Proof of Useful Work: Ein Alternatives Schema zu Proof of Work mit dem Schwerpunkt auf Nutzen

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 [Naka08]. Bei Proof of Work werden intensive Berechnungen einzig und allein für den Nachweis durchgeführt, dass Arbeit bzw. Rechenleistung aufgewendet wurde. Der Stromverbrauch hat somit keinen weiteren Nutzen für die Gesellschaft und wird deswegen unter anderem auch als "Proof of Waste"[1] bezeichnet. Um der Verschwendung von Energieressourcen entgegenzuwirken gibt es Ansätze, sogenannte Proofs of Useful Work, die im Proof of Work Protokoll nutzlose Berechnung durch nützliche ersetzen, um so der Gesellschaft einen Mehrwert zu bieten. In dieser Ausarbeitung soll ein solches Proof of Useful Work beschrieben werden, das auf der Basis NP-harter Probleme beruht und eine Alternative zu Bitcoin's Proof of Work bieten soll.

1 Einführung

Bitcoin [Naka08] gilt als die erste erfolgreiche digitale Geldwährung. Seinen großen Erfolg verdankt die Währung der zugrundeliegenden Technologie, der Blockchain. Durch die wachsende Aufmerksamkeit, die vor allem durch positive Meldungen vieler Medien vorangetrieben wird [2], wächst die Teilnehmerzahl der Bitcoin-Blockchain immer mehr an.
Anders als es bisher jedoch bei zentralen Institutionen in der heutigen Gesellschaft der Regelfall ist, liegen die Coins von Bitcoin nirgendwo zentral in physischer Form auf einer Bank, sondern werden von Computersystemen erzeugt [Shan17]. Im Bitcoin-Jargon spricht man auch von "Mining". Die Teilnehmer der Blockchain werden als "Miner" bezeichnet. 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 tatsächlich 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. Damit die Berechnung einer Challenge akzeptiert werden kann, muss allerdings sehr viel Rechenleistung investiert werden. In Zahlen ausgedrückt sind es von 2,2 bis 9,7 Tonnen CO2 pro Bitcoin [Mokh17]. Im Verglichen mit der Zahl von 1,48 Tonnen CO2 pro Passagier auf einem Flug von San Francisco nach Bonn ist das extrem viel [Mokh17]. Je größer die Kette der Blockchain wird, desto mehr nimmt die Schwierigkeit der Challenges zu. Das Problem, das hinter dieser Berechnung steckt, ist, dass die Lösung einer Challenge danach nicht weiterverwendet werden kann, da es keinen Nutzen liefert und somit lediglich als Abfallprodukt betrachtet werden kann. Das Problem wird oft in Literaturen zitiert [Shan17].

[1] `https://hackernoon.com/proof-of-work-or-proof-of-waste-9c1710b7f025`
[2] `https://t3n.de/news/bitcoin-erstmals-20000-dollar-887689/`, 17.12.2017

[Ande13] und [Shan17] bezeichnen den Mechanismus hinter Bitcoin sogar als "Desaster für die Umwelt".
Die naive Frage, die sich einem sofort stellt, ist, wieso Proof of Work nicht einfach ersetzt werden kann. Tatsächlich gibt es bereits Ansätze, die diesen Gedanken verfolgen wie Proof of Stake [KiNa12] oder Proof of Retrievability [ShWa08]. Proof of Work hat trotz des hohen Stromverbrauchs allerdings seine Vorteile. Zum einen ist es für seine Zwecke im Hinblick auf Sicherheit sehr erfolgreich und zum anderen sehr robust. Ein zweiter natürlicher Ansatz wird im Zuge dieser Ausarbeitung beschrieben und diskutiert: Das 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 Proofs of Useful Work vor, das für Probleme geeignet ist, die auf NP-schwere Probleme wie Orthogonal Vectors Problem, 3SUM oder All-Pairs Shortest Path reduzierbar sind [Will15].
Zunächst werden hier die grundlegenden Begriffe definiert und erklärt, die für das Verständnis der folgenden Kapitel notwendig sind. Abschnitt 2 dieser Ausarbeitung führt deswegen zunächst in die Grundlagen von Blockchain und Bitcoin ein. Es wird dabei vor allem Wert auf die grundlegende Technologie der Blockchain anhand von Bitcoin gelegt. Zudem werden erste Umsetzungen von "useful" Proof of Work Protokollen durch Primecoin [King13] und Permacoin [MJSP+14] vorgestellt.
In Abschnitt 3 wird das Framework zu Proofs of Useful Work nach [BRSV17] beschrieben. Anhand dessen lassen sich sehr viele Probleme aus der theoretischen Informatik, die beispielsweise auf das Orthognonal Vectors Problem reduzierbar 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.
Zuletzt wird eine Zusammenfassung der Arbeit gegeben, wobei auf offene Fragen hingewiesen wird, die im Laufe der Arbeit entstanden sind, sowie was Thema weiterer Forschungsarbeit sein könnte.

