Udocoin

PROJEKTARBEIT

des Studienganges Angewandte Informatik
an der Dualen Hochschule Baden-Württemberg Mannheim

von

Leo Stadtler, Lorenz Müller, Matthias Heilmann & Sean Schwarz

21.12.2023

Bearbeitungszeitraum	04.10.2023 - 21.12.2023
Kurs	TINF21AI2
Vorlesung	Verteilte Systeme
Dozent	Jochen Kluger

Inhaltsverzeichnis

1. Einleitung

Obwohl die Digitalisierung des Geldes viele Vorteile gebracht hat, wie das vereinfachte und verschnellerte Verschieben von Geld, bringt diese auch einige Herausforderungen mit sich.[1, S. 10] So kann mit nur wenigen Operationen an einem PC innerhalb von Sekunden Geld aus dem Nichts geschaffen werden.[1, S. 10] Daten wie Kontostände sind lediglich durch das Vier-Augen-Prinzip und Vertragsstrafen vor Fälschung geschützt.[2, S. 101] Vertrauen, Zentralisierung und veraltete Strukturen sind die wichtigsten Probleme des aktuellen Geldsystems, an denen Kryptowährungen ansetzen.[1, S. 10]

In dieser Arbeit wird die Entwicklung einer Kryptowährung mit dem Namen “Udocoin” im Rahmen der Vorlesung “Verteilte Systeme” an der DHBW Mannheim vorgestellt. Es werden zuerst grundlegende theoretische Ansätze zu Blockchains und Kryptowährungen erläutert und anschließend wird auf die Implementierung des Projekts eingegangen.

2. Theoretische Grundlagen

Die Theorie hinter Kryptowährungen gibt es bereits seit den 1980er Jahren, in denen Verschlüsselungsexperten versuchten, eine von Banken unabhängige elektronische Währung zu entwickeln.[2, S. 105] Bitcoin wurde jedoch erst später als erste funktionierende Kryptowährung im Januar 2009 von dem Pseudonym Satoshi Nakamoto veröffentlicht.[2, S. 101] Kryptowährungen basieren auf der Blockchain-Technologie und sind damit dezentral aufgebaut.[1, S. 10] Somit werden Daten nicht auf einem Server gespeichert, sondern in einem aus tausenden Rechnern bestehenden verteilten Netzwerk.[1, S. 10] Kryptowährungs-Netzwerke wie Bitcoin zeichnet aus, dass Teilnehmer des Netzwerks für die Validierung von Transaktionen im Netz entlohnt werden. [2, S. 105] Diese Transaktionen werden durch Verschlüsselungsalgorithmen verschlüsselt, durchgeführt und validiert. [1, S. 11]

2.1. Kryptographische Grundlagen

“Kryptographie ist eine Wissenschaft, die sich mit der Verschlüsselung und Erschaffung von Kryptosystemen beschäftigt”[1, S. 11] und ist entscheidend für Kryptowährungen, da mit ihr Sicherheit und Vertraulichkeit gewährleistet wird.[1, S. 10f]

2.1.1. Hashfunktionen

Hashfunktionen sind deterministische Algorithmen, die aus einer Eingabe beliebiger Länge eine Ausgabe mit festgelegter Länge generieren.[3] Kleinste Veränderungen an den Eingabedaten führen zu völlig verschiedenen Ausgabedaten. Für moderne Hash Algorithmen wie SHA-256 ist es praktisch nicht möglich, Kollisionen, also verschiedene Eingaben, die zum selben Hashwert führen, zu finden.[3] Außerdem kann man aus dem Hashwert nicht die ursprünglichen Daten herleiten. Das Hashen ist irreversibel. Das macht den Hashwert zu einem sicheren digitalen Fingerabdruck der ursprünglichen Daten.[3]

```
>>> str1 = bytes("Ich liebe Udocoin!", "utf-8")
>>> str2 = bytes("ich liebe Udocoin!", "utf-8")
>>> hashlib.sha256(str1).hexdigest()
'51062807169609e1ccf0a708a59112005897b33aed111cf567f353ec8b426235'
>>> hashlib.sha256(str2).hexdigest()
'825148b1c75af0538dd1ff813b0065ff646115d43598872eae2d15b33f533e6d'
```

Abbildung 1: Geringe Veränderungen in Eingabedaten führen zu verschiedenen Hashwerten

2.1.2. Asymmetrische Verschlüsselung und Digitale Signaturen

Asymmetrische Verschlüsselung ist eine Form von Verschlüsselung, in der es zwei Arten von Schlüssel gibt, die paarweise erstellt werden können. Ein Schlüsselpaar besteht aus dem öffentlichen “Public-Key” und dem privaten “Private-Key”.[4] Eine Nachricht, die mit einem Public-Key verschlüsselt wurde, kann nur mit dem zugehörigen Private-Key entschlüsselt werden.[4] Einen Public-Key kann man aus einem Private-Key herleiten, das Gegenteil ist jedoch nicht der Fall. Dadurch kann ein Public-Key öffentlich verbreitet werden, ohne die Sicherheit des Private-Keys zu beeinträchtigen.[4]

Im Blockchainbereich wird asymmetrische Verschlüsselung hauptsächlich dazu verwendet, digitale Signaturen zu erstellen, um Transaktionen zu verifizieren. Dabei wird der Private-Key zum Signieren einer Nachricht verwendet. Diese Signatur kann vom Public-Key entschlüsselt werden und mit dem Hash der mitgesendeten Nachricht abgeglichen werden.[5] Das Signieren und Verifizieren einer Nachricht wird in folgender Abbildung dargestellt.

