A=SIT Technology
E-Government Innovationszentrum

2. November 2018 Studie · Blockchain · Transparenz · 22 Min. Lesezeit

Dezentrale Datenspeicherung

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Stand 2018-11-01
Quelle https://technology2.a-sit.at/veroeffentlichungen/dezentrale-datenspeicherung/

Dezentrales Speichern von Daten ist ein komplexes Thema, vor allem im Hinblick auf Authentizität und Integrität der Daten. Lokale Speicherung legt die Kontrolle über die Daten einerseits komplett in die Hände des Benutzers, garantiert aber per se nicht die Integrität der Daten. Außerdem sind zusätzliche Maßnahmen erforderlich, um die Verfügbarkeit der Daten zu garantieren. Auf mobilen Geräten mit limitiertem Speicherplatz werden auch cloudbasierte Datenspeicher zunehmend für die Auslagerung sensibler Daten verwendet. Dadurch ist zwar keine eigene Infrastruktur nötig, es wird aber die Kontrolle über die eigenen Daten abgegeben. Das dezentrale Speichern von Daten kann hierbei Abhilfe schaffen. Im Rahmen dieses Projekt wurden Möglichkeiten der dezentralen Datenspeicherung mit besonderem Schwerpunkt auf der Blockchain-Technologie untersucht. Im Kern steht die Frage, wie Datenblöcke auf unterschiedliche Knoten verteilt werden müssen, um einerseits Datenverlust zu verhindern, aber trotzdem die grundlegende Eigenschaften Authentizität und Integrität der Blockchain-Technologie zu erhalten.

1. Einleitung

Cloudspeicherdienste sind mittlerweile etabliert und der Kontrollverlust über die eigenen Daten, sowie die damit einhergehenden Datenschutzbedenken werden für kostenlosen, bzw. kostengünstigen und wartungsfreien Speicherplatz in Kauf genommen. Gleichzeitig gibt es jedoch mehrere Gegentrends und Bewegungen, die eine Dezentralisierung im großen Stil anstreben. Beispiele hierfür sind jährliche Veranstaltungen wie der *Decentralized Web Summit* [1], welcher unter anderem 1 https://decentralizedweb.net/ © A-SIT, Partner für Sicherheitsfragen von Tim Berners Lee organisiert wurde, zunehmendes Interesse an Peer-to-Peer-Architekturen, sowie Entwicklungen aus dem Blockchain-Umfeld. Der technische Fortschritt und die zunehmende Verfügbarkeit von Trusted-Computing-Mechanismen, sowie die Erfahrungen aus dem Blockchain-Bereich legen den Schluss nahe, dass dezentrale Datenspeicherung, verglichen mit traditionellen Cloudspeicherdiensten, zumindest mit weniger Kontrollverlust und Datenschutzbedenken einhergehen kann. Inwieweit dies tatsächlich zutrifft, wird in diesem Dokument beleuchtet. Abschnitt 2 beschreibt grundlegende Konzepte und Hintergründe zu verteilten und dezentralen Datenspeichern. Ausgehend davon werden in Abschnitt 3 traditionelle (auch

historisch relevante) Konzepte zur verteilen Datenspeicherung wie verteilte Hashtabellen (*Distributed Hash Tables*) und Erfahrungen aus dem Filesharing-Umfeld diskutiert. Darauf aufbauend werden in Abschnitt 4 Blockchain-basierte Konzepte vorgestellt, welche es ermöglichen sollen, Daten dezentral zu speichern, ohne dass anderen Parteien vertraut werden muss. Abschnitt 5 fasst die Ergebnisse zusammen und stellt die unterschiedlichen Ansätze einander gegenüber.

2. Hintergrund

Grundsätzlich gibt es zwei Möglichkeiten, Daten auf mehrere Geräte (und damit dezentral) zu verteilen: Entweder Dateien werden gezielt auf (manuell) ausgewählte Geräte verteilt, oder die gesamte zu verteilende Datenbasis wird in Blöcke aufgeteilt, welche anschließend automatisiert auf die Systeme verteilt werden. Letzteres eignet sich für die Umsetzung verteilter Datenspeicher im eigentlichen Sinn, während Ansätze auf Dateiebene eher im Rahmen von Dateisynchronisationslösungen zur Replikation eingesetzt werden. Hierbei wird lediglich die Verfügbarkeit erhöht, eine Aufteilung eines gemeinsamen Datensatzes auf mehrere Teilnehmer erfolgt nicht. Dies ist insbesondere im Zusammenhang mit Ausfallsicherheit relevant. Nachfolgend werden Anforderungen an dezentrale Datenspeicher dargelegt, auf Basis derer unterschiedliche Lösungsmöglichkeiten Ansätze zur Umsetzung entsprechender Speicherkonzepte diskutiert werden.

2.1. Definitionen und Anforderungen an dezentrale, verteilte Datenspeicher

Im Kern dieses Dokuments werden dezentrale, verteilte Datenspeicherungsmodelle. Bisher existiert jedoch keine einheitliche Definition solcher Systeme. Allerdings lassen sich Anforderungen an dezentrale Datenspeicher definieren, an Hand derer sich eine Definition ableiten lässt. Ausgangspunkt hierfür ist die Definition eines *decentralized storage network* (dezentrales Speichernetzwerk) [1], sowie bekannte Eigenschaften von dezentralen Datenspeichern in Form verteilter Hashtabellen

[2] [3] [4] [5]. In jedem Fall handelt es sich bei der im Rahmen dieses Dokuments diskutierten Systeme um ein Netzwerk, das von seinen Nutzern aufgespannt und bereitgestellt wird (idR. um ein Peer-to-Peer-Netzwerk). Im Speziellen wird der Schwerpunkt auf eine tatsächlich dezentrale Organisation des Datenspeichers gelegt, d.h. es gibt keine zentrale Instanz, welche die Koordination übernimmt, oder Korrektheit des Systems garantiert. Stattdessen sollte ein dezentraler Datenspeicher idealerweise Mechanismen bereitstellen, die korrektes, ausfallsicheres Speichern und Bereitstellen von Daten nachweisbar machen, auch wenn kein Vertrauensverhältnis zwischen einzelnen Teilnehmerinnen und Teilnehmern besteht. Zusammenfassend ergeben sich damit folgende Anforderungen an einen dezentralen, verteilten Datenspeicher:

1. Vollständige Dezentralisierung: Zu keinem Zeitpunkt (z.B. im Zuge des Beitritts in das Datenspeicher-Netzwerk, des Verbindungsaufbaus, der Suche nach Daten, oder der Auswahl eines geeigneten Speicherorts) und auf keiner Ebene soll eine zentrale Instanz involviert sein.