2 Grundlagen

In diesem Abschnitt werden die Grundlagen der *Blockchain* und der darauf basierenden digitalen Währung *Bitcoin* erläutert. Diese bilden das Fundament für eine spätere Definition von *Proof of Useful Work*. Einleitend wird eine informelle Definition darüber gegeben, wie der Begriff *Nutzen* (*engl.: Usefulness*) in dieser Ausarbeitung verwendet wird.

2.1 Informelle Definition von Usefulness

Unter dem Begriff *Usefulness* wird in den nachfolgenden Abschnitten der Benefit verstanden, den man aus dem Gebrauch einer Sache ziehen kann[3]. Dabei kann es eine Unterscheidung in gesellschaftlichen und privaten Nutzen geben. Aufgrund der Tatsache, dass es sich bei der Blockchain um ein verteiltes System in einer Gemeinschaft (engl.: *community*) handelt, wird im Zuge dieser Arbeit die Sichtweise des gesellschaftlichen Nutzens betrachtet. Die Betrachtung des gesellschaftlichen Nutzens führt sehr schnell zu einer philosophischen Frage, auf die hier aber nicht näher eingegangen werden soll.

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

2.2 Blockchain und Bitcoin

Im Allgemeinen handelt es sich bei der *Blockchain* um ein dezentrales öffentliches Hauptbuch (engl.: *public ledger*) [Swan15]. Um bislang Transaktionen durchführen zu können, vertrauten Kunden „vertrauenswürdigen“ dritten Instanzen, beispielsweise Banken, denen sie ihr Geld oder Daten zur Verwaltung anvertrauen. Diese dritten Instanzen besitzen allerdings einen permanenten Zugriff auf private Date und Gelder und können jeder Zeit auf diese zugreifen. Probleme entstehen dann, wenn sich eine dritte Instanz fehlverhält und den Zugang zu Daten resp. Geld missbraucht. Andererseits kann auch ein kompromitierter Angriff auf das zentrale System der Instanz 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. Damit 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.
In der Blockchain wird jede Transaktion zu einem Block zusammengefasst und mittels kryptographischer Berechnungen mit anderen Transaktionsblöcken zu einer Kette, der „Block-Chain“, verbunden [Swan15]. 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 werden. So wird eine mehrheitliche Einigung zwischen den Knoten geschaffen. In der Blockchain wird immer die gesamte Historie an Transaktionen, die jemals durchgeführt wurden, chronologisch linear erweitert [Swan15].

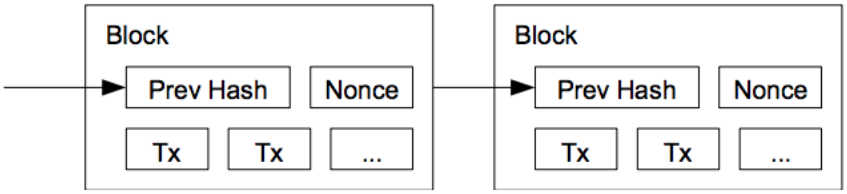

Abbildung 1: Verkettung von Transaktionsblöcken über den Hashwert des vorherigen Blocks [Naka08].

