Vom Bankcoin zum Bitcoin

Dokumentation zum Vortrag von Bernhard Schmitt

Inhalt

[Grundlagen 3](#_Toc42171012)

[Hashfunktionen 3](#_Toc42171013)

[Digitale Signaturen 4](#_Toc42171014)

[Digitales Bargeld 4](#_Toc42171015)

[Digitale Zeitstempel 4](#_Toc42171016)

[Trusted Timestamp 4](#_Toc42171017)

[Linked Timestamp 5](#_Toc42171018)

[Proof of Work 6](#_Toc42171019)

[Das Bitcoin Protokoll 7](#_Toc42171020)

[BankCoin 7](#_Toc42171021)

[NaiveCoin 7](#_Toc42171022)

[TransactionCoin 9](#_Toc42171023)

[PublicAnnouncementCoin 10](#_Toc42171024)

[ElectionCoin 10](#_Toc42171025)

[Proof-of-Work-Coin 11](#_Toc42171026)

[BlockchainCoin 11](#_Toc42171027)

[IncentiveCoin 14](#_Toc42171028)

[Bitcoin 14](#_Toc42171029)

[Zusammenfassung 16](#_Toc42171030)

[Quellen 17](#_Toc42171031)

# Grundlagen

Das Ziel ist es hier die Sicherheitsmechanismen im Bitcoin Protokoll leicht verständlich zu erklären. Daher werde ich hier zuerst auf ein paar Grundlagen eingehen. Kryptografische Grundbegriffe wie Hashfunktionen und digitale Signaturen sind Voraussetzung, um zu verstehen wie eine Blockchain funktioniert. Wir auf bekannte Probleme wie die Double-Spending-Attacke stoßen wenn wir versuchen Digitales Bargeld mithilfe von Digitalen Signaturen zu erschaffen.

Wir werden uns zu beginn auch Digitale Timestamps anschauen, weil man mithilfe dieser gut die Sicherheitskonzepte von Hashfunktionen und digitalen Signaturen illustrieren kann. Eine Blokchain ist ja praktisch nichts anderes als eine digitale Timestamp Methode.

Zum Schluss werden wir bei den Grundlagen nochmal auf das Konzept „Proof-of-Work“ eingehen da dieser ein wichtiger Bestandteil der Blockchain ist.

## Hashfunktionen

Mit einer Hashfunktion kann man die Echtheit eines Datensatzes prüfen. Hier ein Beispiel:

Print(hashlib.sha256(b“Diese Dokumentation ist nur eine drei“).hexdigest())

* a709067a6150468297588c67b72202aae237db898f3ea6ba24837bd24fbe1edf

Herr Lehmann hat die Benotung für die Dokumentationen gehasht und sich den Hash schnell per mail geschickt. Danach hat er seinen Rechner an gelassen und Bernhard ist schnell an den Rechner gegangen und hat die Bewertung in „Diese Dokumentation ist eine glatte eins“ geändert. Zuhause will Herr Lehmann das Zeugnis schreiben und hasht sicherheitshalber nochmal seine Bewertungen:

Print(hashlib.sha256(b“Diese Dokumentation ist eine glatte eins“).hexdigest())

* b45b570ac288afb313198be752b74fe0c4eed60cc11d13e69cb00ac8fe95e9ed

Man sieht das sich der Hash geändert hat. Herr Lehmann kann nicht sagen was genau geändert wurde, aber es ist nicht sein ursprünglicher Text.

Bei einer Hashfunktion hat die Menge des Inputs keinen Einfluss auf die länge des Outputs. Man kann die ganze Bibel hashen und hat trotzdem den 256-Bit ausgabewert (meist wird dieser Wert in Hexadezimalwerte umgewandelt).

An eine Hashfunktion werden zwei Sicherheitseigenschaften gestellt. Sie muss eine Einwegfunktion sein und Kollisionsresistent.

Eine Einwegfunktion bedeutet das es quasi unmöglich ist aus dem Hash den ursprünglichen Text zu rekonstruieren.

Kollisionsresistent bedeutet das es quasi unmöglich ist einen Text zu finden welcher den gleichen Hash liefert.

## Digitale Signaturen

Wir werden hier nur kurz auf asynchrone Signaturen eingehen da diese ausführlich in einer anderen Dokumentation erklärt werden.

Herr Lehmann will seine Zeugnisnoten zu Frau Becker schicken. Um sicherzugehen das die Noten wirklich von Herrn Lehmann kommen will er seine Nachricht digital Signieren. Dafür erstellt er sich zuerst ein Schlüsselpaar. Einen PrivateKey den Signierschlüssel, diesen hält er geheim und mit diesem kann er seine Signatur „Das hier ist wirklich von Herrn Lehmann“ verschlüsseln. Den Publickey oder Verifikationsschlüssel lässt Herr Lehmann Frau Becker so zukommen das Sie sich sicher sein kann das dieser von Herr Lehmann kommt. Nur mit Hilfe des Verifikationsschlüssel kann Frau Becker die Nachricht entschlüsseln und wenn ein lesbarer Text rauskommt weiß Sie das die Noten tatsächlich von Herr Lehmann kommen.

## Digitales Bargeld

Regeln für Digitales Bargeld

Geld wird nur direkt vom Zahlenden zum Zahlungsempfänger gegeben ist die wichtigste Regel. Außerdem muss gewährleistet sein:

1. Jeder kann die Echtheit des Geldes bestätigen
2. Nur der Herausgeber kann Geld erschaffen

Mit einer vom Herausgeber Signierten Nachricht wie:

*Diese Geldnote hat Seriennummer 8124 und ist 100€ wert*

Sind beide Kriterien erfüllt. Im Gegensatz zu echtem Bargeld kann digitales Bargeld aber einfach vervielfältigt werden. So kann Frau Becker diese Geldnote einfach an Herr Lehmann und Herr Trutz schicken. Das ist die Double-Spending-Attacke der wir uns noch widmen werden.

