Proof of Useful Work: Eine Nützliche Variante von Proof of Work

Jean-Marc Hendrikse

Abstract

Der Proof of Work-Mechanismus ist ein sehr wichtiger Bestandteil für die Konsensbildung und die Sicherheit vieler Kryptowährungen. Das erste und bekannteste Beispiel, das diesen Mechanismus in der Blockchain umsetzt ist das dezentrale Peer-to-Peer Geldsystem Bitcoin. Bei Proof of Work werden intensive Berechnungen einzig und allein für den Nachweis durchgeführt, dass Abreit bzw. Rechenleistung aufgewendet wurde. Der Stromverbrauch hat somit keinen weiteren Nutzen für die die Gesellschaft und wird deswegen häufig als "Proof of Waste" bezeichnet. Um dem entgegenzuwirken gibt es Ansätze, ein Proof of Work zu entwickeln, das Berechnungen mit einem Mehrwert für die Menschheit durchführt, sogenannte Proofs of Useful Work.

1 Einführung

Bitcoin [Naka08] gilt als die erste erfolgreiche digitale Geldwährung und dessen Aufmerksamkeit steigt stetig an. Seinen riesen Erfolg verdankt die Währung der zugrundeliegenden Technologie, Blockchain. Durch die wachsende Aufmerksamkeit, die vorallem durch positive Meldungen aus den Medien vorangetrieben wird [1], wachsen die Teilnehmer der Blockchain immens an. Anders als es jedoch bei zentralen Banken in der heutigen Gesellschaft der Fall ist, liegen die Coins von Bitcoin nirgendwo zentral in physischer Form in einem Institut, sondern werden von Computersystemen erzeugt [Shan17] . Im Bitcoin-Jargon spricht man auch von "Mining". Es ist sogar für jede Person mit entsprechender Hardware möglich selbst "Mining" zu betreiben, indem Rechenleistung investiert wird. Das auf der Blockchain basierende elektronische Geldsystem hat hierfür ein System entwickelt, Benutzern eine Vergütung (engl.: *reward*) in Form von Bitcoins zu überlassen, wenn diese Rechenleistungen zum Lösen einer bestimmten Aufgabe aufwenden. Gleichzeitig gilt der Coin als Nachweis darüber, dass Arbeit für eine bestimmte Aufgabe aufgewendet wurde [Naka08]????. Dieser Mechanismus wird als Proof of Work bezeichnet. Proof of Work ist das Herzstück von Bitcoin und der Grund für dessen Wert und Sicherheit vor Angriffen wie Double-Spending. Damit die Berechnung einer Challenge akzeptiert werden kann, muss allerdings sehr viel Rechenleistung investiert werden. So viel, dass man hier von 2,2 bis 9,7 Tonnen CO2 pro Bitcoin zählt. Verglichen mit der Zahl von 1,48 Tonnen CO2 pro Passagier auf einem Flug von San Francisco nach Bonn ist das extrem viel. Die Berechnungen nehmen auch mit wachsender Blockchain zu, da die Schwierigkeit der Challenges zunimmt. Das Problem das hinter dieser Berechnung steckt ist, dass die Lösung danach nicht weiterverwendet werden kann und somit ein Abfallprodukt ist. Das Problem der sogenannten "nutzlosen" Berechnungen von Bitcoin wird oft in verschiedenen Literaturen zitiert [**?**]. [Ande13] und [**?**] bezeichnen den Mechanismus hinter Bitcoin sogar als "Disaster für die Umwelt".
Hier könnte sich nun die Frage stellen, wieso Proof of Work nicht einfach ersetzt wird? Die

[1] , dass ein Bitcoin die 20 Tausend Dollar marke erreicht hat und viele reich geworden sind Datum und bedeutung

Problematik liegt klar auf der Hand: Es hat trotz des hohen Stromverbrauchs sehr viele Vorteile. Zum einen ist es für seine Zwecke im Hinblick auf Sicherheit sehr erfolgreich und zum anderen sehr robust. Ein natürlicher Ansatz, der dicht gefolgt von der Ersetzungsstrategie kommt, wird im Zuge dieser Ausarbeitung beschrieben und diskutiert: Ersetzen der nutzlosen Berechnungen durch nützliche, die der Gesellschaft einen Mehrwert liefern. Diese Variante des Proof of Work wird auch als Proof of Useful Work [BRSV17] bezeichnet. Mithilfe dieser Variante soll letztlich sichergestellt werden, dass jede investierte Rechenleistung auch einen Nutzen für die Gesellschaft hat, wodurch der Stromverbrauch global gesenkt werden kann. [BRSV17] stellt hierfür ein Framework für ein Proof of Useful Work Schema vor, das für Probleme, die auf NP-schwere Probleme wie Orthogonal Vectors Problem, 3SUM oder All Pairs Shortest Path reduzierbar sind [Will15], geeignet ist.
In Abschnitt 2 dieser Ausarbeitung werden zunächst die Grundlagen von Blockchain und Bitcoin eingeführt. Wobei vor allem Wert auf die grundlegende Technologie der Blockchain anhand des Beispiels von Bitcoin gelegt wird. Weiterhin werden erste Umsetzungen von useful Proof of Work Protokollen durch Primecoin [King13] und Permacoin [MJSP+14].
Im zweiten Abschnitt wird das Framework zu Proofs of Useful Work nach [BRSV17] vorgestellt. Anhand lassen sich sehr viele Probleme aus der theoretischen Informatik, die beispielsweise auf das Orthognonal Vecotr-Problemreduzierbar sind, in die Berechnung eines Proof of Work integrieren. Weiter werden Kandidaten für ein Proof of Useful Work Schema untersucht und evaluiert. Am Ende dieses Kapitels soll ein neues Blockchain-Schema mit einem Proof of Useful Work Mechanismus das alte Schema von Bitcoin ersetzen können.

Abschließend wird ein Ausblick gegeben.

2 Grundlagen

TODO In diesem Abschnitt werden die Grundlagen der Blockchain und dem elektronische Geldsystem Bitcoin erläutert. Diese bilden das Fundament für spätere Definitionen des Proof of Useful Work. Zunächst wird eine informelle Defenition darüber gegeben, wie der Begriff Nutzen (*engl.: Usefulness*) in dieser AUsarbeitung verwendet wird.

