



**HOCHSCHULE OSNABRÜCK**

UNIVERSITY OF APPLIED SCIENCES

Institut für Duale Studiengänge

PRÜFUNGSLEISTUNG

IM STUDIENGANG WIRTSCHAFTSINFORMATIK

# Blockchain als verteilte Datenbank

Eingereicht von:

Matthias Fischer (700643)  
Fabian Hagengers (701292)

Studiengruppe:

15DWF1

Betreuer:

Prof. Dr. Heiko Tapken

Modul:

Datenbank-Engineering

Abgabedatum:

02.04.2018

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>II</b>
<b>Tabellenverzeichnis</b>	<b>III</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Blockchain in der Theorie</b>	<b>1</b>
2.1 Transaktionsablauf . . . . .	2
2.2 Blockaufbau . . . . .	3
2.3 Konsensbildung . . . . .	4
2.3.1 Proof-Of-Work . . . . .	5
2.3.2 Proof-Of-Stake . . . . .	5
2.4 Zusammenfassung . . . . .	5
<b>3 Blockchain in der Praxis am Beispiel Multichain</b>	<b>6</b>
3.1 Installation, Einrichtung und Generierung der Blockchain . . . . .	6
3.2 Installation und Einrichtung des MultiChain-Explorers . . . . .	8
3.3 Anbinden eines zweiten Knoten an die Blockchain . . . . .	9
3.4 Betrachtung der Blockchain im Explorer . . . . .	12
3.5 Manuelle Durchführung einer Transaktion . . . . .	14
3.6 Speichern von Daten . . . . .	15
<b>4 Kritische Reflexion</b>	<b>17</b>
<b>5 Fazit</b>	<b>17</b>
<b>Literaturverzeichnis</b>	<b>18</b>
<b>Eidesstattliche Erklärung</b>	<b>19</b>

## Abbildungsverzeichnis

1	Zentrales und dezentrales Netzwerk. Quelle: cryptocoin20.com . . . . .	2
2	Vereinfachter Transaktionsablauf. Quelle: Fraunhofer FIT . . . . .	2
3	Blockaufbau. Quelle: Satoshi Nakamoto . . . . .	3
4	Hash-Baum. Quelle: David Göthberg . . . . .	4
5	Startseite des Explorer . . . . .	9
6	Ausgabe des Kommandos multichaind db -daemon im Terminal des Servers . . .	10
7	Initialisierung der Blockchain auf dem Client . . . . .	10
8	Vergleich des /.multichain/db-Ordners auf dem Server und dem Client . . . . .	11
9	Vergleich des wallet-Ordners auf dem Server und dem Client . . . . .	12
10	Übersicht der Blöcke der Blockchain . . . . .	13
11	Ansicht eines Blocks im Explorer . . . . .	13
12	Detail-Ansicht einer Transaktion . . . . .	14
13	Explorer-Ansicht eines Knotens . . . . .	15
14	Unbestätigte Transaktion des Servers an den Client . . . . .	15
15	Betrachtung der Transaktion des Servers an den Client . . . . .	16
16	Transaktion im Explorer, die Daten enthält . . . . .	16

## Tabellenverzeichnis

1	Konfigurierbare Parameter der Explorer-Blockchain-Verbindung . . . . .	9
---	--	---

## 1 Einleitung

Seit der Schöpfung der ersten Bitcoins 2009 steigt die Popularität der Kryptowährungen und der Blockchain-Technologie stetig an.<sup>1</sup> Besonders die wirtschaftliche Institutionen beschäftigen sich intensiver mit dem Thema Blockchain, da diese Technologie die Möglichkeit bietet Dokumente fälschungssicher, irreversible und dezentral zu speichern.<sup>2</sup> Satoshi Nakamoto veröffentlichte 2008 ein Artikel mit dem Titel „Bitcoin: A Peer-to-Peer Electronic Cash System“, indem eine Währung ohne zentrale Entität erläutert wurde.<sup>3</sup> Diese Ausarbeitung stellt die Basis jeder heutigen Blockchain. Eine Blockchain ist eine auf Kryptographie basierende verkettete Liste von Blöcken. In den Blöcken werden die Daten der Blockchain gespeichert. In den meisten Fällen handelt es sich hierbei um Überweisungen der Kryptowährung von einer zur anderen Person, jedoch besteht auch die Möglichkeit andere Daten in diesen Blöcken zu speichern.

Diese Ausarbeitung verfolgt zwei Hauptziele. Zunächst soll das theoretische Grundwissen der Blockchain-Technologie als Datenbank erläutert werden. Hierbei steht die strukturelle Aufbau und die Konsensbildung der Blöcke im Vordergrund. Aufbauend auf diesem Wissen verfolgt diese Ausarbeitung das Ziel eine Blockchain mit der Software Multichain<sup>4</sup> zu erstellen und anschließend Transaktionen durchzuführen. Hierbei steht die Installation, Konfiguration und die Verwendung der Blockchain im Vordergrund.

In Kapitel 2 wird das theoretische Fundament gebildet. Es erläutert den Transaktionsablauf, den Blockaufbau und die Konsensbildung der Blockchain. Aufbauend auf der Theorie folgt in Kapitel 3 die praktische Anwendung der Blockchainsoftware Multichain. Hierbei wird erläutert wie die Software installiert, konfiguriert und anschließend verwendet wird. Die Ausarbeitung schließt mit einer kritischen Betrachtung in Kapitel 4 und einem Fazit in Kapitel 5 ab.

## 2 Blockchain in der Theorie

Die Blockchain ist eine verkettete Liste von Blöcken, welche mithilfe von kryptographischen Verfahren irreversible und manipulationsfrei verbunden sind. Ein Block enthält Transaktionen in denen Daten gespeichert sind. Eine Blockchain besteht nicht aus einer zentralen Datenbank, sondern wird in einem dezentralen Peer-To-Peer Netzwerk gespeichert. In einem Peer-To-Peer Netzwerk werden die Netzwerkteilnehmer als Nodes bezeichnet.<sup>5</sup> In einem Blockchain-Netzwerk hat jeder Teilnehmer alle Daten zur Verfügung. Dies hat den Vorteil, dass einerseits die Ausfallsicherheit des Netzwerkes gesichert ist und andererseits keiner einzelnen Entität vertraut werden muss. Dadurch entsteht ein geringes Manipulationsrisiko. In Abbildung 1 ist der Unterschied zwischen einem zentralen und dezentralen Netzwerk visualisiert.

---

<sup>1</sup>[Neugebauer, 2018, S. 312]

<sup>2</sup>[Neugebauer, 2018, S. 312]

<sup>3</sup>[Satoshi Nakamoto, 2008, S. 1]

<sup>4</sup><https://www.multichain.com/>

<sup>5</sup>[Korzun and Gurtov, 2013, S. 5]

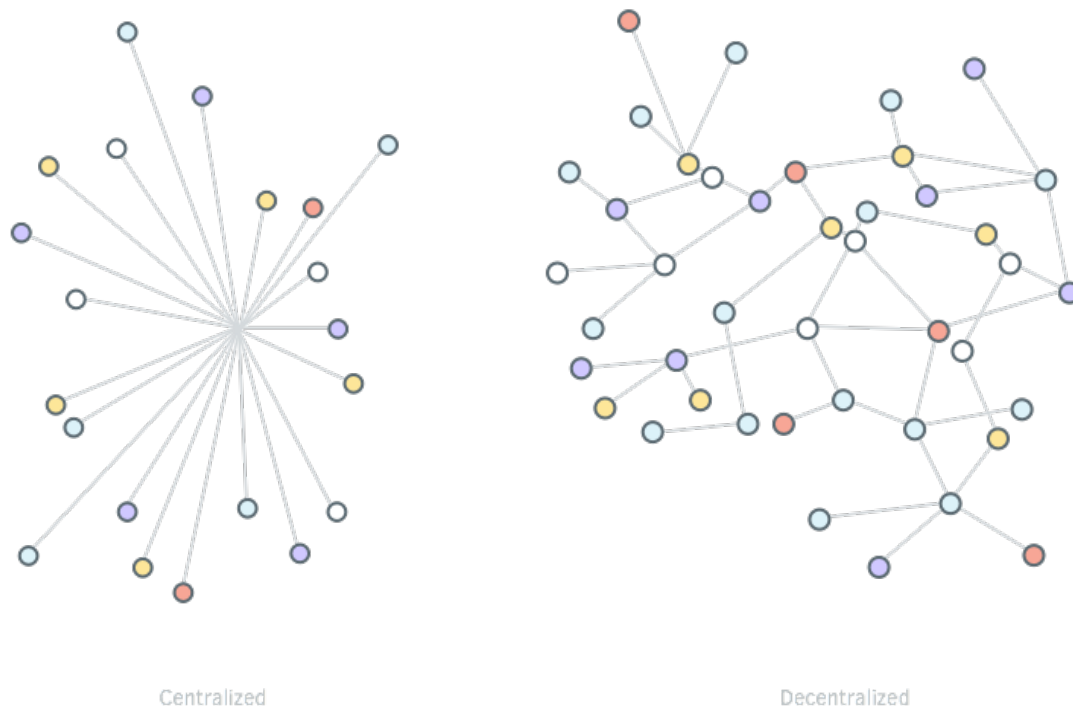


Abbildung 1: Zentrales und dezentrales Netzwerk. Quelle: [cryptocoin20.com](http://cryptocoin20.com)

Folgend wird erläutert wie in einer Blockchain die Transaktionen in die Blockchain eingebettet werden. Anschließend wird der Aufbau eines Blockes erläutert. Daraufhin folgt die Erläuterung der Konsensbildung der Blockchain.

