HSB
Hochschule Bremen
City University of Applied Sciences

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

Untersuchung der Distributed Ledger Technologien auf Zukunftstauglichkeit

---

Fakultät 4	Elektrotechnik und Informatik
Erstprüfer:	Prof. Dr. Lars Braubach
Zweitprüfer:	Prof. Dr. Martin Hering-Bertram
Abgabetermin:	15.10.2018

vorgelegt von

Jens Altrock 399144

Janusz Spatz 399341

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Seit 2017 besteht ein starker Wachstum des Interesses an Blockchain, wobei Bitcoin als prominenteste Implementation angesehen wird.(QUELLE) Am Sonntag, den 17. Dezember 2017 durchbricht Bitcoin erstmals fast die $20.000 Marke ($19,783.21) [Hig17](FOOTNOTE). Doch nicht nur Bitcoin hat von dem starken Interesse profitiert, sondern auch viele weitere Kryptowährungen sowie andere Blockchain-Implementierungen. Damit war die Technologie in das Bewusstsein der breiten Masse gerutscht und zugleich häufen sich Fragen im Bezug auf Zukunftssicherheit und Tauglichkeit der jeweiligen Technologien. Jene Fragen sollen in dieser Bachelorarbeit erörtert werden.

1.1 Kooperationen mit Blockchain/Tangle

Immer mehr kommerzielle Firmen entdecken das Potenzial von Kryptowährungen und weiteren Blockchain-Technologien.(QUELLE) Die Kooperation zwischen dem Automobilhersteller VW und diversen Kryptowährungen ist ein aktuelles Beispiel in dem eine Firma Potential in dieser Technologie sieht. In einer am 8. August, 2018 veröffentlichten Kurznachricht auf der Social-Media-Plattform Twitter schreibt die Volkswagen Group:

> „Bringing #blockchain systems to the road: We're working full steam ahead on making super-safe #cryptosystems available to

> our customers. For filling the tank, unlocking your car – and all kinds of other possibilities: #bitcoin #ethereum #iota“
>
> - Volkswagen Group auf Twitter[1]

In einem Blockeintrag der Volkswagen Group geht hervor, dass sie anhand von Blockchains, die Zuverlässigkeit und Sicherheit von Autos erhöhen wollen. Zum Beispiel indem Kilometerzähler vor Manipulationen geschützt und Hackerangriffe auf selbstfahrende Autos vorgebeugt werden.[2](QUELLE)

Auch weitere Firmen wie Bosch geben in einer Pressemitteilung Ende 2017 bekannt, dass auch sie in IOTA investieren und eng mit ihnen an verschiedenen Fronten zusammenarbeiten werden. [3](QUELLE) Sowie auch Microsoft hat sich die Kryptowährungen zu Nutzen gemacht. Die Nutzer können anhand von Bitcoin käufe im Microsoft Store tätigen [4](QUELLE).

Es ist davon auszugehen, dass viele weitere namhafte Firmen dem Beispiel folgen werden und auch Blockchain/Tangle Technologien in ihren jeweiligen Anwendungsbereichen einbinden werden. So kristallisiert sich die Möglichkeit heraus, dass Blockchains oder Tangles im zukünftigen Alltag eine erhebliche Rolle spielen werden, was zum Beispiel Transaktionen und Sicherheit betrifft.

1.2 Zielsetzung

Grundlegend soll die Frage geklärt werden, wie sich ein, auf Blockchain basierendes, Konzept für die Zukunft durchsetzen kann und mit welchen realistischen Aussichten zu rechnen ist. Hierbei ist es notwendig, vorhandene Implementierungen nach Einsatzmöglichkeit und Art der Technologie zu kategorisieren und zu beschreiben.

Um einen genauen Ausblick für die Zukunft geben zu können, muss die Entwicklung, Sicherheit und Usability von den zuvor kategorisierten Technologien ausführlich und kritisch untersucht werden. Unterstützt wird das Ergebnis der Thesis von einem selbst erstellten Programm, welches mit einer Kryp-

[1] https://twitter.com/VWGroup/status/1027205629436407810

towährung arbeitet. Dieses Programm soll eine mögliche zukünftige Nutzung von Blockchain aufzeigen. Sie soll eine Dienstleistung annehmen können, und anschließend autonom den Dienst mittels einer Kryptowährung bezahlen können. Dies ist ein Anwendungsfall, welcher im Rahmen der zuvor erwähnten Kooperation zwischen VW und IOTA auch ausgearbeitet wird.

Kapitel 2

Grundlagen

2.1 Raspberry Pi

Bei einem Raspberry Pi handelt es sich um einen Einplatinencomputer im Kreditkarten-Format. Das Gerät besitzt unter anderem USB-, Ethernet- und AUX-Eingänge, sowie einen Steckplatz für eine Micro-SD-Karte, die als Speicherort des Betriebssystems und der Daten genutzt wird. Bildinformationen werden an den HDMI-Ausgang gesendet. Als Stromquelle dient der Micro-B-Eingang der mit einer Eingangsspannung von 5 Volt betrieben werden kann. [11](QUELLE) Ursprünglich diente die Entwicklung des Raspberry Pi's dafür, Heranwachsenden in Großbritannien an das Programmieren heranzuführen. [12](QUELLE)

Der Einplatinencomputer wird mit einem auf Linux basierendem Betriebssystem genutzt. Aufgrund der Architektur des CPUs sind Betriebssysteme wie Windows nicht auf dem Raspberry Pi verfügbar. Das offizielle Betriebssystem ist das, auf Debian (Linux) basierende, Raspbian. [17](QUELLE) Es beinhaltet alle Programme die nötig sind um mit dem Programmieren des Raspberry Pis zu beginnen.

Durch die Faktoren von einer umfangreichen und integrierten Programmieroberfläche, der simplen Installation und die geringen Kosten machen den

Raspberry Pi zu einer idealen Umgebung um prototypische Anwendungen umzusetzen.

2.2 GPIO

Zahlreiche GPIO-Pins befinden sich auf dem Raspberry Pi um eine direkte Kommunikation mit externen Schaltkreisen zu erlauben. GPIO ist eine Abkürzung für General Purpose Input Output, übersetzt bedeutet dies Ein- und Ausgabe für allgemeine Einsatzmöglichkeiten. Dafür stehen auf dem Raspberry Pi bis zu 40 Pins zur Verfügung. Die einzelnen Pins können mittels Software und Hardware an- oder ausgeschaltet werden. Darüber hinaus kann ein Wert auch eingelesen oder gesetzt werden. Doch nicht jeder Pin steht für GPIO-Zwecke zur Verfügung. Einige Pins sind für andere Funktionen vorbehalten (z.B. Datenübertragung, Strompotenzial von +5V, +3.3V und 0V). Aus diesem Grund empfiehlt es sich, das Datenblatt über die Verteilung der Pins anzugucken.

Die GPIO-Pins ermöglichen es, angeschlossene Schaltkreise zu steuern oder einzulesen. Sie sind nach belieben programmierbar. Daher finden Raspberry Pis oft Verwendung in Projekten wie Heimautomatisierung, Drohnen und Sicherheitssystemen. [14][15][16](FOOTNOTE)

2.3 Steckbrett (Breadboard)

Ein Steckbrett wird genutzt um lötfrei Komponenten mit den Pins des Raspberry Pis zu verbinden. Durch Kabel werden die Pins mit einer Reihe des Bretts verbunden, wodurch an der Reihe eine Spannung angelegt wird. Nun kann an dieser Reihe eine Komponente angesteckt werden. Damit der Schaltkreis geschlossen wird, muss das andere Ende mit einem Pin verbunden werden, der geerdet ist (ein GND-Pin).

Da es damit möglich ist, schnell neue Komponenten anzuschließen und den Schaltkreis nach belieben zu verändern, ist es gut geeignet um prototypisch Schaltkreise zu erstellen.

2.4 Hash Tree (Merkle Tree)

Im Jahr 1979 patentierte Ralph Merkle das Prinzip des Hash Trees. Dieses ist später auch bekannt als “Merkle Tree”.[5](FOOTNOTE) Der Nutzen des Hash Trees wird bei der effizienten Verifizierung von Daten deutlich. Anfänglich werden Dateien, in der Regel Paare aus zwei Dateien, miteinander gehasht. Der daraus resultierende Kind-Hash wird mit dem Kind-Hash anderer Dateien zu einem gemeinsam Hash verrechnet. So entsteht nach und nach ein Baum aus diversen Hashes die schließlich einen Top-Hash ergeben. Mithilfe dieses Top-Hashes ist es möglich die Unversehrtheit jeder anfänglichen Datei zu gewährleisten.(PICTURE)

2.5 Proof-of-Work (Hashcash)

Das Proof-of-Work Konzept wurde anfänglich gegen denial-of-service Attacken oder für die Abwehr von Spam beim Email Verkehr entwickelt. Der Erfinder Adam Back nannte dieses System auch “Hashcash”.[6](QUELLE) Damit eine Person, die dieses Konzept nutzt, beispielsweise eine Email versenden kann muss sie vorher einen kleinen Aufwand betreiben. Dies könnte eine einfache mathematische Formel sein. Bei erfolgreicher Berechnung ist die Email freigegeben. Da eine Vielzahl dieser kleinen Aufwände sich zu einem großem Rechenaufwand aufstaut und da die Komplexität der Aufwände mit jeder erfolgreichen Abarbeitung steigt, wird in der Schlussfolgerung das schnelle Versenden vieler aufeinander folgender Emails verhindert.

2.6 Internet of Things (IoT)

Das Internet of Things (deutsch: Internet der Dinge) beschreibt die Verknüpfung von Geräten und Sensoren, die vorher eigenständig und isoliert waren, mit dem Internet. Damit ist es ihnen möglich Daten untereinander auszutauschen indem sie ihre eigenen Daten übertragen und externe Daten anfordern. So lassen sich die Geräte über das Internet gleichzeitig von Menschen oder anderen Maschinen fernsteuern. Mit diesem Begriff wird die Idee eingeleitet,

dass die Nutzer des Internets nicht ausschließlich Menschen sind, sondern zunehmend auch „Dinge“ (Things).

Im privaten Gebrauch werden als Beispiele alltägliche Geräte aufgeführt, die durch die Vernetzung im Internet mit anderen Geräten das Leben des Nutzers komfortabler machen oder auf sonstige Weise positiv beeinflussen. Ein Beispiel wäre die Hausautomation mit Sicherheitskameras, die, wenn etwas verdächtiges ermittelt wird, sofort den Hauseigentümer über das Internet warnen und das Videomaterial live auf dem Smartphone übertragen. [18](QUELLE)

Das Internet der Dinge lässt sich auch im industriellen Bereich anwenden. So lassen sich Herstellungsprozesse durch die Vernetzung von Sensoren, Anlagen und Maschinen so automatisieren, dass der Prozess effizienter und menschliche Hilfe immer mehr obsolet wird. [18](QUELLE)

2.7 Industrie 4.0

In diesem Zusammenhang fällt oft der Begriff Industrie 4.0. [19](QUELLE) Damit ist eine vierte industrielle Revolution gemeint. Kurz beschrieben geht die erste industrielle Revolution mit der Mechanisierung einher, die zweite mit der Massenfertigung von Produkten mittels Fließbändern und Fabriken, und die dritte mit der Automatisierung von Arbeitsschritten mittels Maschinen. [20](QUELLE) Insofern handelt es sich bei Industrie 4.0 um die Vernetzung der automatisierten Arbeitsschritte, sodass sie autonom miteinander kommunizieren können und sich dem nächsten Arbeitsschritt anpassen können.

2.8 Trits, Trytes & Tryte-Alphabet

Das Ternäre System ist eine Alternative zu dem Binären System, mit dem Unterschied dass das Ternäre System die Basis 3 hat. Ein Trit kann drei verschiedene Werte annehmen. Es ist die kleinste Einheit in dem Ternären System. Die Werte belaufen sich auf -1, 0 oder 1. Drei Trits ergeben ein

Tryte. Ein Tryte kann $3^3 = 27$ Zustände annehmen. Um die Lesbarkeit einer längeren ternären Zahl zu verbessern wurde von der IOTA-Foundation das sogenannte „Tryte-Alphabet“ erstellt. Es besteht aus den 26 Zeichen des Alphabetes in Großbuchstaben (A-Z) und der Zahl 9. Jedes Zeichen repräsentiert den Wert eines Tryte.

Tryte	Dezimal	Buchstabe	Tryte	Dezimal	Buchstabe
0, 0, 0	0	9			
1, 0, 0	1	A	-1, -1, -1	-13	N
-1, 1, 0	2	B	0, -1, -1	-12	O
0, 1, 0	3	C	1, -1, -1	-11	P
1, 1, 0	4	D	-1, 0, -1	-10	Q
-1, -1, 1	5	E	0, 0, -1	-9	R
0, -1, 1	6	F	1, 0, -1	-8	S
1, -1, 1	7	G	-1, 1, -1	-7	T
-1, 0, 1	8	H	0, 1, -1	-6	U
0, 0, 1	9	I	1, 1, -1	-5	V
1, 0, 1	10	J	-1, -1, 0	-4	W
-1, 1, 1	11	K	0, -1, 0	-3	X
0, 1, 1	12	L	1, -1, 0	-2	Y
1, 1, 1	13	M	-1, 0, 0	-1	Z

Tabelle 2.1: Das Tryte Alphabet

2.9 Distributed Ledger

Eine Distributed Ledger (wörtlich „verteiltes Kontobuch“) Technologie (kurz DLT) kann sich wie ein öffentlich einsehbares Kontobuch vorgestellt werden. Bei handelsüblichen zentralen Transaktionen kann der Zahlende sich seines Handels zwar bewusst sein, jedoch nicht über den Status des Eingangs bei dem Zahlungsempfängers. Des weiteren hat der Zahlungsempfänger keine Einsicht in die Verfügbarkeit des Geldes oder des zu transferierenden Gegenstandes des Zahlenden. Der Handelsgegenstand kann in einer weiteren Transaktion bereits reserviert sein. Bei zentralisierten Handelsaufträgen garantiert eine dritte Partei die Richtigkeit des Handels. Dabei wird das Vertrauen jedoch in diese vorausgesetzt. Eine DLT löst dieses Problem, indem

die Richtigkeit eines Handels nicht durch zentrale Parteien versichert wird, sondern durch systemimmanente Prozesse. Es besteht ein Konsens über die Richtigkeit der Daten, da sie dem System unabhängig und frei zur Verfügung stehen. Darüber hinaus sind die getätigten Handelsaufträge anonymisiert öffentlich einsehbar und nicht auf die Handelsparteien zurückzuführen. Die Richtigkeit basiert nicht länger auf dem Vertrauen in eine Zentrale Partei, sondern wird durch den Konsens des Systems gewährleistet.

