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

Jean-Marc Hendrikse

Abstract

Diese Dokumentation enthält eine sortierte Liste der wichtigsten LaTeX–Befehle. Die einzelnen Listeneinträge sind untereinander durch viele Querverweise verkettet, die ein Auffinden inhaltlich zusammengehöriger Informationen erheblich erleichtern.

1 Einführung

Bitcoin und Blockchain sind mittlerweile in aller Munde und kaum jemand ist nicht daran interessiert, vorangetrieben durch die positive Entwicklung der Kryptowährung in der Vergangenheit, an dem großen Kuchen teilzunehmen. Es ist für jeden möglich mit entsprechender Hardware sogar selbst Geld zu schürfen. Das auf der Blockchain basierende elektronische Geldsystem Bitcoin 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 aufgewendet wurde[LITERATUR BELEG], was als Proof of Work bezeichnet wird. Der Proof of Work Mechanismus ist das Herzstück von Bitcoin, das der Währung erst Wert und Sicherheit vor Angriffen wie Double-Spending verleiht. 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, sondern ein Abfallprodukt dafür ist, dass gezeigt wird, dass lediglich Rechenleistung und somit Arbeit aufgewendet wurde. Diese Problem von Bitcoin wird oft in verschiedenen Literaturen zitiert. Einige sprechen an dieser Stelle sogar von einem "Umweltdisaster" [XYZ]. Doch wieso man nicht einfach auf dieses Protkoll verzichten möchte liegt klar auf der Hand: es ist für seine Zwecke sehr erfolgreich und robust. Ein natürlicher Ansatz, der im Zuge dieser Ausarbeitung beschrieben und diskutiert wird, ist die nutzlose Berechnung durch Berechnungen zu ersetzen, die einen Mehrwert für die Gesellschaft liefert. Diese Variante des Proof of Work wird als Proof of Useful Work 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.

In Abschnitt 2 dieser Ausarbeitung werden die Grundlagen von Blockchain und Bitcoin vermittelt, indem die grundlegende Technologie und Konzepte beschreiben werden, sowie erste Umsetzungen von Proof of Useful Work Protokollen durch Primecoin [] und Permacoin []. Am Ende des Grundlagen Kapitels erläutere ich wichtige Begriffe aus der theoretischen Informatik, die für das Verständnis des Proof of Useful Work Schemas essentiell sind. Im zweiten beschreibe ich das Proofs of Ufeful Work Framework nach [BRSV17] anhand dessen sich viele weitere Probleme aus der theoretischen Informatik reduzieren lassen. Weiterhin werden Kandidaten

für ein Proof of Useful Work Schemas recherchiert und auf Grundlage der Definition und des Beweises von [BRSV17] evaluiert. Am Ende dieses Kapitels soll ein Blockchain-Schema mit dem Proof of Useful Work Protokoll das alte Schema von Bitcoin ersetzen können.

Zum Abschluss geben wir eine Zusammenfassung der Thematik und einen Ausblick mit offenen Fragen.

2 Grundlagen

In diesem Abschnitt möchte ich auf die Grundlagen der Blockchain und auf das daraufbasierende elektronische Geldsystem Bitcoin erläutern, die für das Verständnis der vorangehenden Kapitel notwendig sind.

2.1 Definition von Nutzen

Unterscheidung in gesellschaftlichen und privaten Nutzen. Unter Nutzen (Usefulness) verstehe ich im folgenden den Benefit, den man aus dem Gebrauch einer Sache ziehen kann[https://de.wiktionary.org/wi
Dadruch, dass es sich bei der Blockchain um ein verteiltest System in einer Community handelt, betrachte ich in meiner Arbeit die Sichtweise des gesellschaftlichen Nutzens. 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

Grob gesprochen handelt es sich bei der Blockchain um ein dezentrales öffentliches Hauptbuch (engl. public ledger). Um bislang Transaktionen durchführen zu können, vertrauten „vertrauenswürdigen“ Instanzen, beispielsweise Banken, denen wir unser Geld oder Daten zur Verwaltung übergaben. Diese dritten Instanzen besitzen einen permanenten Zugriff auf private Besitztümer und können jeder Zeit auf diese zugreifen. Mithilfe der Blockchain Technologie wird eine neue Möglichkeit offenbart, vom zentralen Gedanken der Systeme auf ein dezentrales System umzusteigen. Im wesentlichen 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 dabei keine Rolle. Es kann sich 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. Der Entwurf der Blockchain sieht wie folgt aus: Jede Transaktion wird 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, also dass diese Transaktion tatsächlich stattgefunden hat. Nur und nur dann kann dieser Block aufgenommen werden. Dadurch dass jeder Teilnehmer eine Kopie der aktuellen Blockchain-Historie enthält, kann er diese mit der neuen abgleichen und verifizieren. 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.