## 2.1 Transaktionsablauf

Eine Transaktion ist eine Arbeitseinheit, welche eine bestimmte Funktion erfüllt. In Bezug auf Datenbanksysteme bezeichnet eine Transaktion eine Folge von Datenverarbeitungsbefehlen, die vom System atomar ausgeführt werden.<sup>6</sup> Im Kontext der Blockchain bezeichnet eine Transaktion die Persistierung von Daten, wie beispielsweise einer Überweisung von Person a zu Person b.<sup>7</sup> Abbildung 2 beschreibt den einfachen Transaktionsablauf einer Blockchain.

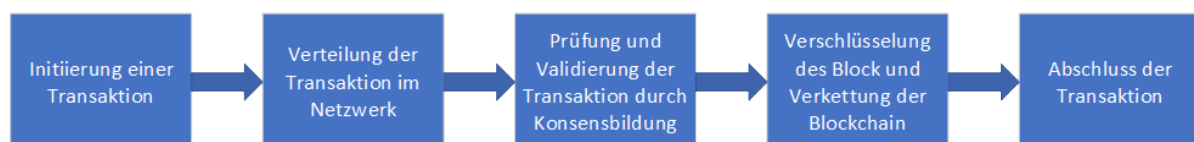


Abbildung 2: Vereinfachter Transaktionsablauf. Quelle: Fraunhofer FIT

<sup>6</sup>[Kemper and Eickler, 2006, S. 301]

<sup>7</sup>[Satoshi Nakamoto, 2008, S. 2]

Zunächst wird die durchzuführende Transaktion im Netzwerk verteilt. Anschließend fassen die Netzwerkteilnehmer mehrere Transaktionen zu einem Block zusammen und versuchen durch eine Konsensbildung die Transaktionen zu validieren. Die Validierung findet bei der Blockchain-Technologie mit dem sogenannten Proof-Of-Work statt, welche es fordert, dass aus den Transaktionen und dem vorherigen Block ein Hash mit einer bestimmten Voraussetzung gebildet wird (beispielsweise zehn führende Nullen). Nachdem ein Netzwerkteilnehmer einen validen Hash ermittelt hat, wird der Block an die Kette angehängen und anschließend die veränderte Kette im Netzwerk verteilt. Alle anderen Teilnehmer überprüfen die Erweiterung der Kette mithilfe einer Validierung der Blockhashs. Die Nodes überprüfen auch, ob eine Transaktion schon in mehreren Blöcken vorhanden ist oder nicht. Je nach Ergebnis der Validierung wird anschließend die neue Blockchain angenommen oder abgelehnt und die Anpassungen zurückgesetzt.<sup>8</sup>

## 2.2 Blockaufbau

Grundsätzlich besteht ein Block einer Blockchain aus den Transaktionen und Informationen über den aktuellen Block. Mithilfe der Merkle-Root wird aus den einzelnen Transaktionen ein Hashwert ermittelt, jener anschließend in Kombination mit dem vorherigen Blockhash und der frei wählbaren Nonce den Hash des aktuellen Blockes bildet. Mithilfe dieser Einbindung des vorherigen Hash wird die Kette fest miteinander verbunden und die Integrität der Blockchain gewährleistet. In Abbildung 3 ist ein Schema der verketteten Blöcke dargestellt.

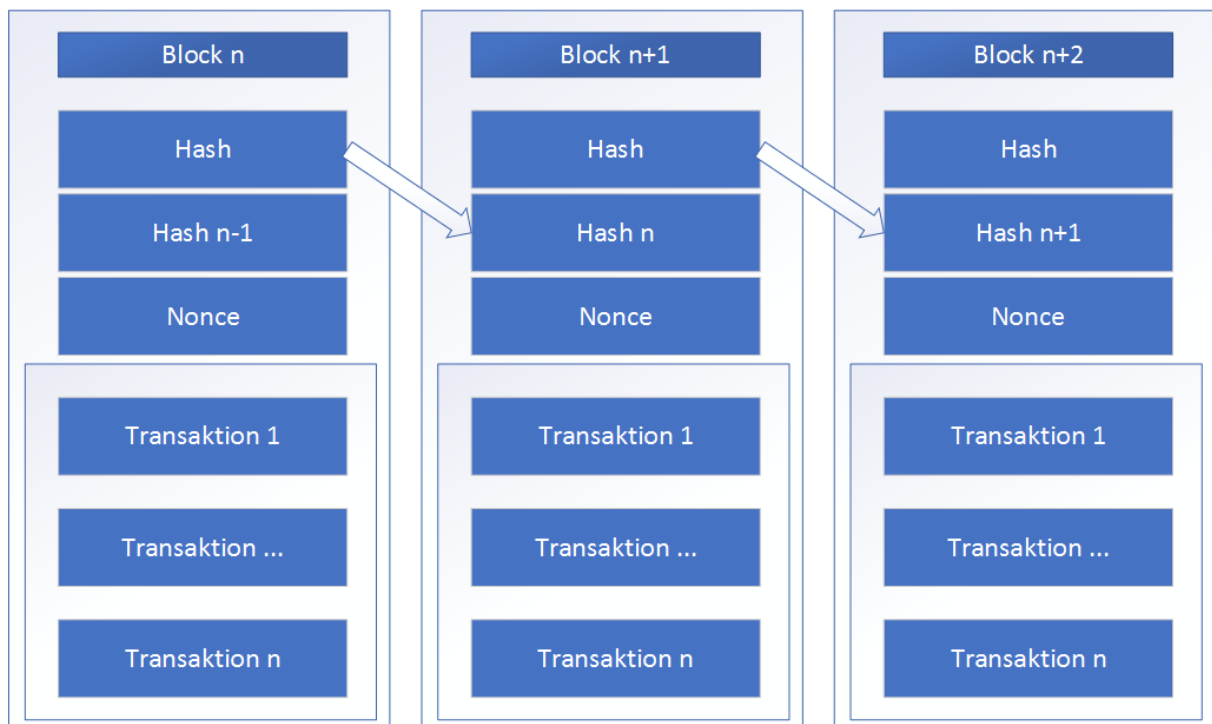


Abbildung 3: Blockaufbau. Quelle: Satoshi Nakamoto

<sup>8</sup>[Satoshi Nakamoto, 2008, S. 2-4]

Die Merkle-Root oder auch Hash-Baum wird für die Sicherstellung der Integrität der Transaktionen verwendet. In Abbildung 4 ist eine Merkle-Root dargestellt. Zunächst werden alle Transaktionen gehasht. Daraufhin werden zwei gehashte Transaktionen erneut gehasht. Diese Ergebnisse werden anschließend erneut in Zweierpärchen gehasht. Dieser Vorgang wird solange wiederholt, bis nur noch ein Hash übrig ist. Das Ergebnis dieses Verfahren wird als Top-Hash bezeichnet und fließt anschließend in den Blockhash mit ein.<sup>9</sup>

