Digitaler Posteingangsstempel

Blockchain als spezialisierte Datenbank im Solution Stack

Marcus Klüsener, Mahbouba Gharbi

Ein möglicher Anwendungsfall für eine Blockchain könnte so beschrieben werden: Ein Kunde schickt ein digitales Dokument an ein Unternehmen. Dieses erzeugt einen Hash-Wert des Dokumentes und veröffentlicht ihn in einer Blockchain. Der Kunde scannt die Blockchain nach diesem Dokument-Hash und erhält dadurch einen unveränderbaren digitalen Posteingangsstempel. So können Geschäftsprozesse optimiert, Kosten gesenkt und das Betrugsrisiko verringert werden. Der Artikel führt anhand dieses Anwendungsfalls und seiner Java-Implementierung als Zeitstempel-App in die Verwendungsmöglichkeiten von Blockchain in existierenden Client-/Server-Anwendungen und in dezentralen Anwendungen ein.

Wesentliches Element einer Blockchain ist eine Datenstruktur, die das sequenzielle Anfügen digitaler Transaktionen ermöglicht. Diese Datenstruktur befindet sich nicht bei einem einzigen vertrauenswürdigen Anbieter, sondern wird innerhalb eines verteilten Computernetzwerks freigegeben. Daraus ergibt sich ein offenes, transparentes und öffentlich verifizierbares System, das eine neue Art und Weise bietet, wie Anwendungen Werte und Assets austauschen, Verträge durchsetzen und Daten zur Verfügung stellen.

Konkrete Anwendungen außer Bitcoin (der Grund für die Erfindung von Blockchain) bestehen im Zeitstempeln, Verankern von Daten und notarieller Beurkundung.

Digitaler Posteingangsstempel

Werden im Geschäftsalltag wichtige Briefe geöffnet, dann bekommen diese im Normalfall einen Posteingangsstempel. Dadurch wird sichergestellt, dass ein Vertrag oder eine Terminbestätigung zu einem bestimmten Zeitpunkt in spezifizierter Form vorgelegen hat. Bei digitalen Medien ist diese exakte Absicherung schwieriger. Oft wird das Dokument noch ausgedruckt und dann mit einem normalen Stempel versehen und abgeheftet.

Ein digitaler Posteingangs- oder Zeitstempel auf Basis von Blockchain-Technologie wäre eine kostensparende Lösung. Ein weiterer Vorteil wäre die Möglichkeit, dass der Absender ohne Mehraufwand verifizieren kann, dass der Empfänger die Nachricht erhalten hat. Auch im Ernstfall vor Gericht könnte der digitale Posteingangsstempel für die Echtheit des Dokuments bürgen.

Die Erstellung des Zeitstempels funktioniert binnen weniger Augenblicke. Beim „Abstempeln“ einer Datei wird ein sogenannter Hash-Wert generiert. Dieser eindeutige digitale „Fingerabdruck“ wird in der Blockchain gespeichert. Da jeder Block in der Blockchain zu einem eindeutigen und unveränderbaren Zeitpunkt erstellt wurde, ist der Hash-Wert, praktisch eine Prüfsumme, mit einer eindeutigen Zeitmarke versehen und zusätzlich mit dem privaten Schlüssel digital signiert. Dieses ermöglicht es sowohl Sender als auch Empfänger nachzuweisen, dass ein Dokument und damit jede beliebige Information, die Sie in einer Datei unterbringen können, schon zu einem bestimmten Zeitpunkt in der Vergangenheit existierte, ohne den eigentlichen Inhalt des Dokuments zu veröffentlichen.

Da der Absender der Datei unabhängig vom Empfänger den Hash-Wert berechnen kann, solange er den gleichen Hash-Algorithmus verwendet, schafft diese Art der öffentlich verifizierbaren Speicherung Vertrauen beim Absender beziehungsweise Kunden.

