Blockchain Engineering

Konsens-Protokolle

Dr. Lars Brünjes

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Ostbayerische Technische Hochschule
Amberg-Weiden

OTH
REGENSBURG

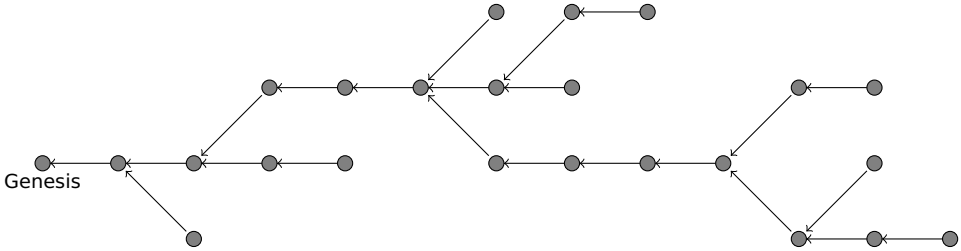
MIND
MODULARES INNOVATIVES
NETZWERK FÜR DURCHLÄSSIGKEIT

AUFSTIEG DURCH
BILDUNG >>
OFFENE HOCHSCHULEN

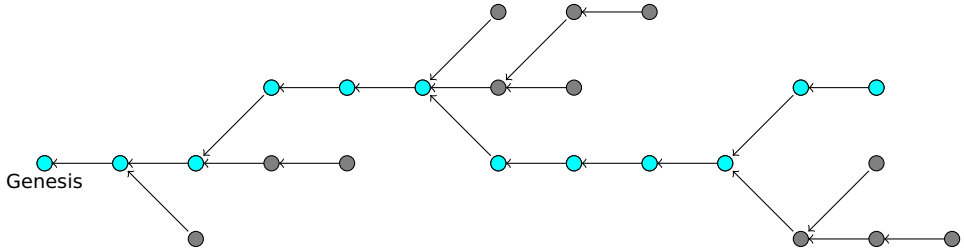
27. September 2019

- Die Konsistenz der individuellen Register-Einträge und Blöcke einer Blockchain wird mittels Kryptografie (insbesondere Hashing und digitale Unterschriften) gewährleistet.
- Für die Konsistenz der Blockchain als Ganzes ist es aber zusätzlich von entscheidender Bedeutung, dass alle Einträge linear geordnet sind.
- Die Einträge in jedem Block sind automatisch geordnet. Das Problem ist, die Blöcke in einer Kette anzuordnen, über die sich alle Knoten einig sind: Im Falle von Forks muss es einen Mechanismus geben, ein sogenanntes Konsens-Protokoll (Consensus Protocol), mit dem sich alle Knoten auf eine der Gabelungen einigen können.
- Forks können nie ganz vermieden werden. Ein Ziel des Konsens-Protokolls ist es, die Forks “zu zähmen”, d.h. zu verhindern, dass sie “zu groß” werden.

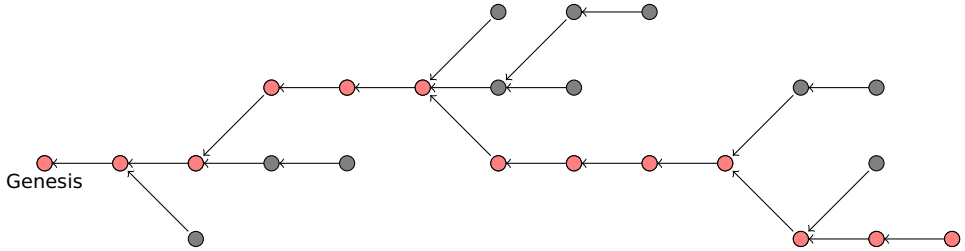
- Wichtiges Ziel eines Konsens-Protokolls ist es, die Common Prefix (CP) Eigenschaft (für einen Parameter k) zu garantieren: Nachdem ein Knoten die jüngsten k Blöcke entfernt hat, ist seine lokale Blockchain ein Präfix der lokalen Blockchains aller anderen Knoten.
- Man beachte, dass dies natürlich nur für Knoten gelten kann, deren Stand aktuell ist: War ein Knoten längere Zeit offline, ist sein Bild der Blockchain veraltet, und er wird für (CP) nicht betrachtet.
- Man beachte außerdem, dass Eigenschaften wie (CP) nie absolut, sondern immer in einem probabilistischen Sinne gelten. Im Falle von (CP) bedeutet dass, dass (CP) für genügend großes k mit überwältigender Wahrscheinlichkeit gilt.
- Bei Bitcoin z.B. wird normalerweise empfohlen, eine Transaktion nach einer Stunde als confirmed (bestätigt) anzusehen. In Bitcoin wird etwa alle zehn Minuten ein neuer Block erzeugt, d.h. für $k = 6$ gilt (CP) für Bitcoin mit hinreichend großer Wahrscheinlichkeit für die meisten Transaktionen.



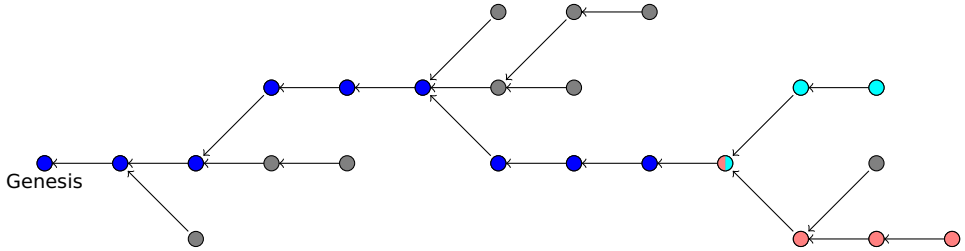
In dem Bild sind die grauen Blöcke alle Blöcke einer Beispielblockchain, die bisher erzeugt wurden. Beachten Sie, dass eventuell nicht alle Blöcke allen Knoten bekannt sind.



