UNIVERSITÄT SALZBURG

Bitcoin

Kryptowährungen

Tomica Sachevski Michael Schreiber Mario Strohmeier

Universität Salzburg
Fachbereich Computerwissenschaften

Inhaltsverzeichnis

Kryptowährungen

Was sind Kryptowährungen?

- Kryptowährungen sind Währungen in Form digitaler Zahlungsmittel. Bei ihnen werden Prinzipien der Kryptographie angewandt, um ein verteiltes, dezentrales und sicheres digitales Zahlungssystem zu realisieren. Sie werden - im Gegensatz zu Zentralbankgeld - bis heute ausschließlich durch Private Personen geschöpft.

Quelle: [1]

Unterschied zwischen Währungen und Kryptowährung

Im Gegensatz zu Währungen, bieten Kryptowährungen:

- Zahlungsfreiheit (Zahlung jederzeit möglich)
- Sicherheit für Händler (z.B. keine “Chargebacks”)
- Neutralität (nicht vom Staat oder von Banken gesteuert)
- Anonymität

Die Nachteile sind jedoch:

- keine staatliche Überwachung
- keine Möglichkeiten der Rückabwicklung
- geringe Verbreitung

Kopieren von Krytpowährungen (“double-spend” Problem)

Die sog. ”Blockchain” ist die Lösung für ein generelles Digitales Problem: Das duplizieren von Daten, bzw. in unserem Fall das mehrmalige Ausgeben von digitaler Währung.
Sie enthält Einträge mit Transaktionen, welche durch Kryptographie verschlüsselt und validiert werden.
Die Blockchain ist für jeden öffentlich einsehbar und Fundamentaler bestandteil der meisten gängigen und beliebten Kryptowährungen.

Überblick beliebter Kryptowährungen

Rang	Währung	Marktkapitalisierung
1	Bitcoin	17.689 Mio. US-Dollar
2	Ethereum	4.508 Mio. US-Dollar
3	Ripple	790 Mio. US-Dollar
4	Dash (ehem. Darkcoin)	494 Mio. US-Dollar

Liste von Kryptowährungen [2]

Bitcoin

Übersicht

Was ist Bitcoin?

- Bitcoin („digitale Münze“) ist ein weltweit verwendbares, dezentrales Zahlungssystem und der Name einer digitalen Geldeinheit.

Quelle: [3]

Wie ist Bitcoin entstanden?

- Satoshi Nakamoto (Pseudonym) startet das Projekt im Mai 2007.
- Das Bitcoin-Zahlungssystem wurde erstmals im Oktober 2008 in einem veröffentlichten White-Paper beschrieben. Im Jahr darauf wurde eine Open-Source-Referenz-Software dazu veröffentlicht.
- Nakamoto half bei der Weiterentwicklung des Quellcodes bis Dezember 2010. Ab diesem Zeitpunkt wurde das Projekt jedoch von anderen Entwicklern übernommen (Open-Source).

Wie funktioniert Bitcoin?

- Basierend auf der Blockchain-Technologie
- Transaktionen finden zwischen sog. ”Wallets” statt
- Jedes Wallet hat einen einzigartigen Private Key, welcher für Transaktionen benötigt wird
- Transaktionen sind öffentlich einsehbar und meist in 10 Minuten bestätigt/validiert, anhand eines Prozesses der ”Mining” genannt wird

Anonymität

Die Transaktionen zwischen Adressen ist öffentlich einsehbar.
Jedoch sind hier auf den ersten Blick lediglich Zufallszahlen sichtbar.
Wenn jedoch einer dieser Adressen eine Identität zugewiesen werden kann, wäre es theoretisch möglich sämtliche Transaktionen nachzuverfolgen.
Dies kann jedoch erschwert werden, indem für jede Transaktion eine neue Wallet Private Key Adresse verwendet wird.

