A=SIT Technology
E-Government Innovationszentrum

25. Juli 2017 Studie · Blockchain · Web & Cloud · 22 Min. Lesezeit

Blockchain & Smart Contracts

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

Quelle https://technology2.a-sit.at/veroeffentlichungen/blockchain-smart-contracts/

Die Studie beschreibt die grundlegenden Blockchain-Typen sowie Konsensus-Protokolle mit Fokus auf die Ethereum Plattform.

Die Blockchain stellt eine verteilte Datenbank dar, die aus verketteten Blöcken besteht. Die Ethereum Plattform wurde ausgewählt, da es sich hierbei um eine erfolgsversprechende Plattform zur Ausführung von Smart Contracts handelt. Bei Smart Contracts handelt es sich um autonome Agenten, welche innerhalb des Blockchainnetzwerkes ausgeführt werden. Dadurch lassen sich ausfallsichere, zensurresistente, verteilte Applikationen entwickeln. Aufbauend auf dem erworbenen Wissen wurde ein blockchain-basierter Messenger umgesetzt.

1. Einleitung

Die Blockchain wurde von einer Person oder einer Gruppe von Personen unter dem Pseudonym Satoshi Nakamoto als Teil der Kryptowährung Bitcoin entwickelt. Im Allgemeinen stellt die Blockchain eine verteilte Datenbank dar, die aus verketteten Blöcken besteht. Dieses Dokument beschreibt die grundlegenden Blockchain-Typen sowie Konsensus-Protokolle mit Fokus auf die Ethereum Plattform. Die Ethereum Plattform wurde ausgewählt, da es sich hierbei um eine Erfolg versprechende Plattform zur Ausführung von Smart Contracts handelt. Bei Smart Contracts handelt es sich um autonome Agenten, welche innerhalb des Blockchainnetzwerkes ausgeführt werden. Dadurch lassen sich ausfallsichere, zensurresistente, verteilte Applikationen entwickeln. Im nächsten Kapitel werden die Grundlagen zur Blockchain-Technologie vorgestellt. Kapitel drei beschäftigt sich mit Smart Contracts. Im vierten Kapitel wird ein auf der Blockchain-Technologie aufbauender Messenger vorgestellt. Abschließend wird im letzten Kapitel ein Fazit gezogen.

2. Grundlagen

Die Blockchain kann als dezentrale sequentielle Transaktionsdatenbank benutzt werden, deren Einträge im Nachhinein weder gelöscht, noch verändert werden können. Daher wird die Blockchain-Technologie bei den meisten modernen Kryptowährungen eingesetzt. Dies hat den Vorteil für den Verkäufer, dass getätigte Transaktionen im Nachhinein nicht widerrufen oder verändert, und dass Währungseinheiten nicht doppelt ausgegeben werden können. Bitcoin verfolgt das Ziel, eine neue dezentral kontrollierte Kryptowährung zu

erschaffen. Ethereum hingegen generalisiert das Blockchain-Konzept. Neben einer verteilten Datenbank in der Konten manipulationsgeschützt gespeichert werden, bietet Ethereum auch eine Ausführungsumgebung, in der für Ethereum entwickelte Applikationen, auch Smart Contracts genannt, ausgeführt werden können. Die von Ethereum unterstützte Sprache erlaubt die Erstellung und Ausführung beliebiger Anwendungen und beseitigt somit die Limitierungen von Bitcoin, mit dessen Architektur sich nur sehr simple Anwendungen umsetzen lassen. Smart Contracts werden in Ethereum in der sogenannten Ethereum Virtual Machine, kurz EVM, ausgeführt. Details dazu werden in Abschnitt 3 diskutiert. Abbildung 1 zeigt einen beispielhaften Transaktionsablauf, bei dem ein Nutzer bzw. eine Nutzerin eine Transaktion erstellt, signiert, und an das Netzwerk übermittelt. Die Transaktion wird von Teilnehmern im Netzwerk, die neue Blöcke erstellen können (Miner), kontrolliert und falls gültig in einen neuen Block aufgenommen. Die folgenden Abschnitte beschreiben die wichtigsten Begriffe rund um die Ethereum Plattform.