Umsetzung der Zeitstempel-App mit bitcoinj

Die Bitcoin-Blockchain ist dank ihrer Bekanntheit und guten Dokumentation bestens für eine Beispielimplementierung geeignet. Mit bitcoinj existiert eine von Mike Hearn während seiner Zeit bei Google entwickelte Java-Bibliothek für die Arbeit mit dem Bitcoin-Protokoll. bitcoinj ist Basis zahlreicher Anwendungen und bietet eine Programmierschnittstelle, um eine Geldbörse (Wallet) zu pflegen, ermöglicht das Senden und Empfangen von Transaktionen, ohne eine lokale Kopie von Bitcoin Core vorzuhalten. Es ist in Java implementiert, kann aber von jeder JVM-kompatiblen Sprache verwendet werden.

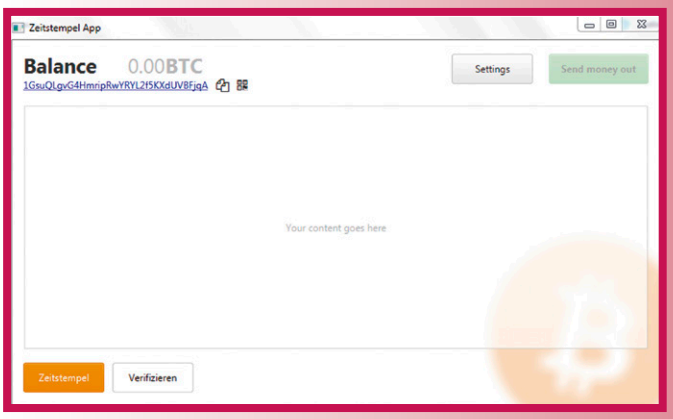

Abb. 1: Angepasstes Wallet

Für den digitalen Posteingangsstempel wird das mitgelieferte GUI-Wallet angepasst. Das Wallet dient der Aufbewahrung der öffentlichen und privaten Schlüssel, verwaltet den Kontostand dieser Adressen und ermöglicht Transaktionen. Damit bietet es schon sämtliche Grundfunktionen, um den Hash-Wert eines Dokumentes als digitalen Posteingangsstempel in der Bitcoin-Blockchain zu persistieren. In der in Abbildung 1 dargestellten Zeitstempel-Anwendung wird ein Wallet um zwei Buttons zum Zeitstempeln und zum Verifizieren ergänzt. Für die Implementierung der Zeitstempel-Funktion werden die in Abbildung 2 dargestellten Schritte umgesetzt.

Das Dokument wird mit SHA-256-Hash-Algorithmus konvertiert

Die erste die Blockchain-Technologie betreffende Teilaufgabe ist die Konvertierung des Dokuments in einen Hash-Wert. Der Prüfwert wird verwendet, um die Integrität einer Nachricht zu sichern. Wenn zwei Nachrichten den gleichen Prüfwert ergeben, soll die Gleichheit der Nachrichten nach normalem Ermessen garantiert sein. Solange eine Hash-Funktion als sicher gilt, werden zwei Dateien, die denselben Wert haben, immer denselben Inhalt haben.