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” [Naka08] von einer unbekannten Einzelperson oder Gruppe mit dem Pseudonym Satoshi Nakamoto entwickelt wurde. Viele Vorgänger digitaler Währungen scheiterten daran, dass sie durch eine zentrale Instanz verifiziert werden mussten, um das Double-Spending-Problem (vgl. Abschnitt 2.2.3) 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 Entscheidungsinstanz existiert bzw. auf diese bewusst verzichtet werden soll, müssen Entscheidungen anders getroffen werden. Im Fall von Bitcoin wird ein mehrheitlicher Konsens durch einen Mechanismus bestimmt. Bei dem Konsensmechanismus handelt es sich um einen Algorithmus, der eine Einigung über den Status des Blockchain-Netzwerks zwischen den beteiligten Netzwerkknoten schafft. Dabei wird sichergestellt, dass alle Teilnehmer eine identische Kopie der aktuellen Blockchain besitzen. Wie bereits im vorherigen Abschnitt 2.2 erwähnt erhöht der Konsensmechanismus die Sicherheit von Bitcoin erheblich, da eine Manipulation von Daten ausgeschlossen ist, solange ein Angreifer nicht über 50% an Stimmen erhält (vgl. Abschnitt 2.2.3). Die Idee, die dahinter steckt ist, dass durch einen dezentralen Konsensmechanismus eine zentrale Kontrollinstanz zur Integritätsbestätigung obsolet wird. Für die Blockchain kommen verschiedene Konsensmechanismen wie beispielsweise Proof of Stake [KiNa12], Proof of Space [ReDe16], Proof of Work [Naka08] u.v.m. in Frage. Letzterer Mechanismus *Proof of Work* wird im nachfolgenden Abschnitt 2.2.2 genauer betrachtet, da dieser die Grundlage für das Proof of Useful Work Protokoll bildet.

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 eine Vergütung, im Fall von Bitcoin ist es eine gewisse Anzahl an Bitcoins, die sich an die Größe der Blockchain anpasst.

Proof of Work ist keine Neuentwicklung von Bitcoin, sondern wurde bereits früher gegen Denial of Service Attacken und für den E-Mail-Versand als Mechanismus gegen "Spams" 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 "Spam"-Mails nicht mehr lohnen sollte. 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 Definition 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. An dieser Stelle wird auch von *Hardness* (dt.: *Schwere*) [Will15] gesprochen. 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. Allerdings 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. Aus der Schwierigkeit, eine bestimmte Ausgabe für die Challenge zu finden ergibt, sich die *Difficulty*. Beispielsweise wird im Fall von Bitcoin eine Anzahl an führenden Nullen der Ausgabe der Hashfunktion vorgegeben. So kann durch mehr Nullen ein höherer Schwierigkeitsgrad erreicht werden, um eine Lösung zu finden.

2.2.3 Sicherheit

Die Sicherheit von Proof of Work basiert auf dem Ansatz, dass kein Teilnehmer mehr als 50% der Rechenleistung besitzen darf. Ansonsten kann ein einziger Teilnehmer die Blockchain kontrollieren, indem er die längste Kette beherrscht. Zwei der bekanntesten Angriffe auf die Blockchain, die hier kurz erläutert werden sollen, sind das Double-Spending-Problem und das Selfisch Mining [GKWG$^+$16] .
Der Gedanke hinter Double-Spending ist einen Coin zweimal für mehrere Transaktionen zu verwenden. Damit würde ein Angreifer mehr Coins verwenden, als er eigentlich besitzt.
Beim Selfish Mining (dt.: egoistisches Schürfen) behält ein einzelner Miner einen gültigen berechneten Block erst einmal für sich und veröffentlicht diesen später, um diesen an die Blockchain zu hängen. So verschafft sich der Miner einen Vorteil gegenüber anderen, indem er den nächsten Block anfängt zu berechnen. Die übrigen Miner würden so Berechnungen durchführen, die dann eventuell wieder verworfen werden müssen, da der Selfish Miner gewinnt. [GKWG$^+$16] hat Studien dazu untersucht, die belegen, dass ein Selfish Miner mit ursprünglichen 33% Mining Power eine effektive Leistung von 50% erhalten kann.
Laut [GKWG$^+$16] können die Angriffe Double-Spending und Selfish Mining dadurch verringert werden, indem alle Knoten des Systems immer synchronisiert sind.