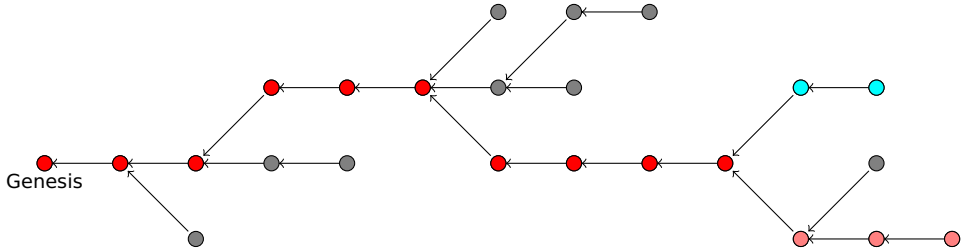
Ein Knoten hat die türkisen Blöcke als seine lokale Kopie der Blockchain.



Ein *anderer* Knoten hat die rosa Blöcke als *seine* lokale Kopie der Blockchain.



Wenn der *erste* Knoten die jüngsten drei Blöcke ignoriert, bleiben ihm die blauen Blöcke, und diese sind tatsächlich ein Präfix der rosa Kette.



Wenn der *zweite* Knoten die jüngsten drei Knoten ignoriert, bleiben ihm die roten Blöcke, und diese sind tatsächlich ein Präfix der türkisen Kette.

- (CP) ist nicht die einzige Eigenschaft, die ein Konsens-Protokoll sicherstellen muss.
- Wo (CP) Konsistenz der Blockchain garantiert, ist die Chain Growth (CG) (Ketten-Wachstum) Eigenschaft für gleichmäßiges Wachsen verantwortlich.
- (CG) besagt, dass die Kette (aus Sicht jedes online Knotens) mit hoher Wahrscheinlichkeit mit einer bestimmten Mindest-Rate wächst.
- Intuitiv bedeutet (CG), dass die Kette nicht “steckenbleiben” kann und dass regelmäßig neue Blöcke hinzugefügt werden.
- Insbesondere werden alle neuen Einträge (wie Transaktionen) mit sehr großer Wahrscheinlichkeit in voraussehbarer Zeit Teil eines Blocks der Kette sein.
- (CG) ist eine sogenannte Liveness (Lebendigkeits-)Eigenschaft, die dafür sorgt, dass das System “am Leben bleibt” und stetig Fortschritte macht.

- Die dritte Eigenschaft, die wir von einem Konsens-Protokoll erwaren, ist Chain Quality (CQ) (Ketten-Qualität).
- (CQ) besagt, dass die Anzahl der Blöcke, die eine Gruppe von Teilnehmern erzeugt, durch eine Funktion nach oben beschränkt ist, die von der relativen Stärke dieser Gruppe abhängt, wobei “Stärke” sich auf die Ressource bezieht, die Block-Erzeugung bestimmt.
- Konkret im Falle von Bitcoin bedeutet dies zum Beispiel, dass eine Gruppe mit bestimmten Anteil der Gesamt-Hashing-Power nur einen gewissen Anteil aller Blöcke erzeugen kann.
- Wenn es also eine Gruppe gibt, die sich nicht an das Protokoll hält und z.B. Transaktionen von bestimmten Benutzern nicht in ihre Blöcke einbaut, so garantiert (CQ), dass diese Gruppe nur einen beschränkten Anteil aller Blöcke “vergiften” kann.
- Wie (CG) ist (CQ) eine Liveness-Eigenschaft, die garantiert, dass alle (legitime) Transaktionen früher oder später in der Kette enden werden.

- Jeder Konsens-Algorithmus, den wir betrachten werden, funktioniert konzeptionell wie folgt:
- Zu jedem Zeitpunkt hat jeder Knoten “seine” Kette.
- Am Anfang ist die Kette jedes Knotens die triviale Kette, die nur aus dem Genesis-Block besteht.
- Jeder Knoten tauscht sich mit einer Anzahl anderer Knoten, seinen Peers aus.
- Wenn ein Knoten von einem Peer eine neue Kette erhält, prüft er diese zunächst auf Gültigkeit und vergleicht sie dann mit seiner eigenen Kette. Ist die Kette *länger* als seine eigene Kette, so macht er die neue Kette zu seiner eigenen Kette. Dies ist die sogenannte Chain Selection Regel.
- Jeder Knoten teilt jede neue (gültige) Kette mit seinen Peers.

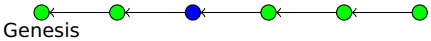
- Jeder Konsens-Algorithmus, den wir betrachten werden, funktioniert konzeptionell wie folgt:
- Zu jedem Zeitpunkt hat jeder Knoten “seine” Kette.
- Am Anfang ist die Kette jedes Knotens die triviale Kette, die nur aus dem Genesis-Block besteht.
- Jeder Knoten tauscht sich mit einer Anzahl anderer Knoten, seinen Peers aus.
- Wenn ein Knoten von einem Peer eine neue Kette erhält, prüft er diese zunächst auf Gültigkeit und vergleicht sie dann mit seiner eigenen Kette. Ist die Kette *länger* als seine eigene Kette, so macht er die neue Kette zu seiner eigenen Kette. Dies ist die sogenannte Chain Selection Regel.
- Jeder Knoten teilt jede neue (gültige) Kette mit seinen Peers.