Beispiel:

- X hat in einem Forum eine Signatur mit seiner öffentlichen Bitcoin Adresse und bekommt Währung transferiert
- X kauft einen Schoko Riegel mit Bitcoins von Person Y
- Person Y will X ausspionieren! Daher analysiert Y die Blockchain und googelt sämtliche Adressen
- Person Y findet X anhand der Signatur im Forum, welche seine Adresse enthält

Wie bekommt man eine eigene Bitcoin ”Adresse”

Bitcoin Adressen

Bitcoin Adressen bestehen aus dem Private und Public Key:

- Private key
  - zufällig generiert (Zahlen und Buchstaben)
  - für Transaktionen benötigt
  - muss geheim bleiben
  - wird benötigt um auf ein Wallet zuzugreifen
- Public key
  - mathematisch zu diesem Private Key dazugehörend
  - kann öffentlich geteilt werden
- Bitcoin Adressen
  - mathematisch zu diesem Public Key dazugehörend
  - wird benötigt um eingehende Zahlungen zu erhalten

Private Key -> Public Key -> Bitcoin Adressen

Wie werden Bitcoins gespeichert?

Wie werden Bitcoins gespeichert?

- Simple Local Storage (Hot Storage)
  - Der einfachster Weg, um Bitcoins zu speichern, ist auf einem lokalen Gerät
- Cold Storage (Offline Speicherplatz)
  - Hierarchical wallet
  - Brain wallet
  - Paper Wallet
  - Hardware wallet
- Online Speicherplatz
  - Online Wallets

Quelle: [4, S. 101-113]

Threshold cryptography

Dient der erweiterten Sicherheit bei Transaktionen. Das Wallet (in welchem Bitcoins gespeichert werden) kann so konfiguriert werden, dass Transaktionen von mehreren Instanzen authorisiert werden müssen.
Z.B. kann eine Transaktion auf dem PC getätigt werden, welche jedoch abschließend von der Handy App authorisiert bzw. bestätigt werden muss.

Quelle: [4, S. 108-112]

Multi-signatures

Eine Multi-Signature Adresse ist eine Adresse, die mit mehr als einem Private Key verknüpft ist. Um eine Transaktion zu betätigen, müssen mind. N aus M Personen ihren Key zur Verfügung stellen.

Quelle: [4, S. 108-112]

Blockchain

Grundlagen I

- Jeder Block hat:
  - Hash des vorherigen Blocks
  - Daten
  - Hash dieses Blocks
- Es gibt einen initialen Block, da dieser referenziert werden muss.
- Durch das referenzieren der vorherigen Blöcke entsteht die Blockchain.

Grundlagen II

- Die Blockchain wird über P2P verschickt
- Was passiert, wenn eine Transaktion getätigt wird?
  - Der neue Block wird erzeugt, durch:
    - Referenzieren des vorherigen Blocks
    - Die Daten werden abgespeichert
    - Der SHA-256 Hash des Blocks wird erzeugt
  - Die neue Blockchain wird zu den Peers geschickt und es wird überprüft, ob es sich um eine korrekte Blockchain handelt

Praktisches Beispiel

Einfache Blockchain in Javascript mit Node.js
Umsetzung der Grundlagen

Download + Anleitung ist unter
`https://github.com/Raydercorp/blockchain`
zu finden.

Aufbauend auf: [5]

Code-Snippets I

Block erstellen

```
generateNextBlock = (blockData) => {
    var previousBlock = getLatestBlock();
    var nextIndex = previousBlock.index + 1;
    var nextTimestamp = new Date().getTime() / 1000;
    var nextHash = calculateHash(nextIndex, previousBlock.hash, nextTimestamp, blockData);
    return new Block(nextIndex, previousBlock.hash, nextTimestamp, blockData, nextHash);
};

calculateHash = (index, previousHash, timestamp, data) => {
    return CryptoJS.SHA256(index + previousHash + timestamp + data).toString();
};
```

Code-Snippets II

Block hinzufügen

```
addBlock = (newBlock) => {
    if (isValidNewBlock(newBlock, getLatestBlock())) {
        blockchain.push(newBlock);
    }
};

isValidNewBlock = (newBlock, previousBlock) => {
    if (previousBlock.index + 1 !== newBlock.index) {
        console.log('invalid index');
        return false;
    } else if (previousBlock.hash !== newBlock.previousHash) {
        console.log('invalid previoushash');
        return false;
    } else if (calculateHashForBlock(newBlock) !== newBlock.hash) {
        console.log(typeof (newBlock.hash) + ' ' + typeof calculateHashForBlock(newBlock));
        console.log('invalid hash: ' + calculateHashForBlock(newBlock) + ' ' + newBlock.hash);
        return false;
    }
    return true;
};

calculateHashForBlock = (block) => {
    return calculateHash(block.index, block.previousHash, block.timestamp, block.data);
};
```