2. Datenverteilung: Daten sollen unabhängig von ihrer Größe verteilt gespeichert werden können. Größere Datenmengen sind entsprechend zu partitionieren.

3. Verfügbarkeit: Daten sollen derart redundant bzw. repliziert im Datenspeichernetzwerk verteilt werden, dass eine den Anforderungen unterschiedlicher Anwendungsfälle entsprechend hohe Verfügbarkeit sichergestellt werden kann, auch wenn Teilnehmer das Netzwerk unerwartet

verlassen.

4. Effizienz: Der Datenspeicher soll den im Netzwerk vorhandenen Speicherplatz möglichst effizient nutzen und nach Möglichkeit Deduplikationsverfahen anwenden.

5. Integrität: Die Integrität einmal gespeicherter Daten, soll bei deren Abruf zumindest überprüft, möglichst jedoch garantiert werden können. Im Speziellen soll es nicht möglich sein, auch nur Teile von gespeicherten Daten zu ersetzen oder anderweitig zu korrumpieren.

6. Vertraulichkeit: Daten sollen nur berechtigten Teilnehmern zugänglich gemacht werden.

Tatsächlich ist bereits die Erfüllung der ersten Eigenschaft unter realistischen Bedingungen kaum möglich. Beispielswiese sind Blockchain-basierte Konzepte per Definition insofern zentralisiert, als dass eine gemeinsame zentrale Datenbasis (welche dezentral verwaltet wird) verwendet wird. Darüber hinaus ergibt sich auch immer ein Wechselspiel zwischen Verfügbarkeit und Effizienz; hohe Verfügbarkeit geht mit hohem Replikationsgrad einher und Speichereffizienz ist de facto nur auf Kosten von Verfügbarkeit umsetzbar. Eine Möglichkeit, beide Kriterien bestmöglich zu erfüllen, ergibt sich durch bestmögliche Ausnutzung der Gesamtspeicherkapazität des Netzwerks, indem ungenutzter Speicher für Replikation herangezogen wird. Weitere Details hierzu sind den Abschnitten 2.3 und 2.4 zu entnehmen. Die Vertraulichkeit von Daten kann im Wesentlichen durch den Einsatz von Verschlüsselung garantiert werden. Die angewandten Verfahren sind allgemeiner Natur und werden daher im Rahmen dieses Dokuments nicht näher erörtert. Datenintegrität kann ebenfalls auf unterschiedliche Weise garantiert, werden. Allerdings liegt im Zusammenhang mit Partitionierung und anschließender Verteilung von Datensegmenten auf mehrere Teilnehmer der Fokus auf der erneuten Zusammensetzung der Daten und anschließender Verifizierbarkeit. Insbesondere soll sichergestellt werden, dass es sich bei einer aus Segmenten zusammengesetzten Datei auch tatsächlich um die ursprünglich partitionierte Datei handelt. Hier sind insbesondere Hash-Bäume hervorzuheben, welche nachfolgend beschrieben werden.

2.2. Verfahren zur Integritätssicherung und Verifikation verteilter Daten

Um effektiv große Datenmengen in einem Netzwerk verteilen zu können, müssen diese unweigerlich partitioniert werden (unter anderem damit eine Verteilung unabhängig von Dateigrößen erfolgen kann, und um die Ausfallsicherheit zu erhöhen). Die Identifikation von Daten und Integritätsprüfung kann prinzipiell durch den Einsatz kryptografischer Hashfunktionen erfolgen. Wird jedoch naiv vorgegangen und direkt der Hashwert von Dateien berechnet, welche anschließend partitioniert und verteilt werden, kann beim erneuten Datenabruf lediglich festgestellt werden ob die Daten verändert wurden oder nicht. Eine Eingrenzung der Fehlerquelle oder Nachvollziehbarkeit der Modifikation ist nicht möglich. Dieser Umstand ist insbesondere kritisch, wenn eine Verteilung von Datensegmenten auf unbekannte Parteien, bzw. Teilnehmerinnen und Teilnehmer erfolgt, zu denen kein Vertrauensverhältnis besteht. In solchen Fällen muss eine Lokalisierung von Integritätsverletzungen möglich sein und der oder die Verantwortliche(n) müssen ausgemacht werden können. Hierfür haben sich *Hash-Bäume* (auch *Merkle-Bäume*) etabliert.

