Sigurd Schacht
Carsten Lanquillon *Hrsg.*

Blockchain und maschinelles Lernen

Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren

EBOOK INSIDE

Springer Vieweg

Blockchain und maschinelles Lernen

Sigurd Schacht · Carsten Lanquillon
(Hrsg.)

Blockchain und maschinelles Lernen

Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren

Springer Vieweg

Hrsg.
Sigurd Schacht
Hochschule Heilbronn
Heilbronn, Deutschland

Carsten Lanquillon
Hochschule Heilbronn
Heilbronn, Deutschland

ISBN 978-3-662-60407-6 ISBN 978-3-662-60408-3 (eBook)
https://doi.org/10.1007/978-3-662-60408-3

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Springer Vieweg

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Inhaltsverzeichnis

1 Einleitung 1
Sigurd Schacht und Carsten Lanquillon
Literatur 2

2 Die Blockchain-Technologie 3
Sigurd Schacht
2.1 Bedeutung und Verbreitung 3
2.2 Blockchain und Distributed-Ledger-Technologie: Eine kurze Einführung 5
2.3 Distributed-Ledger-Technologie im Detail 6
2.3.1 Ebene 1 „Principles“: Komponenten der Distributed Ledger Technologie 7
2.3.2 Ebene 2 „Concept – Base Framework“ 37
2.3.3 Ebene 3 „Concept – Application Framework“ 69
2.3.4 Ebene 4 „Implementation Frameworks“ 75
2.4 Ausblick 82
Literatur 82

3 Grundzüge des maschinellen Lernens 89
Carsten Lanquillon
3.1 Definition und Entwicklung 89
3.1.1 Informatik, KI und maschinelles Lernen 90
3.1.2 Machine-Learning-Definitionen 90
3.1.3 Maschinelles Lernen und verwandte Disziplinen 93
3.2 Lernformen: Wie wird gelernt? 95
3.2.1 Überwachtes Lernen 96
3.2.2 Unüberwachtes Lernen 97
3.2.3 Halbüberwachtes Lernen 97
3.2.4 Bestärkendes Lernen 98
3.2.5 Weitere Kategorisierungen 99

3.3 Aufgabentypen: Was wird gelernt? 100
3.3.1 Modellanwendung: Wie wird ein Modell verwendet? 100
3.3.2 Eingabe: Die Daten als Matrix 101
3.3.3 Gängige Aufgabentypen 102
3.4 Grundlegende Vorgehensweisen 111
3.4.1 CRISP-DM: Ein Prozessmodell für Analyseprojekte 112
3.4.2 Automatisierung und Operationalisierung 117
3.5 Lernverfahren: Ein kurzer Überblick 122
3.5.1 Grundbausteine maschineller Lernverfahren 123
3.5.2 Verfahrensklassen 124
3.5.3 Ausgewählte Lernverfahren 126
3.6 Zentralisiertes und verteiltes Lernen: Wo wird gelernt? 131
3.6.1 Parallelisierung von Berechnungen 131
3.6.2 Klassisches verteiltes Lernen 132
3.6.3 Föderiertes Lernen 135
3.7 Zusammenfassung und Ausblick 137
3.7.1 Einfluss der Daten 137
3.7.2 Einfluss der Lernverfahren 139
Literatur 140

4 Blockchain und maschinelles Lernen – Ein Literaturüberblick 143
Jerome Tagliaferri
4.1 Einleitung 143
4.2 Maschinelles Lernen zur Unterstützung der Blockchain 144
4.3 Blockchain zur Unterstützung des maschinellen Lernens 146
4.3.1 Datensicherheit 146
4.3.2 Smart Contracts 148
4.3.3 Incentivierung 150
4.3.4 Datenplattform 151
4.3.5 Anwendungsformen 153
4.4 Anwendungsfälle 156
4.4.1 Einleitung 156
4.4.2 Anwendung 1 156
4.4.3 Anwendung 2 157
4.4.4 Anwendung 3 159
4.5 Zusammenfassung 160
Literatur 164

5 Der Analytics-Marktplatz 167
Carsten Lanquillon und Sigurd Schacht
5.1 Motivation 167
5.2 Zielsetzung 169

5.3 Ordnungsrahmen . . . 171
5.3.1 Ressourcen und Rollen. . . . 172
5.4 Herausforderungen und Lösungsansätze . . . 177
5.4.1 Vertrauen . . . 177
5.4.2 Datenschutz . . . 180
5.4.3 Anreizsystem . . . 182
5.4.4 Automatisierung. . . . 183
5.5 Aktuelle Ansätze . . . 183
5.5.1 Kurtulmus und Daniel: Trustless Machine Learning Contracts. . . . 184
5.5.2 Özyilmaz et al.: IDMoB: IoT Data Marketplace on Blockchain . . . 187
5.5.3 Fitchain. . . . 189
5.6 Zusammenfassung und Ausblick . . . 192
Literatur. . . . 193

6 DLT im Energiesektor – Wie blockchainbasierte Werkzeuge und maschinelles Lernen ein dekarbonisiertes Energiesystem möglich machen. . . . 195
Thomas Brenner
6.1 Dezentrales Energiesystem – verteiltes IT-Netzwerk . . . 195
6.1.1 Herausforderungen. . . . 196
6.1.2 DLT und neutrale lokale Märkte als Lösungsansatz . . . 198
6.2 Von der Anlage zum Kunden – Das Allgäu Microgrid . . . 200
6.2.1 Status Quo und Ziele . . . 200
6.2.2 Architektur . . . 201
6.2.3 Ein Blick hinter die Kulissen – das Blockchain-Backend und exemplarische Smart Contracts . . . 204
6.2.4 Die Kundenschnittstelle . . . 206
6.2.5 Einsatzgebiete . . . 208
6.3 Machine Learning für ein agentenbasiertes Energiemanagement . . . 210
6.3.1 Data Mining zur Disaggregaton von Energieverbrauchsdaten. . . . 211
6.3.2 Deep Neural Networks und SMPC zur Prognoseoptimierung und Prozesssteuerung . . . 211
6.3.3 Predictive Maintenance und Pay Per Use . . . 213
6.4 Zusammenfassung . . . 214
Literatur. . . . 215

Stichwortverzeichnis. . . . 217

Einleitung

1

Sigurd Schacht und Carsten Lanquillon

Sowohl die Blockchain als auch die künstliche Intelligenz, die auch das maschinelle Lernen umfasst, zählen zu den disruptiven Technologien, die unsere Arbeitswelt und insbesondere die Zusammenarbeit und Interaktion von Unternehmen und Arbeitnehmern fundamental ändern werden [1]. Was aber haben Blockchain und maschinelles Lernen miteinander zu tun?