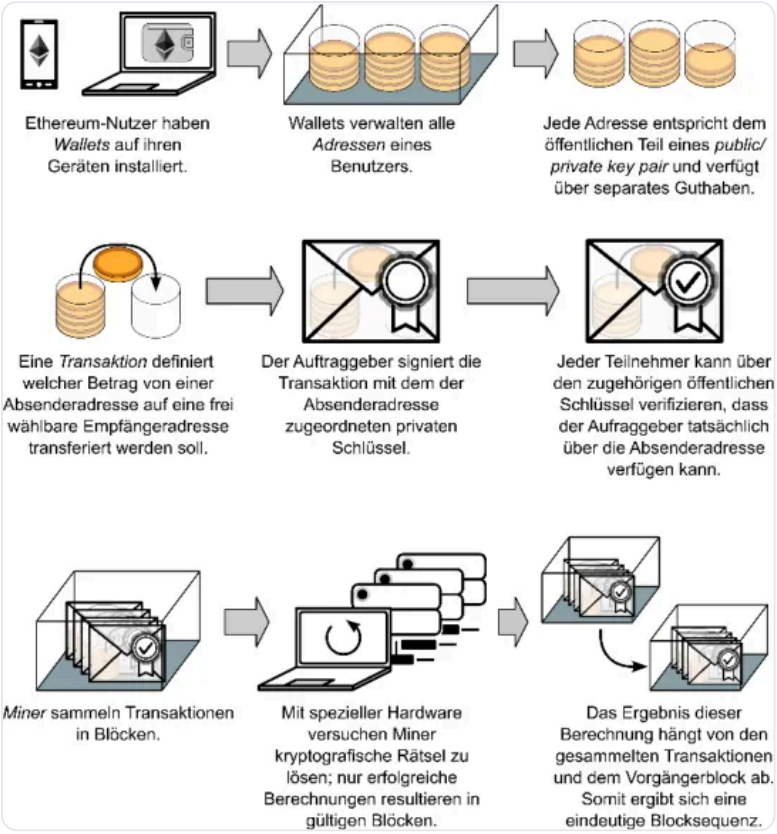


Abbildung 1: Ablauf einer Transaktion in Ethereum

2.1. Adresse

Eine Adresse kann im Zusammenhang mit Kryptowährungen als Konto betrachtet werden, mit dem Unterschied, dass jeder Teilnehmer selbstständig beliebig viele Adressen generieren kann. Guthaben kann von Konto zu Konto transferiert werden. Erst wenn eine neu erstellte Adresse als Empfänger in einer Transaktion genannt wird, erfährt das restliche Netzwerk von ihrer Existenz. Aus technischer Sicht handelt es sich bei der Adresse um eine Repräsentation des öffentlichen Schlüssel eines Public-Key-Kryptosystems.

2.2. Konten

In Ethereum hat jedes Konto eine 20-Byte lange Adresse, ein Guthaben in sogenanntem „Ether“, einen Speicherplatz und eventuell einen Vertragscode. Transaktionen finden immer zwischen Konten statt. Es gibt zwei verschiedene Kontotypen, extern kontrollierte Konten und Vertragskonten. Extern kontrollierte Konten werden mit dem zugehörigen privaten Schlüssel kontrolliert. Vertragskonten werden von einem das Konto definierenden Code, dem Smart Contract Code, kontrolliert. Der Vertragscode wird jedes Mal ausgeführt, wenn eine Nachricht an das dazugehörende Konto geschickt wird. Ohne externe Aktivierung per Nachricht kann der Vertragscode nicht gestartet werden. Nachrichten können beispielsweise von extern kontrollierten Konten verschickt werden, indem eine Transaktion erstellt, signiert und ans Netzwerk übermittelt wird.

2.3. Nachrichten und Transaktionen

Bei einer Transaktion handelt es sich um eine signierte Nachricht die von einer Adresse an eine andere Adresse geschickt wird. Transaktionen können zum Transfer von Guthaben oder zum Transfer von Daten verwendet werden. Transaktionen werden, sofern sie gültig sind, dezentral im Netzwerk verbreitet. Empfängt ein Miner, also ein Teilnehmer im Netzwerk, der neue Blocke erstellen kann, eine Transaktion, wird diese überprüft und falls diese gültig ist, wird sie in einen Block aufgenommen. Es werden keine vertrauenswürdigen Drittinstanzen zur Abwicklung der Transaktion benötigt. Eine Transaktion besteht aus folgenden Elementen:

- Empfänger
- Signatur des Senders
- Der Menge der zu transferierenden Währungseinheiten
- Optionales Datenfeld
- Dem GASLIMIT, welches angibt, wie viele Berechnungsschritte die Ausführung der Transaktion maximal benötigen darf
- Dem GASPRICE, welcher angibt, wieviel GAS der Sender bereit ist, pro Rechnungsschritt zu zahlen. Die Felder GASLIMIT und GASPRICE dienen als Schutz für das Ethereum-Netzwerk. Ethereums Anti-Denial-Of-Service-Modell beruht auf der Idee, dass jeder für verwendete Ressourcen zahlen muss. Mit GASLIMIT und GASPRICE können unbeabsichtigte aber auch bösartige Endlosschleifen verhindert werden. Alle Operationen werden in der Ethereum Virtual Machine ausgeführt. Dabei handelt es sich um eine spezielle Laufzeitumgebung zur sicheren Ausführung von nicht vertrauenswürdigem Code. Jede Operation in der Ethereum Virtual Machine kostet einen bestimmten Betrag in GAS. Die meisten Operationen kosten 1 GAS, aufwendige Operationen können aber auch mehrere GAS-Einheiten kosten. Neben Rechenleistung kostet auch der benötigte Speicherplatz bzw. jedes Byte in den Transaktionsdaten. Da ein Angreifer für alle aufgewendeten Ressourcen zahlen muss, ist ein Angriff unwirtschaftlich.

