A=SIT Technology
E-Government Innovationszentrum

1. Oktober 2018 Studie · Blockchain · Privatsphäre & Krypto · 17 Min. Lesezeit

# Redactable Blockchain

**Archiviert.** Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

**Stand** 2018-09-07
**Quelle** https://technology2.a-sit.at/veroeffentlichungen/redactable-blockchain/

In diesem Projekt wurden zuerst die verschiedenen Möglichkeiten, sowie deren Effizienz, Daten in der Bitcoin-Blockchain zu hinterlegen zusammengefasst. Anschließend werden verschiedene Ansätze vorgestellt, welche Schwärzungen von Daten in einer Blockchain ermöglichen, ohne die Integrität der Blockchain zu zerstören. Im Speziellen werden die Ansätze von Ateniese u. a. und Puddu u. a. zur Schwärzung von Daten vorgestellt. Der erste Ansatz beruht auf der Verwendung von Chameleon-Hashfunktionen, der zweite Ansatz auf der Verwendung von Verschlüsselung. Als dritter Ansatz wird eine Idee vorgestellt, die die Schwärzung der Daten über eine weitere, verlinkte Blockchain legitimiert.

## 1. Einleitung

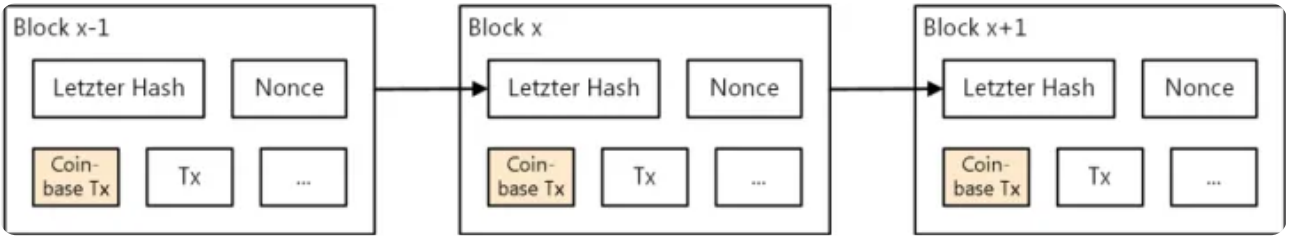
Bei einer Blockchain handelt es sich um eine kryptografisch verkettete Liste von Datensätzen mit eindeutiger Reihenfolge. Diese Datensätze werden in Blöcken gespeichert. Neben den Nutzdaten enthält jeder Block auch eine Referenz zum vorherigen Block sowie andere nützliche Informationen wie einen Zeitstempel oder Daten, die die Gültigkeit des Blocks bestätigen. Die Blockchain wurde 2008 von Satoshi Nakamoto als verteilte Datenbank im White Paper zur Kryptowährung Bitcoin beschrieben [1]. Bei Bitcoin wird die Blockchain zur dezentralen Speicherung aller Transaktionen verwendet. Dadurch kann beispielsweise verhindert werden, dass die selben Währungseinheiten mehrfach ausgegeben werden ( *Double Spending* ). Bei Bitcoin muss ein Rätsel gelöst werden, um einen neuen Block erstellen zu können. Der Schwierigkeitsgrad des Rätsels wird regelmäßig angepasst, mit dem Ziel eines mittleren Blockerstellungsintervalls von 10 Minuten. Die Lösung des Rätsels dient als Proof-of-Work. Der Proof-of-Work dient einerseits als Schutz gegen Veränderungen und Manipulationen der Blockchain und wird andererseits benötigt, damit mehrere verschiedene Parteien, die sich nicht kennen oder vertrauen müssen, ein gemeinsames Ziel erreichen können. Bei Proof-of-Work stellt jeder Teilnehmer (sog. „Miner“) seiner Rechenleistung zur Verfügung. Als Belohnung bekommt ein erfolgreicher Miner, der sich an alle Regeln hält, neue Wahrungseinheiten ausgeschüttet. Es gibt jedoch inzwischen auch andere Konsensus-Algorithmen die das gleiche Ziel verfolgen, aber deutlich weniger Rechenleistung benötigen. Ein Beispiel hierfür ist Proof-of-Stake, wo jeder Teilnehmer einen Teil des Kapitals als „Pfand“ hinterlegt. Abbildung 1 zeigt eine simple Blockchain.

*Abbildung 1: Blockchain: Eine Kette bestehend aus mehreren Blöcken (von links nach rechts gezeichnet).*
Durch die Verkettung erhält man neben einer eindeutigen Reihenfolge der Blöcke auch noch weitere Eigenschaften wie einen gewissen Schutz vor Manipulationen. Um eine Transaktion oder einen Block zu entfernen, müsste ein Angreifer bzw. eine Angreiferin auch alle danach folgenden Blöcke neuberechnen, da die ehrlichen Teilnehmer im Netzwerk immer diejenige Kette, die den größten akkumulierten Proof-of-Work enthält, als gültig werten und diese Kette erweitern versuchen. Die Unveränderbarkeit bringt neben einigen Vorteilen für gewisse Anwendungen auch ein paar Nachteile mit sich. Beispielsweise können fehlerhafte Daten nicht entfernt werden bzw. kann das „Recht auf Vergessenwerden" nicht ohne weiteres umgesetzt werden. Für mehr Details zu Bitcoin und Blockchains wird auf [1] und [2] verwiesen. Im nächsten Abschnitt werden die verschiedenen Möglichkeiten zur Datenablage in Blockchains anhand von Bitcoin erklärt.