2.1 Informelle Definition von Usefulness

Unterscheidung in gesellschaftlichen und privaten Nutzen. Unter Nutzen (Usefulness) wird nachfolgend der Benefit, den man aus dem Gebrauch einer Sache ziehen kann verstanden[2]. Dadruch, dass es sich bei der Blockchain um ein verteiltest System in einer Community handelt, wird im Zuge dieser Arbeit die Sichtweise des gesellschaftlichen Nutzens betrachtet. Mehrere Akteure handeln auf der Blockchain. Wieder unterscheide zwischen Sozialem oder wirtschaftlichen Nutzen. Die Betrachtung des gesellschaftlichen Nutzens führt sehr schnell zu einer Philosophischen Frage, auf die ich hier nicht genauer eingehende werde, sondern nur eine Definition ablegen, wie Nutzen im Rahmen meiner Arbeit gewählt wird.

2.2 Blockchain und Bitcoin

Im Allgemeinen handelt es sich bei der Blockchain um ein dezentrales öffentliches Hauptbuch (engl.: `public ledger`). Um bislang Transaktionen durchführen zu können, vertrauten Kunden „vertrauenswürdigen“ Instanzen, beispielsweise Banken, denen sie ihr Geld oder Daten zur Verwaltung anvertrauten. Diese dritten Instanzen besitzen einen permanenten Zugriff auf

[2] https://de.wiktionary.org/wiki/Nutzen

privates Geld und Eigentum und können jeder Zeit auf diese zugreifen. Probleme entstehen dann wenn sich eine dritte Instanz fehlverhält und den Zugang zu Daten oder Geld missbrauchen. Andererseits kann auch ein kompromitierter Angreiff auf das zentrale System dazu führen, dass private Daten oder Geld geändert werden oder gestohlen werden können. Mithilfe der Blockchain-Technologie wird nun eine neue Möglichkeit geschaffen, vom zentralen Gedanken der Systeme auf ein dezentrales System umzusteigen. Allgemein formuliert handelt es sich bei der Blockchain um eine dezentrale Datenbank in einem Peer-to-Peer Netzwerk von Computersystemen. Was der Inhalt der Daten auf dieser Datenbank ist, spielt in erster Linie keine Rolle. Es kann sich sowohl um Transaktionsdaten einer Überweisung, digitale Ereignisse oder um Grundbucheinträge handeln. Anders als es bei herkömmlichen Datenbanken jedoch der Fall ist, wird die Blockchain nicht auf einem einzelnen zentralen Server, sondern verteilt auf mehreren Rechnern eines großen Netzwerks, den Netzwerkknoten, gehalten. Jeder Teilnehmer dieses Netzwerks erhält bei Eintritt in die Blockchain eine lokale Kopie von allen Einträgen.

Abbildung 1: Vom Zentralen Gedanken eines System (links) zum dezentralen System (rechts).

In der Blockchain wird jede Transaktion zu einem Block zusammengefasst und mittels kryptographischer Berechnungen mit anderen Transaktionsblöcken zu einer Kette, der „Block-Chain“, verbunden. Ein einzelner Block besteht sowohl aus einem Zeitstempel, Transaktionsdaten sowie aus einem kryptographischen Hash des vorhergehenden gültigen Blocks. Bei der Neuaufnahme eines neuen Blockes in die Blockchain muss per Konsens die Mehrheit aller Teilnehmer des Systems verifizieren, dass dieser Block gültig ist, d.h. dass diese Transaktion tatsächlich stattgefunden hat. Nur und nur dann kann der neue Block in die Blockchain aufgenommen werden. Dadurch dass jeder Teilnehmer eine Kopie der aktuellen Blockchain-Historie erhält, kann die lokale Historie mit der neuen abgeglichen und verifiziert werden. So kann eine mehrheitliche Einigung zwischen den Knoten geschaffen werden. In der Blockchain wird immer die gesamte Historie an Transaktionen, die jemals durchgeführt wurden, chronologisch linear erweitert. Die kryptographische Verkettung von Transaktionen bildet das Fundament vieler Kryptowährungen. Das wohl bekannteste Beispiel ist die digitale Währung Bitcoin, die im November 2008 mit dem Titel „Bitcoin: A Peer-to-Peer Electronic Cash System”1 von einer unbekannten Einzelperson oder Gruppe mit dem Pseudonym Satoshi Nakamoto entwickelt wurde. Viele bis dahin vorangehende Digitale Währungen scheiterten daran, dass sie durch eine zentrale Instanz verifiziert werden mussten, um das Double Spending Problem zu lösen. Mithilfe der Blockchain-Technologie konnte Nakamoto einen Ansatz beschreiben, der eben dieses Problem löste.

2.2.1 Konsensmechanismus

In zentralisierten Organisationen werden Entscheidungen meistens getroffen, indem eine zentrale Instanz die Entscheidung trifft. Da in der Blockchain keine zentrale Entschiedungsinstanz existiert bzw. auf diese bewusst verzichtet werden soll, müssen Entscheidungen anders getroffen werden. Im Fall von Bitcoin wird ein mehrheitlicher Konsens getroffen, indem ein Konsensmechanismus eingeführt wird. Der Konsensmechanismus bezeichnet dabei einen Algo-