2.4. Nachrichten

Bei Nachrichten handelt es sich um virtuelle Objekte, die von Verträgen verschickt werden können. Nachrichten werden nie serialisiert und existieren nur innerhalb der Ausführungsumgebung von Ethereum. Nachrichten bestehen aus den folgenden Elementen:

- Implizit dem Sender
- Dem Empfänger
- Der Menge der zu transferierenden Währungseinheiten
- Einem optionalen Datenfeld
- Einem GASLIMIT, welches angibt, wie viele Berechnungsschritte die Ausführung der Transaktion maximal benötigen darf Im Wesentlichen entspricht eine Nachricht einer Transaktion, mit dem Unterschied, dass eine Nachricht von einem Vertragskonto und nicht von einem extern kontrollierten Konto erstellt wird. Nachrichten werden erstellt und verschickt wenn der Vertrag den „CALL" Opcode ausführt. Durch diesen Opcode kann ein Vertrag eine Nachricht an einen anderen Vertrag schicken und dadurch dessen Ausführung starten. Das GASLIMIT in einer Nachricht oder Transaktion gilt für alle dadurch ausgelösten Ausführungen. Wenn beispielsweise ein extern kontrolliertes Konto A ein Limit von 10000 GAS für die Ausführung von B definiert und B 6000 GAS verbraucht bevor B eine Nachricht an C schickt, und C 3000 GAS verbraucht, kann B danach noch maximal 1000 GAS verbrauchen bevor die Ausführung gestoppt wird.

2.5. Wallet

Bei einem Wallet handelt es sich um eine Software für den Endbenutzer bzw. die Endbenutzerin, welche die Erstellung von Transaktionen ermöglicht. Dazu verwaltet und/oder generiert die Software die benötigten Schlüsselpaare. Zusätzlich verbindet sich die Software mit dem Netzwerk, um immer über aktuelle Transaktionen und Blöcke informiert zu werden. Aus den empfangenen Blöcken kann die Software den jeweiligen „Kontostand" aller verwalteten Konten/Adressen berechnen. Alle Transaktionen müssen mit dem zur Absenderadresse gehörenden privaten Schlüssel signiert werden, um sicherzustellen, dass die Transaktion vom jeweiligen Eigentümer bzw. der jeweiligen Eigentümerin ausgelöst wurde.

2.6. Blockchain und Mining

Die Ethereum Blockchain besteht, wie der Name andeutet, aus mehreren verketteten Blöcken. Der erste Block wird Genesis Block genannt. Jeder Block außer dem Genesis Block enthält eine Referenz auf den vorherigen Block, wodurch sich eine verkette Liste ergibt. Im Gegensatz zur Blockchain von Bitcoin enthalten die Blöcke der Ethereum Blockchain nicht nur die Transaktionsliste sondern auch einen State. Daher wird im Gegensatz zu Bitcoin nicht die gesamte Blockchain benötigt, um Transaktionen zu prüfen. Neben der Transaktionsliste und dem State enthält jeder Block eine Blocknummer und den aktuellen Schwierigkeitsgrad. Der Schwierigkeitsgrad wird für den Proof-of-Work Algorithmus benötigt. Durch diesen Algorithmus wird sichergestellt, das Blöcke im Nachhinein nicht verändert oder gelöscht werden können, solange mehr als die Hälfte der Rechenleistung im Netzwerk von ehrlichen Teilnehmern zur Verfügung gestellt wird. Bei Proof-of-Work handelt es sich um den derzeit verwendeten Konsensus-Algorithmus (siehe Abschnitt 2.7). Dieser wird mit dem Update auf *Serenity,* der vierten großen Release Version von Ethereum,

durch *Casper* (siehe Abschnitt *2.7.2),* einen Proof-of-Stake Konsensus-Algorithmus ausgetauscht. Dieses Update wird voraussichtlich als Hard Fork (siehe Abschnitt 2.8) ausgeführt. Generell kann man je nachdem, wer was kann und darf, zwischen drei Blockchain-Typen unterscheiden. Diese Typen werden in den nächsten Abschnitten behandelt.

2.6.1. Permissionless / Public Blockchain

Bei einer öffentlichen Blockchain kann jeder teilnehmen, einen Knoten betreiben und alle Transaktionen und Blöcke abrufen und überprüfen [1]. Bei öffentlichen Blockchains steht meist Anonymität, Unveränderbarkeit und Transparenz im Vordergrund, dafür werden Einbußen in der Effizienz hingenommen. Öffentliche Blockchains werden auch als Permissionless Blockchains bezeichnet. Der große Vorteil von öffentlichen Blockchains ist, dass keine besonderen Schutzmaßnahmen gegen unehrliche Teilnehmer benötigt werden, da das System sich selbst überprüft und wie erwartet funktioniert, solange sich die Mehrheit der Teilnehmer ehrlich verhält. Jeder volle Knoten überprüft alle Blöcke und Transaktionen auf Gültigkeit. Es gibt zumindest zwei verschiedene Motivationen einen vollen Knoten zu betreiben. Einerseits um selbst neue Blöcke erstellen zu können („minen“) und hierfür eine Belohnung, in Form von Transaktionsgebühren der aufgenommenen Transaktionen, oder der Ausschüttung von neuen Währungseinheiten, zu erhalten und andererseits um sicher zu sein, dass eine Transaktion wirklich aufgenommen wurde und man somit den Betrag erhält. Der zweite Grund ist speziell für Händler interessant, da sie sich sonst auf externe Dienste verlassen müssten.

Da Jeder teilnehmen darf, benötigen öffentlichen Blockchains auch keine Zugangsregeln bzw. Beschränkungen. Dadurch kann jeder Nutzer bzw. jede Nutzerin Anwendungen hinzufügen, ohne dass diese von Dritten genehmigt werden müssen. Ein weiterer Vorteil ist, dass die Daten typischerweise weltweit verteilt gespeichert werden, wodurch sich eine hohe Ausfallsicherheit ergibt.