Bemerkung

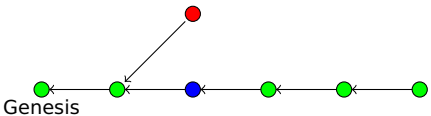
In der Praxis werden nicht ganze *Ketten*, sondern *Blöcke* ausgetauscht.

- Wir zuvor kurz bemerkt ist die Idee der meisten Konsens-Algorithmen, das Recht, einen neuen Block zu erstellen, an eine begrenzte “Ressource” zu binden.
- Bei Bitcoin, Ethereum und fast allen anderen Kryptowährungen ist diese Ressource Rechenleistung, sogenannte Hashing-Power.
- Bei Cardano ist die Ressource die Kryptowährung Ada selbst, sogenannter Stake.
- In beiden Fällen ist die Idee, dass Benutzer umso höhere Chancen haben, den nächster Block zu erstellen, je mehr von der Ressource sie kontrollieren.
- Blockchains sind sicher (erfüllen (CP) und (CG)), sofern die Mehrheit der Ressource in der Hand ehrlicher Benutzer liegt, d.h. in der Hand von Benutzern, die sich an das Protokoll halten.
- Wenn Knoten im Zuge der Chain Selection neue Ketten auf Gültigkeit prüfen, prüfen sie insbesondere auch, ob alle Blöcke von Benutzern erzeugt wurden, die gemäß dem jeweiligen Protokoll das Recht dazu hatten.

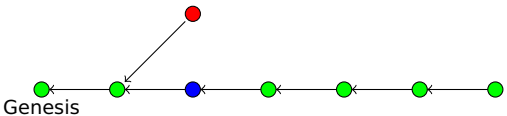
- *Satoshi Nakamoto*, der geheimnisumwogene anonyme Erfinder von Bitcoin, beschrieb als erster ein praktikables Konsens-Protokoll, das bis heute fast alle Blockchains benutzen — Proof of Work (PoW).
- Die begrenzte “Ressource” bei PoW is Rechenleistung (Hashing-Power).
- Die Grundidee ist, jedem Block zusätzlich zum Hash des Vorgängers und der Liste der Transaktionen eine Zahl, die sogenannte Nonce, hinzuzufügen, so dass der Hash des Blocks einschließlich dieser Nonce mit einer vorgeschriebenen Zahl von n 0-Bits beginnt.
- Wie wir in der Vorlesung über Hashing gesehen haben, ist eine kryptografische Hashfunktion Puzzle-freundlich, was bedeutet, dass eine geeignete Nonce nur durch Ausprobieren von durchschnittlich 2^n Zahlen gefunden werden kann.
- Für Bitcoin wird n so gewählt, dass es durchschnittlich 10 Minuten dauert, bis irgendwo auf der Welt jemand eine passende Nonce findet.



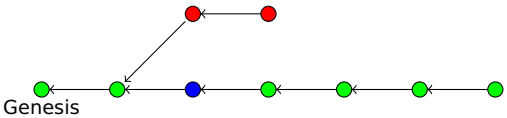
Nehmen wir an, Alice möchte einen Fork erzeugen, der den blauen Block aus der Blockchain wirft.



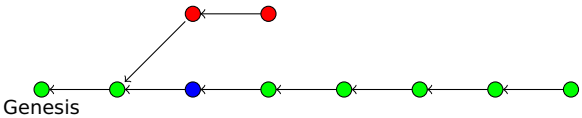
Sie beginnt, an einer alternativen roten Kette zu arbeiten.



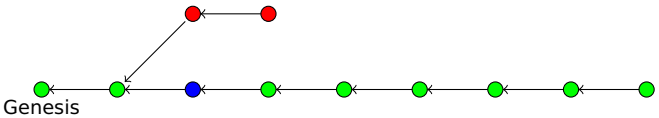
Aufgrund der Chain Selection Regel arbeiten alle ehrlichen Parteien weiter an der längeren grünen Kette.



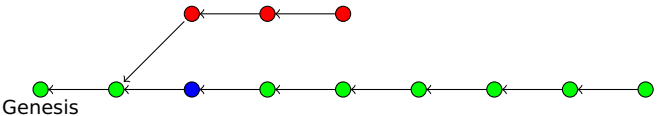
Mit etwas Glück gelingt es Alice, weiter an ihrer roten Kette zu arbeiten.



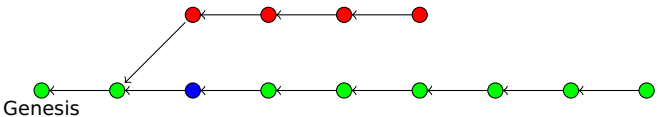
Aber wenn die ehrlichen Parteien die Mehrheit der hashing power besitzen, werden sie auf lange Sicht schneller sein.



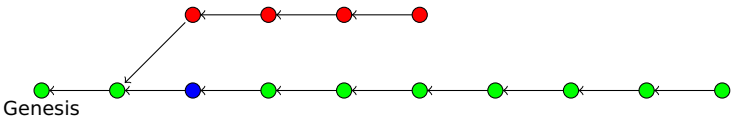
Aber wenn die ehrlichen Parteien die Mehrheit der hashing power besitzen, werden sie auf lange Sicht schneller sein.



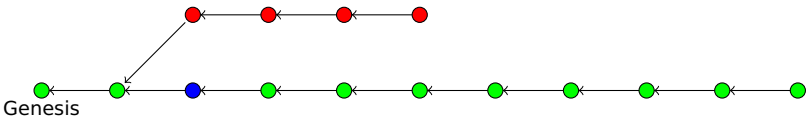
Alice kann durch pures Glück eventuell zeitweilig aufholen.