## 2. Blockchain als Datenspeicher

Typischerweise enthalten Blöcke in der Blockchain eine variable Anzahl von Transaktionen. Diese Transaktionen können unter anderem benutzt werden, um Daten abzulegen oder Währungseinheiten zu verschieben. Ein Beispiel für eine auf Datenablage spezialisierte Blockchain ist die Multichain [1] Plattform. Im Folgenden werden Möglichkeiten zu Datenablage in öffentlichen Blockchains, am Beispiel von Bitcoin, diskutiert [1]. Da Bitcoin primär als Zahlungssystem entworfen wurde, lag der Fokus nicht auf Datenablage. Dennoch bietet Bitcoin verschiedene Möglichkeiten zur Datenablage, durch Encodierung der Daten in die verschiedenen Felder. Die folgenden Abschnitte fassen die Ergebnisse von Sward et al. [2] zusammen.

1 https://www.multichain.com/




### 2.1. Pay-to-Public-Key-Hash

Bei dieser Methode der Datenablage, wird der Public-Key-Hash [2] des Empfängers durch die zu speichernden Daten ersetzt. Pro Tranaktions-Output können 20 Byte an Daten abgelegt werden. Der Sender schickt die minimal notwendige Menge („non-dust") an Bitcoins, damit die Transaktion aufgenommen wird. Die überwiesenen Währungseinheiten gehen dabei „verloren", da höchstwahrscheinlich niemand über den zu dem Hash gehörenden Private Key verfügt. Für Miner ist nicht überprüfbar ob ein Public-Key-Hash zu einem gültigen Schlüssel gehört. Diese Methode der Datenspeicherung ist umstritten, da sich die Menge der Unspent Transaction Outputs (UXTO) vergrößert. Dadurch vergrößert sich auch der Speicheraufwand für alle Miner.

### 2.2. Pay-to-Public-Key

Die Pay-to-Public-Key Methode ähnelt der Pay-to-Public-Key-Hash Methode. Im Wesentlichen werden die Daten statt im Public-Key-Hash des Empfängers in den Public-Key des Empfängers kodiert. Dadurch können 65 Byte pro Transaktions-Output hinterlegt werden. Wie auch bei der Payto-Public-Key-Hash Methode gehen die verschickten Währungseinheiten verloren. Bei dieser Methode kann der Miner jedoch

überprüfen, ob es sich um echte unkomprimierte Public-Keys handelt und gegebenenfalls die Aufnahme in einen Block verweigern. Ein Workaround dazu könnte die Verwendung von komprimierten Public-Keys darstellen. Diese erlauben allerdings nur die Speicherung von 33 Byte an Daten. Auch diese Methode der Datenspeicherung ist umstritten, da sich die Menge der UXTO vergrößert.

### 2.3. OPRETURN

Um der unnötigen Vergrößerung des UXTO-Sets entgegenzuwirken, wurde eine Möglichkeit zur Datenspeicherung geschaffen, bei der sich das UTXO-Set nicht vergrößert. Die OP_RETURN [3] Standard Script Methode erlaubt die Speicherung von 80 Byte an Daten pro Transaktion.

### 2.4. Pay-to-Multisig

Bitcoin bietet eine Möglichkeit, Währungseinheiten zu überweisen, bei der mehrere Unterschriften bzw. Signaturen zur Behebung benötigt werden. Diese Pay-to-Multisig Methode erlaubt beispielsweise zu definieren, dass zwei der drei „Kontoinhaber° signieren müssen, um die Währungseinheiten zu transferieren. Bei dieser Methode können wiederum die Daten anstelle der Public-Keys hinterlegt werden. Um verlorene UTXO zu vermeiden, könnten beispielsweise 2 unkomprimierte Public-Keys (je 65 Byte) als Datenablage verwendet werden und ein dritter komprimierter echter Public-Key hinterlegt werden. Da die Behebung allerdings unökonomisch ist, ist es sinnvoller, gleich alle Public-Keys zur Datenablage zu verwenden.

### 2.5. Pay-to-Script-Hash / Data Drop / Data Hash

Bei dieser Methode können mittels der OP_PUSHDATA2 [4] Operation 517 Byte an Daten gespeichert werden. Mittels der „Data Drop“ [3] Methode können 1529 Byte an Daten in einem Inputscript gespeichert werden. Diese Methode legt Daten auf den Stack und entfernt sie anschließend wieder. In einer Transaktion können mehrere Inputs verwendet werden, wodurch sich Transaktionen mit der maximal erlaubten Größe (100 KB) erzeugen lassen. Die Data Hash Methode erlaubt die Speicherung von 1560 Byte an Daten in dem Inputscript. Auch hier lassen sich mehrere Inputs in einer Transaktion verwenden. Im Vergleich zur Data Drop Methode kann bei der Data Hash Methode die Integrität der Daten gewährleistet werden. Für Details zu diesen Verfahren wird auf [3] verwiesen.