## Digitale Zeitstempel

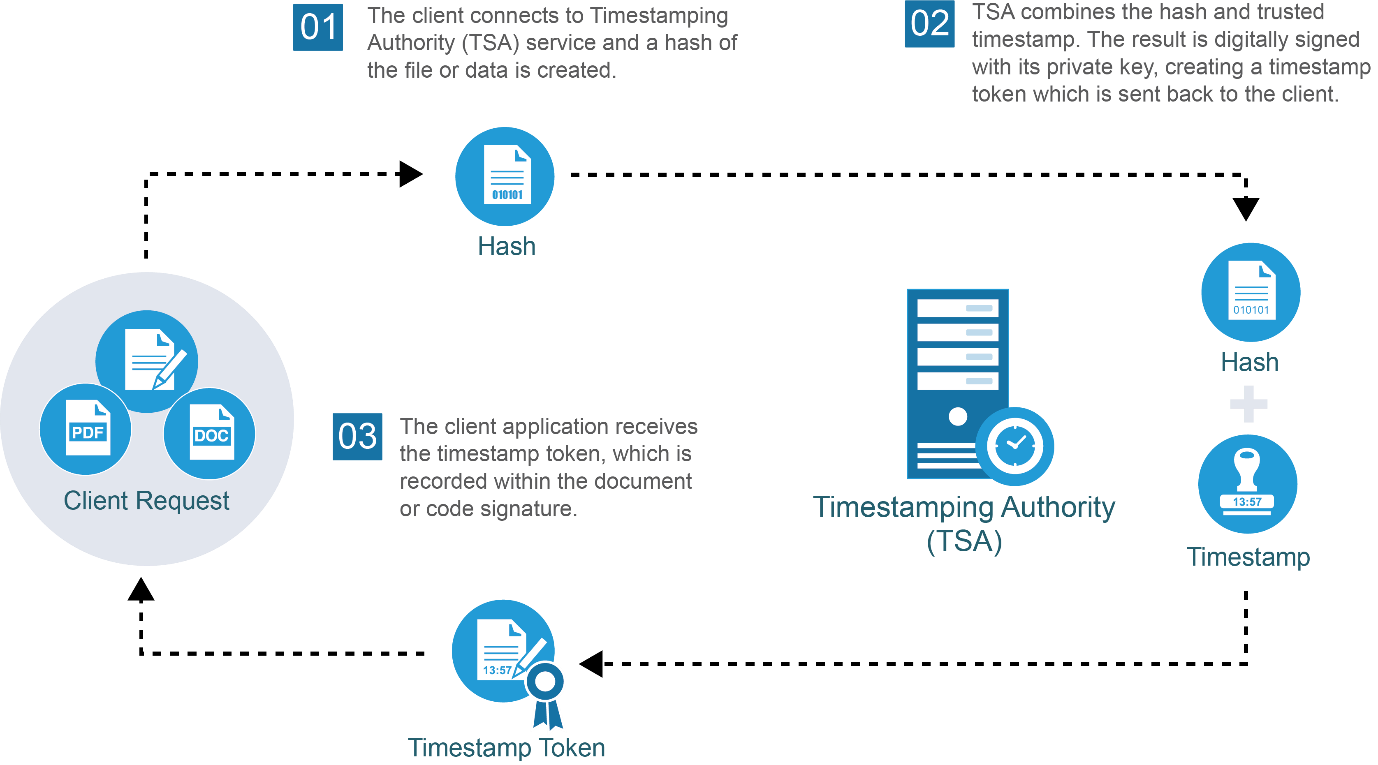
Digitale Zeitstempel dienen dazu die Existenz einer bestimmten Datei zu einem bestimmten Zeitpunkt zu bestätigen.

Es gibt hierfür 2 möglichkeiten die wir uns kurz angucken wollen.

* Trusted Timestamp
* Linked Timestamp

### Trusted Timestamp

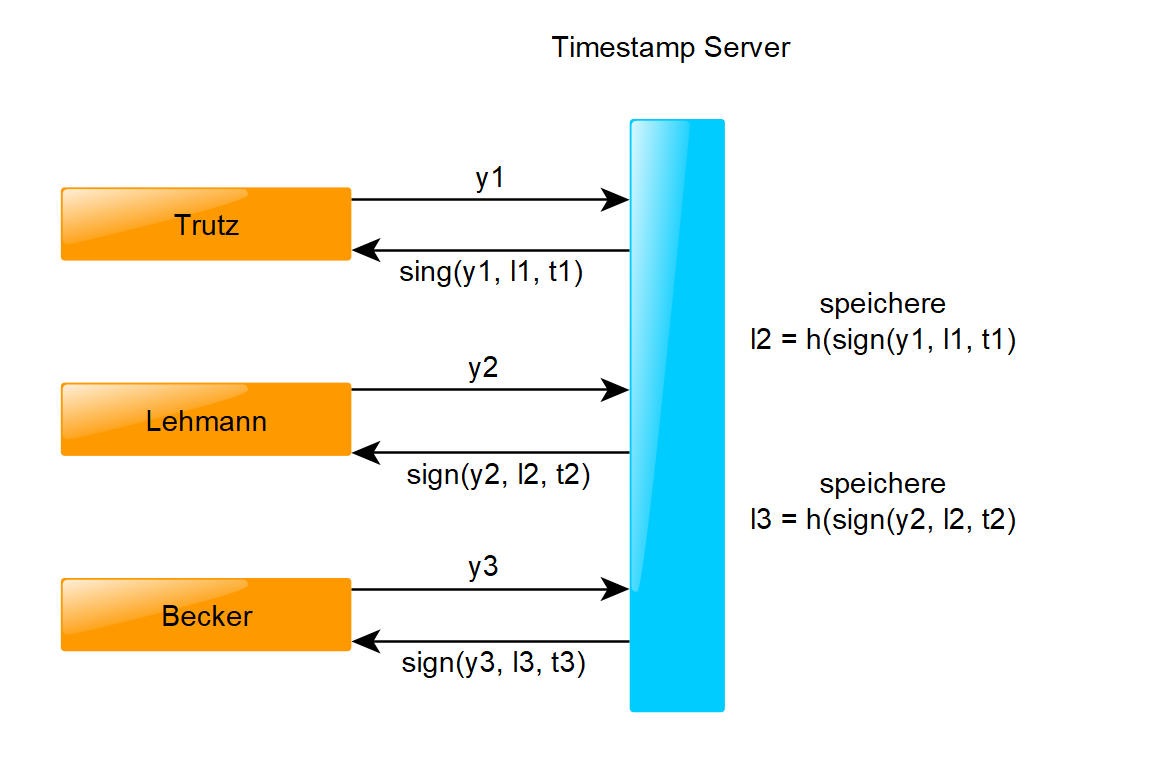
Hierbei wird die Datei welche mit einem Zeitstempel versehen werden soll gehashed. Dieser Hash wird an eine Vertrauenswürdige Institution geschickt. Dort wird der Timestamp mit dem Hash verbunden und digital signiert dann wird es zurück zum Kunden geschickt.