Eine DLT zeichnet den gesamten Handelsverlauf auf. Jeder Handel kann auf den Ursprung zurückverfolgt werden. Dem Handelspartner ist es ein Leichtes, den Verlauf eines Transaktionswertes nachzuweisen, indem er die Historie dessen zurückverfolgt.[1]

2.10 Hash

2.11 Wallet

2.12 Restful API

2.13 Random Walk Montecarlo

2.14 Schaltkreis erstellen

2.15 Blockchain Konzept

Gentle introduction to Blockchain Technology

[1] Daten bezogen von: https://www.bafin.de/SharedDocs/Veroeffentlichungen/DE/Fachartikel/2016/fa_bj_1602_blockchain.html

Teil I

Theoretischer Teil

Kapitel 3

Einführung

- welche Distributed Ledger Technologien werden verwendet
- warum werden sie verwendet
- welche Kryptowährungen werden als beispiel für die ausgewählten Distributed Ledger technologien genommen
- Bitcoin für blockchain und IOTA für tangle

Kapitel 4

Funktion

4.1 Bitcoin

Bitcoin ist ein Distributed Ledger Protokoll das erstmalig Geldtransfer über eine dezentrale, anonyme Übereinstimmung der Richtigkeit verfügt. Diesen Konsens erreicht es über die implementierte Blockchain, die wie ein global einsichtiges Kassenbuch funktioniert. Die Blockchain kann auf verschiedene Art und Weise genutzt und entwickelt werden. Sie ermöglicht es über das Internet Werte auszutauschen ohne von einem Mittelsmann abhängig zu sein [EGSvR16]. Durch kryptographische Techniken sind Daten die in die Blockchain gelangen nur durch sehr großen Aufwand manipulierbar, was sie für das Speichern von sensiblen Daten attraktiv macht. Im Folgenden wird die grundlegende Funktion der Blockchain sowie dessen Einsatz bei Bitcoin sachlich untersucht.

4.1.1 Allgemeine Funktion

Für die gegenwärtige Funktion von Bitcoin und dessen Blockchain sind zwei elementare Techniken notwendig. Zum Einen ist das die Public-Key-Kryptographie zum anderen die kryptographischen Hashfunktionen.

Bei der Public-Key-Kryptographie resp. digitalen Signatur erstellt der Sender

einer Nachricht ein Schlüsselpaar bestehend aus einem privaten sowie einem öffentlichen Schlüssel. Die beiden Schlüssel haben zwei ergänzende Funktionen. Der Autor der Nachricht verwendet den privaten Schlüssel um diesen mit seinen Informationen zu verbinden und dadurch zu signieren. Die signierten Daten werden gemeinsam mit dem öffentlichen Schlüssel an den Empfänger weitergegeben. Dank des öffentlichen Schlüssel ist es dem Empfänger möglich die Daten zu authentifizieren. Des Weiteren ist durch die Verknüpfung der Daten mit dem privaten Schlüssel die inhaltliche Integrität vor Manipulation geschützt.

Durch kryptographische Hashfunktionen entstehen aus Zeichenketten mit variabler Länge Zeichenketten fester Länge. Diese Zeichenketten sind "deterministisch". Die gleichen Eingangsdaten werden nach der Hash-Berechnung immer den gleichen Hash verursachen. Veränderte Eingangsdaten führen zu einem stark abweichenden Hash.

Außerdem gibt es drei weitere nennenswerte Eigenschaften von Hashfunktionen. Zum Einen ist es nahezu unmöglich durch den Hash die anfänglichen Daten wiederherzustellen. Des Weiteren ist es nahezu unmöglich mit abweichenden Eingangsdaten denselben Hash zu generieren. Zuletzt ist es nahezu unmöglich zwei verschiedene Eingangsdaten zu finden aus denen sich derselbe Hashwert ergibt. [RS]

4.1.2 Privater & öffentlicher Schlüssel

Der private Schlüssel in Bitcoin kennzeichnet den Besitz der übertragenden BTCs (Geldeinheit). Sollte der Schlüssel verloren gehen, ist es nicht möglich die BTCs, die dem privaten Schlüssel zugeordnet sind, wiederherzustellen. Der öffentliche Schlüssel wird für die Verifikation einer Transaktion benötigt, da er durch den privaten Schlüssel generiert wurde. Man kann durch den öffentlichen Schlüssel, nach heutigem Stand, nicht den privaten Schlüssel erraten. Dieses Ziel wird durch die folgende Kryptographie erreicht:

Endliche Felder Ein endliches Feld ist eine Gruppe aus Zahlen, das, wie der Name suggeriert, endlich ist. Ein endliches Feld, das mit Zahlen des

Modulo P gebildet wird, bei dem P eine Primzahl ist, führt zu nützlichen Funktionen in der Kryptographie.

Elliptische Kurven In der Mathematik sind elliptische Kurven aufgrund ihrer Eigenschaft, mathematische Gruppen zu sein, interessant. Eine elliptische Kurve wird nach der folgenden Formel gebildet:

$$\{(x, y) \in \mathbb{R}^2 \mid y^2 = x^3 + ax + b,\ 4a^3 + 27b^2 \neq 0\} \cup \{0\}$$

(PICTURE) Für jeden Punkt der elliptischen Kurve gelten die Gesetze der Kommutativität, Assoziativität und Distributivität. Darüber hinaus gelten weitere, für elliptische Kurven spezifische, Regeln:

- Die Inverse -P eines Punktes P kann durch das Spiegeln des Punktes P an der x-Achse bestimmt werden.
- Wenn zwei Punkte (P, Q) einer elliptischen Kurve bekannt sind, kann immer auch ein dritter Punkt (R) bestimmt werden. Die Addition der drei Punkte ergibt in jedem Fall $P + Q + R = 0$. Insofern wäre die Rechnung der Punkte zur Bestimmung von Punkt R: $P + Q = -R$. Punkt -R ist die Inverse von Punk R gespiegelt an der x-Achse.

Eine Formel die diese beiden Konzepte, der elliptischen Kurven und der endlichen Felder, vereint sieht wie folgt aus:

$$\{(x, y) \in \mathbb{R}^2 \mid y^2 \equiv x^3 + ax + b\,(mod\,p),\ 4a^3 + 27b^2 \not\equiv 0\,(mod\,p)\} \cup \{0\}$$

Gruppe aus den Reellen Zahlen der elliptischen Kurven des modulo P [Bro09]

Dank des Moduls p befindet sich die Gleichung in einem endlichen Feld. Eine elliptische Kurve die sich in einem endlichen Feld befindet ist weiterhin in eine mathematische Gruppe. Alle Formeln können wie zuvor beschrieben angewandt werden.

Tritt der Fall ein, dass Punkt P sich wie eine Tangente zu der elliptischen

Kurve verhält, wird das sogenannte Prinzip der Punktdopplung genutzt:

$$P + Q = -R;\ Q = P \rightarrow 2P = -R$$

Es ist möglich den Punkt P um eine beliebige Anzahl x zu skalieren um einen Punkt R zu erreichen.

$$xP = R$$

Da sich Punkt P in einem endlichen Feld befindet, wird durch ein, abhängig des Punktes P, gewählter bestimmter Skalar x dafür Sorgen, dass das Produkt auf die Ausgangsposition P verweist. Dieses Prinzip lässt sich gut durch den Zeiger einer Uhr verdeutlichen. Die Ausgangsuhrzeit ist 3 Uhr. Der Zeiger wandert in drei Stunden Schritten voran. Nach vier Schritten zeigt der Stunden Zeiger erneut auf die Ausgangsuhrzeit 3 Uhr.

Es entsteht eine Untergruppe die durch den Punkt P definiert ist. Die Ordnung n dieser Gruppe P wird so gewählt dass nP = 0 ist. In dem Beispiel der Uhrzeit entspricht die Ordnung n = 4 denn $4(3)\ mod(12) = 0$.

Bitcoin nutzt folgende Werte für die elliptische Kurven Kryptographie:

1. Der primäre Wert des Moduls zum Bestimmen des endlichen Raumes entspricht: $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \rightarrow$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F (hex).

2. Die elliptische Kurve wird mit den Parametern $a = 0$ und $b = 7$ gebildet.

3. Die Basis der Untergruppe P beträgt in hexadezimal: 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8 (hex).

4. Die Ordnung n der Untergruppe P beträgt in hexadezimal: FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C

D0364141 (hex).

Der private Schlüssel ist eine Zahl zwischen 1 und der angegebenen Ordnung. Um den öffentlichen Schlüssel zu errechnen, wird der private Schlüssel mit dem Basispunkt der Untergruppe P multipliziert. Das Ergebnis *mod* des primären Wertes des Moduls ergibt den öffentlichen Schlüssel. Dieser entspricht einem Wert der durch nahezu unendlich Möglichkeiten erreicht werden kann. Nur der Besitzer des privaten Schlüssels kann diesen Wert gezielt reproduzieren. [Bro09]

4.1.3 Transaktionen

Um eine Transaktion im Bitcoin Netzwerk zu veranschaulichen wird ein fiktives Beispiel herangezogen. Alice möchte an Bob zwei Bitcoins (BTC) versenden. Es existieren jedoch keine Konten oder Kontostände, sondern ausschließlich die öffentliche Liste aller jemals getätigten Transaktionen, also die Bitcoin - Blockchain selbst, sowie die Hashwerte der einzelnen öffentlichen Schlüssel denen die existierenden Transaktionen zugeordnet werden. Aufgrund der vergangenen Transaktionen kann der Wertbestand an BTCs des zugehörigen privaten Schlüssels ermittelt werden. Mithilfe einer digitalen Software (Wallet) kann dieser Wertbestand des privaten Schlüssels eingesehen und neue Transaktionen eingereicht werden. Um eine Transaktion zu veranlassen werden mithilfe des privaten Schlüssels die zurückliegenden eingegangenen Transaktionen signiert und zusammen mit der ausgehenden Transaktion an das Bitcoin Netzwerk versandt. Dies entspricht in dem anfänglichen Beispiel einer Auflistung aller Transaktionen in denen Alice involviert war, sowie Alice' Intention zwei BTCs an den öffentlichen Schlüssel von Bob zu senden.

Diese Nachricht gelangt an einen Netzknoten der über verschiedene Ressourcen verfügt. Neben einer aktuellen und vollständigen Kopie der Blockchain, einen Cachespeicher („Unspent Transaction Output“ UTXO), der Transaktionswerte der Blockchain enthält die noch nicht für neue Transaktionen verwendet wurden, existiert noch eine Datenbank mit unbestätigten Transak-

tionen. In diesem Netzknoten wird die Legitimität von Alice' Transaktion überprüft, indem zum einen ihre Liquidität aufgrund ihrer bisherigen Transaktionen, sowie gleichzeitig ihre Signatur anhand des öffentlichen Schlüssels, bestätigt wird. Mithilfe der UTXO wird außerdem geprüft, ob die genutzten BTCs nicht bereits anderweitig reserviert wurden. Wenn dies fehlerfrei geschieht, wird die unbestätigte Transaktion in die dafür eingerichtete Datenbank aufgenommen. Der Netzknoten meldet diese eingereichte Transaktion an möglichst viele weitere Netzknoten weiter, die wiederum die Daten nach dem erklärten Schema überprüfen und anschließend in die Datenbank der unbestätigten Transaktionen aufnehmen. [SSUF16]

4.1.4 Mining

Das Bitcoin-Netzwerk verfügt über zwei Arten von Netzwerkknoten. Einer ist ausschließlich für das zuvor genannte Verifizieren und Speichern der eingehenden Transaktionen sowie dessen Weiterleitung an weitere Netzknoten zuständig. Der zweite ist darüber hinaus befähigt neue Blöcke in der Blockchain zu speichern. Dieser sogenannte "Mining-Netzknoten" speichert zunächst unbestätigte Transaktionen in einem Block zusammen. Dieser Block besitzt zusätzlich einen Blockheader, der für die kommende Abspeicherung in der Blockchain essentiell ist.

Es ergeben sich ab diesem Zeitpunkt einige Problematiken. Aufgrund von Verzögerungen im Netzwerk oder möglichen Ausfällen sind einige Netzknoten aktueller. Dies führt dazu, dass verschiedene Transaktionen, welche sich voneinander unterscheiden, in den einzelnen Netzknoten existieren. Außerdem können bösartige Absender gefälschte Transaktionen weiterleiten, welche dieselben Wert-Ressourcen aufweisen. Bitcoin verwendet für diese Problematiken die Proof-Of-Work (PoW) Technologie. In der Regel wird diese Technik für das Verhindern von missbräuchlicher Benutzung von Diensten eingesetzt. Damit ein Dienst genutzt werden kann muss bei dieser Technik ein gewisser Aufwand erbracht werden. Im Falle der Bitcoin-Blockchain muss gemäß dieses Schemas eine rechenintensive Aufgabe gelöst werden. Der Block-Header ergibt einen gewissen Hashwert. Dieser muss solange manipuliert werden bis

das Ergebnis unterhalb eines gewissen Zielwertes liegt. Diese partielle Hashinversion beruht auf dem Hashcash-Prinzip von Adam Back (QUELLE). Der Hash entspringt dem Block-Header und wird aus einer Referenz zum vorherigen Block, einem Zeitstempel, dem Zielwert des Hash-Rätsels, dem Top Hash eines Merkle-Trees, welcher alle Transaktionen durch aufeinander aufbauende Hashes zusammenfasst, sowie einer variablen Zeichenfolge (Nonce) errechnet. Die Nonce wird so oft geändert bis der Hashwert unterhalb des Zielwertes liegt. In anderen Worten muss dieser mit einer gewissen Anzahl von Nullen beginnen. Der erste Mining-Netzknoten der dieses Rätsel löst versendet seinen Block an das Netzwerk. Dort wird der Block ebenfalls auf Validität geprüft und bei Erfolg in die eigene Blockchain integriert.