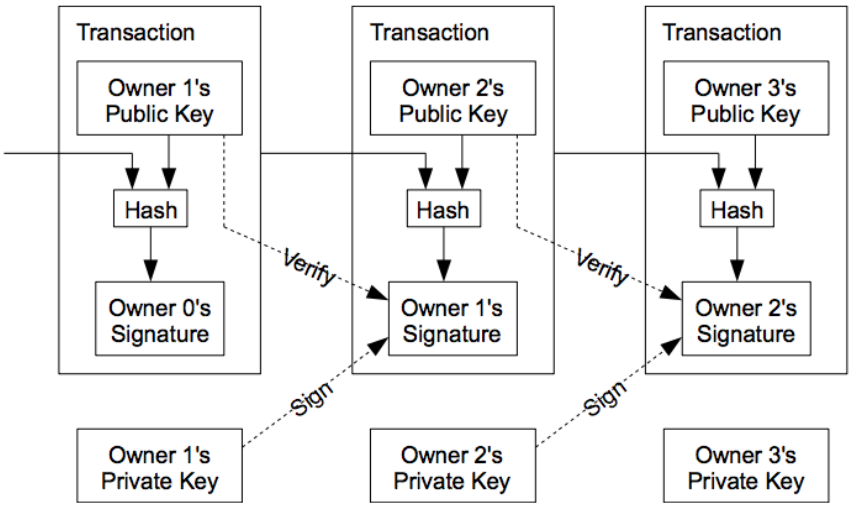

Abbildung 1: Dies ist die Bildunterschrift

2.2.1 Konsensmechanismus

In zentralisierten Organisationen werden Entscheidungen meistens getroffen, indem eine 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. Ein Konsensmechanismus bezeichnet einen Algorithmus, 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 erschwert ein Konsensmechanismus die Manipulation von Daten erheblich. Der Hauptgedanke der dahinter steckt ist, dass durch einen dezentralen Konsens-Mechanismus ein zentrale Kontrollinstanz zur Integritätsbestätigung obsolet wird.

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 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 ei-

ne kleine Transaktionsgebühr. [https://www.coindesk.com/short-guide-blockchain-consensus-protocols/]

Doch Proof of Work ist keine Neuerfindung von Bitcoin sondern wurde auch bereits für den E-Mail-Versand als Mechanismus gegen Spaming eingesetzt. Um eine E-Mail zu versenden, musste der Sender vereinfacht gesagt zusätzliche Berechnungen durchführen. Da die Maschinen in ihrer Rechenkapazität beschränkt waren, ist es notwendig gewesen, hohe Investitionen 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. Denn der Empfänger einer E-Mail konnte auf einfache Art prüfen, ob die Berechnungen korrekt durchgeführt wurden, während die Berechnungen für den Sender mit einem gewissen Aufwand verbunden waren.

Definition 1. *Ball et al. definieren drei wesentliche Algorithmen, die POW ausmachen:*

1. `Gen`(1^n)*, randomisierter Algorithmus, der eine challenge c erzeugt.*
2. `Solve(c)`*, Algorithmus, das die Challenge c entgegen nimmt und eine Lösunge s dafür findet.*
3. `Verify(c,s)`*, verifiziert die Lösung s für die challenge c.*