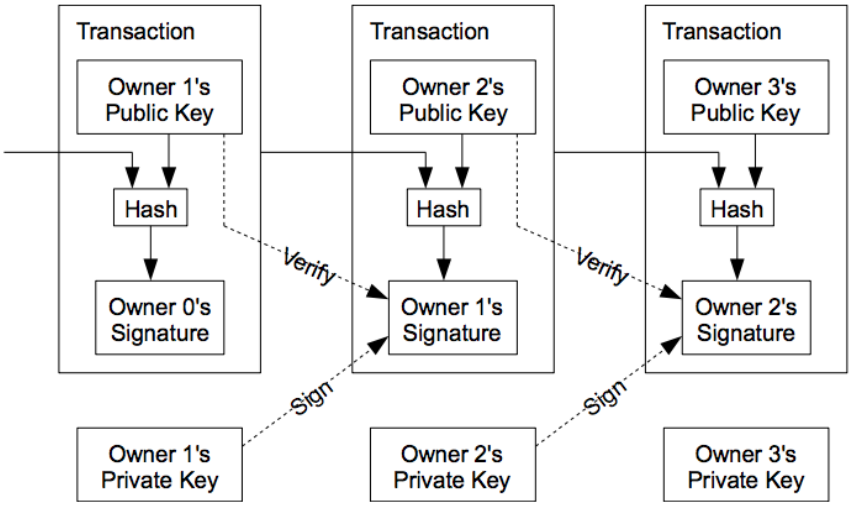

Abbildung 2: Dies ist die Bildunterschrift

rithmus, der eine Einigung über den Status des Netzwerks zwischen den beteiligten (meistens anonymen) Netzwerkknoten schafft. Dies ist die Basis der Blockchain Technologie. Dabei wird sichergestellt, dass alle Teilnehmer eine identische Kopie der Blockchain besitzen. Wie bereits im vorangegangenen Kapitel erwähnt erhöht der Konsensmechanismus die Sicherheit erheblich, da eine Manipulation von Daten ausgeschlossen ist, solange ein Angreifer nicht über 50% an Stimmen (in Form von Rechenleistung siehe nächstes Kapitel) erhält. Der Hauptgedanke der dahinter steckt ist, dass durch einen dezentralen Konsens-Mechanismus ein zentrale Kontrollinstanz zur Integritätsbestätigung obsolet wird. Es gibt verschiedene Konsensmechanismen wie Proof of Stake, Proof of Space[ABFG14, DFKP15, KK14], Proof of Work und viele mehr. Letzterer Konsensmechanismus wird im Nachfolgenden Abschnitt detaliert betrachtet.

2.2.2 Proof of Work

Das Grundprinzip von Proof-of-Work basiert auf der Idee, dass Miner im Netzwerk nachweisen müssen, dass sie einen gewissen Aufwand in Form von Arbeit erbracht haben. Dabei wird Arbeit in Form von Energieverbrauch gemessen. Durch einen konkurrierenden Wettstreit untereinander versuchen verschiedene Miner ein extrem schweres "kryptographisches Puzzle" zu lösen. Der erste Miner, der dieses Puzzle löst gewinnt und kann den neuen Block in die Blockchain anfügen. Damit Miner einen Anreiz bekommen an diesem Wettstreit teilzunehmen, bekommt jeder Gewinner einen Reward, im Fall von Bitcoin sind es 12.5 neu erstellte Bitcoins und eine kleine Transaktionsgebühr. [https://www.coindesk.com/short-guide-blockchain-consensus-protocols/]

Proof of Work ist keine Neuentwicklung von Bitcoin [Naka08], sondern wurde bereits früher gegen Denial of Service Attacken und für den E-Mail-Versand als Mechanismus gegen "Spam" eingesetzt [BRSV17] [DwNa92]. Um eine E-Mail zu versenden, musste der Sender vereinfacht ausgedrückt zusätzliche Berechnungen durchführen. Da die Maschinen in ihrer Rechenkapazität 1992 beschränkt waren, ist es notwendig gewesen, hohe Investitionen zu tätigen, um die Berechnungen durchzuführen und letztlich E-Mails zu versenden. Dadurch sollten die Kosten zum Versenden von E-Mails hoch genug sein, dass sich der Versand für "Spammer" nicht mehr lohnt. Der Empfänger einer E-Mail konnte nämlich auf einfache Art prüfen, ob die Berechnungen korrekt durchgeführt wurden. Die Berechnungen für den Sender hingegen waren mit einem gewissen Aufwand verbunden.

Definition 1. *In [BRSV17] werden drei für das Proof of Work wesentliche Algorithmen vorgestellt:*

1. `Gen`(1^n)*: randomisierter Algorithmus in* $\mathcal{O}(n)$*, der eine Challenge* c *schnell und effizient erzeugt.*
2. `Solve`(c)*: Algorithmus, der eine zufällig erzeugte Challenge* c *entgegen nimmt und daraus eine Lösunge* s *findet und ausgibt. Eine notwendige Bedingung ist, dass* s *schwer zu finden ist.*
3. `Verify`(c, s)*: Algorithmus in* $\mathcal{O}(n)$*, der schnell und effizient die Lösung* s *für die challenge* c *verifiziert.*

Im späteren Abschnitt 3 wird auf Grundlage obiger Defintion 1 eine Modifizierung des Proof of Work Protokolls im Hinblick auf Usefulness nach [BRSV17] beschrieben.
Besonders auffällig ist, dass die Algorithmen `Gen` und `Verify` eine lineare Laufzeit als obere Schranke besitzen und `Solve` möglichst schwer eine Lösung für eine zufällig generierte Challenge finden soll. Man spricht an dieser Stelle auch von Hardness (dt.: Schwere) [Will15]. Für die Laufzeit von `Solve` wird eine Zeit $t(n)$ vorgegeben. Das Ziel ist es, dass jede richtige Lösung eines Miners, die durch den `Solve`-Algorithmus berechnet wird, von `Verify` akzeptiert wird. Alelrdings soll `Verify` mit möglichst hoher Wahrscheinlichkeit alles ablehnen, das die Zeit von $t(n)^{1-\varepsilon}$ für jedes $\varepsilon > 0$ unterschritten hat.[BRSV17]
Insgesamt ergibt sich also für ein Problem, das sich für ein Proof of Work Schema eignen soll, dass es so schwer sein soll, dass eine Lösung schwer zu finden ist und falls eine Lösung gefunden wird, soll diese schnell und effizient verifizierbar sein. Im Fall von Bitcoin wird als Challenge ein SHA256-Hash berechnet, der eine gewisse Anzahl an führenden Nullen enthält.

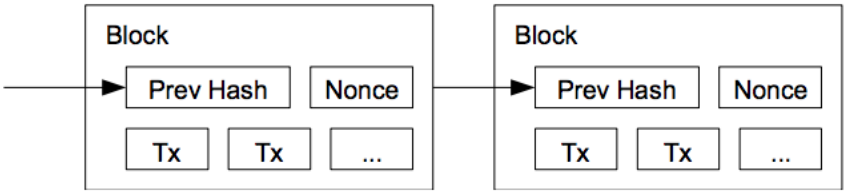

Abbildung 3: Dies ist die Bildunterschrift

Die Idee hinter der Blockchain ist, ein dezentrales Transaktionsbuch zu speichern. Daraus ergibt sich ein Vertrauensproblem, da viele Teilnehmer des Netzwerks in der Regel auf eine zentrale „trusted party“ vertrauen müssen. Um diesen unerwünschten Nebeneffekt zu umgehen, nutzt man Konsens-Algorithmen, die es ermöglichen, dass jeder Teilnehmer prüfen kann, ob seine Blockchain den Regeln entspricht.