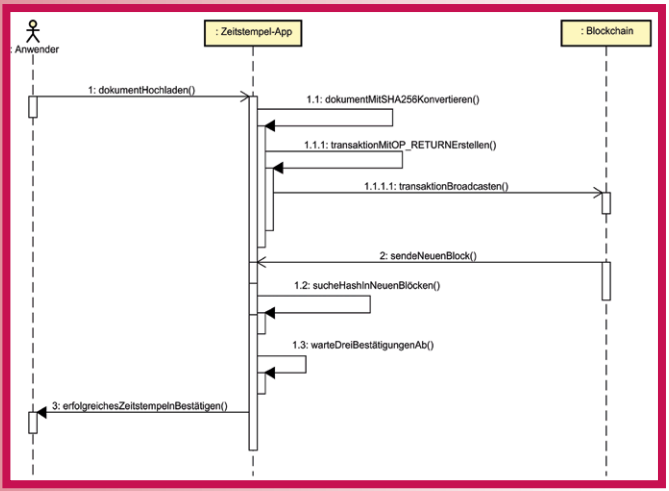

Abb. 2: Der Anwendungsfall zur Erzeugung eines Zeitstempels als Sequenzdiagramm

Theoretisch kann eine Kollision dann auftreten, wenn unterschiedlichen Eingabedaten derselbe Hash-Wert zugeordnet wird. Da die Menge der möglichen Hash-Werte meist kleiner ist als die der möglichen Eingaben, sind solche Kollisionen prinzipiell unvermeidlich. Für eine sichere Hash-Funktion sollte die Wahrscheinlichkeit dafür so klein sein, dass es in der Praxis nicht möglich ist, eine ausreichende Anzahl von Berechnungen durchzuführen, um sie zu finden.

```
1  FileInputStream in = new FileInputStream(datei);
2  byte[] dokument = ByteStreams.toByteArray(in);
3
4  MessageDigest digest = MessageDigest.getInstance("SHA-256");
5  Sha256Hash hash = new Sha256Hash(digest.digest(dokument));
6
7  // Erzeugen einer Transaktion mit OP_RETURN
8  Transaction tx = new Transaction(Main.params);
9  tx.addOutput(Coin.ZERO, ScriptBuilder.
10   createOpReturnScript(hash.getBytes()));
11 // Senden der Transaktion an das Bitcoin-Netzwerk
12 Main.bitcoin.wallet().sendCoins(Wallet.SendRequest.forTx(tx));
13
14 // Erzeugen und Speichern des Nachweises der Transaktion
15 Proof proof = new Proof();
16 proof.tx = tx.bitcoinSerialize();
17 proof.filename = data.toString();
18 pendingProofsList.getItems().add(proof);
```

Listing 1: Zeitstempelfunktionalität

In der Bitcoin-Blockchain und für den digitalen Posteingangsstempel wird der SHA-256-Algorithmus verwendet. Der Begriff Secure Hash Algorithm (SHA) bezeichnet eine Gruppe standardisierter kryptografischer Hash-Funktionen. Der SHA-256-Algorithmus gehört zu der als sicher geltenden SHA-2-Familie und erzeugt einen Hash-Wert mit einer Länge von 256 Bit. In Listing 1, Zeile 4 wird der SHA-256-Algorithmus über die `MessageDigest`-Klasse im `java.security`-Package aufgerufen.

Es wird eine Transaktion erzeugt

Um arbiträre Daten in einer Transaktion zu persistieren, muss eine Transaktion mit einem geringen Betrag durchgeführt werden. Ohne Betrag ist es nicht möglich, Transaktionen auf der Bitcoin-Blockchain auszuführen. Die Zeitstempel-App erstellt eine Transaktion, wie beispielhaft in Listing 2 dargestellt. Da sie der alleinige Besitzer des privaten Schlüssels ist, kann über `scriptSig` die Authentizität nachgewiesen werden. Authentizität bedeutet in unserem Anwendungsfall, dass sich die Zeitstempel-App mit ihrem privaten Schlüssel und einer öffentlichen Bitcoin-Adresse als Quelle des Zeitstempels authentifizieren kann.

In Listing 1, Zeile 8 wird diese Transaktion erstellt. `Main.params` enthält die Netzwerkparameter, die für die Kommunikation mit der Bitcoin-Blockchain benötigt werden. Die Transaktion besteht nach der Initialisierung nur aus einer serialisierbaren Datenstruktur.

```
1  "hash": "0b5ad418305c554c2dc80aae1d839f2a1221f...",
2  "in": [
3    {
4      "prev_out": {
5        "hash": "251484e98feaf8294...",
6        "n": 1
7      },
8      "scriptSig": "304402201b036527c..."
9    }
10 ],
11 "out": [
12   {
13     "value": "0.00000000";
14     "scriptPubKey": "OP_RETURN 7febe04ba94fc8f534ab2a3e7..."
15   },
16   {
17     "value": "0.1982999"
18     "scriptPubKey": "";
19     "address": ""
20   }
21 ],
22 "nid": "";
23 "block": "000000000000000000f061205567dc79...",
24 "blocknumber": 443886,
25 "time": "2016-12-17 20:26:21"
```

Listing 2: Verkürzter JSON-Dump einer Transaktion mit OP_RETURN

Es wird ein OP_RETURN-Output an die Transaktion angehängt

Die Daten müssen über eine Skript-Sprache in der Blockchain gespeichert werden. Bitcoin verwendet ein Scripting-System für Transaktionen. Die Script genannte Sprache ist einfach, Stack-basiert und verarbeitet Befehle von links nach rechts. Sie enthält keine Schleifen und ist gezielt nicht Turing-vollständig. Das Speichern von Daten geschieht durch die Erzeugung einer speziellen Bitcoin-Transaktion, die den Hash über ein `OP_RETURN`-Skript codiert. Dies ist ein Bitcoin-Scripting-Opcode, der die Transaktionsausgabe als ungültig markiert und bis zu 80 Byte an Daten einfügt.