Nach dem Abflachen des „Krypto-Hype“ von 2017, bei dem die unterschiedlichen Kryptowährungen, allen voran *Bitcoin,* großes Aufsehen erregt hatten, rückte die Technologie dahinter mehr und mehr in den Vordergrund. Dabei gehört die Blockchain zu den jüngeren Technologien. Aufgrund ihres Charakters als Infrastruktur-Technologie kann sie nicht schnell eingeführt werden bzw. führt sie auch nicht zu schnellen Erfolgen. Dementsprechend gibt auch Gartner in seinem Hype-Cycle an, dass die Technologie erst zwischen 2024 und 2029 den Bereich der Produktivität erreichen wird. Gartner behandelt in seinem Beitrag „The Reality of Blockchain“ auch die „programmable economy“, also eine „programmierbare Wirtschaft“, die ein intelligentes Wirtschaftssystem, das die Produktion und den Konsum von Waren und Dienstleistungen unterstützt, verwaltet und verschiedene Szenarien des Werteaustausches ermöglichen soll [3].

In diesem Kontext spielt die Blockchain-Technologie eine essenzielle Rolle. Es geht um die Optimierung, Standardisierung und Automatisierung von Prozessen und Abläufen nicht nur im Unternehmen selbst, sondern über die Unternehmensgrenzen hinweg. Bevorzugt soll ein hoher Grad an Automatisierung erreicht werden, der nicht nur die operationalen Abläufe betrifft, sondern auch die monetären Ströme zwischen den beteiligten Parteien abbildet. Die

S. Schacht (✉) · C. Lanquillon
Hochschule Heilbronn, Heilbronn, Deutschland
E-Mail: sigurd.schacht@hs-heilbronn.de

C. Lanquillon
E-Mail: carsten.lanquillon@hs-heilbronn.de

S. Schacht und C. Lanquillon (Hrsg.), *Blockchain und maschinelles Lernen,*
https://doi.org/10.1007/978-3-662-60408-3_1

Entwicklung einer „programmable economy“ benötigt aus Sicht von Gartner eine passende Infrastruktur, die durch die Blockchain-Technologie bereitgestellt wird. Außerdem ist die Möglichkeit der Automatisierung durch Smart Contracts notwendig, der Einsatz dezentraler Applikationen und natürlich die Partizipation der Kunden.

Eine blockchainbasierte Infrastruktur alleine wird jedoch nicht genügen, um den eingebetteten Programmen auch die Möglichkeit zu bieten, nicht nur nach vordefinierten Regeln zu entscheiden, sondern Entscheidungen auf intelligente Art und Weise zu treffen, also insbesondere kontextabhängig und adaptiv die Zusammenhänge zwischen relevanten Einflussfaktoren berücksichtigend. Dafür wiederum ist der Einsatz künstlicher Intelligenz erforderlich. Insbesondere das maschinelle Lernen, durch das Programme ohne explizites Programmieren datenbasiert erstellt werden können, ist der entscheidende Baustein für einen hohen Grad der Automatisierung in der programmierbaren Wirtschaft, denn es kann das Automatisieren automatisieren [2].

Bereits diese Überlegung lässt erahnen, dass die beiden Technologien jeweils Eigenschaften und Potenziale besitzen, die sich sehr gut ergänzen können. Wie aber kann dieses Zusammenspiel aussehen und gestaltet werden? So vielversprechend und naheliegend eine Kombination der beiden Technologien auch ist, gibt es auf dem Weg zur ihrer Gestaltung und Umsetzung noch viele Fragen und Herausforderungen zu klären.

Dieses Buch soll einen Einblick in die gegenwärtige Entwicklung der Kombination aus maschinellem Lernen und der Blockchain-Technologie bieten. Dazu wird im folgenden Kapitel zunächst eine Einführung in die Blockchain und die dahinter stehende Distributed-Ledger-Technologie gegeben. In Kap. 3 werden sodann die wichtigsten Aspekte des maschinellen Lernens mit Blick auf die folgende Kombination mit der Blockchain kompakt zusammengefasst. Kap. 4 gibt einen Überblick über die aktuelle Forschung in diesem Bereich. Dabei entpuppt sich insbesondere der Aufbau von Marktplätzen zum nahtlosen und automatisierbaren Austausch von Ressourcen als vielversprechende Anwendungsform. Ein branchenneutraler Analytics-Marktplatz sowie ein Use-Case aus dem Energiesektor werden schließlich als aktuelle Beispiele der Kombination aus maschinellem Lernen und der Blockchain-Technologie in den Kap. 5 und 6 detailliert erläutert.

Literatur

1. Dinh, T.N., Thai, M.T.: AI and blockchain: a disruptive integration. Computer **51**(9), 48–53 (2018). https://doi.org/10.1109/MC.2018.3620971
2. Domingos, P.: The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books Inc., New York (2015)
3. Heather Pemberton Levy: The Reality of Blockchain – Smarter With Gartner. https://www.gartner.com/smarterwithgartner/the-reality-of-blockchain/ (2018). Zugegriffen: 20. Juni 2019

Die Blockchain-Technologie

2

Sigurd Schacht

Zusammenfassung

In diesem Kapitel wird ein Überblick über das Ecosystem Distributed-Ledger-Technologie dargestellt. Ziel ist es, ein Verständnis über die neue Technologie zu vermitteln und einen möglichst breiten Überblick über die unterschiedlichsten Technologien und Verfahren zu liefern. Dabei wird als Struktur ein Ordnungsrahmen, gegliedert nach vier Ebenen der Distributed-Ledger-Technologie, herangezogen. Auf der ersten Ebene werden die wichtigsten Prinzipen der Distributed-Ledger-Technologie dargestellt. Darauf aufbauend werden in Ebene zwei Blockchain, Tangle und Hashgraph als Ausprägungen der DLT erläutert. In Ebene drei werden die wichtigsten Applikationen, Smart Contracts und dezentrale autonome Organisationen dargestellt. Als Abschluss wird jeweils eine DLT-Implementierung für öffentliche und private DLTs erläutert.

2.1 Bedeutung und Verbreitung

Die Blockchain-Technologie steht noch am Anfang ihrer Entwicklung, weshalb momentan in Unternehmen überwiegend Pilotprojekte gestartet werden. So entwickelt Facebook die eigene Kryptowährung „Libra“, die den Zahlungsverkehr zwischen Individuen länderübergreifend schnell und einfach ermöglichen soll. Aber auch andere Unternehmen werden zukünftig in diese Technologie investieren. Nach einer Studie der International Data Corporation sollen die Ausgaben für Blockchain-Lösungen von 0,95 US$ im Jahr 2017 auf bis zu 11,7 Mrd. US$ im Jahr 2022 ansteigen. Diese Investitionssumme verteilt sich vor allem auf die USA, West-Europa und China [44].

S. Schacht (✉)
Hochschule Heilbronn, Heilbronn, Deutschland
E-Mail: sigurd.schacht@hs-heilbronn.de

S. Schacht und C. Lanquillon (Hrsg.), *Blockchain und maschinelles Lernen*,
https://doi.org/10.1007/978-3-662-60408-3_2

Auch Deloitte bietet mit einer Umfrage über den Status aktueller Blockchain-Projekte vom April 2018 einen interessanten Einblick.

Es wird deutlich, dass sich über alle Branchen hinweg mehr als 31 % der Befragten in der Phase des Experimentierens (Proof-of-Concept) befinden und dass einige wenige Projekte schon produktiv genutzt werden, besonders im Bereich Konsumgüter & Maschinenbau. Weiterhin zeigt die Umfrage, dass ein Großteil der Befragten eine Produktivsetzung schon innerhalb des nächsten Jahres sieht (Abb. 2.1).