In 1,69% der Fälle, finden zwei Netzknoten zu nahezu der gleichen Zeit eine Lösung des Hash-Rätsels und versenden ihre jeweiligen Blöcke. In diesem Fall erhalten unabhängige Netzknoten verschieden Versionen der Blockchain. Infolgedessen teilen sich die Netzknoten auf. Dieses Phänomen ist als Gabelung bekannt. Die jeweiligen Netzknoten arbeiten weiter auf Grundlage ihrer bekannten Blockchain bis zu dem Moment bei dem sie über eine längere Blockchain Version informiert werden. Tritt dieser Fall ein, wird die bekannte Blockchain durch die längere ausgetauscht. Sollte die längere Version einige Transaktionen der zuvor bekannten Blockchain nicht enthalten, werden diese wieder in die Datenbank der nicht bestätigten Transaktionen aufgenommen. Momentan wird ca. alle zehn Minuten ein neuer Block in die Blockchain aufgenommen. Damit sich diese Zeitspanne weiter in einem angemessenen Rahmen befindet, wird stetig die Komplexität des Hash-Rätsels angepasst. Für das Finden der Lösung eines Hash-Rätsels und das erfolgreiche Integrieren eines neuen Blockes in der Blockchain, erhält der Mining-Netzknoten eine gewisse Menge an BTCs, wodurch neue BTCs(QUELLE) erzeugt werden. [SSUF16]

4.2 IOTA

IOTA ist ein Distributed Ledger Protokoll, welches eine grenzenlose Skalierbarkeit des eigenen Netzwerkes zulässt. Dies wird durch die Eigenschaft ermöglicht, das der Benutzer und der Miner nicht länger voneinander getrennt sind. In IOTA wird das Prinzip der durch Bitcoin bekannten Blockchain umgedacht. Transaktionen werden nicht länger in einem Block gespeichert, sondern über ein Netz aus Transaktionen verteilt. In diesem Netz bestätigen sich die jeweiligen Transaktionen gegenseitig ihre Richtigkeit. Dieses Netz wird Tangle genannt und kann sich wie ein „Directed Acyclic Graph“ vorgestellt werden. Die folgende Untersuchung liefert einen sachlichen Einblick in die Funktion IOTAs und der implementierten Tangle.

4.2.1 IOTA Seed

Der IOTA Seed ist essentiell wichtig für die Funktion von IOTA. Er fungiert als „Kontonummer“ und verweist auf die Menge an IOTAs die dem Konto zugeordnet werden (das IOTA-Wallet). Wenn der Besitzer des Seeds eine Transaktion durch die IOTA-Tangle veranlassen möchte, wird mit Hilfe von kryptographischen Hashfunktionen eine Adresse aus dem Seed erstellt. Diese Adresse kann nun genutzt werden um IOTAs zu empfangen oder, sollte sie bereits einen Wert besitzen, zu versenden. Adressen können nur durch den Seed generiert werden und es ist (technisch) unmöglich von der Adresse den Seed wiederherzustellen. Ein Seed besteht aus 81 Trytes die dank des von der IOTA-Foundation zur Verfügung gestellten „Tryte-Alphabet“ durch die 26 Großbuchstaben A-Z des Alphabets und der Zahl 9 dargestellt werden. [1]

4.2.2 Tangle

Bei Tangle handelt es sich um ein Distributed Ledger Software Protokoll, welches sich grundlegend von Blockchain unterscheidet. Die Tangle-Technologie wird von den Entwicklern als nächste evolutionäre Weiterentwicklung der

[1] Diese Vorgaben sind von der IOTA Foundation veröffentlicht: https://iota.readme.io/docs/seeds-private-keys-and-accounts

Blockchain beschrieben [Pop17]. Die Technologie nimmt sich den Schwächen der Blockchain an und integriert Lösungen für diese. Konzeptionell bedeutet dies, dass die heterogene Unterscheidung von Minern und Usern in Blockchains homogenisiert werden soll, indem Transaktionen von Nutzern verifiziert werden sollen, die selber eine Transaktion veranlassen möchten. So löst sich das Problem der Skalierung und zugleich fallen die Transaktionsgebühren weg. Letzteres ist möglich, da es zwingend notwendig ist, andere Transaktionen zu verifizieren, falls eine eigene Transaktion aufgegeben werden soll. So „zahlt“ der Nutzer die Gebühren mit Rechenleistung.

Tangle wurde von der IOTA-Foundation entwickelt und kommt in der gleichnamigen Kryptowährung IOTA zum Einsatz.

Anders als bei Blockchain ist es nicht möglich, neue Tokens zu „minen“ (bedeutet: herzustellen). Es besteht seit der Entwicklung ein fester Betrag an verfügbaren Tokens von genau 2.779.530.283 MIOTA (1 MIOTA = 1.000.000 IOTA) [2]. Die Tangle richtet sich nach einem Mathematischen Konzept, dem Directed Acyclic Graph. Mit diesem Konzept speichert IOTA die Transaktionen. Eine Transaktion muss von einem Node aufgegeben werden. Sobald eine Transaktion nicht verifiziert werden kann, kann die ursprüngliche Transaktion nicht ausgehen. [Pop17]

4.2.3 Directed Acyclic Graph

Die IOTA-Tangle ist aufgebaut in Form eines „Directed Acyclic Graph“ (DAG) was in diesem Kontext bedeutet, dass sie sich in eine Richtung bewegt und niemals kreisförmig ist. Damit eine neue nicht verifizierte Transaktion in die Tangle aufgenommen werden kann, ist neben dem Proof-of-Work eine Verifikation von zwei ebenfalls nicht verifizierten Transaktionen notwendig. Diese beiden Transaktionen werden in dem Transaktions-Bundle abgespeichert und sind dementsprechend referenziert. Da nur nicht verifizierte Transaktionen in diesem Prinzip verwendet werden, entspricht der entstehende

[2] Daten bezogen von: https://coinmarketcap.com/currencies/iota/

Graph eines DAG. [3]

4.2.4 Transaktion

Eine Transaktion in IOTA besteht aus mehreren Hashes und Werten, die jeweils untereinander eine verschiedene Aufgabe erfüllen. In den folgenden Abschnitten werden diese detailliert aufgeführt.

signatureMessageFragment Sollte die Transaktion eine ausgehende Zahlung sein, wird dieses Feld benötigt um die Signatur des privaten Schlüssels zu enthalten. Die Länge der Signatur ist davon abhängig, mit welcher Sicherheitsstufe diese Transaktion gehasht wird. Sollte die Sicherheitsstufe 2 oder 3 sein, wird eine weitere wertlose Transaktion benötigt. Diese speichert in dem noch freien Signature Message Fragment den Rest der Signatur der vorangehenden ausgehenden Zahlung.

Sollte jedoch die Signatur des privaten Schlüssels nicht erforderlich sein, verbleibt dieses Feld leer und kann für das Übertragen einer Nachricht genutzt werden. Diesem Feld werden 2187 Trytes reserviert.

hash In diesem Feld wird der „transaction Hash“ nach dem Finden der „nonce“ und dem Proof-of-Work abgespeichert. Die Länge beträgt 81 Trytes.

address Die Adresse der Transaktion wird erneut für verschiedene Aufgaben verwendet, die sich aus der Art der Transaktionen definieren. Sollte eine Zahlung durchgeführt werden, wird in diesem Feld die Adresse des Empfängers angegeben. Handelt es sich jedoch um eine ursprüngliche Geldeingangs-Adresse so ist hier eine Adresse aufgeführt die aus einem private Key des Besitzer's Seed erstellt wurde. Für dieses Feld sind 81 Trytes reserviert.

value Der Wert einer Transaktion definiert dessen Art. Sollte dieser Wert positiv sein, handelt es sich um eine zahlende Transaktion (Output). In dem

[3] Daten bezogen von: https://www.forbes.com/sites/shermanlee/2018/01/22/explaining-directed-acylic-graph-dag-the-real-blockchain-3-0

Feld „Address“ wird sich in diesem Fall die Adresse des Empfängers befinden und das Feld „Signature Message Fragment“ ist entweder leer oder beinhaltet eine benutzerspezifische Nachricht.

Sollte der Wert negativ sein handelt es sich bei der Transaktion um eine empfangene Transaktion (Input). Das Feld „Address“ enthält entsprechend eine Adresse die dank des Seeds und einem daraus entstandenen private Key generiert wurde. Eine eingehende Transaktion enthält einen negativen Wert, um den positiven Wert einer ausgehende Zahlung im Bundle zu rechtfertigen. Zwangsläufig wurde diese Adresse zuvor von einer fremden IOTA-Transaktion als Empfänger Adresse genutzt.

Um den Besitz des entsprechenden privaten Schlüssels zu beweisen, wird in dem „Signature Message Fragment“ die Signatur durch den privaten Schlüssel abgespeichert. Sollte die Sicherheitsstufe größer als 1 sein, werden weitere Transaktionen benötigt, um die gesamte Signatur zu speichern.

Trägt dieses Feld keinen Wert und entspricht 0, handelt es sich bei der Transaktion entweder um das Versenden einer einfachen Nachricht an eine Empfängeradresse oder wird genutzt um die Signatur einer vorangehenden Transaktion zu speichern.

Für dieses Feld wurden 27 Trytes reserviert.

obsoleteTag Der obsolete Tag enthält einen vom Benutzer definierten Tag. Dieser könnte in zukünftigen IOTA - Iterationen entfernt werden. Die reservierte Länge dieses Feldes beträgt 27 Trytes.

timestamp Der Zeitpunkt ist nicht verpflichtend und kann ausgelassen werden. Ihm werden 9 Trytes reserviert.

currentIndex Dieses Feld zeigt die momentane Position im Bundle. Ihm werden ebenfalls 9 Trytes reserviert.

lastIndex Dieses Feld führt den Index der letzten Transaktion des umfassenden Bundles auf und liefert dementsprechend die Länge dessen. Reserviert

werden erneut 9 Trytes.

bundle Dieses Feld enthält den Hash des umfassenden Bundles. Er wird genutzt um alle Transaktionen dieses Bundles zu gruppieren. Jede Transaktion in einem Bundle enthält in diesem Feld denselben entsprechenden Bundle Hash. Das Feld umfasst 81 Trytes Länge.

branch- & trunkTransaction Diese beiden Felder sind jeweils 81 Trytes lang und enthalten zwei zufällig gewählte, nicht verifizierte Transaktionen aus der Tangle. Nicht verifizierte Transaktionen im Tangle werden “Tips“ genannt. Jede Transaktion ist verpflichtet, zwei zufällig gewählte Tips in der Tangle zu verifizieren, um selbst als Tip aufgenommen zu werden.

tag Dieser Tag wird vom Benutzer gewählt und kann frei vergeben werden. Er kann die Suche einer Transaktion im Tangle unterstützen. Seine Länge in der Transaktion beträgt 27 Trytes.

attachmentTimestamp Dieses neun Trytes große Feld beinhaltet den Zeitpunkt direkt nachdem der Proof-of-Work durchgeführt wurde.

attachmentTimestampLowerBound & attachmentTimestampUpperBound Diese beiden Felder zeigen ein Intervall auf in der die Aufnahme in die Tangle geschah. Sie sind jeweils 9 Trytes groß.

nonce Die nonce ist ein besonders wichtiger Teil einer Transaktion, denn sie wird benötigt um den Proof-of-Work durchzuführen. Die Länge dieses Feldes beträgt 27 Trytes.

4.2.5 Zusammenfassung

Rechnet man die Längen aller Felder zusammen, so erhält man die gesamte Länge von 2673 Trytes. Dies ist die von der IOTA-Foundation vorgegebene Länge die eine Transaktion haben soll. Die Art einer Transaktion bestimmt sich durch den Wert. Sollte eine Transaktion einen negativen Wert haben,

so handelt es sich dabei um eine „Input“ Transaktion. Bei einem positiven Wert, wird Balance an eine „Output“ Adresse geschickt. Viele Felder tragen eine wichtige Aufgabe für die Aufnahme in der Tangle. So wird das Feld „signatureMessageFragment„ entweder für benutzerspezifische Nachrichten genutzt oder, falls benötigt, für die Signatur.[4]

4.2.6 Proof-of-Work

Um Transaktionen in die Tangle zu platzieren, sind keine Gebühren notwendig. Das ist aufgrund zwei essentieller Eigenschaften möglich. Zum Einen muss jede Transaktion zwei weitere nicht verifizierte Transaktionen bestätigen, des Weiteren wird mit Rechenleistung in Form eines Proof-of-Work die Transaktion beglaubigt. Im Detail läuft dieses Verfahren wie folgt ab: Nachdem alle Felder der Transaktion, bis auf die „nonce“ und den Hash für „bundle“, einen Wert erhalten haben, wird mit Hilfe eines Algorithmus die nonce gesucht. Diese ist 27 Trytes lang. Sobald die nonce gefunden wurde, wird sie für die letzten 27 Trytes der insgesamt 2673 Trytes der Transaktion verwendet. Diese Trytes werden durch den Curl Hash Algorithmus zu einem 81 Trytes langen Hash umgewandelt. Der Hash wird im Anschluss in Trits umgerechnet. Am Ende der resultierenden 243 Trits soll eine Folge aus Nullen stehen. Die länge der Folge wird von der IOTA-Foundation vorgegeben. So müssen die letzten Trits der Trytes, eines Transaktions Hashes, momentan mindestens 14 mal Null in Folge für die Aufnahme in der Tangle und Neun mal Null in Folge für die Aufnahme in dem Testnetzwerk betragen. Diese Vorgabe wird „Minimum Weight Magnitude (MWM)“ genannt. Der Vorgang wird sooft wiederholt bis das vorgegebene MWM erreicht ist. Grundsätzlich kann die Aussage getroffen werden, dass mit einer steigenden MWM auch die Zeit, die für das Finden der passenden nonce benötigt wird, exponentiell ansteigt.