2.3 Altcoins

Neben Bitcoin gibt es zahlreiche alternative digitale Währungen, sogenannte "Altcoins" [4], die versuchen die Technologie hinter Bitcoin zu verbessern. Nachfolgend werden zwei reale Umsetzungen von Usefulness vorgestellt: 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.

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

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

Primecoin [King13] versucht diesen Trade-off auf effiziente Weise zu lösen. Es ist die erste digitale Peer-to-Peer Währung, 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 von [King13] 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. Die Cunningham Chain 1. Art gibt vor, dass jede Primzahl um eins verringert als das doppelte des Vorgängers ist.

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

Cunningham Chain 2. Art. Bei der Cunningham Chain 2. Art muss jede Primzahl der Kette um eins größer als das doppelte der vorhergehenden Primzahl sein:

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 auch nur wieder 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 [5] oder das Primzahlen Theorem[6]. Die folgende Liste der University of Tennessee 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 zum vorherigen Beispiel 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. In POR 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.

[5] `https://en.wikipedia.org/wiki/Mersenne_prime`
[6] `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 Proof of Work (PoW) der Hardness-Anforderung gerecht werden. Damit ist gewährleistet, dass der Prover tatsächlich Arbeit aufgewendet hat. Weiterhin muss uPoW aber auch einen Nutzen (engl. *Usefulness*) erzeugen, der effizient zu verifizieren 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**. Die Lösungen delegierter Aufgaben sollen Aufwand in Form von Arbeit oder Energie garantieren.
>
> **Usefulness**. Aufgaben so an Teilnehmer delegiert werden, dass eine Lösung schnell und effizient verifizierbar ist und rekonstruiert werden kann.

Die größte Herausforderung ist, dass beide Eigenschaften immer für ein uPoW gelten müssen und stets zusammen betrachtet werden müssen, da ohne Hardness einerseits kein Proof of Work möglich ist und andererseits ohne Usefulness die Berechnungen von Proof of Work keinen Nutzen liefern. 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 nicht erfüllt. Das ist allerdings die Grundvoraussetzung für Proof of Work. Wird dagegen Hardness gewährleistet, aber zufällige Challenges so generiert, 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)`. 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 der neue `Recon`-Algorithmus) eingeführt:

- `Gen(x)` - randomisierter Algorithmus, der eine Challenge c_x in $\mathcal{O}(n)$ 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 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 die folgenden Eigenschaften gelten sollen:*

- ***Efficiency****: `Gen`, `Verify` und `Recon` sollen eine effiziente Laufzeit besitzen*
- ***Completeness****: Pr[Verify(c,s) = accept] = 1, für jede Challenge c und Lösung s*
- ***Soundness****: Pr[Verify(c,s) = accept] < neg(n), für jede Challenge c, Lösung s, sodass Recon(c,s)* $\neq f(x)$

- ***Hardness**: Für jedes Polynom l, jedes $x_1, \ldots x_{l(n)}$ und jeden Algorithmus $Solve_l$, der eine Laufzeit in $l(n) \cdot t(n)^{1-\varepsilon}$ besitzt, wobei $l(n)$ Challenges gegeben sind, soll gelten: $Pr[\forall i : Verify(c_i, s_i) = accept | (c_i \leftarrow Gen(x_i))_{i \in [l(n)]}, (s_1, \ldots, s_{l(n)}) \leftarrow Solve_l(c_1, \ldots, c_{l(n)})] < \delta(n)$*
- ***Usefulness**: Recon(c,s) = f(x)*