Ganz vorne stehen hier die Länder China und Mexiko, während überraschenderweise die USA, die erwartungsgemäß auf Grund der technologischen Dominanz im Silicon Valley normalerweise eine führende Rolle einnimmt – zumindest laut dieser Studie – in Bezug auf produktive Blockchain-Projekte im Vergleich zu anderen Nationen weit abgeschlagen sind (Abb. 2.2).

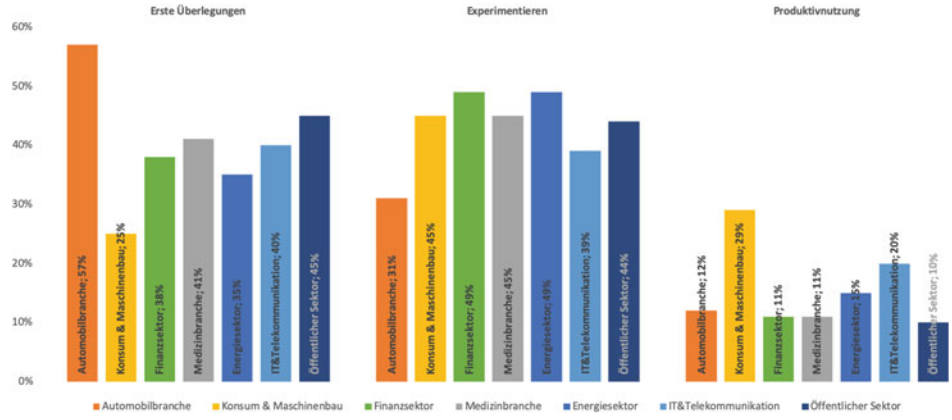

Abb. 2.1 Weltweite Blockchain-Einführungsphasen in Unternehmen [77]

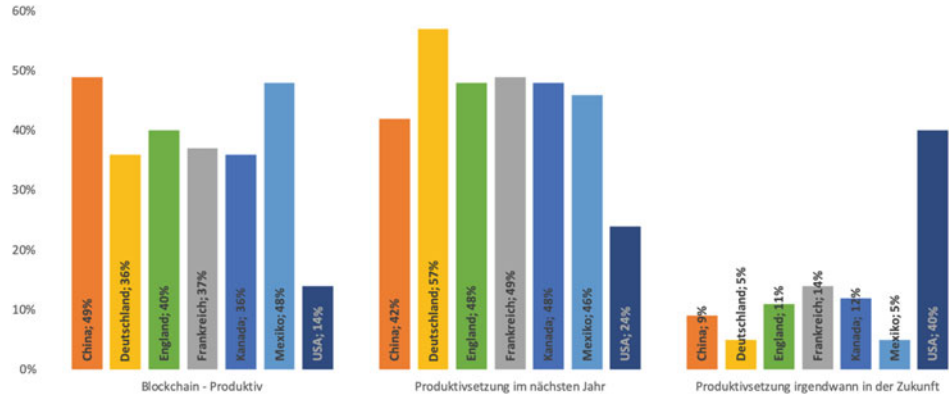

Abb. 2.2 Geplante Produktivsetzungen von Blockchain-Projekten [77]

Es ist anzunehmen, dass die Blockchain-Technologie in den nächsten ein bis zwei Jahren in immer mehr produktiven Projekten, besonders in kleinen, speziellen Anwendungsbereichen, wie das Nachvollziehen von Lieferflüssen, eingesetzt werden wird, eine große Verbreitung aber erst in fünf bis zehn Jahren zu erwarten ist.

2.2 Blockchain und Distributed-Ledger-Technologie: Eine kurze Einführung

Distributed-Ledger-Technologie (DLT) bedeutet, dass Transaktionen nacheinander in einem auf viele unabhängige Rechner verteilten Hauptbuch (Ledger) gespeichert werden [58]. Die in Form eines dezentralen Netzwerks organisierten Rechner werden dabei als Knoten oder Nodes bezeichnet. Bei der Blockchain, dem bekanntesten DLT-Verfahren, werden Transaktionen in Form von (Informations-) Blöcken zusammengefasst und mittels kryptografischer Verfahren miteinander verkettet [72]. Das Besondere an der Blockchain ist die transparente, nachvollziehbare und verifizierte Übertragung von Transaktionen sowohl in öffentlichen wie auch geschlossenen Transaktionsketten zwischen unbekannten Marktteilnehmern ohne eine zentrale Instanz.

Fünf wesentliche Eigenschaften zeichnen die Blockchain aus:

Dezentral und belastbar
Die Blockchain ist dezentral organisiert. Das heißt, es gibt keine zentrale Kontrollinstanz, sondern alle Knoten im Netzwerk sind in ihrer Funktion gleichberechtigt. Jederzeit kann ein Knoten das Netz verlassen oder ein anderer hinzukommen [94]. Die dezentrale Organisation bewirkt, dass eine zentrale Kontrolle durch ein Individuum oder eine Organisation erschwert wird. Außerdem wird dadurch das Netzwerk wesentlich belastbarer und ein Ausfall mit zunehmender Anzahl an Knoten immer unwahrscheinlicher. Die Teilnehmer des Netzwerks sind nicht auf ein Unternehmen, eine Organisation oder ein Land beschränkt, sondern können über Landes- und Unternehmensgrenzen hinweg organisiert sein. Auf diese Weise sind Blockchains auch gegenüber politischen oder organisatorischen Eingriffen weitgehend geschützt.

Verifizierbar und transparent
Auf jedem Knoten des Netzwerks sind die Daten der gesamten Blockchain gespeichert, was den Zugriff auf alle Transaktionen innerhalb der Blockchain für jeden Teilnehmer ermöglicht [94]. Jede Transaktion, die in der Blockchain gespeichert werden soll, wird freiwillig durch immer wieder wechselnde andere Netzwerkteilnehmer verifiziert. Hierbei werden unterschiedliche Algorithmen, auch Konsens-Algorithmen genannt, verwendet. Der bekannteste Konsens-Algorithmus heißt „Proof-of-Work“ und verifiziert Transaktionen auf Basis von Rechenleistung [22]. Sowohl der Konsens-Mechanismus, als auch die Möglichkeit für jeden Netzteilnehmer, alle Transaktionen zu betrachten, sorgen für die Belastbarkeit der Daten sowie eine hohe Nachvollziehbarkeit.

Unveränderlich
Alle Daten, die in der Blockchain gespeichert werden, sind „by design“ unveränderlich. Zum Einen ist jede Transaktion in einem eigenen Block gespeichert und dieser mittels kryptografischer Hash-Verfahren mit seinem nachfolgenden Block verkettet. Zum Anderen liegen diese verketteten Blöcke in identischer Form auf vielen unterschiedlichen Knoten [94]. Das heißt: Wird eine Transaktion innerhalb eines Knotens verändert, ist die Block-Kette auf diesem speziellen Knoten nicht mehr konsistent, da die errechneten Werte der Blöcke nicht mehr den definierten Regeln entsprechen.