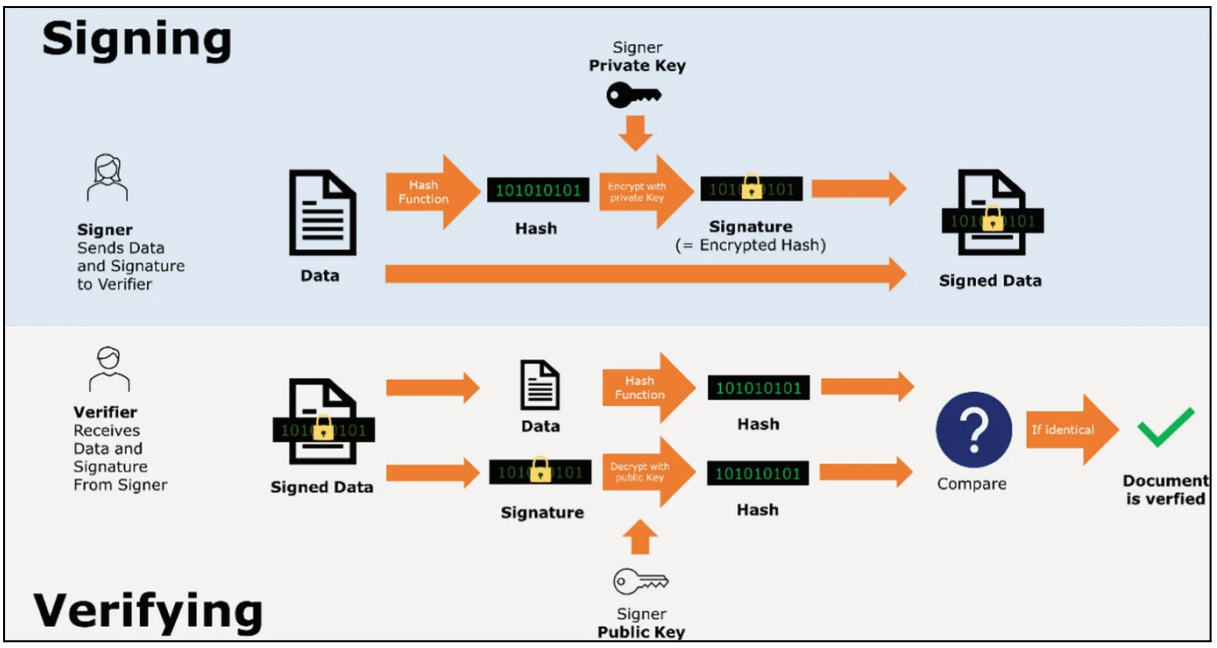

Abbildung 2: Signieren und Verifizieren einer digitalen Signatur[5]

Der Ersteller der Signatur wendet eine Hashfunktion auf die Daten an, die er signieren möchte. Diesen Hash signiert er mit seinem Private-Key und sendet die Signatur zusammen mit den ursprünglichen Daten an den Verifizierer. Der Verifizierer wendet dieselbe Hashfunktion auf die Daten an und entschlüsselt die Signatur mit dem Public-Key, welchen er auch mit den signierten Daten erhalten kann. Den daraus resultierenden Hash vergleicht der Verifizierer mit dem Hash, den er mit den Daten erstellt hat. Wenn die Hashes identisch sind, dann ist sichergestellt, dass die Signatur nur von der Entität stammen kann, die Zugriff auf den Private-Key hat.[5]

2.2. Blockchain-Technologie

Eine Blockchain ist eine linear verbundene Sammlung von “Blöcken”, die mit kryptographischen Hashes sicher verbunden werden.[6] Blockchains werden in Peer-to-Peer-Netzwerken, wie dem Bitcoin Netzwerk dezentral gepflegt, sodass Nutzer Währungen auch ohne Vertrauen in eine zentrale Bankinstanz austauschen können.[7, S. 1]

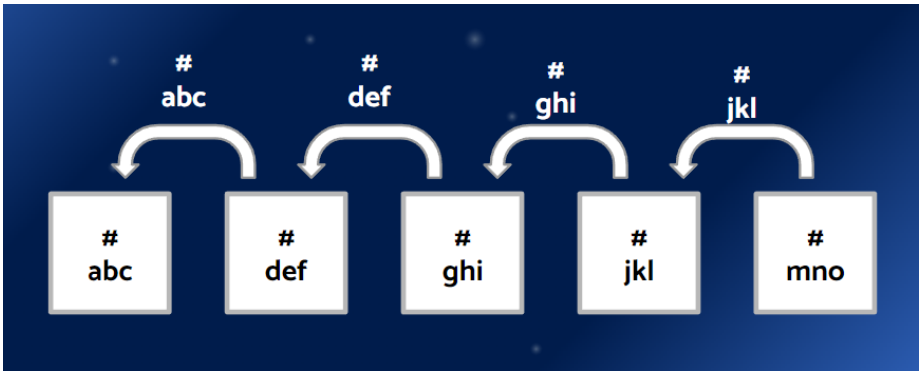

Abbildung 3: Referenzen auf Hashwerte

Jeder Block bis auf den ersten Block einer Blockchain enthält Daten und eine Referenz auf den Hashwert des gesamten vorherigen Blocks.[7, S. 2] Würde ein Angreifer versuchen, die Daten aus einem Block in der Kette zu verändern, würde sich auch der Hashwert dieses Blocks verändern. Da der Hashwert dann nicht mehr mit dem Hashwert, der im nächsten Block gespeichert ist, übereinstimmen würde, würde die Blockchain dadurch invalidiert werden. Ein Blockchain-Server, der einen solchen manipulierten Block erhalten würde, würde ihn dementsprechend ablehnen.

