HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Marianne Poser

Probabilistische Mikrozahlungen auf der Blockchain

Mittweida, 2019

Fakultät Angewandte Computer- und Biowissenschaften

BACHELORARBEIT

Probabilistische Mikrozahlungen auf der Blockchain

Autor:
Frau Marianne Poser

Studiengang:
Angewandte Informatik, IT-Sicherheit

Seminargruppe:
IF15wl-B

Erstprüfer:
Prof. Dr. – Ing. Andreas Ittner

Zweitprüfer:
M. Sc. Steffen Kux (Slock.it)

Einreichung:
Mittweida, 08.11.2019

Verteidigung/Bewertung:

Mittweida, 2019

Faculty Applied Computer Sciences &
Biosciences

BACHELOR THESIS

Probabilistic Micropayments on the Blockchain

author:
Ms. Marianne Poser

course of studies:
Applied Computer Sciences, IT-Security

seminar group:
IF15WI-B

first examiner:
Prof. Dr.-Ing. Andreas Ittner

second examiner:
M. Sc. Steffen Kux (Slock.it)

submission:
Mittweida, 08.11.2019

defence/evaluation:

Mittweida, 2019

Bibliografische Angaben

Poser, Marianne: Probabilistische Mikrozahlungen auf der Blockchain, 71 Seiten, 8 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2019

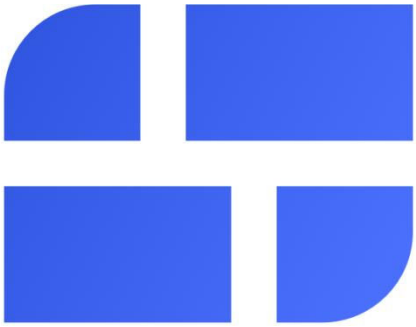
Dieses Werk ist urheberrechtlich geschützt.

Satz: LaTeX

Referat

Diese Bachelorarbeit befasst sich mit der Herausforderung, kleine Zahlungen effizient mithilfe der Blockchain umsetzen zu können. Dafür werden verschiedene existierende Zahlung-Schemata vorgestellt werden. Sie werden dabei auch auf ihre Eignung im Bezug auf die Anforderungen eines IoT-Gerätes geprüft. Besonderer Fokus wird auf die probabilistischen Zahlungen gelegt. Abschließend wird die Implementierung eines Prototypen vorgestellt und die Anwendbarkeit eines solchen Systems ausgewertet.

Diese Bachelorarbeit wurde betreut von der Slock.it GmbH.



Slock.it

Inhaltsverzeichnis

Inhaltsverzeichnis ..... **I**

Abbildungsverzeichnis ..... **III**

1 Einleitung **1**

1.1 Motivation *1*

1.2 Kapitelübersicht *2*

2 Blockchain **3**

2.1 Grundlagen *3*

2.2 Ethereum *4*

2.3 Incubed *7*

3 Micropayments **9**

3.1 Grundlagen *9*

3.2 Bestehende Micropayment-Lösungen *11*

3.2.1 Serverbasierend 11

3.2.2 Blockchainbasierend 15

3.2.3 Potenzial und Herausforderungen 23

4 Probabilistische Zahlungen **27**

4.1 Grundlagen *27*

4.2 Existierende Lösungen - zentral *28*

4.2.1 Micropayments Revisited 28

4.2.2 Peppercoin 29

4.2.3 Coin Flipping 30

4.2.4 Polling 31

4.3 Existierende Lösungen - dezentral *32*

4.3.1 Ansätze nach Pass und Shelat 32

4.3.2 Ansatz nach Caldwell 36

4.3.3 Orchid 37

4.3.4 Decentralized Anonymous Micropayments 39

4.3.5 Streamflow 40

4.3.6 Poollösung 43

4.3.7 Fazit vorgestellter Lösungen 43

5 Umsetzung 47

5.1 Smart Contracts 47

5.2 Client Seite 53

6 Fazit 57

Literaturverzeichnis 59

Erklärung 71

II. Abbildungsverzeichnis

5.1 Flowchart 1 - Set-up 47
5.2 Codebeispiel 1 - Ticket (Struktur übernommen von Streamflow) 48
5.3 Flowchart 2 - redeemWinningTicket und Overspend 49
5.4 Codebeispiel 2 - requireValidWinningTicket Funktion 50
5.5 Codebeispiel 3 - redeemWinningTicket Funktion 51
5.6 Codebeispiel 4 - claimFromReserve Funktion 52
5.7 Codebeispiel 5 - withdrawDeposit Funktion 53
5.8 Flowchart 3 - Ticketexchange 54

1 Einleitung

1.1 Motivation

Die Blockchain ist als Technologie inzwischen ein Trendthema. Sowohl in der öffentlichen Diskussion in den Medien, als auch in der Finanzbranche und im Technologiebereich. Eine weitere Entwicklung, die in verschiedenen Bereichen immer wieder thematisiert wird, ist das Internet of Things (Internet der Dinge), kurz IoT. Beide Begriffe gelten inzwischen als „Buzzwords“, werden also genutzt, um Aufmerksamkeit zu erzeugen. [1] Beide Begriffe sind auch auf dem Gartner Hype-Zyklus für neue Technologien von 2018 zu finden und sollen erst in fünf bis zehn Jahren auf ihrem „Plateau der Produktivität“ ankommen. [2] Das zeigt die Aktualität und Präsenz der Themen, mit denen sich diese Arbeit beschäftigt.

Ein Thema, welches die Menschheit hingegen vermutlich schon immer beschäftigt, sind Zahlungen. Von der Entwicklung erster Zahlungsmittel bis zum Ablauf einer elektronischen Zahlung liegen Tausende Jahre von Weiterentwicklung und Forschung. [3] Diese Weiterentwicklung von neuen Konzepten und der Einsatz neuer Technologien, wie der Blockchain, hält an. Aus dieser Entwicklung sind unter anderem Schemata für sehr kleine Zahlungen, sogenannte Micropayments, hervorgegangen.

Die Firma Slock.it hat einen Client entwickelt, welcher IoT-Geräten die vollständig dezentrale Verbindung mit der Blockchain ermöglicht. Dabei werden Abfragen über Server getätigt, deren Antworten der Client selbst prüfen und validieren kann. Allerdings wird für diesen Ablauf noch eine Lösung für ein Anreizsystem (Incentivisation) für die Informationsbereitstellung der Server gesucht. Sie sollen mit minimalen Beträgen für ihren Service belohnt werden.

In dieser Arbeit wird das Potenzial von probabilistischen Mikrozahlungen geprüft. Dafür wird ein Zahlungsschema gesucht, welches

- geringe Anforderungen an die IoT-Geräte stellt,
- nur minimale Kosten pro Zahlung verursacht,
- ein gewisses Maß an Sicherheit bietet und
- ohne den Einsatz einer zentralen Instanz umgesetzt werden kann.

Welche Ansätze dieser Herausforderung begegnen, welche neuen Probleme dabei aufkommen werden und ob sich darunter eine Lösung für das Schaffen eines Incentives befindet, wird Thema dieser Arbeit sein. Besonders die Unterkategorie probabilistischer Micropayments, welche in dieser Betrachtung besonderen Stellenwert hat, wird ausführlich behandelt werden. Es wird gezeigt werden, was diese von anderen Ansätzen

unterscheidet, welches Potenzial sie haben, aber auch mit welchen Herausforderungen sie verbunden sind.

Mithilfe von Literaturarbeit werden verschiedene Micropayments und insbesondere probabilistische Micropayment Ansätze untersucht werden. Dabei bleibt der Fokus auf den Anforderungen, welche eine Umsetzung mit IoT-Geräten auf der Blockchain mit sich bringen und welche verschiedenen Lösungsstrategien dafür gefunden werden können. Um eine breite Auswahl an Lösungsansätzen zu haben und um ein grundlegendes Verständnis der Thematik zu gewährleisten, werden sowohl zentrale als auch dezentrale Ansätze untersucht werden.

Zudem wird ein probabilistisches Micropaymentsystem durch die Entwicklung eines eigenen Prototypen in der Praxis verwirklicht werden. Die Überführung eines theoretischen Konzepts in die Praxis veranschaulicht die Problematik und offenbart weitere Herausforderungen.

1.2 Kapitelübersicht

Auf dieses erste einleitende Kapitel folgen fünf weitere, deren Inhalt kurz vorgestellt werden wird.

Im Fokus des zweiten Kapitels liegt die Technologie der Blockchain. Es wird kurz erläutert werden, was eine Blockchain ist, wie sie aufgebaut wird und welche Besonderheiten sie mit sich bringt. Der Schwerpunkt ist dabei die Ethereum Blockchain. Außerdem wird die Arbeitsweise des Incubed Protokolls vorgestellt werden.

Kapitel drei befasst sich mit dem Begriff Micropayment und den verschiedenen Ansätzen. Dabei wird zwischen zentralen und dezentralen Lösungen unterschieden und auf deren unterschiedlichen Herausforderungen in Bezug auf Blockchain und Incubed hingewiesen.

Im darauffolgenden vierten Kapitel werden probabilistische Micropaymentsysteme erklärt und verschiedene Konzepte vorgestellt. Auch hier werden Vor- und Nachteile bezüglich einer Anwendung in Incubed abgewogen. Darüber hinaus werden auch Lösungen für andere Blockchains untersucht.

Das fünfte Kapitel behandelt die Umsetzung eines probabilistischen Micropayments auf der Ethereum-Blockchain. Die Hauptfunktionen dieses Prototypen werden vorgestellt und erklärt. Dabei wird sowohl die On-Chain, als auch die Off-Chain Seite betrachtet.

Im abschließenden Kapitel wird ein Fazit gezogen und die Antwort auf die Kernfrage der Arbeit zusammengefasst. Außerdem wird ein Blick auf die zukünftige Entwicklung geworfen werden.

2 Blockchain

2.1 Grundlagen

Die Blockchain ist eine Kette von Blöcken, welche mittels Kryptografie unveränderlich miteinander verbunden sind. In diesen Blöcken werden (Transaktions-)Daten abgelegt, die von jedem Teilnehmer des Blockchain-Netzwerkes gelesen werden können. Dadurch können alle die Geschehnisse auf der Blockchain nachvollziehen. Allerdings können diese Daten nicht mehr geändert werden, wenn der Block einmal Teil der Kette geworden ist. Dadurch ist die Blockchain vor Datenmanipulation geschützt.

Das Netzwerk bei einer Blockchain verzichtet auf eine zentrale Einheit, welche steuert und verwaltet und der die Nutzer vertrauen müssen. Stattdessen funktioniert das Netzwerk dezentral und wird durch die Nutzer selbst organisiert. Jeder Knoten ist eine Absicherung, da alle Knoten redundant die Blockchain speichern und alle abgeschaltet werden müssten, um die Blockchain außer Betrieb zu nehmen. Dadurch ergibt sich eine hohe Verfügbarkeit der Blockchain. Die IT-Schutzziele Vertraulichkeit, Authentizität und Integrität werden durch den Einsatz von Kryptografie erreicht, was später erläutert werden wird.

Das erste Paper, welches die Blockchain Technologie umfangreich erläutert, wurde 2008 von Satoshi Nakamoto veröffentlicht. Darin wird Bitcoin, ein Peer-to-Peer Electronic Cash System, also ein elektronisches Bezahlsystem in einem verteilten Netzwerk, erklärt. Zu diesem Zeitpunkt lag der Fokus darauf, Geldtransaktionen ohne Intermediäre abwickeln zu können. Inzwischen gilt die Blockchain als Technologie, die für verschiedene Zwecke eingesetzt werden kann. Der erste Block der Bitcoin Blockchain wurde am 03.01.2009 erstellt und ist der Startpunkt der noch heute bestehenden Bitcoin Blockchain. In diesen Blöcken werden die Transaktionen, welche die Teilnehmer auslösen, gespeichert. Sobald eine Transaktion in einem Block abgelegt wurde und auf diesen Block aufbauend die Kette weiter erstellt wird, kann diese Transaktion weder geändert, noch rückgängig gemacht werden.

Damit ein Block in die Kette aufgenommen wird, muss sich innerhalb des Netzwerkes geeinigt werden. Für diesen Mechanismus, um Übereinstimmung beziehungsweise Konsens zu finden, existieren verschiedene Methoden. Bitcoin benutzt einen Proof-of-Work („Arbeitsnachweis“) Algorithmus, ursprünglich Hashcash, welcher auf Hashfunktionen basiert. Dabei weist der Ersteller eines Blocks nach, dass er dafür eine gewisse Arbeit geleistet hat, indem er einen Hash erzeugt, welcher bestimmte Kriterien erfüllt. Da ausgehend vom Input nicht auf den ausgegebenen Hash geschlossen werden kann, sind Hashfunktionen Einwegfunktionen. Es ist nur durch wiederholtes Ändern des Inputs und einer neuen Hasherzeugung möglich, Hashs zu finden, die den Kriterien entspre-

chen. Bei Bitcoin ist die benötigte Rechenleistung durch die Anzahl führender Nullen, die der Hash haben soll, anpassbar. Im Durchschnitt wird alle zehn Minuten ein passender Hash gefunden und ein neuer Block erstellt. Die Ersteller eines Blocks werden Miner genannt und für ihren Aufwand bei der Berechnung belohnt.

Verschiedene Techniken aus der Kryptografie werden verwendet, um eine Transaktion sicher durchzuführen. Eine dieser Techniken sind asymmetrische Verschlüsselungssysteme. Dabei verfügt der Besitzer eines Accounts über ein Schlüsselpaar, bestehend aus einem öffentlichen (public) und einem geheimen (secret/private) Schlüssel (key). Nur der Besitzer des geheimen Schlüssels ist in der Lage, Transaktionen von dem dazugehörigen Account auszuführen. Denn um eine Transaktion von einem Konto auszuführen, muss diese Transaktion mit dem geheimen Schlüssel signiert werden. Das Netzwerk kann deren Richtigkeit überprüfen, indem sie die Signatur mithilfe des öffentlichen Schlüssels validiert und verifiziert. Dadurch wird Authentizität in diesem Netzwerk erreicht. [4, 5]

Die Transaktionen werden im Netzwerk verteilt und von den sogenannten Minern aufgenommen. Diese speichern Transaktionen in Blöcke und berechnen den passenden Hash. Der neue Block wird an die bestehende Kette angefügt. Existieren zwei widersprüchliche Blöcke in einer Kette, entscheiden sich die Miner nach dem Consensus Algorithmus für die längere Kette, auf welche weiter aufgebaut wird. Die Transaktionsdaten werden in einer bestimmten Datenstruktur, dem sogenannten Merkle Tree, in einen Block abgelegt. Die Transaktionen werden dabei als Hash in die Blätter des Baums gespeichert und alle weiteren Knoten enthalten einen Hash über ihre Kind-Knoten. Der letzte Knoten, welcher Merkle Root genannt wird, bildet die Daten aller Transaktionen in einem Hashwert ab. Jede Änderung einer Transaktion bewirkt eine Änderung des Merkle Root. Dieser Hashwert wird im Header des Blocks abgelegt und sichert damit die Datenintegrität. Außerdem kann mit dieser Struktur schnell geprüft werden, ob eine Transaktion im Block vorhanden ist, ohne alle Transaktionsdaten zu durchsuchen. [6, 7]

2.2 Ethereum

Ethereum ist eine dezentralisierte, öffentliche Blockchain Plattform, welche 2015 gestartet wurde. 2013 wurde das White Paper dazu von Vitalik Buterin veröffentlicht. Ethereum konzentriert sich nicht auf ein reines Bezahlsystem, wie Bitcoin. Stattdessen bietet Ethereum die Möglichkeit, aufbauend auf die Blockchain verschiedene Anwendungen zu entwickeln. [8] Um das zu erreichen, unterscheidet sich die Ethereum Blockchain von der Bitcoin Blockchain in verschiedenen Aspekten. Im Folgenden soll erklärt werden, wie Programme in die Ethereum Blockchain integriert und ausgeführt werden können.

Ethereum basiert auf Accounts, welche eine eindeutige Adresse haben, dabei können beliebig viele Accounts erstellt werden. Es existieren zwei Arten von Accounts:

sogenannte External Owned Accounts (EOA), welche einen externen Besitzer haben (Mensch oder Maschine) und über ein Schlüsselpaar verfügen, sowie Contract Accounts. Diese Contract Accounts (CA) enthalten Code, welcher durch Senden einer Transaktion an diesen ausgeführt wird. CA haben internen Speicher, welchen sie auslesen und schreibend verändern können. Diese Programme, welche auf der Ethereum Infrastruktur ausgeführt werden, werden auch Smart Contracts genannt. Smart Contracts haben ein funktionsbasiertes Interface und können sowohl Nachrichten versenden, als auch Funktionen eines anderen Smart Contracts aufrufen. Allerdings muss jede Aktion initial von einem EOA ausgelöst werden. Ein EOA kann, wenn er einen Smart Contract aufruft, Parameter für die entsprechende Funktion mitschicken und auch Assets übertragen.

Bei Ethereum können diese Assets die Währung „Ether“ sein oder zum Beispiel auch Token. Token kann als Wertmarke übersetzt werden und bildet einen gewissen Wert auf der Blockchain ab. Der bekannteste Standard für Token ist ERC20.[1] Neben dem ERC20 Standard existiert außerdem der ERC721, welcher sich durch seine fehlende Fungibilität auszeichnet. Das bedeutet, dass sich ein Token durch seine Historie auszeichnet und jeder Token einmalig und damit nicht austauschbar ist. [9] Token und Ether können unter allen Accounts ausgetauscht werden. Mithilfe sogenannter „payable“ Funktionen im Smart Contract werden Einzahlungen an einen CA ermöglicht. Dieser CA kann sie dann verwalten und unter bestimmten, festgelegten Bedingungen wieder auszahlen.

Um einen eigenen Smart Contract für die Ethereum Blockchain zu schreiben, nutzen die meisten die Programmiersprache Solidity. Anschließend wird das Programm in EVM (Ethereum Virtual Machine) Bytecode kompiliert und durch Senden an die Nulladresse registriert. Die Nulladresse ist eine spezielle Ethereum Adresse, welche nur aus Nullen besteht und an welche Transaktionen geschickt werden, die einen Contract erstellen. Die EVM ist das Herz des Ethereum Protokoll und ist für die Ausführung von Programmen zuständig. Sie ist eine quasi-turing vollständige virtuelle Maschine. Alle Prozesse dürfen nur eine beschränkte Anzahl an Berechnungsschritten haben, weshalb sie nur als „quasi“-turing vollständig gilt. Es ist also möglich in einem Smart Contract Schleifen zu programmieren, jedoch ist es nicht möglich, die EVM mit Endlosschleifen zu besetzen. Damit begegnet man dem Halteproblem, das besagt, dass es nicht vorhersagbar ist, ob ein Programm enden oder für immer laufen wird.

Die Beschränkung wird durch eine minimale Bezahlung für jeden Berechnungsschritt und Speichernutzung gesetzt. Die Währungseinheit, die dafür verwendet wird, wird „Gas“ genannt. Jede Operation, die die Ausführung eines Smart Contracts oder eine Transaktion benötigt und auch jedes Speichern von Daten in die Blockchain muss mit Gas bezahlt werden. Diese Kosten können ausschließlich vom Sender der Transaktion bezahlt werden und Gas existiert auch nur in der EVM. Neben den „gas cost“, also der

[1] ERC (Ethereum Request for Comments) ist ein Label, welches vergeben wird, um spezifische Standards zu kennzeichnen.

Menge an Gas, die benötigt wird, gibt es den „gas price“, was die Menge an Ether angibt, die man gewillt ist, pro Gas zu zahlen. Dadurch entkoppelt man die Höhe der Kosten vom schwankenden Wert des Ether. Jeder Block hat außerdem ein „block gas limit“, welches die Anzahl an Operationen in einem Block beschränkt. Wird für eine Transaktion nicht ausreichend Gas im Voraus mitgeschickt, werden die bereits ausgeführten Operationen rückgängig gemacht und eine Fehlermeldung ausgegeben. Bei einer erfolgreichen Berechnung wird der State, also der Zustand der Blockchain aktualisiert und der entsprechende Miner ist der Empfänger der Gas Zahlung. [10, 11]

Full Nodes verifizieren die Blöcke, die sie erhalten und verbreiten sie weiter im Netzwerk. Sie synchronisieren kontinuierlich die komplette Blockchain und brauchen dafür entsprechende Hardwareressourcen. Aktuell benötigt die Blockchain 200 GB Speicherplatz. Um auch Umgebungen mit geringerer Kapazität den Zugang zu Ethereum zu ermöglichen, bietet Ethereum ein Light Client Protokoll. Es basiert darauf, dass nicht dauerhaft die komplette Blockchain synchronisiert werden muss, sondern analog zu Bitcoin mit simplified payment verification (SPV) gearbeitet wird. Statt jeden Block vollständig zu synchronisieren, lädt der Light Client nur Blockheader herunter, welche die Merkle Tree Root enthalten. Davon ausgehend prüft er nur den kleinen Teil, der für ihn relevant ist. Möchte er beispielsweise eine bestimmte Information prüfen, betrachtet er ausgehend von der Merkle Tree Root rekursiv die entsprechenden Knoten des Baums, bis er die gewünschte Information hat.

Als Nachteil können diese Light Clients nicht direkt Daten aus der Blockchain lesen. Stattdessen nutzen sie einen Full Node, um auf die gewünschten Transaktionsdaten zugreifen zu können. Deren Richtigkeit können sie mit Hilfe der Blockheader validieren. Sie selbst verifizieren allerdings keine Blöcke aus dem Netzwerk, sondern vertrauen dabei auf die Full Nodes. Diese Light Clients gehen einen Kompromiss zwischen Sicherheit und Ressourcenanforderungen ein. [12–14]

Neben Light Clients gibt es sogenannte Remote Clients, welche sich ohne Verifizierung auf den Full Node verlassen. Das bedeutet Verlust von Sicherheit und Anonymität, aber auch ein schnelleres Set-up und weniger Speichernutzung. Remote Clients beschränken sich auf eine Teilmenge an Funktionen, die ein Full Node hat und können beispielsweise für mobile Wallets[2]genutzt werden. Durch den Einsatz von Remote Clients, welche über einen Full Node Zugang zur Blockchain erhalten, verliert die Blockchain für den Client ihren dezentralen Charakter. [11] Ein weiterer Ansatz, sich sehr ressourcenschonend und mit hoher Sicherheit mit der Blockchain zu verbinden, soll im folgenden Kapitel beschrieben werden.

[2] Wallets enthalten die Zugangsinformationen (Schlüsselpaar) zu Objekten auf der Blockchain.

2.3 Incubed

Das Incubed Protokoll ist Teil der Incubed SDK, welche von slock.it entwickelt wurde. Incubed ist ein „INcentivized remote Node Network“ und möchte IoT-Geräten den Zugang zur Blockchain sicher ermöglichen. Deren bisheriges Problem war, dass Blockchain Clients zu hohe Anforderungen an die eingeschränkte Hardware der IoT-Geräte stellen. Selbst Light Clients benötigen viel Bandbreite und dauerhaften Internetzugang, da sie sich mit dem Netzwerk synchronisieren müssen. Auf der anderen Seite benötigen existierende Remote Client Lösungen eine dritte Partei, der sie vertrauen müssen (trusted third party). Ziel von Incubed ist es einen Client zu kreieren, der keine Synchronisierung mit der Blockchain benötigt, aber Informationen, die er von Dritten erhält, verifizieren kann. [15]

Incubed basiert auf zwei Akteuren:

- Incubed Nodes: Full Nodes, welche sich registrieren und ein Deposit hinterlegen und das Interface zwischen den Incubed Clients und der Blockchain darstellen.
- Incubed Clients: Remote Nodes, welche aus der Registry wählen, mit welchen Incubed Nodes, sie kommunizieren möchten, um Informationen aus der Blockchain zu erhalten.