2 Bitcoin verwendet ECDSA Schlüsselpaare als Adresse/Konto. Der Public-Key-Hash ist der 160 Bit lange Hashwert des öffentlichen Teils des dazugehörigen Schlüsselpaares. Für Details wird auf [6] verwiesen. 3 Bei OP_RETURN handelt es sich um einen von vielen Opcodes der von Bitcoin verwendeten Script-Sprache. Für Details wird auf [8] verwiesen. 4 Bei OP_PUSHDATA2 handelt es sich um einen von vielen Opcodes der von Bitcoin verwendeten Script-Sprache. Für Details wird auf [7] verwiesen.

### 2.6. Coinbase

Die vorherigen Methoden zur Datenspeicherung können von allen Teilnehmern des Netzwerkes verwendet werden. Sie benötigen nur eine geringe Menge an Bitcoin. Miner haben zusätzlich noch die Möglichkeit, Daten in der Coinbase Transaktion abzulegen. Diese bietet 100 Byte Speicherplatz. Diese Methode führt zu keinen zusätzlichen Kosten für den Miner. Der Miner muss jedoch als Erster das Rätsel lösen bzw. den Block möglichst schnell verteilen, damit sich dieser im Netzwerk durchsetzt.

### 2.7. Zusammenfassung

Wenn man die Coinbase Transaktion ignoriert, da diese nur von Minern verwendet werden kann, ergeben sich im Beispiel Bitcoin je nach Datengröße unterschiedlich effektive Möglichkeiten zur Speicherung von Daten. Für kleine Mengen an Daten bis 80 Byte stellt die OP_RETURN Methode die effektivste Datenspeichermethode dar. Für mittelgroße Daten (80 bis 800 Byte) eignet sich die Pay-to-Multisig Methode. Für größere Daten eignen sich die Data Drop bzw. Data Hash Methoden.

## 3. Redactable Blockchain

Sowohl Industrie als auch Forscher haben erkannt, dass die Unveränderlichkeit der Daten in der Blockchain für gewisse Anwendungen ein Problem darstellt kann und arbeiten an Lösungen, die es erlauben, Daten bzw. Fehler nachträglich zu bearbeiten bzw. korrigieren. Prinzipiell kann man derzeit zwischen zwei verschiedenen Lösungsansätzen unterscheiden:

1. Verwendung spezieller Hashfunktionen, um die Verkettung nachträglich zu ändern. Dieser Ansatz wird im nächsten Abschnitt beschrieben.
2. Verwendung von Verschlüsselung. Dieser Ansatz wird im Abschnitt 3.2 behandelt.

Zusätzlich zu den beiden Ansätzen wird im Abschnitt 3.3 ein weiterer Ansatz beschrieben, der die Löschung bzw. Bearbeitung von Daten via spezieller Regeln, unter Zuhilfenahme einer zweiten Blockchain, erlaubt.

### 3.1. Redaction unter Verwendung spezieller Hashfunktionen

Ateniese u. a. [2] beschreiben in ihrer Publikation „ *Redactable Blockchain – or – Rewriting History in Bitcoin and Friends* “ einen Ansatz, der die nachträgliche Modifikation der Blockchain durch die Verwendung von Chameleon-Hashfunktionen erlaubt. Chameleon-Hashfunktionen sind spezielle Hashfunktionen, die das Finden einer Hash-Kollision (zwei unterschiedliche Datensätze bzw. Blöcke, die auf denselben Hash-Wert abgebildet werden) effizient ermöglichen, sofern das dafür benötigte Geheimnis bekannt ist. Wenn das Geheimnis unbekannt ist, ist es wie bei herkömmlichen kryptografischen Hashfunktionen sehr schwer bis praktisch unmöglich, eine Kollision zu finden.

Bei diesem Ansatz kann man sich die Blockchain als eine Sequenz von Blöcken vorstellen, die über eine Kette mittels Schloss verbunden sind. Das Schloss symbolisiert dabei das benötigte Geheimnis, um eine Hash-Kollision zu finden. Die Kette hingeben symbolisiert die Referenz zum Vorgängerblock. Abbildung 2 zeigt für einen Ausschnitt einer Blockchain die nötigen Schritte, um einen Block auszutauschen.