### Linked Timestamp

Dieses verfahren ist für uns besonders Interessant da es wie die Blockchain funktioniert.

Hier werden die gesendeten Dateien miteinander verknüpft. So wird gewährleistet das Herr Lehmann seine Datei nach Herr Trutz aber vor Frau Becker „gestempelt“ hat.



## Proof of Work

Als Proof of Work bezeichnet man einen Beweis das eine bestimmte Arbeit geleistet wurden ist. Eine Anwendung ist das Hashchash Protokoll welches wir uns ansehen, weil es auch im Bitcoin Protokoll verwendet wird.

Wir sagen haben ein Textdokument wo „ITS ist das beste Fach“ drin steht. Wenn wir diesen Text Hashen bekommen wir einen Hash.

*beb873e36c6a0a216fa8e57ad8db5617b997e72c7c04612bdfd04f9e3e535b6a*

in binär:

*10111110101110000111001111100011….*

Jetzt sage ich, ich will das du mir beweist das dein Rechner Arbeit geleistet hat (Proof of Work). Also will ich das du am Anfang der Datei etwas einfügst so das der Hash mit 20 x 0 beginnt.

Nun Probiert der Rechner die Möglichkeiten durch:

*„1234 ITS ist das beste Fach“*

*„Klaus ITS ist das beste Fach“*

*„x412 ITS ist das beste Fach“*

*„0101 ITS ist das beste Fach“*

Bis er einen Hash erhält welcher mit 20 x 0 beginnt.

Ein durchschnittlicher Rechner braucht dafür 152 millisekunden. Bei 30 x 0 braucht er 3min und bei 40 x 0 braucht er 1,8 Tage. Die Schwierigkeit steigt also Exponentiell und lässt sich leicht anpassen. Um ein Bitcoin Hashpuzzle zu lösen muss man 70 x 0 erschaffen.

# Das Bitcoin Protokoll

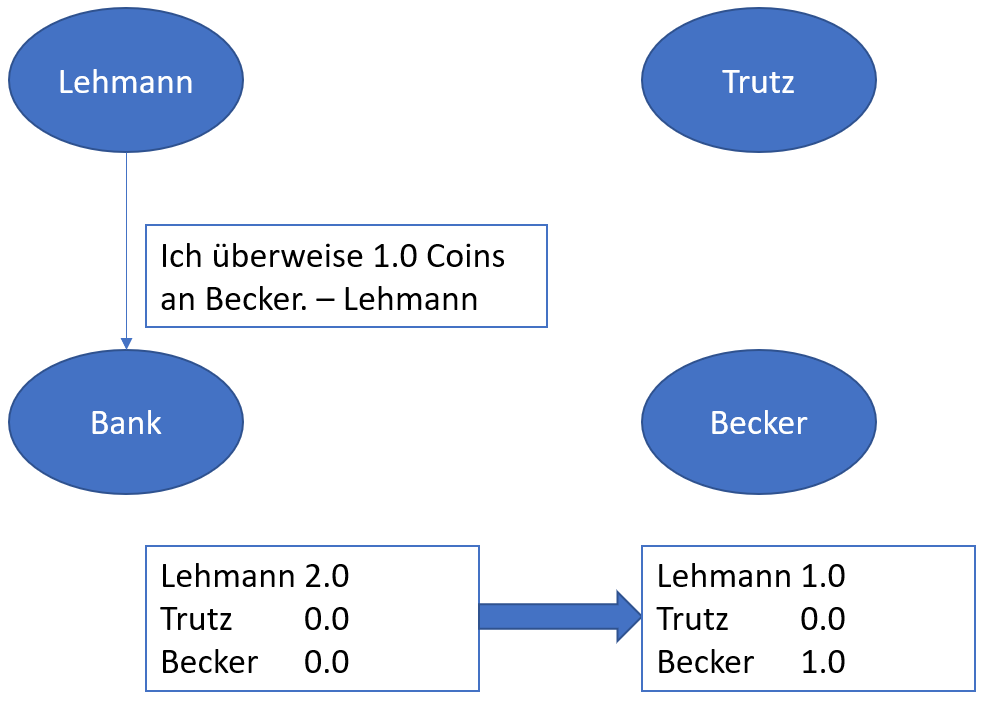
Wir versuchen ein Protokoll für Digitales Bargeld zu entwickeln. Wir gehen dafür Stück für Stück vom E-Banking bis zum Digitalem Bargeld stück für stück die Notwendigen Regeln durch.

## BankCoin

Hier haben wir E-Banking wie wir es kennen.

Regeln:

* Die Bank verwaltet alle Kontostände
* Jeder Teilnehmer schickt seine Transaktionen an die Bank
* Die Bank akzeptiert gültige Transaktionen und aktualisiert entsprechend die Kontostände



Probleme:

* Bank kann beliebig viel Geld erzeugen (Inflation)
* Transaktionen können abgewiesen werden (Zensur)

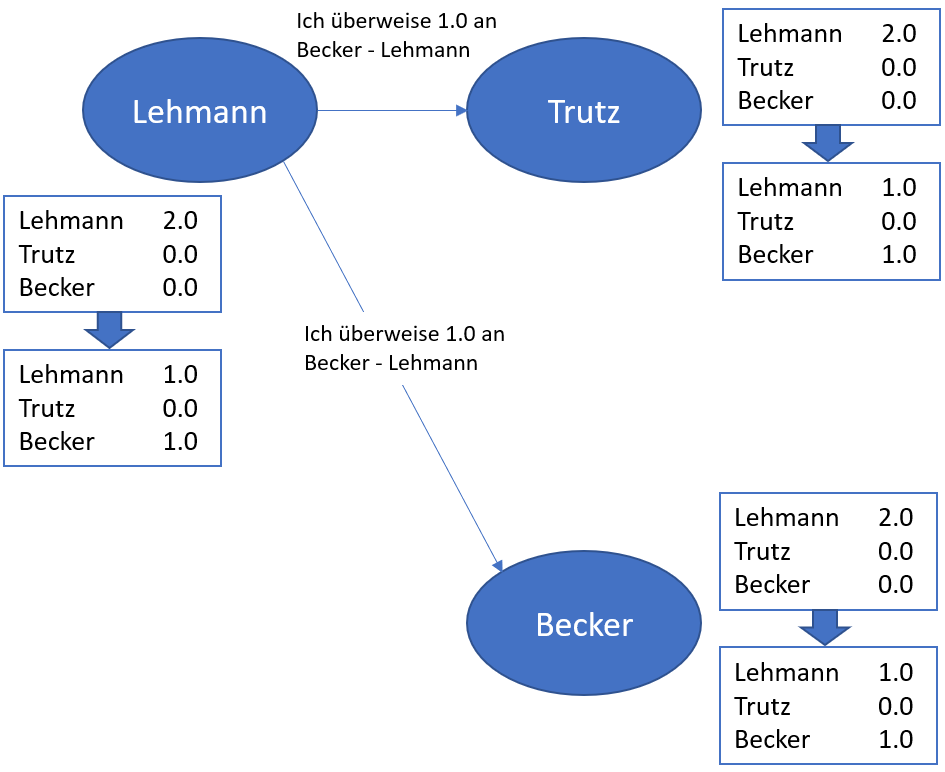
Deshalb brauchen wir Digitales Bargeld bei dem es keine Institution zwischen den Teilnehmern gibt!