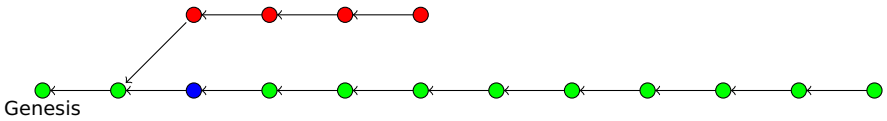
Alice kann durch pures Glück eventuell zeitweilig aufholen.



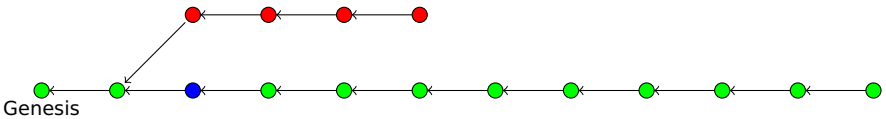
Aber auf lange Sicht wird sie weiter und weiter hinter der “ehrlichen” grünen Kette zurückbleiben.



Aber auf lange Sicht wird sie weiter und weiter hinter der “ehrlichen” grünen Kette zurückbleiben.



Aber auf lange Sicht wird sie weiter und weiter hinter der “ehrlichen” grünen Kette zurückbleiben.



Je tiefer der Fork, den Alice erzeugen will, desto unwahrscheinlicher wird es, dass sie Erfolg haben wird. Bei Bitcoin gelten schon Forks der Tiefe *sechs* als äußerst unwahrscheinlich.

Und was ist mit Liveness?

- Die Überlegungen auf der letzten Folie machen es hoffentlich plausibel, dass PoW (CP) sicherstellt.
- Dies kann auch — unter geeigneten Modellannahmen an das Netz und die Kommunikation der Knoten — mathematisch bewiesen werden.

Und was ist mit Liveness?

- Die Überlegungen auf der letzten Folie machen es hoffentlich plausibel, dass PoW (CP) sicherstellt.
- Dies kann auch — unter geeigneten Modellannahmen an das Netz und die Kommunikation der Knoten — mathematisch bewiesen werden.
- Was ist mir der zweiten wichtigen Eigenschaft, der Liveness-Eigenschaft (CG)?

Und was ist mit Liveness?

- Die Überlegungen auf der letzten Folie machen es hoffentlich plausibel, dass PoW (CP) sicherstellt.
- Dies kann auch — unter geeigneten Modellannahmen an das Netz und die Kommunikation der Knoten — mathematisch bewiesen werden.
- Was ist mir der zweiten wichtigen Eigenschaft, der Liveness-Eigenschaft (CG)?
- Grob gesagt wird die ehrliche Mehrheit der Hashing-Power aufgrund der Chain Selection Regel an der längsten Kette arbeiten und demnach im Durchschnitt schneller als alle 20 Minuten einen neuen Block an diese Kette anhängen.
- In der Praxis wächst die Bitcoin-Kette um etwa einen Block alle zehn Minuten.

Und was ist mit Liveness?

- Die Überlegungen auf der letzten Folie machen es hoffentlich plausibel, dass PoW (CP) sicherstellt.
- Dies kann auch — unter geeigneten Modellannahmen an das Netz und die Kommunikation der Knoten — mathematisch bewiesen werden.
- Was ist mir der zweiten wichtigen Eigenschaft, der Liveness-Eigenschaft (CG)?
- Grob gesagt wird die ehrliche Mehrheit der Hashing-Power aufgrund der Chain Selection Regel an der längsten Kette arbeiten und demnach im Durchschnitt schneller als alle 20 Minuten einen neuen Block an diese Kette anhängen.
- In der Praxis wächst die Bitcoin-Kette um etwa einen Block alle zehn Minuten.
- Es ist intuitiv klar, dass auch (CQ) erfüllt ist, da die Anzahl der Blöcke, die eine Gruppe erzeugt, proportional zur Hashing-Power der Gruppe ist.

- Vorteile
  - PoW ist (relativ) einfach zu implementieren, und die Sicherheit des Protokolls ist plausibel und (relativ) leicht zu beweisen.
  - Betrugsversuche sind teuer: Jede falsche alternative Kette erfordert immense Rechenzeit.
  - PoW hat sich in der Praxis bewährt und scheint robust zu funktionieren.

- Vorteile
  - PoW ist (relativ) einfach zu implementieren, und die Sicherheit des Protokolls ist plausibel und (relativ) leicht zu beweisen.
  - Betrugsversuche sind teuer: Jede falsche alternative Kette erfordert immense Rechenzeit.
  - PoW hat sich in der Praxis bewährt und scheint robust zu funktionieren.
- Nachteile
  - PoW ist teuer und verschlingt enorme Rechenleistung und Energie: Einem Artikel in der Zeitschrift Joule zufolge verbraucht das Bitcoin-Netz im Jahr 45.8 TWh Elektrizität und verursacht einen CO_2-Ausstoß von mehr als 22 Millionen Tonnen, was zwischen dem Verbrauch von Jordanien und Sri Lanka liegt.
  - PoW ist ineffizient: Die maximale Anzahl an Transaktionen wird durch die erzwungene Verzögerung von zehn Minuten pro Block drastisch eingeschränkt.
  - PoW hat zu einem “Wettrüsten” geführt, bei dem nur die Besitzer der neuesten spezialisierten Hardware eine realistische Chance haben, einen Block zu finden, was die Zentralisierung der Hashing-Power in riesigen Mining Pools begünstigt hat.