Zu Beginn registriert sich der (Incubed) Node mit seinem Deposit und der (Incubed) Client erhält eine List an registrierten Nodes. Aus dieser kann er nun wählen, mit welchen er kommunizieren möchte. Er fragt beispielsweise Node B nach einer Information und sendet ihm dafür ein RPC (Remote Procedure Call) Request. Außerdem schickt er ihm ein Validation Request (Bestätigungsanfrage) für Node A und C. Der Node B soll also nicht nur die Information zurück geben, sondern auch signierten Block Hashs von Node A und C. Node B prüft deren Korrektheit und schickt ein RPC Response mit diesen Informationen zurück an den Client. Diese Antwort enthält den Merkle Proof, den Blockheader und die signierten Block Hashs. Mit diesen Informationen kann der Client verifizieren, dass sein Request korrekt vom Node B beantwortet wurde.

Dabei muss der Client weder synchronisieren, noch etwas speichern und arbeitet daher als stateless und ist ein „Minimal Verficiation Client“. Auch IoT-Geräte mit sehr geringer Performance und beschränkten Ressourcen erhalten dadurch einen vertrauenswürdigen und schnellen Zugang zur Blockchain. Die Absicherung erfolgt durch das Deposit, welches die Full Nodes verlieren, sollten sie Fehlinformationen weitergeben. Watchdogs können eine zusätzliche Maßnahme sein, um bösartige Nodes aufzudecken und zu bestrafen. Sie sind Full Nodes, agieren allerdings als wären sie ein Client und stellen Anfragen an Nodes, deren Antwort sie selbst mit der Blockchain abgleichen können. So können auch bösartige Nodes, die sich gegenseitig ihre falschen Antworten bestätigen, entdeckt werden. Als Anreiz können Watchdogs einen Teil des Deposits erhalten, wenn sie einen bösartigen Node finden.

Um einen Vorteil für die Full Nodes zu haben, die als Incubed Nodes zur Verfügung stehen, enthält Incubed ein Belohnungssystem. Demnach wird bei jedem RPC Request eines Clients auch eine Zahlung ausgeführt. Der Zahlungsfluss ist vom Client zum Node B, welcher einen Teil behält und aber auch an Node A und Node C etwas weitergibt. Diese werden für ihre signierten Block Hashs dadurch ebenfalls bezahlt. Eine Herausforderung von Incubed ist es, dieses Belohnungssystem möglichst effizient und dennoch sicher zu gestalten. Der Betrag wird gering sein und entsprechend sollte der Aufwand pro Zahlung minimal sein und sich an die Anforderungen eines IoT-Geräts anpassen. Die Zahlungen gelten daher als Micropayments, welche Thema des nächsten Kapitels sind. [16, 17]

3 Micropayments

3.1 Grundlagen

Micropayments sind Zahlungen sehr kleiner Beträge. Sie sind durch die voranschreitende Digitalisierung stärker in den Vordergrund gerückt. Ein Grund dafür ist, dass sie eingesetzt werden können, um digitale Güter zu monetarisieren. Bisher wurden diese größtenteils kostenfrei oder unter Einsatz von Werbung angeboten. Das Ziel von Micropayments an dieser Stelle ist es, eine Alternative zu Abonnements zu bieten und eine “Pay-per-Use” Bezahlung zu ermöglichen. Micropayments existieren natürlich auch in der offline Welt, wo diese durch spezifische Infrastrukturen, vor allem an Automaten abgewickelt werden. Beispiele hierfür sind Kaugummi-, Getränke- und Parkautomaten, aber auch Mautstationen können als Umsetzung eines Micropayment-Schemas gesehen werden. [18]

Im digitalen Bereich müssen andere Ansätze gefunden werden, um die Transaktionskosten soweit zu minimieren, dass sie geringer sind, als der Betrag der Zahlung. Micropayments bringen teilweise andere Anforderungen mit sich, als Macropayments und es existieren verschiedene Ansätze diesen zu begegnen. Grundlegend muss dabei zwischen zentralisierten Systemen, die einen Punkt haben, über den alle Zahlungen abgewickelt werden und dezentralen Systemen, die darauf verzichten, unterschieden werden. Für beide Systeme sollen sowohl Micropayment-Ansätze, als auch die Anforderungen, Probleme und Anwendungen in diesem Kapitel behandelt werden.

Der Hauptunterschied zwischen Micro- und Macropayments liegt in der Höhe des Betrags. Micropayments decken dabei den Bereich der Kleinstbeträge, wie wenige Cent oder Bruchteile dieser, bis zu wenigen Euro ab. Würden diese Beträge durch Macropayments transferiert werden, wären unter Umständen die Verarbeitungskosten höher als der zu zahlende Betrag. Durch spezielle Mircopayment Schemata sollen die Kosten pro Transaktion minimiert werden. Außerdem soll die Bezahlung schnell und wenn möglich, direkt erfolgen. Dadurch kann die bezahlte Leistung unmittelbar erbracht werden, was in vielen Anwendungen zwingend nötig ist. [19]

Makropayments nutzen häufig aufwendigere Verifikationsmaßnahmen, beispielsweise durch die Bearbeitung einer zentralen, vertrauenswürdigen Instanz. Daher ist die Sicherheit in Makropayments meistens höher, was aufgrund der größeren Beträge auch nötig ist.

Micropayments können in verschiedenen Bereichen Anwendung finden. Dabei sollen sie das Angebot an Zahlungsmethoden um Pay-per-Use erweitern. Sie können zum Beispiel eingesetzt werden, um werbefreien Inhalt bereitzustellen. Dazu zählen der

Abruf einzelner Zeitungsartikel, die Abrechnung eines Streams pro Minute oder das Nachschlagen in einem Online-Wörterbuch. Außerdem kann es für Belohnungen genutzt werden, beispielsweise für Knoten in einem P2P Netzwerk oder das Bereitstellen von Bandbreite.

Auch in der Kommunikationsbranche sind Einsatzmöglichkeiten vorhanden, man könnte das Verschicken, Lesen und Veröffentlichen von Nachrichten monetarisieren und ein Belohnungssystem zur Spam-Vermeidung einsetzen. Andere, vermeintlich kleine Dienste, wie Datenbankabfragen, können mit Micropayment-Schemata effizient abgerechnet werden.

Der Grundaufbau eines Zahlungsschemas beruht in zentralen Systemen auf drei Parteien. Es gibt einen Zahlenden, auch User, Kunde oder Sender genannt, der einen monetären Wert ausgibt. Das Ziel dieser Zahlung ist der Verkäufer beziehungsweise Lieferant. Die dritte Partei, die in verschiedenen Formen auftritt, kann eine Bank sein, wird aber auch Broker genannt. Ihre Funktion variiert, sie kann für das Auszahlen der Macropayments verantwortlich sein, oder als vertrauenswürdige Einheit angenommen und für Sicherheitsaspekte genutzt werden. Basiert ein Schema auf Accounts, so werden diese durch sie zur Verfügung gestellt und organisiert.

Um die Transaktionskosten zu minimieren, wird immer versucht, mehrere kleinere Zahlungen zu wenigen großen Zahlungen zusammen zu fassen. Die vielen kleinen Payper-Use Zahlungen sollen also vereint werden und können entweder im prepaid oder im postpaid Ansatz final bezahlt werden. Im prepaid Ansatz sind alle Zahlungen zuvor gedeckt, der postpaid Ansatz funktioniert auf Kredit-Basis, es wird also im Nachhinein erst bezahlt. [20]

An ein Micropayment Schema werden verschiedene Anforderungen gestellt, die je nach Nutzung variieren. Eine offensichtliche Forderung ist, dass die Verarbeitungskosten pro Zahlung minimal gehalten werden. Der Erfolg eines Schemas ist jedoch auch an weitere Faktoren geknüpft. Ein wichtiger Aspekt ist die Akzeptanz des Konzeptes. Damit der Sender ein System akzeptiert, muss es einfach zu nutzen, beziehungsweise schnell und günstig einsetzbar sein. Ein Empfänger hingegen könnte eine höhere einmalige Investition tätigen, wenn die Effizienz pro Zahlung dafür steigt. [21]

Für beide Seiten ist außerdem eine schnelle Abwicklung von Zahlungen wichtig, damit die Güter direkt übertragen werden können. Wie umfangreich das Set-up, also wie hoch die Eintrittsschwelle für das System ist, variiert nach Anwendung. Ist die Kunden-Lieferanten-Beziehung sehr kurzweilig, sollte das Schema unabhängig davon arbeiten können. Der Aufwand für Registrierung, das Erstellen eines Kontos und Einzahlung hat Einfluss auf die Akzeptanz aller Nutzer.

Für größere Systeme sollte das eingesetzte Schema auch nach seinen Skalierungs-

möglichkeiten ausgewählt werden. Der Sicherheitsaspekt sollte trotz geringer Beträge nicht vernachlässigt werden. Ein erschwerender Aspekt kommt hinzu, wenn der Wunsch nach Anonymität beim Sender beachtet werden soll. Eine weitere Herausforderung, der sich alle umfangreicheren Schemata stellen müssen, ist das Double Spend Problem. Wenn nicht jede Transaktion online verarbeitet werden kann, wie wird dann sichergestellt, dass Angriffe entdeckt und abgewehrt oder bestraft werden können?

Im Folgenden sollen verschiedene existierende Lösungen vorgestellt und der jeweilige Umgang mit diesen Anforderungen erforscht werden.

3.2 Bestehende Micropayment-Lösungen

3.2.1 Serverbasierend

Ein Server ist eine zentrale Einheit, welcher Dienste für andere zur Verfügung stellt. Viele Micropayment Schemata arbeiten mit einem Server, wobei dieser verschiedene Aufgaben erfüllen kann. Zum einen wird er häufig als Aggregator eingesetzt, welcher die Micropayments koordiniert und zu einer Geldübertragung von höheren Wert zusammenfasst. Dafür müssen sowohl Nutzer, als auch der Verkäufer ein Konto bei diesem Server besitzen, sodass dieser die Beträge auch transferieren kann.

Ein Beispiel für eine solche Herangehensweise ist NetBill. Dieses sogenannte „Security and Transaction Protocol“ wurde 1995 veröffentlicht. Der NetBill Server fungiert als Account Server und hat sowohl für jeden Nutzer, als auch für jeden Verkäufer einen Account. Das Protokoll sieht zunächst eine Preisverhandlung zwischen Nutzer und Verkäufer vor, anschließend wird das angefragte Produkt verschlüsselt an den Nutzer übertragen. Für die Bezahlung schickt der Nutzer eine Zahlungsaufforderung zum Verkäufer. Der Verkäufer stimmt dieser Zahlungsaufforderung zu und schickt sie mit der Entschlüsselungsinformation zum NetBill-Server. Dieser belastet den Account des Nutzers und schreibt diesen Wert dem Account des Verkäufers gut. Anschließend gibt er diese Info an den Verkäufer, welcher daraufhin dem Nutzer die Entschlüsselungsinformation gibt.

Sollte etwas in diesem Schema nicht funktionieren, beispielsweise weigert sich der Verkäufer die Entschlüsselungsinformation an den Nutzer zu übergeben, kann er sich an den NetBill Server als zentrale Einheit wenden. Dieser Ansatz ist ein klassischer Prepaid Ansatz für den Nutzer. Er füllt sein Konto durch eine normale Bank- oder Kreditkarte auf und kann von da aus an verschiedene Verkäufer beliebig große Zahlungen durchführen. Der Verkäufer kann zu jedem Zeitpunkt Geld von seinem NetBill Account auf sein reguläres Konto transferieren. [22]

Diese Herangehensweise modifiziert den Vorgang eines Macropayments nur in dem

Maße, dass es eine weitere zentrale Einheit zwischen Nutzer, Verkäufer und Bank einsetzt. Auf diese Weise ist allerdings weiterhin der Server an jedem Zahlungsvorgang beteiligt. Auf gleiche Weise könnte man PayPal [23] nutzen, wenn Empfänger und Sender ein Konto bei PayPal besitzen und Zahlungen nicht direkt vom oder zum Bankkonto weiterleiten. [24] Eine Art Weiterentwicklung, ausgehend von dieser Verwendung eines Servers, ist das Verwenden von Token.

Nutzung von Token

Viele Ansätze für Micropayments nutzen digitale Token als eine Art Unterwährung. Diese Token werden initial meist durch einen Server, in diesem Fall Broker genannt, erstellt und werden dann für Geschäfte ausgetauscht. Ein Nutzer würde also reales Geld gegen Token beim Broker eintauschen. Mit diesen Token kann er dann bei einem Verkäufer Ware bezahlen. Der Verkäufer sammelt diese Token und löst sie später in größerer Menge beim Broker wieder gegen reales Geld ein.

Dabei gibt es verschiedene Abwandlungen dieses grundsätzlichen Ansatzes. Der Broker kann entweder in jede Transaktion einbezogen sein oder nur für das Umtauschen zuständig sein. Im zweiten Fall ist ein Double Spend für den Sender möglich. Er kann mehrfach den gleichen Token an verschiedene Empfänger ausgeben, was erst beim Einlösen durch den Empfänger auffallen wird. Bindet man den Token an den Sender, so kann er anschließend bestraft werden, dennoch kann der Verlust dadurch enorm sein. Der Token kann allerdings auch an den Empfänger gebunden sein, sodass ein Token nur an ihn ausgegeben werden kann. Ein anderer Ansatz, dem Problem zu begegnen, ist das Aufzeichnen und dem Verkäufer zugänglich machen, der bisher ausgegebenen Token. Auf diese Weise wäre die Verantwortung bei dem Verkäufer nur rechtmäßige Zahlungen anzunehmen. [25] Ein Ansatz, welchem die Idee einer Unterwährung zugrunde liegt, soll folgend kurz vorgestellt werden.

Das Milicent Protocol wurde 1995 von Mark S. Manasse veröffentlicht. Er erstellte es mit dem Wunsch nach einem guten, günstigen Zahlungsschema für digitales Bargeld. Dabei stützt sich sein Schema auf eine eigene Unterwährung (Token), die scrip genannt wird. Es sind drei Parteien involviert: Nutzer, Verkäufer und Broker. Wobei auch mehrere Broker in einem System vorkommen können. Es existieren von scrip zwei verschiedenen Arten: broker scrip und (vendor) scrip. Bevor der Nutzer einen Verkäufer bezahlen kann, muss er beim Broker broker scrip kaufen. Dieser Kauf wird in Form eines Macropayments mit normaler Währung getätigt. Als geschätzter Wert sind dabei 5-10 € genannt. Anschließend kauft er mit diesem broker scrip beim Broker verkäuferspezifischen vendor scrip. Der Wert der gekauften vendor scrips liegt im Centbereich. Mit vendor scrip kann der Nutzer einen bestimmten Händler bezahlen. Möchte der Nutzer also mit verschiedenen Verkäufern interagieren, muss er bei seinem Broker verschiedene vendor scrips kaufen. Der Händler wiederum kann seine erhaltenen vendor scrip bei seinem Broker in normale Währung eintauschen.

Der Broker kann auf verschiedene Arten vendor scrips erhalten. Er kann diese in großer Menge bei einem Verkäufer kaufen, er kann die Lizenz zum eigenständigen Erstellen beim Verkäufer kaufen oder er interagiert mit dem Broker des Verkäufers. Der vendor scrip besteht aus zwei Teilen, dem scrip body und dessen Zertifikat. Der scrip body wiederum beinhaltet zum einen Informationen zum Identifizieren, wie die ID des Verkäufers, des scrips und des Nutzers. Zum anderen umfasst es Zertifizierungsinformationen, wie Wert, Ablaufdatum/-zeit und andere Parameter. Das Zertifikat des body scrip entsteht durch Erstellen eines Hashs in Kombination mit dem sogenannten Master-scrip-secret, welches nur der Verkäufer beziehungsweise der Ersteller des scrips kennt. [26]

Bei PPay soll der Broker, als eventuelle Engstelle entlastet werden, indem die Token innerhalb des P2P Netzwerkes ohne dessen Beteiligung ausgetauscht werden können. Stattdessen ist ein Token immer an den ursprünglichen Besitzer gebunden. Er kann ihn an einen Verkäufer bei einer Transaktion ausgeben und möchte dieser Verkäufer selbst etwas kaufen, lässt er ihn durch den ersten Besitzer neu zuordnen. Der Besitzer muss jede Transaktion zu seinen Coins aufzeichnen, um Double Spend zu vermeiden. Außerdem muss er für mögliche Transaktionen online sein. [27]

Neben Token gibt es sogenannte DebtToken. Das ist eine Art übertragbare Zahlungsaufforderung, die beispielsweise auch innerhalb eines P2P Netzwerkes als Bezahlung weitergegeben werden kann. Dabei wird er vom Verkaufenden an den Nutzer überbracht. Es gibt also anfangs einen Ersteller des DebtTokens, welcher diesen an den Käufer aushändigt. Der Käufer wiederum kann selbst auch als Verkäufer agieren und auf diese Weise den DebtToken innerhalb des Netzwerkes weitergeben. Der letzte Besitzer muss schließlich den Ersteller des Token auszahlen. Wer der letzte Besitzer ist, wird durch einen Verfalls-Zeitstempel innerhalb des Tokens bestimmt. Alle Teilnehmer des Netzwerkes müssen sich anfangs mit persönlichen Daten registrieren und ein Deposit hinterlegen.

Im Protokoll von S.-M. Yen et al. [28] wird weiterhin ein Broker eingesetzt. An diesen schickt der letzte Besitzer, also der Zahlende eine Zahlungsanfrage. Der Broker prüft die Richtigkeit und verifiziert mit dem Public Key des Erstellers diesen als den Empfänger der Zahlung. Anschließend transferiert er Geld vom Konto des Einlösenden zum Konto des Erstellers. Läuft ein DebtToken ab, ohne dass jemand für ihn aufkommt, kann sich der Ersteller an den Broker wenden. Für diesen Fall erhält immer derjenige, der den DebtToken weitergegeben hat, eine signierte Bestätigung vom neuen Besitzer. Der Broker kann daher bei einem abgelaufenen DebtToken alle Peers nach dieser Bestätigung fragen. Kann einer keine Bestätigung vorzeigen, ist er der letzte Besitzer und wird bestraft als unehrlicher Peer. Dabei behält der Broker Übersicht über alle geforderten Einlösungen. Double Spend ist bei DebtToken nicht lohnenswert und aufgrund der Übersicht kann kein DebtToken mehrfach eingelöst werden. Der Ersteller, der einen DebtToken zweimal vergibt, hat keinen Vorteil, da er auch die Leistung zweimal bringen muss. Sollte ein Peer wiederholt den gleichen DebtToken erhalten haben, kann er

dennoch als letzter Besitzer erkannt werden, aufgrund eines Zeitstempels in der Bestätigung. [28]

Nutzung von Hash-Chain

PayWord [31] wurde entwickelt mit dem Ziel Public-Key-Operationen pro Zahlung zu minimieren. Das wird durch das Verwenden von Hash-Operationen, die weniger Rechenaufwand haben, erreicht. Das Set-up gestaltet sich durch den Einsatz eines Brokers sehr einfach. Sowohl der Nutzer, als auch der Verkäufer legen einen Account beim Broker an. Der Nutzer hinterlegt eine gewisse Summe Geld und erhält im Gegenzug ein Zertifikat. Mit diesem ist er nun in der Lage selbstständig Micropayments auszuführen. Dafür erstellt er eine Hash-Chain: Er wählt einen zufälligen Wert (seed) w_n und berechnet anschließend $w_i = H(w_{i+1})$ mit $i = n-1, n-2, ...0$. Aufgrund der rekursiven Anwendung der Hashfunktion H entsteht eine Kette an Hashwerten. Jeder dieser Hashwerte, bis auf w_0 und w_n, wird nun als Coin betrachtet und PayWord genannt. Eine solche Kette erstellt der Nutzer für jeden Verkäufer, mit dem er interagieren möchte.

Als erste Transaktion sendet der Nutzer sowohl w_0, als auch sein Zertifikat in einem Commitment zum entsprechenden Verkäufer. [3]Dies ist keine Bezahlung, sondern ist als Set-up Transaktion zu betrachten. Ab dann kann der Nutzer als Micropayment dem Verkäufer ein PayWord inklusive dem dazugehörigen Index mitteilen, wobei der Index höher ist, als der vorherige. Den Wert eines PayWords müssen Sender und Empfänger vereinbaren, das ist der Minimalwert einer Transaktion. Ein Vielfaches dessen kann bezahlt werden, indem Indizes übersprungen werden. Über eine bestimmte Zeitspanne sammelt der Verkäufer bei jedem seiner Kunden PayWords, wobei er nur das mit dem höchsten Index speichern muss. Am Ende beispielsweise eines Tages teilt er diese dem Broker mit, welcher sie verifiziert und die entsprechenden Beträge von den Nutzern zum Verkäufer transferiert.

Das Schema funktioniert größtenteils offline, nur zum Einlösen der Micropayments kontaktiert der Verkäufer den Broker und der Sender nur zur Rezertifizierung. Der Rechenaufwand pro Zahlung beläuft sich auf eine Hashberechnung des Verkäufers zur Überprüfung, ob das mitgeteilte PayWord Teil der Hash-Chain ist. Die Verkäufer-Nutzer-Beziehung kann variabel in ihrer Dauer und Häufigkeit sein, das Schema profitiert jedoch durch wiederholende Zahlungen. Die Beziehung zum Broker sind dauerhaft. [29–32] Basierend auf dieses Schema existieren verschiedene Weiterentwicklungen zu Multi-Dimensionalen Hash-Chains.

MicroMint [31] hingegen nutzt zwar ebenfalls Hash-Funktionen und verzichtet vollständig auf Public-Key-Verschlüsselungen, jedoch funktioniert es über Kollisionen der Hash

[3] In einem Commitment bindet der Ersteller sich durch Einsatz von Kryptografie an Informationen, ohne diese zu veröffentlichen. Dieses Commitment kann zu einem späteren Zeitpunkt durch den Ersteller geöffnet werden und die hinterlegte Information überprüft werden.

Funktion. Der Broker erzeugt Coins, indem er diese Kollisionen erzeugt beziehungsweise sucht. Er profitiert dabei durch seinen Größenvorteil, wodurch die Kosten pro Coin geringer sind. Inwiefern der Größenvorteil entsteht, wird später erläutert. Die Coins werden an die Nutzer verkauft, welche sie an Verkäufer weitergeben und letztendlich lösen die Verkäufer sie wieder beim Broker ein. Die Schwierigkeit beim Finden einer Hash-Funktion Kollision dient als Absicherung des Systems.