Lösung: Alle verwalten alle Kontostände

## NaiveCoin

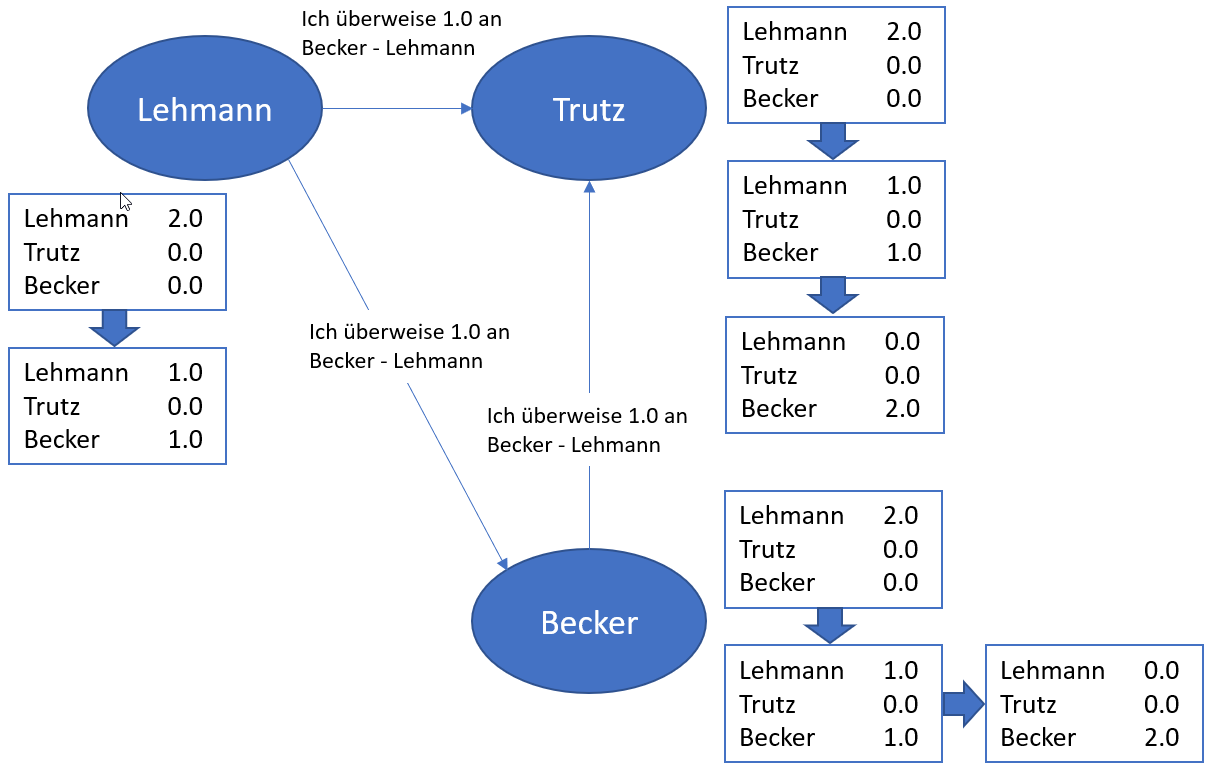
Regeln:

* Jeder Verwaltet alle Kontostände
* Jeder schickt seine Transaktionen an alle Teilnehmer. Diese sind natürlich Digital Signiert.
* Jeder Teilnehmer akzeptiert alle gültigen Transaktionen und aktualisiert entsprechend die Kontostände



Probleme:

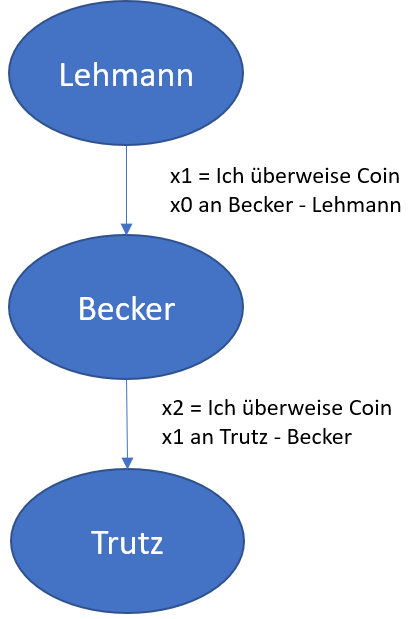
* Eine Replay-Attacke. Ein Teilnehmer nimmt die Nachricht und schickt diese gezielt nochmal an alle Teilnehmer des Netzwerkes mit Ausnahme des Ursprünglichen Verfassers. Nun hat die Mehrheit im Netz die gefälschten Kontostände.
* Auch eine Seriennummer würde das Problem nicht lösen da man den Coin gezielt an den Ursprünglichen Besitzer schicken kann und dann erneut eine Replay Attacke ausführen kann.