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 die Bedingung von Soundness auskommen. Bei einem PoW sind leigt kein Interesse daran, ob eine Lösung korrekt ist, sondern nur, ob das Finden einer Lösung Arbeit gekostet hat, was die Hardness-Bedingung erfüllt. Umso wichtiger wird diese Bedingung für ein uPoW-Schema, bei dem es uns daran gelegen ist aus einer Berechnung einen Mehrwert zu ziehen und keine Fake-Berechnungen zuzulassen. [BRSV17] beweist Proofs of Useful Work für das k-Orthogonal Vectors Problem (kOV), 3SUM und APSP. Darüber hinaus lassen sich weitere NP-schwere Probleme auf diese Probleme reduzieren [BRSV17] [Will15]. Besonders Eigenschaften aus der Graphentheorie passen auf das kOV-Problem.

3.1 Herausforderungen

Die Herausforderungen, die sich also für ein Proof of Useful Work ergeben 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 Probleminstanzen ist, dass gefundene Lösungen effizient verifizierbar sind. Da die Wahl willkürlicher Probleminstanzen dazu führen kann, dass eine Hardness nicht gewährleistet werden kann. Würde man beispielsweise eine einfache Primzahl vom Miner verlangen so wäre eine mögliche einfache Antwort eines Provers die Zahl "2". Es ist offensichtlich, dass beim Finden einer Lösung für diese Challenge 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-Bedingung. Darüber hinaus muss im Sinne der Usefulness ebenfalls gelten, dass falsche Berechnungen erkannt werden können. Zudem muss betrachtet werden ob ein entsprechendes generisches Framework in die Blockchain passt.

3.2 k-Orthogonal-Vectors-Problem

Der Gedanken von nützlichen Berechnungen wird wieder aufgegriffen und das Ziel wird sein die nutzlose SHA256-Hash-Berechnung gegen etwas, das wiederverwendet werden kann, auszutauschen.
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 [Will15]. Die Komplexitätsklassen vieler Probleme dieser Arbeit behandeln eben solche Probleme.

Nach [Will15] ergibt sich für NP-schwere Entscheidungsprobleme:

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 (k-OV-Problem) [Will15]. Nach [Will15] und [BRSV17] können Probleme aus der Graphenteorie 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 im Worst-Case hart ist [BRSV17] 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 $gOV_{n,d,p}^k$ 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:

$$gOV_{n,d,p}^k(U_1, ..., U_k) = \sum_{i_1,...,i_k \in [n]} \prod_{l \in [d]} (1 - u_{i_1 l}^1 \cdots u_{i_k l}^k)$$

[BRSV17] zeigt, dass das entwickelte Proof of Useful Work für die Probleme 3SUM und All-Pairs Shortest Paths basierend auf den Erkenntnissen von [Will15] eingesetzt werden kann. Hierbei gilt allerdings auch wieder, dass jedes Problem, das auf das k-OV, 3SUM oder APSP Problem reduzierbar ist, sich ebenso für das uPoW-Schema eignet. [BRSV17] weist besonders auf alle Probleme der Graphentheorie hin, die auf das k-OV-Problem reduzierbar sind. 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.

3.3 3SUM

Gegeben ist 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? Mit dieser Fragestellung im Hintergrund lassen sich Probleme finden, die sich auf dieses Problem reduzieren lassen und eventuell häufig in der Praxis auftreten. Diese Probleme werden auch 3SUM-schwer bezeichnet. Ein Problem heißt genau dann 3SUM-schwer, wenn es durch eine Lösung in $\mathcal{O}(n^{2-\varepsilon})$ eine Laufzeit in $\mathcal{O}(n^{2-\varepsilon})$ für das 3SUM Problem impliziert [GaOv95]. [GaOv95] hat gezeigt, dass es eine große Klasse an 3SUM-Problemen aus der Algorithmischen Geometrie gibt. Darunter sind nachfolgend drei Beispiele aufgelistet:

- Gegeben eine Menge von Linien in der Ebene. Gesucht werden drei Linien, die sich in einem Punkt treffen.
- Gegeben eine Menge von Liniensegment, die sich nicht kreuzen und achsenparallel sind. Gesucht wird eine Linie die diese in zwei nicht leere Mengen unterteilt.
- Gegeben eine Menge von n Punkten. Gibt es eine Linie, die 3 Punkte enthält?
- Given a set of triangles in the plane, does their union have a hole?