Im Kontext von Kryptowährungen enthalten Blöcke Transaktionsdaten, die angeben, von welcher Adresse eine Transaktion ausgeht und zu welcher Adresse die Währung übertragen wird. Diese Transaktionsdaten werden von der Entität, die die Kryptowährung versendet, digital signiert, um sicherzustellen, dass die Transaktionsanfrage tatsächlich von dieser Entität stammt.

Transaktionsanfragen werden an Netzwerkknoten des Blockchainnetzwerks gesendet und im sogenannten “Mempool” zwischengespeichert, bis sie in einem Block veröffentlicht werden.

Um die Blockchain rückwirkend verändern zu können, müsste ein Angreifer einen Block und alle darauffolgenden Blöcke neu veröffentlichen. Durch das Proof-of-Work Modell, das beispielsweise von Bitcoin implementiert wird, ist dies unpraktisch, da die dazu benötigte Rechenleistung mehr als 50% der Rechenleistung des Netzwerks einnehmen müsste.[8]

3. Implementierung

3.1. Aufbau der Blockchain

Da es sich bei Udocoin um eine möglichst einfache Kryptowährung handeln soll, wurde sich für eine Account-Struktur entschieden, in der jeder Benutzer einen einfachen Kontostand hat, den er ausgeben kann. Jede Transaktion wird in der implementierten Lösung im Klartext zusammen mit der Signatur in einem Block gespeichert. Der Kontostand eines Benutzers setzt sich dabei aus der Summe der zugehörigen Transaktionsbeträge zusammen. Bitcoin verwendet im Gegensatz dazu das sogenannte Unspent Transaction Outputs (UTXO) Modell, in dem sich die Kontostände der Benutzer aus nicht-ausgegebenen Transaktionen zusammensetzen.[9] Außerdem verwendet Bitcoin Merkle Bäume, um Speicherplatz zu sparen.[5, S. 4] Zum Vereinfachen der Implementierung wurden bei Udocoin keine solcher Komprimierungstechniken angewendet.

3.1.1. Schlüssel/Adressen

Um Udocoins empfangen und ausgeben zu können, ist ein Schlüsselpaar nötig, mit dem man Transaktionen freigeben und empfangen kann. Es wurde sich für die RSA-Verschlüsselung entschieden, da sie aus vorherigen Vorlesungen schon bekannt war. Ein Udocoin Nutzer muss sich ein RSA-Schlüsselpaar generieren, um sich somit ein Guthabenkonto zu erstellen.

Der Public-Key agiert dabei als öffentliche Adresse, die man bei einer Transaktionsanfrage adressieren kann und mit der man einen Kontostand auf der Blockchain auslesen kann. Der Private-Key wird dazu verwendet, um Transaktionsanfragen zu signieren, damit Angreifer keine Transaktionen mit fremdem Guthaben durchführen können. Dadurch wird garantiert, dass die Transaktion vom tatsächlichen Besitzer des Private-Keys freigegeben wurde.

Bitcoin verwendet ebenfalls eine asymmetrische Verschlüsselung, allerdings mit dem ECDS Algorithmus, der zu kürzeren Schlüsseln als bei der in diesem Projekt verwendeten RSA Implementation führt. Die Bitcoin Implementation von ECDSA hat

32 Byte lange Private-Keys und 33 Byte lange Public-Keys.[10] Die Udocoin RSA-Schlüssel sind jeweils 493 und 182 Bytes groß.

3.1.2. Proof-of-Work und Mining

Udocoin verwendet das sogenannte Proof-of-Work Modell, um Konsens zwischen verschiedenen Servern über den Status der Blockchain zu erreichen. Dabei enthält jeder Block zusätzlich zu den Transaktionsdaten eine Zahl, die, wenn sie mit den Blockdaten gehashed wird, dazu führt, dass der Hashwert mit einer festgelegten Anzahl von Nullbytes beginnt oder endet. Das Finden dieser Zahl bezeichnet man als “Mining”. Die Schwierigkeit des Minings steigt exponentiell mit der Anzahl der benötigten Nullbytes.[7, S. 3] Wenn ein Netzwerkknoten einen validen Proof-of-Work findet, erwirbt er somit das Recht, den nächsten Block zur Blockchain hinzuzufügen. Um einen Anreiz zum Betreiben von minenden Netzwerkknoten zu schaffen, werden in jedem neu veröffentlichten Block dem Autor des Blocks Udocoins ausgeschüttet. Dies bezeichnet man als Mining-Reward. Nur durch Mining werden neue Coins in Umlauf gebracht.

Die Schwierigkeit des Minings der Kryptowährung Bitcoin variiert je nach der aktuellen Hashrate, sodass neue Blöcke mit relativ konstanter Frequenz gemined werden.[7, S. 3] Der Einfachheit halber wurde bei Udocoin eine konstante Schwierigkeit implementiert.