2.6.2. Permissioned / Private Blockchain

Private Blockchains werden vom Eigentümer betrieben. Der Eigentümer kann entscheiden, wer Schreibrechte hat, also Transaktionen durchführen darf und Blöcke erstellen kann. Auch der Lesezugriff kann beschränkt werden. Bei privaten Blockchains steht meistens die Effizienz im Vordergrund, dafür werden Abstriche bei Anonymität, Unveränderbarkeit und Transparenz in Kauf genommen [2]. Der Betreiber einer privaten Blockchain kann nach Belieben die Regeln verändern, Transaktionen rückgängig machen oder das Guthaben verändern. Zu den Vorteilen von privaten Blockchains gehören die Tatsache, dass Angriffe einer Mehrheit an Rechenleistung der Teilnehmer („51% Angriffe“) keine Rolle spielen, da nur vertrauenswürdige Miner teilnehmen dürfen. Außerdem können Transaktionen generell deutlich günstiger durchgeführt werden [1]. Bei privaten Blockchains bringt das Betreiben eines vollen Knoten hauptsächlich den Vorteil eines dezentralen Backups und der Möglichkeit, Regelverstöße zu bemerken bzw. aufzuzeichnen, gegen die je nach abgeschlossenem Vertrag möglicherweise zivilrechtlich vorgegangen werden kann.

2.6.3. Consortium Blockchain

Consortium Blockchains stellen eine Mischung aus öffentlichen und privaten Blockchains dar und sollen die Vorteile beider Varianten vereinen. Bei öffentlichen Blockchains besteht kein oder ein geringes gegenseitiges Vertrauen, bei privaten Blockchains hingegen gibt es eine zentrale Instanz der gänzlich

vertraut werden muss. Consortium Blockchains streben eine Mischung aus den beiden Ansätzen an, indem sie die zentrale vertrauenswürdige Instanz durch eine Gruppe von Instanzen ersetzen. Diese Art von Blockchain dürfte sich gut für die Zusammenarbeit von unterschiedlichen Organisationen eignen [3].

2.7. Konsensus-Protokoll

Der Konsensus-Algorithmus oder das Konsensus-Protokoll definiert, wer einen Block erstellen darf und ob dieser gültig ist. Als Belohnung bekommt der Miner bei öffentlichen Blockchains meist ein paar Währungseinheiten ausgeschüttet oder darf die Transaktionsgebühren behalten. Der Konsensus-Algorithmus hat das Ziel, dass sich die Teilnehmer des Netzwerkes auf eine gültige Blockchain einigen können, ohne sich untereinander vertrauen zu müssen und ohne auf eine zentrale Instanz vertrauen zu müssen. Die folgenden Abschnitte beschreiben unterschiedliche Verfahren für die Umsetzung eines Konsensus-Protokolls. Neben diesen gibt es noch eine Vielzahl von anderen Verfahren, von denen sich derzeit noch keines durchsetzen konnte.

2.7.1. Proof-of-Work

Bei Proof-of-Work Algorithmen geht es darum, eine schwierige Aufgabe zu lösen, deren Lösung einfach validiert werden kann. Ethereum verwendet derzeit *Ethash* [4] als Proof-of-Work Algorithmus. Der Mining-Prozess besteht aus den folgenden vier Schritten:

1. Zuerst wird ein *Seed* aus den Blockköpfen berechnet.
2. Aus diesem *Seed* wird ein 16MB großer, pseudozufälliger Cache errechnet.
3. Dieser Cache dient zur Berechnung einer 1GB großen Datenmenge. Jeder Eintrag in dieser Menge hängt nur von einem kleinen Teil der Daten im Cache ab.
4. Der Miner nimmt zufällige Einträge aus dieser Datenmenge und berechnet deren Hashwert. Dieser Vorgang wird wiederholt, bis ein Hash gefunden wird, dessen Wert als Zahl interpretiert kleiner als der Zielwert ist. Der Zielwert wird aus dem zu diesem Zeitpunkt festgelegtem Schwierigkeitsgrad errechnet.

Ethereum verwendet Keccak-256 und Keccak-512 als Hash-Algorithmen. Die beiden Hash-Algorithmen unterscheiden sich nur in Details von SHA3-256 bzw. SHA3-512. Der Proof-of-Work Algorithmus ermöglicht die Einigung auf eine gültige Blockchain, da im Falle von mehreren Pfaden jener mit der größten Gesamtschwierigkeit als gültig betrachtet wird. Dadurch müsste ein Angreifer, um einen Block auszutauschen, alle nachfolgenden Blöcke ebenfalls neu berechnen und dies schneller als das restliche Netzwerk gemeinsam. Für einen erfolgreichen Angriff werden in etwa 51% der Rechenleistung des Netzwerkes benötigt. Es gibt jedoch die Vermutung, dass bereits Angreifer mit geringerer Rechenleistung das Netzwerk stark stören können. Um den Energieverbrauch und die Ressourcennutzung zu reduzieren, plant Ethereum, auf einen Proof-of-Stake Algorithmus umzusteigen.