Code-Snippets III

Peer broadcast bearbeiten

```
handleBlockchainResponse = (message) => {
    var receivedBlocks = JSON.parse(message.data).sort((b1, b2) => (b1.index - b2.index));
    var latestBlockReceived = receivedBlocks[receivedBlocks.length - 1];
    var latestBlockHeld = getLatestBlock();
    if (latestBlockReceived.index > latestBlockHeld.index) {
        console.log('blockchain possibly behind. We got: ' + latestBlockHeld.index + ' Peer got: ' +
              latestBlockReceived.index);
        if (latestBlockHeld.hash === latestBlockReceived.previousHash) {
            console.log("We can append the received block to our chain");
            blockchain.push(latestBlockReceived);
            broadcast(responseLatestMsg());
        } else if (receivedBlocks.length === 1) {
            console.log("We have to query the chain from our peer");
            broadcast(queryAllMsg());
        } else {
            console.log("Received blockchain is longer than current blockchain");
            replaceChain(receivedBlocks);
        }
    } else {
        console.log('received blockchain is not longer than current blockchain. Do nothing');
    }
};
```

Code-Snippets IV

Blockchain ersetzen

```
replaceChain = (newBlocks) => {
    if (isValidChain(newBlocks) && newBlocks.length > blockchain.length) {
        console.log('Received blockchain is valid. Replacing current blockchain with received
              blockchain');
        blockchain = newBlocks;
        broadcast(responseLatestMsg());
    } else {
        console.log('Received blockchain invalid');
    }
};

isValidChain = (blockchainToValidate) => {
    if (JSON.stringify(blockchainToValidate[0]) !== JSON.stringify(getInitialBlock())) {
        return false;
    }
    var tempBlocks = [blockchainToValidate[0]];
    for (var i = 1; i < blockchainToValidate.length; i++) {
        if (isValidNewBlock(blockchainToValidate[i], tempBlocks[i - 1])) {
            tempBlocks.push(blockchainToValidate[i]);
        } else {
            return false;
        }
    }
    return true;
};
```

Zusätzliche Funktionen für Cryptocurrencies?

- Frühere Transaktionen müssen in den Daten referenziert werden können.
  - Es muss nicht die ganze Blockchain überprüft werden.
- Es wird eine Signatur benötigt.
- Eine Möglichkeit, um eine Transaktion mit bestimmten Usern zu machen.
- Effizientes bearbeiten mehrerer Transaktionen zum “gleichen“ Zeitpunkt.

Diese Funktionen sind alle in der Blockchain umgesetzt!

Quelle: [4, S. 78]

Umsetzung dieser Funktionen! I


```
"hash": "5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
"ver": 1,
"vin_sz": 2,
"vout_sz": 1,
"lock_time": 0,
"size": 404,
"in": [
  {
    "prev_out": {
      "hash": "3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
      "n": 0
    },
    "scriptSig": "30440..."
  },
  {
    "prev_out": {
      "hash": "7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
      "n": 0
    },
    "scriptSig": "3f3a4ce81..."
  }
],
"out": [
  {
    "value": "10.12287097",
    "scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY
          OP_CHECKSIG"
  }
]
```

Umsetzung dieser Funktionen! II

- Frühere Transaktionen werden durch die Inputs referenziert.
  - Die Hashes dienen als Pointer.
  - Außerdem wird der Index der vorhergehenden Transaktion des zu überprüfenden Outputs übergeben.
- Jeder input wird zusätzlich noch signiert, um zu beweisen, dass wir die vorherigen Outputs auch verwenden dürfen.

```
"in": [
  {
    "prev_out": {
      "hash": "3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
      "n": 0
    },
    "scriptSig": "30440..."
  },
```

Umsetzung dieser Funktionen! III

- Bestimmte User werden über die Outputs referenziert.
  - Zuerst gibt es das value Feld. Das sind die zu überweisenden Bitcoins.
    - Ist die Summe dieser Felder kleiner als die Summe der referenzierten Inputs, dann handelt es sich um eine Überweisung an den Aussteller des Blocks.
    - Ansonsten um eine Überweisung des Ausstellers.


```
"out": [
  {
    "value": "10.12287097",
    "scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY
          OP_CHECKSIG"
  }
]
```