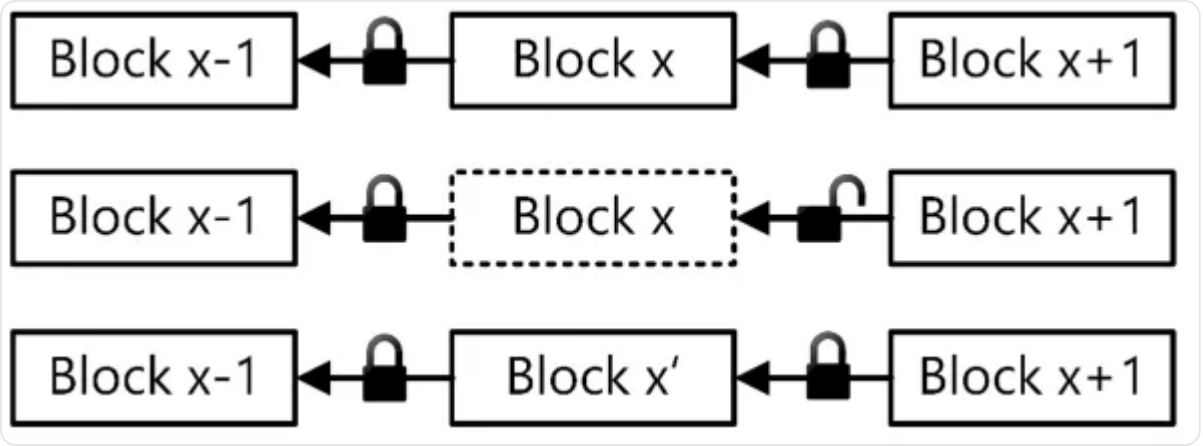



Abbildung 2: Austausch eines Blockes in der Blockchain.

Will man Block *x* verändern, öffnet man zuerst das Schloss, welches Block *x* und Block *x+1* verbindet. Dadurch kann Block *x* durch Block *x'* ausgetauscht werden. Das Schloss zwischen Block x und Block x-1 muss nicht geöffnet werden, da sich der Vorgängerblock bzw. dessen Hashwert nicht ändert. Anschließend wird das Schloss wieder verschlossen, um eine durchgehende Kette zu erhalten. Technisch gesehen wird Block *x'* unter zu Hilfenahme des Geheimnisses so erstellt, dass dessen Hashwert dem ursprünglichen Hashwert von Block *x* gleicht. Laut den Autoren eignet sich dieser Ansatz sowohl für private als auch für öffentliche Blockchains.

Bei privaten Blockchains, die typischerweise in Firmen eingesetzt werden, bekommt eine zentrale vertrauenswürdige Instanz das benötigte Geheimnis, um Schwärzungen bzw. Änderungen durchzuführen. Diese Instanz kann dann nach eigenem Ermessen Blöcke einfügen, entfernen oder bearbeiten.

Bei öffentlichen Blockchains, wo zentrale vertrauenswürdige Instanzen unerwünscht sind, schlagen die Autoren vor, das Geheimnis auf mehrere Miner aufzuteilen. Durch geeignete Protokolle kann dabei sichergestellt werden, dass nur m aus n Miner gemeinsam Änderungen durchführen können. Dabei kennt jeder Miner nur einen kleinen Teil des Geheimnisses. Die Autoren beschreiben eine Integration in Bitcoin. Der verwendete Ansatz dürfte jedoch nicht für stark dezentralisierte Währungen mit mehreren Tausend Minern geeignet sein, da der Aufwand der Schlüsselaufteilung bzw. Zusammenführung stark steigt und es keinen (finanziellen) Anreiz für Miner gibt, sich an dem Protokoll zu beteiligen.

## 3.2. Redaction via Verschlüsselung

Puddu u. a. [3] präsentieren in ihrem Paper „µchain: How to Forget without Hard Forks“ einen Ansatz, die aktuell gültige Sicht auf die Blockchain zu verändern. Der Ansatz beruht auf der Verschlüsselung der Transaktionen. Die Grundidee ist, dass jede Transaktion durch ein Set von Transaktionen ausgetauscht wird. Aus diesen Sets kann immer nur eine Transaktion aktiv sein. Um eine Transaktion zu schwärzen wird die aktive Transaktion gewechselt und die Auslieferung des alten Entschlüsselungsschlüssels gestoppt. Neben dem Transaktionsset benötigt der Ansatz noch einen Verweis auf die derzeit gültige Transaktion sowie eine Richtlinie, die definiert, wer die aktive Transaktion umschalten darf und wie lange dies möglich ist. Jede Transaktion im Transaktionsset wird verschlüsselt, und es werden nur die Entschlüsselungsschlüssel der derzeit aktiven Transaktion verteilt. Zusätzlich muss jedes Transaktionsset noch eine unverschlüsselte „ *leere* “ Transaktion beinhalten. Diese Transaktion wird als „Nope“-Transaktion

bezeichnet. Neben der „ *Nope* "-Transaktion definieren die Autoren noch eine „ *Classical* ", eine „ *Deployment* ", eine „ *Mutant* " und eine „ *Extending* " Transaktion. Die verschiedenen Transaktionstypen werden in der folgenden Aufzählung kurz vorgestellt:

- Die „ *Nope* "-Transaktion bedeutet „keine Operation". Wird sie aktiv geschalten, wirkt es so als wäre das Transaktionsset nicht vorhanden. „ *Nope* "-Transaktion haben keinen Sender, Empfänger sowie kein Datenfeld.
- Die „ *Classical* "-Transaktion entspricht einer herkömmlichen Transaktion und dient dem Transfer von Währungseinheiten von einem Sender zu einem Empfänger.
- „ *Deployment* "-Transaktionen werden benötigt, um Smart-Contracts zu erstellen.
- Mittels „ *Mutant* "-Transaktionen kann die aktive Transaktion in einem Transaktionsset umgeschaltet werden. Dabei müssen die Regeln der zum Transaktionsset gehörigen Richtlinie erfüllt werden. Die Blockchain wird dann so gelesen, als wäre schon immer diese Transaktion gültig gewesen.
- „ *Extending* "-Transaktionen erlauben das nachträgliche Erweitern eines Transaktionssets. Auch „ *Extending* "-Transaktionen unterliegen den Regeln, der zum Transaktionsset gehörigen Richtlinie. Bezüglich des Schlüsselmanagements unterscheiden die Autoren zwischen zwei Designmöglichkeiten. Die erste Variante wird als „Simple Encryption" bezeichnet. Dabei hat jeder Miner die Entschlüsselungsschlüssel für alle Transaktionen. Dieser Ansatz eignet sich vor allem für private Blockchains, wo davon ausgegangen werden kann, dass sich Miner ehrlich und

verantwortungsvoll verhalten. Der zweite Ansatz „Encryption with Secret Sharing" ist für öffentliche Blockchains gedacht. Dabei werden die Entschlüsselungsschlüssel in Stücke aufgeteilt und diese Teile werden unter den Minern verteilt. Der eigentliche Schlüssel kann aus den Teilen nur rekonstruiert werden, wenn sich genügend Miner beteiligen. Der Vorteil dieses Ansatzes liegt in der Tatsache, dass hier weder reguläre Benutzer oder Benutzerinnen noch Miner Zugriff auf die inaktiven Transaktionen haben. Durch den verwendeten Ansatz könnte es zu negativen Kontoständen kommen. Die folgende Aufzählung liefert eine beispielhafte Sequenz von Transaktionen, die im Endeffekt zu einem negativen Kontostand führen würde:

1. Angenommen die Teilnehmer A und B haben zu Beginn jeweils 10 Einheiten und C hat null Einheiten.
2. Anschließend überweist A 10 Einheiten an B.
3. Daraufhin überweist B 15 Einheiten an C. Nun hätte A null Einheiten, B 5 Einheiten und C 15 Einheiten.
4. Tauscht nun Teilnehmer A die aktive Transaktion durch eine „Nope"-Transaktion aus, hätte B plötzlich -5 Einheiten. Diesen ungültigen Zustand verhindern die Autoren, indem alle Transaktionen, die von anderen Transaktion abhängen ebenfalls verändert werden, wenn die ursprüngliche Transaktion angepasst wird. Da dieses Verhalten für Händler möglicherwiese negative Konsequenzen haben kann, schlagen die Autoren die Aufteilung des Guthabens in fixe und veränderbare Anteile auf. Zusätzlich können Empfänger von Transaktionen definieren ob sie nur fixe oder auch veränderbare Transaktionen akzeptieren. Bei veränderbaren Transaktionen kann der Empfänger zudem den maximal erlaubten Veränderungszeitraum definieren.

### 3.3. Redaction via Blockchain-Regeln

Im Folgenden wird ein neuer Ansatz vorgestellt, der die Veränderung der Blockchain erlaubt. Dieser Ansatz wurde im Rahmen dieses Projektes weiterentwickelt und implementiert. Im Gegensatz zu den zuvor vorgestellten Ansätzen basiert dieser Ansatz auf der Idee eine zweite Blockchain parallel zu führen, die die Korrekturen in der ersten Blockchain quasi „legitimiert". Dieser Ansatz ist auf öffentliche Proof-of-Work passierte Systeme ausgelegt. Dabei sollen folgende Ziele erreicht werden:

- Keine Veränderung der Vertrauensannahme. D.h., wie auch bei herkömmlichen Proof-of-Work-basierten Systemen soll die Blockchain sicher gegen Manipulationen sein, solange mehr als die Hälfte der Rechenleistung von ehrlichen Nodes, die den Regeln folgen, zur Verfügung gestellt wird.
- Wenn die Mehrheit der Miner für eine Korrektur ist, muss diese ausgeführt werden, um eine gültige Kette zu erhalten.
- Der Ansatz soll unabhängig von der Anzahl der Miner skalieren und im Speziellen ohne spezielles Schlüsselmanagement und dessen Nachteile auskommen.