Lösung:

Jeder Coin braucht die gesamte Transaktionshistorie

## TransactionCoin



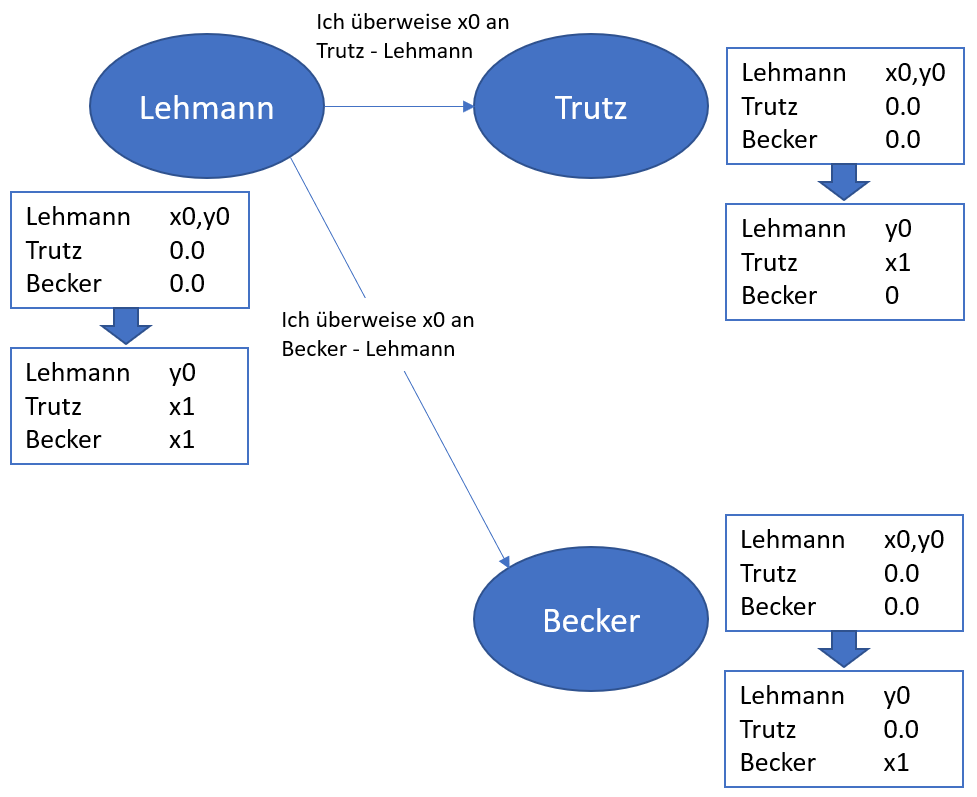
Coin x2: „Ich überweise Coin (x1 = ich überweise Coin x0 an Becker – Lehmann) an Trutz – Becker“

Regeln:

* Jeder Coin ist die Transaktion die ihn Übertragen hat.

Problem:

* Double Spending Attacke



Lösung: Eine Öffentliche Verkündung verhindert das man gleiche Coins an verschiedene Teilnehmer sendet.

## PublicAnnouncementCoin

Regeln:

* Es werden nur Überweisungen akzeptiert die öffentlich verkündet wurden.
* Das ist möglich, wenn sich alle im selben Raum befinden und einer Laut die Transaktionen vorliest

Problem:

* Wie lösen wir das für das Internet
* Wenn immer die gleiche Person Verkündet haben wir wieder quasi eine Bank mit einer Person die alles kontrolliert
* Alle weiteren schritte versuchen dieses Problem zu lösen

Lösung: Als erstes lösen wir das Kontroll-Problem in dem immer eine zufällige Person die Verkündung übernimmt.

## ElectionCoin

Regeln:

* Teilnehmer führen Transaktionen nicht direkt aus sondern speichern diese in einen Zwischenspeicher (pool).
* Periodisch wählt das System einen Teilnehemr aus. Jeder weiß wer der Auserwählte (leader) ist.
* Der Leader signiert (bestätigt) den Transaktionspool und schickt ihn an alle Teilnehemer
* Wenn ein Teilnehmer den signierten Pool erhält validiert er diesen und aktualisiert seine Transaktionsdatenbank.

Damit wir nicht zu viele Probleme auf einmal zu lösen haben treffen wir die Annahme das der Leader immer ehrlich ist und an alle den gleichen signierten Pool schickt.

Da eine Transaktion die im Pool ist nicht garantiert aufgenommen wird nennt man das unbestätigt (unconfirmed)

Wenn eine Transaktion in der Datenbank aufgenommen wurde nennt man das bestätigt (confirmed)

Probleme:

* Wie wählen wir den Leader?
* Ein Betrüger die die Macht über das Netzwerk erlangen will könnte sich mehrere Accounts oder IP’s oder ähnliches erstellen und sich als mehrere Teilnehmer ausgeben.

Lösung: Wir brauchen Merkmale die nicht einfach und billig in großer Zahl zu generieren sind. Daher wählen wir Rechenkapazität!

## Proof-of-Work-Coin

Regeln:

* Alle Arbeiten kontinuierlich am Hashtag Puzzle
* Sobald ein teilnehemer das Puzzle löst ist er Leader und Schickt den Transaktionspool zusammen mit der Puzzlelösung
* Sobald ein Teilnehmer den Pool des Leaders erhält validiert er anhand der Puzzlelösung und verwirft den eigenen Transaktionpool
* Es geht von vorne los…

Probleme:

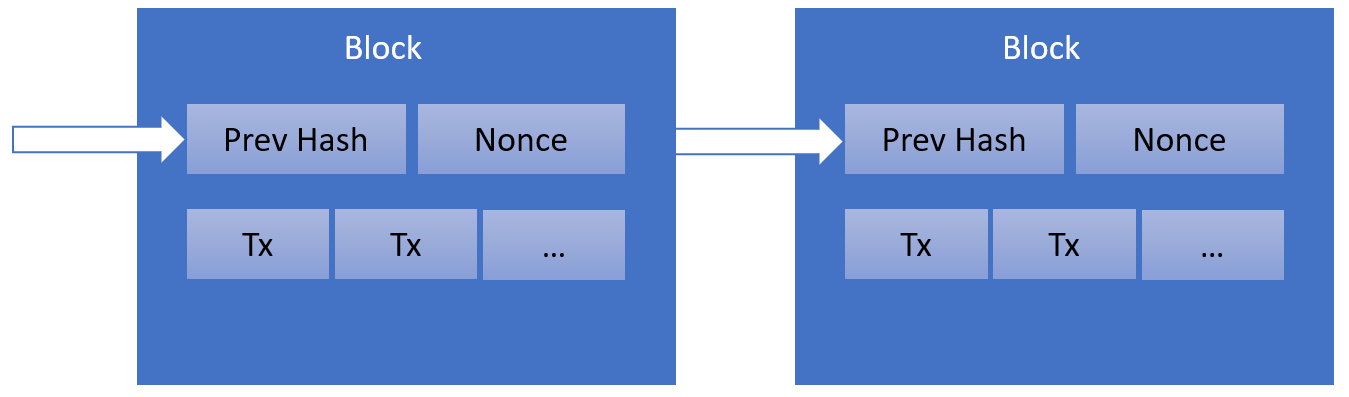
* Zwei Teilnehmer lösen das Puzzle zur selben Zeit. Nun haben wir zwei Transaktionsdatenbänke da die Transaktionspools durch Übertragungsfehler und Attacken unterschiedlich aussehen können. Das System ist nicht mehr im Konsens

Lösung: Wir erinnern uns an Linked Timestamps! Wir verlinken nun auch unsere Transaktionspools und haben nun unsere Blockchain. Hier gilt immer nur die längste gültige Kette und das führt dazu das unser System immer wieder zu einem Konsens findet.