2.2.3 Difficulty

TODO Im Zusammenhang von PoW ergibt sich die Difficulty aus der Schwierigkeit, den gewünschten Output der Hashfunktion zu finden. Im Fall von Bitcoin wird bspw. vorgegeben, wie viele Nullen der Output am Anfang des Strings besitzen muss. Je mehr Nullen gefordert sind, desto schwieriger wird es schließlich, den Output zu finden. Das lässt sich leicht mit einem Lottospiel veranschaulichen: Zwei Richtige zu treffen ist wesentlich einfacher, als Sechs Richtige der vorgegebenen Zahlen zu treffen.

2.2.4 Sicherheit

TODO PoW's security relies on the principle that no entity should gather more than 50% of the processing power because such an entity can effectively control the system by sustaining the longest chain. We now briefly outline known attacks on existing PoW-based blockchains. First, an adversary can attempt to double-spend by using the same coin(s) to issue two (or more) transactions—thus effectively spending more coins than he possesses. Recent studies have shown that accepting transactions without requiring blockchain confirmations is insecure [23]. The more confirmations a transaction obtains, the less likely this transaction will be reversed in the future. Second, miners might attempt to perform selfish mining [15] attacks in order to increase their relative mining share in the blockchain, by selectively withholding mined blocks and only gradually publishing them [15, 32]. Recent studies show that, as a result of these attacks, a selfish miner equipped with originally 33% mining power can effectively earn 50% of the mining power. Double-spending attacks and selfish mining can be alleviated if all nodes in the blockchain system are tightly synchronised. Note that, in addition to network latency, synchronisation delays can be aggravated due to eclipse attacks [18,19] where an adversary creates a logical partition in the network, i.e., provides contradicting block and transaction information to different blockchain network nodes.

Erwähne unbedingt double spending!

2.2.5 The Byzantine Fault Tolerance

TODO Als byzantinische Fehler bezeichnet man in der Informationstechnik Fehler, bei denen sich ein System beliebig falsch verhalten kann. Beispielsweise schickt ein Server gelegentlich falsche Antworten und erreicht gelegentlich falsche Systemzustände. Ein byzantinischer Fehler beschreibt im Allgemeinen ein schwierig zu erfassendes Fehlermodell.

In Mehrprozessor-Systemen bezeichnet der byzantinische Fehler eine Fehlerklasse. Falls eine Komponente an verschiedene Prozessoren unterschiedliche (protokollkonforme) Ergebnisse liefert, spricht man von einem byzantinischen Fehler. Bei der Planung wird davon ausgegangen, dass x Prozessoren bösartig arbeiten und das System maximal stören wollen.

2.3 Altcoins

Neben Bitcoin gibt es zahlreiche alternative Kryptowährungen, sogenannte "Altcoins" [3], die versuchen die Technologie hinter Bitcoin zu verbessern. Nachfolgend werden zwei reale Umsetzungen vorgestellt, die Usefulness umsetzen: Primecoin [King13] und Permacoin [MJSP+14]. Erstere Währung beschreibt eine Umsetzung von Usefulness, die sehr stark in die Richtung von [BRSV17] geht und statt Hash-Berechnungen eine Berechnung von Primzahlen vorschlägt. Permacoin setzt die Usefulness interessanter Weise auf einen ganz anderen Weg um. Statt einem Proof of Work wird ein Proof of Space vorgenommen, bei dem ein Teilnehmer des Blockchain-Netzwerks beweist eine gewisse Kapazität an Speicherressourcen zu besitzen.

2.3.1 Primecoin

Der größte Kritikpunkt der immer wieder im Zusammenhang mit Bitcoin und dessen Proof of Work Protokoll unter Experten auftaucht ist, dass die Berechnung zum Minen von Blöcken, also das Lösen einer Challenge, keinen nachhaltigen Wert besitzt [MJSP+14] [BRSV17] [**?**]. Wie schon bereits erwähnt hat das Proof of Work bei Pitcoin keinen anderen Sinn als zu

[3] "Altcoins" steht für engl.: alternative coins. `https://en.bitcoin.it/wiki/Altcoin`

beweisen, dass Energie aufgewendet wurde und man sich somit qualifiziert Blöcke zur Blockchain hinzuzufügen. Trotzdem ist das Proof of Work Protokoll ein wichtiger Kernmechanismus von Bitcoin, um das Double-Spending-Problem zu lösen und Sicherheit zu garantieren. Wie in den vorangegangen Abschnitten bereits erwähnt haben die Berechnungen von SHA256-Hashes keinen Nutzen außerhalb der Bitcoin-Welt.

Primecoin versucht diesen Trade-off auf effiziente Weise zu lösen. Es ist die erste Kryptowährung, die auf dem Proof of Work-Mechanismus basiert und Berechnungen durchführen lässt, die für die weitere Verwendung von Nutzen ist. Statt also einer SHA256-Hashes müssen Miner bei Primecoin lange Primzahlenketten finden. Dafür werden drei Arten von Primzahlenketten definiert, die sich für das Proof of Work Protokoll eignen: Cunningham Chain (CC) 1. Art, CC 2. Art und Bi-twin chain.

Cunningham Chain 1. Art. Bei der Cunninghamchain muss jede Primzahl der Kette um ein größer als das doppelte der vorhergehenden Primzahl sein:

Beispiel 2.1. *Cunningham-Kette der Länge 5: 1531, 3061, 6121, 12241, 24481*

Cunningham Chain 2. Art. Die Cunningham Chain 2. Art gibt vor, dass jede Primzahl um eins verringert als das doppelte des Vorgängers ist.

Beispiel 2.2. *Cunningham-Kette der Länge 5: 2, 5, 11, 23, 47*

Bi-Twin Chain. In der Bi-Twin chain werden Paare gehalten, deren Differenz 2 ergibt.