Der Script-Befehl `OP_RETURN` (in Listing 2, Zeile 14) ermöglicht das Einfügen von Daten in eine Transaktion. Dazu wird eine Nullwerttransaktion mit einem Script, das ein `OP_RETURN` mit dem Hash des Dokuments enthält, ausgeführt.

Die Transaktion wird an das Bitcoin-Netzwerk gesendet

Bis jetzt wurde noch kein Betrag übertragen und kein Hash persistiert. Die erste Anlaufstelle, die die erzeugte Transaktion nun erreicht, sind die Knoten des Bitcoin-Netzwerks.

Das Bitcoin-Netzwerk ist primär ein Peer-to-Peer-Payment-Netzwerk, das auf einem kryptografischen Protokoll basiert. Transaktionen werden in der Blockchain einer verteilten, replizierten öffentlichen Datenbank mit einem Konsens, der durch ein Proof-of-Work-System namens „Mining“ erreicht wird, aufgezeichnet. Das Protokoll wurde 2008 entworfen und 2009 als Open-Source-Software von „Satoshi Nakamoto“, dem Namen oder Pseudonym der ursprünglichen Entwickler, veröffentlicht.

In Listing 1, Zeile 12 wird ein `SendRequest` erzeugt, der angibt, wie eine Transaktion ausgeführt werden soll. Im `SendRequest` kann unter anderem konfiguriert werden, wie hoch die Netzwerkgebühr sein soll oder wohin das Wechselgeld gesendet werden soll. Die Methode `sendCoin` sendet die Coins gemäß dem `SendRequest` über einen TransactionBroadcaster an das Netzwerk.

Das Netzwerk erfordert eine minimale Struktur, um Transaktionen zu teilen. Ein ad hoc gebildetes, dezentrales Netzwerk von Freiwilligen reicht aus. Nachrichten werden nach dem Best-Effort-Prinzip übertragen, und die Knoten können nach Belieben das Netzwerk verlassen und wieder betreten. Nach der Wiederverbindung lädt ein Knoten die fehlenden Blöcke herunter und überprüft die neu hinzugekommenen. Danach ist der Knoten imstande, die neue Transaktion mit dem Dokument-Hash anhand bestimmter Vorgaben zu validieren. Das bedeutet, dass er unter anderem überprüft, ob die Zeitstempel-App genügend Kapital aufweisen kann, um die Transaktion durchzuführen. Dabei liegt der Speicherort dieses Kapitals in der Blockchain und nicht in dem Wallet.

Der Merkle Tree wird aus der Blockchain gelesen

Ein Merkle Tree oder Merkle-Baum [Wiki] ist ein Baum von Hash-Werten, die Blätter dieses Baumes sind Hash-Werte von Datenblöcken. Ein Datenblock kann beispielsweise den digitalen Posteingangsstempel einer Datei enthalten. Knoten weiter oben im Baum sind Hash-Werte ihrer Kinder. Die Bitcoin-Blockchain verwendet SHA-256 als Hash-Funktion.

Die Wurzel des Merkle-Baums wird als Merkle Root bezeichnet und im Block-Header gespeichert. Zum Verifizieren einer Transaktion in einem P2P-Netzwerk reicht es aus, die Merkle Root von einer vertrauenswürdigen Quelle zu beziehen. Liegt die Merkle Root vor, so kann der restliche Merkle-Baum auch von einer nicht vertrauenswürdigen Quelle bezogen werden. Er kann dann gegen die vertrauenswürdige Merkle Root geprüft und gegebenenfalls abgelehnt werden.