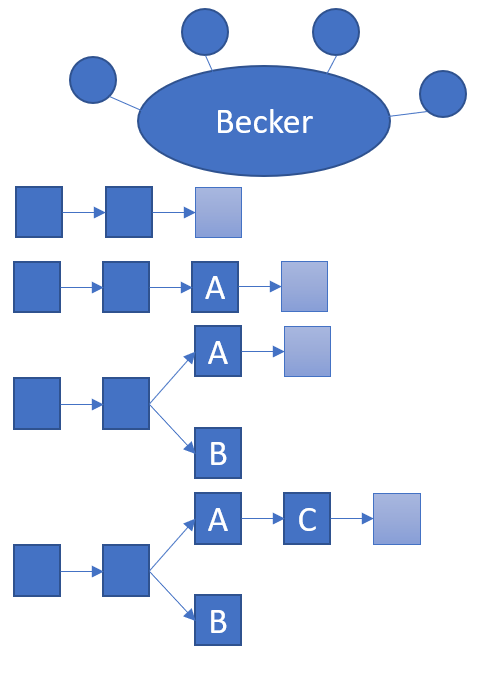
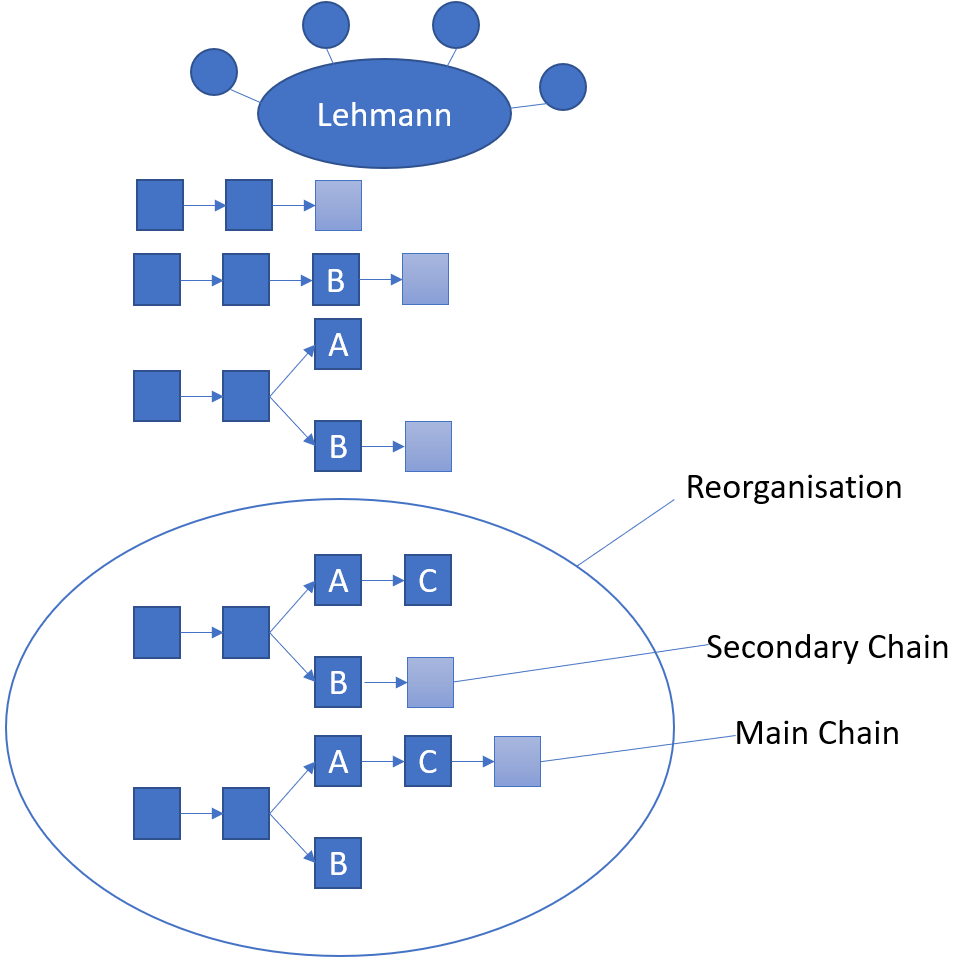
## BlockchainCoin

Regeln:

* Der Leader verkündet den Block. Dieser enthällt Den Transaktionspool, Die Puzzlelösung (nonce) und den Hash des vorigen Blocks (mittlerweile kann man auch andere Infos in die Blöcke schreiben aber das ist für unser Beispiel erstmal unrelevant). Blöcke Bilden eine Kette. In jedem Block ist der Hash des vorangegangenen Blocks. Ausnahme bildet der erste Block (Genesisblock).



* Das Hashcash-Puzzle ist nicht nur der Pool sondern auch der Hash des vorigen Blocks.
* Jeder Teilnehmer validiert alle Blöcke
* Jeder Teinlehmer arbeitet an dem Puzzle basierent auf dem Block in der längsten ihm bekannten Blockchain

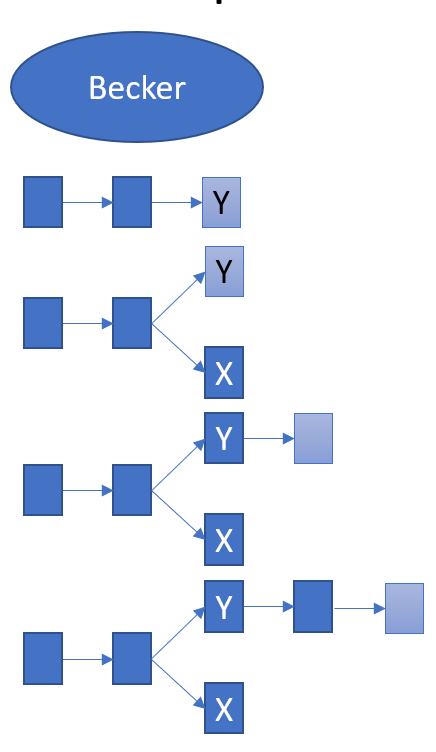
Das Problem einer Spaltung ist nun gelöst das man immer zu einem Konsens finden wird.