- Proof of Useful Work ist die Idee, sinnvolle Arbeit anstelle von nutzlosen Hashing-Puzzles in die Erzeugung von Blöcken zu stecken.
  - In der Tat gibt es schon Bitcoin-Miner, die die Abwärme ihrer Hardware zum Heizen zur Verfügung stellen...
  - Normalerweise geht es bei *Proof of Useful Work* aber eher um Dinge wie Simulation von Proteinfaltungen für die Pharmaforschung, der Suche nach großen Primzahlen und Ähnlichem.
  - Ein Problem dabei ist, sicher zu stellen, dass der Schwierigkeitsgrad konfigurierbar und berechenbar bleibt.
- Memory-Bound PoW
  - Hashing-Puzzles sind CPU-Bound, d.h. die Hashing-Power hängt von der Geschwindigkeit der CPU ab, was Besitzern von spezialisierter und teurer Hardware einen großen Vorteil gibt.
  - Alternativ gibt es Memory-Bound Probleme, die zu ihrer Lösung viele Zugriffe auf den Hauptspeicher erfordern, was unabhängiger von der Qualität der Hardware ist und Dezentralisierung fördern könnte.

- Proof of Nonoutsourcable Work ist die Idee, eine Aufgabe zu stellen, die schwer an viele Parteien zu delegieren ist, was die Bildung von mächtigen Mining Pools wie bei Bitcoin verhindern würde.
  - Eine Vorschlag, dies zu implementieren, ist, *zwei* Dinge für die Gültigkeit eines Blocks zu fordern:
  - Erstens muss wie gehabt eine Nonce gefunden werden, so dass der Hash des Blocks inklusive Nonce mit hinreichend vielen Nullen beginnt.
  - Darüber hinaus muss der Block digital unterschrieben werden, und der Hash der Unterschrift muss *auch* mit hinreichend vielen Nullen beginnen.
  - Dies bedeutet, ein Mining Pool müsste seinen geheimen Schlüssel teilen, wenn er den zweiten Teil outsourcen wollte.

- Die Idee von Proof of Stake (PoS) ist es, Besitz der Kryptowährung selbst als Ressource zu verwenden, die das Erstellen neuer Blöcke einschränkt.
- Wo PoW sicher ist, falls die Mehrheit der *Rechenleistung* im Besitz ehrlicher Parteien ist, ist Proof of Stake sicher, wenn die Mehrheit der Kryptowährung in ehrlichen Händen ist.
- Diese Annahme ist plausibel, weil jemand, der viel in eine Kryptowährung investiert hat, ein entsprechend großes Interesse daran hat, dass diese Währung Erfolg hat und nicht durch Zusammenbruch des Konsens-Protokolls scheitert, denn dann würde seine Investition wertlos werden.

- Als Beispiel eines praktischen PoS-Algorithmus schauen wir uns Ouroboros an, den PoS-Algorithmus von Cardano.
- Grundidee von Ouroboros ist es, dass Zeit in Runden (Slots) aufgeteilt wird und dass in jeder Runde eine Lotterie abgehalten wird, deren Gewinner, der sogenannte Slot Leader, den nächsten Block erzeugen darf.
- Diese Lotterie ist so konstruiert, dass die Gewinnchancen proportional zum Stake, d.h. zu der Summe an Ada ist, die man besitzt.

Abb.: Die mythische altägyptische Schlange Ouroboros, die ihren eigenen Schwanz verschlingt.

- Eine solche Lotterie muss fair sein, damit die Gewinnchance tatsächlich proportional zum Stake ist.
- Sie muss unvorhersehbar sein. Wenn es möglich wäre, die Gewinner vorab zu ermitteln, könnte jemand versuchen, gezielt zukünftige Slot Leader zu bestechen oder zu erpressen oder sich selbst durch geeignete Aktionen (Transaktionen auf der Blockchain) zum Slot Leader zu machen.
- Sie muss nachprüfbar sein, denn Knoten müssen überprüfen können, ob ein Slot Leader tatsächlich gewonnen hat.

- Eine solche Lotterie muss fair sein, damit die Gewinnchance tatsächlich proportional zum Stake ist.
- Sie muss unvorhersehbar sein. Wenn es möglich wäre, die Gewinner vorab zu ermitteln, könnte jemand versuchen, gezielt zukünftige Slot Leader zu bestechen oder zu erpressen oder sich selbst durch geeignete Aktionen (Transaktionen auf der Blockchain) zum Slot Leader zu machen.
- Sie muss nachprüfbar sein, denn Knoten müssen überprüfen können, ob ein Slot Leader tatsächlich gewonnen hat.

Follow the Satoshi

Um die erste Eigenschaft zu gewährleisten, kann man zufällig eine einzelne *Münze* (Satoshi bei Bitcoin, Lovelace bei Ada) ziehen und ihren Besitzer zum Slot Leader erklären.

- Eine solche Lotterie muss fair sein, damit die Gewinnchance tatsächlich proportional zum Stake ist.
- Sie muss unvorhersehbar sein. Wenn es möglich wäre, die Gewinner vorab zu ermitteln, könnte jemand versuchen, gezielt zukünftige Slot Leader zu bestechen oder zu erpressen oder sich selbst durch geeignete Aktionen (Transaktionen auf der Blockchain) zum Slot Leader zu machen.
- Sie muss nachprüfbar sein, denn Knoten müssen überprüfen können, ob ein Slot Leader tatsächlich gewonnen hat.

Entropie

Insbesondere für die zweite Eigenschaft ist es nötig, hinreichend viel Entropie (Zufälligkeit) deterministisch zu erzeugen.