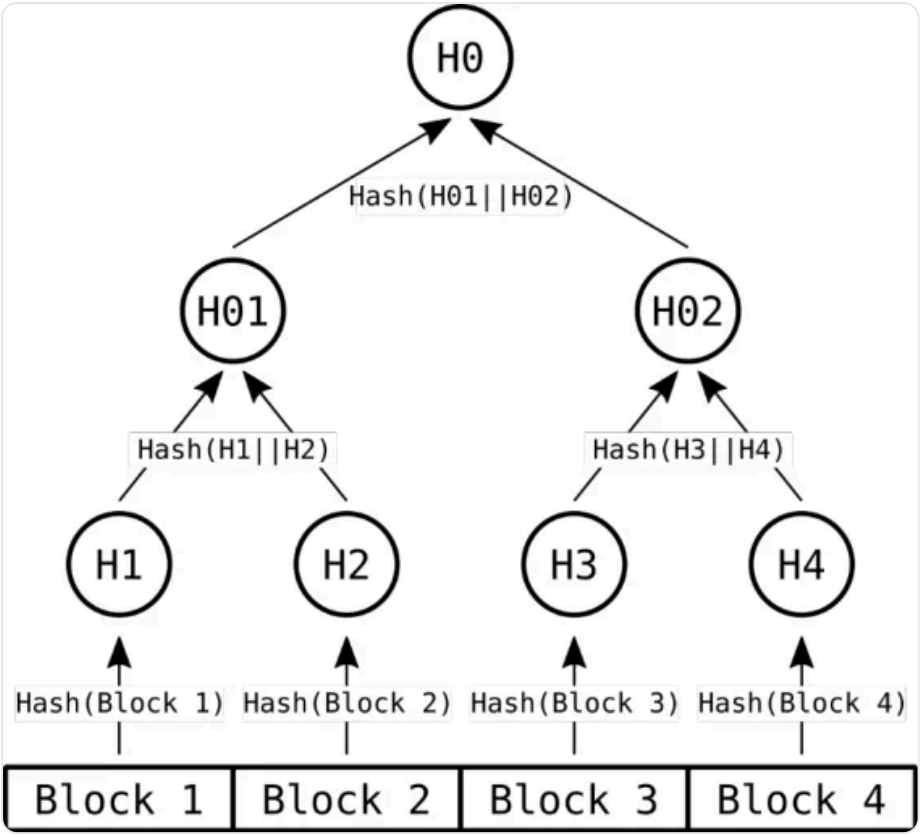


Abbildung 1: Funktionsweise und Struktur eines Hash-Baums (Merkle-Baum) Ein Hash-Baum ist eine Datenstruktur, welche eine effiziente Integritätsprüfung partitioniert abgerufener Dateien ermöglicht. Ursprünglich diente die von Ralph Merkle 1979 erdachte Struktur dem Schlüsselmanagement im Rahmen von Lamport-Einmalsignaturen [6]. Im Kontext dezentraler Datenspeicher wurden Hash-Bäume bereits in der Blütezeit des Filesharing erfolgreich eingesetzt um zu verifizieren, ob eine Datei, deren Segmente aus verschiedenen (üblicherweise nicht vertrauenswürdigen Quellen) bezogen wurden, auch integer ist, und wenn nicht, welche(s) Segmente(e) korrumpiert wurden. Die Funktionsweise ist folgende:

- Eine Datei wird zuerst partitioniert, und dann verteilt gespeichert, bzw. im Kontext von Filesharing anderen Teilnehmern segmentweise zur Verfügung gestellt.
- Im Zuge der Partitionierung wird von jedem Block ein kryptografischer Hashwert berechnet.
- Anschließend werden (im Falle eines binären Hash-Baums) immer zwei Hashwerte konkateniert und erneut gehasht, wodurch sich eine Baumstruktur ergibt. Der Wurzelknoten des Baums fungiert als eindeutiger Identifikator der Datei.

Ist der Identifikator bekannt, kann die Vollständigkeit und Integrität einer zuerst partitionierten und anschließend wieder zusammengesetzten Datei durch die Rekonstruktion des Hash-Baums und anschließenden Vergleich des Wurzelknotens sichergestellt werden. Je nachdem wie detailliert die Struktur (bzw. die Werte der Blätter) eines von einer partitionierten Datei erzeugten Hash-Baums im Rahmen der Verifikation bekannt ist, umso exakter können eventuelle Integritätsverletzungen lokalisiert werden. Die Struktur eines Hash-Baums ermöglicht es, Baumknoten nach Bedarf abzufragen, um Fehler (wie zuvor beschreiben) lokalisieren zu können. Dabei ist es auf Grund der Eigenschaften kryptografischer Hashfunktionen nicht notwendig, der Quelle, aus der die Baumknoten bezogen wurden, zu vertrauen, da

Falschinformationen sofort erkannt werden können. Durch die Wahl einer kleineren Blockgröße können Fehler noch exakter eingegrenzt werden. Allerdings steigt der Overhead bei kleiner werdender Blockgröße. Abbildung 1 die Struktur und Funktionsweise eines Hash-Baums grafisch dar.

Hash-Bäume eignen sich auch unter bestimmten Voraussetzungen für Deduplikationsverfahren. Der nachfolgende Abschnitt geht näher auf diese Thematik ein.

2.3. Deduplikationsverfahren

Um den vorhandenen Speicherplatz möglichst effizient nutzen zu können, bietet es sich an, Daten, die vielfach nachgefragt sind, nur einmal (und dafür entsprechend repliziert) zu speichern. Derartige Verfahren arbeiten üblicherweise auf Block-Ebene: Daten werden partitioniert und Prüfsummen, bzw. Identifikatoren (üblicherweise kryptografische Hashwerte) für jeden Block berechnet. Anschließend wird im gesamten Netzwerk, bzw. innerhalb der gesamten Datenbasis nach Duplikaten von Blöcken gesucht. Die Effizienz dieser Verfahren ist stark von der Struktur der zu Grunde liegenden Daten und deren Änderungsrate abhängig. Werden Änderungen (im Sinne der Nachvollziehbarkeit und um Rollback-Mechanismen zur Verfügung zu stellen) inkrementell gespeichert (d.h. Änderungen an Daten werden lediglich als solche an die Daten angefügt und die Ausgangsdaten bleiben unverändert), kann dieser Ansatz effizient arbeiten und so den Overhead inkrementeller Änderungen teilweise kompensieren. Verfahren auf Blockebene haben die Eigenschaft der *content-addressability* . Daten werden nicht über Dateinamen identifiziert und adressiert, sondern an Hand ihres Inhalts, bzw. an Hand von vom Inhalt abgeleiteten Identifikatoren. Konkret fungieren in diesem Kontext kryptografische Hashwerte als Identifikator für den Dateninhalt.