Der Mining-Reward für Blöcke halbiert sich im Fall von Udocoin alle 1000 Blöcke, damit die Währung nicht durch Inflation an Wert verliert. Um trotz eines geringeren Mining-Rewards Anreiz zum Minen zu schaffen, könnte man in Zukunft Transaktionsgebühren in die Blockchain Server einbauen, wie auch im originalen Bitcoin Paper vorgeschlagen.[7, S. 4]

3.1.3. Forken der Blockchain

Sollte es durch Netzwerklatenz oder fehlende Verbindungen zwischen Teilen des Netzwerks dazu kommen, dass verschiedene Server verschiedene Ausprägungen der Blockchain als valide ansehen, bezeichnet man dies als Blockchain Fork (siehe Abb. 4)

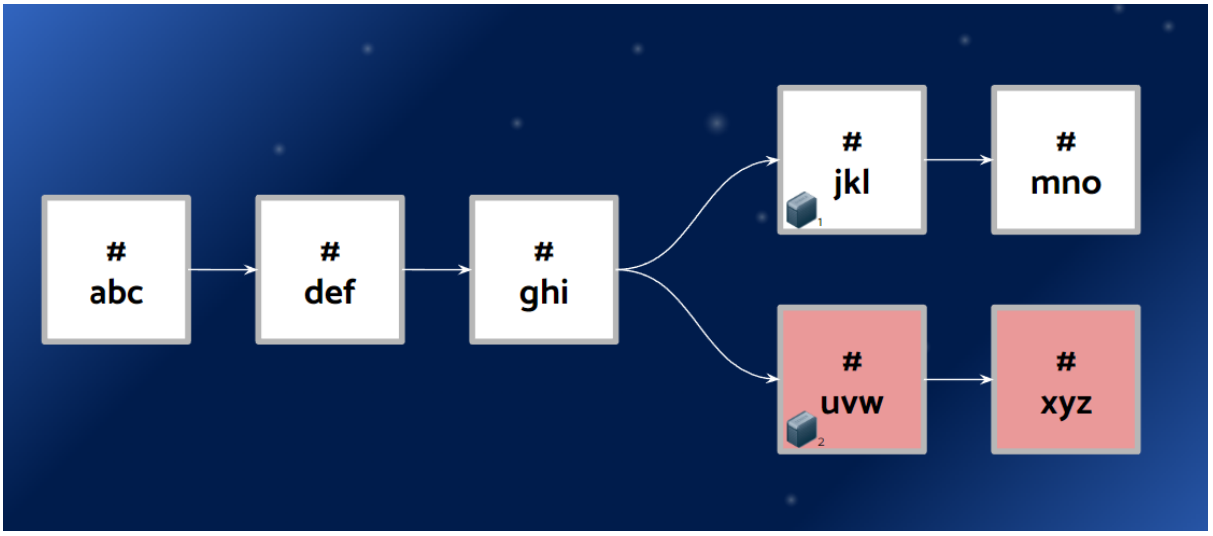

Abbildung 4: verschiedene Stränge der Blockchain

Je tiefer ein Block in der Blockchain ist, desto unwahrscheinlicher ist es, dass er Teil eines Forks ist, der eliminiert wird. Anders gesagt: Je tiefer der Block ist, desto sicherer ist es, dass die Transaktionen, die im Block festgehalten werden, langfristig gespeichert werden. Wenn ein Block allerdings erst kürzlich gemined wurde, ist es nicht auszuschließen, dass er aufgrund der Eliminierung seines Blockchain Forks gelöscht werden könnte. Um zu verhindern, dass Udocoin Benutzer Coins ausgeben, die sie aufgrund von Fork Eliminierung irgendwann nie besessen haben, ist Udocoin Guthaben erst valide, sobald die Transaktion bzw. der Mining-Reward, der zum aktuellen Kontostand des Benutzers geführt hat, fünf Blöcke tief in der Blockchain ist.

Um einen allgemeinen Konsens zu finden, müssen Regeln definiert werden, sodass einer der beiden Stränge langfristig als kanonische Blockchain erfasst wird. Ansonsten würden zwei Server verschiedene Kontostände und verschiedene Transaktionen beinhalten.

Es wurden dazu folgende Regeln definiert:

1. Die längste Blockchain ist die valide Blockchain.
2. Wenn die Blockchains gleich lang sind, ist die Blockchain valide, die im letzten Block den höheren Proof-of-Work hat.

3. Wenn auch der Proof-of-Work gleich ist, bleibt der Fork bestehen, bis ein neuer Block auf einem der beiden Forks gemined wird, was dazu führt, dass eine der beiden Blockchains die längere wird.

Das ist die Grundidee des Udocoin Konsensalgorithmus. Dieser Algorithmus läuft, wenn der Udocoin Server zum ersten Mal startet und wenn der Server erkennt, dass seine Blockchain so stark von der allgemein anerkannten Blockchain abweicht, dass eine Neuberechnung der Kontostände notwendig ist, also wenn fünf oder mehr Blöcke gleichzeitig verändert werden würden. Da dies in das valide Guthaben des Nutzers eingreifen würde, wird in diesem seltenen Fall die gesamte neue Blockchain von allen verbundenen Server angefordert und alle Kontostände neu berechnet. Dadurch müssen keine komplizierten und fehleranfälligen Kontostandrollbacks durchgeführt werden.

3.2. Netzwerkarchitektur

Das Erstellen einer passenden Netzwerkarchitektur stellte sich im Verlauf des Projekts als eine der größten Herausforderungen heraus. Wie im theoretischen Teil dieser Arbeit beschrieben, basieren Blockchains auf Peer-to-Peer-Netzwerken. Diese Netzwerktopologie setzt voraus, dass die einzelnen Knoten zueinander Verbindungen aufbauen, um kommunizieren zu können. Notwendig hierfür sind statische IP-Adressen, die eine eindeutige Identifikation jedes Netzwerkknotens ermöglichen. Statische IP-Adressen bleiben unverändert und sind im Kontext von Blockchains besonders relevant, da sie eine dauerhafte Erreichbarkeit der Netzwerkknoten sicherstellen.

Bezogen auf das Projekt würde dies bedeuten, dass eine Bedingung zum Minen eine öffentliche und statische IP-Adresse ist. Um es Nutzern der Blockchain zu ermöglichen, auch ohne dies aus dem eigenen Heimnetzwerk minen zu können, wurde sich entschlossen, zwei Klassen an Knoten in das Blockchain-Netzwerk einzubinden.