Um diesen Mangel zu beheben, müsste ein potenzieller Angreifer nun alle nachfolgenden Blöcke mit viel Rechenaufwand neu verketten. Der Einsatz der dafür notwendigen Rechenleistung stellt die erste Hürde dar. Die zweite Hürde ist, dass die so veränderte Blockchain auf dem einen Knoten nicht mehr mit der Mehrheit der Kopien auf den übrigen Knoten übereinstimmt. Das hat zur Folge, dass der so manipulierte Knoten im Netzwerk als nicht mehr gültig erachtet und ihm eine weitere Teilnahme am Netzwerk verwehrt wird. Damit wäre die Manipulation eliminiert.

„Unveränderbarkeit“ bedeutet in diesem Zusammenhang nicht, dass keinerlei Daten verändert werden können, sondern dass das System der Blockchain ungerechtfertigte Änderungen erkennt und aus dem Netzwerk eliminiert.

Eine Erläuterung über den genauen Ablauf des Blockchain-Protokolls erfolgt im Abschnitt „Ebene 2 ‚Concept Base – Framework‘“ gegeben.

2.3 Distributed-Ledger-Technologie im Detail

Unter Distributed-Ledger-Technologie versteht man ein sozio-technisches Informationssystem, das wie eine dezentrale Datenbank funktioniert, bei dem aber keine dominante zentrale Validierungsinstanz für die zu speichernden Informationen vorhanden ist. Stattdessen erfolgt die Konsensbildung über ein demokratisches Prinzip mit Hilfe von Konsens-Algorithmen.

In Anlehnung an Burkhard et al. wird die Distributed-Ledger-Technologie im vorliegenden Buch in vier wesentliche Ebenen aufgeteilt [22]. Ebene 1 „Principles“ geht auf die Komponenten und technischen Aspekte der Distributed-Ledger-Technologie ein. In der Ebene 2 „Concept – Base Framework“ werden die wesentlichen Distributed-Ledger-Technologien wie Blockchain, Tangle und Hashgraph erläutert sowie die Unterschiede zwischen diesen Technologien dargestellt. Auf Ebene 3 „Concept – Application Framework“ werden aufbauend auf den Base Frameworks die Aspekte Smart Contracts und dezentrale autonome Organisationen erklärt. Auf Ebene 4 „Implementation Frameworks“ werden die am Markt gängigen DLT-Projekte nach öffentlichen und permissioned bzw. private DLTs erläutert und die Vor- und Nachteile dargelegt (Abb. 2.3).

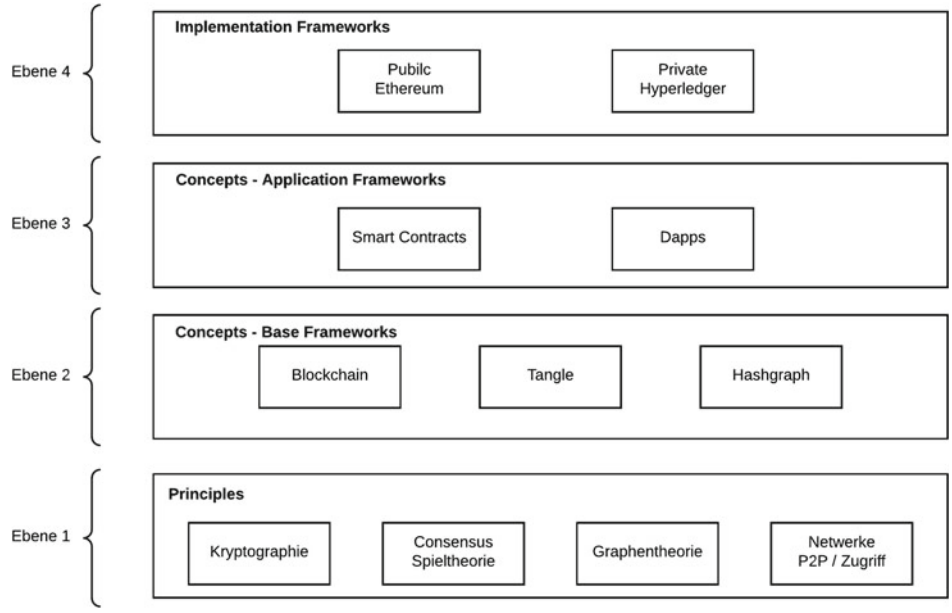

Abb. 2.3 Ebenen der Distributed-Ledger-Technologie. (In Anlehnung an [22])

2.3.1 Ebene 1 „Principles": Komponenten der Distributed Ledger Technologie

Um ein verteiltes Hauptbuch zu generieren, nutzt die Distributed-Ledger-Technologie unterschiedliche Technologien: Für die Verteilung der Informationen im Netzwerk ohne zentrale Instanz wird auf die Peer-to-Peer-Netzwerk-Technologie zurückgegriffen. Für die Verkettung der einzelnen Blöcke findet zur Bildung von eindeutigen Fingerabdrücken ein Hash-Algorithmus Verwendung. Für die Sicherstellung der Identität von Transaktionen wird die asymmetrische Verschlüsselung herangezogen, und für die Schaffung einer Einigung darüber, welche Transaktion überhaupt in die Blockchain geschrieben werden darf, werden unterschiedliche Konsens-Algorithmen wie beispielsweise Proof-of-Work benutzt. Bei der Konsensbildung ist wichtig, dass neben dem Willen aller Teilnehmer zur Sicherung des Netzwerks auch ein Anreiz zur Beteiligung vorhanden ist. Die Implementierung solcher Anreizsysteme geschieht durch die Anwendung spieltheoretischer Ansätze, auf die später noch genauer eingegangen wird. Im Grunde kann man die Blockchain-Technologie als Datenstruktur und Algorithmus in Analogie zur Software beschreiben. Eine Software ist die Summe aus Daten (Datenstruktur) und Algorithmen, die die Daten beliebig bearbeiten [66]. Ähnlich ist es bei der Blockchain; Sie ist die Summe aus den Datenstrukturen, hier Distributed-Ledger-Technologie, und den Algorithmen, die diese Daten verarbeiten. Zu den Algorithmen gehören unter anderem auch die Konsens-Algorithmen [66].

$$Blockchain = Distributed\text{-}Ledger\text{-}Technologie + Consensus \tag{2.1}$$

2.3.1.1 Kryptografie

Kryptografie spielt in der Blockchain-Technologie eine wichtige Rolle, da über die Verwendung eines Verschlüsselungsverfahrens sichergestellt wird, dass die Person, die eine Transaktion in das Blockchain-Netz eingibt, nachweislich auch diese Person ist und dass die Transaktion nicht während der Übertragung verändert wurde. Mittels der Kryptografie wird somit der Schutz der Daten gegenüber Zugriff unberechtigter Dritter gewährleistet.

Des Weiteren werden kryptografische Hashes verwendet, um Veränderungen an Daten aufzuzeigen und nachvollziehbar zu machen.