[4]Diese Vorgaben sind von der IOTA Foundation veröffentlicht: https://iota.readme.io/docs/the-anatomy-of-a-transaction

4.2.7 Bundle

IOTA ist nach einem Account - Schema aufgebaut, was genauer bedeutet, dass Adressen benötigt werden auf die Balance (der Fachausdruck des Geldwertes) eingegangen ist, um Balance weiter zu senden. Ein Bundle in IOTA umfasst in der Regel vier Transaktionen. Die erste Transaktion enthält die Adresse des Empfängers. Diese Transaktion wird mit einer positiven Balance versehen. In dem Bundle ist dies eine sogenannte „Output“ - Transaktion. Daraufhin folgen in der Regel, abhängig von der gewählten Sicherheitsstufe, zwei Transaktionen. Jede dieser Transaktionen basiert auf der gleichen Adresse, auf der zuvor Balance eingegangen ist. In anderen Worten muss diese Adresse zuvor Teil einer „Output“ - Transaktion gewesen sein. Es werden mehrere Transaktionen benötigt, da die Signatur des privaten Schlüssels mit versandt wird. Diese Signatur vergrößert sich, abhängig von der gewählten Sicherheitsstufe. Eine Transaktion, die sich einer Adresse bedient, die zuvor Balance empfing, nennt sich „Input“ - Transaktion. Sollte die Balance der Input Transaktion nicht ausreichen um den Betrag der Output Transaktion zu decken, müssen weitere „Input“ - Transaktionen angefügt werden, die jeweils durch ihren privaten Schlüssel signiert werden. Sollte die Balance der „Input“ - Adressen den Wert der „Output“ - Transaktion überschreiten, wird der Restwert an eine „Output“ - Adresse des Besitzers übertragen. Solange der summierte Wert der „Input“-Adressen nicht überschritten wird, können beliebig viele „Output“-Adressen einem Bundle angehangen werden.

Da Bundles atomar sind, werden entweder alle oder keine Transaktion verifiziert.

4.2.8 Signatur

Bei IOTA werden ausgehende Zahlungen (Output) mit zuvor eingegangenen Zahlungen (Input) durchgeführt. Um den Besitz des notwendigen Inputs nachzuweisen, verwendet IOTA die „Winternitz One Time Signature“. Diese Art der Signatur veröffentlicht einen Teil des privaten Schlüssels und wird aus diesem Grund für den einmaligen Gebrauch von Signaturen verwendet. Um

die Funktionsweise einer „One Time Signature“ (kurz OTS) zu verdeutlichen, wird die ähnliche Lamport OTS in einem fiktiven Beispiel herangezogen.

Ein privater Schlüssel besteht aus zwei gleichlangen Zahlenfolgen „k1“ & „k2“. Die Länge des jeweiligen Schlüssel Teils stimmt mit der Länge der zu signierenden Nachricht überein. In diesem Beispiel beträgt die Schlüssellänge 512 (2x 256). Die Nachricht kann eine beliebige Zeichenkette sein, die bspw. durch den SHA-256 Hash Algorithmus zu einem Hash „H“ mit der Länge von insgesamt 256 Bits umgewandelt wurde. Der Wert eines Bits kann ausschließlich Null oder Eins betragen. In einer Schleife wird jeder Wert des Hashes durchlaufen. Beträgt der Wert H(n) des Hashes an der Stelle n => H(n) = 0, so wird der Wert des ersten Schlüssel Teils an selbiger Stelle für die Signatur Sig(n) = k1(n) verwendet. Sollte jedoch der Wert H(n) des Hashes an der Stelle n => H(n) = 1 betragen, wird der Wert des zweiten Schlüssel Teils in der Signatur Sig(n) = k2(n) verwendet. Nach dem erfolgreichen Durchlaufen des Hashes „H“ wurde eine Signatur „Sig“ mit einer Länge von 256 angefertigt. Die Signatur enthält 50% der Werte des privaten Schlüssels.(PICTURE)

Damit der Empfänger die Signatur überprüfen kann, benötigt er den öffentlichen Schlüssel. Dieser ist bspw. ein, durch den SHA-256 Hash Algorithmus angefertigter, Hash „pub“ des privaten Schlüssels. Der Empfänger erhält die Nachricht, die Signatur “Sig” und den öffentlichen Schlüssel „pub“. Er wiederholt den Vorgang, den der Versender der Nachricht für die Signatur angewandt hat, mit dem öffentlichen, mitgelieferten Schlüsselpaar „pub1“ & „pub2“ anstatt des privaten Schlüsselpaars „k1“ & „k2“. Als Resultat erhält er eine gehashte Variante „HSig“. Sollte jeder Wert der „HSig(n)“ identisch mit jedem Hash-Wert der Signatur „SHA-256(Sig(n))“ sein, ist die Integrität der Nachricht gewährleistet. [CY04](PICTURE)

Kapitel 5

Sicherheit

Für die Beantwortung der zugrundeliegenden Fragestellung, ob die ausgewählten Distributed Ledger Technologien zukunftsfähig sind, müssen die Sicherheitsmechanismen untersucht werden, durch die diese Technologien vor Angriffen geschützt sind. Die gewählten Technologien werden darüber hinaus auch darauf geprüft, wie gut die Zahlungsmittel in der jeweiligen virtuellen Geldbörse (Wallet) vor Diebstahl geschützt sind. Dies beinhaltet auch zum einen, mit welchen Verschlüsselungsalgorithmen gearbeitet wird und zum anderen wie sicher sie sind. Weiterhin soll ergründet werden, welche Ausmaße fehlerhafte Nutzung der Technologie auf die Sicherheit hat.

5.1 Bitcoin

Bitcoin setzt bei seiner Implementierung auf die Blockchain-Technologie, diese enthält intrinsische Sicherheitsrisiken, die bei der Umsetzung dieser Kryptowährung adressiert werden mussten. Im folgenden werden diese Risiken und das entsprechende Sicherheitsmerkmal auf Zukunftssicherheit geprüft.

5.1.1 Konto Sicherheit

Bitcoins gehören zu einer digitalen Geldbörse. Diese Geldbörse ist vereinfacht betrachtet ein digitaler Schlüssel zu dem nur der Eigentümer Zugang

haben sollte. Es ist nur mit diesem Schlüssel möglich, zu beweisen welche und wie viele Bitcoins einer bestimmten Person zugeordnet sind. Die Generierung der Schlüssel basiert auf ein ECC (Elliptic Curve Cryptography) Verschlüsselungsverfahren (siehe zu ECC 4.1.2 auf Seite 16).

Aus diesem privatem Schlüssel kann der öffentliche Schlüssel nachvollzogen werden, die eine Adresse darstellt, an die Bitcoins gesendet oder, von ihr ausgehend, Bitcoins versendet werden. Der öffentliche Schlüssel wird nochmals mit SHA256 und dann mit RIPEMD-160 gehasht. Die resultierende Zeichenfolge nennt sich eine P2PKH-Adresse (Pay To Public Key Hash). Sie besteht aus 27-34 alphanumerischer Zeichen mit Ausnahme von einigen Buchstaben/Zahlen für eine bessere Lesbarkeit. Dies geschieht als Absicherung für den Fall, dass es durch einen neuentdeckten Algorithmus möglich sein sollte, das ECDSA-Verschlüsselungsverfahren zu invertieren und den privaten Schlüssel aus dem öffentlichen Schlüssel zu generieren.

Ein privater Schlüssel bei Bitcoin besteht aus 256 Bits. Daraus erschließen sich $2^{256} \approx 1,16 \times 10^{77}$ Kombinationsmöglichkeiten für einen privaten Schlüssel. Die Wahrscheinlichkeit, dass eine andere Instanz denselben Schlüssel generiert, stellt somit kein Sicherheitsrisiko dar.

5.1.2 Anonymität

Durch die vollständige, öffentlich zugängliche Historie *aller* Bitcointransaktionen ist das Netzwerk transparent. Es ist möglich Bitcoins zurückzuverfolgen, um nachvollziehen zu können, durch welche Adressen diese gelaufen sind.

Aufgrund dessen liegt der Aspekt der Anonymität nicht in der Verhüllung der Transaktionen, sondern in der Verschleierung der Identität. Private und öffentliche Schlüssel-Paare sind von der Identität des Nutzers entkoppelt, was wiederum bedeutet, dass der Schutz der Identität bei der Verantwortung des Nutzers liegt. Der Nutzer kann durch neue Schlüsselpaare versuchen seine Person weiter zu verschleiern und eine Zurückverfolgung zu erschweren. Aber auch ein sparsames Herausgeben von Informationen sollte angeführt werden

als ein „Best-Practise“-Umgang mit Kryptowährungen allgemein.

5.1.3 Validität einer Transaktion

Eine Transaktion wird von allen Nodes verifiziert an die die Transaktion geschickt wird. Diese überprüfen die Transaktion auf diverse Kriterien wie unter anderem auf Syntax und ob die Inputs mit den Outputs übereinstimmen. Nachdem die Transaktion als valide eingestuft wurde, wird sie in die Liste hinzugefügt, die sich in dem - zu minenden - Block befindet.

Wenn der passende Hash zu dem Block gefunden wurde, wird dieser Block von der Node in das Netzwerk ausgesendet. Andere Nodes verifizieren diesen Block auf Korrektheit der Syntax und der enthaltenen Transaktionen. Nur wenn diese korrekt sind, dann wird an der Erzeugung des nächsten Blocks auf Basis des vorherigen Blocks gearbeitet.

Es ist also für einen Angreifer nicht möglich ungültige Transaktionen in das Netzwerk zu veröffentlichen, da sie nicht von ehrlichen Nodes angenommen werden würden. Dasselbe gilt auch für ungültige Blöcke. Das wird garantiert durch die vollständige Historie aller Bitcoins seit dem ersten Block, die jede Full-Node gespeichert hat. Es wäre realistisch nicht wahrscheinlich, dass ein Angreifer langfristig eine längere Kette als die Hauptblockchain erzeugen kann, wenn er nicht 50% der Gesamtleistung des Netzwerkes besitzt (mehr hierzu bei 5.1.4 auf Seite 34). Das ist nämlich nötig um als korrekte Kette angesehen zu werden, es gilt:
Die Kette, die am meisten Rechenleistung bürgt (längste Kette) ist die Hauptkette.

Jedoch sollte angemerkt werden, dass durch die Wahrscheinlichkeit, dass ein Angreifer kurzfristig mehr Blöcke erzeugen könnte als die wahre Hauptkette, eine Transaktion erst nach ca. sechs weiteren Blöcken als bestätigt angesehen wird.(QUELLE)

5.1.4 Angriffsszenarien

Im Folgenden werden diverse Angriffsszenarien auf das Bitcoin-Netzwerk beschrieben. Es wird erläutert mit welchen Mechanismen sich die Blockchain-Technologie vor diesen Angriffen schützt.

51%-Attacke

In Satoshi Nakamotos Bitcoin-Whitepaper wird erklärt, wie ein möglicher Angriff durchzuführen wäre. Er beschreibt ein Szenario, welches als 51%-Attacke bekannt ist. Grundlegend hiermit gemeint, das ein Angreifer mehr als 50% der Hash-Rate (Gesamteistung) des Netzwerkes besitzt und diese Leistung nutzt um schneller eigene Blöcke in das Netzwerk zu publizieren.

Dies wird nützlich, um Transaktionen die man getätigt hat ungültig zu machen indem man eine weitere Transaktion von der Adresse an eine andere, eigene Adresse erstellt. Zum Beispiel kann damit ein Verkäufer getäuscht werden, sodass die Person denkt, man hätte die Bitcoins gezahlt. Im Hintergrund jedoch erstellt der Angreifer seine eigene Blockchain mit der, zu diesem Zweck, erstellten Transaktion an die eigene Adresse, womit zugleich die anfängliche Transaktion nicht mehr in einen Block aufgenommen werden kann, da die Bitcoins nicht mehr an der hinterlegten Adresse sind. Es müssen solange Blocks erzeugt werden, bis die falsche Blockchain länger ist als die ehrliche.

Bei einem erfolgreichen Angriff würde die eigene Wallet nicht mit der Zahlung an den Verkäufer belastet werden. Ein weiterer Nebeneffekt ist, dass womöglich viele Transaktionen in der Hauptkette nicht mehr validiert sind und erneut aufgenommen werden müssen.

Der Angriff hat jedoch Grenzen. Es können keine ungültigen Blöcke vom Angreifer erstellt werden, da diese von anderen ehrlichen Nodes nicht angenommen werden würden. Der Konsens liegt weiterhin bei der Hauptkette.

Dieses Szenario ist auch denkbar bei einer Hash-Rate von unter 50%, wird jedoch immer unwahrscheinlicher je mehr Blöcke der Angreifer zurückliegt.

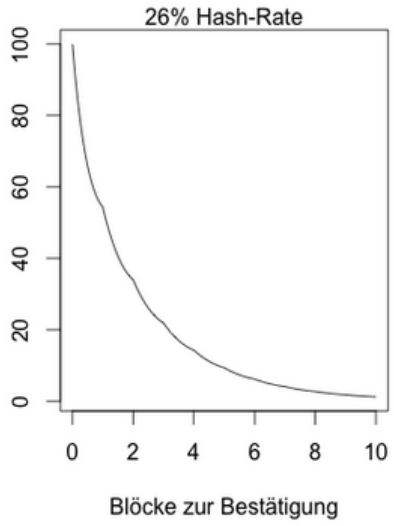

Abbildung 5.1: Veranschaulichung zeigt wie unwahrscheinlich es wird, je weiter man zurückliegt (Quelle https://www.btc-echo.de/tutorial/bitcoin-51-attacke/)

Langfristig kann die Hauptkette nur ersetzt werden, wenn über 50% der Hash-Rate dem Angreifer zur Verfügung stehen.