Hauptsächlich in zentralisieren Szenarien, (wie z.B. serverbasierte Backup-Lösungen) kommen so genannte *Singe-Instance* Speicherverfahren zum Einsatz, welche mehrfach vorhandene Daten tatsächlich auf eine einzige gespeicherte Instanz zusammenführen. Besonders bei zentral verwalteten Arbeitsplätzen, deren Konfigurationen und Datenbestand sich in weiten Teilen überschneiden, bewähren sich diese Verfahren.

Allen Ansätzen ist gemein, dass eine Mustererkennung möglich sein muss, um Daten derart partitionieren zu können, dass sich wiederholende Muster auch auf getrennte Blöcke aufgeteilt werden. Dies steht jedoch praktisch im Widerspruch zum Einsatz von Verschlüsselung, wenn die Vertraulichkeit verteilter Daten gewahrt werden soll. Im Kontext verteilter Anwendungen mit wenigen, bzw. keinen Ansprüchen an Vertrauensverhältnisse zwischen Teilnehmern, bzw. Teilnehmerinnen stellt dies eine Herausforderung dar. Dies ist insofern ein nicht zu vernachlässigender Aspekt, als dass, besonders in solchen Szenarien, Verfügbarkeit garantiert werden muss. Dies geht immer mit redundanter Datenspeicherung und damit erhöhtem Speicherplatzbedarf einher. Der nachfolgende Abschnitt befasst sich näher mit diesem Thema.

2.4. Verfahren zur redundanten, verteilten Datenspeicherung

Im Rahmen dezentralisierter Datenspeicher, welche von voneinander unabhängigen Parteien aufgespannt werden, nimmt Ausfallsicherheit, bzw. Verfügbarkeit einen besonderen Stellenwert ein. Ohne zentrale Instanz steht es allen im Netzwerk frei, dieses jederzeit zu verlassen. Ohne Sanktionsmaßnahmen oder Anreize, Daten anderer zu speichern, können diese schlicht gelöscht werden. Unabhängig von den Gründen für Ausfälle, müssen Kompensationsmechanismen vorhanden sein. Im einfachsten Fall werden Daten(blöcke) repliziert, bzw. redundant gespeichert. Die Verfügbarkeit einer Datei ist dadurch nur so hoch wie die Verfügbarkeit des am wenigsten verfügbaren Datenblocks. Eine Abschätzung des

Replikationsfaktors kann sich in diesem Zusammenhang als durchaus schwierig erweisen, da Wissen über das Verhalten von Netzwerkteilnehmerinnen und Netzwerkteilnehmern vorliegen muss. Dieses Wissen ist insbesondere in frühen Entwicklungs- bzw. Deploymentphasen nicht vorhanden und empirische Daten bestehender Netzwerke lassen sich auf neue Ansätze oftmals nur bedingt umlegen. Eine Möglichkeit, die Verfügbarkeit zu erhöhen, sind *Reed-Solomon-Codes* [7]. Dabei handelt es sich um einen Erasure-Code, welcher es erlaubt, n Segmente auf $n+ k$ Elemente zu kodieren. Der Vorteil gegenüber primitiver Replikation, ist, dass bis lediglich n viele Elemente aus dieser $n+ k$ großen Menge verfügbar sein müssen, um alle n ursprünglichen Segmente rekonstruieren zu können. Der Ausfall einzelner Knoten, welche Teile einer Datei speichern, kann dadurch kompensiert werden. In Kombination mit Replikation ergibt sich in dezentralen Szenarien ein hoher Grad an Ausfallsicherheit.

Wie ein dezentrales Netzwerk aufgebaut, betrieben und für dezentrales Datenspeicherung verwendet werden kann, wird nachfolgend beschrieben. Im Zuge dessen werden Grundlagen von Peer-to-Peer-Netzen und dezentraler Speicher auch im Kontext von Filesharing diskutiert.

3. Distributed Hash Tables und Filesharing

Der Unterbau eines dezentral organisierten Datenspeichers ist übelicherweise ein Peer-to-Peer-Netzwerk. Dabei handelt es sich um ein logisches Overlay, welches die Struktur des darunterliegenden IP-Netzwerks verschleiert und stattdessen einen flachen Identifikatorraum aufspannt. Um Verbindungen zu anderen Teilnehmern und Teilnehmerinnen aufbauen zu können, muss der einem Teilnehmer, bzw. einer Teilnehmerin zugeordneten Identifikator wieder auf eine IP-Adresse aufgelöst werden. Hierfür kommen spezielle Overlay-Routingprotokolle zum Einsatz, welche ebenfalls ohne eine zentrale Instanz auskommen. Ausgehend von einer (wie auch immer gearteten) Distanzfunktion, wird eine Anfrage nach der IP-Adresse eines Zielknotens an bekannte Knoten ausgesendet. Diese antworten mit der Untermenge ihnen bekannter Knoten, welche am nächsten am Zielknoten liegen. Dieser Vorgang wird so oft wiederholt, bis der gesuchte Knoten lokalisiert ist, oder Gewissheit herrscht, dass dieser aktuell nicht Teil des Netzwerks ist.

Ausgehend von einem logischen Overlay-Netzwerk mit flachem Identifikatorraum lässt sich ein dezentraler, verteilter Datenspeicher mittels Content-Addresability umsetzen: Wenn Daten und Netzwerkknoten denselben Identifikatorraum okkupieren, werden Daten am ihrem Identifikator nächstgelegenen Knoten gespeichert. Üblicherweise werden Datenidentifikatoren durch die Anwendung einer kryptografischen Hashfunktion berechnet. Aus diesem Grund werden verteilte Datenspeicher, welche auf diesem Prinzip beruhen, als *Distributed Hash Table* (verteilte Hashtabelle) bezeichnet. Ausführlichere Informationen zum Thema Peer-to-Peer-Netze und Distributed Hash Tables können vorangegangenen A-SIT-Projekten [8] [9] entnommen werden. Gängige Peer-to-Peer-Netzwerkdesigns verfügen über integrierte Replikationsmechanismen; im einfachsten Fall werden Daten nicht nur am nächstgelegenen Knoten gespeichert sondern an den k nächsten Knoten (wobei k den Replikationsfaktor beschreibt).