2.7.2. Proof-of-Stake

Im Gegensatz zu Proof-of-Work Algorithmen, bei denen alle Miner um die Wette rechnen und nur der Erste gewinnt, wird bei Proof-of-Stake Algorithmen der nächste Miner durch einen deterministischen Prozess ausgewählt. Dieser Auswahlprozess bezieht den Einsatz der einzelnen Kandidaten in die Berechnung ein. Ethereum plant, den Proof-of-Stake Algorithmus *Casper* zu verwenden [5]. Bei *Casper* werden Miner,

welche sich nicht an die Regeln halten, bestraft und können den zuvor deponierten Einsatz verlieren. Dies soll zu einem stabilen Netzwerk führen. Durch den notwendigen Einsatz von Guthaben bei Proof-of-Stake Algorithmen sind beispielsweise 51% Angriffe deutlich riskanter, als bei Proof-of-Work Algorithmen [5]. Ein weiterer Vorteil von Proof-of-Stake Algorithmen ist der deutlich geringere Energieaufwand im Vergleich zu Proof-of-Work Algorithmen. Dies führt dazu, dass weniger Währungseinheiten ausgeschüttet werden können und trotzdem ein Anreiz zum Minen vorhanden ist.

2.7.3. Proof-of-Authority

Bei Proof-of-Authority Algorithmen werden Miner an sich nicht mehr benötigt, stattdessen können autorisierte Signatoren jederzeit nach eigenem Ermessen neue Blöcke erstellen [6]. Dadurch ergeben sich neue Herausforderungen, wie die Kontrolle der Blockerstellungsfrequenz, welche Signatoren wann autorisiert sind und wie die Liste der in Frage kommenden Signatoren dynamisch angepasst werden kann. Ethereum verwendet derzeit das *Clique* -Protokoll im Testnetz. Beim Testnetzwerk handelt es sich um eine Alternative zum Hauptnetzwerk, in dem ohne den Einsatz von echten Ethern experimentiert werden kann. Die im Testnetzwerk generierten Ether sind de facto wertlos. Das Clique-Protokoll sieht vor, dass ein Signator maximal einen von x Blöcken signieren darf, dadurch kann ein bösartiger Signator keinen verheerenden Schaden im Netzwerk anrichten. Des Weiteren sieht das Protokoll vor, dass der Miner, der an der Reihe ist, exakt zum optimalen Zeitpunkt, entsprechend der gewünschten Blockfrequenz, einen Block erstellt und alle anderen Signatoren, die berechtigt, aber nicht an der Reihe sind, eine gewisse Zeit warten. Dadurch hat der Signator, der an der Reihe ist, einen kleinen Vorteil gegenüber anderen Signatoren und es wird zusätzlich die Netzwerklast reduziert. Um neue Signatoren aufzunehmen bzw. bestehende aus der Liste zu entfernen, ist ein Voting-Prozess vorgesehen. Jeder Signator auf der Liste hat pro Block ein Stimmrecht, um neue Signatoren aufzunehmen oder bestehende auszuschließen.

2.7.4. Proof-of-Burn

Bei Proof-of-Burn Algorithmen wird im Gegensatz zur Proof-of-Work Algorithmen nicht Elektrizität „verbrannt" sondern digitale Währungseinheiten [7]. Die Währungseinheiten werden „verbrannt" indem sie auf zufällige Adressen überwiesen werden. Die Wahrscheinlichkeit, dass jemand den dazugehörigen privaten Schlüssel hat oder findet, ist extrem gering. Durch das Vernichten der Währungseinheiten erhält man das Recht, an der Mining-Lotterie teilzunehmen. Je mehr Währungseinheiten man vernichtet, desto höher ist die Wahrscheinlichkeit, dass man zum Minen ausgewählt wird. Dies kann mit einem Proof-of-Stake Einsatz verglichen werden, den man niemals zurückbekommt.

2.7.5. Proof-of-Capacity

Proof-of-Capacity Algorithmen sind eine Implementierung der Idee „Megabytes als Ressource" [7]. Bei diesem Ansatz werden große Mengen Speicherplatz benötigt, wodurch dieser Ansatz als relativ sicher gegen Botnetz-Angriffe zu sehen ist, da ein riesiger Speicherverbrauch eher auffällt als beispielsweise eine erhöhte Netzwerklast. Des Weiteren ist der Energieverbrauch deutlich geringer, als bei Proof-of-Work Ansätzen. Bei diesem Ansatz werden Daten nach einem bestimmten Algorithmus erstellt, beispielsweise dem wiederholten anwenden einer Hashfunktion auf den öffentlichen Schlüssels bzw. dessen Hashwert.

Die Wahrscheinlichkeit, einen Block erstellen zu dürfen, steigt mit der Größe des verwendeten Speicherplatzes. Kritiker sehen Probleme bei diesem Ansatz, da ein Miner nichts zu verlieren hat, wenn er oder sie sich unehrlich verhält.

2.8. Soft Fork und Hard Fork

Bei einem Hard Fork gibt es Regel- oder Protokolländerungen, die nicht rückwärtskompatibel sind. Dies hat zur Folge, dass ältere Knoten beispielsweise Transaktionen erstellen könnten, die nach den neuen Regeln ungültig sind. Ein Hard Fork benötigt Anpassungen in der Software der Nutzer und der Miner [8]. Im Gegensatz dazu, können ältere Anwendungen nach einem Soft Fork weiter verwendet werden. Sie verfügen möglicherweise aber nicht über den gesamten Funktionsumfang. Ethereum hat beispielsweise beim Übergang auf Homestead, die zweite große Release Version, einen Hard Fork durchgeführt. Ein weiterer Hard Fork wurde nach dem Angriff auf die Decentralized Autonomous Organization (kurz DAO) durchgeführt. Bei DAO handelt es sich um einen komplexen Smart Contract, bei dem Angreifer einen Fehler ausnutzen um Guthaben abzuziehen. Mittels Hard Fork wurde das bei diesem Angriff verlorengegangene Guthaben wiederhergestellt [9]. Aus diesem Hard Fork ist Ethereum Classic entstanden, da nicht alle in der Ethereum Gemeinschaft den Hard Fork akzeptiert haben [10].