Hash-Verfahren

Bei den Distributed-Ledger-Technologien spielen die Hash-Funktionen – auch Streuwertfunktion genannt – eine sehr wichtige Rolle. Mittels dieser Verfahren ist es möglich, für jede Nachricht einen eindeutigen Fingerabdruck zu erzeugen, indem eine Zeichenkette mit einer vorgegebenen Länge generiert wird. Dabei ist es egal, wie groß die Nachricht – also der Input – ist, die erzeugte Zeichenkette ist immer gleich lang. Der Vorteil von solchen Hash-Verfahren ist, dass mittels dieses Werts überprüft werden kann, ob eine Nachricht im Nachhinein verändert wurde oder nicht.

Wichtige Anforderungen für eine Hash-Funktion sind, dass für identische Eingaben an Informationen auch immer ein identischer Hash-Wert erzeugt wird. Des Weiteren muss die Hash-Funktion so aufgebaut werden, dass es so wenig Kollisionen wie möglich gibt, dass also die Ausgabe identischer Hash-Werte bei unterschiedlichen Eingaben vermieden wird. Hash-Verfahren dürfen nicht mit Verschlüsselung gleichgesetzt werden. Zwar können Nachrichten in einen Hash-Wert umgewandelt werden, der umgekehrte Weg ist aber nicht mehr möglich. Der Hash-Wert ist nicht invertierbar [59].

Eine typische Verwendung des Hash-Werts ist die sichere Ablage von Passwörtern. Hierzu wird das Passwort mittels einer Hash-Funktion in einen Hash-Wert gewandelt und dieser in einer Datenbank abgelegt. Ein unberechtigter Lesezugriff auf die Datenbank offenbart dann nicht das Passwort, sondern nur den korrespondierenden Hash-Wert, mittels dem das Passwort nicht wiederhergestellt werden kann. Ein Zurückrechnen ist in dem genannten Beispiel bei der Prüfung des Kennwortes aber auch nicht notwendig, da dies durch den Vergleich des berechneten Hash-Werts aus dem eingegebenen Passwort und dem gespeicherten Hash-Wert geschieht.

Gängige Hash-Funktionen im kryptografischen Umfeld sind MD5 in der Länge von 128 bits oder der Secure-Hash-Algorithmus (SHA) in unterschiedlichen Längen. Also z. B. SHA256.

Als Beispiel wird aus dem unten stehenden Input-Text bei Verwendung des gängigen Verfahrens SHA256 ein Hash-Wert erzeugt:

Input: „Dieses Buch handelt von Blockchain und Machine Learning!“
Hash-Wert: „93e4833c9e363723bfb406b1c92a8c6faf421a1a32ee69e2e1180ed3827db9ce“

Der SHA256 hat einen sehr guten Lawineneffekt, also die Eigenschaft, dass kleine Änderungen der Input-Nachricht zu einem komplett anderen Hash-Wert führen. Im Beispiel ändern wir das Ausrufezeichen am Ende des Beispieltextes in ein Fragezeichen:

Input: „Dieses Buch handelt von Blockchain und Machine Learning?“
Hash-Wert: „a3c778f3b7a22c2e4b77fd32e0cfa13d0ab0df99e05e36eff923eb8f68f55717“

Kaum ein Zeichen ist in diesem neuen Hash-Wert gleich geblieben, obwohl nur ein einziges Input-Zeichen verändert wurde.

Die Länge des Hash-Werts gibt an, wie viele unterschiedliche Werte erzeugt werden können. Eine Hash-Funktion mit der Länge 256 Bit ermöglicht die Erzeugung von 2^{256} Hash-Werten ohne Kollision. Eine Kollision ist bei dieser Anzahl an möglichen Ergebnissen – nämlich mehr als Sandkörner auf der Erde – überhaupt unwahrscheinlich, besonders wenn die Nebenbedingung erfüllt ist, dass die möglichen Ergebnisse gleich wahrscheinlich auftreten, also gleichverteilt sind [62].

Die Hash-Funktion ist eine wichtige technische Komponente der Blockchain bzw. der Distributed-Ledger-Technologie allgemein, da sie dazu dient, jeden Block, in dem Transaktionen gespeichert sind, mit einem eindeutigen Fingerabdruck zu versehen, der wiederum im nächsten Block gespeichert wird. Auf diese Weise werden die Blöcke miteinander verbunden und unbemerkte Manipulationen einzelner Blöcke unmöglich.

Des Weiteren wird die Hash-Funktion bei der digitalen Signatur benötigt, die wiederum bei allen Transaktionen angewandt wird.

Asymmetrische Verschlüsselung & digitale Signatur

Zum Verschlüsseln von geheimen Informationen wird sowohl die symmetrische als auch die asymmetrische Verschlüsselung angewandt. Bei der symmetrischen Verschlüsselung einigen sich zwei Parteien auf einen gemeinsamen geheimen Schlüssel, der zur Verschlüsselung der Nachricht herangezogen wird. Dieser geheime Schlüssel muss beiden Parteien bekannt sein, denn nur so kann Individuum A die Nachricht ver- und Individuum B sie mit demselben Schlüssel wieder entschlüsseln. Es muss also gewährleistet sein, dass beide Parteien sich vor der eigentlichen Verschlüsselung der sensiblen Nachricht über einen geeigneten geheimen Schlüssel austauschen und diesen über einen – meist ungesicherten – Kanal übertragen. Dies birgt Risiken, da ein Dritter den geheimen Schlüssel abfangen und so die Nachricht übersetzen könnte.

Das asymmetrische Verschlüsselungsverfahren versucht, das Übermitteln des geheimen Schlüssels durch die Verwendung von einem Schlüsselpaar aus einem privaten und einem öffentlichen Schlüssel überflüssig zu machen. Dieses Verfahren wird bei den meisten Blockchain-Technologien verwendet, schon, weil ein direkter Austausch des geheimen

Schlüssels für die symmetrische Verschlüsselung bedingt, dass alle Teilnehmer eines Netzwerks sich persönlich kennen.

Bei der asymmetrischen Verschlüsselung wird für jeden Teilnehmer des Netzwerks jeweils ein öffentlicher und ein privater Schlüssel generiert. Beide Schlüssel bestehen aus mathematisch generierten alphanumerischen Zeichen einer bestimmten Länge und sind nicht identisch, hängen aber voneinander ab. Die Berechnung des Schlüsselpaares muss so gestaltet sein, dass sie in die eine Richtung einfach zu bestimmen, aber schwierig wieder zurückzuberechnen ist [20]. Zum Beispiel werden beim RSA-Verfahren – das seinen Namen von den Erfindern Rivest, Shamir und Adleman bekommen hat – Einwegfunktionen auf Basis von Primzahlen-Multiplikationen vorgenommen. Diese Berechnung ist sozusagen eine mathematische Einbahnstraße, da die Multiplikation zwar simpel ist, das Ergebnisprodukt aber nur sehr schwer wieder zurück in die ursprünglichen Primzahlen zerlegt werden kann [54].

Das Produkt 42.444.277 aus den Primzahlen 8311 und 5107 kann beispielsweise bei Kenntnis dieser beiden Ausgangswerte sehr schnell berechnet werden. Anders herum dauert es schon etwas länger, ohne Kenntnis der einzelnen Primzahlen die Teiler von 42.444.277 zu finden. Die genannte Zahl hat aufgrund der Wahl der Primfaktoren als Ausgangsprodukt nämlich nur vier Teiler: 1, die Zahl selber und die beiden Primzahlen, aus der sie gebildet wurde.