Eine Sonderstellung nehmen Filesharing-Systeme wie *BitTorrent* [10] ein. Diese dienen der Verteilung von Daten an Unbekannte, sowie dem Abruf von Daten aus unbekannten Quellen. Einen entsprechend hohen Stellenwert nehmen Integritätssicherungsmaßnahmen ein. Gleichzeitig stellt sich die Frage nach Verfügbarkeit, bzw. Replikation nicht, da *populäre* Daten im Netzwerk geteilt werden. Typischerweise regelt die Nachfrage das Angebot und nicht mehr nachgefragte Daten verschwinden nach und nach aus dem Netzwerk. Deduplikation ist ebenfalls Großteils irrelevant, da jeder Teilnehmer, welcher Interesse an Daten hat, diese auch lokal gespeichert haben möchte. Im Rahmen früher Distributed Hash Tables war auch

Deduplikation effizient umsetzbar, da Daten im Klartext abgelegt wurden. Dasselbe gilt für Filesharing-Systeme. Sobald jedoch Vertraulichkeit gefragt ist, und Daten verschlüsselt gespeichert werden sollen, stoßen Deduplikationsmaßnahmen an ihre Grenzen.

Im Rahmen von Filesharing-Szenarien lässt sich noch verhältnismäßig glaubwürdig argumentieren, dass sich Teilnehmer ehrlich verhalten, bzw. kann das Geben und Nehmen im Rahmen einfacher Reputationssysteme (Teilnehmer *X* muss mindestens k viele Blöcke bereitstellen, um seinerseits r viele Blöcke von anderen Teilnehmern abrufen zu können) geregelt werden. Auch zeigt sich am Beispiel BitTorrent, dass derartige Systeme keine besonders hohen Sicherheitsmargen bieten müssen, da das Interesse der Teilnehmer, diese zu überwinden, beschränkt ist, nachdem alle ein gemeinsames Ziel verfolgen. Im Rahmen dezentraler Speichersysteme, welche als Alternative zu Cloudspeicherdiensten eingesetzt werden sollen, hält diese Argumentation nicht, da es an sich keinen Anreiz gibt, anderen den eigenen Speicherplatz zur Verfügung zu stellen.

Auch ein Vergütungssystem für das Bereitstellen von Speicherplatz alleine reicht nicht aus, um die Verfügbarkeit von Daten und ehrliches Verhalten zu gewährleisten. Ohne eine Möglichkeit, die Speicherung von Daten tatsächlich zu verifizieren, könnten Teilnehmerinnen und Teilnehmer schlichtweg vorgeben, Daten zu speichern, tatsächlich aber keinen Speicherplatz zur Verfügung stellen. Eine Entlohnung nach Wiederabruf von Daten, um die Speicherung zu gewährleisten ist auch nicht zielführend, da Angreifer dieses System dahingehen ausnutzen können, vorhandenen Speicherplatz zu füllen, ohne Daten abzurufen. Nachfolgend beschriebene Blockchain-basierte Systeme versuchen hier Abhilfe zu schaffen.

4. Blockchain-basierte verteilte Datenspeicherkonzepte

Distributed Ledger Technology, gemeinhin als *Blockchain* bezeichnet, beschreibt die zentralen Transaktionsregister moderner, dezentral konzeptionierter Kryptowährungen wie *Bitcoin* [11]. Dabei handelt es sich um eine Liste kryptografisch verketteter Datenblöcken welche – im Rahmen von Kryptowährungen – Transaktionen beinhalten. Um neue Blöcke erstellen zu können, muss im einfachsten Fall die Menge an Transaktionen, die in einen Block aufgenommen werden soll, sowie ein frei wählbarer Wert (die so genannte *Nonce*) gehasht werden. Die Schwierigkeit dieses Unterfangens besteht darin, dass der Hashwert eines Blocks ein festgelegtes Präfix bestimmter Länge aufweisen muss. Dies kann (auf Grund der Eigenschaften kryptografischer Hashfunktionen) nur mittels „Durchprobieren“ einer Vielzahl von Noncewerten erreicht werden, und benötigt entsprechend viel Rechenleistung. Der Rest des dezentral organisierten Kryptowährungsnetzwerk kann entsprechend erzeugte Blöcke hingegen effizient verifizieren, wodurch ein Konsens erreicht wird. Es ergibt sich somit eine zentrale, gemeinsame Datenbasis im Netzwerk, welche dezentral fortgeführt wird, nachweisbar integer ist und deren Aktualisierung keinerlei Vertrauensverhältnis zwischen den beteiligten Parteien erfordert. Entsprechend der aufzuwendenden Rechenleistung werden solche Konsensfindungsmethoden als *Proof-of-Work* bezeichnet. Die Motivation hinter der Erstellung von Blöcken liegt in der damit verbundenen Ausschüttung von Kryptowährungstoken an den Blockersteller. Weitere Details zum Thema Blockchain können einer A-SIT-Studie zum Thema entnommen werden [12].

Viele Eigenschaften von Blockchain-basierten Systeme eigenen sich, um dezentrale Datenspeicher umzusetzen, bzw. um Anreize und Sanktionsmechanismen zu schaffen, welche ehrliches Verhalten fördern und unehrliches Verhalten mit (finanziellen) Verlusten koppeln. Dies wird nachfolgend an Hand von drei konkreten Beispielen illustriert.