Die Idee hinter diesem Ansatz ist, dass beide Blockchains, die „herkömmliche" und die Schwärzungs-Blockchain von einem gemeinsamen Startblock (Genesis-Block) aus starten. Wie auch bei Bitcoin dient dieser als Vertrauensanker. Die Regeln des Netzwerkes entsprechen im Wesentlichen den Regeln des Bitcoin Netzwerks. Beispielsweise können Währungseinheiten nur einmal ausgegeben werden und es können auch nicht mehr Währungseinheiten ausgegeben werden, als vorhanden sind. Wie auch bei Bitcoin wird die Blockchain mit dem größten kombinieren Proof-of-Work als gültig angesehen. Abbildung 3 zeigt den Start einer solchen Blockchain, ohne Schwärzungskette. Im Gegensatz zu herkömmlichen Blockchains hat jeder Block noch ein zusätzliches Feld, welches auf den letzten Schwärzungsblock zeigt. Ist noch kein Schwärzungsblock vorhanden, zeigt das Feld auf den Genesis-Block. Die in Abbildung 3 eingezeichneten Pfeile symbolisieren die Verkettungen. Das Pfeilende zeigt auf das Feld, welches den Wert bzw. die Referenz speichert und die Pfeilspitze visualisiert welche Daten einfließen, bzw. woher diese kommen. Als Beispiel, der Pfeil vom Feld „Vorgänger Kette 1" im „Block 1" zum „Genesis Block" visualisiert, dass das Feld „Vorgänger Kette 1" auf den „Genesis Block", bzw. dessen Hashwert, referenziert. Für bessere Sichtbarkeit sind die Verlinkungen auch farblich enkodiert.

Um eine Änderung durchzuführen, muss eine Wahl durchgeführt und gewonnen werden. Die Details zur Wahl werden im Abschnitt 3.3.1 beschrieben. Abbildung 4 zeigt die Blockchain für den Fall, dass eine Wahl gewonnen wurde (der Wahlvorgang wurde der Übersicht wegen nicht eingezeichnet), die Block 2 schwärzen möchte. Dementsprechend wird ein Schwärzungsblock erstellt.




Abbildung 4: Blockchain mit einer herkömmlichen Kette und einem Schwärzungsblock.

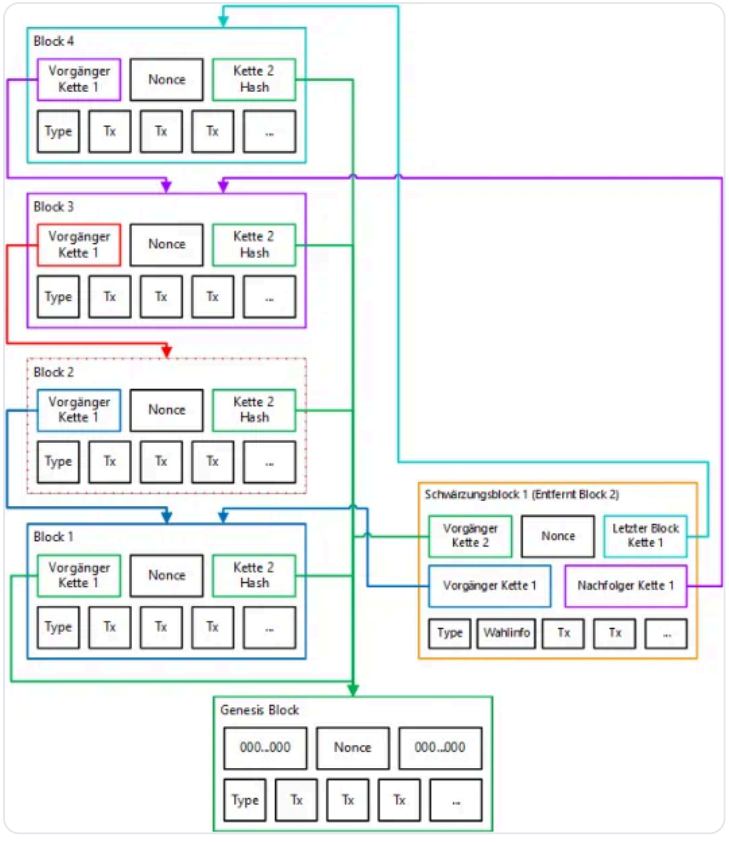



Abbildung 3: Blockchain ohne Schwärzungskette (von unten nach oben gezeichnet). Der Schwärzungsblock enthält eine Referenz zu dem letzten Schwärzungsblock, zu dem letzten Block in der Kette 1, zu dem Block in der Kette 1, der geschwärzt werden soll (enthalten in den Wahlinformationen), zu dessen Vor- und Nachfolge Block sowie zur Wahl.

Der Schwärzungsblock bürgt im Wesentlichen dafür, dass die Unterbrechung in der „herkömmlichen“ Blockchain „ok“ ist. Alternativ kann man sich vorstellen, dass ein virtueller Block erstellt wird, der die Lücke schließt.

### 3.3.1. Voting-Prozess

Es muss nachvollziehbar sein, worüber abgestimmt wurde und wie viele Stimmen dafür oder dagegen abgegeben wurden. Aus diesem Grund benötigt der vorgestellte Ansatz zwei weitere Transaktionstypen: Wahltransaktionen und Abstimmungstransaktionen. Der Einfachheit halber und um das Vertrauensmodell nicht zu ändern, dürfen derzeit nur Miner Wahltransaktionen oder Abstimmungstransaktionen erstellen. Jede Wahltransaktion enthält Informationen darüber, welcher Block bzw. welche Transaktion verändert werden soll. Der Einfachheit wegen beschränken wir uns vorerst auf Datentransaktionen, da hierbei keine unerwünschten Seiteneffekte wie negative „Kontostände“ auftreten. Zusätzlich enthält die Wahltransaktion noch eine Referenz zum Vor- und Nachfolgeblock des zu schwärzenden Blocks.