Satoshi Nakamoto führt folgende Formel zur Berechnung der Wahrscheinlichkeit an, dass ein Angreifer die Blockchain einholen kann aus z Blöcken im Rückstand. Der zeitliche Rahmen ist unbegrenzt.

p = Wahrscheinlichkeit, dass ein ehrlicher Node den nächsten Block erstellt
q = Wahrscheinlichkeit, dass der Angreifer den nächsten Block erstellt
q_z = Wahrscheinlichkeit, dass der Angreifer die ehrliche Blockchain überhohlt

$$q_z = 1 \qquad wenn\ p \leq q$$
$$q_z = \left(\frac{p}{q}\right)^z \quad wenn\ p > q$$

Es geht hervor, dass die Chancen des Angreifers die Hauptkette zu überhohlen bei unter 50% Hash-Rate exponentiell schwinden.

Diese Angriffsfläche eröffnet sich aufgrund der dezentralen Architektur von Blockchain und ist ein integraler Bestandteil eines, auf Konsens beruhenden,

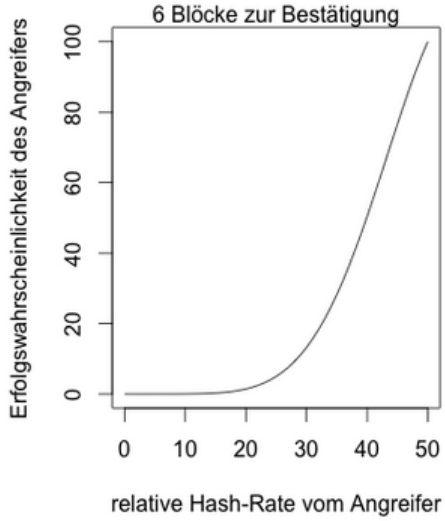

Abbildung 5.2: Zeigt wie hoch die Wahrscheinlichkeit ist, sechs Blöcke hintereinander zu generieren (Quelle https://www.btc-echo.de/tutorial/bitcoin-51-attacke/)

Systems.

5.1.5 Anreiz zur Ehrlichkeit

Durch die Entlohnung in Form von Bitcoins durch das Minen und durch die Transaktionsgebühren wird Anreiz geschaffen, das Bitcoin-Netzwerk nicht zu manipulieren. Eine Kette mit ungültigen Blöcken würde auch nicht von ehrlichen Minern angenommen werden. Daher wird erhofft [Nak08], dass die Belohnung und das Eigeninteresse an der Integrität von Bitcoin ausreicht um Angreifer abzuwehren.

5.1.6 Quantenprozessorresistenz

Mit der Einführung leistungsstarker Quantenprozessoren, die darauf ausgelegt sind, kryptographische Algorithmen zu entschlüsseln, könnten Algorithmen (unter anderem ECDSA) unsicher werden. Auch die starke Leistung von Quantencomputern eröffnet eine neue Angriffsfläche unter dem Aspekt, dass der Anteil von über der Hälfte der Gesamtleistung des Miner-Netzwerkes

dem Nutzer langfristig eine längere Blockchain gewähren kann. Auf dieses Problem wird in Kapitel 5.1.4 eingegangen.

Die genutzten kryptographischen Algorithmen sind nicht bewiesen, dass sie Quantenprozessoren standhalten könnten(QUELLE). Indes ist das Bitcoin-Netzwerk durch ihre Architektur und Nutzung nicht gegen Quantencomputern geschützt. Aber komplett ungeschützt bleibt die Technologie nicht.

Auch wenn der ECDSA mit einem entsprechenden Algorithmus gelöst werden sollte, oder durch Quantenprozessoren in anderer Hinsicht unsicher werden sollte, wird durch das hashen des öffentlichen Schlüssels genau dieser verschleiert. Er kommt nur zum Vorschein, wenn von dieser Adresse eine Zahlung ausgeht, aufgrund der Signatur. Wenn nun von da an ein neues Schlüsselpaar generiert und genutzt wird, sollten die Bitcoins an der neuen Adresse sicher sein.

5.2 IOTA

IOTA nutzt im Gegensatz zu Bitcoin keine Blockchain-Technologie, sondern setzt auf ein sogenanntes Tangle, welches wiederum auf einem mathematischem Konzept basiert, welches sich Directed-Acyclic-Graph (DAG) nennt (siehe zu Tangle 4.2.2 auf Seite 22). Dieses Konzept unterscheidet sich grundsätzlich von Blockchain und weist unter diesem Aspekt andere Sicherheitsrisiken auf. Im folgenden sollen diese beschrieben werden.

5.2.1 Konto Sicherheit

Auch bei IOTA besteht das Konto aus einem digitalen Schlüssel, dem Seed. Dieser Seed sollte nur dem Eigentümer verfügbar gemacht werden, da jede Person mit diesem Seed, über die zugeordneten IOTAs verfügen kann. Für einen optimalen Schutz vor digitalem Diebstahl sollten Seeds nur lokal (ohne Internetzugang) erstellt und gespeichert werden. Wenn diese allgemeinen Sicherheitsvorschläge eingehalten werden, besteht hier noch kein nennenswertes Risiko.

Ein Seed besteht aus 81 Trytes (siehe zu Trytes Grundlagen 2.8 auf Seite 8). Ein Tryte enthält drei Trits und kann folglich 27 ganze Zahlen darstellen. Daraus ergeben sich $27^{81} \approx 8,72 \times 10^{115}$ verschiedene Kombinationsmöglichkeiten. Das heißt, die Wahrscheinlichkeit, dass eine andere Instanz denselben Seed generiert wie der Eigentümer ist $1 : 8,72 \times 10^{115}$, damit ist die Anzahl der Kombinationsmöglichkeiten deutlich höher als die Anzahl der Atome im bekannten Universum.(FOOTNOTE)

Durch das One-Time-Winternitz hashing (siehe zu One-Time-Winternitz Funktion 4.2.8 auf Seite 28) ist eine Adresse für ausgehende Zahlungen nur einmal zu benutzen, da ein Teil des privaten Schlüssels bekannt wird. Sollte die Adresse mehr als einmal genutzt werden, so wird zu viel des privaten Schlüssels öffentlich gemacht und eine Entschlüsselung wird zunehmend einfacher. Ist der private Schlüssel vollständig enthüllt, kann der Angreifer auf die IOTAs zugreifen die auf der Adresse hinterlegt sind.

Hierdurch eröffnet sich ein Risiko, welches der Nutzer von IOTA selbst entgegnen muss, wenn er nicht das offizielle IOTA-Wallet Programm nutzt um Transaktionen zu erstellen. Hierzu muss angemerkt werden, dass durch die Kompromittierung des privaten Schlüssels nicht der Seed oder alle anderen Adressen gefährdet sind.

5.2.2 Anonymität

Es ist wichtig zu betrachten, wie Anonym die Identität im Tangle-Netzwerk ist. Zum einen was einige Nutzer über andere Nutzer wissen können und zum anderen was in einer Transaktion über die Identität preisgegeben wird.

Vorerst muss notiert werden, dass Anonymität keine Priorität in der Entwicklung von IOTA gewesen ist und auch nicht primär für diesen Zweck entwickelt worden ist. (QUELLE)

Um eine Transaktion zu veranlassen wird ein Paar aus privatem und öffentlichem Schlüssel benötigt, welche aus dem Seed und einem Adressindex generiert werden. Der öffentliche Schlüssel dient hierbei als Adresse die man an seinen Transaktionspartner senden kann. Der private Schlüssel sollte verbor-

gen bleiben, denn jener Schlüssel wird genutzt um die Inputs eines Bundles zu signieren. Damit wird bewiesen, dass der Ersteller der Transaktion den privaten Schlüssel zu der Adresse besitzt und folglich der Eigentümer ist.

- Erklären was das mit anonymität zu tun hat

5.2.3 Validität einer Transaktion

Eine Transaktion wird von anderen Nodes validiert. Typischerweise geschieht das wenn ein Nutzer eine Transaktion in das Tangle publizieren will. Dafür muss der Nutzer zwei Transaktionen validieren. Bei diesen Transaktionen handelt es sich um „Tips“, die noch nicht referenziert sind. Das Transaktionsobjekt muss die Tips als „branch“- und „trunkTransaction“ referenzieren und überprüfen.

Das Objekt wird auf Syntax und Gültigkeit geprüft. Dafür wird der Transaction-Hash abgeglichen und die Signierung überprüft.

- wie wird eine einzelne transaktion verifiziert, wie wird bestimmt dass sie gültig ist
- wann ist eine transaktion verifiziert

5.2.4 Angriffsszenarien

Die Tangle ermöglicht durch ihre DAG spezifische Arten von Attacken die im weiteren Fortgang erläutert werden sollen. Durch die Darstellung der möglichen Angriffe, werden weitere Sicherheitsmechanismen veranschaulicht.

34%-Attacke

Ein böswilliger Nutzer könnte mit mehr als 34% der Gesamtrechenleistung im Tangle-Netzwerk einen Angriff starten indem er falsche Transaktionen mit vielen anderen selbst initiierten Transaktionen verifiziert und damit im DAG einen Strang erzeugt der falsche Informationen über die verfügbaren IOTAs enthält.

Dem entgegenwirken soll der, von der IOTA-Foundation bereitgestellte, Coordinator. Dieser ist ein Node, der eigenständig Transaktionen erstellt, die als Meilenstein (Milestone) gelten. Der Milestone dient als Beweis, dass die vorherigen Transaktionen korrekt sind. Jede Transaktion muss diese Coordinators indirekt oder direkt referenzieren.(QUELLE)

Diese Coordinators bieten jedoch eine neue Angriffsfläche. Dabei geht es um das derzeit nicht vollkommen dezentrale System der Tangle bei IOTA. IOTA löst also noch nicht das Problem, ohne dritte Instanz Transaktionen vermitteln zu können. Laut IOTA-Foundation sollen die Coordinators ihren Dienst einstellen, sobald das Tangle Netzwerk ausreichend groß ist, um 34%-Attacken durch eine zu hohe Anforderung an Rechenleistung abwehren zu können.(QUELLE)

Parasiten-Kette

- wie schützt sich das Bitcoin netzwerk vor Angriffen
- welche angriffsszenarios sind denkbar
- was kann dagegen getan werden

5.2.5 Quantenprozessorresistenz

Bei der Frage nach Zukunftssicherheit bei IOTA geht es auch um die Resistenz vor Quantenprozessoren. Genauer soll betrachtet werden, ob sich die Fertigstellung von diesen leistungsstarken Prozessoren auf die Sicherheit und Beständigkeit von IOTAs auswirkt.

Hierbei ist es nötig, auf den Winternitz onetime hashing algorithmus zurückzugreifen um zu verstehen wie sich der Algorithmus auf Quantenresistenz auswirkt.

Kapitel 6

Programmierbarkeit

In diesem Abschnitt folgt eine Beschreibung der Programmierschnittstellen bei Bitcoin und IOTA. Dies ist relevant da gegebene Schnittstellen und Ressourcen für Programmierer die Einsetzbarkeit erhöhen. Wenn Entwickler auf den vorhandenen Code aufbauen und erweitern können, können die Eine aussage über die zukünftigen einsatzmöglichkeiten

6.1 Bitcoin

Der Code für Bitcoin ist Open-Source und damit frei zugänglich Ja was gibts bei bitcoin so keine ahnung

6.2 IOTA

Iota erlaubt 0 vlaue transactions für meta daten zb json datein (nutzen hiervon für alltag umstritten wegen stormverbrauch), das erlaubt also quasi vershcicken von daten in der tangle Dadurch erschließt sich ein weiterer Anwendungsbereich. Diese Funktion erlaubt eine nützliche Anwendung für das „Internet der Dinge“ (internet of things **??** auf seite **??**

Kapitel 7

Nutzen

7.1 Bitcoin/Blockchain

Bitcoin wurde als eine kryptographische Währung implementiert, die erstmalig das anonyme, dezentrale Bezahlen ermöglicht. Der Nutzen wird dadurch deutlich, dass die Bezahlvorgänge unabhängig von Gesetzen oder Richtlinien durchgesetzt werden können. Einer Transaktion im Bitcoin Netzwerk ist nicht ersichtlich, welche Ware eingekauft wurde und welche Person für die Bezahlung zuständig ist. Das Netzwerk ist gänzlich unabhängig von der Kontrolle einer zentralen Partei. Jeder Bezahlvorgang ist in der Blockchain dokumentiert und für jedermann einsehbar.

Diese Eigenschaften sorgen dafür, dass das Vertrauen nicht mehr auf einer zentralen Vertrauenspartei basiert, sondern das Netzwerk dafür einsteht. Jede je getätigte Transaktion kann zu einem beliebigen Zeitpunkt in der Zukunft rückwirkend verifiziert werden, ohne dass Rechte Dritter verletzt werden.

Die Blockchain hinter Bitcoin setzt den Grundstein für einen sicheren Umgang mit finanziellen und kommerziellen Gütern. So könnte die Echtheit von sensiblen Dokumenten durch deren digitalen Fingerabdruck unwiderruflich in der Blockchain abgespeichert werden. Ein Mietvertrag, dessen Mietkosten monatlich zu begleichen sind, könnten in Form eines „Smart Contracts“ durch

die Blockchain auf Richtigkeit überprüft und ausgeführt werden. [CNP+16]

7.2 IOTA

IOTA wurde in erster Linie für das „Internet of Things“ entworfen und soll Transaktionen und Kommunikationen sowohl zwischen Menschen als auch zwischen Gerätschaften ermöglichen. Die Vision der IOTA-Foundation beschreibt die Tangle und das dazugehörige Distributed Ledger Protokoll, als eine einheitliche Übereinstimmung der Richtigkeit von Daten. Dieses Prinzip wird über kryptographische Hash Verfahren erreicht. Des Weiteren möchte IOTA eine Lösung für den wachsenden globalen Datentransfer liefern, indem das Server - Client Modell durch eine dezentrale Lösung ersetzt wird. Im Folgenden wird die Kommunikation in der Tangle durch die Verwendung von „Masked Authenticated Message“ (kurz MAM) im Detail beleuchtet.