Eine Hash-Funktion bildet aus einem m-bit String x einen n-bit String y ab. Eine Kollision entsteht dann, wenn $h(x_1) = h(x_2) = y$und $x_1 \neq x_2$. Das ist konkret eine sogenannte 2-way-collision, da aus genau zwei Eingaben der gleiche Hash entstanden ist. Geht man davon aus, dass eine Hash-Funktion nicht vorhersagbar ist, also zufällige Ergebnisse hat, muss für eine Kollision Hashberechnungen mit $\sqrt{2^n} = 2^{n/2}$ verschiedenen x-Werten durchgeführt werden. berechnet man den Hash für c mal so viele x-Werte, wie für eine Kollision nötig sind, erhält man etwa c^2 mal so viele Kollisionen mit $1 \leq c \leq 2^{n/2}$. Das Finden der Kollisionen wird also immer effizienter. Als eine k-way-collision bezeichnet man ein Set von k verschiedenen x-Werten, die denselben Hashwert y haben. Die Anzahl an x-Werten die getestet werden müssen, um diese Kollision zu finden, ist ungefähr $2^{n(k-1)/k}$, berechnet man c-mal so viele x-Werte, für $1 \leq c \leq 2^{n/k}$ können c^k k-fach Kollisionen erwartet werden. Der Broker erstellt Coins immer für einen Monat, kann jedoch auch jederzeit eine neue Periode einführen.

Für kleine Angriffe lohnt sich der Aufwand nicht und bei Angriffen im großen Stil erkennt der Broker fremde Coins und kann eine neue Periode mit neuen Parametern beginnen. Aus diesem Grund ist auch nur der Broker in der Lage Coins vorzuproduzieren. Bei Double Spend weiß der Broker wem er diesen Coin ausstellte und kann dadurch auf den betrügenden Nutzer schließen und ihn bestrafen oder verbannen. Allgemein liefert dieses Schema keine hohe Sicherheit. Das ist den Verfassern bewusst und sie verweisen auf die geringe Größe des Betrags, welcher verloren gehen würde. [29–32]

Es gibt ausgehend von diesem Grundschema außerdem Ansätze mit Nutzer-gebundene und Verkäufer-gebundene Coins. Verallgemeinert lehnt sich der Ansatz an die Nutzung von Token an, da der Broker die Coins erstellt, welche dann als Unterwährung ausgetauscht werden.

3.2.2 Blockchainbasierend

Die bisher vorgestellten Ansätze basierten auf einer zentralen Einheit - einem Server. In diesem Unterkapitel soll zum einen untersucht werden, inwiefern diese sich auf einer Blockchain anwenden lassen. Zum anderen welche weiteren Ansätze es gibt und warum Micropayment Schemata auf der Blockchain benötigt werden.

Auch wenn die Blockchain Technologie bereits Transaktionskosten reduziert, vor allem im Vergleich zu Überweisungen im internationalen Raum, sind sie dennoch auf public

Blockchains zu hoch. Vor allem Micropayments bleiben unrentabel. Ein Ziel ist es also, die Transaktionskosten zu reduzieren. In der Blockchain-Technologie zahlt man für jede On-Chain Transaktion eine Gebühr. Diese Gebühren werden potenziell eher steigen, nämlich dann, wenn die Miner keine Coins mehr als Belohnung für den Block erhalten. Ab dann werden sie nur noch durch die Transaktionsgebühren bezahlt. [33] Die Kosten bei Ethereum, basieren darauf, wie viel Gas die Transaktion benötigt und wie hoch der Gaspreis ist.

Neben der Kostenreduktion benötigen aktuelle Blockchains auch Lösungen für Skalierung und Beschleunigung. Eine On-Chain Transaktion auf der Bitcoin Blockchain braucht etwa zehn Minuten, bevor sie in einem Block auf der Blockchain steht. Anschließend benötigt sie weitere sechs erfolgreiche Blöcke, um als bestätigt zu gelten. Um wirklich sicher zu sein, müsste ein Verkäufer etwa eine Stunde warten, bis er sicher sein kann, bezahlt worden zu sein und die Ware zu liefern. Das ist in vielen Szenarien nicht praktikabel. Auch die Skalierbarkeit ist eine Herausforderung. Um Blockchain wirklich weitläufig einsetzen zu können, braucht es Lösungen, um mehr als etwa zehn Transaktionen pro Sekunde verarbeiten zu können.

Dabei geht es nicht nur um Zahlungen, sondern auch andere allgemeine Vorgänge. Beispielsweise steigt der Speicherplatz, den die komplette Blockchain benötigt, je mehr Daten auf ihr abgelegt werden. Bei Bitcoin handelt es sich dabei um etwa zwölf GB die pro Quartal hinzukommen. [34]

Alle Ansätze, die auf eine Blockchain aufbauen, sollten die Vorteile, die eine Dezentralisierung mit sich bringt, erhalten. Es sollte also auf den Einsatz von zentralen Einheiten oder Parteien, denen man vertrauen muss, verzichtet werden. Allgemein gilt zu beachten, dass es grundsätzlich zwischen den Teilnehmern keine Vertrauensbasis gibt. Es existieren verschiedene Ansätze Micropayments auf der Blockchain umzusetzen. Einige dieser Ansätze sollen nun vorgestellt werden.

MicroBTC

Ein Ansatz, der auf Hash Chain basiert, ist MicroBTC. Es ist ein Schema, welches auf die Bitcoin Blockchain aufbaut. Als Set-up erstellt der Sender eine Hash Chain. Außerdem erzeugt er ein Deposit und veröffentlicht die Wurzel h_0 signiert in einer Commitment Transaktion. Nun erfolgt die Micro-Bezahlung, indem der Sender dem Empfänger Hash-Werte aus der Hash Chain mitteilt. Zum Einlösen dieser veröffentlicht der Verkäufer bei dem Deposit seinen höchsten Hash und erhält daraufhin den entsprechenden Betrag. Der Restbetrag wird zurück an den Sender übertragen. Als Sicherheitsmaßnahme kann eine Ablaufzeit für das Deposit eingestellt werden, nach welcher auch der Sender Zugriff auf das Deposit hat. Zuvor sollte der Zugriff nur für den Empfänger möglich sein, da sonst der Sender selbst, mithilfe eines Hashes, das Deposit leeren könnte. [33]

Auch wenn das veröffentlichte Paper die Anwendung auf der Bitcoin-Blockchain vorgesehen hatte, so ist es auch für Ethereum mit entsprechenden Smart Contracts umsetzbar. Jeder Sender baut eine Verbindung mit dem Empfänger auf, den er bezahlen möchte, indem er für ihn ein Deposit hinterlegt und eine Hash Chain berechnet. Das kann als eine Art Kanal (Channel) zwischen den Beiden angesehen werden. Im Folgenden soll näher auf die Arbeitsweise von Channels eingegangen werden.

State und Payment Channels

Viele bestehende Lösungen für Micropayments auf der Blockchain nutzen Channels. Diese Channels, werden sowohl auf der Ethereum Blockchain als State Channels, als auch bei Bitcoin als Payment Channels eingesetzt. In den Channels agieren zwei Parteien Off-Chain miteinander und nur das Ergebnis wird On-Chain verarbeitet. Dabei erzeugen sie einen Raum mit erhöhter Privatsphäre in einem Netzwerk. Sie können für Zahlungen genutzt werden und sowohl unidirektional als auch bidirektional arbeiten. Im ersten Fall können nur in eine Richtung Zahlungen erfolgen, im anderen können beide Parteien Zahlungen miteinander austauschen. Dadurch werden On-Chain Transaktionen eingespart und somit auch Transaktionskosten. Pro Channel werden unabhängig von der Menge der Statusänderungen, zwei On-Chain Transaktionen benötigt. Eine wird zum Eröffnen und eine zum Schließen des Channels genutzt.

Bei einem unidirektionalen Channel ist der Ablauf folgendermaßen. Die Eröffnungstransaktion beinhaltet das Einzahlen des Senders an eine Adresse mit einem 2-von-2-Multi-Signatur Contract. Die Auszahlung aus diesem Contract kann also nur durch eine Transaktion erfolgen, die von beiden privaten Schlüsseln signiert wurde. Der Sender kann den Empfänger Off-Chain bezahlen, indem er ihm eine von ihm unterschriebene Transaktion mit einem neuen Verhältnis schickt. Das Verhältnis (Balance) gibt an, wie viel des Deposits dem Sender und wie viel dem Empfänger aktuell zusteht. Von diesen signierten Nachrichten kann der Sender nun beliebig viele an den Empfänger senden.

Der Channel kann geschlossen werden, indem der Empfänger die einseitig unterschriebene Transaktion mit der aktuellen Balance ebenfalls signiert und an den Contract schickt. Dieser prüft die Multisignatur und veranlasst das Auszahlen des Deposits entsprechend der Balance. Dieses Auflösen durch den Empfänger muss innerhalb einer gewissen Zeit geschehen, in der das Deposit gesperrt ist.

Bei der Eröffnung des Channels erhält der Sender eine signierte Berechtigung vom Empfänger, die er nach einer Locktime einlösen kann, um sein Deposit vollständig zurückzuerhalten. Dies verhindert, dass das Deposit nicht mehr erreichbar wäre, falls der Empfänger nicht mehr reagiert. Diese Locktime kann flexibel, je nach Anwendung durch eine Blockzahl angegeben werden. Auf diese Weise kann der Sender, bis das Deposit vollständig an den Empfänger überschrieben ist, die Balance immer wieder erneuern und so den Empfänger bezahlen. In unidirektionalen Channel können nur Zahlungen in

eine Richtung erfolgen.

Ist der Empfänger gleichzeitig auch ein Sender und möchte einen anderen Empfänger bezahlen, ist der Einsatz von bidirektionalen Payment Channel sinnvoll. Ein bidirektionaler Channel wird durch die Einzahlung beider Parteien eröffnet, diese On-Chain Transaktion erzeugt erneut ein Deposit, welches eine Multisignatur zur Auszahlung benötigt. Beide Tauschen einseitig signierte Berechtigungen aus, die den jeweils anderen dazu ermächtigen, ihren eingezahlten Betrag, nach einer gewissen Zeit wieder einzufordern. Die beiden Parteien bezahlen sich nun gegenseitig, in dem sie signierte Berechtigungen mit einer veränderten Balance dem anderen geben. Ein Sicherheitsproblem bei dieser Technik besteht darin, dass einer der Teilnehmer eine veraltete Berechtigung veröffentlicht, die eine für ihn bessere Balance zeigt. Diesem Problem wird auf unterschiedliche Weise begegnet.

Eine einfache Lösung ist das Einbringen einer Nonce, welche mit jeder neuen Balance Veränderung erhöht wird. Löst nun einer der Teilnehmer eine Berechtigung ein, bekommt der Gegenspieler eine gewisse Zeit, in welcher er ebenfalls eine neuere Balance mit entsprechender Berechtigung zeigen kann. Die korrekte Balance mit der höchsten Nonce wird dann tatsächlich in einer On-Chain Transaktion umgesetzt.

Ein anderer, etwas komplexerer Ansatz beruht auf dem Einsatz von einem geheimen Wert. Beide Teilnehmer erstellen für jede Balance Aktualisierung, die sie austauschen, ein neues Geheimnis. Beim Übermitteln einer neuen Berechtigung mit einer neuen Balance, wird das alte Geheimnis dem Anderen mitgeteilt. Veröffentlicht nun jemand eine veraltete Balance, besitzt der andere das entsprechende Geheimnis zu der Berechtigung. Das ermächtigt ihn, den für den Anderen angedachten Teil, zu erhalten. Löst also ein Teilnehmer eine On-Chain Transaktion mit einer veralteten Balance aus, verliert er in diesem Fall den kompletten Teil der Balance, der für ihn gedacht war. [33–35]

Die Arbeit mit Channels kann erweitert werden, durch die Nutzung von indirekten Channels, bis hin zu Channel Netzwerken. Bei indirekten Channels verbinden sich beide Parteien mit einer dritten Instanz als Brücke und nicht mehr direkt. Der ursprüngliche Sender könnte beispielsweise einen anderen Empfänger über seinen ursprünglichen Empfänger, der einen Channel mit dem anderen hat, bezahlen. Auf diesem Konzept beruht Lightning.

Lightning ist ein Netzwerk aus Micropayment Channels basierend auf Bitcoin. Damit ein solches Netzwerk trustless, also ohne einander vertrauen zu müssen, funktioniert, setzt Lightning auf sogenannte Hash Time-Locked Contracts (HTLC). Der Sender schickt dabei seinen Betrag an einen solchen Contract, um dem Empfänger in Zukunft zu bezahlen. Es ist also eine Art Deposit. Der Empfänger hat nun ein gewisses Zeitfenster diesen Betrag wirklich zu erhalten. Dafür benötigt er seine Signatur und einen geheimen Wert, den nur der Sender kennt. Off-chain kann der Sender nun den Empfänger bezahlen,

indem er ihm das Geheimnis mitteilt. Geschieht dies nicht innerhalb des Zeitfensters, kann der Sender sein Deposit auflösen und es an sich selbst auszahlen. Möchte nun der Sender einen Dritten bezahlen, braucht er nicht direkt einen eigenen HTLC mit diesem. Es reicht, wenn der ursprüngliche Empfänger bereits einen mit diesem Dritten hat.

Der Ablauf zwischen Sender S, Empfänger E1 und dem Dritten E2 läuft folgendermaßen ab:

- S hat bereits einen Channel mit E1 und E1 einen Channel mit E2 etabliert.
- S möchte nun E2 über den Channel mit E1 bezahlen.
- E2 erstellt einen geheimen Wert und teilt S den Hash davon mit.
- S tätigt eine Zahlung an E1, wovon der Teil, der für E2 bestimmt wird, auf eine andere Adresse eingezahlt wird.
- Um von dieser Adresse eine Zahlung zu erhalten, gibt es zwei Möglichkeiten:
  - E1 kann diese bekommen, wenn er beweisen kann, dass er den entsprechenden Wert an E2 übertragen hat. Dieser Beweis wäre, dass er den geheimen Wert von E2 im Austausch erhalten hat und diesen dem HTLC zusammen mit seiner Signatur zeigt.
  - S selbst kann wieder auf die Adresse und den Betrag zugreifen, und an sich auszahlen, sollte im Zeitfenster des HTLC nicht der Austausch zwischen E1 und E2 erfolgt sein.

Auf diese Art und Weise, kann S eine Zahlung an E2 tätigen, ohne einen Channel mit ihm zu eröffnen und ohne dem Mittelmann E1 oder E2 vertrauen zu müssen. Lightning ist dadurch ein umfassendes, recht komplexes System, um in einem Netzwerk Micropayments auszutauschen. [36–40]

Es existieren ebenfalls, ein auf Channels basierende Systeme für Ethereum. Eines dieser Systeme ist Raiden. In diesem System wird eine Zahlung über einen Dritten zwischen zwei Mitgliedern des Netzwerks wie folgt vermittelt.

- Zunächst erstellt S einen geheimen Wert für einen Hash Lock, mit welchem er einen Betrag zu E1 überträgt, ohne dass dieser darauf zugreifen kann.
- E1 nutzt diesen Hash Lock ebenfalls, um den Betrag an E2 weiterzuleiten.
- E2 bestätigt die Übertragung von E1
- Daraufhin sendet S das Geheimnis (den Schlüssel) für den Hash-Lock an E2.
- Dieser kann die Zahlung von E1 erhalten, in dem er den Schlüssel an E1 weitergibt.
- Daraufhin kann E1 wiederum die Zahlung von S an sich durchführen.

Aufgrund dieser Art Vorkasse, muss S zunächst eine passende Route durch das Netz-

werk finden, bei dem alle Teilnehmer mindestens den Betrag besitzen, den S bezahlen möchte. Sollte jemand in dieser Zahlungskette beispielsweise E1, nicht reagieren, kann E2 das Geheimnis veröffentlichen und es somit allen in der Kette ermöglichen, auf den Schlüssel zuzugreifen und die Zahlungen zu realisieren. Außerdem haben auch hier die Deposit eine Auslaufzeit, nach welcher sie ohne andere Beteiligung wieder ausgezahlt werden können. [37, 41, 42]

Channels werden auf verschiedene Weise eingesetzt und genauso unterschiedlich sind auch die Umsetzungen. Channels sind eine Level 2 Skalierung, bauen also auf die darunter liegende Blockchain auf. Somit geht durch ihren Einsatz nicht die Charakteristik einer Blockchain verloren. Möglichst viele Prozesse werden dabei ausgelagert und Off-Chain erledigt. Ein Vorteil von Channels ist, die geringere Latenz, im Vergleich zu einer On-Chain Transaktion, welche erst bestätigt und in Blöcke gespeichert werden muss. Durch den Einsatz von kryptografischen Verfahren entsteht eine direkte Finalität der Off-Chain Transaktion. Durch das Auslagern muss weniger auf der Chain gespeichert werden, allerdings muss oftmals der Off-Chain Transaktionsverlauf gespeichert werden oder verschiedene Schlüssel abgelegt werden. Teilweise müssen diese Informationen für unbestimmte Zeit gespeichert werden, was vor allem bei hoher Skalierung zu einem Problem werden kann. Technologien und Netzwerke, die Channel nutzen, sind Celer, FunFair, Perun. Protokolle, die Channel verwenden, sind unter anderem Counterfactual und Sprites.

Plasma

Ein weiterer Ansatz, Transaktionen und Vorgänge von der Blockchain auszulagern, ist das Nutzen von Side Chains. Dabei wird die ursprüngliche Blockchain, beispielsweise die von Ethereum, als Root Chain, also als eine Art Anker oder Wurzel genutzt. Von dieser aus können sich verschiedene Side Chains abspalten und auf ihnen Transaktionen durchgeführt werden, ohne die Root Chain zu belasten. Diese Child Chains können auch untereinander wieder aufeinander aufbauen und dadurch eine Art Hierarchie an Chains entstehen. Das wurde ursprünglich entwickelt, um eine höhere Skalierung zu erreichen.

Plasma Chain ist eine besondere Technik, die auf Side Chains basiert und auch für Micropayments genutzt werden kann. Dieses Modell soll nun kurz vorgestellt werden. Um eine Child Chain zu erzeugen, wird ein sogenannter Plasma Smart Contract auf der Root Chain und auch auf der Child Chain erstellt. Verschiedene Operatoren werden dabei genutzt, um den Austausch zwischen beiden zu realisieren. In einem Ansatz erstellt ein Plasma Operator den Smart Contract auf der Root Chain, erzeugt die Blöcke auf der Child Chain und teilt deren Hashs regelmäßig der Root Chain mit. Dabei funktioniert die Root Chain als oberstes Gericht, welcher falsche Blöcke gemeldet werden können und entsprechend Verantwortliche bestraft werden.

Teilnehmer registrieren sich zunächst auf der Root Chain und hinterlegen ein Deposit im Smart Contract. Im Gegenzug erhalten sie den Wert als Unterwährung (Plasma Cash) der Plasma Chain. Wäre die Grundwährung Ether, würde man Plasma Ether (PETH) für die Child Chain erhalten. Sie interagieren ab da nur noch mit den Coins der Plasma Cash auf der Child Chain. Dabei hat jeder Coin eine Historie, welche aufzeichnet, wer der Besitzer ist. Wird ein Coin übertragen, wird dies vermerkt und die aktualisierte Historie an den neuen Besitzer weitergegeben. Diese kann ohne Interaktion mit der Root Chain ausgetauscht werden.

Möchte ein Teilnehmer die Child Chain verlassen, ruft er die entsprechende Funktion mit folgenden Parametern auf:

- die Coins, die er noch besitzt inklusive ihrer Historie,
- ein sogenanntes „bond“, was als Deposit funktioniert
- Gas Gebühren für die Transaktion auf der Root Chain

Anschließend erfolgt eine Challenge Periode, in welcher jeder Teilnehmer überprüfen kann, ob die Historie der Coins korrekt ist und ob er wirklich (noch) der Besitzer ist. Wird der Teilnehmer beim Betrügen erwischt, verliert er neben den Coins auch das bond. Übersteht er die Challenge Periode, erhält er auf der Root Chain den der Coins entsprechenden Betrag an Grundwährung zurück. Eine problematische Situation entsteht, wenn ein Betrug, erst nach der Challenge Periode auffällt und der Betrüger mehr Grundwährung aus dem Smart Contract erhalten hat, als ihm zustand. Das würde dazu führen, dass für die restlichen ehrlichen Teilnehmer nicht mehr ausreichend Grundwährung hinterlegt ist. Dann würden alle Teilnehmer gleichzeitig die Child Chain verlassen wollen, also ihre Unterwährung mit dem Smart Contract zurücktauschen wollen. Das wiederum würde zu einer hohen Anzahl an Transaktionen in der Root Chain führen, was eventuell andere Transaktionen verdrängt.

Es haben also alle Teilnehmer einen Anreiz die Challenge Periode zur Überprüfung zu nutzen. Für weitere Probleme, wie die erhöhte Speichernutzung, haben die Entwickler von Plasma bereits verschiedene Ansätze gefunden. [43–45]

Libra

Die Libra Blockchain ist von der Libra Association geplant und wird als dezentralisierte, programmierbare Datenbank beschrieben. Dabei soll sie in der Lage sein, eine Kryptowährung zu transferieren und dadurch eine Finanzinfrastruktur darzustellen. Das Whitepaper wurde am 18. Juni dieses Jahres veröffentlicht und auch ein Testnetz wurde bereits aufgesetzt. Das Hauptnetzwerk (Mainnet) soll Anfang 2020 zur Verfügung stehen. Die Libra Association ist eine gemeinnützige Organisation, welche von namhaften Firmen gegründet wurde, unter anderem von Mastercard, PayPal, Uber und Vodafone. Auch Calibra, eine Tochterfirma von Facebook gehört zu den Gründungsmitgliedern

und wird aktuell als Hauptverantwortlicher genannt. [46, 47] Was Libra neben seiner aktuellen Präsenz und namhaften Unterstützern besonders macht und wie es mit Micropayments umgeht, soll hier kurz beschrieben werden.

Libra unterscheidet sich von traditionellen Blockchains, wie die von Bitcoin und Ethereum in mehreren Aspekten. Zum einen ist sie nicht vollständig dezentralisiert, da die Libra Association als Vertrauensbasis gilt. Nur Mitglieder dieser Organisation sind aktuell in der Lage Validator Nodes zu haben und sich am Konsensus-Protokoll zu beteiligen. Zum anderen setzte Libra auf eine Variante des HotStuff Konsensus Protokolls, das sogenannte LibraBFT. Der Algorithmus basiert auf Proof-of-Stake und der Konsensus wird in drei Kommunikationsrunden gefunden. Im Paper sind für die aktuelle Umsetzung dafür hundert Validatoren zuständig, aber es ist geplant bis zu tausend dieser im Netzwerk zu haben. Um die Performance des Netzwerks zu erhalten, werden Anforderungen an Validatoren gestellt, unter anderem bezüglich ihrer Rechenkraft und Bandbreite. Bezüglich des Gebührenmodells basiert es wie Ethereum auf dem Gas-Modell und auch wie Ethereum basiert es auf einem Konto Modell. Die Gebühren sind gering gehalten für normale Transaktionen und sollen vor allem Spam und DoS-Attacken vermeiden.

Der Wert des Libra Coin ist durch andere unterschiedliche Währungen gedeckt und daher ein Stable Coin. Der Erzeuger eines Coins hat also zunächst eine Reserve erstellt, woraufhin der Coin erzeugt wurde. Entzieht er dem System seine Reserve wird auch der Coin wieder zerstört. Durch dieses System fluktuiert der Wert der Währung nur sehr leicht im Wert - basierend auf der Währung, die ihn deckt. Der LibraCoin ist also nicht als Investment, sondern als tatsächliches Zahlungsmittel gedacht.