Für die Rückrechnung des Produktergebnisses auf die ursprünglichen Primzahlen gibt es momentan keine guten Algorithmen, es muss viel durch ausprobieren ermittelt werden. Dazu sind Computer natürlich in der Lage, aber wenn für das Produkt ausreichend große Primzahlen herangezogen werden, dauert dies sehr lange. Bei einem 300-stelligen Produkt ist aktuell kein Computer in der Lage, die Berechnung in akzeptabler Zeit vorzunehmen. Auch eine Steigerung der Rechenleistung bringt da nicht viel, weil parallel zur Steigerung der Rechenleistung auch die Größe der Ausgangs-Primzahlen verändert werden und damit das Lösen des Rätsels ebenfalls nicht schneller vorgenommen werden kann [54].

Da es nun aber wichtig ist, dass die Person, die den privaten Schlüssel besitzt, eine verschlüsselte Nachricht schnell und einfach entschlüsseln kann, wird auf sogenannte Falltürfunktionen zurückgegriffen. Das sind Funktionen, die mit einer Zusatzinformation leicht rückzurechnen sind. Im RSA-Verfahren setzt sich der private Schlüssel aus den beiden Primfaktoren und einer weiteren Zahl zusammen, deren größter gemeinsamer Teiler 1 sein muss. Aus diesen drei Komponenten können nun öffentlicher und privater Schlüssel erzeugt werden.

Folgende Formel ergibt den öffentlichen Schlüssel:

Beispielhaft würden die Primzahlen 13 und 19 sowie die Zahl e mit 7 einen privaten Schlüssel d von 31 ergeben:

$$1 = (e * d) mod ((p-1) * (q-1)) \tag{2.2}$$

p, q Primzahlen
e Weitere Zahl
d Privater Schlüssel

$$1 = (7 * d) mod((13 - 1) * (19 - 1)) =$$
$$1 = (7 * d) mod 216 =$$
$$(7 * d) = 216 + 1 =$$
$$d = 31$$

Da die Restberechnung 1 sein muss, ergibt sich für den Term (7 * d) das notwendige Ergebnis von 217. d ist folglich 31.

Der öffentliche Schlüssel setzt sich zusammen aus dem Produkt o, der Multiplikation beider Primzahlen sowie der weiteren Zahl e:

$$o = (p * q) = 12 * 19 = 247 \tag{2.3}$$

Somit ergibt sich dieser als 247 und 7. Mittels dieser beider Zahlen kann nun eine beliebige Nachricht verschlüsselt werden. Hierbei wird die Nachricht N zunächst in eine Zahl umgewandelt, z. B. könnte man hierfür die ASCII-Code-Tabelle für den Buchstaben T, der durch den Wert 84 repräsentiert wird, heranziehen. Eine Verschlüsselung dieses Werts wird nun mittels nachstehender Formel vorgenommen:

$$C = N^e mod o \tag{2.4}$$

In dem aufgeführten Beispiel würde folgende Berechnung vorgenommen:

$$C = 84^7 mod 247$$
$$C = 46$$

Eine Entschlüsselung der Zahl 46 zurück zur 84 ist nun nurmehr möglich, wenn der private Schlüssel d bekannt ist. Dann wird folgende Formel angewandt:

$$N = C^d mod o \tag{2.5}$$

In dem Beispiel würde folgende Rückrechnung vorgenommen werden:

$$C = 46^{31} mod 247$$
$$C = 84$$

Die 84 entspricht unserem vorher ausgewählten ASCII-Zeichencode. Auf diese Weise kann nun eine Nachricht verschlüsselt werden.

Da es sehr schwierig ist, aus dem öffentlichen Schlüssel den privaten Schlüssel zu bestimmen, können die öffentlichen Schlüssel jedem bekannt gemacht werden, was einen leichten Austausch der Schlüssel ermöglicht. Dadurch reduziert sich die Anzahl der benötigen Schlüssel enorm.

Mittels der Graphentheorie, die Graphen und ihre Beziehungen zueinander untersucht, kann ermittelt werden wie viele Schlüssel bei einem symmetrischen Verschlüsselungsverfahren für eine gegebene Anzahl an Teilnehmer notwendig ist. Die Knoten stellen die Anzahl der Teilnehmer und die Kanten die notwendigen geheimen Schlüssel zwischen den Individuen da.

Es wird nun ein geschlossener Graph gebildet, bei dem jeder Knoten per Kante auf jeden anderen Knoten verweist. In diesem Beispiel wird von 1000 Personen ausgegangen. Gemäß der Formel $N = \frac{n*(n-1)}{2}$ würden, wenn vereinfacht davon ausgegangen wird, dass kein Schlüssel mehrfach verwendet werden darf, für diese Konstellation $N = \frac{1000*(1000-1)}{2} =$ 499.500 verschiedene Schlüssel benötigt.

Bei Anwendung des asymmetrischen Verfahrens dagegen werden für jeden Teilnehmer des Netzwerks lediglich ein öffentlicher und ein privater Schlüssel, also insgesamt 2n Schlüssel gebraucht. In den konkreten Zahlen unseres Beispiels ausgedrückt ergibt sich somit eine benötigte Schlüsselanzahl von 2000 [48].

Seinen Namen trägt das Verfahren wegen der asymmetrischen Verteilung der Schlüssel unter den Teilnehmern. Wollen zum Beispiel die zwei Personen S und E eine verschlüsselte Nachricht austauschen, so gibt Person E Person S ein offenes Schloss – dies stellt vereinfacht den öffentlichen Schlüssel aus dem Primzahlenprodukt und einer weiteren Zahl dar. Person S nimmt nun das offene Schloss und verschließt damit die Nachricht in einer Kiste. Das übergebene Schloss kann S nur zum Verschließen verwenden, öffnen kann S es nicht mehr. Sobald die Kiste verschlossen ist, kann nur E diese wieder öffnen, da nur E den entsprechenden Schlüssel besitzt. Der Schlüssel, mit dem das Schloss wieder geöffnet werden kann, liegt ausschließlich in der Hand von Person E und stellt ihren privaten Schlüssel dar [54].

Eine Abwandlung der Verschlüsselung von ganzen Nachrichten ist das digitale Signieren. Unter einer Signatur versteht man die Versicherung, dass eine Nachricht nach der Übertragung nicht verändert wurde. Ein Dokument, das digital signiert ist, kann von jedem gelesen werden, aber durch die Signatur ist gewährleistet, dass das Dokument nach dem Signieren durch den Verfasser von niemandem verändert wurde.