7.2.1 MAM

MAM steht für „Masked Authenticated Message“ und ist die Lösung für das Kommunizieren innerhalb der Tangle. Es ist möglich durch eine wertlose Transaktion eine Nachricht zu verschicken. Sollen jedoch mehrere Nachrichten versendet werden, bspw. soll die Temperatur eines Motors alle 15 Minuten aktualisiert werden, treten einige Probleme auf. Der Versender kann die Nachricht zwar immer an die gleiche Adresse versenden, aber jeder andere IOTA Benutzer auch. Bei einem möglichen böswilligen Angriff könnten die validen Nachrichten nicht von den böswilligen unterschieden werden.

MAM basiert auf Kanälen die ein interessierter Hörer „abonnieren“ kann. Es werden Transaktionen generiert die jeweils den Index des Branches, die Geschwister - Knoten des Merkle Trees, die Adresse der nächsten Generation und die Nachricht als Hash enthalten. MAM’s verweisen immer auf die nächste Generation, aber nie auf die vergangene Generation oder den Genesis.

Verschlüsselung Abhängig davon in welchem Modus die Daten publiziert sind, benötigt der Hörer entweder ausschließlich den „root“ oder zusätzlich einen “sideKey” um die Daten zu entschlüsseln. In dem Modus „public“ kann der Inhalt durch den „root“ entschlüsselt werden. Sollte der Modus „restricted“ sein, ist es zwar möglich die MAM mit Hilfe des „root“ in der Tangle zu finden, um die Daten jedoch zu entschlüsseln wird ein „sideKey“ benötigt. Derjenige, der die Daten publiziert, kann dementsprechend entscheiden, wer die Daten einsehen kann.

Signatur Ein böswilliger Hörer, der von einem Kanal den „root“ und den „sideKey“ erhält, darf nicht die Möglichkeit haben, dank dieser beiden Daten, den Kanal für sich zu beanspruchen um zukünftig Beiträge in diesem zu verfassen. Der Kanal benötigt eine Signatur. Diese wird nach dem „Merkle tree based signature scheme“ erreicht. Eine Generation ist aufgebaut wie ein Merkle Tree. Die Blätter sind private Schlüssel. Diese Schlüssel werden verwendet um IOTA-Adressen zu generieren. Adressen werden jeweils paarweise miteinander gehasht. Das paarweise Hashen der Ergebnisse wird solange fortgeführt, bis der root Hash erreicht ist. Es können beliebig viele Generationen erstellt werden. Der Hörer nutzt die Adresse der MAM und hasht diesen mit dem ersten, in der MAM angegebenen Geschwister Hash. Das Ergebnis wird mit dem nächsten Geschwister Hash zusammengeführt. Dieser Vorgang wiederholt sich, bis das letzte Paar zu dem sogenannten „temp_root“ zusammengeführt wurde. Sollte der „temp_root“ identisch mit dem root der MAM sein, ist die Nachricht authentisch.[1](PICTURE)

[1]Daten bezogen von: https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413

Kapitel 8

Ergebnis

Kapitel 9

Zusammenfassung

Teil II

Praktischer Teil

Kapitel 10

Tanken in der Zukunft

10.1 Projektbeschreibung

loremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum lorem ipsumloremipsum loremipsum

Wie sieht die kommunikation mit dem server aus, wie wird es realisisert und für welche Zwecke und wieso rest api über itnernet, und nicht zum beispiel NFC oder sonstwas

10.2 Projektbegründung

Um eine bessere Aussage über die Zukunftstauglichkeit der Blockchain zu gewinnen, wird gemeinsam mit der IOTA Technologie ein Anwendungsfall prototypisch implementiert. Bei diesem Anwendungsfall handelt es sich um eine Tankstelle an der ein Auto möglichst autonom den Tankvorgang startet und im Anschluss den anfallenden Betrag mit der Kryptowährung IOTA

begleicht. Die kommende Ausführung befasst sich mit der Implementation eines Prototypen.

10.3 IOTA als Währung

Warum haben wir uns für IOTA entschieden, welche alternativen gibt es, wieso waren die nicht so gut wie IOTA...ä

Kapitel 11

Fahrzeug - Client

11.1 Ziel und Anforderungen

Das Programm soll für einen realitätsnahen Anwendungsfall auf einem relativ leistungsschwachen System arbeiten können. Daher liegt es nahe für die Programmierung des Programmierteils, der für das Tanken und Bezahlen mit IOTAs zuständig ist, einen Raspberry Pi anzusteuern.

Der Raspberry Pi soll in dieser prototypischen Umsetzung die Steuereinheit eines zukünftigen Autos ersetzen. Es soll mittels zwei Knöpfen bedienbar sein, die die Steuerelemente in der Armatur des Autos darstellen sollen.

Da der Anspruch an starke Rechenleistung in einem Auto nicht zu hoch sein sollte (um zum Beispiel Kosten zu minimieren) ist es sinnvoll, keinen IOTA Full-Node zu betrieben, sondern nur einen Light-Node, der die Arbeit an einen Full-Node delegiert. Die Limitation der Hardware-Ressourcen ist ein Grund für die Umsetzung auf einem Raspberry Pi.

(PICTURE)Hier noch beschreiben was die Anforderungen an das Smartauto sind, und graphische nutzerinteraktion anzeigen. Hier soll erklärt werden, was von der tankanwendung gefordert ist, und wie sich das auf die tatsächlich umsetzung auswirkt.

11.2 Aufbau

Unter diesem Aspekt wird der Aufbau und die Einrichtung der einzelnen Komponenten beschrieben die für die Umsetzung nötig sind.

11.2.1 Vorbedingungen

Unter diesem Aspekt werden die Voraussetzungen für die Umsetzung der clientseitigen Tankanwendung beschrieben. Diese Vorbedingungen wurden genutzt um den Raspberry Pi so einzurichten und zu konfigurieren, damit das Gerät die Anforderungen an das Programm erfüllen kann.

Um die prototypische Anwendung umsetzen zu können, werden folgende Komponenten genutzt:

Raspberry Pi 3 Model B

Auf dem Raspberry Pi wird die Anwendung ausgeführt, Daten des Benutzers verarbeitet und Anfragen auf Webservices vollzogen. Auch die Transaktionen von IOTAs sollen mithilfe des CPUs erstellt werden.

Steckbrett

Das Steckbrett dient der Verbindung von Bedienelementen mit den GPIO-Pins des Raspberry Pis. Die Knöpfe für die Bedienung der Tank-Anwendung werden auf diesem befestigt und mit den GPIO-Pins elektronisch verbunden.

Micro SD Karte

Diese Speicherkarte wird für den Raspberry Pi gebraucht. Auf dieser befindet sich das Betriebssystem mit dem der Raspberry Pi erst genutzt werden kann. Auch das Programm und sonstige Benutzerdaten werden auf dieser Speicherkarte gespeichert.

Micro USB Netzteil 5,1 Volt 3,1 Ampere

Der Raspberry Pi muss mittels eines Micro USB Eingangs mit Strom versorgt werden. Das Netzteil sollte eine Ausgangsspannung von mindestens 5,1 Volt aufweisen und es werden mindestens 2,5 Ampere empfohlen.[25](QUELLE)

5x Weiblich-Männlich Kabel, 2x Männlich-Männlich Kabel

Verbindungskabel für das Steckbrett um elektronische Komponenten miteinander zu Verbinden.

2x Taster, 1x Rote LED

Elektronische Komponenten um dem Benutzer der Anwendung Bedienelemente zur Verfügung zu stellen. Die LED dient für ein visuelles Feedback.

Widerstände 1x 330 Ohm, 2x 10.000 Ohm, 2x 1.000 Ohm

Die Widerstände sind für den kontrollierten Stromfluss unentbehrlich und sie verhindern, dass die Komponenten durch zu hohen Stromfluss beschädigt werden.

11.2.2 Einrichtung

Um den Raspberry Pi so einzurichten, dass es möglich ist, Anwendungen zu programmieren und zu testen, sollte dieser mittels eines HDMI-Kabels an einen Monitor angeschlossen werden. Darüber hinaus sollten Peripheriegeräte wie Tastatur und Maus über die USB-Ausgänge angeschlossen werden um sich durch die Bedienoberfläche navigieren zu können.

Damit das Gerät jedoch vollständig bedienbar wird, muss es mit einem Betriebssystem ausgestattet sein. Aufgrund von offiziellem Support seitens der Raspberry Pi Foundation [26](Quelle) wurde das Raspbian Betriebssystem für die Umsetzung ausgewählt. Bei Raspbian handelt es sich um ein, auf Linux basierendes Betriebssystem, welches von der Raspberry Pi Foundation veröffentlicht wurde.

Das Raspbian Betriebssystem muss auf die Micro SD Karte geladen werden, damit es für den Raspberry Pi zur Verfügung steht. In folgenden Schritten wurde die Speicherkarte mit Raspbian beschrieben:

1. Download von Raspbian (https://www.raspberrypi.org/downloads/)
2. Gegebenenfalls .zip/.rar entpacken
3. Speicherkarte in ein Kartenlesegerät stecken
4. Mithilfe eines Brennprogramms die Imagedatei auf die Speicherkarte brennen (flashen)

Die Speicherkarte ist nun formatiert und mit Raspbian ausgestattet, sie kann nun aus dem Kartenlesegerät genommen werden und in den Raspberry Pi gesteckt werden.

Nachdem das Gerät zum ersten Mal gestartet wird, muss das Betriebssystem einmalig eingerichtet werden. Wenn dies geschehen ist, sollte man jegliche Software-Pakete in Raspbian updaten um mögliche Fehler vorzubeugen. Dies kann mit folgenden Befehlen in dem Terminal des Betriebssystems vollzogen werden:

```
> sudo apt-get update
> sudo apt-get dist-upgrade
```

Steckbrett und GPIO-Pins

Die Taster und LEDs müssen elektrisch mit den GPIO-Pins verbunden werden, um von dem Raspberry Pi angesprochen werden zu können. Die Taster stellen Knöpfe auf der Armatur des Autos dar. Die LED dient dafür, ein visuelles Feedback zu geben, zum Beispiel im Bezug auf die Tank-Anwendung ob nun getankt werden kann.

Dafür ist es nützlich, mit dem Pin-Layout des Raspberry Pis vertraut zu sein. Mit dem Terminal-Befehl

```
> pinout
```

ist es möglich einen Überblick über das Layout des Raspberry Pis zu erhalten. Auch die Benennung der einzelnen Pins ist aufgeführt.

[Bild der „pinout“-Ausgabe einfügen](PICTURE)

Die Komponenten müssen in das Steckbrett gesteckt werden, sodass sie erreichbar bleiben, um sie zu verkabeln und zugleich nutzen zu können. Für eine optimale Übersicht wurden die Komponenten mittig plaziert.

[Bild von der Plazierung der Komponenten Button 1 und 2 und LED](PICTURE)

Für die Umsetzung müssen diese Komponenten für den Nutzer interagierbar sein, dazu werden sie mit zufälligen, freien GPIO-Pins verbunden.

[Tabelle hier einfügen, wie buttons 1 und 2 und LED mit gpios verbunden sind](PICTURE)

Dies erfolgt indem die weibliche Seite des Kabels in den jeweiligen Pin geführt wird, die andere Seite des Weiblich-Männlich Kabels wird nun in die Reihe des Steckbretts gesteckt an der sich auch die Komponente befindet, die mit dem GPIO-Pin interagieren soll.

Um auf dem Steckbrett einen Stromkreis erzeugen zu können, sollten an den Querreihen des Steckbretts Spannungen angelegt werden. Dafür gibt es auf dem Raspberry Pi Pins, die entweder eine Spannung von 5V, 3,3V oder 0V (GND) haben. Für diese Umsetzung werden die 3.3V Pins und GND-Pins genutzt um einen Stromkreis zu erzeugen, da 3,3V ausreichen um ein Signal auf den Pins zu erfassen. Diese werden jeweils auf die korrespondierende Querreihe mittels Weiblich-Männlich Kabeln verbunden.

Zum Schutz vor Überspannung an der LED muss ein Widerstand in den Stromkreis eingebaut werden, der verhindert, dass die LED zu viel Energie erhält und sie dadurch zu heiß wird und durchbrennt.

Für die Berechnung des passenden Widerstandes empfiehlt es sich, mit der

Formel

[Formel für berechnung des widerstandes](PICTURE)

zu arbeiten. Wobei Im = Eingangspannung also 3.3V und Ia = Spannungsverringerung der LED also 0.2A ist. Dies ergibt

[Formel mit eingesetzten parametern](PICTURE)

Hierbei handelt es sich um einen Wert der eine geringe Abweichung erlaubt. Aus diesem Grund wird der nächsthöhere verfügbare Wert angenommen. In diesem Fall ist dies 330 Ohm. Dieser Widerstand muss nun in den Stromkreis mit der LED eingebunden werden. Nach Einfügen des Widerstandes ist der Stromkreis geschlossen. Dies wiederholt man auch für die Taster.

PyOTA

Bei PyOTA handelt es sich um eine Python-Bibliothek für IOTA. Sie implementiert die API und unterstützt Funktionalitäten um eine Transaktion eigenständig erstellen zu können[1].

Um die Bibliothek für die Umsetzung zur Verfügung zu stellen kann sie über das Paketverwaltungsprogramm für die Umgebung installiert werden.

```
> pip install pyota
```

Eine wichtige Abhängigkeit (Dependancy) von PyOTA ist die Cryptography-Bibliothek. Diese wird nicht automatisch heruntergeladen und installiert, sondern dies muss manuell geschehen[2].