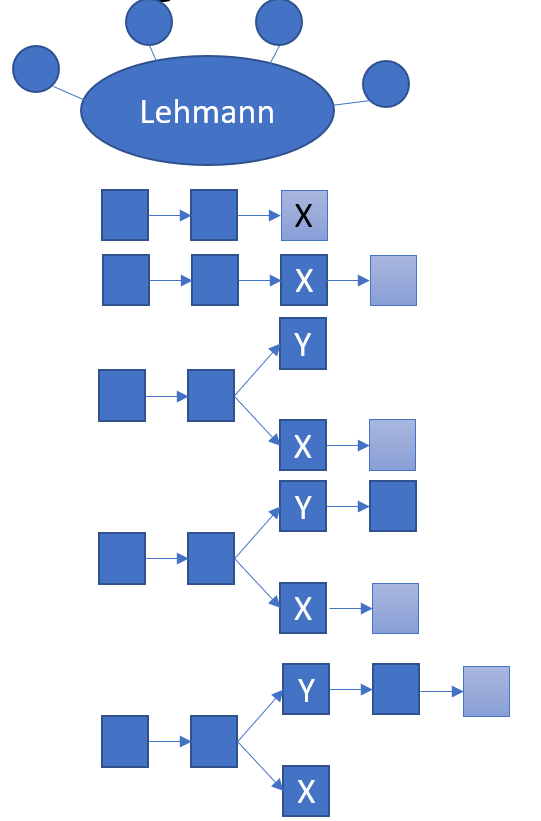
Hier werden Herr Lehmann (und Teilnehmer die ihm nahe sind) und Frau Becker(und Teilnehmer die ihm nahe sind) gleichzeitig fertig und haben nun zwei unterschiedliche Transaktionsdatenbanken A und B. Da beide Ketten gleichlang sind arbeiten beide einfach mit ihrer Datenbank weiter. Nun ist Frau Becker aber schneller als Herr Lehmann. Sobald Herr Lehmann Frau Beckers neue Blockchain mit Datenband A und C bekommt sieht er das diese Kette nun länger ist und wechselt auf die längere Kette. Es kann natürlich auch vorkommen das Herr Lehmann und Frau Becker zwei oder drei oder x mal gleichzeitig fertig werden, aber irgendwann ist einer schneller und der andere übernimmt dessen Kette.

Das Puzzles gleichzeitig gelöst werden ist unwahrscheinlicher mit:

* Jedem mal das mal gleichzeitig fertig wurde
* Je größer die Zwichenblockzeit ist. Das bedeutet wie lange dauert es durchschnittlich bis eine Puzzlelösung gefunden wurde
* Je kleiner die Blockpropagierungszeit ist. Das bedeutet wie schnell kann im Netzwerk der gelöste Block an alle verkündet werden.

Da es zu einer kurzzeitigen Spaltung kommen kann wartet man gerne mehrere Transaktionsbestätigungen ab. Wenn Herr Lehmann nachdem Frau Becker ihm 1 Bitcoin geschickt hat sofort sein Auto verkauft kann es dazu kommen das die Transaktion nie bestätigt wird und er sein Geld nie bekommt. Wie wir oben gesehen haben kann es selbst wenn seine Transaktion ein mal bestätigt wurde dazu kommen das diese verworfen wird weil das Netzwerk kurzzeitig keinen Konsens hat. Die Faustregel ist man soll 6 Blöcke abwarten. Es gilt als unwahrscheinlich das es jemals eine so große Reorganisation gegeben hat.

Problem: Die Double Spending Attake

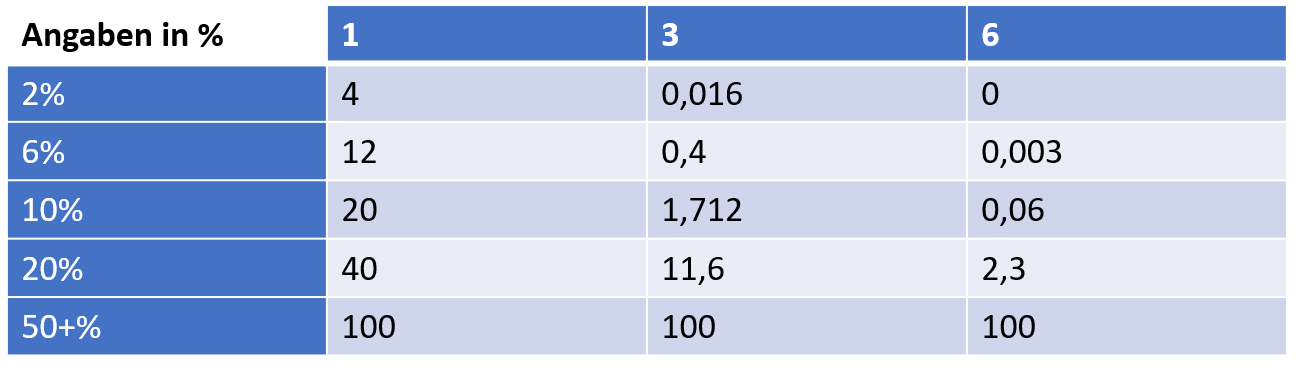
*Y: Ich überweise 1 coin an Lehmann – Becker*

*X: Ich überweise 0 coin an Lehmann – Becker*

Hier haben wir eine Double-Spending-Attacke. Frau Becker überweist Herr Lehmann einen Coin (Y). Herr Lehmann wartet ab bis diese Transaktion bestätigt ist(in der Grafik ist es eine Bestätigung es können aber auch mehr sein). Nun gibt Herr Lehmann das Auto raus und arbeitet mit dem Netzwerk an der Blockchain mit dem X weiter da diese auch die längste ist.

Sobald Frau Becker das Auto hat hält Sie sich nicht an die Regel an der längsten Kette zu Arbeiten sondern Arbeitet einfach an ihrer eigenen Kette weiter mit der Überweisung X wo sie kein Geld bezahlt. Jetzt beginnt ein Wettrennen. Da das Lösen des Hashcash-Puzzle ein Zerfallsprodukt ist kann man auch direkt die richtige Lösung raten. Darauf hofft Frau Becker und tatsächlich hat Sie für ihre manipulierte kette schneller den nächsten Block gelöst. Nun hat Sie die längste Kette und weil sich Herr Lehmann und sein Netzwerk an die Regeln halten arbeiten Sie nun an Frau Beckers Kette weiter und Herr Lehmann’s kurzzeitig bestätigtes Geld ist wieder weg.