Als Programmiersprache wird Libra Move verwenden, welche eigens dafür entwickelt wurde. Es priorisiert die Sicherheit und Korrektheit der Smart Contracts, die hier Module genannt werden. Dadurch ist die Funktionalität eingeschränkt und die Sprache auch nicht turing-vollständig. Unter anderem ist auch keine dynamische Bindung von Methodenaufrufe oder Unterprogammen möglich. Die Move Virtual Machine führt die entsprechenden Funktionen aus. [48]

Die Blockchain von Libra soll vor allem durch Performance überzeugen. Es sollen alle zehn Sekunden Blöcke erstellt werden und somit tausend Transaktionen pro Sekunde ermöglicht werden. Um eine Transaktion auf der Blockchain auszuführen reicht der Nutzer diese ein. Ein Validator führt diese Anfrage an die Blockchain aus und nimmt die Änderung entsprechend der Transaktion vor. Ein Anführer der Validatoren (Leader), schlägt diese vorbereiteten Transaktionen den anderen Validatoren vor und mithilfe des Konsensus Protokolls wird entschieden, ob der Status der Blockchain entsprechend geändert wird. Würden nun zwei Milliarden Menschen die aktuell 27 Validatoren nutzen wollen, würde das die Performance der Blockchain beeinträchtigen. In ihrem Paper nennt Libra die Möglichkeit, Off-Chain Technologien einzubringen, unter anderem Channel, um tausend Transaktionen pro Sekunde nicht zu übersteigen. [49]

Libra möchte effizient Micropayments behandeln. In der Anwendung sollen dadurch Autoren und Verfasser von Posts, Artikeln und Videos auf sozialen Plattformen wie Facebook bezahlt werden. Aufgrund der geringen Transaktionskosten bezeichnen sie Libra bereits als fähig, Micropayments durchführen zu können. [50, 51] Welche Technologien neben Channels Libra dafür anbieten wird und auch inwiefern andere Micropayment Ansätze mit Move programmierbar sind, wird sich nach Start des Mainnets zeigen. Inwieweit diese Technologien tatsächlich für die Blockchain von Libra nötig sind, wird sich erst danach in den verschiedenen Anwendungsszenarien zeigen.

3.2.3 Potenzial und Herausforderungen

Mit den vermehrten Einsatzmöglichkeiten von Micropayments stehen auch Entwicklung und Weiterentwicklung der Micropayment Schemata im Fokus. Entsprechend viele unterschiedliche Ansätze wurden veröffentlicht und einige in diesem Kapitel vorgestellt. Neben grundlegenden Ideen, die auf zentrale Systeme bauen, wurden auch auf Blockchain basierende Systeme erklärt. Das Ziel viele kleine Transaktionen zu wenigen Großen zusammenzufassen, wurde auf verschiedene Arten erreicht. Doch weitere wichtige Aspekte wie Kosten, Sicherheit und Akzeptanz wurden nicht immer vollumfänglich behandelt. Einige Erkenntnisse sollen folgend zusammengefasst werden.

Die vorgestellten Systeme können, wenn sie eingesetzt werden, vermutlich die Kosten pro Zahlung verringern. Allerdings ist ein System nur bedingt rentabel, wenn die Kosten zum Einsetzen des Systems enorm sind. Eine Umstrukturierung eines bestehenden Systems ist immer mit Kosten verbunden und in manchen Fällen lohnt sich, trotz Kostenreduktion in der Anwendung, ein Wechsel nicht. Ein Beispiel dafür ist die Nutzung eines Token. Dieser benötigt ein funktionierendes Konzept, ein Broker muss eingerichtet und betrieben werden und die Einsparungen sind je nach System überschaubar. Bei der Umsetzung von Micropayment Systemen auf der Ethereum Blockchain müssen Smart Contracts entwickelt werden und auf die Blockchain geschrieben werden. Beides ist mit Kosten verbunden. Auch auf Seite der Sender kostet die Umsetzung und Programmierung der Konzepte Zeit und Geld. Dennoch ist verallgemeinert zu sagen, dass die meisten hier behandelten Lösungen, die auf Blockchain basieren, Potenzial haben Kosten zu senken.

Der Aspekt der Sicherheit wird von den Schemata unterschiedlich behandelt. In einigen wird die Absicherung der zentralen Einheit überlassen und ist nicht Teil der Betrachtung. Andere verzichten zumindest auf die Absicherung jedes einzelnen Micropayments, aufgrund des geringen Wertes. Dennoch kann jede Sicherheitslücke kritisch werden, sobald sie im großen Stil ausgenutzt werden kann. Ist dies möglich, kann ein Angriff trotz Bestrafung durch Verlust einer Kaution, rentabel werden. Es sind also andere Sicherheitsprinzipien nötig, um die verschiedenen Attacken erfolgreich abwehren zu können. Viele Sicherheitsprinzipien wirken anderen Aspekten, wie Akzeptanz der Parteien und

Performance des Systems entgegen. Dennoch gilt der Sicherheitsaspekt als existenziell. Vor allem Systeme auf der Blockchain müssen hier eigene Ideen entwickeln, um der Gefahr von Double Spend, Overspend und Replay zu begegnen. Auch der Sender muss vor der Waiting Merchant Attacke geschützt werden.

Eine Bestrafung auf der Blockchain muss vorbereitet werden, dies geschieht häufig durch das Hinterlegen eines Deposits. Wie hoch das Deposit ist, wird von Fall zu Fall unterschieden und kalkuliert werden. Der Verlust des Deposits soll den Angriff dabei unrentabel machen, weshalb zunächst berechnet werden muss, wie viel ein böswilliger Sender mit einer Attacke erreichen kann. Daraus leitet sich die Höhe des Deposits ab. Ist der Betrag allerdings zu hoch, sinkt der Wille des Senders dieses System zu nutzen. Außerdem könnte es nicht ausreichen, Angriffe unrentabel zu machen, wenn ein Angreifer eine andere Motivation hat. Diese Motivation könnte sein, einen Konkurrenten aus dem System zu entfernen oder zu schädigen. Es müssen also andere Schutzmechanismen gesucht und eingesetzt werden.

Die Akzeptanz von Sender und Empfänger ist abhängig von:

- Der Eintrittsschwelle, also dem Aufwand, um am System teilnehmen zu können. Dieser entsteht zum einen durch das Hinterlegen eines Deposits ergibt und zum anderen, wenn eine Registrierung oder eine Installation nötig ist.
- Der Höhe des Risikos durch das System Verlust zu machen. Das kann geschehen, weil man Opfer eines Angriffes wurde oder weil ein System fehlerhaft oder unzureichend ist. Ein System sollte beispielsweise eine abgesicherte Auszahlung des hinterlegten Deposit ermöglichen.
- Der Höhe des Vorteils durch Einsetzen des Systems. Der Vorteil kann je nach Einsatz variieren. Ein Vorteil sowohl für Sender als auch für Empfänger kann eine geringere Latenz und damit eine schnellere Finalität einer Transaktion sein. Denn umso schneller die Übertragung eines Wertes final ist, umso zeitnaher kann auch die Leistung oder die Ware anschließend erbracht werden. Abschließend sollen die vorgestellten Systeme aufgrund dieser Aspekte verglichen werden.
- Die Kosten und Anforderungen durch Einsatz des Systems. Einige Systeme setzen beispielsweise Public-Key-Verschlüsselungen ein, welche rechenintensiv sind oder erfordern das Ablegen einer Historie, was Speicherplatz benötigt.

Eine schnelle Abwicklung der Zahlungen kann durch zu viele Interaktionen beim Austausch von Zahlungen eingeschränkt werden. Neben einer Kostensenkung pro Zahlung sollte daher auch die Latenz bei den Micropayments gering bleiben und eine Zahlung schnell Finalität erreichen. Bei den meisten vorgestellten Schemata ist dies der Fall.

Der Fakt, dass Channels sowohl für Ethereum, als auch auf Bitcoin umgesetzt wurde, zeigt, wie viel Potenzial in Channels gesehen wird. Die Grundstruktur des Schemas ist simpel und die On-Chain Transaktionen können stark reduziert werden. Dies ist aller-

dings abhängig von der Dauer der Sender-Empfänger Beziehung ist. In einem Netzwerk von Channels werden daher Vermittler genutzt, damit keine neue Verbindung initiiert werden müssen. Nur das Speichern des Transaktionsverlaufs mindert die Skalierbarkeit. Die Performance ist als sehr gut einzuschätzen, da schnell Finalität erreicht werden kann. Auch Plasma ist ein vielversprechender Ansatz für Micropayments. Wie viel Aufwand der Einsatz dieses Systems mit sich bringt, ist allerdings schwer abschätzbar. Für eine Einschätzung von Libra ist diese Blockchain zu jung, es ist aber zu erwarten, dass vor allem Wert auf Sicherheit und Performance gelegt werden wird.

Eine weitere Art von Micropayments sind probabilistische Zahlungen. Diese sollen im folgenden Kapitel vorgestellt werden.

4 Probabilistische Zahlungen

4.1 Grundlagen

Probabilistische Zahlungen (Probabilistic Payments) sind ein eigener Ansatz, um den Herausforderungen von Micropayments zu begegnen. Geprägt wurde der Begriff 1996/97 von Wheeler und Rivest. Grundlegend beruht die Idee darauf, Micropayments mit Losen/Tickets zu realisieren. Diese Tickets können mit einer gewissen Wahrscheinlichkeit p ein Gewinn sein und bewirken damit eine große Zahlung. Diese sogenannte Makrozahlung hat dabei einen Wert X. Das Gegenereignis ist eine Niete und tritt mit einer Wahrscheinlichkeit von $1 - p$ ein. In diesem Fall wird keine Zahlung durchgeführt, weshalb es auch Nullpayment genannt wird. Auf lange Sicht gesehen und aufgrund des Gesetzes der großen Zahlen hat jedes Ticket einen Erwartungswert von $p * V$. Da nur jedes p-te Ticket zu einer wirklichen Zahlung führt, werden die Transaktionskosten theoretisch um ein p-faches reduziert. Die Zahlungen, deren erwarteter Wert ein Kleinstbetrag ist, werden also durch eine entsprechende Lotterie für größere Zahlungen ersetzt und damit in einer Makrozahlung zusammengefasst. [52] Dieser grundlegende Ansatz wurde von verschiedenen Personen aufgefasst und darauf aufbauende Schemata erstellt.

In den meisten Schemata ist dem eigentlichen Tickeaustausch aus Sicherheitsgründen ein Set-up vorangestellt. Dabei „verbindet“ sich der Sender mit dem Empfänger. Dieses Set-up sollte kostenarm sein, also der Aufwand für das Aufbauen der Beziehung relativ gering. Nur dann ist eine Geschäftsbeziehung, die nur wenige Mikrozahlungen beinhaltet ökonomisch. Dadurch bleibt das Versenden einer beliebigen Menge an Zahlungen an eine beliebige Menge von Empfängern effizient. [53]

Dennoch muss insgesamt in einem System regelmäßig Ticketaustausch stattfinden, damit das Gesetz der großen Zahlen gilt. Das bewirkt, dass ein Konsument nicht über- oder unterbezahlt und auch ein Verkäufer entsprechend bezahlt wird. Da probabilistische Micropayments ihre Kostenminimierung durch Effizienzerhöhung über mehrere Zahlungen hinweg erreichen, benötigt es viele Zahlungen in dem System, wo sie eingesetzt werden. [54] Ein Ansatz sollte also grundlegend mit einer hohen Kapazität einhergehen.

Neben probabilistischen Zahlungen gibt es auch die sogenannte probabilistische Verifikation. Dieser Ansatz wird auch Polling genannt und basiert darauf, dass die zentrale Einheit des Systems nur noch einen bestimmten Prozentsatz der Transaktionen mitbekommt und daraus die wirkliche Höhe der Transaktionen abschätzen kann. Auch wenn der Fokus in diesem Kapitel auf probabilistische Zahlungen gesetzt werden soll, wird dieses System kurz vorgestellt. Wie auch im vorherigen Kapitel, werden zunächst zentrale und anschließend dezentrale Ansätze erklärt.

4.2 Existierende Lösungen - zentral

In diesem Kapitel sollen bestehende zentrale Lösungen für probabilistische Zahlungen vorgestellt werden. Dabei wird der Ablauf kurz erläutert, der Umgang mit Sicherheitsanforderungen erläutert und ein kurzes Fazit gezogen.

4.2.1 Micropayments Revisited

In ihrem Paper Micropayments Revisited [58] stellen die Autoren drei verschiedene Variationen ihres Schemas vor. Es gibt drei Parteien - einen Sender, einen Empfänger und eine zentrale Bank. Als Set-up erzeugen Sender und Empfänger Schlüsselpaare. Der darauf folgende Ablauf variiert je nach Ansatz.

Beim ersten Ansatz, MR1, erfolgt die Bezahlung für eine Information in Form eines sogenannten Checks. Dieser Check C besteht aus der Signatur des Senders über die Information I, sodass $C = Sig_S(I)$. Wenn der Empfänger diesen Check erhalten hat, signiert er ihn und kann mithilfe einer festgelegten öffentlichen Hash-Funktion $F()$ überprüfen, ob dieser Check ein Gewinn ist. Wenn $F(Sig_E(C)) < s$, mit s als Gewinnwahrscheinlichkeit, dann zieht dieser Check eine Zahlung nach sich. In dem Fall sendet der Empfänger $Sig_E(C)$ zu der Bank, welche einen Betrag von $1/s$ vom Konto des Senders auf das Konto des Empfängers überträgt. Der Empfänger oder die Bank können den Sender im Fall eines Gewinnes darüber informieren.

Das große Sicherheitsproblem eines Overspends (Kontoüberziehung) durch den Sender wird bewusst der Bank überlassen und ist nicht Thema des Papers. Es ist abgesichert, dass sich nicht zwei der drei Parteien zusammenschließen können, um die dritte zu betrügen. Damit ist es ein solides Grundschema, welches mit wenig Interaktionen funktioniert. Es ist vorstellbar, die Bank durch einen Smart Contract zu ersetzen, um das Schema auf einer Blockchain anwenden zu können. In diesem Fall müsste aber noch eine Lösung für das Double Spend Problem gefunden werden.

Der zweite Ansatz (MR2) begegnet dem (psychologischen) Problem von zu vielen Bezahlungen durch den Sender. Dabei soll der Sender insofern geschützt werden, dass er nie mehr zahlen muss, als er genutzt hat. Stattdessen trägt die Bank das Risiko, welches bei ihr aufgrund der höheren Anzahl an Zahlungen geringer ist. Sie gilt als Puffer und kann individuelle Disbalancen über die hohe Anzahl an Konten ausgleichen.

Dafür wird eine Serial-Nummer in jeden Check integriert und entsprechend mit jedem Check des Senders erhöht. Soll ein Sender unglücklicherweise unverhältnismäßig viel bezahlen, gleicht die Bank die Differenz zum tatsächlichen Verbrauch aus. Dafür nutzt sie Sender, welche unverhältnismäßig wenig bezahlen müssen und wirkt so Ungerechtigkeiten entgegen.

Dieser Gedanke erinnert an einen Coupon-basierenden Ansatz, nur dass es zufällig ist, wann der Coupon eingetauscht wird. Die Bank kann aus den Daten Statistiken erstellen und Sender oder Empfänger mit zu hohen Anzahl an zu einer Zahlung führenden Tickets verbannen. MR2 lässt sich wesentlich schwieriger dezentral umsetzen, da ein Smart Contract keine Zahlungen (unbegrenzt) ausgleichen kann. Auch der Einsatz einer Blacklist[4] gestaltet sich in einem dezentralen anonymen System schwierig. [55, 56]

Der letzte Ansatz MR3 erweitert die vorherigen Ansätze und arbeitet mit Listen, in denen die Checks abgelegt werden. Er erinnert noch deutlicher an ein auf Coupon basierendes System. Für einen gewissen Zeitraum sammelt der Empfänger alle Checks in n Listen $L_1, ..., L_n$. Die Summe der Checks in Liste L_i sei V_i und die Summe aller V_i sei V. Nach diesem Zeitraum erstellt der Empfänger die Commitment C_i zur Liste L_i und V_i und sendet alle Commitments zur Bank. Die Bank prüft den Zeitraum und wählt Indizes zu denen sie ein Öffnen der Commitments möchte. Nachdem sie diese erhalten hat, überträgt sie den Betrag V an den Empfänger und belastet die Sender, deren Checks in den Listen entsprechend vorkommen.

Dieser Ansatz verliert an Sicherheit, da nicht alle Checks geprüft werden und der Empfänger Sendern schaden könnte. Außerdem ist der Speicheraufwand bei dem Empfänger größer, um die Transaktionen zur Bank zu minimieren. Außerdem lässt sich dieser Ansatz nur schwer in einem dezentralen Netzwerk umsetzen. [57, 58]

Zusammenfassend decken die drei Ansätze verschiedene Ideen ab und liefern konkrete Abläufe dafür. Nur einer von ihnen würde sich dezentral umsetzen lassen, doch eventuell können Ideen der anderen auf spätere Konzepte umgesetzt werden. Ein weiterführendes System ist Peppercoin, welches als Nächstes kurz vorgestellt werden soll.

4.2.2 Peppercoin

Peppercoin Micropayments [59] baut auf die zuvor vorgestellten Micropayments Revisited Schemata auf. Es wird erneut dem Problem begegnet, dass ein Sender nicht zu viel zahlt. Peppercoin verspricht also dem Nutzer, dass ihm nicht mehr berechnet wird, als er tatsächlich genutzt hat. Erneut funktioniert hier die Bank als Puffer, der Ansatz basiert allerdings nicht auf einer Serial-Nummer. Stattdessen nutzt er eine universelle Aggregationsmethode. Dabei werden Zahlungen über verschiedene Konsumenten, Händler und Zahlungsdienstleister aggregiert. Ein Sender verschickt ein Ticket an den Empfänger, welcher dieses überprüft und testet, ob es zu einer Makrozahlung führt. Ob ein Ticket ein Gewinn ist, wird erneut unter Nutzung einer Signatur über das Ticket überprüft.

Die Absicherung erfolgt dadurch, dass im Ticket zu einem entsprechenden Zeitstem-

[4] Teilnehmer können aufgrund von Fehlverhalten in einer öffentlichen Blacklist genannt werden und so von den restlichen Teilnehmern gemieden werden. Das Gegenteil ist eine Whitelist, in welcher vertrauenswürdige Teilnehmer genannt werden.

pel der kumulative Gesamtwert der tatsächlichen Ausgaben des Senders angegeben wird. Ist das Ticket nun ein Gewinn und wird vom Empfänger an seine Bank weitergeleitet, so sieht diese am angegebenen Gesamtwert, wie viel sie dem Sender tatsächlich berechnen darf. Die Differenz zwischen dem Wert des Makropayments und des kumulativen Wertes gleicht die Bank aus. Sie dient also als Puffer zwischen den Ausgaben des Verbrauchers und den Einnahmen des Händlers. [59, 60]

Da dieser Ansatz nur ein Aufbau ist, kann er unterschiedlich angewendet werden. Seine Vorteile sind die gute Skalierbarkeit, auch da es non-interaktiv funktioniert. Da die Bank aufgrund ihrer Funktion als Puffer eine zentrale Position hat, ist Peppercoin nicht direkt auf dezentrale Systeme anwendbar. Sie kann jedoch aufgrund ihrer universellen Aggregation eventuell als Inspiration für eine Pool-Lösung genutzt werden.

4.2.3 Coin Flipping

Der Ansatz „Efficient Coin Flipping“ [61] vereint zwei Ideen miteinander. Zum einen bedient es sich der Technologie einer Hash-Chain, zum anderen ist es eine probabilistische Zahlungsweise. Es interagieren erneut die drei Parteien - Empfänger, Sender und eine Bank miteinander. Sender und Empfänger verfügen über je ein Schlüsselpaar. Das Set-up läuft folgendermaßen ab:

- Der Empfänger wählt eine Zufallszahl x und berechnet eine Chain mit $y = f(f(f...(x)))$.
- Der Empfänger schickt die Wurzel, also y und einen Zero Knowledge Proof [5] über x zum Sender.
- Der Sender wählt ebenfalls eine Zufallszahl x' und berechnet eine eigene Chain $y' = f(f(f...(x')))$.
- Der Sender erstellt und signiert eine Nachricht (y, y') und schickt diese sowie einen Zero Knowledge Proof über x' zum Empfänger.

Das gegenseitige Schicken eines Zero Knowldege Proofs verhindert, dass $y' = y$. Denn dies würde die Zufälligkeit des Münzwurfes eliminieren. Nachdem der Empfänger alles auf Richtigkeit überprüft hat, können die beiden Parteien Micropayments wie folgt austauschen.

Wie auch im Hash-Chain Ansatz erfolgt die Bezahlung durch das Aufdecken des nächsten pre-images der Chain. Der Sender beginnt und schickt sein pre-image dem Empfänger, woraufhin dieser seins wiederum dem Sender zeigt. Die beiden pre-images werden verknüpft (XOR) und das Ergebnis zeigt an, ob es ein Gewinn oder eine Niete ist. Welche Struktur das Ergebnis für einen Gewinn haben muss, kann vorher individuell festgelegt werden. Je nach vereinbarter Wahrscheinlichkeit kann die Anzahl der letzten

[5] Zero Knowledge Proofs sind eine Art Commitment, bei dem der Ersteller beweisen kann, eine Information zu besitzen, ohne sie dem anderen mitzuteilen.

Stellen, die auf 0 geprüft werden, variieren. Die Bank benötigt die beiden pre-images, die Root der Chains und wird dann alles auf Richtigkeit und auf Gewinn prüfen. Damit die Zahlung als korrekt gilt, muss die Wurzel auch mit den vorher mitgeteilten Zero Knowledge Proof übereinstimmen.

Der Ein- und Auszahlungsprozess, sowie die Absicherung gegen Overspending wird erneut der Bank überlassen und ist nicht Teil des Papers. Der Aufwand im Set-up Prozess ist je nach Größe der Chain recht hoch. Da eine direkte Bindung zwischen Empfänger und Sender entsteht, ist die Skalierung auf mehrere Hunderte bis Tausende Teilnehmer nur bedingt möglich. In diesem Fall würde auch der Speicheraufwand für die Menge an Chains sehr hoch werden. Der Ticketaustausch ist relativ kostenarm, da hier keine Signaturen oder Hashes unbedingt benötigt werden. Nur zum Überprüfen der Richtigkeit des erhaltenen pre-images kann der Empfänger die Einwegfunktion selbst noch einmal durchführen. Eine grundlegende Sicherheit ist gegeben, da weder Sender noch Empfänger das Ergebnis des nächsten Münzwurfes voraussagen können. [61, 62]

Zusammenfassend ist es ein solider Ansatz, der es schafft Hash-Chain und probabilistische Payments zu vereinen. Aufgrund des relativ hohen Aufwands während des Set-ups und die Bindung zwischen Sender und Empfänger sind die Skalierungsmöglichkeiten allerdings eingeschränkt.

4.2.4 Polling

Wie bereits in der Einführung beschrieben, handelt es sich beim probabilistischen Polling nicht wirklich um probabilistische Zahlungen. In manchen Quellen wird es auch probabilistisches Audit genannt, also eine zufallsgesteuerte Überwachung. Um auch dieses Gebiet aufzuzeigen, welches Inspiration für weiterführende Ideen bieten kann, soll ein interessanter Ansatz kurz vorgestellt werden. Die entscheidende Partei in diesem Ansatz ist die Bank. Diese würde im Normalfall in jeder Transaktion zwischen den Mitgliedern des Systems involviert sein. Probabilistic Polling möchte den Aufwand der Bank reduzieren, indem es die Häufigkeit, mit der eine Bank Kenntnis von einer Transaktion nimmt, verringert.