3. Smart Contracts

Smart Contracts bestehen in Ethereum aus einer low-level, Stack basierten Bytecodesprache, die als Ethereum Virtual Machine Code oder EVM Code bezeichnet wird. Die Sprache ist Turingvollständig, dies bedeutet, dass jeder beliebige Algorithmus abgebildet werden kann. Der Code wird in der Ethereum Virtual Machine, einer speziellen Ausführungsumgebung, exekutiert. Smart Contracts verfügen über drei verschiedene Speicher: einem Stack, einem temporären Speicher und einem Langzeitspeicher. Neben diesen Speichern kann der Code noch auf den Sender, die in der Nachricht enthaltenen Daten sowie auf die Metadaten des Blockes zugreifen. Als Ergebnis kann ein Smart Contract ein Bytearray zurückgeben. Der große Vorteil von Smart Contracts gegenüber herkömmlichen Anwendungen ist, dass ihr Programmcode für jeden einsehbar ist und dass die Anwendungen ausfallsicher sind, solange das Ethereum-Netzwerk funktioniert. Nachfolgend werden ein paar Smart Contracts, die auch als Distributed Apps bezeichnet werden, kurz vorgestellt.

ENS: ENS steht für Ethereum Name Service [11]. ENS gibt eine sichere, dezentralisierte Möglichkeit, Ressourcen über menschenlesbare Namen zu adressieren. Dadurch können beispielsweise hexadezimale Adressen, wie sie zum Transfer von Guthaben benötigt werden, auf menschenlesbare Formen abgebildet werden. Mittels ENS kann auch ein alternatives Domain Name System betrieben werden. **Alice.Si:** Bei Alice.Si handelt es sich um einen Smart Contract, welcher das Vertrauen in Hilfsorganisationen weiter erhöhen soll [12]. Bei Alice.Si zahlen Spender bzw. Spenderinnen in den Smart Contract ein. Dieser zahlt die Spenden an die Hilfsorganisation nur aus, wenn zuvor definierte Ziele erreicht wurden, ansonsten bekommen die Spender bzw. Spenderinnen ihr Geld zurück. Da alle Transaktionen sichtbar sind, kann transparent nachvollzogen werden, wofür das Geld ausgegeben wird bzw. wurde. **WeiFund:** Bei WeiFund handelt es sich um einen Smart Contract für Schwarmfinanzierungen [13]. Im Gegensatz zu anderen Schwarmfinanzierungsplattformen bietet WeiFund mehr Sicherheit und geringere Kosten durch den Einsatz eines öffentlich einsehbaren Smart Contracts. **vDice.io:** Bei vDice handelt es sich um einen Smart Contract, der Glückspiel ermöglicht [14]. Die Blockchain-basierte Anwendung verarbeitet Wetten unter zu Hilfenahme

eines Orakels. **EtherOpt:** EtherOpt ermöglicht das Handeln mit Optionen, ähnlich wie es derzeit bei Börsen üblich ist [15]. Bei den Optionen handelt es sich um Verträge, die dem Käufer das Recht geben, etwas zu einem bestimmten Preis, an oder bis zu einem bestimmten Zeitpunkt, zu kaufen oder zu verkaufen.

Um die Möglichkeiten der Ethereum Blockchain zu demonstrieren und die Einsatzgebiete noch weiter auszudehnen, wurde das Ziel gesetzt, einen sicheren Messenger zu entwickeln. Der im nächsten Abschnitt vorgestellte Messenger verwendet ebenso wie die zuvor genannten Beispiele die Blockchain sowie einen Smart Contract.

4. Messenger

Ziel war es, eine sichere Messaging-Anwendung auf Basis von Ethereum zu erstellen die den sicheren Austausch von Nachrichten ermöglicht. Des Weiteren soll die Messaging Anwendung folgende Eigenschaften aufweisen:

- Verschlüsselte Kommunikation, damit nur berechtigte Personen Nachrichten lesen können.
- Sender und Empfänger dürfen für Außenstehende nicht nachvollziehbar sein.
- Der Absender darf nicht wie etwa bei E-Mail Adressen fälschbar sein.
- Sende- und Lesebestätigung.
- Es soll nicht nachvollziehbar sein, ob oder wann oder wie oft zwei Teilnehmer miteinander kommunizieren, außer einer der beiden Teilnehmer will die Kommunikation offenlegen.
- Wenn einer der beiden Teilnehmer beweisen will, dass er eine Nachricht bekommen oder versandt hat, soll der andere Teilnehmer dies nicht abstreiten können.
- Nachrichten sollen im Nachhinein nicht gelöscht, verändert oder verloren gehen können.
- Nachrichten sollen über einen vertrauenswürdigen Zeitstempel verfügen.
- Die Anwendung soll gegenüber Zensur und Denial of Service resistent sein.
- Die Nachrichten sollen dezentral gesichert werden.
- Das System soll skalieren.
- Teilnehmer sollen nicht die gesamte Blockchain laden müssen, wenn sie dies nicht wollen.