3.2.1. Serverklassen

Im Udocoin-Netzwerk wird zwischen zwei Server-Klassen unterschieden, den Seed-Servern und den Peer-Servern. Seed-Server zeichnet aus, dass sie eine

öffentliche und statische IP-Adresse besitzen, sodass andere Server eine Verbindung zu ihnen aufbauen können. Sobald eine dieser Voraussetzungen nicht erfüllt wird, muss ein Server als Peer-Server aufgebaut werden.

Beide Server-Klassen basieren auf demselben Code und werden auf dieselbe Weise gestartet. Beim Starten des Servers wird der Nutzer allerdings aufgefordert, die Klasse des Servers anzugeben, da diese sich in zwei kleinen technischen Funktionsweisen unterscheiden.
Wird ein Server als Seed-Server gestartet, gilt es, ihn in das Netzwerk einzugliedern. Er muss sich also bei den anderen Seed-Servern mit seiner öffentlichen IP-Adresse als Seed-Server registrieren, sodass sich im weiteren Zeitverlauf Peer-Server auch über ihn ins Netzwerk eingliedern können.
Der zweite Unterschied besteht in der Anzahl der Verbindungen, die ein Server zu Seed-Servern aufbaut. Seed-Server werden beim Starten eine Verbindung zu allen bekannten Seed-Servern aufbauen, wohingegen Peer-Server nur zu einem Seed-Server eine Verbindung aufbauen. Daraus ergibt sich ein Netzwerk, wie in Abbildung 5 dargestellt.

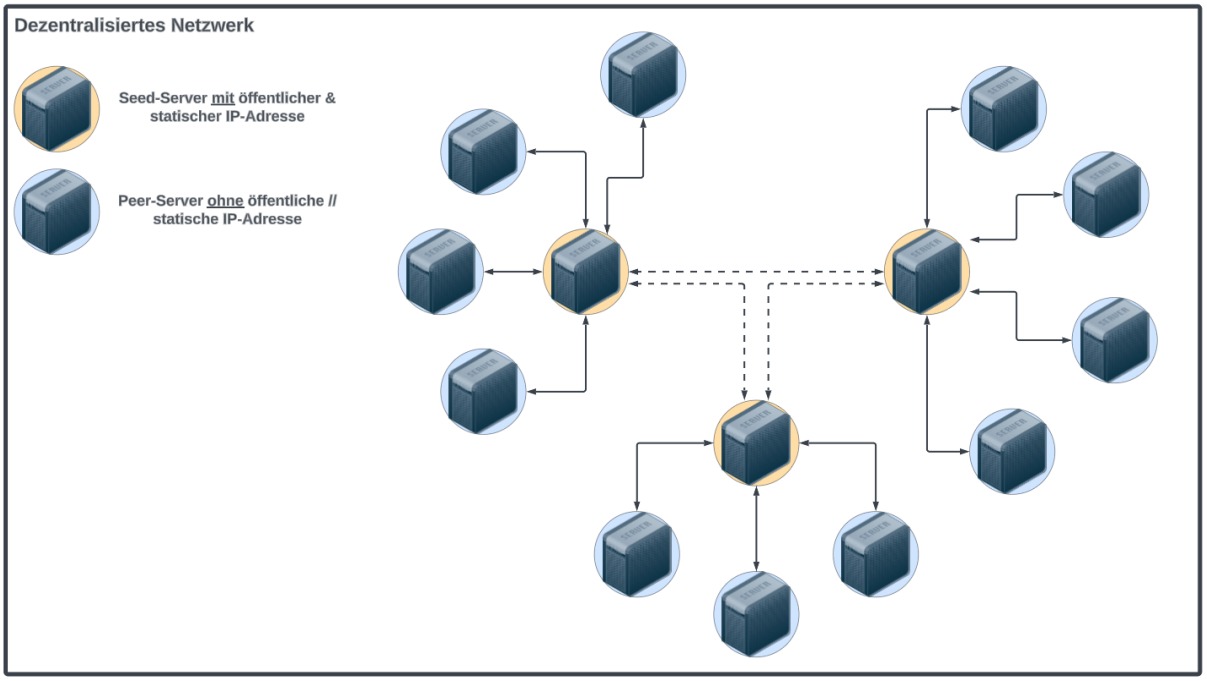

Abbildung 5: Udocoin Netzwerk Sturktur

3.2.2. Netzwerktopologie

Um die Eingliederung von nicht-öffentlich zugänglichen Knoten zu ermöglichen, basiert das Udocoin-Blockchain-Netzwerk nicht auf einem wie in gewöhnlichen Blockchain-Netzwerken (voll-) vermaschten Peer-to-Peer-Netzwerk. Die Kommunikation zwischen Peer-Servern ist immer nur über einen zentralen Knoten der Seed-Server-Klasse möglich. Aus der Sicht eines Peer-Servers lässt sich daher die Topologie des Netzwerks am besten durch eine Stern-Topologie beschreiben. Die Topologie zwischen den Seed-Servern ist vollvermascht, da ein Seed-Server beim Starten eine Verbindung zu allen anderen bekannten Seed-Servern aufbaut.

Durch die Möglichkeit für jeden Miner als Seed-Server Teil des Netzwerks zu werden, kann trotz der Ähnlichkeiten zu einem zentralisierten Netzwerk, wie in der Stern-Topologie, das Netzwerk als dezentralisiert bezeichnet werden.