Um eine Wahl zu starten, muss ein Miner eine Wahltransaktion erstellen und diese direkt nach der Coinbase-Transaktion in den Block aufnehmen. Nachdem der neue Block erstellt wurde, überprüfen die anderen Teilnehmer dessen Gültigkeit. Zusätzlich zu den herkömmlichen Überprüfungen müssen noch die Informationen in der Wahltransaktion überprüft werden. Ist der Block gültig, können alle Miner einen definierten Zeitraum (Blockmenge) lange abstimmen. Dazu muss eine Abstimmungstransaktion erstellt und direkt nach der Coinbase-Transaktion in einen Block aufgenommen werden. Die Abstimmungstransaktion enthält eine Referenz zur Wahltransaktion (deren Fingerabdruck) sowie ein Feld für die Stimmabgabe (dafür/dagegen). Nachdem der definierte Wahlzeitraum verstrichen ist, muss der nächste Miner alle abgegebenen Stimmen auswerten und überprüfen, ob die Mehrheit dafür war. Ist dies der Fall, erstellt der Miner einen Schwärzungsblock, ansonsten wird ein herkömmlicher Block erstellt. Dadurch, dass die gesamte Abstimmung in der Blockchain nachvollziehbar hinterlegt ist, kann jeder überprüfen, wie die Wahl ausging. Hierdurch ist klar definiert, ob der Nachfolgeblock ein Schwärzungsblock sein muss oder nicht. Dadurch gibt es nach Wahlende eine definierte gültige Kette. Wie auch bisher ist jene Kette gültig, die den höchsten akkumulierten Proof-of-Work hat, wobei jeder Block in dieser Kette wiederum allen Regeln folgen muss. Für die Berechnung werden beide Blockchains herangezogen, wobei für jeden virtuellen Block der ursprüngliche „Proof-of-Work“ Wert herangezogen wird. Dieser kann aus dem ursprünglichen (in der Wahltransaktion hinterlegten) Fingerabdruck ermittelt werden.

### 3.3.2. Schwärzungsablauf

In Abbildung 3 und Abbildung 4 wurde der Wahlvorgang nicht dargestellt. Abbildung 5 visualisiert die drei Phasen eines Schwärzungsvorganges inklusive der dafür benötigten Wahl. Abbildung 5 zeigt links die Blockchain ohne Schwärzungsblock. Vier Blöcke nach dem Genesisblock wurde eine Wahltransaktion erstellt, die den zweiten Block nach dem Genesisblock schwärzen will. Unter der Annahme, dass die (verkürzt dargestellte) Wahl gewonnen wurde, zeigt die mittlere Grafik die Erstellung eines Schwärzungsblocks (in blau). Abbildung 5 rechts, zeigt die Blockchain nach dem Hinzufügen eines weiteren Blocks. Der neu hinzugefügte Block zeigt sowohl auf dessen Vorgänger, als auch auf den neu erstellten Schwärzungsblock. Nach einer definierten Anzahl von Bestätigungsblöcken löschen alle Knoten den ursprünglichen Block 2 und stoppen somit auch die Weiterverteilung im Netzwerk. Dadurch wird die Weiterverbreitung der zu schwärzenden Daten verhindert.