Dafür wird zunächst mittels eines Hash-Algorithmus eine eindeutige Prüfsumme aus dem gesamten Text ermittelt – sozusagen der Fingerabdruck der Nachricht. Verschlüsselt wird nun nicht mehr die gesamte Nachricht, sondern nur der Hash-Wert, der dann ans Ende der Nachricht angehängt wird. Besitzt nun Empfänger E der signierten Nachricht den öffentlichen Schlüssel des Senders S, kann er die Signatur am Ende der Nachricht entschlüsseln und erhält den durch S berechneten Hash-Wert. Empfänger E berechnet nun ebenfalls einen Hash-Wert für die empfangene Nachricht. Ist der entschlüsselte Hash-Wert identisch mit dem berechneten Hash-Wert der empfangenen Nachricht, ist klar, dass die Nachricht nach der Signatur durch den Sender S nicht mehr verändert wurde.

Dieses Verfahren wird bei vielen Blockchains verwendet, um die Identitäten von Konten und deren Transaktionen sowie deren Integrität zu gewährleisten. Die Nummernfolge, die

sich aus dem öffentlichen Schlüssel ergibt, wird auch häufig als Kontonummer, in Blockchains z. B. für Kryptowährungen wie Ethereum u. a., verwendet.

2.3.1.2 Spieltheorie – Konsens

Wesentlicher Schlüssel für die Funktionsfähigkeit eines dezentralen Netzwerks ist ein Mechanismus, der sicherstellt, dass die Knoten des Netzwerks Transaktionen verifizieren und sich außerdem alle Knoten darüber einig sind, in welcher Reihenfolge Transaktionen im Hauptbuch erfasst werden. Dieser Mechanismus wird Konsens-Mechanismus genannt und ist zentraler Bestandteil der Distributed-Ledger-Technologie. Er gewährleistet, dass Transaktionen nicht doppelt und invalide Daten, die die Integrität des Hauptbuches verletzen könnten, gar nicht gespeichert werden [88].

In den meisten Informationssystemen wird eine solche Kontrolle durch eine zentralisierte Instanz vorgenommen, wie beispielsweise das System zur Verwaltung von Konten einer Bank. Eine Überweisung von Konto A auf Konto B wird durch die Bank und die zentral in das Informationssystem eingearbeiteten Kontrollen verifiziert und die ordnungsmäßige Erfassung der Überweisung im System sichergestellt. Es existiert somit ein durch eine zentrale Instanz geführtes Kontrollsystem, dem die Teilnehmer vertrauen.

Im dezentralen System der Distributed-Ledger-Technologie wird diese zentrale Kontrollinstanz entfernt. Die Aufgabe der Verifizierung und Sicherstellung der ordnungsgemäßen Erfassung im Hauptbuch wird durch den Konsens-Mechanismus vorgenommen, der so als Mittel zur Entfernung von Intermediären, also von Zwischenakteuren, dient [58].

Um die Notwendigkeit des Konsens-Mechanismus zu erklären, greift Anthony Stevens beispielhaft auf die Geschichte der Datenbank-Applikationen zurück. Er zeigt auf, wie sich aufgrund der Verfügbarkeitsdebatte aus einem zentralen Datenbanksystem verteilte Datenbanksysteme entwickelt haben [88]:

In den Anfangszeiten wurde für eine Applikation eine Master-Datenbank eingesetzt, die alle Daten der Applikation speicherte. Da sämtliche Daten jedoch nur an einer Stelle gespeichert wurden, agierte die Datenbank als gefährlicher Single Point of Failure. Durch Backup-Mechanismen konnte man dieses Problem zwar lösen, aber um eine schnellere Rücksicherung im Schadensfall zu gewährleisten, wurde später neben der Master-Datenbank auch eine Slave-Datenbank eingesetzt, in der alle Daten in Echtzeit mit abgelegt wurden. Fiel die Master-Datenbank aus oder war defekt, konnte zügig auf die Slave-Datenbank zurückgegriffen werden. Ein direktes Speichern in die Slave-Datenbank oder das Auslesen daraus war in diesem Szenario nicht vorgesehen, da die Slave-Datenbank erst zur Verwendung kam, wenn die Master-Datenbank ausfiel.

Die Weiterentwicklung dieses Szenarios führt zu dezentralen Datenbanken, bei denen mehrere Master-Datenbanken miteinander vernetzt sind und die Daten so an unterschiedlichen Knoten gespeichert und ausgelesen werden. Dieser Aufbau führt allerdings wieder zu Problemen, denn wenn gleiche Daten innerhalb der Applikation verändert werden, kommt

es zu einem Synchronisations-Konflikt, den jemand lösen muss. Mit zunehmender Zahl an Knoten und gegebenenfalls auch Benutzern nehmen solche Konflikte immer mehr zu [88].

Die Darstellung von Stevens zeigt auf, dass eine rein technische Lösung zu Konflikten führt und dass diesen Konflikte schon bei der Erstellung des Algorithmus vorgebeugt werden muss.

Werden dezentrale Datenbanken über mehrere Unternehmen hinweg betrieben, wie es z. B. in Konsortien vorkommen kann, stellt sich die Frage, wer die Hoheit und damit die Verantwortung über das System übernimmt. In der Regel wird dies ein dominanter Konsortiums-Teilnehmer sein, der durch die Verwaltung eine gewisse Machtposition erlangt. In einer dezentralen Multi-Master-Welt sollte das Ziel sein, dass die Konsens-Bildung über die Richtigkeit der Daten sowie ihre Ablage ebenfalls dezentral und nicht durch einen einzelnen Teilnehmer gewährleistet ist.

Ferner zeigt Stevens auf, dass für die Erstellung eines solchen Algorithmus drei Bedingungen zu erfüllen sind [88]:

1. Der Algorithmus muss sicherstellen, dass sich alle Teilnehmer, die eine Instanz, also einen Master, der Datenbank kontrollieren, auf eine Reihenfolge der zu speichernden Transaktionen einigen und diese auch in der festgelegten Reihenfolge ablegen.
2. Ferner darf kein Teilnehmer die Fähigkeit haben, die definierte und gemeinsam freigegebene Reihenfolge von Transaktionen zu ändern.
3. Schon freigegebene Transaktionen, die über die Knoten hinweg verteilt werden, dürfen nicht durch einen Teilnehmer gestoppt oder verhindert werden können.

Mit diesen drei Bedingungen zeigt Stevens, dass es sich bei der Konsens-Bildung um einen Prozess handelt, der komplett durchlebt werden muss, damit es sicher zu einer Einigung kommen kann.

Verallgemeinert wird der Prozess der Konsens-Bildung in folgenden Schritten beschrieben [88]:

1. Jeder Knoten erstellt eigene Transaktionen.
2. Die so erstellten Transaktionen müssen mit allen beteiligten Knoten ausgetauscht werden, sodass alle Knoten den gleichen Datenstand haben.
3. Es wird Konsens über die Reihenfolge und die Richtigkeit der Daten zwischen den Knoten hergestellt.
4. Im letzten Schritt aktualisieren die Knoten das Ergebnis der Konsensbildung in den jeweiligen Transaktionen.

Um unterschiedlichen Anforderungen gerecht zu werden, existieren unterschiedlichste Konsens-Algorithmen. Dabei versteht man unter einem Algorithmus laut Nölle Acheson eine genaue operative Abfolge von Handlungsanweisungen bzw. Befehlen, die eine Maschine in

immer derselben Reihenfolge abarbeiten muss. Dahingegen sei ein Protokoll eine Beschreibung, wie etwas ausgestaltet sei, also die Eigenschaften, die eine Applikation oder ein Objekt besitzt [1].