- Angenommen, Alice will mit Bob ein simples Glücksspiel über das Internet spielen: Ein zufälliges Bit wird “gewürfelt”, ist es 0, so gewinnt Alice, ist es 1, so gewinnt Bob.
- Alice und Bob wollen sich nicht auf einen externen “Würfel” verlassen, sondern die Entropie selbst erzeugen. — Wie können Sie das tun?

- Angenommen, Alice will mit Bob ein simples Glücksspiel über das Internet spielen: Ein zufälliges Bit wird “gewürfelt”, ist es 0, so gewinnt Alice, ist es 1, so gewinnt Bob.
- Alice und Bob wollen sich nicht auf einen externen “Würfel” verlassen, sondern die Entropie selbst erzeugen. — Wie können Sie das tun?
- Die Grundidee ist, dass Alice und Bob beide für sich ein zufälliges Bit a und b ermitteln, dieses dann teilen und als Ergebnis $a \oplus b$ (xor) verwenden. Also z.B. $a = 1$, $b = 1$, $a \oplus b = 0$, Alice gewinnt,

- Angenommen, Alice will mit Bob ein simples Glücksspiel über das Internet spielen: Ein zufälliges Bit wird “gewürfelt”, ist es 0, so gewinnt Alice, ist es 1, so gewinnt Bob.
- Alice und Bob wollen sich nicht auf einen externen “Würfel” verlassen, sondern die Entropie selbst erzeugen. — Wie können Sie das tun?
- Die Grundidee ist, dass Alice und Bob beide für sich ein zufälliges Bit a und b ermitteln, dieses dann teilen und als Ergebnis $a \oplus b$ (xor) verwenden. Also z.B. $a = 1$, $b = 1$, $a \oplus b = 0$, Alice gewinnt,
- Das Problem ist: Wie tauschen Alice und Bob ihre Bits a und b aus? Wenn Alice Bob 1 schickt, kann er einfach behaupten, b sei 0 und gewinnen.

- Angenommen, Alice will mit Bob ein simples Glücksspiel über das Internet spielen: Ein zufälliges Bit wird “gewürfelt”, ist es 0, so gewinnt Alice, ist es 1, so gewinnt Bob.
- Alice und Bob wollen sich nicht auf einen externen “Würfel” verlassen, sondern die Entropie selbst erzeugen. — Wie können Sie das tun?
- Die Grundidee ist, dass Alice und Bob beide für sich ein zufälliges Bit a und b ermitteln, dieses dann teilen und als Ergebnis $a \oplus b$ (xor) verwenden. Also z.B. $a = 1$, $b = 1$, $a \oplus b = 0$, Alice gewinnt,
- Das Problem ist: Wie tauschen Alice und Bob ihre Bits a und b aus? Wenn Alice Bob 1 schickt, kann er einfach behaupten, b sei 0 und gewinnen.
- Eine Lösung haben wir in der Vorlesung über Hashing (Stichwort: “Versteckende Hash-Funktionen — Schere, Stein, Papier”) kennengelernt: Alice und Bob benutzen ein Commitment Scheme und tauschen zunächst die Hashs $H(r_a \mid a)$ und $H(r_b \mid b)$ aus, womit sie sich auf a und b festlegen.

- Es gibt jedoch ein weiteres Problem mit Alices und Bobs Spiel: Selbst wenn Bob b nicht nachträglich ändern kann, nachdem er a gesehen hat, weil er sich mit $H(r_b \mid b)$ schon festgelegt hat, könnte er das Spiel abbrechen, wenn er sieht, dass $a \oplus b = 0$, ohne Alice b zu schicken.
- Im Kontext einer Kryptowährung bedeutet das, Parteien könnten sich weigern, ihren Beitrag zur Entropie zu teilen, sobald sie sehen, dass das Wahlergebnis für sie ungünstig wäre.

- Es gibt jedoch ein weiteres Problem mit Alices und Bobs Spiel: Selbst wenn Bob b nicht nachträglich ändern kann, nachdem er a gesehen hat, weil er sich mit $H(r_b \mid b)$ schon festgelegt hat, könnte er das Spiel abbrechen, wenn er sieht, dass $a \oplus b = 0$, ohne Alice b zu schicken.
- Im Kontext einer Kryptowährung bedeutet das, Parteien könnten sich weigern, ihren Beitrag zur Entropie zu teilen, sobald sie sehen, dass das Wahlergebnis für sie ungünstig wäre.
- Zum Glück gibt es Verfahren zum verifizierten Tauschen von Geheimnissen (Verifiable Secret Sharing (VSS)),
- VSS ist ein Commitment Scheme, dass es zusätzlich verbleibenden Parteien ermöglicht, das “Geheimnis” einer Partei zu rekonstruieren, die das “Spiel” verlässt.

- Es gibt jedoch ein weiteres Problem mit Alices und Bobs Spiel: Selbst wenn Bob b nicht nachträglich ändern kann, nachdem er a gesehen hat, weil er sich mit $H(r_b \mid b)$ schon festgelegt hat, könnte er das Spiel abbrechen, wenn er sieht, dass $a \oplus b = 0$, ohne Alice b zu schicken.
- Im Kontext einer Kryptowährung bedeutet das, Parteien könnten sich weigern, ihren Beitrag zur Entropie zu teilen, sobald sie sehen, dass das Wahlergebnis für sie ungünstig wäre.
- Zum Glück gibt es Verfahren zum verifizierten Tauschen von Geheimnissen (Verifiable Secret Sharing (VSS)),
- VSS ist ein Commitment Scheme, dass es zusätzlich verbleibenden Parteien ermöglicht, das “Geheimnis” einer Partei zu rekonstruieren, die das “Spiel” verlässt.
- Ouroboros Classic, der erste PoS-Algorithmus von Cardano, hat ein VSS-Verfahren benutzt, um Entropie auszutauschen und dann mittels *Follow the Satoshi* den Slot Leader zu wählen.
- Leider erfordert VSS viel Kommunikation zwischen den Parteien (quadratisch in der Zahl der Beteiligten) und skaliert daher nicht gut. Es ist außerdem ziemlich kompliziert.