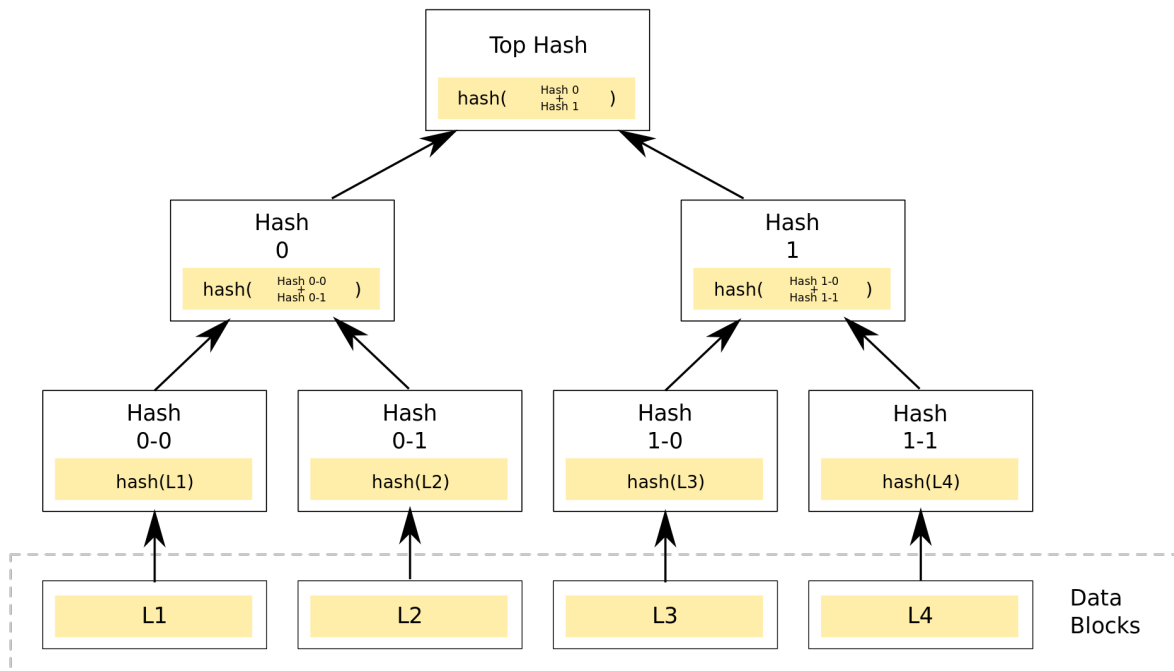


Abbildung 4: Hash-Baum. Quelle: David Göthberg

## 2.3 Konsensbildung

Einer der wesentlichen Aspekte des Blockchain-Konzeptes ist die Konsensbildung. Hierbei soll ein Netzwerkteilnehmer beweisen können, dass er berechtigt ist die Blockchain anpassen zu dürfen. Für alle anderen Teilnehmer soll dieser Beweis einfach validierbar sein. Ein weiterer Punkt der Konsensbildung ist die vereinheitlichte Anpassung während der Benutzung der Blockchain. Im den folgenden Kapiteln werden die beiden relevanten Varianten der Konsensbildung vorgestellt.

<sup>9</sup>[Satoshi Nakamoto, 2008, S. 4]



### 2.3.1 Proof-Of-Work

Für Kryptowährungen wird üblicherweise die Proof-Of-Work Variante genutzt. Hierbei muss ein Teilnehmer Arbeit aufwenden, welche anschließend von den anderen Teilnehmern überprüft werden kann.<sup>10</sup>

Für die Bitcoin Blockchain müssen beispielsweise Blockhashs mit einer spezifischen Anzahl von führenden Nullen generiert werden. Da eine Hashfunktion eine Einwegfunktion ist, bleibt dem Teilnehmer nichts anderes übrig, als solange den freiwählbaren Teil des Blockes, also den Nonce, anzupassen und anschließend die Hashfunktion auszuführen, bis der gewünschte Hash vorhanden ist.<sup>11</sup>

In einer Kryptowährung wie Bitcoin wird in diesem Schritt auch eine Ausschüttung der Währung durchgeführt. Das bedeutet der Teilnehmer, der als erstes die Konsensbildung löst, erhält einen definierten Betrag von der Kryptowährung. Da jedoch mit steigender Nachfrage auch die Anzahl der Netzwerkteilnehmer, bei einer Kryptowährung auch als Miner bezeichnet, steigt, sinkt gleichzeitig die Dauer der Blockerstellung. Um dieser Entwicklung entgegenzuwirken besteht die Möglichkeit die Anforderung an die Konsensbildung regelmäßig zu steigern und somit den steigenden Rechenleistungen entgegenzuwirken.<sup>12</sup>

Die Proof-Of-Work Variante ist für öffentliche Blockchains geeignet, da es keine verifizierte Teilnehmer benötigt und trotzdem die Integrität der Blockchain gewährleistet.

### 2.3.2 Proof-Of-Stake

Für private Blockchains dagegen ist es nicht immer sinnvoll CPU-Ressourcen für die Konsensbildung zu nutzen. Eine Alternative hierfür wäre das Proof-Of-Stake Verfahren. Hierbei haben einige Nodes besondere Berechtigungen die Blockchain anzupassen und es wird per Zufall ausgewählt, welcher Teilnehmer die Transaktionen an die Blockchain anhängen soll. Die Nodes mit der Berechtigung neue Blöcke an die Blockchain anzuhängen, müssen jedoch in dieser Variante zuvor definiert werden und damit entfällt auch die Unabhängigkeit von einer zentralen Instanz.<sup>13</sup>

## 2.4 Zusammenfassung

Eine Blockchain ist eine verkettete Liste von Blöcken in einem verteilten Peer-To-Peer Netzwerk. Jeder Netzwerkteilnehmer besitzt eine komplette Kopie der Datensätze. Um Daten in das Netzwerk einspeisen zu können, muss der Nutzer eine Transaktion per Broadcast an das Netzwerk

---

<sup>10</sup>[Neugebauer, 2018, S. 314]

<sup>11</sup>[Neugebauer, 2018, S. 314]

<sup>12</sup>[Neugebauer, 2018, S. 315]

<sup>13</sup>[Neugebauer, 2018, S. 315–316]

senden. Die Netzwerkteilnehmer, auch Nodes genannt, bündeln mehrere Transaktionen und führen je nach Blockchaineigenschaften entweder Proof-Of-Work oder Proof-Of-Stake zur Konsensbildung durch. Hierbei werden zunächst die Transaktionen mithilfe eines Hash-Baums zu einem Hash zusammengefasst. Anschließend wird dieser Hash mit dem Hash des vorherigen Blockes kombiniert. Im Falle der Proof-Of-Work Variante wird danach ein Nonce-Wert gesucht, welcher in Kombination mit den anderen Hashs einen Hash ergibt, der den Ansprüchen der Blockchain genügt. Im Falle der Proof-Of-Stake-Variante wird einfach der Top-Hash der Transaktionen mit dem vorherigen Blockhash kombiniert. Anschließend wird in beiden Verfahren die erweiterte Blockchain im Netzwerk verteilt. Jeder andere Teilnehmer validiert daraufhin die Blockchain.

Da die Blockchain dezentral aufgebaut ist, ergibt sich ein niedriges Ausfallrisiko. Ein weiterer Vorteil der Dezentralität ist das es keine Vertrauensinstanz existieren muss und durch den Proof-Of-Work trotzdem die Integrität der Daten gesichert ist. Eine Blockchain eignet sich aus diesen Gründen für eine öffentliche Währung. Jedoch können Blockchains auch in Unternehmen verwendet werden, da Daten manipulationssicher gespeichert werden können. Diese Eigenschaft könnte beispielsweise in der Rückverfolgung von Lebensmitteln genutzt werden.

### 3 Blockchain in der Praxis am Beispiel Multichain

Im nun folgenden Praxisteil soll eine Blockchain aufgesetzt, verwaltet und mit dieser verschiedene Aktionen durchgeführt werden, die einige im Theorieteil vorgestellten Aspekte praktisch vorführen.

#### 3.1 Installation, Einrichtung und Generierung der Blockchain

Zur Realisation des Praxisteils wird MultiChain verwendet. MultiChain ist eine offene Software mit deren Hilfe schnell und einfach Blockchains erstellt und verwaltet werden können.<sup>14</sup> Zudem bietet MultiChain einen eigens entwickelten Web-Explorer, um die Blockchain leichter nachvollziehen zu können.<sup>15</sup> Um die Dezentralität der Blockchain darzustellen, müssen mindestens zwei Nodes die Blockchain verwenden. Um dies allerdings auf einem Computer realisieren zu können und um den Stand des Praxisteils exportierbar zu machen, wird der Praxisteil mit Hilfe zweier virtueller Maschinen verwirklicht. Auf der ersten virtuellen Maschine wird die Blockchain erstellt (im Folgenden Server genannt), sodass die andere virtuelle Maschine (im Folgenden Client genannt) sich dieser Blockchain später anschließen kann.