3.4 Mögliche Kandidaten

Es liegt in der Natur der Sache die Hashberechnung von Bitcoin durch eine nützliche Berechnung zu ersetzen. Dadurch werden immer wieder verschiedene Vorschläge getätigt[7], die Berechnungen sollten aus großen Projekten aus verschiedenen Bereichen der Astrophysik oder Biologie wie SETI@Home resp. Folding@Home stammen, um die Menschheit bei der Suche nach außerirdischer Intelligenz voranzutreiben oder DNA-bedingte Krankheiten zu heilen. Wie im vorhergehenden Abschnitt beschrieben kann jedes Problem, das auf das k-OV-Problem reduziert werden kann, an Prover delegiert 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 einen möglichen Einsatz im Proof of Useful Work Framework nach [BRSV17] evaluiert werden. Bei der Suche nach möglichen Kandidaten fällt der erste Gedanke auf Projekte aus Bereich des Distributed Computing[8]. Aus der Liste der Distributed Computings wurden zwei der berühmtesten und größten Projekte dem SETI@Home und Folding@Home die Eignung des Proof of Useful Work Schemas untersucht. Daraus ergeben sich folgende Kandidaten für ein Useful Proof of Work:

Orthogonal Vectors Problem. Wie bereits in Abschnitt 3.2 eignet sich das Orthogonal Vectors Problem sehr gut und wurde bereits von [BRSV17] bewiesen.

3SUM. Eignet sich nach [BRSV17] und [Will15] ebenfalls für ein uPoW.

All-Pairs Shortest Path. Gegeben ist ein gerichteter oder ungerichteter Graph mit Kantengewichten. Gesucht werden alle Distanzen zwischen jedem Knotenpaar aus dem Graphen. Ein Algorithmus der dieses Problem löst ist der Floyd-Warshall Algorithmus $\mathcal{O}(n^3)$. [BRSV17] beschreibt, dass ein uPoW für APSP möglich ist.

Folding@Home. Volenteer-Computing-Projekt der Stanford University aus dem Bereich der Biologie und Medizin. Das Projekt hat den Hintergrund, dass viele Krankheiten wie Krebs oder Alzheimer durch Fehler in der Faltung von Proteinen entsehen können. Das Ziel des Projekts ist es mittels eines Netzwerks von verteiltetn Computersystemen den räumlichen Aufbau von Proteinen verstehen zu können, um somit die Entstehung von Krankheiten identifizieren und gegebenenfalls sogar heilen zu können. Eine Berechnung auf den Universitätsinternen Rechensystemen würde mehrere Jahrhunderte dauern [9]. Aus diesem Grund wird das Projekt auf mehrere Rechner eines Netzwerks verteilt und gehört damit zu den Projekten des Distributed Computing.

SETI@Home. SETI (engl.: *Search for Extra-Terrestrial Intelligence*)[10] ist ähnlich wie das vorherige Projekt ein Projekt des Distributed Computing. Inhalt des Projekts ist die Suche nach außerirdischer Intelligenz im Weltall. Ein Radioteleskop zeichnet Radiosignale aus der Himmelsrichtung auf, die dann im Anschluss ausgewertet werden.

Bei den Kandidaten handelt es sich zwar um interessante Vorschläge, jedoch sind sie nicht so einfach integrierbar. Der wesentliche Unterschied zwischen Bitcoins SHA256-Hashberechnungen und den Distributed Computing Projects wie SETI@home und Folding@home ist, dass letztere beiden nicht effizient verifizierbar sind. 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 das jeweilige Projekt mit richtigen Lösungen zu befüllen. Werden diese Projekte allerdings zum Bitcoin Mining verwendet