- Ouroboros Praos, der neue Konsens-Algorithmus von Cardano, geht einen anderen, konzeptionell und praktisch einfacheren Weg.
- Anstatt Entropie auszutauschen, was viel Kommunikation erfordert, halten alle Parteien private Lotterien ab, d.h. jeder “würfelt seinen eigenen Würfel” und sieht für sich, ob er Slot Leader ist.
- Andere Parteien können später prüfen, ob jemand, der behauptet, seine private Lotterie gewonnen zu haben, recht hat.
- Es ist möglich, dass mehr als eine Partei ihre privaten Lotterie in derselben Runde gewinnt, was aber selten ist und kein Problem darstellt, da der resultierende Fork klein ist.
- Einer der Parameter von Ouroboros Praos ist der active slots coefficient $f \in (0, 1]$, der bestimmt, welcher Anteil von Slots einen Slot Leader haben wird — in der Praxis wird f relativ klein gewählt (d.h. die meisten Slots haben *keinen* Leader), was Kollisionen (Slots mit mehreren Leadern) selten macht.

- Zur Implementierung der privaten Lotterien in Ouroboros Praos ist das kryptografische Konzept einer verifizierbaren, zufälligen Funktion (Verifiable Random Function (VRF)) nötig.
- Eine VRF ist ein Paar von Funktionen F und p, die einen Bitstring fester Länge x und einen geheimen Schlüssel sk auf Bitstrings fester Länge $F(x, sk)$ und $p(x, sk)$ abbilden, wobei f und p die folgenden Eigenschaften haben:
  - Funktion F ist pseudo-zufällig (pseudo-random), d.h. praktisch ununterscheidbar von einer zufälligen Funktion.
  - Es gibt eine dritte Funktion v, die F verifiziert: $v(y, q, pk)$ ist 1, falls $y = F(x, sk)$ und $q = p(x, sk)$ und sonst 0. Konkret bedeutet dies, dass andere Parteien das Ergebnis von F mit Hilfe des "Beweises" q prüfen können, ohne sk zu kennen.
  - Ohne Kenntnis des geheimen Schlüssels sk ist es praktisch unmöglich, $F(x, sk)$ zu erraten, wenn man nur x kennt.

- Woher stammt die Entropie in Ouroboros Praos?
- In Praos (und anderen Ouroboros Protokollen) werden mehrere Slots in sogenannte Epochen (Epochs) aufgeteilt. In jeder Epoche ep wird eine neue Entropie η_{ep} ermittelt, und in Slot sl werden $x_{\mathsf{NONCE}} = H(\eta_{ep} \mid sl \mid \text{NONCE})$ und $x_{\mathsf{TEST}} = H(\eta_{ep} \mid sl \mid \text{TEST})$ als Inputs für die VRF F benutzt.
- $F(x_{\mathsf{TEST}}, sk)$ bestimmt den Ausgang der Lotterie für diesen Slot, und die $F(x_{\mathsf{NONCE}}, sk)$ werden am Ende der Epoche kombiniert, um η_{ep+1} zu bestimmen.
- Es ist wichtig, dass alle Parteien denselben Wert für η_{ep} benutzen, was dadurch sichergestellt wird, dass die $F(x_{\mathsf{NONCE}}, sk)$, die in die Entropie einfließen, so weit in der Vergangenheit liegen, dass sie wegen (CP) stabil sind.

- Vorteile
  - PoS ist wesentlich schneller und effizienter als PoW und verbraucht nur einen Bruchteil der Energie.
  - Anstelle des etwas willkürlichen Konzepts der “Hashing-Power” ist die Sicherheit des Protokolls an Stake im System selbst geknüpft, was konzeptionell sauberer erscheint.

- Vorteile
  - PoS ist wesentlich schneller und effizienter als PoW und verbraucht nur einen Bruchteil der Energie.
  - Anstelle des etwas willkürlichen Konzepts der “Hashing-Power” ist die Sicherheit des Protokolls an Stake im System selbst geknüpft, was konzeptionell sauberer erscheint.
- Nachteile
  - PoS ist komplizierter und weniger intuitiv als PoW.
  - PoW hat sich über Jahre hinweg bewährt, wohingegen PoS sich erst in der Praxis bewähren muss.
  - In PoW ist dauert es genauso lange, eine falsche, alternative Kette zu erzeugen wie an der echten zu arbeiten. Im Gegensatz dazu geht es schnell, eine alternative Kette in PoS zu erzeugen. Um trotzdem sicher zu sein, müssen PoS-Protokolle besondere Rücksicht auf diese Tatsache nehmen.

- Ein mögliches Problem bei PoS-Systemen ist, dass ehrliche Parteien mit viel Stake in der Zukunft ihren Stake verlieren und dann korrumpierbar werden könnten.
- Nehmen wir an, dass Alice viel Stake und damit ein großes Interesse an der Stabilität der Kryptowährung hat. Wegen ihres hohen Stakes darf sie viele Blöcke erzeugen.
- Nehmen wir weiter an das Alice später ihren Stake verliert oder verkauft und sich dann nicht mehr um die Geschicke der Währung kümmert.
- Bob könnte Alice dann bestechen, ihm ihren geheimen Schlüssel zu geben. Mit diesem Schlüssel könnte er einen tiefen Fork in der Zeit beginnen, in dem Alice noch hohen Stake hatte, und ihren Schlüssel benutzen, um in ihrem Namen Blöcke für diesen Fork zu erzeugen.
- Um solche Angriffe zu verhindern, benutzt Ouroboros Praos Key Evolving Signatures (KES) (Unterschriften mit evolvierenden Schlüsseln). Im Prinzip bedeutet dies, dass Alice nach jeder Unterschrift ihren geheimen Schlüssel vernichtet und einen neuen bekommt. Sie kann Bob dann höchstens ihren aktuellen Schlüssel verraten, nicht aber ihre alten, so dass Bob die Vergangenheit nicht manipulieren kann.