Zunächst meldet sich der Nutzer initial bei der Bank mit seinem Konto an und erhält ein Zertifikat mit einem Betrag, den er maximal ausgeben darf. Möchte er nun einen Empfänger bezahlen, benötigt es eine initiale Transaktion zwischen den beiden Parteien, welche in jedem Fall von der Bank zur Kenntnis genommen wird. Nach dieser Initialisierung können Sender und Empfänger nach beliebigem System (Mikro-)Zahlungen ausführen. Wenn die Summe der Ausgaben des Senders seinen maximalen Wert erreicht hat, veröffentlicht der Empfänger alle Zahlungen an die Bank, welche den entsprechenden Betrag transferiert.

Das probabilistische Polling wird während der (Mikro-)Zahlungen eingesetzt, um Over-

spending einzudämmen. Dabei wählt die Bank eine gewisse Wahrscheinlichkeit, mit welcher ihr Zahlungen zwischen initialer und letzter Transaktion angezeigt werden. Entsprechend leitet der Empfänger eine gewisse Anzahl an (Mikro-)Zahlungen an die Bank weiter. Diese bekommt dadurch einen Zwischenstand und kann abschätzen, ob ein Sender mehr ausgibt, als er hinterlegt hat. Die Höhe der Wahrscheinlichkeit, mit der die Bank eine Zahlung sieht, kann beispielsweise von der Höhe der Transaktion abhängig gemacht werden. Da der Empfänger bei einem Overspending Verlust machen würde, hat er ebenfalls Interesse daran Transaktionen mit höherem Wert direkt der Bank mitzuteilen. [61]

Das hybride System soll die Kosten kleiner Zahlungen reduzieren, aber die Sicherheit bei großen Zahlungen erhalten. Das System schränkt den Nutzer, außer im Maximalwert, nicht ein. Er kann mit dem ausgestellten Zertifikat mit beliebig vielen Verkäufern arbeiten. Die Verkäufer müssen je nach Anzahl an Transaktionen und Kunden mehr oder weniger viele Transaktionen an die Bank weiterleiten. Und abhängig davon, wie viel die Bank riskieren möchte, muss sie Transaktionen verarbeiten. Die limitierende Partei ist die zentrale Bank, da sie je nach Größe des Netzwerkes am meisten Transaktionen verarbeiten und prüfen muss. Auch ihr Speicherbedarf ist am höchsten. [63] Zusammenfassend ist es ein guter Aufbau für verschiedene Ansätze, welche das Overspending Problem der Bank überlassen haben. Es ist jedoch aufgrund der starken Zentralisierung vermutlich nicht auf der Blockchain umsetzbar.

4.3 Existierende Lösungen - dezentral

4.3.1 Ansätze nach Pass und Shelat

In ihrem Dokument „Micropayments for Decentralized Currencies“ [52] von 2016 erläutern Pass und Shelat ihre Ansätze für auf Kryptowährung basierende Micropayments. In der Umsetzung konzentrieren sie sich auf die Bitcoin Blockchain. Ihre drei Ansätze MICROPAY1, 2 und 3 geben einen Überblick, wie ein Schema für probabilistische Micropayments aussehen kann. Allgemein können auch die meisten dezentralen Schemata in eine Vorbereitung (Set-up), eine Ticketerstellung/-übertragung und ein Einlösen des Tickets eingeteilt werden. Die drei Ansätze sollen nun im Einzelnen vorgestellt werden.

In MICROPAY1 werden in der Vorbereitung und dem Austausch die folgende Schritte durchlaufen:

- Der Sender erzeugt zwei Schlüsselpaare, eines für ein Depositkonto mit der Adresse a^{esc} und eines für sein Strafkonto mit der Adresse a^{pen}. Auf das Deposit überträgt er einen gewissen Betrag und auf das Strafkonto ebenfalls. Der Betrag des Strafkonto sollte ein Vielfaches des Betrags des Deposits sein.

- Der Empfänger erstellt eine Zufallszahl $r_1 \leftarrow \{0,1\}^{128}$ und schreibt diese mithilfe eines geheimen Seed s in ein Commitment $c \leftarrow Com(r_1; s)$. Der Empfänger erzeugt außerdem eine neue Adresse a_2, an welche eine Makrozahlung ausgezahlt werden soll. Er schickt c und a_2 zum Sender.
- Der nächste Schritt ist die Ticketerstellung und Übertragung, welche in MICROPAY1 sehr simpel ist. Der Sender wählt ebenfalls eine Zufallszahl r_2 und erzeugt eine Signatur σ auf c, r_2, a_2 und sendet r_2 und σ zum Empfänger.
- Der Empfänger verifiziert die Signatur und prüft, ob es ein Gewinn ist. Dafür berechnet er die XOR-Verknüpfung von $r_1$und r_2. Hat das Ergebnis eine zuvor festgelegte Struktur, gilt das Ticket als Gewinn.

Beide können die Gewinnwahrscheinlichkeit nicht beeinflussen, da beide zum Ergebnis beitragen, ohne den Beitrag des anderen zu kennen. Der Empfänger wählt unabhängig seine Zufallszahl und bindet sich durch das Commitment an sie. Der Sender kennt daher diese Zahl nicht und erzeugt seine Zufallszahl ebenfalls unabhängig davon. Der Empfänger kennt erst nach der Übertragung die beiden Zufallszahlen und kann zu diesem Zeitpunkt seine Zahl nicht mehr ändern.

Dieses Schema kann an unterschiedliche Szenarien angepasst werden, so können beispielsweise die Gewinnhöhe und die Gewinnwahrscheinlichkeit für jedes System neu definiert werden. Trotz oder gerade wegen seiner Einfachheit zeigt das Schema verschiedene Herausforderungen eines dezentralen probabilistischen Micropayments auf. Die im Paper genannten Angriffsszenarien werden folgend kurz erläutert:

- Bei einer Double Spend Attacke wird ein Wert mehrfach ausgegeben, sodass mindestens eine Übertragung nicht gedeckt ist. Dabei können verschiedene Werte gemeint sein. Zum einen kann ein Sender dasselbe Ticket, von dem er nach der ersten Übertragung weiß, dass es kein Gewinn ist, mehrfach ausgegeben. Der Empfänger hingegen könnte ein Winning Ticket mehrfach einlösen wollen.
- Die Overspend Attacke beschreibt das Überziehen des hinterlegten Deposits durch den Sender. Das bedeutet, er gibt mehr aus, als er besitzt.
- Bei einer Front-Running Attacke versucht der Sender, vor der Auszahlung an den Empfänger, selbst eine Auszahlung von dem Konto zu bewirken.

Den Angriffen von der Seite des Senders soll durch das Strafkonto begegnet werden. Das Risiko mehr zu verlieren, als man durch eine Attacke erhalten würde, soll die Angriffe verhindern.

In diesem Schema wird bei Double Spending, also bei Präsentation von zwei Winning Tickets, der Wert im Strafkonto an eine invalide Adresse geschickt und dadurch „verbrannt“. Dadurch erhält auch der Empfänger keinen Vorteil durch das Aufzeigen eines Double Spends. Denn sonst könnte dieser mit dem Einlösen seines Winning Tickets warten, bis er eine Transaktion für das gleiche Konto im Netzwerk sieht und damit ein

Double Spend erzwingen. Diese Verhaltensweise wird Waiting Merchant genannt. Das Problem entsteht auch durch den Ansatz, dass der Sender nicht erfährt, ob er gerade ein Winning Ticket versendet hat oder ob er gefahrenfrei weitere Tickets für sein Konto ausstellen kann.

Eine weitere Herausforderung, die außerhalb des eigentlichen Zahlungsverfahrens liegt, ist das Auszahlen des Deposits beziehungsweise des Strafkontos. Dem Sender soll es ermöglicht werden, seine Konten aufzulösen und den hinterlegten Wert wieder zurückzuerhalten. MICROPAY1 behandelt das Auszahlen (withdraw) des Strafkontos nicht im Detail. Es wird der Einsatz einer „locktime“ angesprochen, also einer Zeitspanne, in welcher es dem Sender nicht möglich ist, auf sein Konto selbst zu zugreifen.

Dieses Schema beschreibt gut die Grundstruktur der meisten existierenden Ansätze für dezentralisierte probabilistische Micropayments. Diese Lösung ist allerdings nur bedingt umsetzbar. Zum einen erhält ein betrogener Empfänger nichts, wenn er den Double Spend aufdeckt, was die Akzeptanz der Empfänger senkt. Die Akzeptanz des Senders ist unter anderem abhängig von der Höhe des Strafkontos. Je unprofitabler man den Double Spend machen möchte, um die Sicherheit des Systems zu erhöhen, desto höher muss das Strafdeposit sein.

In MICROPAY2 wird eine dritte Partei in das System eingebracht. Diese gilt als „teilweise vertrauenswürdig“ und wird daher Verifiable Transaction Service (VTS) genannt. Prinzipiell werden alle Tätigkeiten der VTS veröffentlicht und daher wird ein falsches Verhalten schnell entdeckt und dieser bestimmte VTS nicht mehr verwendet. Ein VTS wird genutzt, um Winning Tickets auszuzahlen und er ist verantwortlich für das Strafkonto. Seine Aufgabe umfasst sowohl das Zerstören des Strafkonto als Bestrafung, als auch die Rückzahlung dessen an den Sender. Das Strafkonto, welches der Sender zu Beginn erstellt, benötigt dafür eine Multisignatur 2-von-2 durch den Sender und den VTS. Der Sender erhält eine einseitig unterschriebene Transaktion des VTS zu Beginn und kann diese nach Ablauf einer gewissen locktime nutzen, um sein Strafkonto wieder aufzulösen. Im Gegenzug erhält der VTS nach dem Ticketversand ebenfalls eine einseitig unterschriebene Transaktion des Senders, was ihm das Auszahlen des Strafkontos innerhalb der nächsten k-Blöcke ermöglicht. Der Sender schickt diese Transaktion auch an den Empfänger, damit dieser sie bei Fehlverhalten des Senders auch selbst dem VTS zur Verfügung stellen kann. So ist sichergestellt, dass der Sender bei unerlaubten Verhalten bestraft werden kann.

Der Empfänger benötigt zur Auszahlung des Deposits ebenfalls zwei signierte Transaktionen. Die eine erhält der Empfänger vom VTS, wenn er diesem ein gültiges Winning Ticket zeigen konnte. Die andere erhält der Empfänger vom Sender bei der Ticketübertragung. Auf diese Weise muss ein Ticketaustausch erfolgt sein, bevor der Sender eine Bezahlung erhält, sonst könnte ein böswilliger VTS leicht mit einem Empfänger kooperieren. Der VTS wird bezüglich anderer Angriffsszenarien innerhalb des Schemas

grundlegend als eine Art Sicherheitspunkt behandelt. Er könnte auch aufgrund eines eigentlichen Nullpayment Ticket eine Auszahlung bewirken oder auch anders herum. Die einzige Sicherheitsbegrenzung ist, dass diese Taten nachvollzogen werden können, da der VTS seine Transaktionen auf einer alternativen Chain veröffentlichen muss. Dem Waiting Merchant Problem wird durch eine zusätzliche Interaktion begegnet, in welcher der Empfänger nach Ticketempfang dem Sender die Informationen zur Verfügung stellt, um die Art des Tickets zu erfahren. Verweigert der Empfänger diese Information, so muss der Sender k-Blöcke warten, bevor er wieder Tickets ausstellt und würde diesen Empfänger eventuell vermeiden.

Zusammenfassend ist dies ein etwas umfangreicherer Ansatz, als MICROPAY1, dafür wird einigen Sicherheitsproblemen begegnet. Den VTS direkt als Smart Contract umzusetzen ist so nicht möglich, da dieser nicht signieren kann. Jedoch ist seine Funktionalität an sich in einem Smart Contract implementierbar. [64] Mit erhöhter Sicherheit steigt allerdings auch der Aufwand, sowohl On-Chain, als auch Off-Chain. Die drei Parteien interagieren sehr viel miteinander, was aber die Performance des Ansatzes vermindert. Der letzte Ansatz soll Sicherheit und Performance miteinander kombinieren.

MICROPAY3 nutzt ebenfalls einen VTS als dritte Partei, setzt ihn jedoch anders ein. Um die Anzahl der Interaktionen zu verringern und die Performance zu steigern, ohne an Sicherheit zu verlieren, wird der VTS nur genutzt, wenn es im Austausch zwischen Sender und Empfänger ein Problem gibt. Der Austausch zwischen Sender und Empfänger soll inklusive der Erweiterung mit dem Strafkonto (mit Adresse a^{pen}) folgend erläutert werden.

- Der Empfänger erstellt wie bei MICROPAY1 ein Commitment c zu seiner Zufallszahl r_1 und schickt c und seine Adresse a_2 zum Sender.
- Der Sender wählt eine eigene Zufallszahl r_2 und erzeugt eine Signatur σ auf $c, r_2, a_{2,} a^{pen}$. Außerdem erstellt er eine Signatur $\sigma_1$auf die Transaktion a^{esc}, a_2, welche eine Auszahlung seines Depositkontos an das Konto des Empfängers bewirken würde.
- Der Empfänger überprüft die Deckung der Konten, die Korrektheit der Signaturen und dass die locktime noch nicht abgelaufen ist.
- Anschließend prüft er nach dem gleichen Verfahren, wie bei MICROPAY1, ob das Ticket ein Gewinn ist.

Ist dies der Fall, erstellt der Empfänger $x = (c, r_1, s, r_2, \sigma)$ und sendet x, a^{esc}, a_2 zum Sender. Der Sender kann mit s das Commitment öffnen und prüfen, ob r_1 korrekt ist. Er prüft außerdem, ob das Ticket wirklich ein Gewinn ist und erstellt eine weitere signierte Transaktion σ_2. Mit dieser und σ_1 kann der Empfänger nun sich seinen Gewinn auszahlen.

Der VTS kommt dann zum Einsatz, wenn innerhalb eines gewissen Time-outs der Sen-

der die zweite signierte Transaktion nicht schickt. Dann sendet der Empfänger x, a^{esc}, a_2 zum VTS, der alles prüft und selbst die Transaktion a^{esc}, a_2 signiert. Mit dieser Signatur σ_T kann nun der Empfänger mit σ_1 auf seinen Gewinn zugreifen. Dieser Ansatz benötigt also mehr Interaktion zwischen Sender und Empfänger, um im Gegenzug den VTS zu entlasten. Auf diese Weise ist allerdings die Sicherheit durch erhöhte Latenz gesunken. Der Sender hat wesentlich mehr Übersicht, da er immer weiß, wann ein Winning Ticket entstand. Er könnte daraufhin gezielt Double Spend Attacken machen, da der Empfänger einen gewissen Time-Out abwarten muss, bevor er sich an den VTS wenden kann. Auf diese Weise muss der Empfänger länger auf seine Zahlung warten, da die Verantwortung zunächst beim Sender liegt.

Nach Betrachtung der drei verschiedenen MICROPAY-Ansätzen wird die größte Herausforderung bei probabilistischen Zahlungsmethoden deutlich - Sicherheit. Vor allem der Angriff durch Double Spend birgt enorme Gefahr. In jedem Schema hat der Sender die Möglichkeit (im großen Stil) eine größere Ticketmenge auszugeben, als er in seinem Deposit deckt. Die Autoren würden in diesem Fall nur den Sender bestrafen, den Empfängern aber keine Sicherheit für eine Auszahlung geben. Ihr Argument ist, dass wenn Sender einen Vorteil durch das Aufdecken von Double Spends hätten, sie dann warten könnten, ihr Winning Ticket zu veröffentlichen, bis sie selbst ein weiteres haben, oder jemand anderes eins veröffentlicht. Dieses Problem muss natürlich auch betrachtet werden, dennoch sinkt die Akzeptanz der Empfänger, wenn sie für ein Winning Ticket im Falle eines Double Spends leer ausgehen.

Zusammengefasst liefern diese Schemata die Grundlagen für probabilistische Micropayments mit vielen Denkanstößen, einigen Lösungen und viel Raum für Verbesserungen und Änderungen. Prinzipiell sind sie auf eine dezentrale Lösung ausgelegt, die Umsetzung in Smart Contratcs bedarf dennoch einiger Änderungen. Das Paper hat dabei bereits Ansätze auf Grundlage von Bitcoin gefunden und auch Micro-Benchmarks aufgeführt. [52]

4.3.2 Ansatz nach Caldwell

Eine weitere Idee, wie probabilistische Micropayments auf einer Blockchain umgesetzt werden können, wurde 2012 von Mike Caldwell in einem Bitcoin Forum (bitcointalk.org) veröffentlicht. Er bezeichnet es bereits als Ansatz für Nanopayments, da ein winziger Betrag beispielsweise ein zehntausendstel Bitcoin übertragen werden soll. Seine Erklärungen basieren auf Bitcoin, auch wenn sein Ansatz zu dem Zeitpunkt der Veröffentlichung nicht vollumfänglich auf der Blockchain von Bitcoin umsetzbar ist.

Das Set-up umfasst folgende Schritte. Zunächst informiert der Sender den Empfänger, dass er mit ihm interagieren möchte. Daraufhin erzeugt der Empfänger eine neue Bitcoin Adresse, dessen öffentlicher Schlüssel noch ungenutzt, also geheim ist. Diese

Adresse teil er dem Sender mit. Der Sender hinterlegt einen Bitcoin in einer TxOut[6]an die mitgeteilte Adresse, welche folgende Bedingungen zur Auszahlung hat:

- Transaktion muss durch Sender signiert sein.
- Kenntnis des öffentlichen Schlüssels zur mitgeteilten Adresse
- Transaktion muss die Gewinnbedingung erfüllen. Als Gewinnbedingung wird ein Wert Modulo gerechnet, wobei der Divisor die Wahrscheinlichkeit beeinflusst. Ist das Ergebnis null, so gilt die Gewinnbedingung als erfüllt. Der Wert kann beispielsweise eine vom Sender gewählte Zufallszahl sein, welche vom Empfänger signiert wird, sodass er für den Sender nicht vorhersagbar ist.

Als Micropayment schickt der Sender Transaktionen zum Empfänger, welche der Sender signiert hat, wobei die darin mitgeteilte Zufallszahl variiert. Erhält der Empfänger eine Transaktion, welche die Gewinnbedingung erfüllt, leitet er sie zur TxOut und erhält den hinterlegten Bitcoin. Um Front Running durch den Sender zu vermeiden, soll eine locktime genutzt werden. Und der Empfänger soll außerdem die Chain beobachten, um die Ausgabe „seines“ Coins festzustellen und den Dienst für den Sender einzustellen. [66]

Der Ansatz deckt die Anforderungen an ein Micropayment ab. Es bindet allerdings den Empfänger stark an den Sender, da dieser eine Adresse eigens für den Sender erstellt. Außerdem muss der Sender für jeden Empfänger, mit dem er interagieren möchte, einen Bitcoin hinterlegen. Es entsteht somit kein Vorteil gegenüber der Nutzung von State/Payment Channels.

4.3.3 Orchid

Das Orchid Netzwerk [64] will mit einem dem Tor Browser ähnlichen Prinzip anonymes Internet ermöglichen. Dabei soll der Node, welcher Bandbreite anbietet, kontinuierlich von seinen Nutzern bezahlt werden. Innerhalb des Orchid Netzwerkes wird ein ERC20 Token namens Orchid Token eingesetzt. Allerdings ist dies kein Token im Sinne von Micropayments, sondern hat ausschließlich sozioökonomische Vorteile. Es werden also Orchid Token für die Macropayments genutzt, statt Ether, was aber an der Handhabung nichts ändert.

Die Lösung von Orchid lehnt sich an MICROPAY1 an und wurde auch durch MICROPAY2 und 3 inspiriert. Dabei wurde es in sofern abgeändert, dass die Partei, welcher man in diesen Ansätzen vertrauen musste, wegfällt und ihre Funktionalität durch einen Smart Contract abgedeckt wird. Orchid strebt eine ähnliche Skalierung an wie Incubed. Im Netzwerk werden viele Clients mit wenigen Nodes interagieren und die Clients

[6] Ausgabe in einer Transaktion, welche ein Wertefeld mit einem gewissen Betrag und ein Skript enthält, welches Bedingungen beinhaltet, um den Betrag zu erhalten. [65]

werden unterschiedliche Nodes nutzen. Ein Client-Node-Beziehung gebundenes Set-up hätte Nachteile für beide Seiten. Ein Node müsste Informationen für jeden Client, mit dem er interagiert für eine gewisse Zeit speichern, ohne zu wissen wie lang die Beziehung sein wird. Auch für einen Client ist es unrentabel für jede Interaktion mit einem neuen Node beispielsweise ein neues Deposit zu erstellen.

Das Set-up des Clients ist aus diesem Grund unabhängig vom Node. Hierbei wird ein Deposit und ein sogenanntes penalty escrow (Strafkonto) in einem Smart Contract hinterlegt. Das Deposit wird für die Bezahlungen der Macropayments genutzt und das Strafkonto soll Double Spend unprofitabel machen. Diese Strafkaution wird im Fall eines Overspending verbrannt. Der Ticketaustausch wird folgendermaßen abgewickelt:

Der Empfänger wählt eine Zufallszahl, erstellt einen Hash zu dieser und schickt diesen zum Sender. Der Sender wählt die Werte für Gewinnwahrscheinlichkeit und Gewinnwert, aus denen er ein Ticket erstellt. Im Ticket ist außerdem der Hash der Zufallszahl, ein Zeitstempel und der Hash des Tickets abgelegt. Diesen Hash signiert der Sender mit seinem privaten Schlüssel und diese Signatur wird ebenfalls dem Ticket angefügt. Anschließend schickt der Sender das Ticket zum Empfänger. Dieser verifiziert die Korrektheit des Tickets und überprüft, ob es ein Gewinn ist. Dies ist der Fall, wenn der Hash über den signierten Ticket-Hash und seiner anfangs gewählten Zufallszahl kleiner ist, als die Gewinnwahrscheinlichkeit.

Ist es ein Gewinn, wird das Ticket und die Zufallszahl an den Smart Contract übertragen und der Empfänger erhält seine Makrozahlung. Wenn das Deposit für diese Zahlung zu klein ist, wird eine Art Flag, also Zeichen in der Datenstruktur gesetzt, dass die Strafkaution zerstört werden kann. Würde der Sender allerdings Double Spend im großen Stil ausführen, könnte die Strafkaution zu klein sein, um diesen Angriff wirklich unprofitabel zu machen. Um diesem Problem und auch einem Waiting Merchant entgegenzuwirken, hat das Ticket einen Zeitstempel und der Wert des Tickets wird mit der Zeit exponentiell kleiner. Der Empfänger hat also das Ziel ein Ticket möglichst schnell einzulösen. Auf diese Weise würde bei einem Double Spend im großen Stil das Deposit zeitnah zu klein werden und die Strafkaution zerstört werden. Ab diesem Zeitpunkt würde kein Empfänger mehr Tickets von dem Sender entgegennehmen. Dieser Ansatz kann die Attacke nur einschränken und nicht wirklich verhindern. Ob und wie der Empfänger wieder Zugriff auf sein Deposit oder seine Strafkaution erhält, wird im Paper nicht beschrieben.