[7] `http://learningspot.altervista.org/blockchain-mining-proof-of-useful-work`
[8] `https://en.wikipedia.org/wiki/List_of_distributed_computing_projects`
[9] `https://de.wikipedia.org/wiki/Folding@home`
[10] `https://de.wikipedia.org/wiki/SETI@home`

geht das Ziel des Projektes verloren, da ein Teilnehmer im Bitcoin-Netzwerk eventuell kein Bedürfnis für sich sieht ein Projekt voranzutreiben und einfach Fake-Daten für eine Gewinnmaximierung berechnet. Eine Unterscheidung zwischen falschen und echten Werten ist nicht möglich. Das wiederum hätte keinen Mehrwert für das jeweilige Projekt, was wiederum einen Bruch der Usefulness von uPoW zur Folge hat.

3.5 Blockchain-Schema

[BRSV17] stellt ein Blockchain-Schema vor, das den Anforderungen eines Proof of Useful Work gerecht werden soll. Jedes Blockchain-basierte System soll das uPoW nutzen können, um einen Proof of Work einerseits zu ermöglichen und andererseits als Quelle von Berechnungen für Delegierende gelten [BRSV17].
Bitcoin setzt PoW in einem riesigen Umfang ein. Dahinter steht eine große Community, die mit Problemen versorgt werden kann, um diese zu lösen. Ball et al. merken allerdings an, dass ein generisches uPoW nicht einfach in ein Blockchain-Schema, wie Bitcoin es verwendet, hineinpassen würde. Ungültige oder modifizierte Transaktionen müssen dazu führen, dass das PoW oder uPoW ungültig wird. Die Blockchain ist weiterhin eine Chronik von Transaktionsblöcken, die miteinander verkettet sind. Wie bei PoW muss auch für das uPoW gelten, dass Änderungen an Blöcken immer erkannt werden sollen. Eine Verkettung ergibt sich weiterhin aus dem Link zum Hash des vorherigen Transaktionsblocks, wie in Abbildung 2 dargestellt. Über ein Public Board können Delegierende ihre Probleme an Prover delegieren, die ein Proof of Work durchführen müssen. In Abbildung 2 wird ein solches Public Board dargestellt, das für Delegierende dazu dient ihre Probleme (f, x) zur Berechnung zur Verfügung zu stellen. Beim Mining-Verfahren nimmt sich ein Miner für die Durchführung eines PoW ein Problem vom *Problem Board* und löst es. Sobald die Verifizierung erfolgreich war kann der generierte Block an die Blockchain angehängt werden. Dabei kann jede Art von Priority Scheduling oder Zufallsverfahren zum Wählen der nächsten Challenge verwendet werden. Wie auch in 2.2.2 beschrieben kann der Worker nun mit der Instanz x sich die Challenge $c_x = x + rt | t \in [D + 1]$ mit `Gen(x)` generieren wobei $r = H(currentBlock)$ für ein zufällig gleichverteiltes Orakel H.

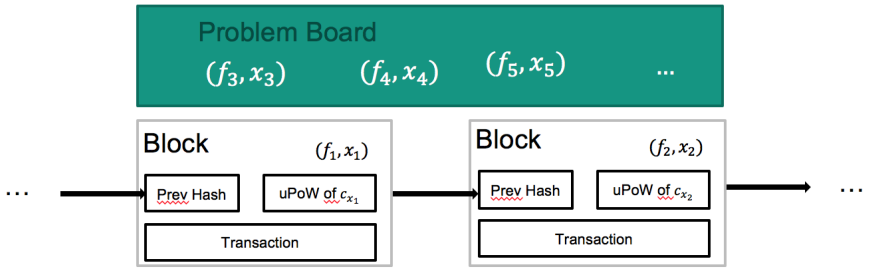

Abbildung 2: Blockchain-Schema für ein uPoW nach [BRSV17]. Ein Problem hat die Form (f,x), wobei f ein Label des Problems ist. x ist die Instanz, für die es $f(x)$ zu lösen gilt. Ein Block besteht aus: dem Problem (f, x); dem Hash des Vorgängers, prevHash; einer Transaktion; der Challenge $c_x = x + rt | t \in [D + 1]$.