Beispiel 2.3. *Bi-Twin-Kette: 211049, 211051, 422099, 422101, 844199, 844201*

Nun könnte hier die Frage gestellt werden, inwiefern ein Nutzen in den Berechnungen von Primzahlen vorliegt oder ob es lediglich eine ebenso sinnlose Berechnung wie die Berechnung von SHA256 Hashes sei. Es gibt viele Anwendungsgebiete aus der Mathematik, die sich mit der Suche nach großen Ketten von Primzahlen beschäftigt. Unter anderem die Suche nach Mersenne Primzahlen [4] oder das Primzahlen Theorem[5]. Die folgende Liste der University of Tennesseee listet Gründe für die Suche nach Primzahlen auf: `http://primes.utm.edu/notes/faq/why.html`.

2.3.2 Permacoin

Einen anderen Ansatz einer nützlichen Alternative zu Bitcoin im Vergleich zu Primecoin liefert Microsoft in Zusammenarbeit mit der University of Maryland mit Permacoin[MJSP+14], bei dem die Blockchain als verteilter Speicher von Archivdaten dienen soll. Miner sollen nicht mehr nur reine Rechenleistung aufwenden, sondern vielmehr ungenutzten Speicherplatz zur Verfügung stellen, weshalb er vielmehr zu dem Konsensmechanismus Proof of Retrievabilty (POR) eingeordnet werden kann.

Der Grundgedanke, der hinter Permacoin steckt ist, dass statt Rechenleistung Speicherplatz für das Erzeugen neuer Block aufgewendet wird. Im POR Protokoll wird bewiesen, dass ein Knoten des Netzwerks Speicherressourcen für das Abspeichern von Dateien oder Dateifragmenten aufwendet[MJSP+14]. [MJSP+14] schlägt ein Verteiltes Daten-Archiv vor, das so vor Datenverlusten geschützt werden soll.

[4] `https://en.wikipedia.org/wiki/Mersenne_prime`
[5] `https://en.wikipedia.org/wiki/Prime_number_theorem`

3 Proofs of Useful Work

Ein Proof of Useful Work (uPoW) nach [BRSV17] muss ebenso wie ein PoW der Hardness-Anforderung gerecht werden. Damit ist gewährleistet, dass der Prover tatsächlich Abreit aufgewendet hat. Weiterhin muss uPoW aber auch einen Nutzen (engl. Usefulness) erzeugen, der schnell verifizierbar und einfach rekonstruierbar ist.

Die Ziele, die sich nach [BRSV17] für den Entwurf eines Proof of Useful Work Schemas ergeben ist die Betrachtung der folgenden beiden Eigenschaften:
Hardness. Challenges können so verteilt werden, dass eine Lösung für die Challenge Aufwand in Form von Arbeit oder Energie garantiert.
Usefulness. Aufgaben können als Challenge so delegiert werden, dass die Lösung der Worker schnell und effizient verifizierbar rekonstruiert werden kann.

TODO [NOCHMAL ÜBERARBEITEN. SIEHT AUS WIE IM PAPER]

Die größte Herausforderung ist, dass beide Eigenschaften immer für ein uPoW gelten und stets zusammen betrachtet werden müssen, da ohne Hardness einerseits kein Proof of Work möglich ist und andererseits ohne Usefulness kein die Berechnungen von Proof of Work keinen nutzen haben. Angenommen es werden nur triviale Challenges generiert, die ein Worker ohne intensive Rechenleistung löst, so gibt es keinen Nachweis der Arbeit und die Hardness-Bedingung wird gebrochen. Das wiederum ist allerdings die Grundvoraussetzung für Proof of Work. Wird dagegen Hardness gewährleistet, aber zufällige Challenges so geneiert, dass sie nicht mehr rekonstruiert werden können, geht der Nutzen verloren, ein bestimmtes Problem zu lösen, was somit gegen die Usefulness verstößt. Bei Bitcoin's Proof of Work werden Challenges mit dem Algorithmus `Gen()` zufällig generiert, sodass das Mapping von x auf $f(x)$ nicht möglich ist. In einem Proof of Useful Work Protokoll benötigen wir allerdings zusätzlich noch einen Algorithmus, der die Verbindung zwischen einer Lösung und einer Challenge wiederherstellt und ein $f(x)$ rekonstruiert. [BRSV17] definieren dafür einen Algorithmus `Recon(`c_x`, s)` zur Rekonstruktion (engl.: reconstruction) von $f(x)$. Der Algorithmus nimmt eine Challenge c_x entgegen und rekonstruiert basierend auf der Challenge $f(x)$. Insgesamt werden vier Algorithmen (drei aus dem Proof of Work Protokoll 2.2.2 und zusätzlich der `Recon`-Algorithmus) eingeführt:

- `Gen(x)` - randomisierter Algorithmus, der eine Challenge c_x in $mathcalO(n)$ erzeugt.
- `Solve(`c_x`)` - Algorithmus, der eine Lösung s zu einer Challenge c_x findet.
- `Verify(`c_x`, s)` - Algorithmus, der verifizieren kann, ob eine Lösung s zu einer Challenge c_x richtig ist.
- `Recon(`c_x`, s)` - Algorithmus, der eine Verbindung zwischen Lösung und Probleminstanz erzeugt.

Daraus ergibt sich die nachfolgende Definition 2 für ein Proof of Useful Work nach [BRSV17]:

Definition 2 (Proof of Useful Work)**.** *Ein Proof of Useful Work Mechanismus besteht aus den vier Algorithmen* `Gen`*,* `Solve`*,* `Verify` *und* `Recon`*, für die folgende Eigenschaften gelten sollen:*

- ***Efficiency****:* `Gen`*,* `Verify` *und* `Recon` *sollen eine effiziente Laufzeit besitzen*
- ***Completeness****: Pr[Verify(c,s) = accept] = 1*
- ***Soundness****: Pr[Verify(c,s) = accept] = 1*

- ***Hardness:***
- ***Usefulness:***

Ohne Hardness und Usefulness beschreibt die Definition 2 ein Proof-System wie es auch [Will16] beschreibt. Für die gewährleistung eines Proof of Useful Work Frameworks müssen Hardness und Usefulness immer enthalten sein. Ein einfaches Proof of Work würde ohne dem `Recon`-Algorithmus und ohne der Bedingung von Soundness auskommen. Bei einem PoW sind wir nicht daran interessiert, ob eine Lösung korrekt ist, sondern nur ob das FInden der Lösung Arbeit gekostet hat, das wiederum durch Hardness gegeben ist. Umso wichtiger wird diese Bedingung für einen uPoW-Schema, bei dem es uns daran gelegen ist aus einer Berechnung einen Mehrwert zu ziehen und nicht Fake-Berechnungen zuzulassen, die wir gar nicht erhalten wollten. Ball et al. beweisen Proofs of Useful Work für das k-Orthogonale Vektoren Problem, 3SUM und APSP. Darüberhinaus lassen sich weitere NP-schwere Probleme auf diese Probleme reduzieren. Besonders Eigenschaften aus der Graphentheorie passen auf das kOV-Problem