Im Kontext der Distributed-Ledger-Technologie liefert das Protokoll den grundsätzlichen Rahmen, wie die Kommunikation zwischen den Knoten abläuft, bzw. wie die Daten verteilt werden. Dem Algorithmus obliegen nach den vordefinierten Regeln des Protokolls Funktionen wie die Verifizierung von Signaturen oder die Entscheidung, ob ein Block valide ist.

Die Unterscheidung zwischen Protokoll und Algorithmus ist laut Acheson dahingehend wichtig, da durch eine getrennte Betrachtung der Rahmenbedingungen (Protokoll) und der Umsetzung (Algorithmus) unterschiedlichste kreative Ansätze gefunden werden können [1]. Diese Kreativität spiegelt sich auch in der Masse der existierenden Konsens-Algorithmen wieder, die unterschiedliche Eigenschaften besitzen, aber trotzdem den von Stevens verallgemeinerten Prozessschritten folgen.

Überblick Konsens-Algorithmen

In einem dezentralen System können drei mögliche Fehler auftreten. Erstens: Eine Komponente des Systems stellt seine Funktion ein und ist damit nicht mehr für alle Teilnehmer erreichbar. Zweitens: Nachrichten, über die die Knoten miteinander kommunizieren, werden zwar versendet, kommen aber nicht bei den Zielknoten an. Drittens: Es werden Nachrichten mit gefälschtem Inhalt übermittelt. Dieser Fehlertyp wird byzantinischer Fehler genannt [50]. Der byzantinische Fehler beruht auf dem Gedankenspiel der byzantinischen Generäle, das von Lamport in seinem Paper „The Byzantine Generals Problem“ veröffentlicht wurde: Drei Generäle wollen eine Stadt erobern. Dabei sind sie soweit voneinander entfernt, dass eine direkte Kommunikation nicht möglich ist – sie können lediglich Nachrichten über Boten austauschen. Die Stadt kann aber nur dann besiegt werden, wenn alle Generäle gleichzeitig angreifen. Sie müssen sich also auf eine einheitliche Strategie einigen, nämlich angreifen oder nicht angreifen [56]. Bezogen auf die Problematik in verteilten Systemen sind die Generäle die Knoten, die am Netzwerk partizipieren und die Nachrichten sind ebenfalls Nachrichten, die in verteilten Systemen unter den Knoten ausgetauscht werden, um das Netzwerk aktiv zu halten. Lambert geht weiter davon aus, dass sich unter den Generälen Verräter befinden, die an der Kommunikation des Netzwerks teilnehmen, aber das Ziel verfolgen, den Verbund zu destabilisieren. Alle Generäle wie auch der Kommandant sind in diesem Beispiel gleichberechtigt, so dass keine Entscheidung per Hierarchie herbeigeführt werden kann. General 1, der Kommandant, gibt den Befehl zum Angriff und schickt diesen Befehl per Bote an General 2 und 3. General 3 stellt in unserem Gedankenspiel den Verräter dar. Er bekommt die Nachricht für den Angriff, gibt aber an General 2 weiter, man solle sich zurückziehen. General 2 hat nun kaum eine Möglichkeit, zu entscheiden, ob der Angriff durchzuführen ist oder nicht – er hat widersprüchliche Befehle erhalten und weiß nicht, wem zu trauen ist (Abb. 2.4).

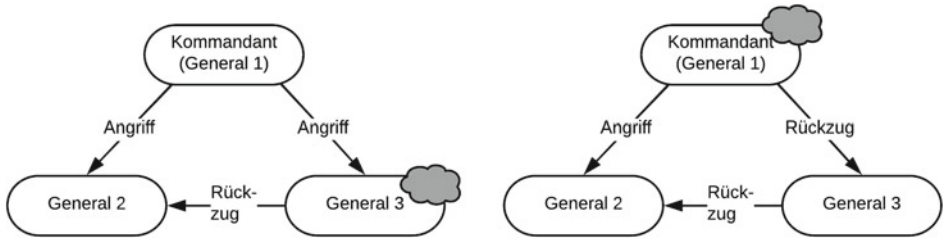

Abb. 2.4 Byzantine Generäle. (In Anlehnung an [56])

Das Gedankenspiel kann in unterschiedlichen Richtungen durchgespielt werden. Wenn beispielsweise der Kommandant der Verräter ist und beiden Generälen unterschiedliche Nachrichten schickt.

Lambert führt weiter aus, dass es bei einem Netzwerk aus drei Knoten nie zu einer Lösung kommt, die den Verräter identifiziert oder die Entscheidung zugunsten des Netzwerks durchführt. Bei mehr als drei Generälen dagegen zeigt Lambert auf, dass eine Konsensbildung über die Mehrheit der Generäle möglich ist, wenn mindestens $3\,m + 1$ Generäle im Netzwerk existieren, wobei m die Menge der Verräter im Verbund darstellt. Die Anzahl der Verräter muss also kleiner als ein Drittel sein.

Diese Art des Fehlers, also das manipulierende Handeln von Netzteilnehmern, tritt vor allem in dezentralen Systemen mit vielen unbekannten bzw. sich nicht per se vertrauende Knoten bzw. Teilnehmern auf. Vor allem dann, wenn das Individuum ein größeres Individualinteresse an einer Fehlentscheidung hat, als es Vorteile aus dem gemeinsamen Betrieb des Netzwerks zieht. Aus diesem Grund müssen, neben dem Versprechen eines funktionierenden Netzwerkes zusätzliche Anreize zur Einhaltung der Konsensregeln geboten werden.

Um nun einen Überblick über die unterschiedlichen Konsens-Algorithmen zu bekommen, können die drei eingangs dargestellten Fehlertypen herangezogen werden. Bei den Fehlertypen eins und zwei handelt es sich um Fehler, die in einem dezentralen System leicht identifiziert werden können; Es sind binäre Fehler. Ein binärer Fehler kann zwei Zustände annehmen: Ein Knoten des verteilten Systems funktioniert ordnungsgemäß oder er funktioniert nicht bzw. Nachrichten werden empfangen oder werden nicht empfangen.

Der byzantinische Fehler lässt sich nicht so einfach erkennen, da es sich hierbei nicht um einen binären Fehler handelt, sondern um einen ordnungsgemäß funktionierenden Knoten des Netzes, der Nachrichten gemäß Protokoll übermittelt, allerdings mit manipulierten, fehlerhaften Inhalten. Zur Erkennung des byzantinischen Fehlers wurden spezielle Konsens-Algorithmen entwickelt, sodass es sich anbietet, die Algorithmen in die Kategorien „Byzantinischer Fehler“ und „Nicht byzantinischer Fehler“ zu unterteilen (Siehe Abb. 2.5). Die Kategorie „Nicht byzantinischer Fehler“ kann gemäß Preethi Kasireddy auch als einfache Fehlertoleranz bezeichnet werden und wird oft bei kontrollierten Umgebungen, in denen alle Beteiligten sich kennen und vertrauen, angewandt [50].