4.1. Storj

Storj [2] ist ein auf dem *Kademlia* [5], bzw. *S/Kademlia* [13] basiertes Peer-to-Peer-Netzwerk, das ein dezentrales Speichernetzwerk zur Verfügung stellt. Das Projekt verwendet die Kryptowährung *Ethereum* [3], um Teilnehmer mit *STORJ* -Token für das Bereitstellen von Speicherplatz zu belohnen. Bei STORJ-Token handelt es sich um so genannte *ERC20* -Token. ERC20-Token beschreiben einen standardisierten Mechanismus, um eigene Token auf Basis der Ethereum-Kryptowährung zu erstellen. Dabei handelt es sich im Prinzip um speziell markierte Ethereum-Token, deren Markierung (außerhalb des Kryptowährungsnetzwerks) eine bestimmte Semantik zugeordnet werden kann. Weitere Einsatzgebiete der Blockchaintechnologie zur Integritätssicherung oder als Anreizsystem für korrektes Verhalten sind im Storj-Whitepaper [14] nicht beschreiben. Nichts desto trotz konnte Storj über mehrere Finanzierungsrunden bis Juni 2017 über 30 Mio USD an Kapital zur Umsetzung des Gesamtkonzepts lukrieren [4,5] . Anstatt auf Blockchain-basierte Verfahren, setzt Storj ein so genanntes *Sharding* -Verfahren ein, um Verfügbarkeit zu garantieren, bzw. Speicherplatzanbieter insofern in die Pflicht zu nehmen, als dass diese beweisen müssen, Daten auch tatsächlich zu speichern. Dabei handelt es sich um ein Verfahren basierend auf Hash-Bäumen (siehe Abbildung 2):

1. Daten werden verschlüsselt, um Vertraulichkeit zu wahren. Die Schlüsselverwaltung obliegt dem Dateneigentümer, bzw. der Dateneigentümerin.
2. Daten werden wir in Abschnitt 2.2 beschrieben partitioniert.

2 https://storj.io/ 3 https://www.ethereum.org/ 4 https://www.coindesk.com/blockchain-startup-storj-targets-enterprise-cloud-3-million-raise/ 5 https://www.forbes.com/sites/forbesfinancecouncil/2017/11/21/what-to-consider-in-an-ico/#543f81855c44 3. Bevor ein Hash-Baum konstruiert wird, wird jedem Datensegment ein (vorerst nur dem Eigentümer, bzw. der Eigentümerin der Daten bekannter) nur einmal verwendbarer (Zufalls)wert, eine so genannte kryptografische Nonce, vorangestellt. 4. Jedes um die kryptografische Nonce erweiterte Segment wird gehasht 5. Immer jeweils zwei Hashwerte werden erneut gehasht. Dieser Vorgang wird wiederholt, um einen Hash-Baum aufzubauen. 6. Datensegmente werden (exklusive der kryptografischen Nonce) verteilt. 7. Zu jedem beliebigen Zeitpunkt können Dateneigentümer, bzw. Dateneigentümerinnen, diejenigen Parteien, welche ihre Daten speichern, auffordern einen Teilbereich des Hash-Baums zu rekonstruieren und zu übermitteln, indem ihnen eine oder mehrere kryptografische Nonces übermittelt werden. Da sich erst aus der Kombination von Datensegment und kryptografischer Nonce die entsprechenden Hashwerte ableiten lassen, um Teile des ursprünglichen Hash-Baums zu rekonstruieren, beweist eine erfolgreiche Rekonstruktion das Vorhandensein der betroffenen Datensegmente. Derartige Beweise sind per Definition nur einmalig pro Datensegment sinnvoll, da ein einmal berechneter Hashwert gespeichert werden kann, sodass eine unehrliche Partei die entsprechenden Datensegmente anschließen löschen kann, und den Beweis trotzdem erneut antreten kann.

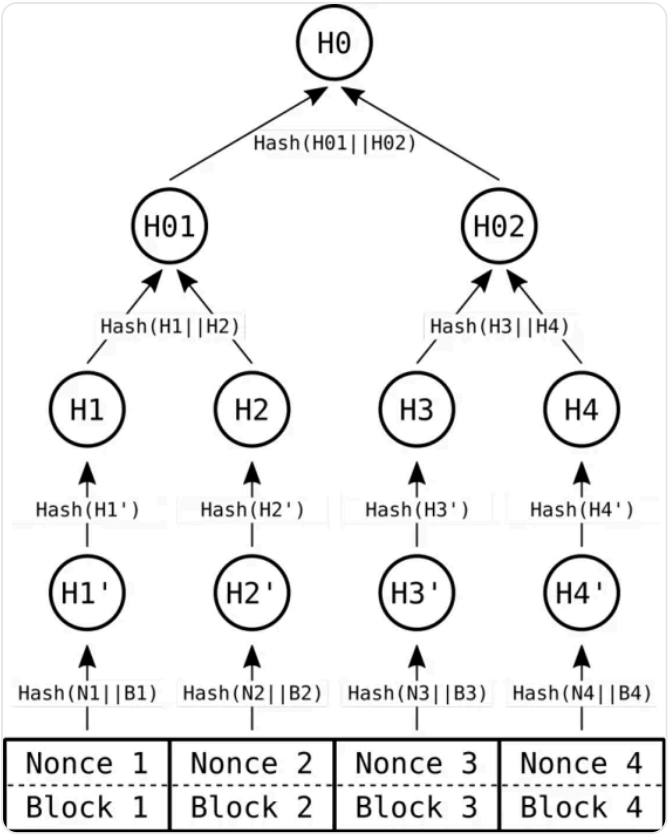


Abbildung 2: Von Stor eingesetztes Hash-Baum-Verfahren Storj überlasst sowohl die Schlüsselverwaltung als auch die Wahl der Verschlüsselungsalgorithmen, sowie den Einsatz von Replikations- bzw. Redundanzverfahren den Teilnehmern und Teilnehmerinnen. Das Zusammenspiel zwischen Peer-to-Peer-Netzwerk, Sharding, und der Blockchain-Technologie geht aus dem Storj-Whitepaper nicht klar hervor. Aus diesem Grund fällt einer Bewertung des Konzepts schwer, wodurch sich keine gesicherten Schlüsse über die Tauglichkeit des Systems für dezentrale Datenspeicherung im Produktiveinsatz ziehen lassen. Ungeachtet dessen scheint Storj stabil am Markt positioniert zu sein.