3.1 Herausforderungen

Die Herausforderungen, die sich also für ein Proof of Useful Work ergben sind zum Einen Challenges zu finden, die die Hash-Berechnungen von Bitcoin ersetzen können und zum anderen müssen Probleminstanzen die fünf Eigenschaften aus Definition 2 erfüllen. Die Größte Herausforderung eines uPoW-Entwurfs ist stets die zwei Bedingungen *Hardness* und *Usefulness* zusammen zu betrachten. Wichtig für Probleminstnanzen ist, dass gefundene Lösungen effizient verifizierbar sind. Da Wahl willkürlicher Probleminstanzen kann dazu führen, dass eine Hardness nicht gewährleistet werden kann. Dies wäre im einfachsten Fall beispielsweise willkürlich sich eine Primzahl ausgeben zu lassen ohne eine Invariante zu definieren. So könnte die Antwort eines Provers die Zahl “2” lauten. Es ist offensichtlich, dass hier keine Arbeit investiert wurde. Gibt es Probleminstanzen, die zwar die Hardness-Bedingung erfüllen allerdings zufällig ausgewählt werden, sodass sie nicht reproduzierbar sind, führt das zum bruch der Usefulness. Darüber hinaus muss im Sinne der Usefulness ebenfalls gelten, dass falsche Berechnungen erkannt werden können. Weiterhin muss betrachtet werden ob ein entsprechendes Framework in die Blockchain passt. So müsste im Fall von Bitcoin ein Upgrade über einen Hardfork, Softfork,.... geschehen.

3.2 k-Orthogonal-Vectors-Problem

Wir greifen den Gedanken von nützlichen Berechnungen auf und wollen die nutzlose SHA256-Hash-Berechnung ersetzen gegen etwas, das weiter verwendet werden kann. Wie bereits im Abschnitt zum Proof of Work Protokoll erwähnt gibt es gewisse Probleme in der Informatik, die NP-schwer sind, das heißt Probleme, die entweder gar nicht lösbar sind oder dass es sich um eines der Schwierigsten Probleme der Klasse NP handelt. Die Komplexitätsklassen vieler Probleme dieser Seminararbeit behandelt eben solche Probleme.

Definition 3. *Ein Entscheidungsproblem E nennt sich NP-schwer wenn es für jedes Problem P aus NP eine polynomielle Reduktion von P nach E existiert.*

Ein beliebtes NP-schweres Problem aus der theoretischen Informatik ist das k-Orthogonal-Vecotrs-Problem. Besonders Graphenprobleme können sehr gut auf das OV-Problem reduziert werden.

Definition 4 (k-Orthogonale Vektoren)**.** *Gegeben sind k Mengen* $(U_1, ..., U_k)$ *für ein festes* $k \geq 2$ *mit* $k \in \mathbb{N}$*, von n Vektoren der Dimension d* $\{0,1\}^{d(n)}$*. Dann heißt eine solche Menge von Vektoren k-orthogonal, wenn* $u^s \in U_s$ *für* $s \in [k]$*, sodass sich*

$$\sum_{l \in [d(n)]} u_l^1, ..., u_l^k = 0$$

ergibt.

Da das k-OV-Problem allerdings worst-case hart ist und es keine Aussagen zum Average-Case gibt, reduziert [BRSV17] das Problem auf ein verwandtes Problem, das sowohl im Worst-Case als auch im Average-Case hart ist. Dafür definiert [BRSV17] ein Polynom $gOV^k_{n,d,p}$ vom Grad kd: $\mathbb{F}_p^k nd \longrightarrow \mathbb{F}_p$, wobei p jede Primzahl sein kann. Alle weiteren Parameter sind zum k-OV-Problem äquivalent. Für eine Berechnung von gOV gilt dann:

$$gOV^k_{n,d,p}(U_1, ..., U_k) = \sum_{i_1,...,i_k \in [n]} \prod_{l \in [d]} (1 - u^1_{i_1 l} \cdots u^k_{i_k l})$$

[BRSV17] zeigt weiterhin, dass das entwickelte Proof of Useful Work für die Probleme 3SUM und ALl-Pairs Shortest Paths eingesetzt werden kann. Hierbei gilt allerdings auch wieder, dass jedes Problem, das auf das k-OV, 3SUM oder APSP Problem reduziertbar ist, eignet sich ebenso für das uPoW-Schema. [BRSV17] weist besonders auf alle Probleme aus der Graphentheorie hin, die auf das k-OV-Problem reduzierbar sind, weil diese sich sehr gut eignen würden. Daraus folgt, dass eine sehr große Menge an praktischen Problemen an das Netzwerk delegiert werden kann und fortan Berechnungen mit einem Nutzen durchgeführt werden können.

3.2.1 3SUM

Beschreibung. Gegeben eine Menge S von n ganzen Zahlen. Gibt es drei Zahlen $a, b, c \in S$ deren Summe $a+b+c = 0$ ergibt? Das 3SUM problem kann in $\mathcal{O}(n^2)$ gelöst werden. Als untere Schranke ist keine bessere Laufzeit als $\Omega(n \log n)$ bekannt. Frage: Gibt es einen Algorithmus, der das 3SUM Problem in $\mathcal{O}(n^{2-\varepsilon})$ für ein $\varepsilon > 0$ löst? Eine weitere Idee ist Probleme zu finden, die sich auf dieses Problem reduzieren lassen und eventuell häufig in der Praxis auftauchen. Ein Problem heißt genau dann 3SUM-schwer, das durch eine Lösung in $\mathcal{O}(n^2)$ eine Laufzeit in $\mathcal{O}(n^2)$ für das 3SUM Problem impliziert. [**?**] Haben gezeigt, dass es eine große Klasse an 3SUM-Problemen gitb. Darunter sind nachfolgende einige Aufgelistet:

- Given a set of lines in the plane, are there three that meet in a point?
- Given a set of non-intersecting axis-parallel line segments, is there a line that separates them into two non-empty subsets?
- Given a set of infinite strips in the plane, do they fully cover a given rectangle?
- Given a set of triangles in the plane, compute their measure.
- Given a set of triangles in the plane, does their union have a hole?
- A number of visibility and motion planning problems, e.g.,
- Given a set of horizontal triangles in space, can a particular triangle be seen from a particular point?