3.2.3. Kommunikation innerhalb des Netzwerks

Mit dem oben erläuterten Netzwerk wird ermöglicht, ohne öffentliche und statische IP-Adresse Teil der Blockchain zu sein und somit Nachrichten zu schicken und zu empfangen. Ein Problem, das mit diesem dezentralisierten Netzwerk einhergeht, macht sich aber schon beim Starten eines Servers bemerkbar: Die öffentliche und statische IP-Adresse eines Seed-Servers muss auch im Netzwerk bekannt sein. Im Gegensatz zu einem zentralisierten Netzwerk, gibt es keine zuverlässige zentrale Server-Instanz, die von Clients angesprochen werden kann. Es kann keine IP-Adresse oder URL fest im Code hinterlegt werden. Bei einem Blockchain-Netz muss einem neuen Peer mindestens eine IP-Adresse des bestehenden Netzwerks bekannt sein, um Teil des Netzes werden zu können. Der Verbindungsaufbau zu einem aktiven Knoten ist ein bekanntes Problem der Blockchain-Technologie, welches aber weniger relevant wird, um so größer das Netzwerk ist, da mehr Knoten konsequent verfügbar sind.

Kommunikationsaufbau

Im Udocoin-Netzwerk wurde dieses Problem gelöst, indem die Liste der IP-Adressen der bekannten Knoten von dem öffentlichen Github-Repository der Kryptowährung abgerufen werden kann. Beim Starten eines Servers wird der neue Server diese abrufen alle vorliegenden IP-Adressen anpingen, um zu prüfen, ob diese verfügbar

sind. Es wird außerdem die Liste der dem Knoten bekannten IP-Adressen abgefragt, um diese mit der eigenen Liste zu vergleichen. Außerdem wird die Anzahl der aktiven Verbindungen zurückgegeben. Peer-Server werden sich somit mit dem Seed-Server verbinden, der die geringste Anzahl an bestehenden Verbindungen hat. Dies ist, um die Auslastung des Netzwerks möglichst gut zu verteilen und eine gleich schnelle Verbindung für alle Knoten zu garantieren. Besonders für Peer-Server ist es wichtig, eine große Zahl an Knoten zu kennen, um beim Ausfall ihrer aktiven Verbindung eine neue Verbindung zu einem anderen Seed-Server aufbauen zu können.

Sich als Seed-Server in das Netzwerk einzugliedern, funktioniert mit dieser Methode sehr effizient. Beim Starten eines neuen Seed-Servers “S” wird sich der neue Server bei allen Seed-Servern mit der eigenen IP-Adresse registrieren. Tritt ein neuer Peer-Server “P” dem Netzwerk bei, wird dieser die ihm bekannten Seed-Server anpingen und nach ihrer Seed-Server-Liste fragen. Teil dieser Liste ist dann die IP-Adresse von “S”. Somit wird “P” auch die Anzahl der aktiven Verbindungen von “S” abfragen, die zum aktuellen Zeitpunkt sicher weniger oder gleich groß ist, als bei anderen Seed-Servern.

Art der Verbindung

Da Peer-Server keine öffentliche IP-Adresse haben, ähnelt ihr Verbindungstyp zu ihrem Seed-Server einer Client-Server-Verbindung. Eine Eigenschaft dieser Verbindungsart ist, dass die Kommunikation einseitig ist, sodass nur der Client Anfragen schicken kann, auf die der Server dann reagiert. Dem Server ist es nicht möglich, eine Verbindung oder Anfrage zu initiieren. Dies hätte bezogen auf das Udocoin-Netzwerk zur Folge, dass Peer-Server nur Nachrichten senden, aber keine Nachrichten aus dem Netzwerk empfangen können. Um dieses Problem zu lösen, benutzen die Server Web-Sockets, um eine dauerhafte bidirektionale Verbindung zu halten. Seed-Server können somit eigenständig Anfragen über diese Verbindung an einen ihrer Peer-Server schicken, ohne dass dieser explizit eine Anfrage gestellt hat.

Broadcasts im Netzwerk

Das Verschicken einer Nachricht an alle Knoten des Netzwerks ist in einem Blockchain-Netzwerk unabdinglich. Auch wenn das Broadcasten im

Udocoin-Netzwerk im Grunde genommen wie in den meisten Netzwerken durch das Senden einer Nachricht an alle Verbindungen funktioniert, soll dies wegen der besonderen Netzwerktopologie dennoch genauer erläutert werden.