Weiterhin muss angemerkt werden, das bei Raspbian kein System-Pfad zu den zusätzlich installierten Paketen (per pip install) für Python2.7 gibt. Demzufolge muss der Pfad vor jedem Programmstart in das Suchverzeichnis hinzugefügt werden. Dies geschieht mit folgendem Code:

```
import sys
```

[1] siehe dazu https://pyota.readthedocs.io/en/latest/
[2] mehr Informationen dazu https://cryptography.io/en/latest/installation/

```
sys.path.append("/home/pi/.local/lib/python2.7/site-packages")
    #path muss erweitert werden damit PyOta librarys gefunden
    werden können
```

11.2.3 Ergebnis der Vorbereitung

Nachdem alle Vorbedingungen erfüllt sind, ist der Raspberry Pi einsatzbereit um ein Programm zu erstellen und auszuführen, welches auf Benutzereingaben an Knöpfen horcht, visuelles Feedback gibt und mit der PyOta-Bibliothek arbeitet.

11.3 Umsetzung

In diesem Abschnitt sollen die ausgewählten Programmieransätze gezeigt und besprochen werden. Es soll ein Überblick über die verwendeten Methoden geschaffen werden. Bei mehreren möglichen Methoden soll darüber hinaus deutlich werden, warum die entsprechende Methode gewählt wurde.

11.3.1 GPIO-Pins

Auf dem Raspberry Pi müssen die GPIO-Pins vorkonfiguriert werden um danach einzelne Pins als Inputs oder Outputs setzen zu können. Die Funktionalität wird der vorinstallierten RPi.GPIO-Bibliothek entnommen. Sie wird zur Abkürzung fortan mit „GPIO“ referenziert.

```
import RPi.GPIO as GPIO  #importiere GPIO-Bibliothek für
    Ein-/Ausgaben von Inputs und Outputs (Buttons/LEDs)
```

Zum Ansprechen der Pins muss bestimmt werden, welche Pins mit welcher Nummer korrespondieren. Zum Beispiel hat Pin 12 eine Zweideutigkeit in dem Sinne, dass er zum einen Pin 12 bei einer, numerisch sortierten Aneinanderreihung, der zwölfte Pin des Panels ist. Zum anderen kann sich Pin 12 auch auf den Namen des Pins beziehen, in diesem Beispiel „GPIO12“,

welcher sich nicht an einer numerischen Anordnung befindet.

Daher muss der Modus des Boards gesetzt werden mit folgender Code Zeile.

```
GPIO.setmode(GPIO.BCM)
```

GPIO.BCM bezieht sich auf den Broadcom SOC (system on chip) Kanal, der angesprochen wird. Die Auswahl von dieser Variante hat keine Vor- oder Nachteile für das Projekt und wurde arbiträr ausgewählt.

Nachdem der Modus eingestellt wurde kann nun die Datenrichtung der Pins bestimmt werden. Dafür wird der Pin als Input oder als Output gesetzt.

```
BUTTON1 = 11
BUTTON2 = 22
LED = 10

GPIO.setup(BUTTON1, GPIO.IN)
GPIO.setup(BUTTON2, GPIO.IN)
GPIO.setup(LED, GPIO.OUT, initial = False)
```

Nun sind die einzelnen Pins manipulierbar, bzw. auslesbar. Je nachdem ob man einen Output-Pin setzten möchte oder einen Input-Pin auslesen möchte, sind verschiedene Funktionen verwendbar.

Zum Auslesen ob ein Knopf gedrückt wurde bietet sich die folgende einfache Implementierung an. Das Programm wartet solange, bis der Knopf gedrückt und losgelassen wurde.

```
def wait_for_button_click(button):
    while not GPIO.input(button):
        pass
    while GPIO.input(button):
        pass
```

Wobei der Parameter 'button' die Pin-Nummer referenziert.
Es bieten sich auch andere, effizientere Methoden an, jedoch ist diese Imple-

mentierung für den Prototypen ausreichend.

Um einen Output wie zum Beispiel die LED zu setzen, benötigen wir diese Code-Zeile:

```
GPIO.output(LED, True)
```

11.3.2 IOTA Einrichtung

Unter der Einrichtung von IOTA ist zu verstehen, die IOTA-Bibliothek so zu utilisieren, dass mit der API eine Transaktion in das Tangle-Netzwerk veröffentlicht werden kann. Für so eine Funktionalität bietet sich das Iota-Objekt an. Dieses wird dank des iota-Moduls bereitgestellt.

```
from iota import Iota
```

Damit nun das Iota Objekt initialisiert werden kann, braucht es zwei Argumente. Den Seed, und eine Adresse zu einer Full-Node.

Seed Der Seed „DSPOAXMVSC99IUIVJXTIBZFATVFKTCLLJYOLAGSMFJGFXAWEB9GNTC wurde aus einer zufälligen Kombination der Großbuchstaben 'A-Z' und der Nummer 9 eigenhändig generiert und wird für den Prototypen genutzt. Im Programmcode ist der Seed hinterlegt, dies könnte bei einer realistischen Umgebung zu Sicherheitsrisiken führen, für die prototypische Umsetzung genügt es.

Full-Node Aus einer Liste im Internet[3], in der einige Full-Nodes aufgelistet sind, wurde die folgende Adresse beliebig ausgesucht: https://potato.iotasalad.org:14265. Der Full-Node übernimmt die Arbeit die erstellte Transaktion in das Tangle zu veröffentlichen, da der Raspberry Pi zu leistungsschwach ist um den Proof-Of-Work zeiteffizient zu übernehmen.

Aus den gegebenen Parametern kann das Iota-Objekt erstellt werden.

[3] siehe https://iotasalad.org/nodes

```
seed =
    "DSPOAXMVSC99IUIVJXTIBZFATVFKTCLLJYOLAGSMFJGFXAWEB9GNTQWEDVRYHKIOQF9T9IZY9IVPKT
fullnode_url = "https://potato.iotasalad.org:14265"
api = Iota(fullnode_url, seed)
```

Es ist nun möglich, grundlegende Funktionen wie Adressgenerierung nutzen zu können.

11.3.3 IOTA Transaktion

Um eine Transaktion zu erstellen werden für diese Implementierung die Klassen ProposedTransaction, ProposedBundle, Address, Tag und TryteString zusätzlich zu Iota von dem iota-Modul genutzt.

```
from iota import Iota, ProposedTransaction, ProposedBundle,
    Address, Tag, TryteString
```

Vorerst muss ein „Vorabtransaktions“-Objekt erstellt werden mit Adresse, Wert, Tag und Message.

```
proposedTrx = ProposedTransaction(
    address = Address(reciever_address),
    value = 0,
    tag = Tag("BACHELORTEST"),
    message = TryteString.from_string("This is a test transaction")
)
proposedBundle = ProposedBundle([proposedTrx])
```

Das Transaktionsobjekt wird gefüllt mit der Empfängeradresse, dem Wert 0 (eine „Meta Transaction“), einem Tag welcher mithilfe des Tag-Objekts erstellt wird und einer beliebigen Nachricht, welche mithilfe von TryteString in Trits umgewandelt wird. Abschließend wird ein „Vorabbundle“ erstellt, welches mit der Vorabtransaktion befüllt wird.

Dieses Vorabbundle muss geprüft und signiert werden, dafür wird es als Pa-

rameter an die vorher erstellte Iota-Instanz übergeben. Der Aufruf gibt ein vorbereitetes Bundle zurück, welches mittels API-Aufruf an die Full-Node verschickt wird. Dort wird sie abschließend verarbeitet und in das Tangle-Netzwerk veröffentlicht.

```
preparedBundle = api.prepare_transfer(proposedBundle)
publishedBundle = api.send_transfer(depth = 3, transfers =
    proposedBundle)
```

Die Hash des PublishedBundle-Objekts kann genutzt werden, um die Transaktion nachzuvollziehen[4].

11.3.4 Kommunikation mit Tankstelle

Die Kommunikation mit der Tankstelle erfolgt über eine REST API. Dies ist vereinfacht betrachtet eine Schnittstelle auf die über HTTP zugegriffen werden kann. Hierfür bietet sich das vorinstallierte requests-Modul an.

```
import requests #importiere requests-Bibliothek um HTTP-Anfragen
    zu erstellen
```

Ein Beispiel eines Anwendungsfalles ist das Beenden des Tankvorgangs. Der Code hierfür ist in einer Methode ausgelagert mit den Argumenten 'route' und 'carId'. Die 'route' ist die Schnittstelle die angesprochen werden soll, in diesem Beispiel „endFueling“. Die 'carId' ist eine von der Tankstelle zugewiesene Nummer. Das Beispiel verschickt eine Anfrage an den Server (die Tankstelle) mit einem Parameter, der Identifikationsnummer des Autos, und gibt die Antwort zurück.

```
endFuelingData = call_api("endFueling", carId)

def call_api(route, carId):
        params = {
                "id": carId
```

[4] z.B auf https://iotasear.ch/

```
        }
    response = requests.get(fueling_url + route, params)
return response.json()
```

11.4 Ergebnis

Was aus den vorherigen Abschnitten folgt, ist ein Programm, welches mit einem Server über eine Rest-Schnittstelle kommunizieren kann und eigenständig eine Transaktion ausführen kann. Mit dieser Applikation werden die Bezahlvorgänge mit IOTAs abgewickelt.

Um das Ergebnis zu veranschaulichen wird das Programm vorgespielt.

..... Einmal probedurchlaufen mit dem Programm und erklären wie das mit dem vorher erklärten zusammenhängt

Kapitel 12

Tankstelle - Server

Die Kommunikation zwischen der Tankstelle und einem Fahrzeug soll durch eine Restful API geschehen. Das Fahrzeug soll imstande sein, durch das Abfragen der implementierten API, einer vorübergehenden ID zugeordnet zu werden, welche für den gesamten Tankvorgang notwendig ist. Auch der Tank - Prozess soll durch gezielte Abfragen an die API gestartet und beendet werden. In einem abschließenden Schritt soll die Tankstellen API die anfallenden Kosten, sowie die notwendige IOTA - Adresse, übermitteln.

Dem Besitzer der Tankstelle muss es möglich sein durch autorisierte Anfragen an die API die IOTA - Adressen zu wechseln oder diesen Prozess durch das hinterlegen eines IOTA - Seed zu automatisieren. Die Adressen, die für den Bezahlvorgang verwendet werden sollen, werden für den autorisierten Besitzer der Tankstelle frei verfügbar sein.

Es ist wichtig anzumerken, dass diese Art der Kommunikation eine von vielen Möglichkeiten ist, dieses Projekt zu realisieren. Im Endeffekt sind folgende Grundvoraussetzung zu erfüllen:

- Dem Kunden darf es nicht möglich sein den IOTA Seed einzusehen.
- Hinterlegte IOTA - Adressen dürfen für unbeschränkt viele Zahlungseingänge, aber ausschließlich für einen Zahlungsausgang verwendet werden.

- Die Aktualisierung der IOTA - Adresse kann automatisiert werden, dies birgt jedoch das Risiko, einen validen IOTA Seed dem Netzwerk zu hinterlegen.
- Autorisierten Personen muss es gestattet sein, neue IOTA - Adressen der Datenbank hinzuzufügen.
- Das System muss Zugang zu dem IOTA - Tangle besitzen und imstande sein dieser Transaktionen hinzufügen zu können.

12.1 Projektaufbau

In dem folgenden Unterkapitel wird der grobe Projektaufbau skizziert sowie die Installation der notwendigen Software aufgeführt. Diese Schritte sind notwendig um den Prototypen den Anforderungen entsprechend ausführen zu lassen.

12.1.1 Vorbedingungen

NodeJS

NodeJS ist eine asynchrone, auf “Events” basierende, JavaScript Laufzeitumgebung, die es ermöglicht mehrere Anfragen zu derselben Zeit abzuarbeiten. Dadurch entstehen niedrige Latenzen zwischen dem Client und dem Server. Des Weiteren verfügt NodeJS über eine “EventLoop” die der in Web - Browsern bekannten “EventLoops” sehr ähnelt. Anders als aus Frameworks bekannt, startet die “EventLoop” mit Beginn des ausführenden Scripts und endet sobald keine „Callbacks“ mehr vorhanden sind. Dank dieses Verhaltens ist NodeJS für das Entwickeln einer Restful API prädestiniert. [27](QUELLE)

Die Installation von NodeJS gestaltet sich für das Betriebssystem Windows nicht besonders herausfordernd. Auf der Homepage von NodeJS wird eine ausführbare Datei angeboten, welche alle nötigen Programme beinhaltet.

NPM

Durch die Installation von NodeJS wird der hauseigene sogenannte “node package manager” (kurz npm) zusätzlich installiert. Dieses Werkzeug befähigt Nutzer vorhandene und publizierte Module für die Entwicklung in NodeJS herunterzuladen oder eigene Module der Öffentlichkeit zur Verfügung zu stellen. Um npm nutzen zu können wird eine funktionierende “Bash” vorausgesetzt.

PowerShell

Um mit NodeJS effektiv arbeiten zu können benötigen alle Programmierer eine geeignete Bash (Shell). Viele Unix und Linux Distributionen besitzen eine geeignete vorinstallierte Bash die für die meisten Befehle ausreicht. Um mit Windows entsprechend zu arbeiten, empfiehlt sich die Installation von der PowerShell [28](QUELLE).

Express.js

Eine Restful API basiert auf Anfragen (requests) die an einen Server übermittelt werden. Grundsätzlich folgt auf einen request stets eine Antwort (response). Ein handelsüblicher Webserver verkörpert dieses Prinzip. Sobald in einem Webbrowser die Anfrage zu einer IP - Adresse verschickt wird, antwortet dieser mit den hinterlegten Ressourcen (Meistens vorhandene .html Dateien) (siehe Grundlagen 2.12 auf Seite 10). Um angefragte Routen wie bspw. “/getAddress” zu verarbeiten, welche keine HTML - Daten als Antwort verlangen, werden sogenannte “Middlewares” geschrieben [“s. Grundlagen “Restful API”]. Diese nehmen den angefragten Request entgegen, verarbeiten ihn und senden entweder einen abschließenden Response oder geben den Request weiter an die nächste Middleware - Instanz. Dieser Vorgang wird so lange durchgeführt bis eine beendende Antwort an den Client zurückgesendet wird.