- Konsens-Protokoll Ouroboros-Genesis ist eine Variante von Ouroboros-Praos, in der nur die Chain Selection Regel leicht verändert ist.
- In der Praxis werden Knoten nicht ständig online sein — neue kommen später, vielleicht Jahre später, hinzu, andere gehen zeitweilig offline.
- In Bitcoin und anderen PoW-Systemen ist dies kein Problem; ausgehend vom Genesis-Block oder einem beliebigen anderen Block der Kette kann ein Knoten, der lange offline war, “aufholen”.
- In Ouroboros Praos könnte ein Betrüger einem neuen Knoten eine lange falsche Kette anbieten, und der Knoten könnte nicht zwischen ihr und der echten Kette unterscheiden.
- Um dieses Problem zu lösen, weicht Ouroboros Genesis in der Chain Selection dahingehend ab, dass eine angebotene alternative längere Kette, sofern sie weit in der Vergangenheit forkt, nur dann übernommen wird, wenn sie dichter (mehr Blocks pro Slot) ist als die aktuelle.
- Der Punkt ist, dass Fälscher zwar lange, aber keine dichten Ketten erzeugen können, wenn sie nicht die Mehrheit des Stakes besitzen.

- Ouroboros BFT (Byzantine Fault Tolerance) ist ein minimales Konsens-Protokoll, dass durch seine Einfachheit besticht.
- BFT funktioniert nur für eine feste Anzahl n von Knoten.
- BFT erfüllt (CP), (CG) und (CQ), falls mindestens $\frac{2}{3}n$ Knoten ehrlich sind.
- Falls unehrliche Knoten (sogenannte byzantinische Knoten) nichts tun dürfen, was öffentlich gegen das Protokoll verstößt (also nicht auffallen dürfen), so genügen $\frac{n}{2}$ ehrliche Knoten. Dies ist das sogenannte Covert Byzantine Setting.

- Die n Knoten werden fortlaufend von 0 bis $(n-1)$ numeriert. Ihre öffentlichen Schlüssel sind fest und öffentlich bekannt.
- Zeit wird in Slots von fester Länge aufgeteilt, die Slots werden beginnend bei 1 numeriert.
- Neu erzeugte Blöcke werden mit der aktuellen Slot-Nummer als Zeitstempel versehen.
- In Slot i hat ausschließlich Knoten k mit $k \equiv i \pmod{n}$ das Recht, den nächsten Block zu erzeugen.
- Ein Block ist gültig, wenn
  - sein Zeitstempel nicht in der Zukunft liegt und
  - er die Unterschrift des zum Slot passenden Knotens trägt.

Knoten	Slots
0	7, 14, 21, 28,. . .
1	1, 8, 15, 22,. . .
2	2, 9, 16, 23,. . .
3	3, 10, 17, 24,. . .
4	4, 11, 18, 25,. . .
5	5, 12, 19, 26,. . .
6	6, 13, 20, 27,. . .

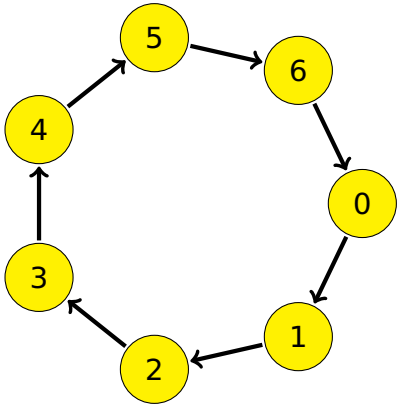


Abb.: Ouroboros BFT mit sieben Knoten

- Normale byzantine Knoten können tun, was immer sie wollen. Sie können Nachrichten ignorieren oder irreführende Nachrichten schicken, wann immer und so oft sie wollen.
- Im Covert Byzantine Setting dürfen sie das Protokoll zwar verletzen (indem sie z.B. schweigen und Netzprobleme vortäuschen oder Nachrichten von anderen Knoten ignorieren), aber Sie dürfen nicht in einer Weise gegen das Protokoll verstoßen, die nach außen hin offensichtlich ist.
- Insbesondere dürfen sie nicht mehr als einen Block mit demselben Zeitstempel erzeugen und im Netz verbreiten.
- Dies ist eine offensichtliche Einschränkung, die es hoffentlich plausibel macht, warum in diesem Setting weniger ehrliche Knoten (bloße 50%) genügen.
- In der Praxis kann dieses Setting zum Beispiel dadurch erzwungen werden, dass alle Knoten am Anfang eine Kaution hinterlegen müssen, die verfällt, wenn zwei Blöcke mit demselben Zeitstempel auftauchen, die von dem Knoten unterschrieben wurden.

Hinweis

Diese Publikation wurde im Rahmen des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Bund- Länder- Wettbewerbs “Aufstieg durch Bildung: offene Hochschulen” erstellt. Die in dieser Publikation dargelegten Ergebnisse und Interpretationen liegen in der alleinigen Verantwortung der Autor/innen.