Dabei ist die Bedinung, dass `Gen` und `Verify` eine schnelle lineare Laufzeit als obere Schranke besitzen und für `Solve` soll eine Hardness1FUSSNOTE ICH BELASSE HIER DIE ENGLISCHE NOTATION (Schwere), also das schwere Finden einer Lösung, gelten. Dabei wird eine Zeit $t(n)$ vorgegeben, die im Schnitt die Arbeitszeit eines Algorithmus vorgibt. Somit soll `Solve` eine Lösung berechnen, die von `Verify` akzeptiert wird, andererseits soll `Verify` mit hoher Wahrscheinlichkeit alles ablehnen, das die Zeit von $t(n)^1 - \epsilon$ für jedes $\epsilon > 0$ unterschritten hat. Diese Bedingungen erschweren allerdings die suche nach nützlichen Problemen. IM Fall von Bitcoin beispielsweise wird soll als Challenge c ein SHA256-Hash berechnet werden, der eine gewisse Anzahl an führenden Nullen enthält.

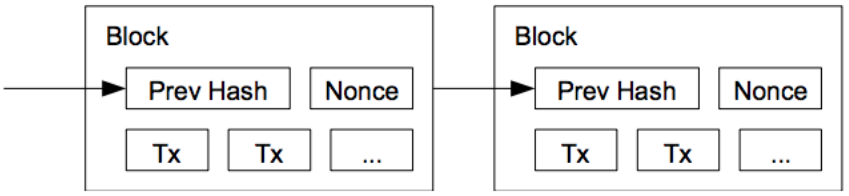

Abbildung 2: Dies ist die Bildunterschrift

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

Die Difficulty (genauer behandelt im Thema “Mining”) ergibt sich 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

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

Unter dem Begriff Altcoins versteht man Kryptowährungen, die Alternativen zu Bitcoin darstellen. Nachfolgend werden zwei Währungen erläutert, die in Richtung Proof of Useful Work gehen.

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 [BELEGE]. Wie schon bereits erwähnt hat das Minen keinen anderen Sinn als zu beweisen, dass Energieaufgewendet wurde und man sich so dafür qualifiziert Blöcke zur Blockchain hinzuzufügen. Trotzdem ist

das Proof of Work Protokoll das Herzstück von Bitcoins Sicherheit. Denn würde es Proof of Work nicht geben, so könnte ein kompromitierter Angreifer sich als Millionen von Bitcoin Knoten zur selben Zeit ausgeben und somit ernsthaft Bitcoins Transaktionsblöcke angreifen oder austauschen. Allerdings werden lediglich SHA256 berechnet, wodurch eine große Menge an Energie verschwendet wird. Primecoin ist die erste Kryptowährung, die auf dem Proof of Work Ansatz basiert und eine effiziente Lösungen für Aufgaben generiert, die weiter verwendet werden können und somit einen Nutzen liefern [Primecoin paper]. Statt also einer nutzlosen Berechnung eines SHA256 Hashes1 müssen Miner beim Proof of Work Protokoll von Primecoin lange Primzahlenketten finden. Es gibt drei Arten von Primzahlenketten, 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 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-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-Kette: 211049, 211051, 422099, 422101, 844199, 844201

Nun könnte man sich hier die Frage stellen wo der Nutzen in Berechnungen von Primzahlen liegt oder ob es lediglich eine ebenso sinnlose Berechnung wie die Berechnung von SHA256 Hashes ist. Doch 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 oder das Primzahlen Theorem2. Nachfolgend seine Liste der University of Tennesseee erwähnt, die Gründe für die Suche nach Primzahlen liefert: http://primes.utm.edu/notes/faq/why.html

1 Dabei wird mit nutzlos im Sinne für Weiterverwendung gemeint 2 Fußnote

2.3.2 Permacoin

Einen ganz anderen Ansatz zur nützlichen Alternative zu Bitcoin im Vergleich zu Primecoin liefert Microsoft in Zusammenarbeit mit der University of Maryland mit Permacoin, 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 dem Protokoll Proof of Retrievabilty ähnelt.

Der Grundgedanke, der hinter Permacoin steckt ist, dass statt Rechenleistung Speicherplatz für zum Minen von neuen Block aufgewendet wird. Im POR Protokoll wird bewiesen, dass ein Knoten des Netzwerks Speicherressourcen für das Abspeichern von Dateien oder Dateifragmenten aufwendet. Die Autoren von [Permacoin] schlagen ein Verteiltes Daten-Archiv vor, das so vor Datenverlusten geschützt werden soll.