- Given a set of non-intersecting axis-parallel line segment obstacles in the plane, can a given rod be moved by translations and rotations between a start and finish positions without colliding with the obstacles?

TODOThey proved that a large class of problems in computational geometry are 3SUM-hard, including the following ones. (The authors acknowledge that many of these problems are contributed by other researchers.)

They proved that a large class of problems in computational geometry are 3SUM-hard, including the following ones. (The authors acknowledge that many of these problems are contributed by other researchers.)

A problem is called 3SUM-hard if solving it in subquadratic time implies a subquadratic-time algorithm for 3SUM. Ein Problem P heißt 3SUM-hard genau dann, wenn gilt 3SUM f (n) P mit f(n) wächst langsamer als n 2 .

Sei P 3SUM-hard: Algorithmen für P haben immer eine Laufzeit größer gleich O(n 2). Findet man einen mit einer kleineren Laufzeit, folgt daraus die Existenz eines Äquivalenten für 3SUM

Die untere Schranke (n log n) von 3SUM überträgt sich auf P.

Da 3SUM für sich alleine in der Praxis keine große Bedeutung hatte war ihre Idee es Probleme zu finden, die mindestens so schwer wie 3SUM sind. A problem is called 3SUM-hard if solving it in subquadratic time implies a subquadratic-time algorithm for 3SUM. The concept of 3SUM-hardness was introduced by Gajentaan Overmars (1995).

By now there are a multitude of other problems that fall into this category. An example is the decision version of X + Y sorting: given sets of numbers X and Y of n elements each, are there n distinct x + y for x X, y Y?[9] - BESCHREIBUNG - NUTZEN - PRO CONTRA - EVALUIERUNG

3.3 Mögliche Kandidaten

Wie im vorhergehenden Abschnitt beschrieben kann jedes Problem, das auf das k-OV-Problem reduziert werden kann, an Prover deligiert werden, die einen Nachweis über Arbeit liefern müssen. Weiterhin zeigt [BRSV17], dass diese Annahme ebenso für 3SUM und APSP gilt. Im folgenden werden Kandidaten vorgestellt, die für einene Einsatz im Proof of Useful Work Framework nach [BRSV17] betrachtet werden. Unter den Kandidaten befinden sich auch Projekte, die auf verteilten Rechnernetzen durchgeführt werden. lche die von nutzvollen Berechnungen weggehen und vorschlagen, dass beispielsweise mit der Abwärme von Servern, Wohnhäuser beheitzt werden sollten, um so einen Nutzen für die Gesellschaft zu ermöglichen. Oder dass von Kraftwerken die Abfallspannung zum Minen verwendet werden solle, da diese sowieso zur Verfügung steht und keine Verwendung sonst hat. In dieser Ausarbeitung liegt der Schwerpunkt allerdings auf die Berechnung von theoretischen sowie praktischen Problemen, die bestimmte Aufgaben mithilfe von Algorithmen lösen sollen. Neben dem bereits vorgestellten OV-Problem sieht [BRSV17] als Möglichkeit 3SUM 3.2.1, APSP und generelle alle Graphenprobleme, die auf das Orthogonale Vektor Problem reduziert werden können.

Mögliche Kandidaten sind:

- OV
- 3SUM
- APSP [Will15]

- Proteinfolding
- Aliensuche
- Birthdayproblem

Es liegt in der Natur der Sache die Hashberechnung von Bitcoin durch eine nützliche Berechnung zu ersetzen. Dadurch werden immer wieder verschiedene Vorschläge getätigt, die Berechnungen sollten große Projekte große Projekte aus verschiedenen Bereichen der Astrophysik oder Biologie wie SETI@Home resp. Folding@Home, um die Menschheit bei der Suche nach außerirdischer Intelligenz voranzutreiben oder DNA-bedingte Krankheiten zu heilen. Diese Lösungen sind sehr gute Vorschläge jedoch sind sie nicht so einfach integrierbar wie sich viele vorstellen. Der wesentliche Unterschied zwischen Bitcoins SHA256-Hashberechnungen und den Distributed Computing Projects wie SETI@home und Folding@home ist, dass letztere nicht effizient verifizierbar sind. Eine Einwegfunktion dagegen kann man schnell schnell und effizient verifizieren??? Momentan handelt es sich bei den meisten Projekten der Distributed Computing Projects um freiwillige Projekte. Das heißt hinter jedem Projekt und jeder Berechnung stehen Teilnehmer, die darauf bedacht sind das Ziel für das jeweilige Projekt mit richtigen Werten zu berechnen. Werden diese Projekte alelrdings zum Bitcoin Mining verwendet geht das Ziel des Projektes verloren, da ein Teilnehmer im Bitcoin-Netzwerk eventuell kein Bedürfnis in der Probleminstanz für sich sieht und einfach Fake-Daten berechnet, um seinen Gewinn zu maximieren. Eine Unterscheidung zwischen gefakten und echten Werten ist damit nicht möglich. Das wiederum hätte keinen Mehrwert für das jeweilige Projekt, sodass hier ein Bruch der Usefulness von uPoW stattfindet.

Die EHER VON PROVER UND VERIFIER SPRECHEN!!!! CHALLENGER MANCHMAL AUCH!!!!!!!!!!!!!!!!!!!!!!

