085
086     #Upload durchführen
087     curl --silent \
088     -X PUT \
089     -H "Authorization: Bearer ${ACCESS_TOKEN}" \
090     -H "Content-Type: $MIME_TYPE" \
091     -H "Content-Length: $FILESIZE" \
092     -H "Slug: $SLUG" \
093     --upload-file "$FILE" \
094     --output /dev/null \
095     "$uploadlink"
096 fi
097
098 ########################################################################
099
100 #Dateiliste laden
101 if [ $1 == "get_onlinefile_list" ]; then
102     #Dateiliste abrufen
```

```
103     retVal=`curl --silent \
104     -X GET \
105     -H "Authorization: Bearer ${access_token}" \
106     "https://www.googleapis.com/drive/v2/files/?q=trashed=false"`
107 
108     #Ausgabe aufbereiten
109     if [ $? -eq 0 ]; then
110         echo $retVal | jsonValue title
111     else
112         echo "error"
113     fi
114 fi
115 
116 ########################################################################
117 
118 #Datei herunterladen
119 if [ $1 == "download_file" ]; then
120     #Variablen setzen
121     FILE=$2
122     DIR_METADATA_DOWNLOAD=$3
123 
124     #File-ID ermitteln
125     retVal=`curl --silent \
126     -X GET \
127     -H "Authorization: Bearer ${access_token}" \
128     "https://www.googleapis.com/drive/v2/files?q=title+contains
        +'$FILE'+AND+trashed=false&fields=items%2Fid"`
129     ID=`echo $retVal | jsonValue id`
130 
131     #Datei herunterladen
132     curl --silent \
133     -X GET \
134     -H "Authorization: Bearer ${access_token}" \
135     "https://www.googleapis.com/drive/v2/files/$ID?alt=media" --output
         $DIR_METADATA_DOWNLOAD/$FILE
136 fi
137 
138 ########################################################################
139 
140 #Datei löschen
141 if [ $1 == "delete" ]; then
142     #Variablen setzen
143     FILE=$2
144 
145     #File-ID ermitteln
146     retVal=`curl --silent \
147     -X GET \
148     -H "Authorization: Bearer ${access_token}" \
149     "https://www.googleapis.com/drive/v2/files?q=title+contains
        +'$FILE'+AND+trashed=false&fields=items%2Fid"`
150     ID=`echo $retVal | jsonValue id`
151 
152     #Datei herunterladen
153     curl --silent \
154     -X DELETE \
155     -H "Authorization: Bearer ${access_token}" \
156     "https://www.googleapis.com/drive/v2/files/$ID"
157 fi
158 ########################################################################
```