Je weniger Rechenkapazität in Bezug auf den Rest des Netzwerkes und je mehr bestätigungen man abwartet je unwahrscheinlicher wird eine erfolgreiche Attacke:



Bei über 50% Netzwerkpower wird man das Rennen immer gewinnen. Wir sehen das ein großes Netz sehr wichtig ist das es somit schwerer bzw. teurer wird genug Rechenleistung für eine erfolgreiche Attacke zu erlangen.

Problem: Wir haben keinen Anreiz Rechenleistung zur Verfügung zu stellen.

Lösung: Wer Rechenleistung zur verfügung stellt wird bezahlt.

## IncentiveCoin

Regeln:

* Für jeden gelösten Block gibt es einen BlockReward (bestimmte anzahl an coins). Hierdurch hat man einen Wirtschaftlichen Ansporn Rechenleistung zur verfügung zu stellen.
* Es ist die Einzige Art wie coins entstehen

Im Bitcoin Netzwerk kriegt man derzeit 6 Bitcoins als Reward. Da man alleine kaum ein chance hat das Puzzle zu lösen finden sich user in großen Mininggruppen zusammen (ähnlich wie bei Lottogruppen wird der Reward geteilt egal wer am ende „gewonnen“ hat).

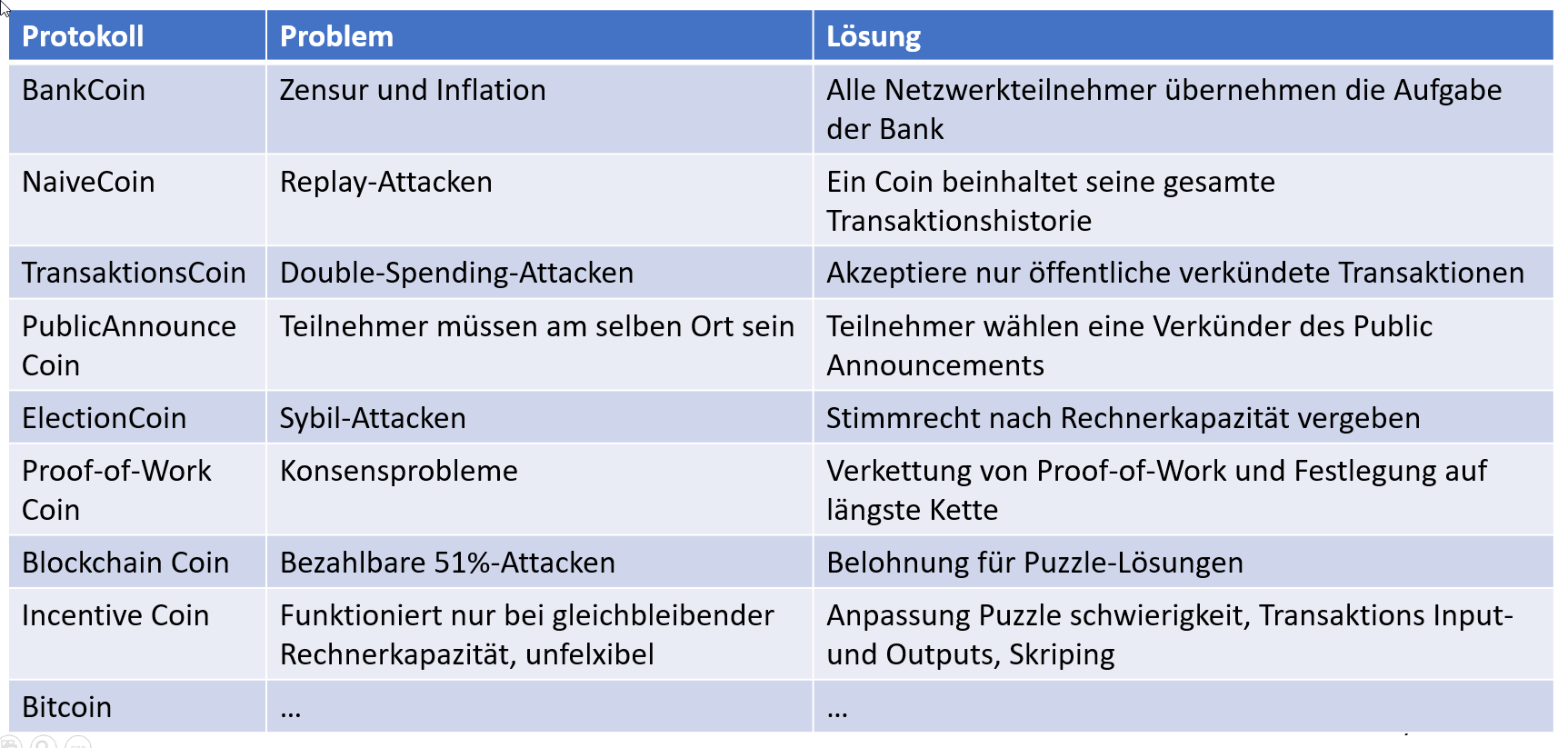
## Bitcoin

Es handelt sich bei den Beispielen natürlich um vereinfachte Approximationen eines Real existierenden Protokolls, nämlich dem Bitcoin.

Der Bitcoin hat noch 3 wichtige Technische Besonderheiten:

1. Man kann die Puzzlegeschwindigkeit anpassen. Sie wird versucht bei ca 10min zu halten.
2. Input und Output von Transaktionen, man kann auch nur teile eines Coins verschicken
3. Er besitzt eine eigene Scriptsprache

# Zusammenfassung



# Quellen

<https://de.wikipedia.org/wiki/Blockchain>

Blockchain kurz & gut von Kai Brünner

Telefonat mit Dr. Nils Werner vom Frauenhofer Institut Erlangen

Gespräch mit M. Sc. Fabian Schulze Leiter IT-Siemens Budapest

<https://de.wikipedia.org/wiki/Hashfunktion>

<https://emn178.github.io/online-tools/sha256.html>

<http://www.hashcash.org/hashcash.pdf>

<https://www.reddit.com/r/Bitcoin/comments/6xhjxf/total_hardware_cost_to_run_bitcoin_network_around/>