2.4 NP-schwere Probleme

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

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.

Beide Eigenschaften müssen immer für ein uPoW gelten und stets zusammen betrachtet werden, da ohne Hardness einerseits kein Proof of Work möglich ist und andererseits ohne Usefulness kein Proof of **Useful** Work erreicht werden kann. Anegnommen 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 resultieren daraus vier Algorithmen (drei aus dem Proof of Work Protokoll 2.2.2 und zusätzlich der `Recon`-Algorithmus). :

- `Gen(x)` - randomisierter Algorithmus, der eine Challenge c_x in O(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 eine eine Definition 3 für uPoW nach [BRSV17]:

Definition 3 (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 (dt.: Effizienz):* ***Gen(x)*** *läuft für*

- *Completeness (Vollständigkeit): Pr[Verify(c,s) = accept] = 1*
- *Soundness (Korrektheit): Pr[Verify(c,s) = accept] = 1*
- *Hardness:*
- *Usefulness:*

Ohne Hardness und Usefulness beschreibt die Definition 3 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 useful Proof of Work ergben sind zum Einen Challenges zu finden, die die Hash-Berechnungen von Bitcoin ersetzen können und zum anderen müssen die Challenges Bedingungen erfüllen.

3.2 OV-Problem

Ein beliebtes NP-schweres Problem aus der theoretischen Informatik ist das k-Orthogonale Vekotren 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 $k \geq 2$ mit $k \in$, 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 $gOV^k_{n,d,p} : F_p^k nd \longrightarrow F_p$, wobei p jede Primzahl sein kann. Alle weiteren Parameter sind zum k-OV-Problem äquivalent. Für eine Berechnung von gOV gilt dann:

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

3.3 Mögliche Kandidaten

Für mögliche Kandidaten gibt es sehr viele interessante Ideen. Darunter sind auch solche 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. Neben dem bereits vorgestellten OV-Problem sieht [BRSV17] als Möglichkeit 3SUM 3.3.1 und APSP und generelle alle Graphenprobleme, die auf das Orthogonale Vektor Problem reduziert werden können.

Mögliche Kandidaten sind:

- OV
- 3SUM
- APSP
- Proteinfolding
- Aliensuche

3.3.1 3SUM

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 $\mathcal{O}(n^{2-\epsilon})$ für ein $\epsilon > 0 lst? Ein Problem Pheit 3SUM - hard genau dann, wenn gilt 3SUM f(n) P mit f(n) wchstlangsamer als n2.$

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). 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.)

- 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?

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]

3.3.2 All Pairs Shortest Path (APSP)

In Floyds Version findet er die kürzesten Pfade zwischen allen Paaren von Knoten eines Graphen und berechnet deren Länge (APSP, all-pairs shortest path)

3.3.3 Folding@Home

Aus dem Bereich der Biologie und Medizin

3.3.4 SETI@Home

Hinter der Abkürzung SETI steckt "**S**earch for **E**xtra-**T**errestrial **I**ntelligence at home" und befasst sich mit der Suche nach außerirdischer Intelligenz im Bereich der Astronomie. Es handelt sich um ein Volenteer-Computing-Projekt

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.

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.

[0]https://en.wikipedia.org/wiki/SETI@home

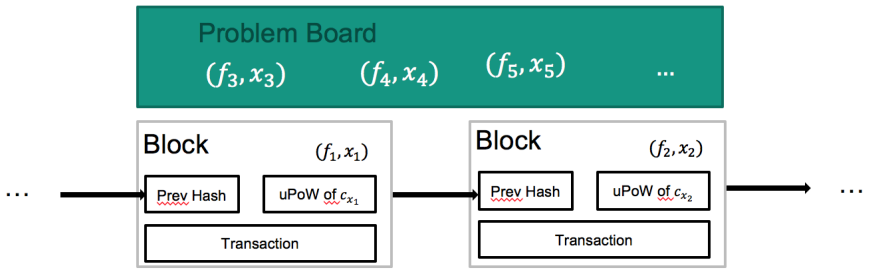

Abbildung 3: Dies ist die Bildunterschrift

3.5 Evaluierung der Kandidaten

4 Diskussion

5 Zusammenfassung und Ausblick

Literatur

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