Um diese Ziele zu erreichen, wurde eine private Blockchain erstellt, auf die jeder lesend zugreifen kann, aber nur bestimmte Adressen minen dürfen. Um keine unnötigen Ressourcen für den Konsensus-Prozess zu verschwenden, wurde das Clique Protokoll gewählt. Des Weiteren wurde die Blockfrequenz erhöht, um eine schnelle Aufnahme von Transaktionen in der Blockchain zu erreichen und der Gaspreis auf Null gesetzt, um eine kostenlose Kommunikation zu ermöglichen. Die folgenden Abschnitte beschreiben die verschiedenen Phasen der Kommunikation.

4.1. Erster Start

Beim ersten Start der Messaging-Anwendung wird ein zufälliges Schlüsselpaar erstellt, welches als Hauptidentität benutzt wird. Alternativ kann auch ein existierendes Schlüsselpaar importiert werden. Die Hauptidentität wird nie zum Senden von Transaktionen verwendet, da sonst nachvollziehbar wäre, wer mit wem kommuniziert. Stattdessen werden zum Senden und zum Empfangen von Nachrichten immer Einwegadressen verwendet. Durch die Verwendung von Einwegadressen kann nicht verfolgt werden, wer mit wem kommuniziert.

4.2. Austausch von Kontaktdaten

Vor der ersten Kontaktaufnahme müssen die Kontaktinformationen, bestehend aus dem öffentlichen Teil der Hauptidentität und dem öffentlichen Teil der ersten Einwegadresse, ausgetauscht werden. Es wird für jeden Kontakt eine eigene Einwegadresse von der Hauptidentität und dem Kontakt abgeleitet. Da die eigene Einwegadresse jederzeit wieder abgeleitet werden kann, muss sie nicht gespeichert werden. Die Kontaktinformationen des Gegenübers hingegen müssen gespeichert werden. Nach dem Austausch dieser Daten können sich beide Parteien gegenseitig Nachrichten schicken bzw. verifizieren, ob erhaltene Nachrichten von der anderen Partei stammen.

4.3. Senden von Nachrichten

Nachrichten werden immer an die zuletzt empfangene Einwegadresse des zugehörigen Kontaktes gesendet. Damit der Empfänger bzw. die Empfängerin weiß, von wem die Nachricht kommt und an welche Adresse er bzw. sie antworten kann, wird jede Nachricht zuerst um eine neu generierte Empfangsadresse erweitert und anschließend mittels der Hauptidentität signiert. Die signierte Nachricht wird anschließend symmetrisch mittels eines zufälligen Schlüssels verschlüsselt. Der Schlüssel wiederrum wird jeweils mit dem öffentlichen Schlüssel der Absender und der Empfangsadresse verschlüsselt. Dadurch ist für Außenstehende nicht sichtbar, wer diese Nachricht signiert hat, bzw. wie die Nachricht lautet. Dennoch kann sowohl der Sender bzw. die Senderin als auch der Empfänger bzw. die Empfängerin die Nachricht jederzeit entschlüsseln. Zuletzt wird die verschlüsselte Nachricht noch mit der Einwegadresse signiert, um eine gültige Transaktion zu erhalten. Diese Transaktion wird anschließend an das Netzwerk übermittelt und von einem Miner in einen neuen Block aufgenommen.

4.4. Empfangen von Nachrichten

Nachdem die Empfängerin bzw. der Empfänger die Nachricht entschlüsselt hat prüft sie bzw. er zuerst die Signatur. Falls diese gültig ist, werden die Nachricht und die neue Empfangsadresse extrahiert. Damit weiß der Empfänger bzw. die Empfängerin, wohin er bzw. sie die Antwort schicken muss. Das Senden der Antwort entspricht gleichzeitig einer Lesebestätigung für die alte Nachricht, da sonst die neue Empfangsadresse nicht bekannt wäre.

4.5. Messenger Eigenschaften

Will eine der beiden Parteien beweisen, dass sie eine Nachricht geschickt hat, muss nur die dazugehörige Einwegadresse offengelegt werden plus die Einwegadresse auf der die Empfangsadresse empfangen wurde. Die alte Empfangsadresse wurde von der anderen Partei mit deren Hauptidentität signiert, daher ist nicht abstreitbar, dass diese von Ihr kommt. Um zu beweisen, dass man eine Nachricht bekommen hat, reicht es, die dazugehörige Einwegadresse offenzulegen, da die enthaltene Nachricht direkt von der