Der Hauptunterschied zu einer Hash-Liste ist, dass jeder Zweig des Merkle-Baums einzeln heruntergeladen und sofort auf Integrität geprüft werden kann, selbst wenn der komplette Baum noch nicht verfügbar ist. So können einzelne Zweige des Baumes (Merkle-Branches) effizient geladen und auf Integrität geprüft werden.

In Listing 1, Zeile 7 wird ein Merkle-Branch aus einem Block geholt und nach dem Hash des Dokuments durchsucht. Falls der Hash gefunden wird, wird der Merkle-Branch mit dem Hash gespeichert.

Es wird auf drei Bestätigungen des Netzwerks gewartet

Die unveränderliche Persistierung in der Blockchain basiert auf dem Proof-of-Work-Algorithmus, um Konsens hinsichtlich Doppelzahlungen in einem verteilten System zu erreichen. Doppelzahlungen sind das Ergebnis der erfolgreichen Ausgabe eines Betrags mehr als einmal. Bitcoin schützt vor Doppelzahlungen, indem jede Transaktion, die der Blockchain hinzugefügt wird, überprüft wird um sicherzustellen, dass die Eingaben für die Transaktion bisher noch nicht ausgegeben wurden. Falls eine Doppelzahlung festgestellt wird, wird nur eine der Transaktionen durchgeführt und die anderen werden verworfen.

Um sicherzustellen, dass die Transaktion nicht für ungültig erklärt wird, sollte eine Bestätigung durch mehrere Blöcke abgewartet werden. Die Sicherheit steigt dabei pro Block stark an. In dem Fall von digitalen Posteingangsstempeln reichen drei Blöcke, um eine ausreichende Sicherheit zu garantieren.

Sobald die Transaktion bestätigt ist, ist der Dokumenten-Hash dauerhaft gespeichert und bestätigt. Der Zeitpunkt entspricht dem Zeitstempel des Blocks. Wenn das Dokument zu dem Zeitpunkt nicht existiert hätte, als die Transaktion in der Blockchain gespeichert wurde, wäre es unmöglich gewesen, seinen Hash in die Transaktion einzubetten. Das Einbetten eines Hashs und die spätere Anpassung eines zukünftigen Dokuments an den Hash ist bei sicheren Hash-Algorithmen nicht möglich.

Überprüfen, ob der Hash-Wert in einem OP_RETURN-Output vorhanden ist

Das Verifizieren eines Dokuments kann bei bekannter Transaktions-ID wie in Listing 3 dargestellt umgesetzt werden. Dabei wird jeder Block nach der Transaktion, die den Hash des Dokumentes enthält, durchsucht. Um Daten in der Blockchain ohne Kenntnis der Transaktions-ID zu finden, muss die gesamte Blockchain durchsucht werden, und mittels String-Vergleich müssen die `OP_RETURN`-Felder analysiert werden.

Da die Bitcoin-Blockchain mittlerweile eine Größe von mehr als 120 GB hat, ist es hilfreich, einen Index über alle `OP_RETURN`-Transaktionen zu erstellen. Zusätzlich können zum leichteren Auffinden Marker-Bytes in das `OP_RETURN` integriert werden. Die Existenz der Transaktion mit dem Dokument-Hash in der Blockchain beweist, dass das Dokument zum Zeitpunkt der Transaktion in einen Block aufgenommen wurde.

```
1 //Sobald ein Merkle-Baum mit dem Hash erscheint, wird er gespeichert
2 Main.bitcoin.peerGroup().addEventListener(
      new AbstractPeerEventListener() {
3   @Override
4   public void onBlockDownloaded(Perr peer, Block block
5       FilteredBlock filteredBlock, int blocksLeft) {
6     List<Sha256Hash> hashes = new ArrayList<>();
7     PartialMerkleTree tree = filteredBlock.getPartialMerkleTree();
8     tree.getTxnHashAndMerkleRoot(hashes);
9     if (hashes.contains(tx.getHash())) {
10      proof.partialMerkleTree = tree.bitcoinSerialize();
11      proof.blockHash = filteredBlock.getHash();
12    }
```

Listing 3: Verifizierungsfunktionalität

Integration in Anwendungen

Um die Integration von Blockchain-Technologie im Unternehmen zu ermöglichen, wird eine Zugangsschicht benötigt. Diese Schicht lässt Anwendungen mit der Blockchain nahtlos arbeiten. Dabei kapselt die Schicht als Vermittler die Komplexität der Blockchain-Technologie, bietet Blockchain-Funktionalitäten an und kommuniziert sie zu Altanwendungen. Besonders die Nachteile der Blockchain-Technologie wie geringer Datendurchsatz, Speicherplatz und Geschwindigkeit bei hohen Kosten und schlechter Skalierbarkeit müssen in dieser Schicht abstrahiert werden.