Umsetzung dieser Funktionen! IV

- Da die Public Keys gehasht sind handelt es sich bei dem Feld scriptPubKey um eine Reihe von Scripts die ausgeführt werden, um die Transaktion durchzuführen.
  - Diese Transaktion kann von einem Public Key der zu X hasht, in Verbindung mit einer Signatur des Besitzers des Public Keys, eingelöst werden.
  - Um dies zu bewerkstelligen, ist die Signatur des Inputs auch ein script. Diese zwei scripts werden nun konkateniert und müssen nacheinander Fehlerfrei ablaufen, damit die Transaktion gültig ist.


```
"out": [
  {
    "value": "10.12287097",
    "scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY
          OP_CHECKSIG"
  }
]
```

Umsetzung dieser Funktionen! V

- Um mehrere Transaktionen zum gleichen Zeitpunkt bearbeiten zu können, werden keine einzelnen Transaktionen gespeichert, sondern eine Baumstruktur die mehrere Transaktionen beinhaltet.

Umsetzung dieser Funktionen! VI

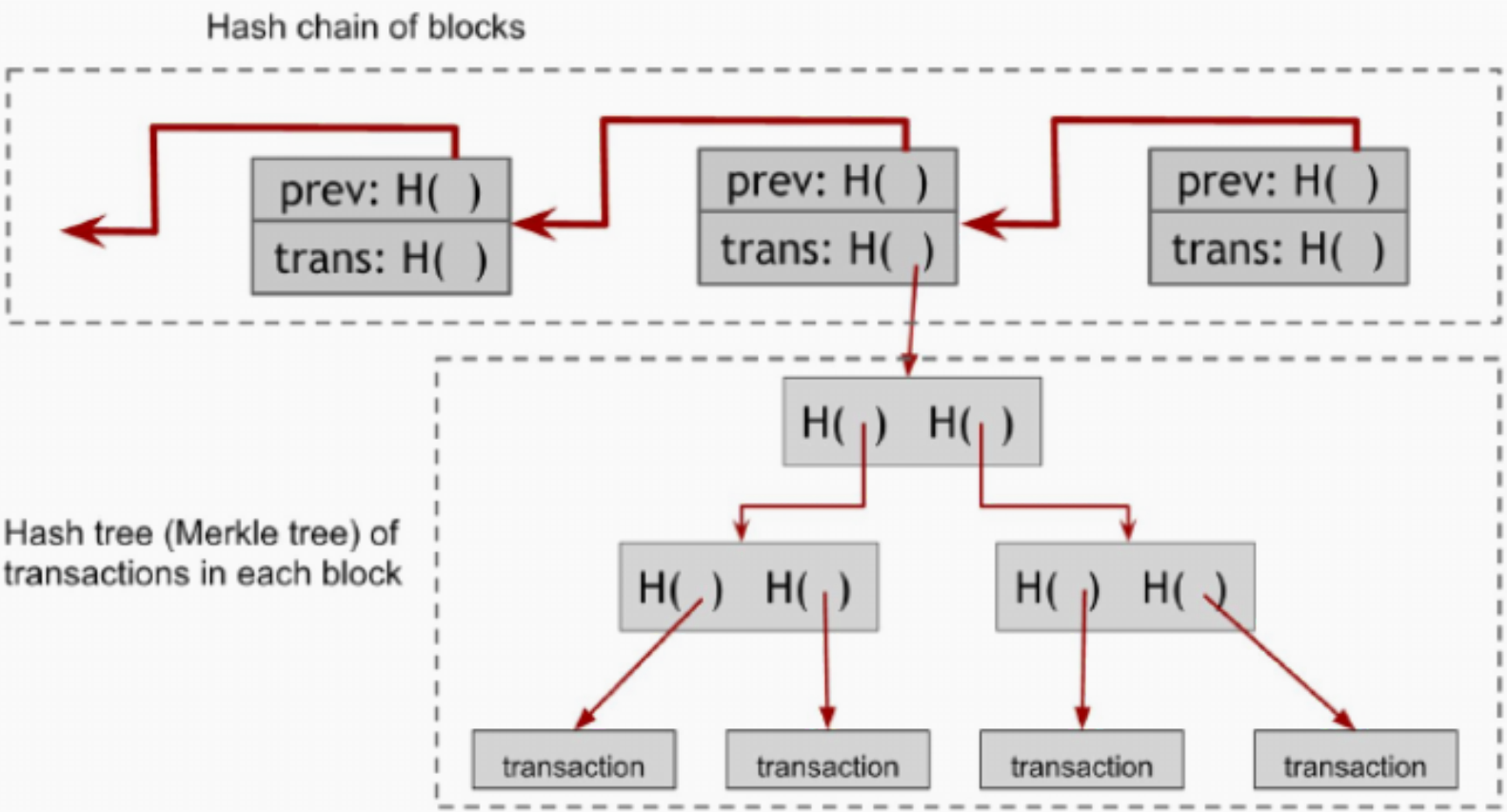

Abbildung 1: Baumstruktur [4, S. 88]

Quelle: [4, S. 78-89]

Bitcoin spezifische Daten

- Älteste Blockchain (Januar 2009)
- ∼126 GB groß
- ∼6700 Peers. Redundant und Öffentlich

Stand: März 2017 [6]

Mining

Wie funktioniert Bitcoin Mining?

Es gibt eine Liste mit Blöcken unbestätigter Transaktionen.
Diese Blöcke enhalten Informationen über die Transaktionen.
Die Miner bestätigen/validieren diese Blöcke und fügen sie der Block Chain hinzu.
Die Blöcke werden dabei in einen Hash umgewandelt, dabei wird auch der Hash des Vorgängerblocks mit einbezogen. Dies verhindert auch die Manipulation der Blockchain bzw. der Blöcke.
Für jeden neuen Hash verwendet die Mining-Software eine andere zufällig generierte Zahl für den Block Header (”nonce”).

Um das Mining schwieriger zu gestalten, gibt es die Ziel-Schwierigkeit:

- gültier Hash muss kleiner als Ziel-Schwierigkeit sein
- Ziel-Schwierigkeit wird alle 2016 Blöcke angepasst