anderen Partei signiert wurde. In beiden Fällen kann die Blockerstellungszeit als Zeitstempel herangezogen werden, zudem die Nachricht spätestens existiert haben muss. Durch die Integration der Nachrichten in die Blockchain ergibt sich automatisch ein dezentrales, ausfallsicheres Backup. Zudem können bereits versendete Nachrichten nicht nachträglich verändert oder gelöscht werden. Da jede Nachricht verschlüsselt ist, hat dies keine negativen Auswirkungen auf die Privatsphäre, nur berechtige Parteien können die Nachrichten lesen. Die Absenderadresse einer Nachricht lässt sich nicht fälschen, da jede Transaktion mit dem dazugehörigen privaten Schlüssel signiert werden muss. Da für jede Nachricht eine neue Sende- und Empfangsadresse verwendet wird, lässt sich auch nicht zuordnen, wer mit wem wie oft kommuniziert. Für Geräte bei denen der Ressourcenverbrauch zu hoch wäre, oder die Privatsphäre nicht das primäre Ziel ist, gibt es die Möglichkeit, einzelne Blöcke zu laden. Mittels eines Smart Contracts kann dabei überprüft werden, ob der empfangene Block Teil der Blockchain ist. Der Smart Contract speichert die Fingerabdrücke aller Blöcke. Daher reicht es, einen bekannten vertrauenswürdigen Block zu kennen, um die Validität einzelner Blöcke davor zu überprüfen. Für volle Anonymität wird empfohlen, sich nur über Tor zur Blockchain zu verbinden, da sonst über die IP-Adresse Rückschlüsse gezogen werden können. Zudem empfiehlt es sich, immer alle Blöcke herunterzuladen, da sonst Rückschlüsse gezogen werden könnten, in welchem Block sich eine Nachricht befindet bzw. dass kommuniziert wurde.

5. Fazit

Je nach gewähltem Blockchain-Typ eignet sich die Blockchain-Technologie für andere Zwecke. Während private Blockchains hauptsächlich innerhalb einer Organisation ihre Stärken ausspielen können, bieten öffentliche Blockchains Vorteile für alle Anwendungen, bei denen eine große Gruppe von Personen die sich untereinander nicht kennen oder vertrauen, ein gemeinsames Ziel verfolgen. Durch Smart Contracts kann die Funktionalität einer Blockchain durch autonome Programme erweitert werden, wodurch sich viele neue Möglichkeiten ergeben. Ethereum scheint derzeit die erfolgversprechendste Plattform zur Ausführung von Smart Contracts zu sein. Deshalb wurde im Rahmen dieses Projektes die Ethereum Plattform zur Umsetzung einer sicheren Messaging-Anwendung gewählt. Durch die Verwendung der Blockchain und von Smart Contracts ergeben sich Eigenschaften wie Transparenz, Nachvollziehbarkeit, Existenzbeweis und Ausfallsicherheit.

6. Literaturverzeichnis

[1] V. Buterin, „On Public and Private Blockchains," 07 08 2015. [Online]. Available:

https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/. [Zugriff am 05 07

[2] B. Noyes, „Public, Permissioned, and Private Blockchains," 29 11 2016. [Online]. Available:

https://medium.com/@BrettNoyes/public-permissioned-and-private-blockchains-3c32965e33c9. [Zugriff am 05 07 2017].

[3] C. Thompson, „The difference between a Private, Public & Consortium Blockchain.," 26 10 2016. [Online]. Available: http://www.blockchaindailynews.com/The-difference-between-a-Private-Public-Consortium-Blockchain_a24681.html. [Zugriff am 05 07 2017].

[4] Ethereum, „Ethash," [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethash. [Zugriff am 04 07 2017].

[5] Ethereum, „Proof of Stake FAQ,“ [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ. [Zugriff am 04 07 2017].

[6] Ethereum, „Clique PoA protocol & Rinkeby PoA testnet,“ [Online]. Available: https://github.com/ethereum/EIPs/issues/225. [Zugriff am 05 07 2017].

[7] R. Patterson, „Alternatives for Proof of Work, Part 2: Proof of Activity, Proof of Burn, Proof of Capacity, and Byzantine Generals,“ 26 08 2015. [Online]. Available: https://bytecoin.org/blog/proof-of-activity-proof-of-burn-proof-of-capacity/. [Zugriff am 05 07

[8] jcrain, „What is a hardfork?,“ [Online]. Available: https://ethereum.stackexchange.com/questions/1952/what-is-a-hardfork. [Zugriff am 06 07

[9] CoinDesk, „Ethereum Executes Blockchain Hard Fork to Return DAO Funds,“ 20 06 2016.

[Online]. Available: http://www.coindesk.com/ethereum-executes-blockchain-hard-fork-returndao-investor-funds/. [Zugriff am 06 07 2017].

[10] Ethereum Classic, „Ethereum Classic,“ [Online]. Available: https://ethereumclassic.github.io/.

[Zugriff am 06 07 2017].

[11] ENS, „Ethereum Name Service,“ [Online]. Available: https://ens.domains/#. [Zugriff am 06 07

[12] ALICE SI Ltd., „Alice,“ [Online]. Available: https://alice.si/. [Zugriff am 06 07 2017].

[13] WeiFund, „WeiFund - Decentralized Fundraising,“ [Online]. Available: http://weifund.io/. [Zugriff am 06 07 2017].

[14] vDice.io, „vDice Game,“ [Online]. Available: https://www.vdice.io/. [Zugriff am 06 07 2017].

[15] EtherOpt, „EtherOpt,“ [Online]. Available: https://etheropt.github.io/. [Zugriff am 06 07 2017].

[16] Ethereum, „A Next-Generation Smart Contract and Decentralized Application Platform,“

[Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Zugriff am 12 05 2016].

[17] Ethereum, „White Paper,“ [Online]. Available: https://github.com/ethereum/wiki/wiki/White Paper. [Zugriff am 03 07 2017].

[18] K. Torpey, „You Really Should Run a Bitcoin Full Node: Here’s Why,“ 09 05 2017. [Online].

Available: https://bitcoinmagazine.com/articles/you-really-should-run-full-bitcoin-node-hereswhy/. [Zugriff am 07 07 2017].