Bei der Integration von Blockchain-Technologie ist es wichtig, die positiven Eigenschaften der Blockchain-Technologie zu erhalten. So bietet die Blockchain als dezentrale Datenbank öffentlichen Zugriff auf die Daten, nicht jedoch auf die gesamte Anwendung. Mit dezentralen Anwendungen (DApps) wird der Blockchain-Ansatz konsequent weiter gedacht und die ganze Anwendung inklusive Datenhaltung, Identitätsmanagement, Verarbeitung, Bandbreite und Währung dezentralisiert. In der Entwicklung befindet sich beispielsweise die Ethereum-Blockchain, die auch als „Weltcomputer“ bezeichnet wird. Hier ist jedoch noch Forschungsaufwand nötig.

Fazit

Die Integration ganzer Anwendungen in eine Blockchain ist im Augenblick Ziel zahlreicher Unternehmen und Forschungsvorhaben. Eine praktikable Integration von Blockchain in eine klassische Anwendung konnte anhand eines digitalen Posteingangsstempels beispielhaft gezeigt werden. Dabei wurde eine Kernfunktion der Blockchain, das unveränderliche Zeitstempeln von Transaktionen, genutzt, um einen digitalen Posteingangsstempel umzusetzen. Diese Funktionalität könnte Basis einer Zugangsschicht innerhalb einer Anwendungsarchitektur sein. Gelingt über diese Schicht die Integration der Blockchain-Technologie in die Anwendung, schafft das eine Ausgangsbasis für ein transparentes und öffentlich verifizierbares System, mit dem das Vertrauen von Nutzern gewonnen werden kann.

Links

[BitcoinDR] `https://bitcoin.org/en/developer-reference`
[bitcoinj] `https://bitcoinj.github.io/`
[TimeStamper] `https://github.com/mikehearn/devcoretalk`
[Wiki] Merkle Tree bzw. Hash-Baum, `https://en.wikipedia.org/wiki/Merkle_tree`

Marcus Klüsener ist Softwareentwickler und IT-Berater bei der ITech Progress GmbH. Sein Schwerpunkt liegt in der Architektur und Entwicklung von JavaEE-Anwendungen. Er verfügt über mehr als zehn Jahre Erfahrung in der Softwareentwicklung vom Start-up zum Enterprise und von der Spielebranche zur Medizintechnik. Zurzeit unterstützt er die öffentliche Verwaltung mit dem Ziel, die Kommunikation mit Bürgern zu verbessern. Vor diesem Hintergrund untersucht er die Schnittmengen zur Blockchain-Technologie. Er hat Erfahrung als Dozent an Hochschulen, als Sprecher auf Konferenzen und Autor mehrerer wissenschaftlicher Veröffentlichungen.
E-Mail: m.kluesener@itech-progress.com

Mahbouba Gharbi ist Gründerin und geschäftsführende Gesellschafterin der ITech Progress GmbH. Ihr berufliches Spektrum umfasst Tätigkeiten als Softwarearchitektin, Trainerin, Systementwicklerin, Reviewerin und Dozentin. Neben ihrer Tätigkeit als CEO der ITech Progress hat sie sich deutschlandweit einen Namen als Chefarchitektin für namhafte Kunden gemacht. Mahbouba Gharbi ist Mitgründerin und Vorstandsvorsitzende des International Software Architecture Qualification Board (iSAQB). Auf dem Gebiet der Softwarearchitektur bringt sie über 13 Jahre Berufserfahrung sowie die nötige Begeisterung mit, die sie durch ihre Vorträge und Veröffentlichungen weitergibt. E-Mail: m.gharbi@itech-progress.com