4.2. Sia

Sia [6] ist wird als dezentrale Cloudspeicherplattform positioniert. Im Kern steht laut offiziellem Whitepaper [15] eine stark an die Bitcoin-Blockchain angelehnte Blockchain. Tatsächlich liegt das Hauptaugenmerk von Sia auf der Verknüpfung eines Marktplatzes für Cloud-Speicher mit der Blockchain-Technologie. Im Wesentlichen können Hoster Speicherplatz anbieten, und völlig frei über den Preis für diesen Speicherplatz entscheiden. Dasselbe gilt für Teilnehmerinnen und Teilnehmer, welche Speicherplatz benötigen. Entscheidend ist in diesem Zusammenhang der Einsatz der Blockchain. Die Zahlungsabwicklung erfolgt im Prinzip analog zu Bitcoin: Eine Transaktion besteht aus so genannten *Inputs* und *Outputs*, welche definieren, von wem an wen welche Menge an Kryptowährungstoken übertragen werden sollen, sowie Freigabebedingungen, welche festlegen, unter welchen Umständen eine Zahlungsabwicklung erfolgt.

Die Neuerung im Rahmen von Sia liegt in der Definition von Freigabebedingungen: Speichert ein Hoster Datensegmente, kann – vereinfacht formuliert – der Hash dieser Daten dazu genutzt werden, periodisch die Freigabebedingung von an diese Daten geknüpfte Transaktionen zu erfüllen, wodurch es zu einer Auszahlung kommt. Hierfür kommen, wie auch im Rahmen von Storj, Hash-Bäume zum Einsatz. Da zufällig ausgewählt wird, welche Segmente für diese Aktion herangezogen werden, muss zumindest anfangs tatsächlich die gesamte Datei gespeichert werden. Für das Vorweisen von Hashwerten gibt es definierte Zeitfenster, innerhalb derer ein Hoster den Hash von gespeicherten Daten vorweisen kann. Im Rahmen

einer Transaktion ist auch definiert, wie oft, bzw. über welchen Zeitraum dies stattfinden kann, und wie oft ein Hoster einen Nachweis schuldig bleiben darf. Weiters lassen sich Transaktionen so konstruieren, dass nach Ablauf aller Zeitfenster eine zusätzliche Prämie für das kontinuierliche Speichern von Daten freigegeben wird.

Die zuvor beschriebenen Transaktionen beinhalten per se noch kein Reglement, welches die Übertragung von zuvor gespeicherten Daten an den Eigentümer betrifft. Hier ist eine weitere Prämie für die Dateiübertragung vorgesehen. Sia vertraut in allen diesen Belangen auf den freien Markt: Hoster können sich als günstig und eher unzuverlässig positionieren, oder als verhältnismäßig teuer und garantieren dafür besonders oft Nachweise für die tatsächliche Speicherung von Daten vorzulegen. Angebot und Nachfrage sollen die Preise (und Zuverlässigkeit) regeln. Hervorzuheben ist in diesem Zusammenhang, dass die Verifikation von Transaktionen (und somit auch von Freigabebedingungen und Speicherungsnachweise) vom gesamten Netzwerk, wie im Rahmen traditioneller Kryptowährungen, stattfindet. Folglich müssen Teilnehmerinnen und Teilnehmer, welche Cloudspeicherplatz im Rahmen von Sia nutzen wollen, keinen Aufwand für Verifikation und Zahlungsabwicklung betreiben. Ein globales Reputationssystem soll Nutzerinnen und Nutzern helfen, korrekt handelnde Hoster auszuwählen. Um zu verhindern, dass unehrliche Parteien mit sich selbst Aufträge zum Speichern großer, gut komprimierbarer Dateien (z.B. bestehend aus Nullen) abwickeln und im Reputationssystem aufsteigen, ist es notwendig einen bestimmten Betrag in Kryptowährungstoken für längere Zeit zu hinterlegen, wenn man als Hoster auftreten will. Dadurch steigen die Kosten für Versuche, das Reputationssystem derart zu unterwandern, wodurch ein solches Vorgehen nicht rentabel ist.

Das Fortführen der Blockchain basiert auf demselben Anreizsystem und Konsensverfahren wie Bitcoin, wodurch dieselben Eigenschaften in Bezug auf Fehlertoleranz, Ausfallsicherheit, Datenintegrität, usw. gelten. Allerdings ergeben sich durch den Einsatz von Proof-of-Work und den damit einhergehenden hohen Energieverbrauch des Netzwerks zwingend auch dieselben Effizienzprobleme wie im Rahmen von Bitcoin. Eine Alternative stellt das auf dem IPFS-Netzwerk aufbauende Filecoin-System dar, welches im nachfolgenden Abschnitt beschrieben wird.

6 https://sia.tech/

4.3. Filecoin

Filecoin [7] ist eine Kryptowährung und ein dezentrales Speichernetzwerk, welche auf Basis des *IPFS* Peer-to-Peer-Netzwerks [8] konzipiert wurde. Im Gegensatz zu Sroj und Sia, handelt es sich dabei um eine umfangreich weiterentwickelte Blockchain im Vergleich zu den von Bitcoin oder Ethereum verwendeten Systemen. In den Grundzügen ähnelt Filecoin sehr stark Sia:

- Hoster (im Rahmen von Filecoin als *Storage-Miner* bezeichnet) hinterlegen eine proportional zum angebotenen Speicherplatz große Menge Filecoin-Token.
- Hoster müssen kryptografische Beweise für die Speicherung von Daten erbringen.
- Jeder Miner gibt unabhängig von anderen Minern an, wie viel Speicherplatz bei ihm oder ihr kostet.
- Nutzerinnen und Nutzer, welche Speicherplatz benötigen, legen fest wie viel sie bereit sind für Speicherplatz zu bezahlen.

- Die Prinzipien des freien Marktes sollen den Preis bestimmen.