Um die Blockchain nun auf dem Server einzurichten, muss zunächst MultiChain installiert werden. MultiChain ist sowohl für Linux als auch für Windows und Mac verfügbar. Da beide virtuellen Maschinen allerdings eine Linux-Distribution (Ubuntu) besitzen, wird lediglich auf die

---

<sup>14</sup><https://www.multichain.com/>

<sup>15</sup><https://github.com/MultiChain/multichain-explorer>

### 3 Blockchain in der Praxis am Beispiel Multichain

Installation für Linux eingegangen. Dazu muss zunächst von der MultiChain-Webseite eine Datei mit den Sourcen heruntergeladen werden, die dann entpackt werden muss. Um MultiChain einfach über das Terminal bedienbar zu machen, werden zudem drei Dateien in den Ordner `/usr/local/bin` verschoben. Diese Schritte können einfach mittels folgender Zeilen im Terminal durchgeführt werden<sup>16</sup>:

```
1 su (enter root password)
2 cd /tmp
3 wget https://www.multichain.com/download/multichain-1.0.4.tar.gz
4 tar -xvzf multichain-1.0.4.tar.gz
5 cd multichain-1.0.4
6 mv multichaind multichain-cli multichain-util /usr/local/bin
7 exit
```

Nachdem MultiChain nun installiert ist, kann die Blockchain erstellt werden. Jede Blockchain muss zudem benannt werden. Für dieses Praxisbeispiel wird die Blockchain „db“ genannt. Mittels folgenden Kommandos kann die Blockchain erstellt werden.

```
1 multichain-util create db
```

Dadurch wird in dem Verzeichnis `/.multichain` nun ein neuer Ordner mit dem Namen der Blockchain angelegt, der die gesamte Blockchain inklusive Daten und Einstellungen enthält. Als nächstes können, falls gewollt, die Parameter der Blockchain angepasst werden. Diese befinden sich in der Datei `params.dat` des jeweiligen Blockchain-Ordners und können in diesem Fall mittels folgenden Kommandos bearbeitet werden.

```
1 sudo nano ~/.multichain/db/params.dat
```

Eine Liste aller verfügbaren Parameter und deren Bedeutung befindet sich ebenfalls auf der Webseite von MultiChain.<sup>17</sup> Erwähnenswert in Bezug auf Kryptowährungen ist hier der Parameter `initial-block-reward` mittels dessen eingestellt werden kann, wie viel Währung ein Node als Belohnung für das erfolgreiche Validieren eines Blocks erhält.

Die Blockchain ist nun erstellt und eingerichtet allerdings noch nicht generiert, d. h. sie besitzt noch keinerlei Blöcke und entsprechend keine Daten. Um die Blockchain nun also zu generieren, muss ein erster Block vom Server validiert werden. Um die Validierung und somit das Errechnen eines gültigen Hashes für den nächsten Block zu starten, wird in diesem (bezogen auf diese Blockchain) folgendes Kommando verwendet.

```
1 multichaind db -daemon
```

Sobald der erste Block validiert ist, ist die Blockchain generiert.

<sup>16</sup><https://www.multichain.com/download-install/>

<sup>17</sup><https://www.multichain.com/developers/blockchain-parameters/>

## 3.2 Installation und Einrichtung des MultiChain-Explorers

Um die Blockchain auch übersichtlich und verständlich betrachten zu können wird nun der von MultiChain zur Verfügung gestellte Explorer installiert. Die Sourcen für den Explorer befinden sich in einem GitHub-Repository von MultiChain und können (sofern Git auf dem Server installiert ist) mittels folgenden Kommandos heruntergeladen werden:

```
1 git clone https://github.com/MultiChain/multichain-explorer
```

Damit der Explorer lauffähig ist, sind allerdings einige zusätzlichen Programme notwendig, die über folgende Kommandos heruntergeladen und installiert werden können:

```
1 sudo apt-get install sqlite3 libsqlite3-dev
2 sudo apt-get install python-dev
3 sudo apt-get install python-pip
4 sudo pip install --upgrade-pip
5 sudo pip install pycrypto
```

Als nächstes muss in den nun angelegten Ordner multichain-explorer gewechselt und dort das Skript setup.py mit dem Argument install ausgeführt werden, um den Explorer zu installieren.

```
1 cd multichain-explorer
2 sudo python setup.py install
```

Als nächstes muss der Explorer mit der generierten Blockchain kommunizieren können, um die Daten dieser später anzeigen zu können. Jede Blockchain besitzt einen sog. RPC-Port, der in der Datei params.dat im Ordner der Blockchain festgelegt ist. Damit die Blockchain mit dem Explorer kommunizieren kann, wurde bereits beim Erstellen dieser die Datei multichain.conf im Ordner der Blockchain angelegt. In diese Datei muss nun der RPC-Port eingetragen werden. Dies lässt sich einfach über folgende Kommandos realisieren (wobei AUSGEGEBENER\_PORT entsprechend ersetzt werden muss):

```
1 cd ~/.multichain/db
2 grep rpc params.dat
3 echo "rpcport=AUSGEGEBENER_PORT" >> multichain.conf
```

Abschließend muss der Explorer noch mit der Blockchain verknüpft werden. Hierzu muss zunächst wieder in den Ordner multichain-explorer navigiert werden. Es muss nun eine Konfigurationsdatei eigens für die erstellte Blockchain angelegt und konfiguriert werden. MultiChain-Explorer bietet hierzu eine vorgefertigte Konfigurationsdatei (chai1.example.conf) die einfach kopiert und unter dem Namen der Datenbank (db.conf) gespeichert werden kann. Innerhalb der Datei können bzw. müssen nun folgende Parameter eingestellt werden:

Nachdem die Parameter entsprechend angepasst wurden, kann die Datei mit der Tastenkombination STRG + X verlassen werden, wobei auf die Nachfrage, ob die Änderungen gespeichert werden sollen mit Y + Enter und danach der Dateiname mit Enter bestätigt werden müssen.

### 3 Blockchain in der Praxis am Beispiel Multichain

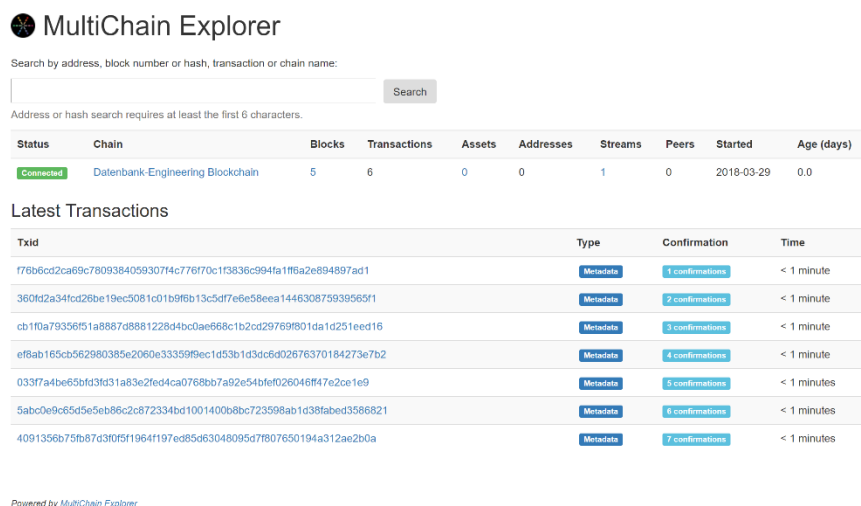
Parameter	Beschreibung
port	Port, über den die Seite im Browser erreichbar sein soll
host	IP des Gerätes, das den Explorer aufrufen können soll (0.0.0.0 für alle Geräte)
dirname	Absoluter Pfad zum Ordner der Datenbank (hier: /.multichain/db)
chain	Name der Blockchain, der im Explorer angezeigt werden soll
connect-args	Speicherort der Explorer-Datenbank (hier: db.explorer.sqlite)

Tabelle 1: Konfigurierbare Parameter der Explorer-Blockchain-Verbindung

Nun kann der Explorer mit folgendem Kommando gestartet werden:

```
1 sudo python -m Mce.abe --config db.conf
```

Der Explorer liest dann zunächst die bislang getätigten Transaktionen und validierten Blöcke in die eigene Datenbank ein. Der Explorer kann dann über die Adresse des Servers und dem angegebenen Port aufgerufen werden (s. Abbildung 5).