Zusammenfassend ist dieser Ansatz sehr klassisch, der Ticketaustausch hält sich an den Ablauf aus MICROPAY1 und die Absicherung gegen Angriffe erfolgt durch eine Bestrafung des Täters. Das Paper bietet außerdem eine Analyse über die Performance und stellt die kryptografischen Operationen als Bottleneck (Engstelle) heraus. Es werden verschiedene Maßnahmen zur Reduktion vorgeschlagen. Für den erhöhten Aufwand durch das per-ticket Commitment durch den Hash der Zufallszahl wird VRF als Lösung genannt. [53, 64]

VRF steht für Verifiable Random Function und ist eine nachprüfbare Zufallsfunktion. Der Output dieser Funktion soll also nicht vorhersagbar, aber die Richtigkeit überprüfbar sein. Dabei wird asymmetrische Verschlüsselung eingesetzt. Aus einem Input x kann der Besitzer des geheimen Schlüssels SK mit der Funktion $y = F_{SK}(x)$ berechnen und einen Beweis $\pi_{SK}(x)$ erstellen. Dabei ist y pseudo-zufällig und jeder kann die Korrektheit mithilfe des Beweises und des öffentlichen Schlüssels überprüfen. [67, 68]

Obwohl es viele nützliche Anwendungen gibt, sind VRF noch nicht ausreichend erforscht und die Umsetzungen oft uneffizient. Auch wenn die EVM inzwischen in der Lage ist, VRF einzusetzen, wurde es noch nicht in Systemen mit erheblichen Wert eingesetzt und die Funktionalität und Sicherheit nachgewiesen. Aus diesem Grund verzichtet Orchid aktuell auf den Einsatz von VRF. [64]

4.3.4 Decentralized Anonymous Micropayments

Decentralized Anonymous Micropayments (DAM) [70]ist der komplexeste Ansatz. Er nutzt verschiedene Unterwährungen und verschiedene Arten der Transaktionen. Darunter befinden sich neben Makrozahlungen, Auszahlungen und Beschwerdemeldungen auch probabilistische Zahlungen. Alle Transaktionen sollen, um wirklich anonym zu sein, keine Information über Herkunft, Ziel oder Betrag von Zahlungen veröffentlichen. Es wird unterschieden zwischen deterministischen Zahlungen, also eines Macropayments, welches non-interaktiv erfolgt und einer probabilistischen Zahlung. Diese basiert auf einem 3-Nachrichten-Protokoll. Das Zahlungsschema wurde durch MICROPAY1 inspiriert und der Ansatz zur Anonymisierung durch DAP (Decentralized Anonymous Payment). Da eine einfache Kombination dieser beiden nicht ausreichend anonymisiert und auch unsicher ist, wurde sie weiterentwickelt.

Die Teilnehmer können drei verschiedene Arten von Coins erstellen - Standard Coins, Deposit-Coins und Tickets. Standard Coins werden genutzt, um Makrozahlungen abzuwickeln und sind die ursprüngliche Währung der Blockchain. [7] Deposit Coins werden zum Bezahlen der Macropayments bei probabilistischen Zahlungen genutzt und als Strafe eingezogen. Tickets werden für Micropayments eingesetzt und sind immer mit dem Deposit verbunden, welches sie deckt. Mit Mining Transaktionen werden Coins erstellt, welche sowohl übertragen werden können, als auch die Coin-Art wechseln können. Tickets können also erzeugt, ausgetauscht, eingelöst und aktualisiert werden. Außerdem gibt es Auszahlungstransaktionen bei denen Tickets wieder in Standard-Coins umgewandelt werden können. Außerdem kann ein Fehlverhalten gemeldet werden und die dementsprechende Bestrafung eingefordert werden. In dieser Arbeit soll sich auf die probabilistischen Zahlungen, also den Austausch von Tickets konzentriert werden.

Jedes Ticket enthält eine einmalige Kennung (Identifier), eine Gewinnwahrscheinlichkeit

[7] DAM soll auf einer eigenen Blockchain ablaufen und baut nicht auf eine vorhandene Blockchain auf.

und den Wert des Macropayment sowie Informationen über das verbundene Deposit. Diese Information enthält den einmaligen Identifier des Deposits und den Wert des Deposits. Ein Deposit ist initial valide und wird invalide, wenn ein Double Spend bei einem Macropayment auftritt. Ist ein Deposit als invalide markiert, akzeptiert der Empfänger keine damit verbundenen Tickets mehr.

Das 3-Nachrichten-Protokoll des Ticket-Austauschs läuft wie folgt ab:

- Die erste Nachricht wird vom Empfänger zum Sender geschickt und beinhaltet einen Session Identifier, den öffentlichen Schlüssel der Session, eine Blacklist an Deposits der aktuellen Periode und den gewünschten Zahlungsbetrag.
- Der Sender erzeugt nun aus dem Ticket einen Coin und nutzt fraktionierte Nachrichtenübertragung (FMT) für eine probabilistische Nachrichtenübertragung. Neben dem Coin erstellt der Sender zwei weitere entscheidende Größen. Einen Worst-Case-Rate-Limit Tag (wcrlt) und einen Double-Spend-Tag. Mit dem Limit Tag kann der Empfänger eine Grenze für den Zahlungswert erzwingen und mit dem Double-Spend Tag kann er das Deposit erhalten, wenn ein Ticket doppelt in Macropayments ausgegeben wurde. Der Sender erstellt zwei Commitments, eines enthält unter anderem den Ciphertext c und das andere den wcrlt. Dabei sind die beiden insofern verknüpft, dass das Öffnen des ersten Commitments das Öffnen des Zweiten ermöglicht. Er schickt dann die beiden Commitments, den Inhalt des ersten Commitments, den Schlüssel davon und weitere Informationen zum Empfänger.
- Der Empfänger überprüft alles auf Richtigkeit und versucht, c aus dem ersten Commitment zu entschlüsseln. Wenn er das tun konnte, öffnet er auch das zweite Commitment und veröffentlicht die Transaktion, um sein Macropayment zu erhalten. Stellt er fest, dass das Ticket bereits ausgegeben wurde, erstellt und veröffentlicht er eine Bestrafungstransaktion. Zuletzt kann er dem Sender mitteilen, ob es ein Nullpayment oder ein Macropayment war.

Dieser Ablauf ist sehr theoretisch und beruht auf verschiedene Verschlüsselungsverfahren und ist somit kostenintensiv. Dies ist allerdings nötig, da das ganze Verfahren anonym bleiben soll und keine Informationen nach außen gelangen sollen. [69, 70] Das Konzept wirkt sehr durchdacht, wird aber viel Arbeit benötigen, um wirklich umgesetzt zu werden. Für die Anwendung im IoT-Bereich sind die Anforderungen an den Client zu hoch und die Transaktionen zu groß.

4.3.5 Streamflow

Streamflow ist ein Konzept des Livepeer Protokolls [72] , welches bessere Skalierbarkeit in das Livepeer Netzwerk bringen soll. Dieses Netzwerk besteht aus sogenannten Broadcastern, welche ein Video transcodieren wollen. Ihre Gegenspieler sind Orche-

strators, welche segmentweise diese Videos bearbeiten. Dabei entsteht ein ständiger Austausch an Videosegmenten, der aus verschiedenen Gründen jederzeit abgebrochen werden kann. Auf der Suche nach einem passenden Zahlungsschema, welches großteils Off-Chain arbeitet und die Eigenschaften des Netzwerkes nicht aufhebt, haben sie Streamflow erstellt und den Ansatz in einem Paper veröffentlicht.

Streamflow basiert grundlegend wieder auf drei Parteien:

- dem Broadcaster, welcher der Sender der Zahlung ist,
- dem Orchestrator, welcher der Empfänger ist
- und ein Broker, der in diesem Fall ein Smart Contract ist.

Neben diesem Smart Contract gibt es noch zwei weitere grundlegende Smart Contracts - einer für die Reserve und ein Manager, der zur Registrierung dient. Diese Funktionalitäten könnten auch in einem Smart Contract zusammengefasst werden, sollen aber für eine bessere Übersicht im Ablauf aufgeteilt werden.

Als Set-up muss sich jeder Sender und jeder Empfänger registrieren. Der Sender erstellt außerdem ein Deposit und eine Reserve. Der Ablauf in Streamflow ist periodisiert, was bedeutet, dass es Runden gibt. Innerhalb einer Runde werden runden-spezifische Tickets erstellt, ausgetauscht und eingelöst. Außerdem wird die Reserve eines Senders virtuell auf alle in der Runde registrierten Empfänger aufgeteilt. Auf diese Weise ist jedem Empfänger ein Teil der Reserve eines Senders zugesichert, auch wenn der Sender eine Double Spend Attacke macht. Zunächst soll allerdings der Ticketaustausch erläutert werden.

Der Sender fragt den Empfänger nach den Ticket-Parametern. Zu diesen Parametern gehören Gewinnhöhe, Gewinnwahrscheinlichkeit und ein Commitment zu der Zufallszahl des Empfängers. Der Empfänger hat aufgrund dieser Anfrage des Senders die Möglichkeit den Sender zu überprüfen. Darauf basierend kann er entscheiden, ob er mit diesem Empfänger arbeiten möchte. Ist dies der Fall, schickt er ihm die gewünschten Parameter. Der Sender nutzt diese und erstellt daraus ein Ticket. Im Ticket steht außerdem die aktuelle Rundennummer und der dazugehörige Runden-Hash.[8] Der Sender berechnet den Hash des Tickets und signiert ihn. Die Signatur, den Hash und das Ticket schickt er an den Empfänger. Dieser prüft alles auf Richtigkeit und ob es ein Winning Ticket ist. Das wird auch in diesem Schema durch eine Hashfunktion über die Signatur des Senders und die Zufallszahl des Empfängers geprüft. Ist dieser Hash kleiner als die Gewinnwahrscheinlichkeit, führt dieses Ticket zu einem Macropayment. In diesem Fall ruft der Empfänger eine Funktion im Broker Smart Contract auf, welcher er das Ticket, die Signatur und seine Zufallszahl übergibt. Der Broker überprüft alles und transferiert den entsprechenden Betrag vom Deposit des Senders an die Adresse des

[8] Um sicherzustellen, dass keine Tickets für die nächste Runde vorproduziert werden können, speichert der Manager zur Rundennummer den Hash des aktuellen Blocks der Ethereum Blockchain.

Empfängers.

Ist der Empfänger Opfer eines Double Spends, wird die Auszahlung der Reserve an den Reserve Smart Contract übergeben. Dieser prüft, ob der Empfänger für die aktuelle Runde registriert ist. Ist dies der Fall, prüft er wie hoch die Reserve des Senders ist und teilt diese auf alle registrierten Empfänger der Runde auf. Er prüft, wie viel dem Empfänger bereits zugesichert wurde und erhöht diesen Anteil entsprechend der Höhe des Gewinns und abhängig vom Anteil, den er bekommen darf. Am Ende der Runde bekommen die Empfänger die ihnen zugesicherten, eingeforderten Reserve-Anteile ausgezahlt.

Beim ersten Zugriff auf die Reserve eines Senders wird diese außerdem als eingefroren (freeze) gekennzeichnet. Dafür wird die aktuelle Runde als Freeze-Round, also die Runde, in der sie eingefroren wurde, gespeichert. Für eine gewisse Rundenanzahl (Freeze Periode), kann die Reserve durch den Sender weder aufgefüllt, noch ausgezahlt werden. Eine andere Rundenanzahl (Unlock Periode) wird genutzt, um Auszahlungen durch den Sender zu ermöglichen. Möchte er sein Deposit verringern oder an sich selbst auszahlen, so wird ebenfalls die Rundennummer verwendet. In diesem Fall wird die Runde notiert, ab welcher der Sender in der Lage ist, auf sein Deposit zuzugreifen. Der Empfänger überprüft diese Information, bevor er einem Empfänger die Ticketparameter zur Verfügung stellt. Auf diese Weise kann ein Empfänger keine Front Running Attacke durchführen.

Das Schema kann sehr variabel an seinen Einsatz angepasst werden. Zum einen kann entschieden werden, wie lange ein Ticket gültig sein soll und wie lang die Freeze- und die Unlock-Periode sind. Zum anderen kann die Anzahl der Empfänger, die auf eine Reserve Zugriff haben reduziert werden. In diesem Fall würde der Sender ein Set erstellen, in welchem er einer Auswahl an Empfängern ihren Teil der Reserve garantiert. Auf diese Weise müsste die Größe der Reserve nicht mit der Anzahl der registrierten Empfänger steigen, sondern könnte variabel reguliert werden. Das Schema liefert damit eine gute Sicherheit und eine umfassende Lösung für Double Spend.

Dies ist allerdings mit einem höheren Aufwand verbunden. Zum einen müssen sich alle Beteiligten registrieren und die Smart Contracts sind recht umfangreich. Zum anderen ist die Interaktion zwischen Sender und Empfänger auf drei Nachrichten erhöht, was auch eine höhere Latenz mit sich bringt. Die Skalierbarkeit ist dennoch gut, da keine Beziehung zwischen Sender und Empfänger erstellt werden muss, der Empfänger muss lediglich für die Reserve registriert sein. Gegen eine Replay Attacke ist die einzige genannte Lösung, das Aufzeichnen und Speichern der Hashs der genutzten Winning Tickets. Der Speicheraufwand dafür ist allerdings begrenzt, da mit Ablauf der Periode, in der sie gültig sind, die Hashs nicht weiter gespeichert werden müssen. [71, 72] Der Ansatz ist komplex, aber gut durchdacht und es liegen bereits Entwürfe der Smart Contracts und des Codes von Empfänger und Sender vor. [73]

4.3.6 Poollösung

In einem System, in welchem Zahlungen abgewickelt werden, gibt es oftmals eine Unterteilung in Sender und Empfänger. So ist es auch bei Incubed der Fall, denn es gibt Clients, welche die Sender sind und Nodes als Empfänger der Zahlung. Eine Idee, um Zahlungen weiter zusammenzufassen, beruht darauf, dass nicht ein Client einen Sender bezahlt, sondern sich Gruppen (Pools) bezahlen. Das kann in einem 1-zu-n Schema umgesetzt werden, also eine Gruppe interagiert mit einem einzelnen. Es kann aber auch in einem n-zu-m Schema geschehen, bei dem eine Gruppe von m Teilnehmern an eine Gruppe von n Teilnehmern Zahlungen durchführt.

Das Absichern, wer wie viel eingezahlt hat und wer wieviel erhält, kann durch eine zentrale Instanz abgesichert werden. Die Sender Gruppe (Pool S) zahlt in ihren Pool ein und es wird zentral vermerkt, wer wie viel eingezahlt hat. Der Empfänger-Pool (Pool E) hat ebenfalls eine zentrale Instanz, die Information darüber hat, welchem Empfänger wie viel Anteil des Pools zusteht. Aus vielen kleinen Zahlungen von Mitgliedern aus Pool S an Mitglieder aus Pool E entsteht eine große Zahlung von Pool S an Pool E. Diese wird dann intern durch den Manager des Pool E aufgeteilt.

Würde man diesen Ansatz beispielsweise mit probabilistischen Zahlungen umsetzen, erinnert es an eine Lotto-Tippgemeinschaft. Es werden also mehrere Tickets in einem Pool gesammelt, eingelöst und sollte eines der Tickets gewinnen, wird der Gewinn auf die Teilnehmer des Pools aufgeteilt. Diese Aufteilung könnte von der Anzahl an Lottoscheinen, die der Teilnehmer dem Pool beigesteuert hat, abhängig gemacht werden. [74]

Wendet man diesen Ansatz auf Incubed an, könnten sich Nodes zusammenschließen und regelmäßiger Gewinne bekommen. So können vor allem Nodes, die nur eine geringe Anzahl an Tickets erhalten, gemeinsam ihre Chance auf einen Gewinn erhöhen und eine Zahlung erhalten. Bei dieser Anwendung müsste ein zentraler Verwalter, dem alle Vertrauen müssen, allerdings entfallen. Stattdessen müsste auf die zur Verfügung stehenden kryptographischen Mittel und Datenstrukturen oder Smart Contracts zurückgegriffen werden. Ein System, welches darauf aufbaut ist Cardstack.

[75–77]

4.3.7 Fazit vorgestellter Lösungen

Beim Vorstellen der verschiedenen Lösungen wurden einige Herausforderungen der Dezentralisierung und im Besonderen der Blockchain festgestellt. Von den vorgestellten Ansätzen haben nur zwei eine wirkliche Umsetzung in Form von programmierten Smart Contracts erfahren. Diese beiden sind Orchid und Streamflow. Eine zentrale Lösung auf

die Blockchain zu übertragen, bedeutet neben großem Aufwand auch den Verlust des Charakters der Lösungen, die auf eine Bank als zentrale Einheit setzen. Viele zentrale Lösungen haben das Ziel, die Bank zu entlasten, während die dezentralen Lösungen das Ziel haben die Transaktionen auf Blockchain zu minimieren. Beim Verringern der Transaktionen auf der Blockchain profitieren auch die beiden anderen Parteien davon, durch die Einsparung an Transaktionskosten. Doch die Verringerung des Einsatzes von Blockchain oder Banken erhöht auch den Aufwand bei Sender und Empfänger. Sie müssen selber prüfen, ob alles valide ist und dadurch steigt der Rechenaufwand.

Auch der Aufwand ein System so weit zu entwickeln, dass es eingesetzt werden kann, sollte beachtet werden. Orchid und Streamflow sind komplexer als die MICROPAY-Ansätze, gerade weil sie in vollem Umfang umgesetzt wurden. DAM ist bereits in der Theorie sehr umfangreich und würde viel Aufwand in der Entwicklung bedeuten. Der Ansatz von Caldwell scheint überschaubar, dennoch wird bei Micropayments für Bitcoin zunächst bitcoinj genannt, welches mit Payment Channels arbeitet. [78]

Neben den Angriffsmöglichkeiten kann die menschliche Psyche ein weiteres großes Problem für probabilistischen Micropayments sein. Bei heutigen Angeboten existieren oft sowohl Abo-Tarife, als auch Pay-per-Use-Tarife und die Tendenz der Konsumenten geht laut empirischen Studien zum Abo-Tarif. Verschiedene psychologische Effekte bewegen den Konsumenten teilweise dazu, die unökonomische Entscheidung für die Flatrate zu fällen [79]:

- Überschätzungseffekt: Der Konsument überschätzt die tatsächliche Nutzung in der Zukunft.
- Versicherungseffekte: Der Konsument kann sicher sein, dass er nicht mehr zahlen wird, als den Abo-Beitrag.
- Taxametereffekt: Durch kontinuierliche, kleine Zahlungen „verliert“ der Konsument immer wieder erneut Geld.
- Bequemlichkeitseffekt: Konsument könnte viele einzelne Transaktionen als aufwendig empfinden.

Vor allem der Versicherungseffekt wird beim Einsetzen von probabilistischen Micropayments verstärkt. Ein Konsument kann davor zurückschrecken, nicht kontrollieren zu können, wann und wie häufig er bezahlen muss. Trotz der rationalen Sicherheit durch das Gesetz der großen Zahlen, kann die Angst überwiegen und das Risiko, überzubezahlen als zu groß erscheinen. Aus der Sicht der menschlichen Psyche spricht also einiges gegen Micropayments und im Besonderen gegen wahrscheinlichkeitsbasierte Zahlungen. Auch wenn eventuell letztendlich nur IoT-Geräte interagieren, so muss das System durch Menschen eingesetzt werden, welche davon überzeugt sein müssen.

Betrachtet man die vorgestellten Ansätze aus Sicht von IoT-Geräten, beziehungsweise in Kombination mit Incubed wird ein weiteres Problem klar: Latenz. Beim Ticketaus-

tausch interagieren in den Ansätzen Sender und Empfänger miteinander. Bis zu drei Nachrichten werden pro Zahlung ausgetauscht. Soll allerdings beispielsweise ein Sensor in kurzer Zeit mehrere Werte in die Blockchain schreiben, so ist die Latenz durch die hohe Interaktion zu groß. Die Interaktionen im Austausch sind jedoch aus Sicherheitsgründen in den Ansätzen nötig.

Auch die Größe der auszutauschenden Nachrichten kann kritisch für ein IoT-Gerät mit geringer Performance werden. Dazu kommen kryptografische Berechnungen, die Rechenkapazität benötigen und gegebenenfalls die Latenz erhöhen. In mehreren Publikationen werden VRF als Möglichkeit genannt, um den Austausch non-interaktiv zu gestalten. Doch dieser Ansatz ist noch nicht ausreichend getestet und vor allem auf der Blockchain mit einigen Schwierigkeiten verbunden. [64]

Obwohl viele Argumente genannt wurden, die gegen die Ansätze der probabilistischen Micropayments, insbesondere in Bezug auf Incubed, sprechen, wurde als Teil dieser Arbeit ein Prototyp erstellt. Der entwickelte Prototyp wird im folgenden Kapitel vorgestellt werden.

5 Umsetzung

5.1 Smart Contracts

Aufgrund des vielversprechenden Ansatzes für den Umgang mit dem Double Spend Problem dient Streamflow als Inspiration für den Prototypen. Die Vorgehensweise von Streamflow ist gut dokumentiert und es sind bereits Ansätze für den Quellcode veröffentlicht. Der Ablauf wurde bereits im Abschnitt 4.3.5 Streamflow erklärt, sodass in diesem Kapitel direkt auf die Umsetzung eingegangen werden kann. Zunächst soll die Arbeitsweise der Smart Contracts vorgestellt werden.

Für den Prototypen wurde sich auf die folgenden grundlegenden Funktionen der Smart Contracts für probabilistischen Zahlungen konzentriert:

- Initialisierung des Systems und der ersten Runde
- Anlegen eines Kontos mit Einzahlung in Deposit und Reserve durch den Sender
- Registrieren durch den Empfänger
- Einlösen eines Tickets und entsprechende Auszahlung aus Deposit und Reserve
- (sichere) Rückzahlung des Deposits an den Sender

Die ersten drei Punkte sind Teil des Set-ups. Deren Ablauf, Funktionen und Datenstrukturen sind zur besseren Übersicht in der Abbildung 5.1 dargestellt.

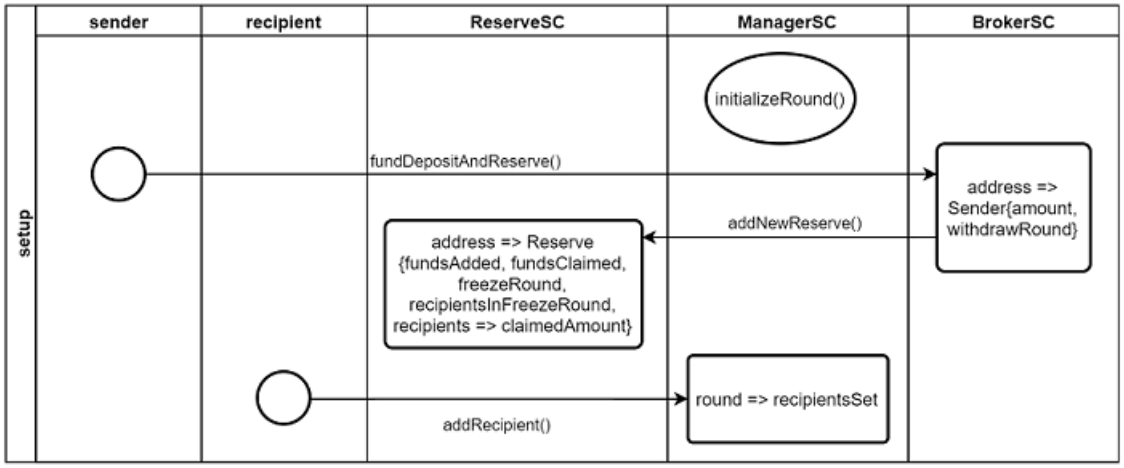


Abbildung 5.1: Flowchart 1 - Set-up

Aus diesem Flowchart ergeben sich die Komponenten des Systems. Es basiert auf drei Smart Contracts:

- Reserve Smart Contract: Er enthält die Datenstruktur für die Reserve und als Hauptfunktionen gelten das Anlegen einer Reserve und das Auszahlen aus ihr.