Der gravierende Unterschied zu den zuvor beschriebenen Systemen liegt in der Art der kryptografischen Beweise, den damit einhergehenden Garantien und dem Konsensverfahren. Filecoin garantiert nicht nur, dass Daten zum Zeitpunkt des Beweisantritts beim Miner gespeichert sind, sondern, dass Daten auch über einen bestimmten Zeitraum beim Miner gespeichert waren. Dies wird im Filecoin-Whitepaper als *Proof-of-Spacetime* bezeichnet [1]. Weiters wird auch Datenreplikation kryptografisch bzw. formal beweisbar im Rahmen eines so genannten *Proof-of- Replication* umgesetzt [16]. Möglich machen dies so genannte *Zero-Knowledge Proofs* . Die Details zur Umsetzung und Integration würden den Rahmen dieses Dokuments sprengen und sind dem Filecoin-Whitepaper zu entnehmen. Auf Basis dieser Beweise und der Absicherung des zur Verfügung gestellten Speicherplatzes durch das Hinterlegen von Filecoin-Token wird im Rahmen von Filecoin ein *Proof-of-Spacetime* -Konsensverfahren vorgeschlagen. Im Gegensatz zu Proof-of-Work oder anderen Konsensverfahren, welche dem Netzwerk gegenüber aus der Tätigkeit selbst heraus keinen Nutzen bringen, legitimiert hier das bloße zur Verfügung stellen von Speicherplatz, bzw. das Speichern von Daten zum Erstellen neuer Blockchain-Blöcke, was (wie im Rahmen traditioneller Kryptowährungen) mit einer Auszahlung von Kryptowährungstoken an den Blockersteller einhergeht. Die im Rahmen von Filecoin eingesetzten Zero-Knowledge-Proofs werden (nach demselben Grundprinzip wie es Sia einsetzt) vom Netzwerk verifiziert, wodurch seitens der Nutzerinnen und Nutzer kein Verifikationsaufwand entsteht. Aktuell befindet sich Filecoin noch in der Testphase, unter anderem da die Konsequenzen des Proof-of-Spacetime-Konsensverfahrens noch weiter evaluiert werden sollen.

5. Fazit

Dezentrale Datenspeicherung kann im Vergleich zu traditionellen Cloudspeicherdiensten Vorteile in Bezug auf Verfügbarkeit und – je nach Vertrauensmodell – und Datenschutz mit sich bringen. Dabei handelt es sich jedoch aktuell nach wie vor um Modelle, welche auf Annahmen basieren, welche noch keiner Langzeitevaluierung unterzogen wurden. Dies liegt darin begründet, dass derartige Systeme ihren Mehrwert gegenüber traditionellen Cloudspeicherdiensten durch den Einsatz der Blockchain-Technologie zu erreichen versuchen. Im Gegensatz zu klassischen Kryptowährungsszenarien ergeben sich in diesem Kontext jedoch zusätzliche Anforderungen, welche weit weniger eng abgesteckt werden können. Systeme wie Filecoin, welche auch auf Blockchain- und Konsensfindungsebene versuchen, neue Wege zu gehen, stecken daher noch in den Kinderschuhen und sind aktuell noch nicht für den Produktiveinsatz ausgelegt.

7 https://filecoin.io/ 8 https://ipfs.io/

Referenzen

[1] Protocol Labs, „Filecoin: A Decentralized Storage Network," 19 07 2017. [Online]. Available:

https://filecoin.io/filecoin.pdf. [Zugriff am 31 10 2018].

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp und S. Shenker, „A Scalable Content Addressable Network," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, Ny, USA, ACM, 2001, pp. 161-172.

[3] Ion Stoica et al., „Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, San Diego, ACM, 2001, pp. 149-160.

[4] A. Rowstron, P. Druschel und R. Guerraoui, „Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware 2001*, Berlin, Springer, 2001, pp. 329-350.

[5] P. Maymounkov und D. Mazières, „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *Peer-to-Peer Systems*, Springer, 2002, pp. 53-65.

[6] R. C. Merkle, „A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, Springer, 1978, pp. 369-378.

[7] I. S. Reed und G. Solomon, „Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics,* pp. 300-304, 1960.

[8] B. Prünster, „Sichere Peer-to-Peer-Netze auf Basis von Selbstzertifizierung," 07 05 2018.

[Online]. Available: https://technology.a-sit.at/veroeffentlichungen/sichere-peer-to-peer-netze-auf-basis-von-selbstzertifizierung/. [Zugriff am 11 07 2018].

[9] B. Prünster, „Verteilter Datenspeicher in Heterogenen Umgebungen," 11 07 2018. [Online].

Available: https://technology.a-sit.at/veroeffentlichungen/verteilter-datenspeicher-in-heterogenen-umgebungen/.

[Zugriff am 31 10 2018].

[10] B. Cohen, „The BitTorrent Protocol Specification," 11 10 2013. [Online]. Available:

http://www.bittorrent.org/beps/bep_0003.html. [Zugriff am 24 04 2018].

[11] S. Nakamoto, „Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available:

https://bitcoin.org/bitcoin.pdf. [Zugriff am 31 10 2018].

[12] A. Marsalek und B. Prünster, „Technologieüberblick Blockchain," 24 10 2016. [Online].

Available: https://technology.a-sit.at/veroeffentlichungen/technologieueberblick-blockchain/. [Zugriff am 24 04 2018].

[13] I. Baumgart und S. Mies, „S/Kademlia: A practicable approach towards secure key-based routing," in *2007 International Conference on Parallel and Distributed Systems*, 2007.

[14] S. Wilkonson, T. Boshevski, J. Brandoff, J. Prestwich, G. Hall, P. Gerbes, P. Hutchins und C.

Pollard, „Storj, A Peer-to-Peer Cloud Storage Network," 15 12 2016. [Online]. Available: https://storj.io/storjv2.pdf. [Zugriff am 29 10 2018].

[15] L. C. David Vorick, „Sia: Simple Decentralized Storage," 29 11 2014. [Online]. Available:

https://sia.tech/sia.pdf. [Zugriff am 30 10 2018].

[16] J. Benet, D. Dalrymple und N. Greco, „Proof of Replication,“ 27 07 2017. [Online]. Available:

https://filecoin.io/proof-of-replication.pdf. [Zugriff am 31 10 2018].