**MultiChain Explorer**

Search by address, block number or hash, transaction or chain name:

Address or hash search requires at least the first 6 characters.

Status	Chain	Blocks	Transactions	Assets	Addresses	Streams	Peers	Started	Age (days)
Connected	Datenbank-Engineering Blockchain	5	6	0	0	1	0	2018-03-29	0.0

**Latest Transactions**

Txid	Type	Confirmation	Time
f76b6cd2ca69c7809384059307f4c776f70c1f3836c994fa1f6a2e694897ad1	Metadata	1 confirmations	< 1 minute
360fd2a34fcd28be19ec5081cd1b9fb13c5d7e6e58eea144630875939565f1	Metadata	2 confirmations	< 1 minute
cb1f0a79356f51a8887d8881228d4b0cae668c1b2cd29769801da1d251eed16	Metadata	3 confirmations	< 1 minute
ef8ab165cb562980385e2060e333599ec1d53b1d3dc6d026730184273e7b2	Metadata	4 confirmations	< 1 minute
033f7a4be65b3d3d31a83e2fed4ca0768bb7a92e54bfe026046f47e2ce1e9	Metadata	5 confirmations	< 1 minutes
5abc0e9c65d5e5eb86c2c872334bd1001400b8bc723598ab1d38faded3586821	Metadata	6 confirmations	< 1 minutes
4091356b75b87d30f5f1964f197ed8563048095d7807650194a312ae2b0a	Metadata	7 confirmations	< 1 minutes

Powered by MultiChain Explorer

Abbildung 5: Startseite des Explorer

### 3.3 Anbinden eines zweiten Knoten an die Blockchain

Um nun unseren Client zum zweiten Knoten der Blockchain zu machen, müssen wir auch auf diesem zunächst MultiChain installieren. Voraussetzung für die Verbindung ist nun, dass sich die beiden Computer (bzw. hier die beiden virtuellen Maschinen) im selben Netzwerk befinden. Als der Server als Knoten der Blockchain gestartet wurde, wurde im Terminal des Servers folgendes ausgegeben:

Hier wird mitgeteilt, wie man sich mit anderen Computern im selben Netzwerk mit der Blockchain verbinden kann. Dazu muss man lediglich den Befehl `multichaind` eingeben gefolgt von

### 3 Blockchain in der Praxis am Beispiel Multichain

```

hs@hs-VirtualBox:~$ multichaind db -daemon

MultiChain 1.0.4 Daemon (latest protocol 10010)

Starting up node...

Other nodes can connect to this node using:
multichaind db@192.168.2.82:7357

Listening for API requests on port 7356 (local only - see rpcallowip setting)

Node ready.
  
```

Abbildung 6: Ausgabe des Kommandos `multichaind db -daemon` im Terminal des Servers

dem Namen der Blockchain, einem @-Zeichen, der IP des Servers, einem Doppelpunkt und einem gesetzten Port. In diesem Falle setzt sich daraus folgendes Kommando zusammen:

```
1 multichaind db@192.168.2.82:7357
```

Wenn dieses Kommando nun im Terminal des Clients eingegeben wird, erscheint allerdings ein Hinweis, dass Blockchain selbst zwar empfangen und erfolgreich auf dem Client initialisiert wurde, der Client nicht die nötigen Rechte zum Verbinden der Blockchain besitzt.

```

hs@hs-VirtualBox:~$ multichaind db@192.168.2.82:7357

MultiChain 1.0.4 Daemon (latest protocol 10010)

Retrieving blockchain parameters from the seed node 192.168.2.82:7357 ...
Blockchain successfully initialized.

Please ask blockchain admin or user having activate permission to let you connect and/or transact:
multichain-cli db grant 18tJkQWoVaoMNVsH4CN5ruazXnLjnGPUnyAdGA connect
multichain-cli db grant 18tJkQWoVaoMNVsH4CN5ruazXnLjnGPUnyAdGA connect,send,receive
  
```

Abbildung 7: Initialisierung der Blockchain auf dem Client

MultiChain stellt eine Blockchain generell so ein, dass sich niemand ohne explizites Recht mit einer Blockchain verbinden kann. Dies kann allerdings vor der Generierung der Blockchain über den Parameter `anyone-can-connect` in der `params.dat` der jeweiligen Blockchain angepasst werden. Neben der Meldung über die fehlenden Rechte zeigt das Terminal des Clients allerdings auch direkt das Kommando an, das auf dem Server eingegeben werden muss, um dem Client die Rechte zum Verbinden zu geben:

```
1 multichain-cli db grant 18tJkQWoVaoMNVsH4CN5ruazXnLjnGPUnyAdGA connect
```

Auffällig ist hierbei die lange, zufallsgenerierte Zeichenkette. MultiChain vergibt jedem Knoten eine einzigartige ID und die hier angezeigte ID ist die ID, die für den Client generiert wurde.

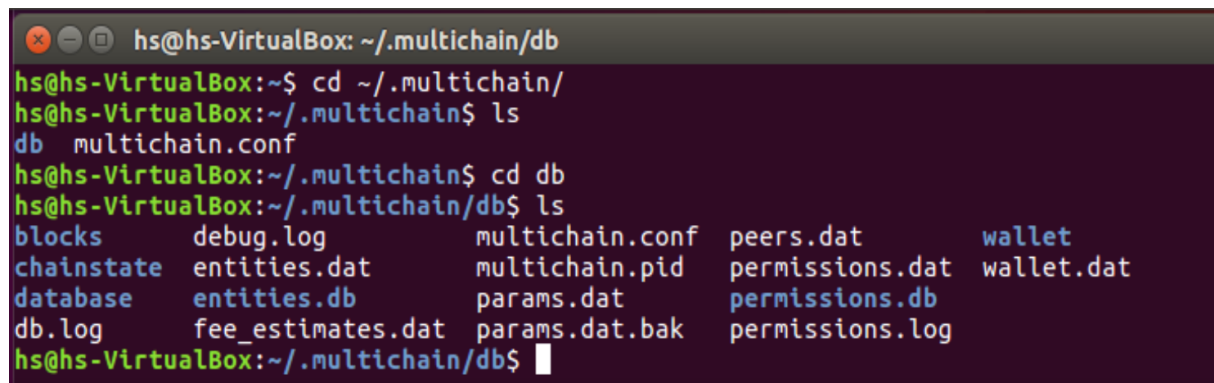
### 3 Blockchain in der Praxis am Beispiel Multichain

Nachdem wir das Kommando im Terminal des Servers eingegeben haben, kann sich der Client mit der Blockchain verbinden und als Knoten gestartet werden.

Vorher sollen dem Client allerdings zusätzlich die Rechte zum Validieren von Blöcken und Tätigen von Transaktionen gegeben werden. Dazu wird folgendes Kommando benötigt:

```
1 multichain-cli db grant 18tJkQWoVaoMNWsH4CN5ruazXnLjnGPUnyAdGA mine,send,receive
```

Bevor der Knoten nun gestartet wird, soll allerdings noch ein Blick in das Verzeichnis `/.multichain` auf dem Client geworfen werden. In diesem befindet sich bereits der Ordner der installierten Blockchain db. Wechselt man in den Ordner der Blockchain und betrachtet den Inhalt, beinhaltet dieser bereits einige generelle Dateien die Blockchain. Vergleicht man den Ordner allerdings mit demselben Ordner auf dem Server, sieht man, dass der Server sowohl mehr Dateien als auch mehr Ordner beinhaltet. Teilweise sind dies solche, die lediglich der Server als Ersteller der Blockchain besitzt, teilweise sind dies wie z. B. der Ordner `wallet` allerdings auch die Transaktionen bzw. Daten der Blockchain.