Die Gültigkeit eines Blocks kann verifiziert werden, indem die Hash des vorhergehenden gültigen Blocks auf Richtigkeit überprüft wird. Außerdem sollte ebenfalls überprüft werden, ob die Challenge (f, x) nicht bereits vorher bereits gelöst wurde. Falls zwei Miner gleichzeitig versuchen einen Block hinzuzufügen und einer schneller ist als der andere, geht die Challenge (f, x) nicht verloren, sondern der Miner versucht diese zu lösen, bis er das Wettrennen für sich entscheiden kann. Durch die Usefulness-Eigenschaft kann ein Delegierender schnell aus dem uPoW ein $f(x)$ rekonstruieren.

4 Zusammenfassung und Ausblick

Beschrieben wurde ein allgemeines Framework zu einem Proof of Useful Work (uPoW) basierend auf der Arbeit von [BRSV17]. Wie beschrieben kann dieses uPoW für Bitcoin eingesetzt werden. Die Miner können weiterhin zur Paralleliseriung der Arbeit "Mining Pools" erstellen wobei das Framework robust gegenüber Fehlern ist [BRSV17]. Proof of Work wird durch die Hardness Bedingung aus Abschnitt 3 gewährleistet. Usefulness gewährleistet, indem die Möglichkeit geboten wird praktische Probleme, die auf das Orthogonal Vectors Problem, 3SUM oder APSP reduziert werden können, über das Board an Miner zu delegieren und die Lösungen zu rekonstruieren.
Berechnungen, die sowieso auf riesigen Supercomputern stattfinden würden, könnten auf Rechnern eines verteilten Netzwerks ausgelagert werden. Somit würde man die Kosten für Stromverbrauch auf der gesamten Welt reduzieren können.
Besonders für die Forschung aus der Theoretischen Informatik ist der Ansatz des uPoW nach [BRSV17] 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* erfüllt. Würde das theoretische Konzept in der Praxis so umgesetzt werden könnte es höchstwahrscheinlich 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.
Eine Implementierung des vorgestellten uPoW gibt es nicht. Somit kann bisher auch keine Aussage über einen praktischen Einsatz getroffen werden. Für weiterführende Arbeiten müsste die Umsetzbarkeit überprüft werden. Weiter ist noch ungeklärt, wer Challenges über das Public Board delegieren darf. Denn dann müssten eventuell auch die delegierten Probleme zunächst auf Hardness überprüft werden. In diesem Zusammenhang müsste auch eine Fairness garantiert werden, denn sonst könnten Probleme einiger Delegierender bevorzugt werden.
Eine weitere Ungeklärtheit ist, was passieren soll, wenn keine Probleme auf das Public Board delegiert werden. Stagniert das System oder können stattdessen Primzahlen-Berechnungen wie bei Primecoin durchgeführt werden?
Für zukünftige Arbeiten wären dies weitere Einstiegspunkte, die es zu erforschen gilt, um ein solides uPoW-Konzept zu enwtickeln.

Literatur

[Ande13] Nate Anderson. Mining bitcoins takes power, but is it an "environmental disaster?". *Ars Technica. April* Band 14, 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.

[GaOv95] Anka Gajentaan und Mark Overmars. *On a class of O(n2) problems in computational geometry*. 1995.

[GKWG+16] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf und Srdjan Capkun. On the Security and Performance of Proof of Work Blockchains. 2016.

[KiNa12] Sunny King und Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August* Band 19, 2012.

[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.

[Mokh17] Melissa Mokhtari. A Bitcoin Miner and an Environmentalist Walk Into a Bar... *Blockchainatberkeley Blog*, 2017.

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

[ReDe16] Ling Ren und Srinivas Devadas. Proof of space from stacked expanders. In *Theory of Cryptography Conference*. Springer, 2016, S. 262–285.

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

[ShWa08] Hovav Shacham und Brent Waters. Compact proofs of retrievability. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2008, S. 90–107.

[Swan15] Melanie Swan. *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc. 2015.

[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.