Express.js ist ein Modul für NodeJS, dass dem Programmierer ermöglicht vollwertige Middlewares zu erstellen. Die Installation gestaltet sich sehr leicht.

Voraussetzung sind eine funktionierende Bash, sowie der Paketmanager "npm". In der Shell wird folgendes Kommando ausgeführt:

```
npm install express
```

Falls das Paket in einem Projekt abgespeichert werden soll, empfiehlt sich das folgende Kommando:

```
npm install --save express
```

MySQL

Eine MySQL - Datenbank wird für das permanente Speichern der IOTA - Adresse genutzt. Um die Datenbank aufzusetzen empfiehlt sich das Hilfsprogramm XAMPP [29]. Durch dieses Programm kann eine MySQL Datenbank gestartet und über eine lokale Webpage konfiguriert werden. Nach dem Download der ausführbaren Installationsdatei wird diese mit einem Doppelklick gestartet. Im Anschluss genügt es den Installationsanweisungen zu folgen. Nicht alle Komponenten sind notwendig, es genügen der Apache Webserver und die MySQL Unterstützung. XAMPP bietet eine GUI (graphical user interface) mit der neben einer Datenbank auch ein Apache Server gestartet werden kann. Dieser Apache Server ist notwendig um die lokale Konfigurations - Webseite für die MySQL Datenbank zu hosten.

Port Weiterleitung

Damit die API auf zukünftige Anfragen außerhalb des lokalen Systems reagieren kann, muss das Netzwerk entsprechend eingerichtet werden. Der Firewall des zentralen Routers (standard Gateway) werden entsprechende Regeln hinzugefügt, die Anfragen auf den Port 1717 (Dieser Port ist frei wählbar und wurde für dieses Projekt festgelegt) zulassen. Anfragen, die den genannten

Port adressieren, werden zu dem Endgerät weitergeleitet, auf welchem die API instanziiert ist.

12.1.2 Projektabgrenzung

Eine geeignete Kommunikation innerhalb eines Netzwerkes kann auf verschiedene Weisen geschehen. Das Verwenden einer Restful API bietet den Vorteil Informationen und Ressourcen auf vielfältige Art und Weise auszutauschen. Die Kommunikation kann dementsprechend mithilfe eines JSON - Objektes oder über andere Protokolle wie XML oder URL-Encoding gelöst werden. Anders als andere Netzwerk Kommunikationstechniken wie bspw. SOAP sind Serverressourcen direkt adressierbar, ohne das im vorhinein der Inhalt der Anfrage ausgewertet werden muss [30].

12.2 Umsetzung

Mit dem Start der API überprüft das System ob eine JSON-Datei hinterlegt ist, in der die notwendigen Datenbank-Anmeldeinformationen gespeichert sind. Falls diese Datei nicht vorhanden ist, wird eine neue erstellt, indem die Informationen abgefragt werden. Sobald der Zugang zur Datenbank gesichert ist, startet der HTTP-Server und wartet anschließend auf dem eingerichteten Port auf kommende Anfragen. An diesem Punkt sind verschiedene Routen eingerichtet, die sowohl Daten verfügbar machen, mit Daten interagieren oder Daten abspeichern. Diese verschiedenen Routen und ihre Funktionen werden im Folgenden erläutert.

12.2.1 Tank-Vorgang

Der Tankvorgang würde in einer realen Implementierung erst starten können sobald das Fahrzeug vorgefahren ist, die Zapfsäule ausgewählt und die Zapfpistole des notwendigen Kraftstoff in den Tankstutzen eingeführt wurde. Während dessen verbindet das Fahrzeug sich mit dem lokalen Netzwerk mit gegebenen Mitteln. Dieser gesamte Vorgang kann entweder von dem Fahrer oder durch einen Automatismus erfolgen und ist Voraussetzung für den

kommenden Tank-Prozesses. Der vorliegende Prototyp schreibt dafür keine Regeln vor.

Die API hört auf den Port 1717 und erwartet dort folgende Anfragen:

- {IP-Adresse des Servers}/fueling/getStationInfo
- {IP-Adresse des Servers}/fueling/initializeFueling?{ID der Zapfsäule} & {Kraftstoffart}
- {IP-Adresse des Servers}/fueling/startFueling?{Kundennummer}
- {IP-Adresse des Servers}/fueling/pauseFueling?{Kundennummer}
- {IP-Adresse des Servers}/fueling/endFueling?{Kundennummer}
- {IP-Adresse des Servers}/fueling/getFueling?{Kundennummer}

getStationInfo Die Tankstelle liefert die aktuellen Tank-Preise der verschiedenen Kraftstofftypen an das Fahrzeug in Form eines JSON-Objektes.

```
router.get('/getStationInfo', (req, resp, next) => {

    resp.status(200).json(Petrolstation.stationInfoObject);

});
```

initializeFueling Die API empfängt von dem Fahrzeug die gewählte Kraftstoffart und die ID der Zapfsäule, an der der Tank-Vorgang gestartet werden soll. Diese Daten werden gemeinsam mit einer neu erstellten Kundennummer in der Datenbank hinterlegt. Diese spezielle Zapfsäule ist ab diesem Zeitpunkt für den Tank-Prozess vorbereitet und wird anschließend nur durch die Kundennummer ansprechbar sein. Der Kunde erhält als Antwort die erstellte Kundennummer und kann den Tankvorgang beginnen.

```
router.get('/initializeFueling', (req, resp, next) => {

    if(! (req.query.fuel_type && req.query.station)) return
        next('You missed the required parameter.');

    petrolstation.initialize_fueling(req.query.fuel_type,
        req.query.station, (err, result) => {

        if(err) return next(err);

        resp.status(200).json({
            id: result
        });

    });

});
```

startFueling Diese Anfrage startet den Tankvorgang des Fahrzeuges und benötigt die in der Initialisierung vergebene Kundennummer zur Freigabe. Das Tanken wird durch eine Schleife simuliert. Es wird davon ausgegangen, dass innerhalb einer Sekunde ein Liter bzw. Kilowattstunde in das Fahrzeug getankt wird. Die Kosten werden entsprechend aufgerechnet.

```
router.get('/startFueling', check_for_initialisation, (req, resp,
    next) => {

    if(petrolstation.start_fueling(req.query.id)){
        resp.status(200).json({
            message: 'Start fueling.'
        });
    }else{
```

```
        next('Anything went wrong.');
    }

});
```

pauseFueling Auch diese Anfrage muss die in der Initialisierung vergebene Kundennummer übergeben um das Fahrzeug für den aktuellen Tank-Prozess zu autorisieren. Diese Anfrage pausiert den Tankvorgang der durch "start-Fueling" begonnen wurde. Technisch wird das erstellte Intervall gelöscht und die momentanen Beträge gespeichert.

```
router.get('/pauseFueling', check_for_initialisation, (req, resp,
    next) => {

    petrolstation.pause_fueling(req.query.id, (err, success) => {

        if(err) return next(err);

        resp.status(200).json({
            message: 'Pause fueling'
        });

    });

});
```

endFueling Diese Anfrage muss die entsprechende Kundennummer enthalten und beendet den Tank-Prozess. Das Kundenfahrzeug erhält die Anzahl der getankten Einheiten, die zugehörigen Kosten und die aktuelle IOTA-Adresse als JSON-Objekt. Die Zapfsäule wird für eine erneute Initialisierung freigegeben. Die Kosten sowie der Betrag der getankten Einheiten werden in der Datenbank zu der entsprechenden Kundennummer gespeichert.

```
router.get('/endFueling', check_for_initialisation, (req, resp,
    next) => {

    petrolstation.end_fueling(req.query.id, (err, result) => {

        if(err) return next(err);

        resp.status(200).json({
            data: result
        });

    });

});
```

getFueling Der Kunde kann über diese Anfrage die Menge des getankten Kraftstoffes und die Kosten erfahren. Dafür muss die entsprechende Kundennummer übergeben werden. Die Antwort geschieht in Form eines JSON-Objektes.

```
router.get('/getFueling', check_for_initialisation, (req, resp,
    next) => {

    resp.status(200).json({

            data: petrolstation.get_fueling(req.query.id)

    });

});
```

12.2.2 Neue Adresse generieren

Ausschließlich ein autorisierter Benutzer darf die Anfrage an die API stellen, eine neue Adresse, für den Empfang von Zahlungen, zu generieren. Die Autorisierung geschieht im Prototypen durch die „basic access authentication“ Methode, bei der ein Hash, erstellt aus einem Benutzernamen und Passwort, an den Header einer HTTP Anfrage an gehangen wird. Die API kann diese Daten auslesen und die Resource freigeben.

```
var authorizeUser = function(username, password, cb){

    db.connect();

    let db_call = new Promise((resolve, reject) => {

        db.get_user_password(username, (err, rows) => {
            db.end();
            if (!arr_is_empty(rows) && passwordHash.verify(password,
                rows[0].password)) resolve(true);
            else resolve(false);

        });

    });

    db_call.then(value => {
        cb(null, value);
    });

}
```

Die autorisierte Anfrage an den Server muss an diese Route gesendet werden:

- {IP-Adresse des Servers}/user/addNewAddress“

Serverseitig werden nach und nach Adressen generiert, solange bis eine gefunden wurde die noch nicht in der IOTA Tangle „attached“ ist. Dementsprechend wird eine verbundene „IOTA Full-Node API“ angesprochen, die Zugriff zu der Tangle hat. Nachdem eine neue Adresse gefunden wurde, wird eine wertlose Transaktion erstellt. Diese ist dafür zuständig, die neue Adresse in der Tangle zu hinterlegen.

```
PetrolstatioIotaInterface.prototype.addNewAddressToTangle = async
    function(callback) {

    //Get the newest unused address
    let get_address = new Promise((res, rej)=>{

        this.iota.api.getNewAddress(this.seed, {index: 0}, (err,
            address)=>{
            if(err){
                callback(err);
                rej(err);
            }else{
                callback(null, address);
                res(address);
            }
        });

    });

    //Put the data into the transfers Array
    try{
        let address = await get_address;
        let depth = 3;
        let minWeightMagnitude = 14;
        let transfers = [{
            value: 0,
            address: address
```

```
        }];

        //send the transfer to the tangle
        this.iota.api.sendTransfer(this.seed, depth,
            minWeightMagnitude, transfers, (err, object)=>{
            if(err){
                console.log(err);
            }else{
                console.log(object);
            }
        });

    }catch(err){
        callback(err);
    }

};
```

Die Adresse wird zudem in der lokalen MySQL Datenbank gespeichert. Als Antwort erhält der autorisierte Benutzer die neue Adresse zusammen mit den veralteten Adressen aus der Datenbank:

```
{
    "newAddress": "LOZMZOKJ...AWSQFFXX",
    "affectedRows": 1,
    "savedAddresses": [
        "A9XZMWXP...TRKKUHZD",
        "GHDGOJFL...TVOETV9Z",
        "YOUZCEEZ...AKNKQSRW",
        "LOZMZOKJ...AWSQFFXX"
    ],
    "message": "Address is successfully added to the database and
        will be used for later transactions."
```

```
}
```

12.3 Ergebnis

Lorem Ipsum

Teil III

Schlussteil

Kapitel 13

Evaluierung

dbsjakdbasdlas

Kapitel 14

Fazit

Literaturverzeichnis

[Bro09] Brown, Daniel R. L.: *Elliptic Curve Cryptography*. http://www.secg.org/sec1-v2.pdf, 2009. Abgerufen am 08.10.2018.

[CNP+16] Crosby, Michael (Google), (Yahoo) Nachiappan, (Yahoo) Pradan Pattanayak, (Samsung Research America) Sanjeev Verma und (Fairchild Semiconductor) Vignesh Kalyanaraman: *BlockChain Technology: Beyond Bitcoin*. https://j2-capital.com/wp-content/uploads/2017/11/AIR-2016-Blockchain.pdf, 2016. Abgerufen am 09.10.2018.

[CY04] Chang, Ming-Hsin und Yi-Shiung Yeh: *Improving Lamport one-time signature scheme*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.924.4691&rep=rep1&type=pdf, 2004. Abgerufen am 07.10.2018.

[EGSvR16] Eyal, Ittay, Adem Efe Gencer, Emin Gün Sirer und Robbert van Renesse: *Bitcoin-NG: A Scalable Blockchain Protocol*. https://www.usenix.org/system/files/conference/nsdi16/nsdi16-paper-eyal.pdf, 2016. Abgerufen am 07.10.2018.

[Hig17] Higgins, Stan: *From $900 to $20,000: Bitcoin's Historic 2017 Price Run Revisited*. https://www.coindesk.com/900-20000-bitcoins-historic-2017-price-run-revisited/, 2017. Abgerufen am 14. Oktober 2018.

[Nak08] Nakamoto, Satoshi: *Bitcoin Whitepaper*. www.whitepaper.de/bitcoin/sn, 2008. Abgerufen am 14. Oktober 2018.

[Pop17] Popov, Serguei: *The Tangle*. http://iotatoken.com/IOTA_Whitepaper.pdf, 2017. Abgerufen am 07.10.2018.

[RS] Rogaway, Phillip und Thomas Shrimpton: *Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance*. https://link.springer.com/content/pdf/10.1007 Abgerufen am 07.10.2018.

[SSUF16] Schlatt, Vincent, André Schweizer, Nils Urbach und Gilbert Fridgen: *Blockchain: Grundlagen, Anwendungen und Potenziale*. https://www.fim-rc.de/Paperbibliothek/Veroeffentlicht/642/wi-642.pdf, 2016. Abgerufen am 07.10.2018.