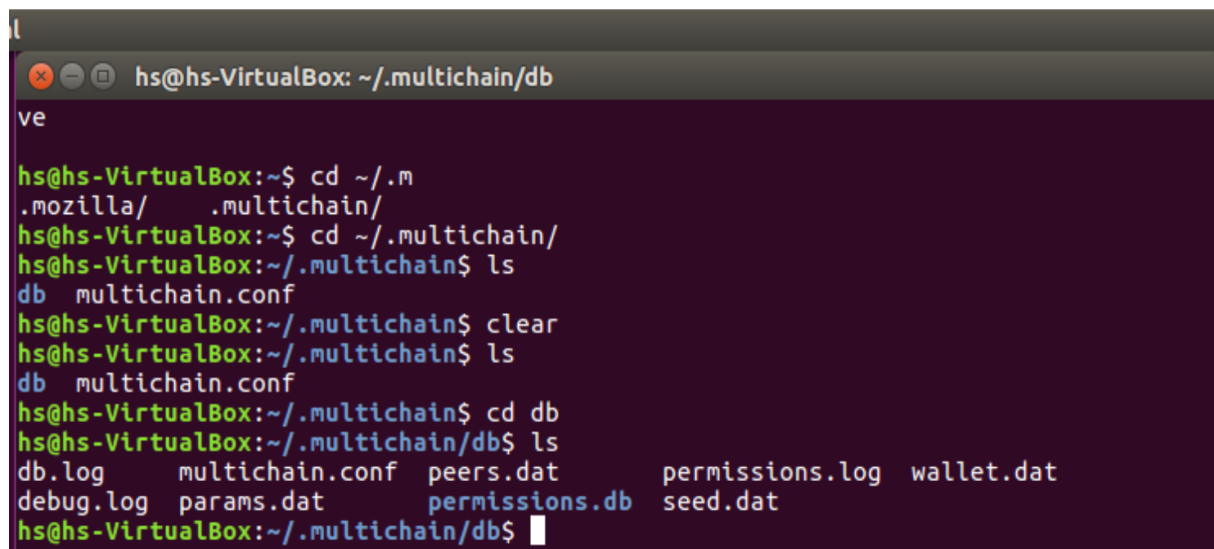


```

hs@hs-VirtualBox: ~/.multichain/db
hs@hs-VirtualBox:~$ cd ~/.multichain/
hs@hs-VirtualBox:~/.multichain$ ls
db  multichain.conf
hs@hs-VirtualBox:~/.multichain$ cd db
hs@hs-VirtualBox:~/.multichain/db$ ls
blocks      debug.log      multichain.conf  peers.dat      wallet
chainstate  entities.dat    multichain.pid   permissions.dat wallet.dat
database    entities.db     params.dat       permissions.db
db.log      fee_estimates.dat params.dat.bak   permissions.log
hs@hs-VirtualBox:~/.multichain/db$

```

Bank-Engineering 2 (DB-Präsi) [wird ausgeführt] - Oracle VM VirtualBox



```

hs@hs-VirtualBox: ~/.multichain/db
ve
hs@hs-VirtualBox:~$ cd ~/.m
.mozilla/  .multichain/
hs@hs-VirtualBox:~$ cd ~/.multichain/
hs@hs-VirtualBox:~/.multichain$ ls
db  multichain.conf
hs@hs-VirtualBox:~/.multichain$ clear
hs@hs-VirtualBox:~/.multichain$ ls
db  multichain.conf
hs@hs-VirtualBox:~/.multichain$ cd db
hs@hs-VirtualBox:~/.multichain/db$ ls
db.log      multichain.conf  peers.dat      permissions.log  wallet.dat
debug.log   params.dat       permissions.db  seed.dat
hs@hs-VirtualBox:~/.multichain/db$

```

Abbildung 8: Vergleich des `/.multichain/db`-Ordners auf dem Server und dem Client



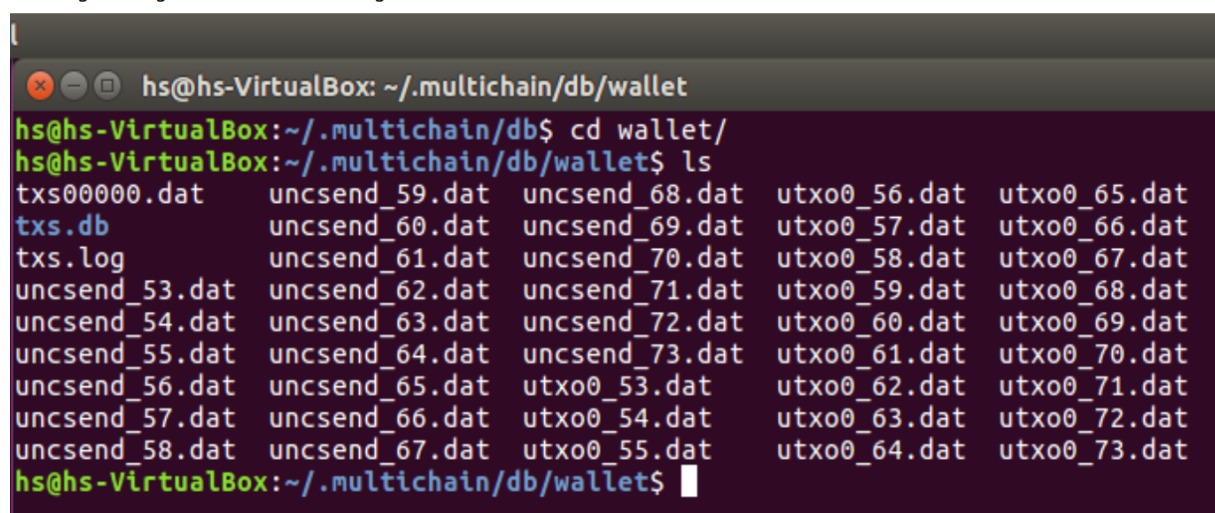
Um den Client nun als Knoten der Blockchain zu starten, wird dasselbe Kommando wie auf dem Server verwendet (`multichaind db -daemon`). An dieser Stelle wird nun die Dezentralität der Blockchain hervorgehoben: der Client empfängt nun alle Daten der Blockchain (Blöcke, Transaktionen, ...), sodass alle Daten der Blockchain auf beiden Knoten liegen. Betrachtet man bspw. den Ordner `wallet` wird deutlich, dass beide Knoten alle Transaktionen beinhalten. Sobald neue Transaktionen hinzukommen, erhalten beide Knoten diese.

```

hs@hs-VirtualBox:~/.multichain/db$ cd wallet/
hs@hs-VirtualBox:~/.multichain/db/wallet$ ls
txs00000.dat    uncsend_59.dat  uncsend_68.dat  utxo0_56.dat   utxo0_65.dat
txs.db          uncsend_60.dat  uncsend_69.dat  utxo0_57.dat   utxo0_66.dat
txs.log         uncsend_61.dat  uncsend_70.dat  utxo0_58.dat   utxo0_67.dat
uncsend_53.dat  uncsend_62.dat  uncsend_71.dat  utxo0_59.dat   utxo0_68.dat
uncsend_54.dat  uncsend_63.dat  uncsend_72.dat  utxo0_60.dat   utxo0_69.dat
uncsend_55.dat  uncsend_64.dat  uncsend_73.dat  utxo0_61.dat   utxo0_70.dat
uncsend_56.dat  uncsend_65.dat  utxo0_53.dat    utxo0_62.dat   utxo0_71.dat
uncsend_57.dat  uncsend_66.dat  utxo0_54.dat    utxo0_63.dat   utxo0_72.dat
uncsend_58.dat  uncsend_67.dat  utxo0_55.dat    utxo0_64.dat   utxo0_73.dat
hs@hs-VirtualBox:~/.multichain/db/wallet$

```

ank-Engineering 2 (DB-Präsi) [wird ausgeführt] - Oracle VM VirtualBox



```

hs@hs-VirtualBox:~/.multichain/db/wallet
hs@hs-VirtualBox:~/.multichain/db$ cd wallet/
hs@hs-VirtualBox:~/.multichain/db/wallet$ ls
txs00000.dat    uncsend_59.dat  uncsend_68.dat  utxo0_56.dat   utxo0_65.dat
txs.db          uncsend_60.dat  uncsend_69.dat  utxo0_57.dat   utxo0_66.dat
txs.log         uncsend_61.dat  uncsend_70.dat  utxo0_58.dat   utxo0_67.dat
uncsend_53.dat  uncsend_62.dat  uncsend_71.dat  utxo0_59.dat   utxo0_68.dat
uncsend_54.dat  uncsend_63.dat  uncsend_72.dat  utxo0_60.dat   utxo0_69.dat
uncsend_55.dat  uncsend_64.dat  uncsend_73.dat  utxo0_61.dat   utxo0_70.dat
uncsend_56.dat  uncsend_65.dat  utxo0_53.dat    utxo0_62.dat   utxo0_71.dat
uncsend_57.dat  uncsend_66.dat  utxo0_54.dat    utxo0_63.dat   utxo0_72.dat
uncsend_58.dat  uncsend_67.dat  utxo0_55.dat    utxo0_64.dat   utxo0_73.dat
hs@hs-VirtualBox:~/.multichain/db/wallet$

```

Abbildung 9: Vergleich des wallet-Ordners auf dem Server und dem Client

### 3.4 Betrachtung der Blockchain im Explorer

Beide Knoten sind nun am validieren der Blöcke. Dies lässt sich über den Explorer verfolgen. Wird die Startseite (s. Abbildung 5) betrachtet, wird die Blockchain mit dem in der Datei `db.conf` angegebenen Namen angezeigt. Darunter werden zudem die letzten getätigten Transaktionen angezeigt. Wird nun neben dem Namen der Blockchain auf die Anzahl der Blöcke geklickt, gelangt man zur Übersicht der Blöcke der Blockchain (s. Abbildung 10).