- Manager Smart Contract: Er stellt die Informationen über Runden und die jeweils dafür registrierten Empfänger bereit.
- Broker Smart Contract: Umfasst die Datenstruktur, die Informationen über den Sender enthält und die Hauptfunktion, die zum Einlösen von Gewinn-Tickets genutzt werden kann.

Zu den Smart Contracts kommen zwei verschiedene Gruppen von Akteuren hinzu. Zum einen Empfänger (recipients), welche im Set-up Vorgang sich registrieren und zum anderen Sender, welche ein Deposit und eine Reserve hinterlegen müssen. Als komplexere Datenstruktur soll die Reserve mit den dazugehörigen Funktionen erklärt werden.

Die Reserve wird als Struct in einem Mapping der Adresse des Senders zugeordnet. [9] In der Reserve selbst stehen neben dem eingezahlten und ausgezahlten Betrag Informationen bezüglich der Freeze-Round. Es wird sowohl die Runde angegeben, in welcher die Reserve eingefroren wurde, als auch die Anzahl der registrierten Empfänger in dieser Runde. Außerdem wird in einem Mapping aufgezeichnet, wie viel ein Empfänger bereits von dieser Reserve erhalten hat. Diese Informationen sind nötig, um in einer späteren Runde, wenn das Deposit nicht ausreicht, dem registrierten Empfänger den ihm zustehenden Teil auszuzahlen.

Die Datenstruktur für das Deposit wurde von Streamflow übernommen und besteht aus dem aktuellen Betrag des Deposits und eine Angabe ab welcher Runde der Sender selbst davon wieder abheben kann. Auch diese Structs werden in einem Mapping zur Adresse des Senders abgelegt.

Nach diesem Set-up können Sender und Empfänger Tickets austauschen, ohne mit den Smart Contracts zu interagieren. Es werden nur Informationen aus ihnen gelesen, beispielsweise um zu überprüfen, wie hoch das Deposit des Senders ist. Dieser Austausch wird im Detail im Abschnitt 5.2 erläutert werden.

Das Ticket hat im Smart Contract folgende Struktur:

```
struct Ticket {
    address recipient;          // Adresse des Empfängers
    address sender;             // Adresse des Senders
    uint256 faceValue;          // Wert eines Winning-Tickets
    uint256 winProb;            // Wahrscheinlichkeit eines Winning-Tickets
    bytes32 recipientRandHash;  // Hash des Zufallswert des Empfängers
    bytes auxData;              // Informationen über Erstellungsrunde
}
```

Abbildung 5.2: Codebeispiel 1 - Ticket (Struktur übernommen von Streamflow)

[9] Mappings sind Hashtabellen und Structs sind Datencontainer, die Variablen gruppieren, deren Datentypen benutzerdefiniert sind. [11]

Erhält der Empfänger beim Austausch ein Winning Ticket, kann er es beim Broker Smart Contract einlösen. Der entsprechende Flow-Chart für den dadurch ausgelösten Vorgang inclusive dem Ablauf bei einem Overspend ist in der folgenden Abbildung dargestellt.

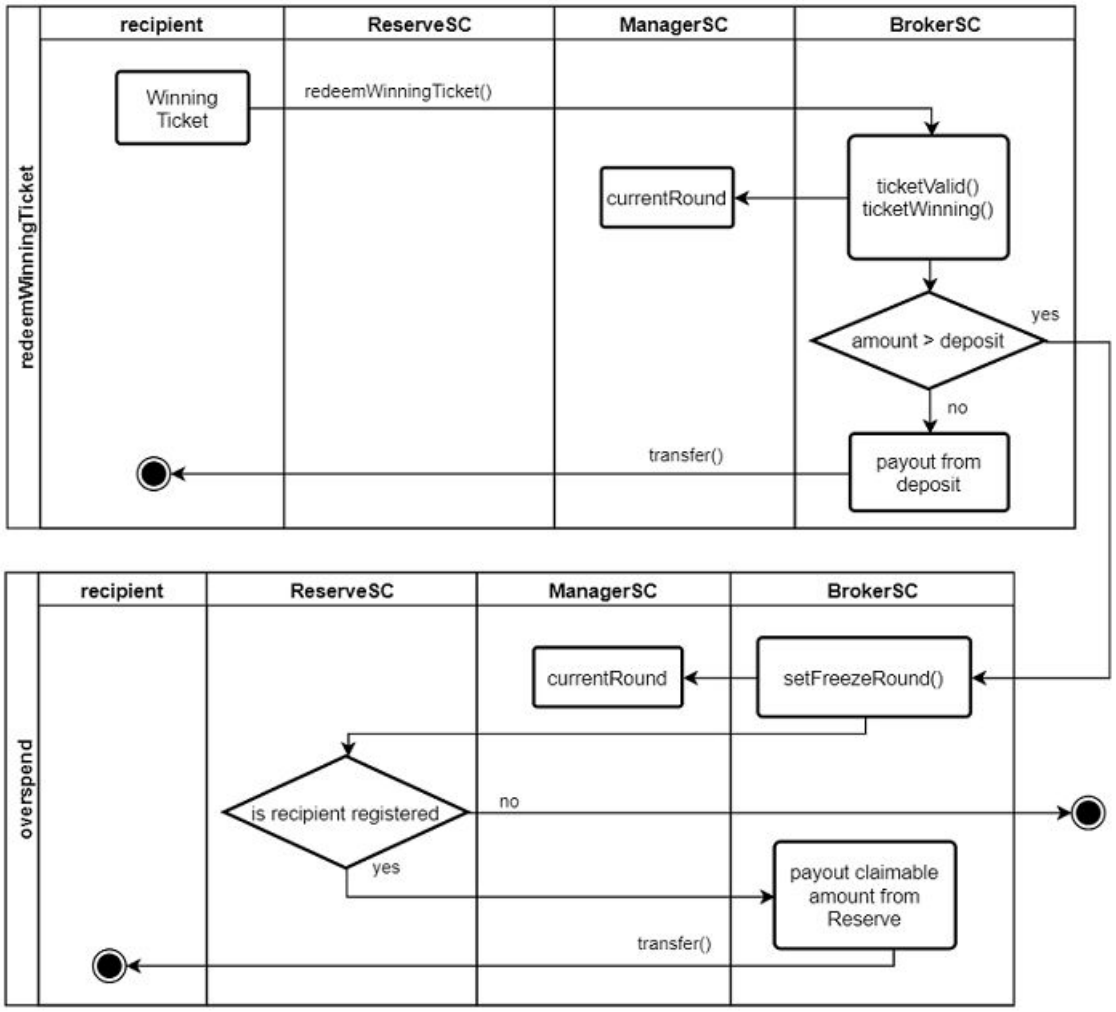


Abbildung 5.3: Flowchart 2 - redeemWinningTicket und Overspend

Die redeemWinningTicket-Funktion ist eine zentrale Funktion und soll daher näher betrachtet werden.[10] Zunächst wird innerhalb dieser Funktion die Hilfsfunktion requireValidWinningTicket aufgerufen. Sie überprüft, ob das Ticket, die Signatur über den Tickethash und die Zufallszahl korrekt sind.

[10] Einige Abläufe und Teile des Codes wurden übernommen von Streamflow. [80]

```
141 function requireValidWinningTicket(Ticket memory _ticket,
142 bytes32 _ticketHash, bytes memory _sig, uint256 _recipientRand) public view
143 {
144     require(_ticket.recipient != address(0), "ticket recipient is null address");
145     require(_ticket.sender != address(0), "ticket sender is null address");
146     require(
147         keccak256(abi.encodePacked(_recipientRand)) == _ticket.recipientRandHash,
148         "recipientRand does not match recipientRandHash");
149     require(!usedTickets[_ticketHash], "ticket is used");
150     require(
151         checkAuxData(_ticket.auxData),
152         "checkauxdata failed");
153     require(
154         isValidTicketSig(_ticket.sender, _sig, _ticketHash),
155         "invalid signature over ticket hash");
156     require(
157         uint256(keccak256(abi.encodePacked(_sig, _recipientRand))) < _ticket.winProb,
158         "ticket did not win");
159 }
```

Abbildung 5.4: Codebeispiel 2 - requireValidWinningTicket Funktion

Mithilfe eines Mappings soll dem Replay-Angriff, also dem wiederholten Einlösen des gleichen Tickets entgegengewirkt werden. Dafür wird geprüft, ob der Tickethash bekannt ist (Zeile 149). In der Funktion checkAuxData wird untersucht, ob das Ticket in der aktuellen Runde noch gültig ist (Zeile 151). Anschließend wird geprüft, ob die Signatur über den Tickethash vom im Ticket genannten Sender erstellt wurde (Zeile 154). Damit wird die Authentizität des Tickets bestätigt. Im letzten Schritt dieser Funktion wird betrachtet, ob das Ticket tatsächlich ein Winning Ticket ist. Dafür wird die Signatur des Senders über den Tickethash und die Zufallszahl des Senders gehasht. Das Ergebnis der Hashfunktion ist eine Zahl. Ist diese Zahl kleiner als die Gewinnwahrscheinlichkeit, so ist das Ticket ein Winning Ticket (Zeile 157). Wurde diese Funktion erfolgreich ausgeführt, erfolgt die Auszahlung in der redeemWinningTicket-Funktion entsprechend der folgenden Abbildung.

```
109   Sender storage sender = senders[_ticket.sender];
110   require(
111       sender.deposit > 0 || reserveSC.remainingReserve(_ticket.sender) > 0,
112       "sender deposit and reserve are zero"
113   );
114   usedTickets[ticketHash] = true;
115   if (_ticket.faceValue > sender.deposit) {
116       uint256 restDeposit = sender.deposit;
117       uint256 claimAmount = _ticket.faceValue.sub(restDeposit);
118       sender.deposit = 0;
119       uint256 claimedFromReserve = reserveSC.claimFromReserve(_ticket.sender,
120           _ticket.recipient, claimAmount);
121       amountToTransfer = restDeposit.add(claimedFromReserve);
122   }
123   else {
124       amountToTransfer = _ticket.faceValue;
125       sender.deposit = sender.deposit.sub(_ticket.faceValue);
126   }
127   if (amountToTransfer > 0) {
128       address payable recipient = address(uint160(_ticket.recipient));
129       recipient.transfer(amountToTransfer);
130   }
```

Abbildung 5.5: Codebeispiel 3 - redeemWinningTicket Funktion

Zunächst wird aus dem Struct des Senders dessen Deposit gelesen und außerdem die verfügbare Reserve überprüft. Ist in beiden kein Geld mehr hinterlegt, kann der Vorgang abgebrochen werden. Ist das Deposit ausreichend groß, wird der zu übertragende Wert abgezogen (Zeile 125) und anschließend an den Empfänger transferiert (Zeile 129). Ist der Wert des Tickets größer als das Deposit des Senders, wird eine Funktion des Reserve Smart Contracts aufgerufen (Zeile 119).

Die Funktion claimFromReserve() soll die Differenz vom Wert des Tickets und dem Betrag des Deposits an den Sender auszahlen.

```
73  function claimFromReserve(address sender, address recipient, uint256 amount)
74      public returns (uint256)
75  {
76      Reserve storage reserve = reserveMap[sender];
77      uint256 currentRound = managerSC.getCurrentRound();
78      if(reserve.freezeRound == 0 ||
79          currentRound >= reserve.freezeRound + freezePeriod) {
80          uint256 currentRecipients = managerSC.getPoolSizeCurrentRound();
81          reserve.freezeRound = currentRound;
82          reserve.recipientsInFreezeRound = currentRecipients;
83      }
84      if(reserve.freezeRound == 0 || reserve.recipientsInFreezeRound == 0 ||
85          !managerSC.isRegisteredRecipient(recipient)) {
86          return 0;
87      }
88      uint256 claimedFunds = reserve.claimedRecipient[recipient];
89      uint256 claimableFunds = reserve.fundsAdded.div(reserve.recipientsInFreezeRound);
90      claimableFunds = claimableFunds.sub(claimedFunds);
91      if(amount > claimableFunds) {
92          claimAmount = claimableFunds;
93      }
94      else {
95          claimAmount = amount;
96      }
97      reserve.fundsClaimed = reserve.fundsClaimed.add(claimAmount);
98      reserve.claimedRecipient[recipient] =
99          reserve.claimedRecipient[recipient].add(claimAmount);
100     return claimAmount;
```

Abbildung 5.6: Codebeispiel 4 - claimFromReserve Funktion

Sobald die Reserve eines Senders genutzt werden muss, wird sie für eine zuvor festgelegte Periode eingefroren (Zeile 81). Zudem wird gespeichert, wie viele Empfänger in der aktuellen Runde registriert sind (Zeile 82). Sind keine Empfänger in der Runde (Zeile 84) oder der Empfänger der Zahlung nicht registriert (Zeile 85) wird nichts von der Reserve an den Empfänger ausgezahlt. Jedem registrierten Empfänger steht der gleiche Anteil an Reserve zu (Zeile 89). Von diesem Anteil wird der Betrag, den der Empfänger bereits von der Reserve erhalten hat (Zeile 88), abgezogen (Zeile 90). Das Ergebnis ist der Betrag, den der Empfänger erhält.

Als letzte komplexere Funktion soll das Rückzahlen (Withdraw) des Deposits an den Sender selbst vorgestellt werden.

```
69  if(sender.deposit > 0) {
70      if(sender.withdrawBlock == 0){
71          sender.withdrawBlock = currentRound + UNLOCK_PERIOD;
72      }
73      else if(sender.withdrawBlock <= currentRound) {
74          if(sender.deposit > amount) {
75              sender.deposit = sender.deposit - amount;
76              sender.withdrawBlock = 0;
77              msg.sender.transfer(amount);
78              return amount;
79          }
80          else {
81              uint256 transferAmount = sender.deposit;
82              sender.deposit = 0;
83              sender.withdrawBlock = 0;
84              msg.sender.transfer(transferAmount);
85              return transferAmount;
86          }
87      }
88  }
89  return 0;
```

Abbildung 5.7: Codebeispiel 5 - withdrawDeposit Funktion

Die Auszahlung erfolgt aus Sicherheitsgründen in zwei Stufen. Mit dem ersten Aufrufen der Funktion kündigt der Sender an, dass er eine Auszahlung vornehmen möchte. In diesem Fall wird in seinem Sender Struct diese Information durch den Withdraw Block vermerkt. Der Withdraw Block gibt die Runde an, ab der der Sender von seinem Deposit abheben darf. Beim ersten Aufruf der Funktion wird auf die aktuelle Runde die Unlock Periode addiert (Zeile 71) und gespeichert. Der Sender muss nun die Unlock Periode abwarten und ruft anschließend die Funktion erneut auf. Ist nun der Withdraw Block kleiner oder gleich der aktuellen Runde, wird der gewünschte Betrag (amount) beziehungsweise maximal die Höhe des Deposits an den Sender übertragen (Zeile 84).

Durch die zwei Stufen und das Abwarten der Unlock Periode wird eine Front Running Attacke auf das Deposit verhindert. Ein Empfänger kann zu jeder Zeit auslesen, ab wann ein Sender auf sein Deposit zugreifen kann und davon abhängig entscheiden, ob er noch Tickets von ihm erhalten möchte. Wie Sender und Empfänger miteinander interagieren und Tickets austauschen wird im nächsten Kapitel beschrieben.

5.2 Client Seite

Die Client Seite beschreibt die Abläufe, die Off-Chain passieren. Hier interagieren der Sender und der Empfänger direkt miteinander und lesen Informationen aus den Smart

Contracts aus. Den Ticketaustausch zwischen den beiden Parteien und ihr Zugriff auf die Smart Contracts wird in der folgenden Abbildung dargestellt.

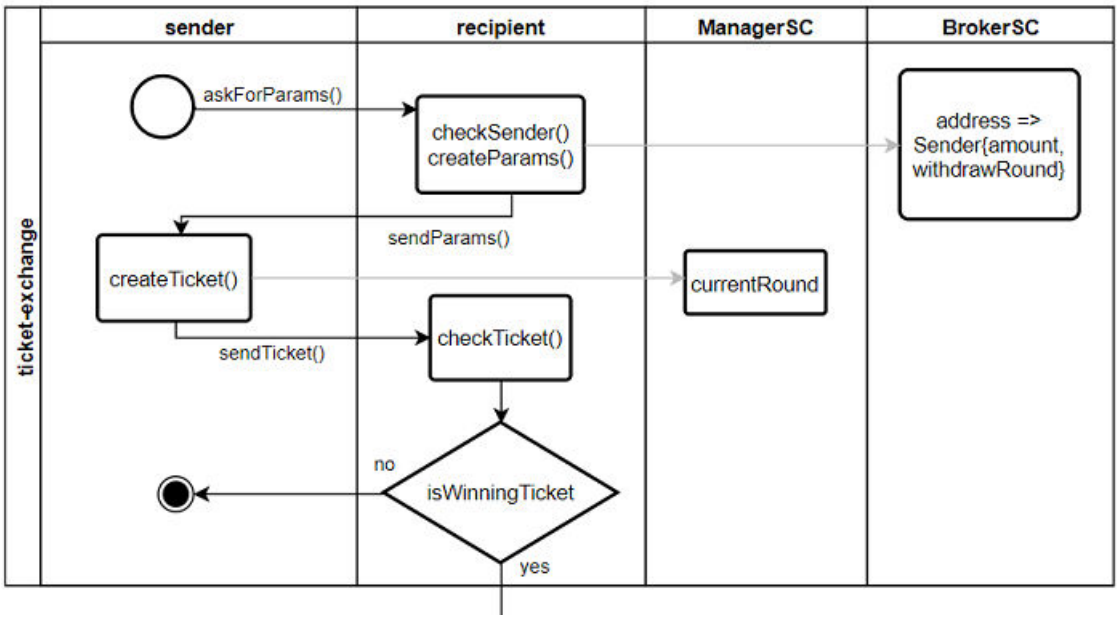


Abbildung 5.8: Flowchart 3 - Ticketexchange

Der Sender fragt zunächst bei dem Empfänger nach den Parametern für ein Ticket. Dadurch kann der Empfänger entscheiden, ob er mit diesem Sender interagieren möchte. Für diese Entscheidung überprüft er den Sender. Dabei kontrolliert er mithilfe der Informationen aus dem Broker Smart Contract, ob der Sender noch ausreichend Deposit hat und ob die Auszahlungsrunde (der withdrawBlock) entweder nicht gesetzt oder noch weit genug in der Zukunft liegt. Hat er sich dafür entschieden von dem Sender Tickets erhalten zu wollen, wählt er die Ticketparameter.

Die Ticketparameter, die durch den Empfänger vorgegeben werden, sind die Gewinnwahrscheinlichkeit, die Höhe der Auszahlung bei Gewinn und der Hash einer von ihm gewählten Zufallszahl. Diese Informationen speichert er, um sie später mit dem Ticket des Senders abgleichen zu können und übermittelt sie an den Sender. Dieser liest aus dem Manager Smart Contract die aktuelle Runde und deren Hash. Diese Information wird im auxData Feld des Tickets abgelegt. Außerdem werden die Adressen von Sender und Empfänger im Ticket festgehalten.

Sind alle Parameter der Tickets gesetzt, erstellt der Sender den Hash des Tickets und signiert ihn. Er schickt das Ticket, den Hash und die Signatur zu dem Empfänger. Dieser prüft, dass seine übergebenen Parameter genutzt wurden, die Signatur tatsächlich vom Sender kommt und zuletzt, ob es ein Winning Ticket ist. Ist dies der Fall, ruft er die bereits beschriebene redeemWinningTicket-Funktion des Broker Smart Contracts auf, um eine entsprechende Auszahlung zu bewirken. Außerdem kann er den Nutzer darüber

informieren und dieser daraufhin prüfen, ob er sein Deposit aufstocken sollte.

Zusammenfassend ist zu sagen, dass dieser Prototyp zeigt, dass probabilistische Mikrozahlungen mithilfe der Blockchain getätigt werden können. Es werden die typischen Funktionen eines probabilistischen Zahlungsschemas abgedeckt und eine grundlegende Sicherung gegen verschiedene Angriffsszenarien geboten. Außerdem bewirkt die Reduktion der On-Chain-Transaktionen auch ein Einsparen von Transaktionskosten. Eine Weiterentwicklung könnte den Fokus auf effizienteren Umgang mit Ressourcen legen. Dieser Prototyp ist ein möglicher Ausgangspunkt für weitere Untersuchungen in der Praxis über probabilistische Micropayments auf der Blockchain.

6 Fazit

Abschließend sollen die wichtigsten Erkenntnisse aus dieser Arbeit zusammengefasst werden. Zu Beginn der Arbeit wurden drei zentrale Themen genannt, die in dieser Arbeit vereint werden sollen: Blockchain, IoT und (Mikro-)Zahlungen. Davon ausgehend wurde nach vorhandenen zentralen und dezentralen Micropayment Ansätzen insbesondere probabilistischen Mircopayment-Konzepten recherchiert. Das Ergebnis ist eine Übersicht über die verschiedenen Arbeitsweisen, Vorteile und Herausforderungen der Ansätze.

Es wurden verschiedene Anforderungen für ein geeignetes Micropaymentsystem definiert und für verschiedene Ansätze analysiert:

- Ablauf ohne eine zentrale Instanz - Viele Konzepte bauen auf ein zentrales System auf. Eine praktikable Anpassung dieser für eine dezentrale Umgebung ist nur bei wenigen möglich. Die Recherche ergab jedoch auch, dass bereits Ansätze für dezentrale Systeme beziehungsweise Blockchain existieren.
- Ressourcenschonend, um den Einsatz in IoT-Geräten zu ermöglichen - Diese Anforderung steht bei den meisten Systemen nicht im Fokus. Dennoch bieten einige Ansätze einfache Abläufe, die dem Sender ein Arbeiten mit begrenzten Ressourcen ermöglicht.
- Absicherung des Systems gegen verschiedene Angriffsszenarien - Inwiefern die Systeme gegen die unterschiedlichen Angriffe abgesichert wurden, unterschied sich stark. Einige zentrale Ansätze überlassen den Sicherheitsfaktor der Bank, andere komplexe Systeme konnten ein sehr hohes Maß an Sicherheit bieten. Eine Herausforderung ist es, Sicherheit und Ressourcenschonung in einem System zu vereinen.
- Verringerung der Kosten pro Zahlungen durch den Einsatz des Systems - Eine Kostenreduktion pro Mikrozahlung ermöglichen die meisten Konzepte. Eine Einschränkung ergibt sich dabei teilweise durch hohe Kosten beim Aufsetzen des Systems.

Aus der Analyse unterschiedlicher Systeme konnten verschiedene Ansätze gesammelt werden, aus denen sich ein geeignetes Zahlungsschema ableiten lässt. Insbesondere ein Ansatz konnte als Inspiration für die Entwicklung eines Prototypen genutzt werden. Die Umsetzung basiert in vielen Teilen auf Streamflow. Grundlegend wurde mit diesem Prototypen ein probabilistisches Micropayment System auf einer Blockchain umgesetzt. Dabei konnte im Wesentlichen auch die Forderung nach einem sicheren System erfüllt werden. Die Komplexität des Systems stieg dabei zwar, dennoch ist der Umfang geringer als der einiger vorgestellter Ansätze.