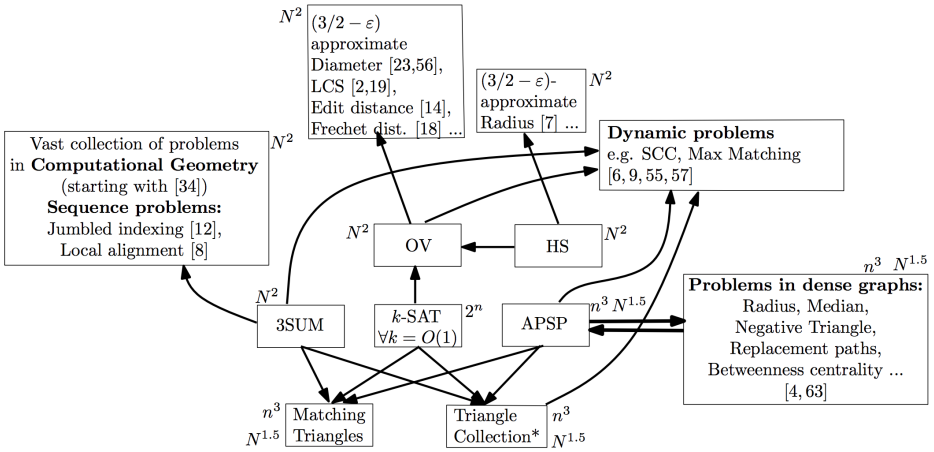

Abbildung 4: Dies ist die Bildunterschrift

3.4 Blockchain-Schema

Die Kernstruktur Bictoin's Blockchain übernehmen wir. DIe Blockchain ist weiterhin eine Chronik von Transaktionsblöcken, die miteinander verkettet sind. Die Verkettung ergibt sich wieterhin aus dem Link zum Hash des vorherigen Transaktionsblocks. Berechnungen, die sowieso auf riesigen SUpercomputern stattfinden würden, könnten auf REchner eines verteilten Netzwerks ausgelagert werden. Somit würde man die Kosten für Stromverbrauch reduzieren auf der Welt. Das Problem ist wenn irgendwann nicht genügend Challenges vorhanden sind, müssen Primzahlen beispielsweise weiter berechnet werden.

In fig1 wird ein öffentliches Board dargestellt, das für Delegierende dazu dient ihre Probleme zur Berechnung zur Verfügung zu stellen. Sobald ein Miner ein PoW durchführen muss, nimmt dieser sich ein Problem vom Problem Board und löst dieses, um den nächsten Block an die Blockchain anzuhängen. Dabei kann jede Art von Priority Scheduling zum Wählen der nächsten Challenge verwendet werden.

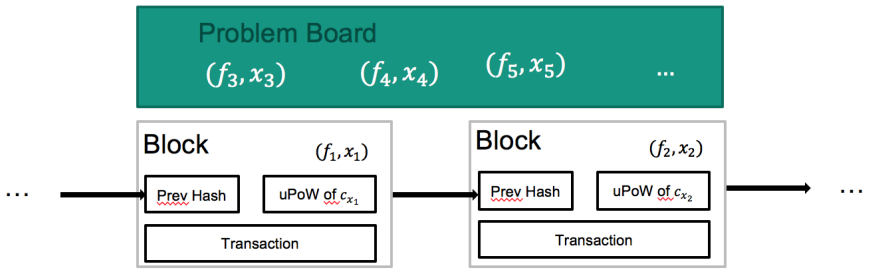

Abbildung 5: Dies ist die Bildunterschrift

Der Worker ist jetzt im Besitz der Probleminstanz x. Nun kann er mit HIlfe eines Zufallsorakel H die CHallenge c_x generieren ohne der Substitution von $r = H(currentBlock)$

$$c_x = x + rt | t \in [D + 1]$$

Für ein echtes Zufallsorakel H, r wird zufällig sein und das wird somit ein Standardchallenge für uPoW werden.

Somit ist das vorgestellte Blockchain Schema und das PoW sensitive to changes in the Block that the proof is made for.

4 Diskussion

Das vorgestellte Framework eines Proof of Useful Work nach [BRSV17] Wer darf Probleme delegieren? Ist Fairness bei der Vergabe von Challenges garantiert? Wie sieht ein Update von Bitcoin aus durch die Integration neuer Mechanismen? Was ist wenn es keine Probleme auf dem Public Board gibt?

5 Zusammenfassung und Ausblick

Beschrieben wurde ein Allgemeiner Ansatz/Framework zu einem Proof of Useful Work basierend auf der Arbeit von [BRSV17]. Momentan existiert noch keine Praktische Umsetzung dieses Frameworks, sodass der praktische EInsatz noch geprüft werden muss. Besonders für die Forschung aus der Theoretischen Informatik ist dieser Ansatz sehr interessant, da viele NP-harte Probleme anstelle von Supercomputern sinnvoll über ein Public Board der Blockchain an Worker delegiert werden können. Mögliche Kandidaten für das Schema sind alle Probleme, die auf eines der Probleme 3SUM, All-Pairs Shortes Paths oder Orthogonal Vectors Problem reduzierbar sind. Dabei ist darauf zu achten, dass ein Problem die fünf Eigenschaften *Hardness*, *Usefulness*, *Efficiency*, *Completeness* und *Soundness*. Würde das theoretische Konzept in der Praxis so umgesetzt werden könnte es auf jeden Fall zu einer Senkung des globalen Stromverbrauchs führen und ein Proof of Work ablösen, das Berechnungen durchführt, die keinen weiteren Nutzen als der Beweis der geleistet Arbeit hat.

Es sind allerdings noch viele Fragen offen, die es in der weiteren recherche gilt zu überprüfen. So ist nicht eindeutig beschrieben was passieren soll, wenn auf dem Public Board nicht genügend Probleme liegen, die Worker bearbeiten könnten. Würde das System stagnieren oder können Primzahlenberechnungen abhilfe verschaffen? Auch wer delegieren darf ist nicht geklärt.

Literatur

[Ande13] Nate Anderson. *Mining bitcoins takes power, but is it an "environmental disaster?"*. http://tinyurl.com/cdh95at. 2013.

[BRSV17] Michael Ball, Alon Rosen, Manuel Sabin und Prashant Nalini Vasudevan. *Proofs of Useful Work*. Columbia University. 2017.

[DwNa92] Cynthia Dwork und Moni Naor. *Pricing via Processing or Combatting Junk Mail*. 1992.

[King13] Sunny King. *Primecoin: Cryptocurrency with prime number proof-of-work*. 2013.

[MJSP+14] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno und Jonathan Katz. *Permacoin: Repurposing bitcoin work for data preservation.* 2014.

[Naka08] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

[Shan17] Daniel Shane. *Bitcoin boom may be a disaster for the environment*. CNN-Tech. 2017.

[Will15] Virginia V. Williams. *Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis*. In Proc. International Symposium on Parameterized and Exact Computation. 2015.