In der Liste wird für jeden Block folgendes angezeigt: die Nummer des Blocks in der Blockchain, die ID des Knotens, der diesen Block validiert hat, der Validierungszeitpunkt, die Anzahl der



### 3 Blockchain in der Praxis am Beispiel Multichain

Block	Miner	Approx. Time	Transactions	Chain Age
73	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:58	1	0.0442708
72	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:52	1	0.0442014
71	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:40	1	0.0440625
70	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:25	1	0.0438889
69	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:22	1	0.0438542
68	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:27:09	1	0.0437037
67	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:26:59	1	0.043588
66	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:26:50	1	0.0434838
65	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:26:40	1	0.0433681
64	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>	2018-03-29 19:26:31	1	0.0432639

Abbildung 10: Übersicht der Blöcke der Blockchain

Transaktionen, die der Block beinhaltet bzw. validiert hat und das Alter der Blockchain zum Zeitpunkt der Validierung des Blocks in Tagen. Auffällig ist hierbei, dass die Blöcke bislang alle lediglich eine Transaktion beinhalten. Dies ist darauf zurückzuführen, dass die einzigen Transaktionen, die bislang ausgeführt wurden, die Gewinnausschüttung der Belohnung für das Validieren eines Blockes ist. Wird nun auf die Zahl eines Blocks geklickt, gelangt man zur Übersicht des Blocks (s. Abbildung 11).

#### Block Summary

Hash	006c4fe42751471bd221617da2e70d011618013b1e5afa889ac6d20e26ee3d9b
Previous Block	<a href="#">0090fa50c44ab4c1283104d239e8d3dd1299d8f7d0db4f33a2777950a0bb9254</a>
Height	73
Miner	<a href="#">1R1TdYqH4ZXmRPphLjRZyyuEL9mGFSz1dH9DNU</a>
Version	3
Transaction Merkle Root	14b1148c7afe4f0e77e192919812a8a7070d0b9209ae6d460d67c9b2c633776f
Time	1522344478 (2018-03-29 19:27:58)
Nonce	50
Transactions	1

#### Transactions

Transaction	Size (kB)
<a href="#">14b1148c7afe4f0e77e192919812a8a7070d0b9209ae6d460d67c9b2c633776f</a> Miner Signature	0.219

Abbildung 11: Ansicht eines Blocks im Explorer

In der Übersicht sieht man eine Liste mit Eigenschaften des Blocks und darunter die beinhalteten Transaktionen. In der Liste befindet sich zunächst der Hash des Blocks, der als gültiger Hash zum Validieren des Blocks errechnet worden ist. Bei MultiChain-Blockchains ist standardmäßig die Regelung eingestellt, dass ein Hash lediglich gültig ist, wenn er mindestens zwei Nullen voransteht. Des Weiteren beinhaltet die Liste den Hash des vorherigen Blocks. Sobald ein nächster Block validiert wurde, wird zudem der Hash des nächsten Blocks eingetragen. Hier wird deutlich, wie die Blöcke verkettet sind und, dass eine entsprechend durch eine Änderung eines Blockes die Hash-Werte aller folgenden Blöcke neu berechnet werden müssten. Der Wert

### 3 Blockchain in der Praxis am Beispiel Multichain

„Height“ zeigt die Nummer des Blocks in der Blockchain an. Darunter befindet sich die ID des Knotens, der den Block validiert hat. Des Weiteren wird die Transaction Merkle Root angegeben, die zur Errechnung des Hash-Wertes verwendet wurde. Auch der Validierungszeitpunkt und die Anzahl der enthaltenen Transaktionen sind in der Liste aufgeführt. Zuletzt wird auch die Nonce angegeben, die zum Berechnen des Hash-Wertes verwendet wurde. Wird nun auf die ID der enthaltenen Transaktion geklickt, öffnet sich eine entsprechende Detail-Ansicht, die die Informationen der Transaktion darstellt.

Hash	14b1148c7afe4f0e77e192919812a8a7070d0b9209ae6d460d67c9b2c633776f			
Appeared in	Datenbank-Engineering Blockchain, Block 73 (2018-03-29 19:27:58)			
Number of inputs	1 – <a href="#">jump to inputs</a>			
Number of outputs	2 – <a href="#">jump to outputs</a>			
Size	219 bytes			

[Bitcoin JSON](#)
[MultiChain JSON](#)
[MultiChain Hex](#)

#### Inputs

Index	Previous output	Native	From address	ScriptSig
0	Generation		1R1TdYqH4ZXmRPphLJrZyyuEL9mGFSz1dH9DNU	1:49 1:01 6:2f50...482f

#### Outputs

Index	Redeemed at input	Native	To address	ScriptPubKey
0	Not yet redeemed	0.5	1R1TdYqH4ZXmRPphLJrZyyuEL9mGFSz1dH9DNU	DUP HASH160 20:b1a1...40da EQUALVERIFY CHECKSIG

Abbildung 12: Detail-Ansicht einer Transaktion

Diese Ansicht stellt unter anderem den Hash-Wert/die ID der Transaktion, die Nummer des Blocks, in dem sie enthalten ist, und die Größe der Transaktion in Bytes dar. Außerdem sind die Inputs und Outputs der Transaktion dargestellt. Die Inputs stellen dar, wer etwas geschickt hat (From Address), wie sich die geschickte Anzahl an Währung vom Sender zusammenstellt (Native) und woher diese kommt (Previous Input). Im Output wird dargestellt, wer das Ergebnis der Transaktion erhält (To address) und wie viel Währung er erhält (Native). In Falle der abgebildeten Transaktion schickt der Server an sich selbst 0.5 Währung, die neu generiert werden (Previous Output = Generation). Dies ist die Belohnung für das erfolgreiche Validieren eines Blockes. Wird nun auf die ID eines Knotens geklickt, gelangt man zur Ansicht dessen (s. Abbildung 13):

Hier werden lediglich die Berechtigungen des Knotens (Permissions) und die Anzahl der Währung, die dieser momentan besitzt (Native Balance) aufgeführt.

### 3.5 Manuelle Durchführung einer Transaktion

Um nun eine manuelle Transaktion durchzuführen, kann man zunächst von einem Knoten zum anderen Währung überweisen. Diese Transaktion wird dann zunächst an alle Knoten der Blockchain übermittelt und dann in den nächsten Block mit aufgenommen und durch Validierung dieses Blockes gültig und durchgeführt. Um eine Transaktion von Währung durchzuführen, muss

 Address 1R1TdYqH4ZXmRPphLJrZyyuEL9mGFSz1dH9DNU

#### Permissions

- Connect
- Send
- Receive
- Issue
- Create
- Mine
- Admin
- Activate

#### Native Balance

- 39.5

Abbildung 13: Explorer-Ansicht eines Knotens

das Kommando `multichain-cli` gefolgt von dem Namen der Blockchain, dem Befehl `send`, der ID des Empfänger-Knotens und die zu überweisende Anzahl der Währung eingegeben werden. So könnte eine Überweisung des Servers an den Client wie folgt lauten:

```
1 multichain-cli db send 18tJkQWoVaoMnW5H4CN5ruazXnLjnGPUnyAdGA 2.0
```

Wird diese Transaktion nun durchgeführt, wird sie auf der Startseite des Explorers aufgeführt ist zunächst allerdings noch nicht bestätigt, da diese noch in keinen validierten Block mit aufgenommen wurde (s. Abbildung 14).