Dadurch entstand in der Praxis ein funktionierendes Konzept, welches auch für den Einsatz bei IoT-Geräten geeignet ist. Der entstandene Prototyp kann für den Produktionseinsatz ausgebaut werden. Für die besonderen Ansprüche von Incubed ist der Prototyp noch nicht passend, da zu viele Interaktionen für eine Zahlung erfolgen müssen.

Diese Arbeit zeigt, dass vorhandene probabilistische Micropaymentsysteme aktuell auch für IoT-Geräte auf der Blockchain eingesetzt werden können, allerdings keine zufriedenstellende Lösung für die speziellen Anforderungen von Incubed bieten. Dennoch konnten viele Ansätze und ihre Vorteile ausgearbeitet werden und eventuell eine Grundlage für neue Ideen bilden. Vor allem der Ansatz der Poollösung bietet aus aktueller Sicht Potenzial und könnte weiterverfolgt werden. Eventuell bringt auch die anhaltende Weiterentwicklung der Blockchain-Technologie von sich aus effizientere und günstigere Zahlungen mit sich und macht damit den Einsatz spezieller Micropaymentverfahren obsolet.

Literaturverzeichnis

[1] The 7 Essential Technology Buzzwords of 2019. THE ESSENTIAL TECHNOLOGY BUZZWORDS YOU NEED TO KNOW FOR THIS YEAR (2019). Online verfügbar unter https://careers.gazprom-mt.com/blog/essential-technology-buzzwords-to-this-year/, zuletzt geprüft am 12.09.2019.

[2] Gartner (2018): Hype Cycle for Emerging Technologies, 2018. Online verfügbar unter https://blogs.gartner.com/smarterwithgartner/files/2018/08/PR_490866_5_Trends_in_the_Emerging_Tech_Hype_Cycle_2018_Hype_Cycle.png, zuletzt aktualisiert am 29.01.2019, zuletzt geprüft am 12.09.2019.

[3] Sepp, Christian; Brzoska Maike (2015): Die Geschichte des Geldes: Von der Muschel zur Kreditkarte. Bayrischer Rundfunk. Online verfügbar unter https://www.br.de/radio/bayern2/sendungen/radiowissen/soziale-politische-bildung/geld-geschichte-100.html, zuletzt geprüft am 12.09.2019.

[4] Dhillon, Vikram; Metcalf, David; Hooper, Max (2018): Blockchain enabled applications. Understand the blockchain ecosystem and how to make it work for you. USA: Apress.

[5] Giese, Philipp; Kops, Maximilian; Wagenknecht, Sven; Boer, Danny de; Preuss, Markus (2016): Die Blockchain Bibel. DNA einer evolutionären Technologie. Kleve: BTC-ECHO.

[6] Protschka, Florian; Heinze, Yannik (2018): Blockchain ABC. Von A wie Altcoin bis Z wie ZCash. 1. Ausgabe 2019. Berlin: Chainsulting UG.

[7] Ray, Shaan (2019): Merkle Trees. Online verfügbar unter https://hackernoon.com/merkle-trees-181cb4bc30b4, zuletzt aktualisiert am 02.08.2019, zuletzt geprüft am 12.08.2019.

[8] Rosic, Ameer (2016): What is Ethereum? [The Most Comprehensive Step-by-Step-Guide!]. Online verfügbar unter https://blockgeeks.com/guides/ethereum/, zuletzt aktualisiert am 13.08.2019, zuletzt geprüft am 13.08.2019.

[9] Nash, Gerald (2017): The Anatomy of ERC721. Online verfügbar unter https://medium.com/crypto-currently/the-anatomy-of-erc721-e9db77abfc24, zuletzt geprüft am 14.08.2019.

[10] Buterin, Vitalik; Wood, Gavin; Lubin, Joseph (2015): Ethereum White Paper [A Next-Generation Smart Contract and Decentralized Application Platform]. Online

verfügbar unter https://github.com/ethereum/wiki/wiki/White-Paper#ethereum, zuletzt geprüft am 14.08.2019.

[11] Antonopoulos, Andreas; Wood, Gavin (2018): Mastering Ethereum. 1st edition: O'Reilly Media, Inc.

[12] Agrawal, Gaurav (2019): Running a Parity Ethereum node in 2019. Online verfügbar unter https://medium.com/quiknode/running-a-parity-ethereum-node-in-2019-aedad24976e0, zuletzt geprüft am 15.08.2019.

[13] Sardan, Thibaut (2018): What is a light client and why you should care? Online verfügbar unter https://www.parity.io/what-is-a-light-client/, zuletzt aktualisiert am 14.08.2019, zuletzt geprüft am 14.08.2019.

[14] Light client protocol. Online verfügbar unter https://github.com/ethereum/wiki/wiki/Light-client-protocol, zuletzt geprüft am 15.08.2019.

[15] Kux, Steffen (2019): IN^3=Incubed. A Trustless, Stateless, INcentivized Remote Node Netzwork, Präsentation im Februar 2019

[16] Jentzsch, Christoph: DAPPCON 2018: Incubed - A Minimal Verification Client - Christoph Jentzsch (Slock It) - YouTube. Online verfügbar unter https://www.youtube.com/watch?v=_vodQubed2A, zuletzt geprüft am 15.08.2019.

[17] Kux, Steffen; Jentzsch, Christoph; Jentzsch, Simon; Depraz, Paul (2018): INCUBED. A trustless stateless incentivized remote node network. Version 0.1 (draft). Online verfügbar unter https://download.slock.it/whitepaper_incubed_draft.pdf, zuletzt geprüft am 15.08.2019.

[18] Robbert, Thomas; Priester, Anna; Roth, Stefan (2018): Micropayments im Erlösmodell digitaler Serviceleistungen. In: Manfred Bruhn und Karsten Hadwich (Hg.): Service Business Development. Wiesbaden: Springer Fachmedien Wiesbaden, S. 187–209.

[19] Dai (2004): Micropayment Token-Making Schemes. Online verfügbar unter http://www.cs.sjsu.edu/faculty/stamp/students/dai.html, zuletzt aktualisiert am 18.10.2004, zuletzt geprüft am 02.05.2019.

[20] Chi, Ellis (1997): Evaluation of micropayment schemes. Online verfügbar unter https://www.hpl.hp.com/techreports/97/HPL-97-14.pdf, zuletzt geprüft am 12.06.2019.

[21] Odlyzko, Andrew (2003): The Case Against Micropayments. Online verfügbar un-

ter http://www.dtc.umn.edu/~odlyzko/doc/case.against.micropayments.pdf, zuletzt geprüft am 02.05.2019.

[22] Cox, Benjamin; Tygar, J. D.; Sirbu, Marvin (1995): NetBill Security and Transaction Protocol. Proceedings of the 1st USENIX Workshop on Electronic Commerce 1995, S. 77–88. Online verfügbar unter https://people.eecs.berkeley.edu/~tygar/papers/Netbill_security_and_transaction_protocol.pdf, zuletzt geprüft am 19.06.2019.

[23] Bargeldloses Bezahlen - Online Shopping | PayPal DE. Online verfügbar unter https://www.paypal.com/de/home/, zuletzt geprüft am 21.08.2019.

[24] Möhring, Cornelia (2019): Paypal-Konto aufladen - so kommt Geld auf Ihren PayPal-Account. Online verfügbar unter https://www.heise.de/tipps-tricks/Paypal-Konto-aufladen-so-kommt-Geld-auf-Ihren-PayPal-Account-4265717.html, zuletzt aktualisiert am 04.01.2019, zuletzt geprüft am 21.08.2019.

[25] Wilusz, Daniel; Rykowski, Jarogniew (2013): The Architecture of Coupon-Based, Semi-off-Line, Anonymous Micropayment System for Internet of Things. Online verfügbar unter https://hal.archives-ouvertes.fr/hal-01348743/document, zuletzt geprüft am 21.10.2019

[26] Manasse, Mark (1995): The Millicent protocols for electronic commerce. Online verfügbar unter https://pdfs.semanticscholar.org/119f/0d57f2b4ef27221404d5145cb0828c70cecb.pdf, zuletzt geprüft am 21.10.2019

[27] Yang, Beverly; Garcia-Molina, Hector (2003): PPay: Micropayments for Peer-to-Peer Systems. Online verfügbar unter http://infolab.stanford.edu/~byang/pubs/ppay_extended.pdf, zuletzt geprüft am 02.08.2019.

[28] Yen, Sung-Ming; Chiou, Kuo-Zhe; Zhang, Je; Lee, Po-Han (2010): A New Peer-to-Peer Micropayment Protocol Based on Transferable Debt Token. In: Gavrilova M.L., Tan C.J.K., Moreno E.D. (eds) Transactions on Computational Science X. Lecture Notes in Computer Science, vol 6340. Springer, Berlin, Heidelberg

[29] Lin, Iuon-Chang; Hwang, Min-Shang; Chang, Chin-Chen (2005): The General PayWord: A Micro-payment Scheme Based on n-dimension One-way Hash Chain. Online verfügbar unter https://doi.org/10.1007/s10623-003-1162-6, zuletzt geprüft am 02.08.2019

[30] Liu, Yining; Yan, Jihong: A lightweight micropayment scheme based on Lagrange interpolation formula - Liu - 2013 - Security and Commu-

nication Networks - Wiley Online Library. Online verfügbar unter https://onlinelibrary.wiley.com/doi/full/10.1002/sec.643, zuletzt geprüft am 13.05.2019.

[31] Rivest, Ronald L.; Shamir, Adi (2001): PayWord and MicroMint: Two simple micropayment schemes. Online verfügbar unter https://people.csail.mit.edu/rivest/RivestShamir-mpay.pdf, zuletzt geprüft am 02.05.2019.

[32] Bayyapu, Praneetha R.; Das, Manik Lal (2009): An Improved and Efficient Micro-payment Scheme. In: J. theor. appl. electron. commer. res. 4 (1). Online verfügbar unter http://www.jtaer.com/statistics/download/download.php?co_id=JTA20090107, zuletzt geprüft 21.10.2019

[33] Wan, Zhi-Guo; Deng, Robert H.; Lee, David; Li, Ying (2019): MicroBTC: Efficient, Flexible and Fair Micropayment for Bitcoin Using Hash Chains. In: J. Comput. Sci. Technol. 34 (2), S. 403–415.

[34] Liu, Shanhong (2019): Bitcoin blockchain size 2010-2019, by quarter. Online verfügbar unter https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/, zuletzt aktualisiert am 01.07.2019, zuletzt geprüft am 02.08.2019.

[35] Green, Matthew; Miers, Ian (2017): Bolt: Anonymous Payment Channels for Decentralized Currencies. In: Bhavani Thuraisingham (Hg.): Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. the 2017 ACM SIGSAC Conference. Dallas, Texas, USA, 30.10.2017 - 03.11.2017. New York, NY: ACM, S. 473–489. Online verfügbar unter https://acmccs.github.io/papers/p473-greenA.pdf, zuletzt geprüft am 02.05.2019.

[36] Decker, Christian; Wattenhofer, Roger (2015): A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels, zuletzt geprüft am 02.05.2019.

[37] Horne, Liam (2017): Generalized State Channels on Ethereum – L4 blog – Medium. https://www.facebook.com/medium. Online verfügbar unter https://medium.com/l4-media/generalized-state-channels-on-ethereum-de0357f5fb44, zuletzt geprüft am 07.05.2019. https://www.facebook.com/hackernoon

[38] van Wirdum, Aaron (2016): Understanding the Lightning Network. Online verfügbar unter https://bitcoinmagazine.com/articles/understanding-the-lightning-network-part-building-a-bidirectional-payment-channel- 1464710791, zuletzt geprüft am 08.07.2019.

[39] McCorry, Patrick; Möser, Malte; Shahandasti, Siamak F.; Hao, Feng

(2016): Towards Bitcoin Payment Networks. Online verfügbar unter http://homepages.cs.ncl.ac.uk/patrick.mccorry/paymentnetworks.pdf, zuletzt geprüft am 24.06.2019.

[40] Poon, Joseph; Dryja, Thaddeus: The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Online verfügbar unter https://lightning.network/lightning-network-paper.pdf, zuletzt geprüft am 03.07.2019.

[41] µRaiden: Micropayments for Ethereum – Hacker Noon (2017). Online verfügbar unter https://hackernoon.com/%C2%B5raiden-micropayments-for-ethereum-f0756cd400b3, zuletzt aktualisiert am 19.09.2017, zuletzt geprüft am 26.06.2019.

[42] What is the Raiden Network? (2019). Online verfügbar unter https://raiden.network/101.html, zuletzt aktualisiert am 18.06.2019, zuletzt geprüft am 03.07.2019.

[43] Butler, Alexander (2018): An introduction to Plasma – Hacker Noon. https://www.facebook.com/hackernoon. Online verfügbar unter https://hackernoon.com/plasma-8bba7e1b1d0f, zuletzt geprüft am 02.05.2019.

[44] Horne, Liam (2018): What is Plasma? Plasma Cash? – cryptoeconomics.study – Medium. https://www.facebook.com/medium. Online verfügbar unter https://medium.com/crypto-economics/what-is-plasma-plasma-cash-6fbbef784a, zuletzt geprüft am 07.05.2019.

[45] Poon, Joseph; Buterin, Vitalik (2017): Plasma: Scalable Autonomous Smart Contracts. Online verfügbar unter https://plasma.io/plasma.pdf, zuletzt aktualisiert am 09.08.2017, zuletzt geprüft am 02.05.2019.

[46] Die Libra Association (2019). Online verfügbar unter https://libra.org/de-DE/association/, zuletzt geprüft am 30.07.2019.

[47] Hosp, Julian: Die Kryptoshow: Bitcoin & Blockchain. Podcast. Online verfügbar unter https://open.spotify.com/episode/60gbeXIElseJOo3bwWBI4k, zuletzt geprüft am 02.08.19.

[48] SECBIT (20.06.19): 'Move' Programming Language: The Highlight of Libra. Online verfügbar unter https://hackernoon.com/move-programming-language-the-highlight-of-libra-122a910d6e0f, zuletzt aktualisiert am 27.07.2019, zuletzt geprüft am 30.07.2019.

[49] Amsden, Zachary; Arora, Ramnik; Bano, Shehar; Baudet, Mathieu; Blackshear, Sam; Bothra, Abhay et al. (2019): The Libra Blockchain. Online verfügbar unter

https://developers.libra.org/docs/assets/papers/the-libra-blockchain.pdf, zuletzt geprüft am 30.07.2019.

[50] Owen, Laura Hazard (2019): What will Libra, Facebook's new cryptocurrency, mean for news? Online verfügbar unter https://www.niemanlab.org/2019/06/what-will-libra-facebooks-new-cryptocurrency-mean-for-news/, zuletzt geprüft am 30.07.2019.

[51] Schomer, Audrey (2019): How Facebook's Libra could help publishers monetize with micropayments. businessinsider. Online verfügbar unter https://www.businessinsider.com/libra-could-enable-micropayments-for-publishers-2019-6?IR=T, zuletzt aktualisiert am 20.06.2019, zuletzt geprüft am 30.07.2019.

[52] Pass, Rafael; shelat, abhi (2016): Micropayments for Decentralized Currencies. Online verfügbar unter https://eprint.iacr.org/2016/332.pdf, zuletzt geprüft am 02.05.2019.

[53] Simonsson, Gustav (2017): Ethereum Probabilistic Micropayments – Gustav Simonsson – Medium. Online verfügbar unter https://medium.com/@gustav.simonsson/ethereum-probabilistic-micropayments-ae6e6cd85a06, zuletzt geprüft am 02.05.2019.

[54] Nanopayments - Bitcoin Wiki (2018). Online verfügbar unter https://en.bitcoin.it/wiki/Nanopayments, zuletzt aktualisiert am 16.05.2018, zuletzt geprüft am 19.07.2019.

[55] Hamad, Ahmed M.; Kouta, Mohamed.; Afify, Yasmine M. (2006): Evaluation of Probabilistic Payment Systems. In: Joint conference. 8th IEEE International Conference on E-Commerce and Technology (CEC 2006). Online verfügbar unter https://www.researchgate.net/publication/221543037_Evaluation_of_Probabilistic_Payment_Systems, zuletzt geprüft am 21.10.2019

[56] Hashem, Mohamed; Kouta, Mohamed; Hamad, Ahmed M.; Afify, Yasmine M. (2005): New probabilistic scheme with Variable Sized Micropayments. Online verfügbar unter http://www.setit.rnu.tn/last_edition/setit2005/electronique/293.pdf, zuletzt geprüft am 02.08.2019.

[57] Rivest, Ronald L. (2002): Micropayments Revisited. Online verfügbar unter https://people.csail.mit.edu/rivest/pubs/MR02a.slides.pdf, zuletzt geprüft am 02.05.2019.

[58] Rivest, Ronald L.; Micali, Silvio : Micropayments Revisited. Online verfügbar un-

ter https://people.csail.mit.edu/rivest/MicaliRivest-MicropaymentsRevisited.pdf, zuletzt geprüft am 02.05.2019.

[59] Rivest, Ronald L. (2004): Peppercoin Micropayments. In: Juels A. (eds) Financial Cryptography. FC 2004. Lecture Notes in Computer Science, vol 3110. Springer, Berlin, Heidelberg. Online verfügbar unter https://people.csail.mit.edu/rivest/Rivest-PeppercoinMicropayments.ps, zuletzt geprüft am 21.10.2019

[60] Rivest, Ronald L.: Peppercoin Micropayments. Online verfügbar unter https://people.csail.mit.edu/rivest/pubs/Riv04c.slides.slides.pdf, zuletzt geprüft am 20.08.2019.

[61] Lipton, Richard J.; Ostrovsky, Rafail (1998): Micro-Payments via Efficient Coin-Flipping. Online verfügbar unter http://web.cs.ucla.edu/~rafail/PUBLIC/33.pdf, zuletzt geprüft am 02.05.2019.

[62] Daphne, Lukas (2004): Improvement in Probabilistic Micropayment Schemes. Online verfügbar unter https://uwaterloo.ca/combinatorics-and-optimization/sites/ca.combinatorics-and-optimization/files/uploads/files/lucas-d.pdf, zuletzt geprüft am 15.05.2019.

[63] Jarecki, Stanislaw; Odlyzko, Andrew (1997): An efficient micropayment system based on probabilistic polling. In: Hirschfeld R. (eds) Financial Cryptography. FC 1997. Lecture Notes in Computer Science, vol 1318. Springer, Berlin, Heidelberg. Online verfügbar unter https://doi.org/10.1007/3-540-63594-7_77, zuletzt geprüft am 06.06.2019.

[64] Salamon, David L.; Simonsson, Gustav; Freeman, Jay; Fox, Brian J.; Vohaska, Brian; Bell, Stephen F.; Waterhouse, Steven: Orchid: Enabling Decentralized Network Formation and Probabilistic Micro-Payments. Online verfügbar unter https://www.orchid.com/assets/whitepaper/whitepaper.pdf, zuletzt geprüft am 02.05.2019.

[65] Output, Transaction Output, TxOut - Bitcoin Glossary (2019). Online verfügbar unter https://bitcoin.org/en/glossary/output, zuletzt aktualisiert am 12.08.2019, zuletzt geprüft am 20.08.2019.

[66] Caldwell, Mike (2012): Sustainable nanopayment idea: Probabilistic Payments. Online verfügbar unter https://bitcointalk.org/index.php?topic=62558, zuletzt geprüft am 07.05.2019.

[67] Dodis, Yevgeniy; Yampolskiy, Aleksandr (2005): A Verifiable Random Function With

Short Proofs and Keys. Online verfügbar unter https://cs.nyu.edu/~dodis/ps/short-vrf.pdf, zuletzt geprüft am 21.08.2019.

[68] Goldberg; Papadopoulus (2017): Verifiable Random Functions (VRFs). Online verfügbar unter https://tools.ietf.org/id/draft-goldbe-vrf-01.html, zuletzt aktualisiert am 01.07.2017, zuletzt geprüft am 15.05.2019.

[69] Chiesa, Alessandro: Decentralized Anonymous Micropayments - YouTube. Online verfügbar unter https://www.youtube.com/watch?v=hGl58A2V-_8, zuletzt geprüft am 13.05.2019.

[70] Chiesa, Alessandro; Green, Matthew; Liu, Jingchen; Miao, Peihan; Miers, Ian; Mmishra, Pratyush (2016): Decentralized Anonymous Micropayments. Online verfügbar unter https://eprint.iacr.org/2016/1033.pdf, zuletzt geprüft am 02.05.2019.

[71] livepeer/go-livepeer. Online verfügbar unter https://github.com/livepeer/go-livepeer/blob/master/cmd/livepeer/livepeer.go, zuletzt geprüft am 08.05.2019.

[72] Fu, Yondon (2019): Streamflow: Probabilistic Micropayments – Livepeer Blog – Medium. Online verfügbar unter https://medium.com/livepeer-blog/streamflow-probabilistic-micropayments-f3a647672462, zuletzt geprüft am 02.05.2019.

[73] Livepeer. Online verfügbar unter https://github.com/livepeer, zuletzt geprüft am 20.08.2019.

[74] Lotto-Tippgemeinschaft: gemeinsam Lotto spielen. Online verfügbar unter https://lotto.web.de/tippgemeinschaften/, zuletzt geprüft am 02.09.2019.

[75] Cardstack Overview Document V104 (2018). Online verfügbar unter https://cardstack.com/assets/media/cardstack-white-paper-v1.0.4.pdf, zuletzt geprüft am 02.09.2019.

[76] Abdel-Rahman, Hassan (2018): Scalable Payment Pools in Solidity. Paying a lot of people without paying a lot of gas. Online verfügbar unter https://medium.com/cardstack/scalable-payment-pools-in-solidity-d97e45fc7c5c, zuletzt geprüft am 02.09.2019.

[77] Thong, Justin; Tse Chris (2018): Cardstack Reward Model: Proportional Attribution and Allocation Model in a Dependency Network with Intervention in the Local Scope (PAAM). Online verfügbar unter https://cardstack.com/assets/media/cardstack-reward-model-v1.0.pdf, zuletzt geprüft am 02.09.2019.

[78] bitcoinj (2019). Online verfügbar unter https://bitcoinj.github.io/#introduction, zuletzt aktualisiert am 04.05.2019, zuletzt geprüft am 21.08.2019.

[79] Robbert, Thomas; Priester, Anna; Roth, Stefan (2018): Micropayments im Erlösmodell digitaler Serviceleistungen. In: Manfred Bruhn und Karsten Hadwich (Hg.): Service Business Development. Wiesbaden: Springer Fachmedien Wiesbaden, S. 187–209.

[80] MixinTicketBrokerCore.sol (2019). Online verfügbar unter https://github.com/livepeer/protocol/blob/pm/contracts/pm/mixins/ MixinTicketBrokerCore.sol, zuletzt aktualisiert am 19.09.2019, zuletzt geprüft am 08.11.2019

Tools

Ganache:

https://github.com/trufflesuite/ganache-cli, verfügbar am 06.11.2019

Visual Studio Code:

https://code.visualstudio.com/, verfügbar am 06.11.2019

Visiual Paradigm:

https://www.visual-paradigm.com/, verfügbar am 06.11.2019

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.



Mittweida, 08. November 2019