- es werden ca. 14 Tage benötigt für 2016 valide Blöcke
- Ziel-Schwierigkeit wird an die Leistung im Netzwerk angepasst

Hardware

- CPU
- GPU (parallele Berechnungen) ca. +50-100% Hashrate im Vergleich zur CPU
- Field Programmable Gate Arrays (FPGA) ca. 5-fache Energieeffizienz im Vergleich zur GPU
- Application Specific Integrated Circuit (ASIC) ca. +100% Hashrate im Vergleich zur GPU / FPGA

Software

- ”Bitcoin Classic”:
  - Grafische Oberfläche (Desktop Programm)
  - Dient als Wallet (Anlegen / Verwalten von Adressen)
  - Überweisen / Empfangen von Bitcoins
  - kann als Server gestartet werden um Bitcoins zu ”minen”
- ”GUIMiner”:
  - Bitcoins ”minen”
  - Grafische Oberfläche
  - Unterstützt GPU / CPU

Pooled mining / Private mining

- Pooled mining:
  - mehrere Clients arbeiten an einem Block
  - geteiltes Einkommen anhand der geleisteten Arbeit (”processing power”)
  - regelmäßiges Einkommen
- Private mining:
  - nur ein Client arbeitet an einem Block
  - volles Einkommen beim finden eines gültigen Blocks
  - unregelmäßiges Einkommen

Noch Fragen?

References I

Wikipedia.
Kryptowährung.
https://de.wikipedia.org/wiki/Kryptow%C3%A4hrung.
Accessed: 24.04.2017.

Wikipedia.
Liste von Kryptowährungen.
https://de.wikipedia.org/wiki/Liste_von_Kryptow%C3%A4hrungen.
Accessed: 20.04.2017.

Wikipedia.
Bitcoin.
https://de.wikipedia.org/wiki/Bitcoin.
Accessed: 24.04.2017.

References II

Narayanana Arvind, Bonneau Joseph, Felten Edward, Miller Andrew, and Goldfeder Steven.
Bitcoin and Cryptocurrency Technologies.
https://d28rh4a8wq0iu5.cloudfront.net/bitcointech/readings/princeton_bitcoin_book.pdf?a=1.
Accessed: 14.04.2017.

lhartikk and lukaswelte.
Naivechain.
https://github.com/lhartikk/naivechain.
Accessed: 06.04.2017.

References III

Wikipedia.
Blockchain.
https://de.wikipedia.org/wiki/Blockchain#Anwendungsbeispiel_Bitcoin.
Accessed: 18.04.2017.