#### Latest Transactions

Txid	Type	Confirmation	Time
<a href="#">f9e89cd1bca6f9829ad091cdf53a0a3f404bfd159dabb2bb5b5bbeaf8df02ea</a>		<b>Mempool</b>	< 1 minute

Abbildung 14: Unbestätigte Transaktion des Servers an den Client

Sobald diese Transaktion in einen validierten Block mit aufgenommen wurde, ist sie bestätigt. Wird diese Transaktion nun im Explorer betrachtet (s. nächste Abbildung), sieht man, dass der Server (From Address) viermal 0.5 Währung (Native) verschickt hat und auch aus welchen Transaktionen diese stammen (Previous Output). Dies ist dazu zurückzuführen, dass der Server bislang lediglich mehrmals 0.5 Währung durch das Validieren von Blöcken erhalten hat. Auf der Empfängerseite der Transaktion sieht man allerdings, dass der Client (To address) 2 Währung (Native) erhalten hat.

## 3.6 Speichern von Daten

Neben dem Verschicken von Währung können in einer Blockchain allerdings auch allgemeine Daten gespeichert werden. Hierzu wird erneut auf das Kommando `multichain-cli` zurückgegriffen. Dabei muss erneut die Blockchain angegeben werden, gefolgt von dem Befehl `publish`, einem sog. Stream (root ist der Standard-Stream der Blockchain), einem Key dem die Daten zugeordnet

### 3 Blockchain in der Praxis am Beispiel Multichain

#### Inputs

Index	Previous output	Native	From address	ScriptSig
0	14b1148c7a...:0	0.5	1R1TdYqH4ZXmRPpHLJrZyyuEL9mGFSz1dH9DNU	71:3044...7701 33:0220...60fb
1	2c5726f96e...:0	0.5	1R1TdYqH4ZXmRPpHLJrZyyuEL9mGFSz1dH9DNU	71:3044...da01 33:0220...60fb
2	35a844eb0f...:0	0.5	1R1TdYqH4ZXmRPpHLJrZyyuEL9mGFSz1dH9DNU	72:3045...b401 33:0220...60fb
3	a4b12d40dd...:0	0.5	1R1TdYqH4ZXmRPpHLJrZyyuEL9mGFSz1dH9DNU	72:3045...2c01 33:0220...60fb

#### Outputs

Index	Redeemed at input	Native	To address	ScriptPubKey
0	Not yet redeemed	2	18tJkQWoVaoMnWsh4CN5ruazXnLjnGPUnyAdGA	DUP HASH160 20:3a56...c19a EQUALVERIFY CHECKSIG

Abbildung 15: Betrachtung der Transaktion des Servers an den Client

werden sollen und mittels dem später alle Daten abgefragt werden können, die diesem Key jemals in dieser Blockchain zugeordnet wurden und den Daten selbst. Die Daten müssen allerdings im Hexadezimalformat angehängt werden. Dies hat den Vorteil, dass ganze Bilder oder Videos im hexadezimal kodierte Format an die Blockchain angehängt werden können. Ein entsprechendes Kommando könnte für diese Blockchain wie folgt aussehen:

```
1 multichain-cli db publish key1 12345d
```

Sobald man das Kommando bestätigt, wird eine neue Transaktion angelegt, die die Daten enthält. Die folgende Abbildung stellt diese Transaktion im Explorer dar:

#### Inputs

Index	Previous output	Native	From address	ScriptSig
0	f9e89cd1bc...:1	0.0	1R1TdYqH4ZXmRPpHLJrZyyuEL9mGFSz1dH9DNU	71:3044...bc01 33:0220...60fb

#### Outputs

Index	Redeemed at input	Native	To address	ScriptPubKey
0	Not yet redeemed	0	None	20:7370...a433 DROP 8:7370...7931 DROP RETURN 3:12345d

Stream	root
Key	key1
Data	12345d

Abbildung 16: Transaktion im Explorer, die Daten enthält

Transaktionen die Daten enthalten, werden von einem Knoten (From address) in einen Stream geschrieben und an niemanden (To address) gesendet. Die Währung der Transaktion ist entsprechend 0. Beim Output der Transaktion wird allerdings eine Tabelle dargestellt, die die versendeten Daten im Feld „Data“, den Key, dem die Daten zugeordnet wurden, und den Stream, in den die Daten geschrieben wurden, enthält. Durch diese Art der Speicherung der Daten durch eine Transaktion und somit den Eingang dieser in die Merkle Root und entsprechend den Hash des validierten Blockes sind auch diese Daten kaum manipulierbar, sofern nicht alle folgenden Blöcke neu berechnet werden.

## 4 Kritische Reflexion

Es muss beachtet werden, dass das Thema Blockchain erst 2009 aufgekommen ist und die Benutzung der Blockchain, außerhalb der Kryptowährungen, nicht weit verbreitet ist. Demnach wurden noch nicht viele Projekte beziehungsweise Studien zu diesem Thema veröffentlicht. Außerdem ist auch zu erwähnen, dass die meiste Literatur sich mit dem Handeln von Kryptowährungen befasst und nicht auf die technische Theorie des Themas eingeht. Die Hauptquelle dieser Ausarbeitung ist über ein Synonym veröffentlicht worden und keiner kann die Authentizität der Ausarbeitung validieren. Außerdem ist zu erwähnen dass diese Ausarbeitung auf einer Literaturrecherche basiert und somit nur ein Abbild der aktuellen Literatur darstellt.

Für die praktische Anwendung wurde ausschließlich eine Blockchainsoftware verwendet. Es wurde nicht in Betracht gezogen, dass andere Blockchains ganz anders funktionieren könnten. Es sollten in Zukunft weitere Anwendung betrachtet werden.

Außerdem stellte sich das ausgewählte Thema als fehlerhaft definiert dar. Der Titel „Blockchain als verteiltes Datenbankmanagementsystem“ ist nicht richtig gewählt, da eine Blockchain eher als einzelne Tabelle zu definieren ist. Ein Datenbankmanagementsystem besitzt jedoch viel mehr Attribute, die in einer Blockchain nicht enthalten sind. Beispielsweise kann mit einem Datenbankmanagementsystem komplexe Manipulationsbefehle ausgeübt werden, jedoch ist die Blockchain nur in der Lage Datensätze an die Kette anzuhängen. Demnach wurde der Titel der Ausarbeitung in „Blockchain als verteilte Datenbank“ umbenannt.

## 5 Fazit

Zusammenfassend lässt sich sagen, dass Blockchain als Datenbank für eine persistente, manipulationsfreie Datenspeicherung sinnvoll ist. Mit der Software Multichain besteht die Möglichkeit mit wenigen Schritten eine funktionierende Blockchain aufzusetzen.

Das Interesse an einer produktiven Nutzung im unternehmerischen Umfeld wird weiter ansteigen. Der Aspekt der Integrität ist hierfür ausschlaggebend. Im Bereich der Blockchain als Datenbank stehen aber noch viele Fragen aus. Beispielsweise besteht theoretisch die Möglichkeit bei einer Kontrolle von mehr als 50 Prozent der Nodes, die Blockchain zu manipulieren. Des Weiteren spielen auch datenschutzrechtliche Fragen eine Rolle. Beispielsweise lassen sich Blockchains nicht rückwirkend verändern, demnach können keine sensiblen oder illegalen Daten gelöscht werden. In Zukunft müssen weitere Studien über die produktiven Ansätze der Blockchain-Technologie durchgeführt werden.

## Literaturverzeichnis

Alfons Kemper and André Eickler. *Datenbanksysteme: Eine Einführung*. Oldenbourg, München, 6., aktualisierte und erw. aufl. edition, 2006. ISBN 3486576909. URL [http://deposit.ddb.de/cgi-bin/dokserv?id=2785967&prov=M&dok\\_var=1&dok\\_ext=htm](http://deposit.ddb.de/cgi-bin/dokserv?id=2785967&prov=M&dok_var=1&dok_ext=htm).

Dmitry Korzun and Andrei Gurtov. *Structured peer-to-peer systems: Fundamentals of hierarchical organization, routing, scaling, and security*. Springer, New York, NY, 2013. ISBN 978-1-4614-5482-3.

Multichain. Multichain explorer, a. URL <https://github.com/MultiChain/multichain-explorer>.

Multichain. Multichain, b. URL <https://www.multichain.com/>.

Reimund Neugebauer, editor. *Digitalisierung: Schlüsseltechnologien für Wirtschaft und Gesellschaft*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1. auflage edition, 2018. ISBN 978-3-662-55889-8.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL <https://bitcoin.org/bitcoin.pdf>.

## Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir unsere Hausarbeit

„Blockchain als verteilte Datenbank“

selbstständig und ohne fremde Hilfe angefertigt haben und dass wir alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen unserer Arbeit besonders gekennzeichnet und die Quellen zitiert haben.

---

Ort, Datum

---

Unterschrift

---

Ort, Datum

---

Unterschrift