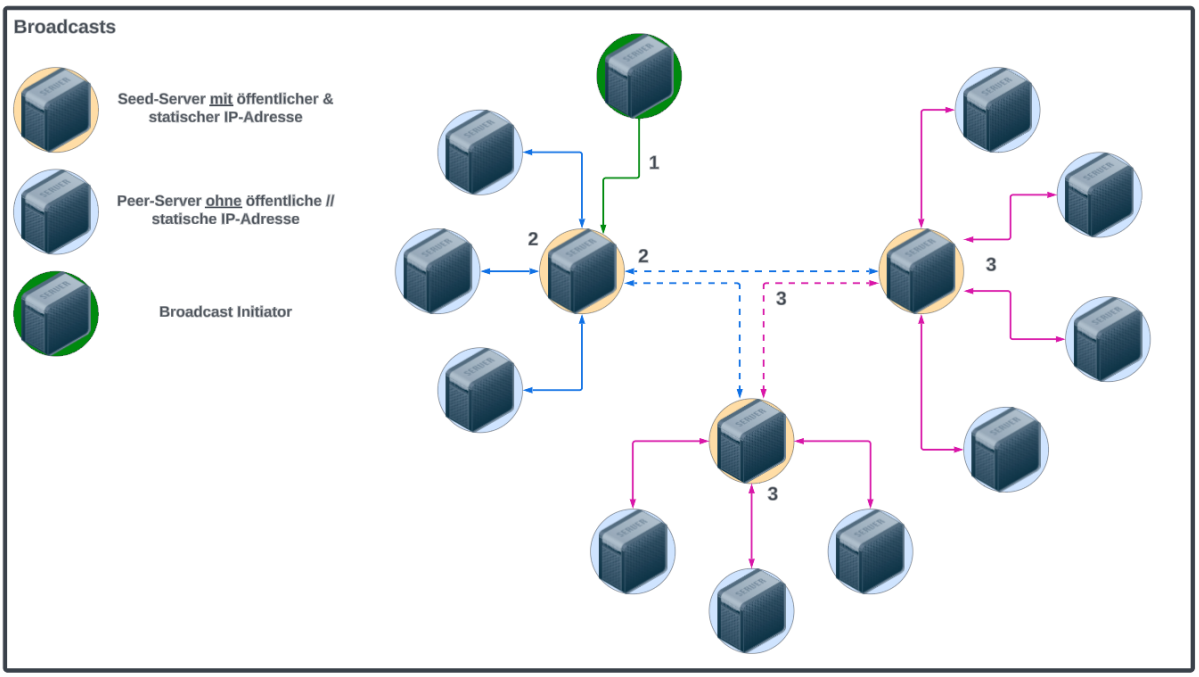

Abbildung 6: Broadcasts im Udocoin Netzwerk

Möchte ein Server, im Beispiel von Abbildung 6 ein Peer-Server, einen Broadcast senden, so wird zunächst die Nachricht an einen Seed-Server geschickt (Schritt 1). Im nächsten Schritt wird der Seed-Server die Nachricht verarbeiten und an alle seine bestehenden Verbindungen weiterleiten (Schritt 2). Hierbei werden alle seine Peer-Server und alle Seed-Server des Netzwerks abgedeckt. Die Seed-Server werden dann im letzten Schritt genauso vorgehen und die Nachricht prüfen und an alle Verbindungen weiterleiten (Schritt 3). Somit wurden alle Knoten im Netzwerk innerhalb von 3 Schritten erreicht.

Beim Benachrichtigen von allen Verbindungen wird der vernachlässigt, von wem die Nachricht ursprünglich geschickt wurde. Im Schritt 2 des oben erläuterten Broadcast-Ablaufs wird somit auch der Broadcast Initiator mit der Nachricht benachrichtigt. Es ist an diesem die Nachricht nicht erneut einen Broadcast zu initiieren. Hierfür werden Broadcasts mit einer eindeutigen ID versehen, welche beim Erhalt beziehungsweise beim Initiieren als “Erhaltener Broadcast” auf dem Server markiert werden. Bereits erhaltene Broadcasts werden nicht mehr weitergeleitet.

Ein weiterer Grund, warum ein Broadcast unterbrochen werden kann, ist, wenn ein Server eine Nachricht als ungültig einschätzt. Dies kann zum Beispiel vorkommen, wenn ein neuer Block geteilt werden soll. Lehnt ein Server den Block für seine Blockchain ab, so leitet er den Broadcast nicht weiter.

Bandbreitenoptimierung

Bei den ersten Testdurchläufen wurde bei jeder neuen Veröffentlichung eines Blocks die gesamte Blockchain gebroadcastet. Dies ist in den meisten Fällen nicht nötig, weil die einzige Änderung in der Blockchain an der letzten Stelle der Kette ist.

Daher wurde sich dazu entschieden, nur den neu veröffentlichten Block zu broadcasten. Wenn ein solcher Block bei einem Server ankommt, überprüft der Server erst, ob der Index des Blocks die Blockchain tatsächlich verlängert. Ein Block, der die Kette nicht verlängert, wird sofort abgelehnt, da er nicht den Konsensregeln entspricht. Verlängert der Block die Blockchain und stimmt der “previous_hash” im Block mit dem Hash des ehemaligen letzten Blocks überein, so wird der Block an die Blockchain angehängt und der Block wird weiter gebroadcastet. Wenn der Block die Kette verlängern würde, die Hashes jedoch nicht übereinstimmen, weil sich der Block beispielsweise auf einem anderen Blockchain Fork befindet, fordert der Server die letzten 5 Blöcke der Blockchains aller verbundenen Server an, um herauszufinden, wo der Fork aufgetreten ist. Die Zahl 5 wurde gewählt, da es sich bei diesen Blöcken um Blöcke handelt, die noch nicht zum validen Guthaben beigetragen haben. Folglich können sie ohne Rollbacks der Kontostände ersetzt werden. Wenn der Fork weiter zurück als 5 Blöcke liegt, fordert der Server die gesamte Blockchain an und berechnet alle Kontostände vom Anfang der Blockchain bis zum jetzigen Zeitpunkt. Dies ist vergleichsweise aufwändig, sollte aber nur sehr selten passieren. Es konnte bisher nicht geschätzt werden, wie häufig dies stattfinden würde, da dazu eine große Anzahl produktiver Systeme betrieben werden müsste.