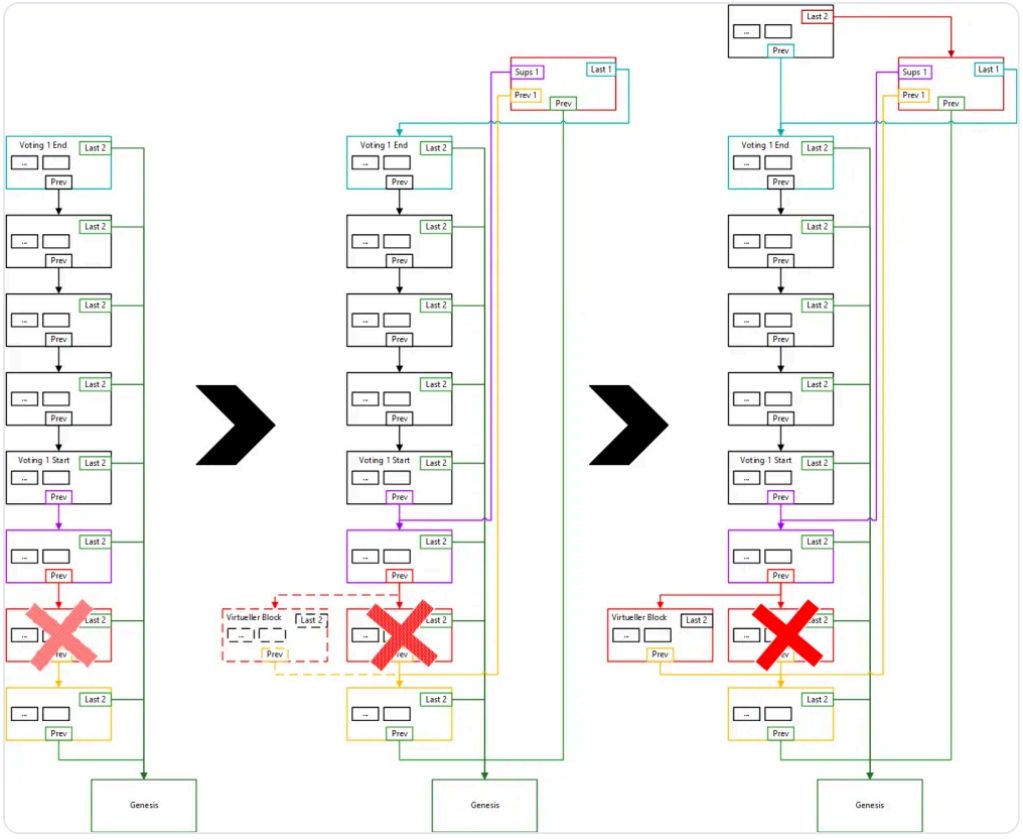



Abbildung 5: Visualisierung eines Schwärzungsvorganges mit vorheriger Wahl.

### 3.3.3. Umsetzung

Der in diesem Kapitel vorstellte Ansatz wurde in Java implementiert. Die aktuelle Umsetzung hat noch einige Limitierungen, wie das Fehlen eines echten Peer-to-Peer Netzwerkes, zeigt aber die prinzipielle Machbarkeit. Da sich der Sourcecode noch in einem frühen Entwicklungsstadium befindet und noch keine Sicherheitsüberprüfung stattfand, wird derzeit auf die Veröffentlichung verzichtet. Es ist die Überprüfung der Sicherheit des vorgeschlagenen Ansatzes mittels eines Modellcheckers geplant.

## 4. Conclusions

Jeder der drei vorgestellten Ansätze verfolgt leicht unterschiedliche Ziele und hat verschiedene Vorund Nachteile. Der erste vorgestellte Ansatz „Redaction unter Verwendung spezieller Hashfunktionen“ bietet eine mathematisch schöne Lösung um Blöcke dauerhaft aus der Blockchain zu löschen. Dieser Ansatz eignet sich jedoch Vorrangig für private Blockchains mit einer geringen Anzahl von Minern. Der zweite Ansatz „Redaction via Verschlüsselung“ versucht, das Problem von negativen Kontoständern zu lösen, dürfte aber je nach Use-Case ungeeignet sein, da die zu löschenden Daten nicht wirklich verschwinden, sondern nur der dazugehörige Entschlüsselungsschlüssel (von ehrlichen Teilnehmern) nicht mehr ausgeliefert wird. Bösartige Nutzer könnten den Schlüssel weiterhin verbreiten und so den Zugriff auf die verschlüsselten Daten, die von den ehrlichen Teilnehmern weiterhin ausgeliefert werden, ermöglichen. Der letzte vorstellte Ansatz hat den Vorteil, dass er unabhängig von der Anzahl der Miner funktioniert und dadurch vor allem für öffentliche Blockchains geeignet ist, sowie dass die zu löschenden Daten nicht weiterverbreitet werden. Der Nachteil dieses Ansatzes ist die derzeit noch fehlende Sicherheitsüberprüfung, wodurch dieser Ansatz derzeit noch nicht verwendet werden sollte.

## Referenzen

[1] S. Nakamoto, „Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available:

https://bitcoin.org/bitcoin.pdf. [Zugriff am 02 03 2018].

[2] A. Marsalek und B. Prünster, „Technologieüberblick Blockchain," www.a-sit.at, Graz, 2016.

[3] A. Sward, Vecna und F. Stonedahl, „Data Insertion in Bitcoins's Blockchain," 2017. [Online].

Available: https://digitalcommons.augustana.edu/cscfaculty/1/. [Zugriff am 04 06 2018].

[4] G. Ateniese, B. Magri, D. Venturi und E. Andrade, „Redactable Blockchain – or – Rewriting History in Bitcoin and Friends," 11 05 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7961975/. [Zugriff am 06 06 2018].

[5] I. Puddu, A. Dmitrienko und S. Capkun, „µchain: How to Forget without Hard Forks," 2017.

[Online]. Available: https://eprint.iacr.org/2017/106.

[6] Bitcoin Wiki, „Technical background of version 1 Bitcoin addresses," [Online]. Available:

https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses. [Zugriff am 17 09 2018].

[7] Bitcoin Wiki, „Script," [Online]. Available: https://en.bitcoin.it/wiki/Script. [Zugriff am 17 09 2018].

[8] Bitcoin Wiki, „OP_RETURN," [Online]. Available: https://en.bitcoin.it/wiki/OP_RETURN. [Zugriff am 17 09 2018].