3.3. Nutzung von Udocoin

Um die Kryptowährung tatsächlich nutzbar zu machen, galt es, weitere Herausforderungen zu bewältigen. Dazu gehört zum Beispiel eine Möglichkeit zu finden, Transaktionen in die Blockchain einzubringen. Wie zuvor beschrieben,

müssen Transaktionen mit einem Private-Key signiert werden. Dies führt allerdings zu einem Sicherheitsproblem, da das gesamte Blockchain-Netzwerk auf HTTP basiert und keine Verschlüsselung vorsieht. Das Mitlesen von Netzwerkverkehr durch Dritte stellt normalerweise in einer Kryptowährung keine Gefahr dar, da die gesamte Blockchain und ihre Inhalte öffentlich einsehbar sind. Deshalb ist keine SSL-Verschlüsselung notwendig und darüber hinaus kann von Minern auch nicht verlangt werden, diese auf ihren Servern einzurichten. Seinen Private-Key innerhalb des Netzwerks zu verschicken, muss also unbedingt vermieden werden, da die Keys in Klartext übertragen werden. Diese könnten von anderen sehr einfach beispielsweise durch Man-in-the-Middle-Attacken abgefangen und für den Zugriff auf eine fremde Wallet genutzt werden.

Statt Transaktionsdaten, also die Public-Keys und den Betrag, zusammen mit dem Private-Key zum Signieren an einen Server zu schicken, bietet es sich an, die Transaktion auf einem eigenen Gerät zu signieren und mit Signatur und ohne Private-Key an einen Server zu schicken. Von dort kann die Transaktion im Netzwerk verteilt und in die Blockchain eingebracht werden.

Um Transaktionen auf einem eigenen Gerät zu erstellen und zu signieren, gibt es mehrere Möglichkeiten. Ein Ansatz ist zum Beispiel, dies durch entsprechende Funktionen beim eigenen Server des Netzwerks durchzuführen. Das würde allerdings dazu führen, dass jeder, der die Kryptowährung nutzen will, einen aktiven Server braucht. Aus diesem Grund wurde beschlossen eine unabhängige und mobile Wallet App zu entwickeln, mit der der aktuelle Kontostand eingesehen werden kann und vor allem eine Transaktion erstellt und signiert werden kann.

3.3.1. Wallet App

Eine grundlegende Voraussetzung der Wallet App ist, kompatibel mit dem Blockchain-Netzwerk zu sein. Dazu gehört vor allem, die Schlüssel gleich zu verarbeiten und mit diesen Transaktionen so signieren zu können, wie es ein Server macht. Um dies zu garantieren, sollte auch von der Wallet App Python dafür genutzt werden.

Die Wallet App wurde daher für Android mit Kotlin entwickelt. Mit dem Plugin Chaquopy wird es ermöglicht, Python-Code nahtlos in Kotlin-basierte

Android-Anwendungen zu integrieren. Es bietet eine Brücke zwischen Kotlin und Python, indem es die Ausführung von Python-Code innerhalb der Android-App ermöglicht. Mit Python können somit Public- und Private-Keys erstellt und verarbeitet werden, sowie Transaktionen erstellt und signiert werden. Auch die Kommunikation mit den Servern findet über Python statt.

Um den Austausch von Schlüsseln zu vereinfachen, bietet die App außerdem die Möglichkeit, QR-Codes von Public-Keys zu lesen. Darüber hinaus wird im eigenen Profil der QR-Code vom eigenen Public-Key angezeigt. Für den Login in der eigenen Wallet wird der Private-Key benötigt, der entweder als QR-Code oder als Datei gelesen werden kann. Unter diesen Voraussetzungen können Transaktionen auf dem eigenen Gerät erstellt und signiert werden. Durch die Verbindung zu Seed-Servern können diese problemlos in das Blockchain-Netzwerk eingebracht werden.

4. Abschluss

Bis zur Vollendung des Projekts konnte eine funktionstüchtige Kryptowährung erstellt werden. Das Netzwerk und dessen Blockchain bleibt funktional, selbst wenn nur ein einziger Seed Server läuft. Es wurde außerdem eine mobile Wallet entwickelt, mit der Nutzer einfach auf die Blockchain zugreifen und mit wenigen Knopfdrücken Überweisungen der Kryptowährung tätigen können.

Für die Weiterführung des Projekts wäre eine Integration der Währung in bereits entwickelte Webseiten von vorherigen Modulen, wie zum Beispiel einem Webshop, im Rahmen der Möglichkeiten.

5. Literaturverzeichnis

[1] B. Spahic, *Bitcoin ohne Vorkenntnisse: innerhalb von 7 Tagen die Blockchain verstehen*. PBD Verlag, 2021.

[2] F. Thelen and M. Schorn, *10xDNA – Das Mindset der Zukunft*. Frank Thelen, 2020.

[3] M. Blanton, “Hash Functions,” in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds., New York, NY: Springer, 2018, pp. 1681–1682. doi: 10.1007/978-1-4614-8265-9_1482.

[4] N. Li, “Asymmetric Encryption,” in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds., New York, NY: Springer, 2018, pp. 184–185. doi: 10.1007/978-1-4614-8265-9_1485.

[5] W. Lin, “Digital Signature,” in *Trends in Data Protection and Encryption Technologies*, V. Mulder, A. Mermoud, V. Lenders, and B. Tellenbach, Eds., Cham: Springer Nature Switzerland, 2023, pp. 77–81. doi: 10.1007/978-3-031-33386-6_15.

[6] Y. Zhang, “Blockchain,” in *Encyclopedia of Wireless Networks*, X. (Sherman) Shen, X. Lin, and K. Zhang, Eds., Cham: Springer International Publishing, 2020, pp. 115–118. doi: 10.1007/978-3-319-78262-1_171.

[7] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”.

[8] “Majority attack - Bitcoin Wiki.” Accessed: Dec. 18, 2023. [Online]. Available: https://en.bitcoin.it/wiki/Majority_attack

[9] “5. Transactions - Mastering Bitcoin [Book].” Accessed: Dec. 17, 2023. [Online]. Available: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html

[10] “Elliptic Curve Digital Signature Algorithm - Bitcoin Wiki.” Accessed: Dec. 18, 2023. [Online]. Available: https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm