Blockchain

Public Key-Management & Sicherheit in Smart Contracts

Marvin Schwieger
Grundseminar - SoSe 2017

M.Sc. Informatik
HAW Hamburg
27.06.2017

Themen

- Überblick Blockchain
- Motivation
- Ausgewählte Forschung
- Blockchain im Master
- Konferenzen
- Quellen

Überblick

Anforderungen an digitalen Transaktionen

Simple Transaktionen

A

Simple Transaktionen

A

B

Simple Transaktionen

A

B

x1

Simple Transaktionen

A → B

x1

Simple Transaktionen

A → B

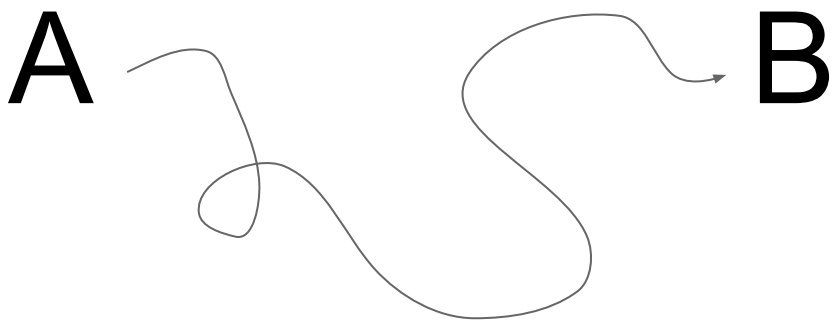
 x1

Simple Transaktionen

A

B

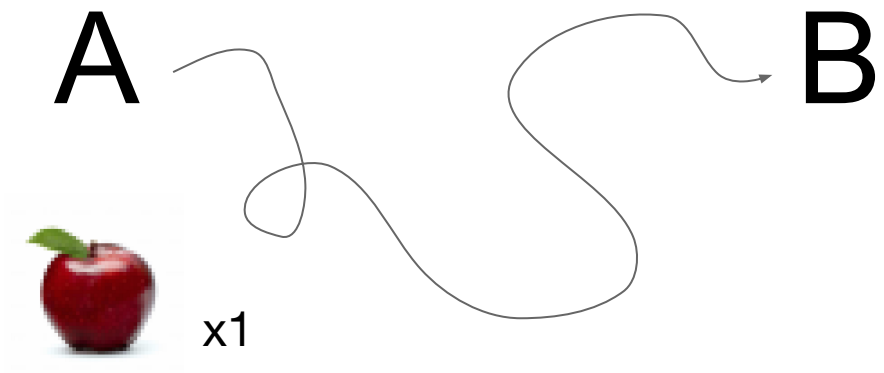
Simple Transaktionen



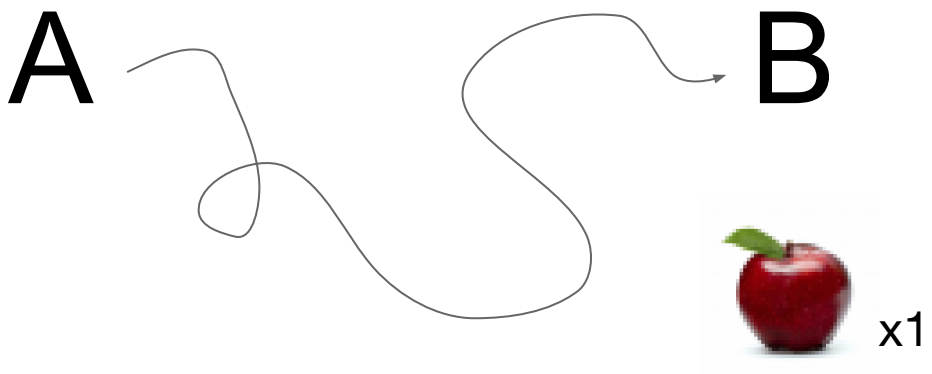
Digitale Transaktionen

Digitale Transaktionen

Digitale Transaktionen



Digitale Transaktionen



Digitale Transaktionen

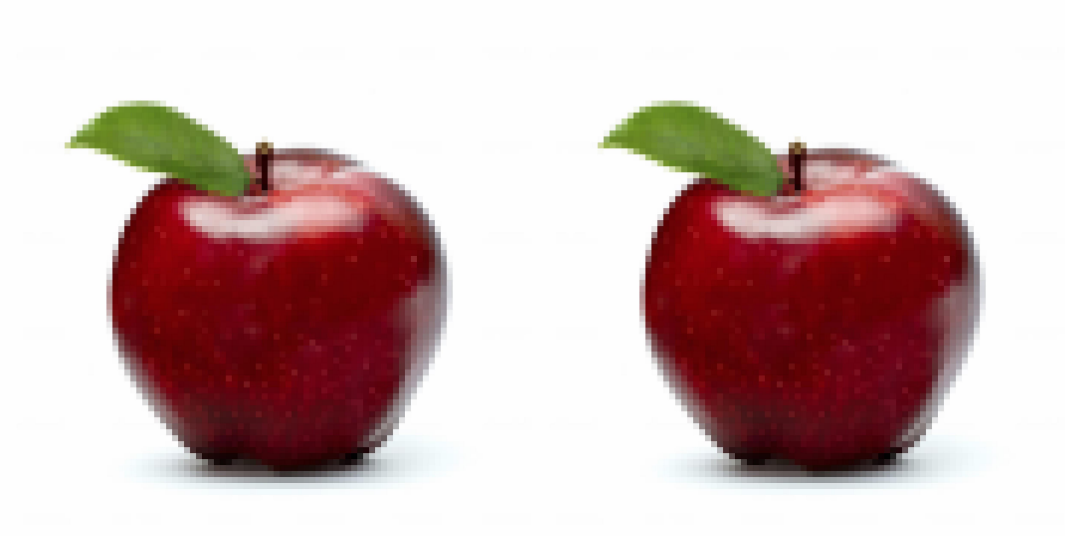
A → B

x1

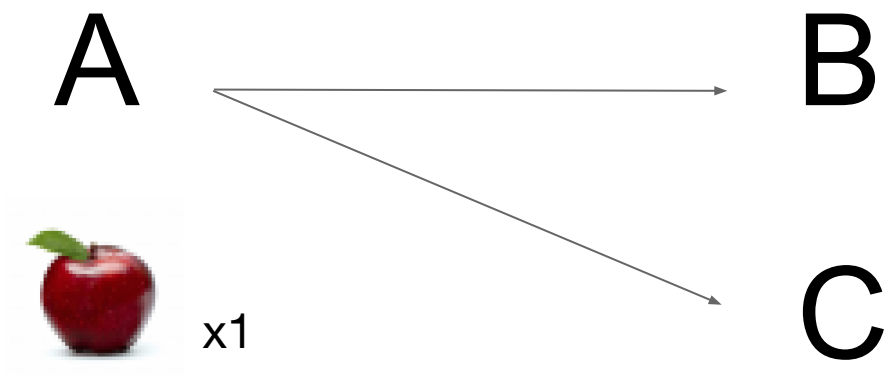
Probleme digitaler Informationen als Wertobjekt

Probleme digitaler Informationen als Wertobjekt

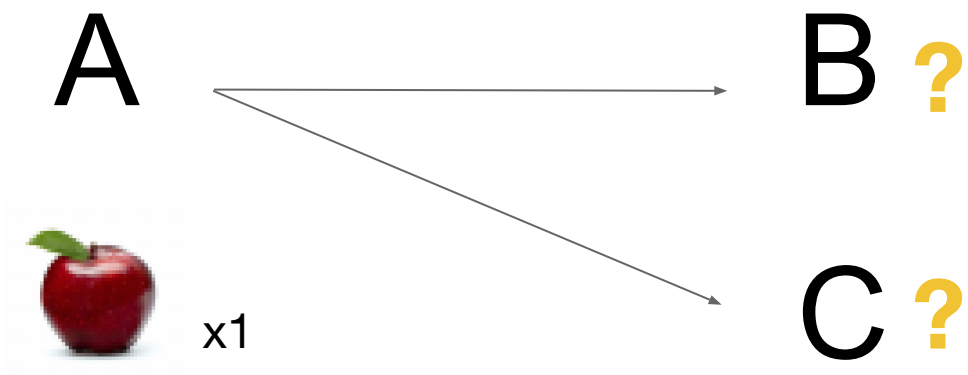
Probleme digitaler Informationen als Wertobjekt



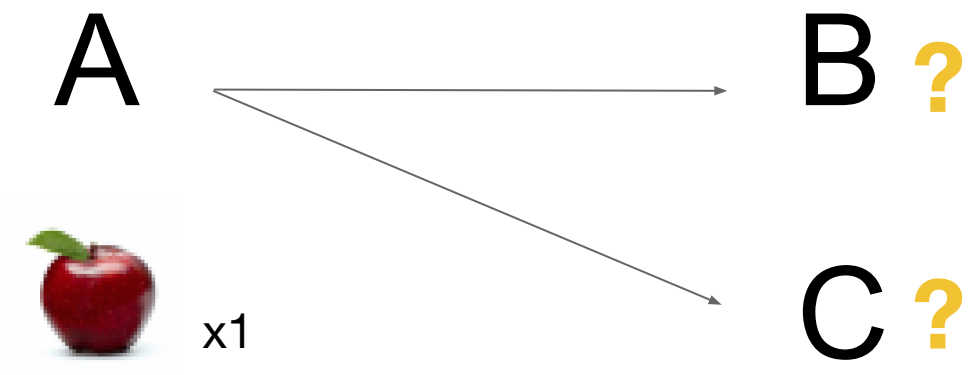
Probleme digitaler Informationen als Wertobjekt



Probleme digitaler Informationen als Wertobjekt



Probleme digitaler Informationen als Wertobjekt



Double Spending-Problem

Zusammenfassung

- Modifizierbarkeit digitaler Informationen
- Double Spending-Problem
- Vertrauen ist eine Notwendigkeit zwischen Teilnehmern einer Transaktion

Lösungsansatz

- Problem: Erfassen von Vermögen problematisch
- Idee: Dokumentation von Transaktionen

“Digital Ledger”

- Ledger - “Kontobuch, Führungsbuch”
- Durch Traversieren von Transaktionen lässt sich Vermögen eines Teilnehmers ermitteln

“Digital Ledger”

From	To	Amount
A	B	2

Abb. 1 - Vereinfachte Darstellung einer Transaktion in einem Ledger

“Digital Ledger”

From	To	Amount
A	B	2

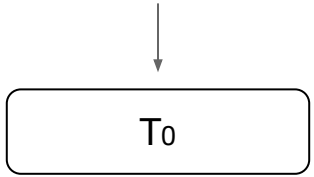
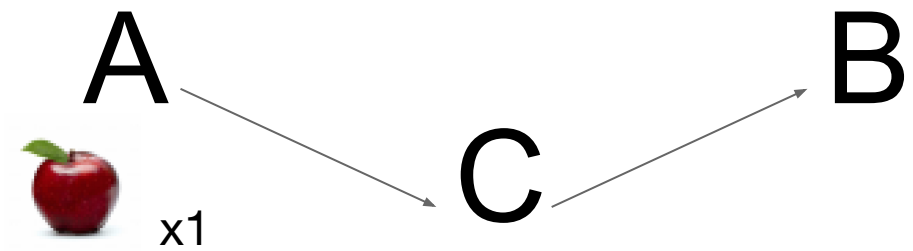
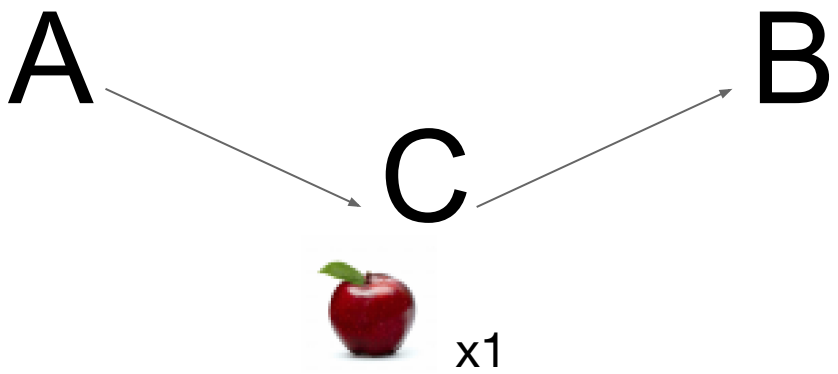


Abb. 1 - Vereinfachte Darstellung einer Transaktion in einem Ledger

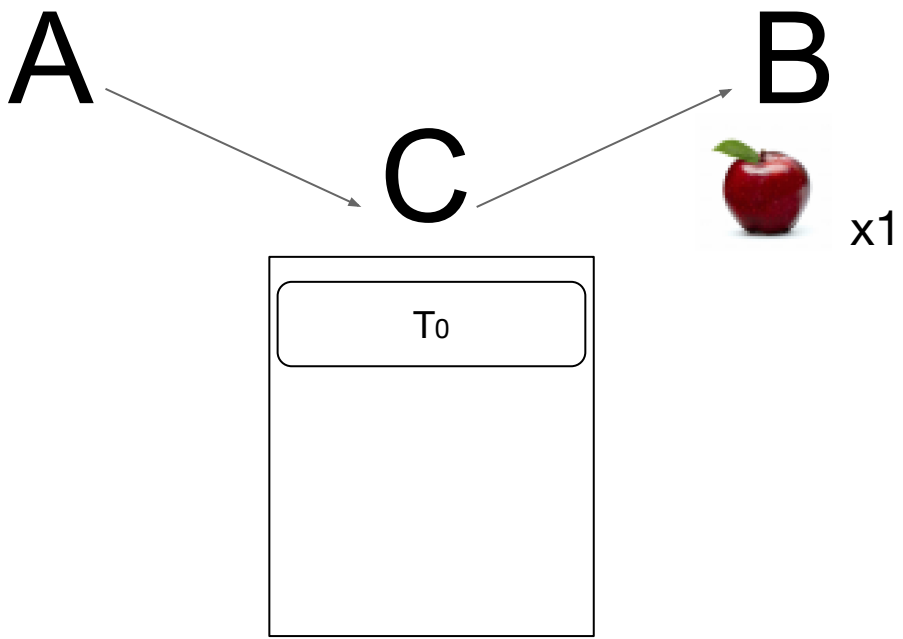
“Digital Ledger”



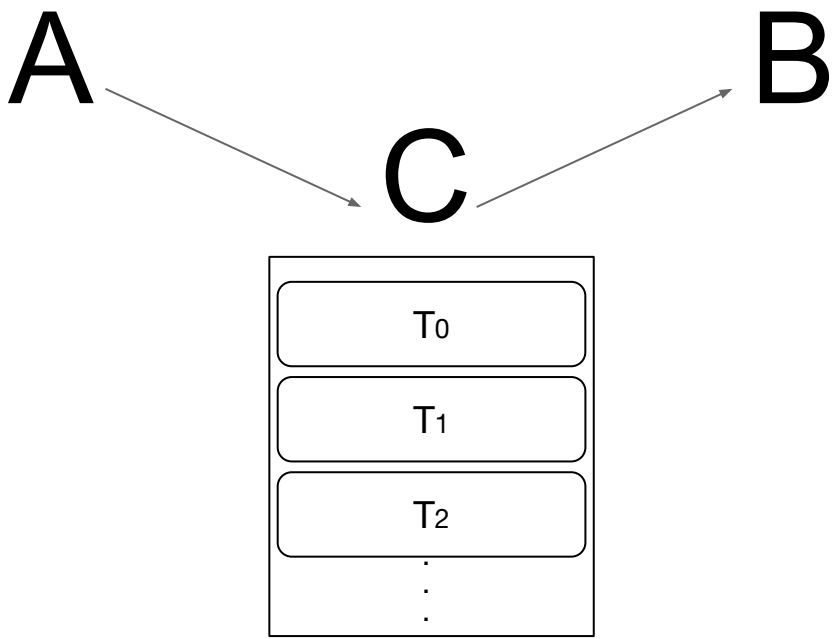
“Digital Ledger”



“Digital Ledger”

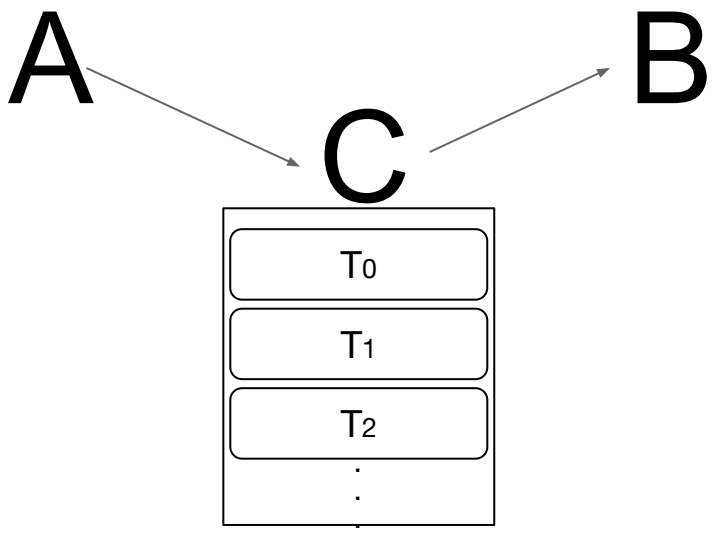


“Digital Ledger”



Aufgaben eines Ledgers

- Persistenz
- Validierung



Probleme heutiger Ledger

- Out-of-Sync
- Verfälschung
- Nicht transparent

Zentralisierte Lösung

- “Single point of failure”
- Nicht transparent
- Abhängigkeiten entstehen

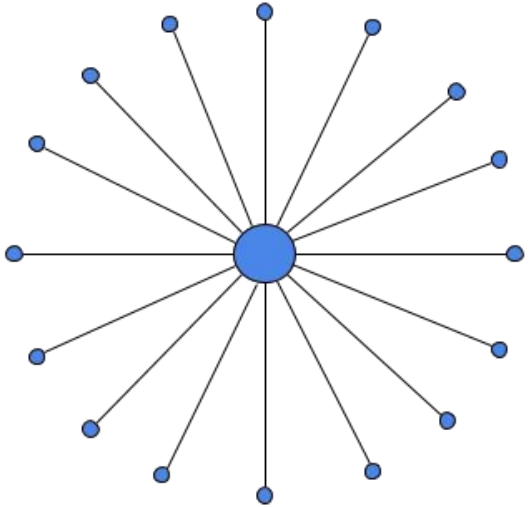
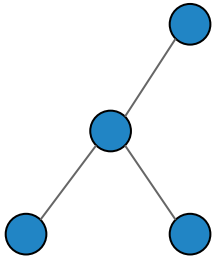


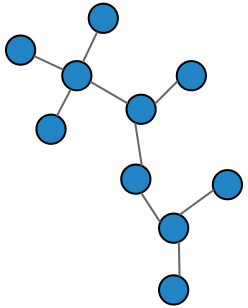
Abb. 2 - Visualisierung eines zentralisierten Netzwerks

Alternative Modelle

Zentralisiert



Dezentralisiert



Verteilt

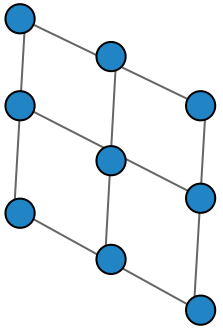


Abb. 3 - Visualisierung von zentralisierten, dezentralisierten und verteilten Netzwerken

Blockchain

“Distributed Digital Ledger”

Blockchain

- “Distributed Ledger Technology”
- Gemeinsame, öffentliche Datenbank
- Transaktionen über ein Peer-to-Peer-Netzwerk

Blockchain

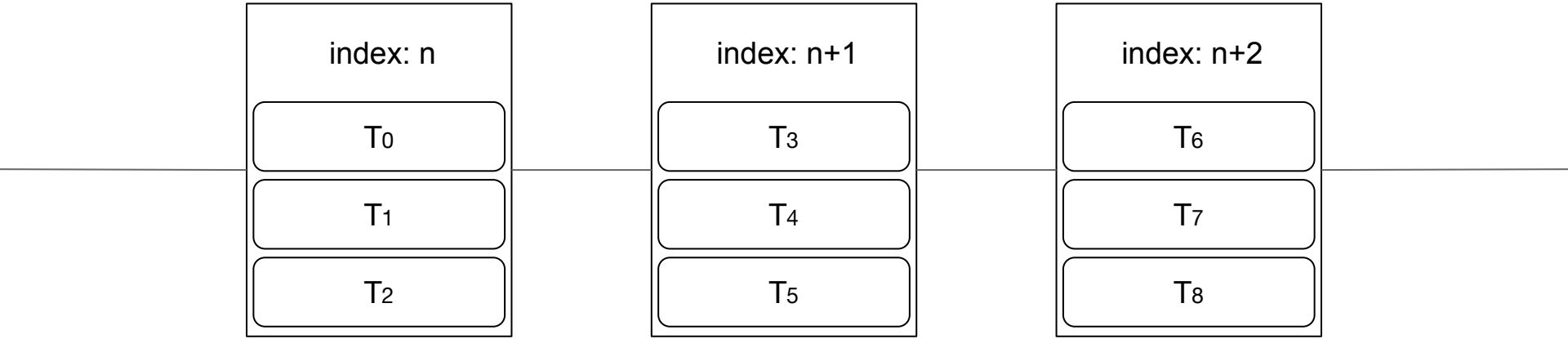


Abb. 4 - Modellierung einer simplen Blockchain

Funktionsweise

- Blockchain für jeden Knoten repliziert
- Mehrheit der Knoten bestimmt die Echtheit der Blockchain
- Fälschung durch einzelnen Knoten sehr unwahrscheinlich

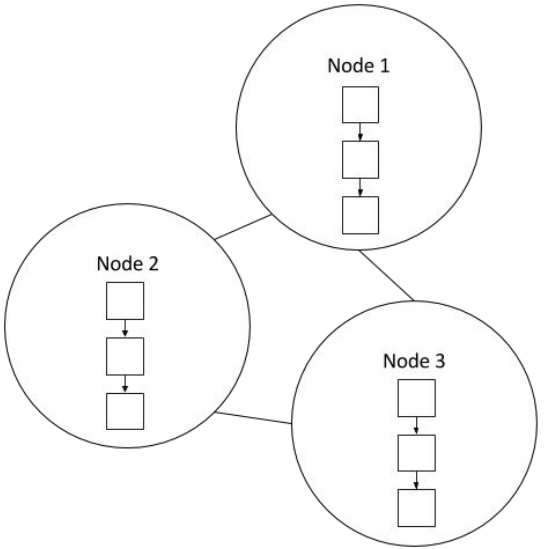


Abb. 5 - Replikation einer Blockchain auf mehreren Knoten [1]

Anforderungen an eine Blockchain

- Transparent
- Fälschungssicher
- Konsens ohne Vertrauen

Komponenten einer Blockchain

- Public Key-Kryptographie
- Hashes
- Merkle Trees
- Hash Puzzles

Wie wird eine Transaktion auf der Blockchain ausgeführt und validiert?

Autorisierung einer Transaktion

- Teilnehmer über Public Key bekannt
- Verifikation des Besitzes einer Adresse über Private Key

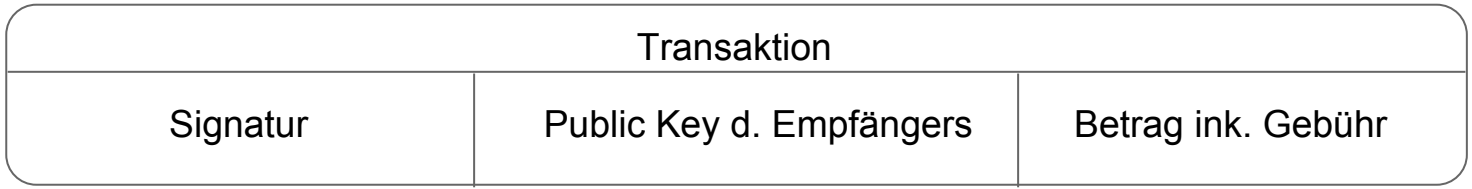


Abb. 6 - Komponenten einer einzelnen Transaktion auf einer Blockchain

Verifikation v. Transaktionen

- Knoten sendet Transaktion ins Netzwerk
- Mining-Knoten empfängt unbestätigte Transaktionen

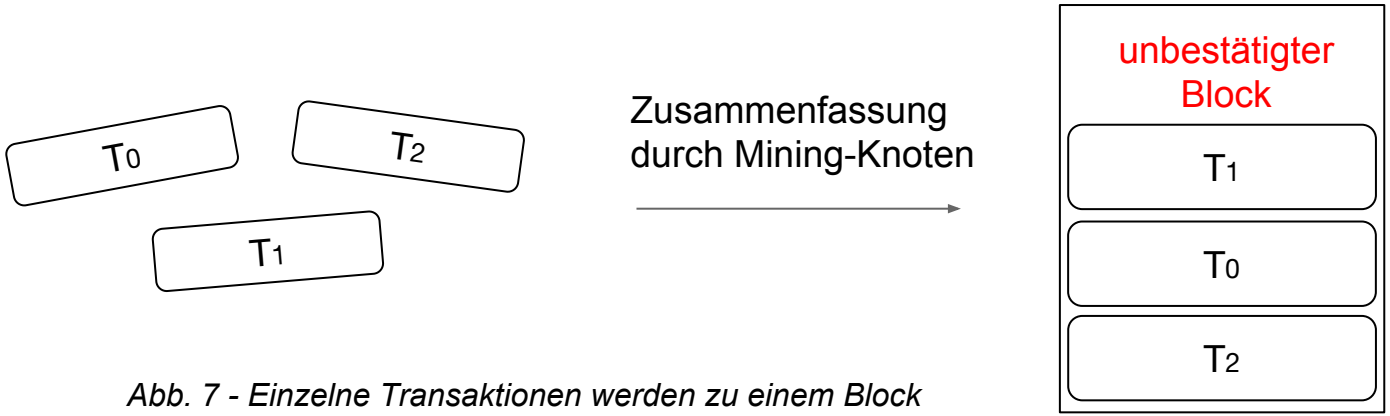


Abb. 7 - Einzelne Transaktionen werden zu einem Block zusammengefasst

Verifikation v. Blöcken

- Hashen der Transaktions-Information
- Bildung eines Root-Hashes
- Modifikationen wären im Hash sichtbar

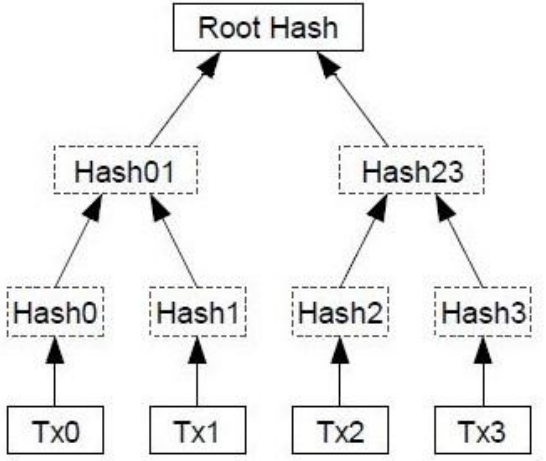


Abb. 8 - Erzeugung eines Root-Hashes in einem Merkle-Tree

Verifikation v. Blöcken

- Lösen eines Hash-Puzzles
- Result-Hash muss Vorgabe-Kriterium erfüllen
- Berechnen des Nonce-Wertes
- Prozess nennt man “Proof of Work”

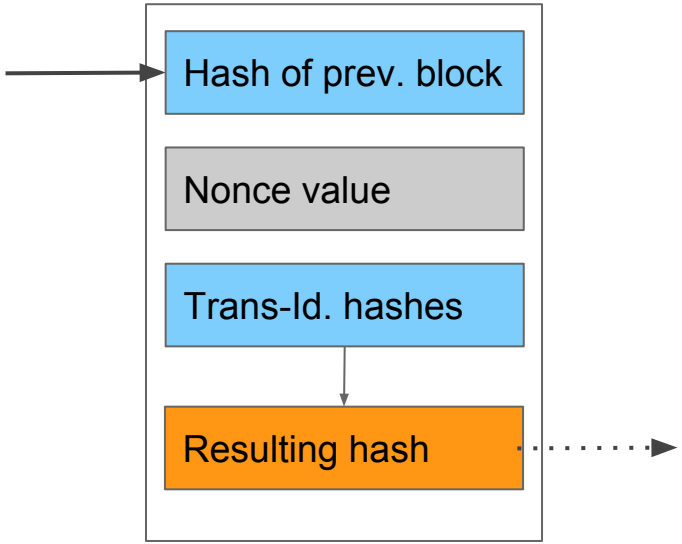


Abb. 9 - Ermittlung eines gültigen Block-Hashes über Vorgänger-, Transaktions-Hashes, sowie einem zu ermittelndem Nonce-Wert

Akzeptieren eines Blocks

- Ein neuer Block wird über das Netzwerk gesendet
- Knoten verifizieren diesen Block
- Erfolg: Neuer Block wird an die Kette gehängt
- Misserfolg: Block wird von Mehrheit der Knoten abgewiesen werden

Resultat

- Konsens durch mehrheitlich bestätigten Block
- Garantie für Sender und Empfänger
- Kein Vertrauen auf einzelne Knoten nötig

“Welche unterschiedlichen Arten v. Blockchains existieren?”

Blockchain-Modelle

Permissioned Blockchain

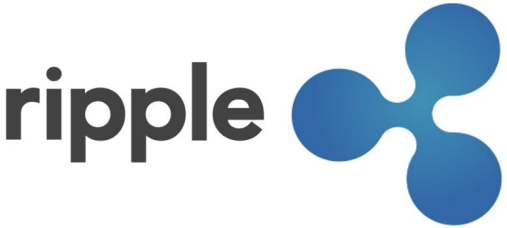
- Geschlossene Systeme
- Spezialisiert
- Optimiert
- Update-Prozess einfach

Permissionless Blockchain

- Öffentlich
- Hohe Reichweite
- Anonym
- Updates brauchen Konsens

“Was sind denkbare Anwendungsfälle?”

Finanz-Branche



Dezentrale Applikation

Anwendungsfälle

- Supply Chain-Tracking
- Asset Management
- Dezentrale Services (“Eliminate the middleman”)
- Smart Identity

“Welche Probleme existieren o. sind absehbar?”

Probleme v. Blockchains

Wartung

- Integration
- Updates
- Kosten

Sicherheit

- Anonymität
- “Fehler” nicht umkehrbar
- 51%-Angriffe

Tech. Limitierungen

- Datenmenge
- Skalierung

Motivation

Erfahrungen

- Bachelor: Keine
- Privat: Als Hobby verfolgt

Erwartungen

- Erfindung nicht umkehrbar
- Potenzial absehbar
- Wie wird sich diese Technologie niederschlagen?

Aktuelle Forschung

Aktuelle Forschung

1. S. Eskandari D. Barrera E. Stobert J. Clark
"A first look at the usability of bitcoin key management" Workshop on Usable Security (USEC)

2. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. 2016. Making Smart Contracts Smarter.
In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (CCS '16).
ACM, New York, NY, USA, 254-269. DOI:
https://doi.org/10.1145/2976749.2978309

Paper 1 - Thema

- **Thema:** Sicherheit und Usability in Public Key-Systemen

- **Punkte:**
  - **Abstrahierung d. Public-/Private-Key-Konzepts**
  - **Angriffe auf verfügbare Bitcoin-Clients**

Paper 1 - Methodik

- Kategorisierung existierender Angebote
- Angriffs-/Verlust-Szenarien
- “Cognitive Walkthrough” (Usability-Evaluation)

Paper 1 - Kategorisierung

Lokale Haltung

Passwort-Verschlüsselung

Externe Haltung

Password-derived Keys

Air-gapped Storage

Offline Storage

Paper 1 - Lokale Haltung

- Einfach
- Diebstahl (digital und physisch)
- Wartung (alternde Dateisysteme)
- Hardware-Versagen

Paper 1 - Verschlüsselte Key-Paare

- Schutz bei Verlust des Geräts
- Passwort-Ansatz verbreitet
- Passwort ohne Key nutzlos

Paper 1 - Offline-Wallet

- Permanent Offline?
- Kein digitaler Diebstahl
- Paper Wallets

Abb. 10 - Vorlage einer Bitcoin-Paper-Wallet [3]

Paper 1 - Air-gapped storage

- Signatur-Orakel
- Export v. Signaturen

Paper 1 - Password-derived Keys

- Passwort als Seed für Key-Paar
- Typische Passwort-Probleme
- Risiko-Verteilung

Paper 1 - Hosted Keys

- Widerspricht der Grundidee
- Usability
- Hybrid-Client:
  - Node extern bereitgestellt
  - Key-Management lokal

Paper 1 - Untersuchung

- Szenarien durchgespielt:
  - Initialisierung
  - Bezahlen
  - Wiederherstellung

Paper 1 - Evaluation

- Komplexe Begriffe
  - “No free outputs to spend”
- Abstrahierung
- Synchronitäts-Bedingung

Paper 1 - Fazit

- Public Key-Systeme sind nicht Mainstream
- Konzepte zu kommunizieren ist kritisch
- Online-Lösungen sehr benutzerfreundlich/unsicher

Paper 2 - Smart Contracts & Security

- Smart Contract
  - Bytecode auf der Blockchain
  - Durchsetzen von Vertragslogik ("Wenn..., dann…")
  - Unabhängige P2P-Economy

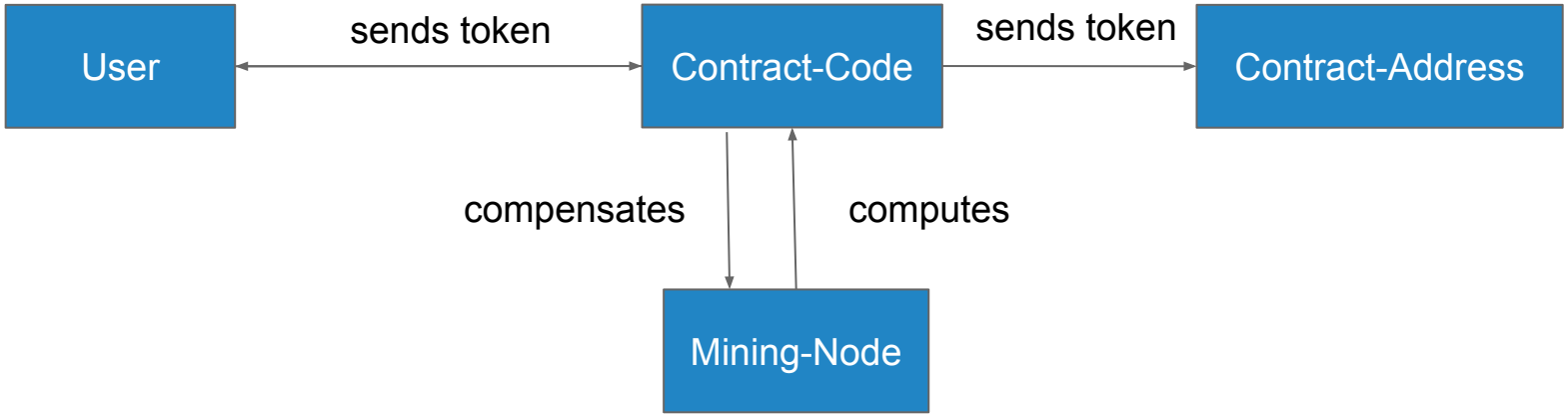
Paper 2 - Idee

- Smart Contracts nicht änderbar
- Bytecode öffentlich
- Gängige Schwachstellen?

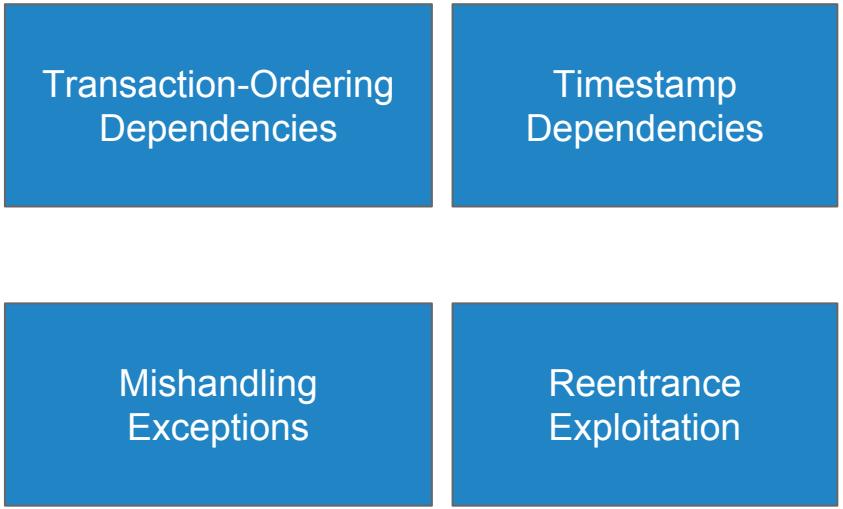
Paper 2 - Umgebung

- “Gas enabled transaction”
- Privater Speicher
- Contract-Adresse

Paper 2 - Smart Contract-Modell



Paper 2 - Schwachstellen



Paper 2 - Transaction-Order-Dependency

```
1 contract Puzzle{
2   address public owner;
3   bool public locked;
4   uint public reward;
5   bytes32 public diff;
6   bytes public solution;
7
8   function Puzzle() //constructor{
9     owner = msg.sender;
10    reward = msg.value;
11    locked = false;
12    diff = bytes32(11111); //pre-defined difficulty
13  }
14
```

Code-Snippet 1.1 - Solidity Smart-Contract zum Berechnen der Lösung eines Hash-Puzzles

Paper 2 - Transaction-Order-Dependency

```
15  function(){ //main code, runs at every invocation
16    if (msg.sender == owner){ //update reward
17      if (locked)
18        throw;
19      owner.send(reward);
20      reward = msg.value;
21    }
22    else
23      if (msg.data.length > 0){ //submit a solution
24        if (locked) throw;
25        if (sha256(msg.data) < diff){
26          msg.sender.send(reward); //send reward
27          solution = msg.data;
28          locked = true;
29        }}}}
```

Code-Snippet 1.2 - Solidity Smart-Contract zum Berechnen der Lösung eines Hash-Puzzles

Paper 2 - Timestamp-Dependency

```
1 contract theRun {
2   uint private Last_Payout = 0;
3   uint256 salt =  block.timestamp;
4   function random returns (uint256 result){
5     uint256 y = salt * block.number/(salt%5);
6     uint256 seed = block.number/3 + (salt%300)
7                    + Last_Payout +y;
8     //h = the blockhash of the seed-th last block
9     uint256 h = uint256(block.blockhash(seed));
10    //random number between 1 and 100
11    return uint256(h % 100) + 1;
12  }}
```

Code-Snippet 2 - Verwendung eines Block-Zeitstempels als Seed

Paper 2 - Timestamp-Dependency

```
1 contract theRun {
2   uint private Last_Payout = 0;
3   uint256 salt =  block.timestamp;
4   function random returns (uint256 result){
5     uint256 y = salt * block.number/(salt%5);
6     uint256 seed = block.number/3 + (salt%300)
7                    + Last_Payout +y;
8     //h = the blockhash of the seed-th last block
9     uint256 h = uint256(block.blockhash(seed));
10    //random number between 1 and 100
11    return uint256(h % 100) + 1;
12  }}
```

Code-Snippet 2 - Verwendung eines Block-Zeitstempels als Seed

Paper 2 - Timestamp-Dependency

- Zeitstempel eines Blocks vom Mining-Knoten gesetzt
- Manipulation durch Precomputing

Paper 2 - Exception Handling

- Häufige Fehlerquellen:
  - Callstack-Depth-Limit erreicht
  - Unzureichende Gebühren

Paper 2 - Exception Handling

```
1 contract KingOfTheEtherThrone {
2   struct Monarch {
3     // address of the king.
4     address ethAddr;
5     string name;
6     // how much he pays to previous king
7     uint claimPrice;
8     uint coronationTimestamp;
9     }
10  Monarch public currentMonarch;
11  // claim the throne
12  function claimThrone(string name) {
13    /.../
14    if (currentMonarch.ethAddr != wizardAddress)
15      currentMonarch.ethAddr.send(compensation);
16    /.../
17    // assign the new king
18    currentMonarch = Monarch(
19        msg.sender, name,
20        valuePaid, block.timestamp);
21  }}
```

Code-Snippet 3 - Verwendung eines Block-Zeitstempels als Seed

Paper 2 - Reentrancy Vulnerability

```
1 contract SendBalance {
2  mapping (address => uint) userBalances;
3  bool withdrawn = false;
4  function getBalance(address u) constant returns(uint){
5   return userBalances[u];
6  }
7  function addToBalance() {
8   userBalances[msg.sender] += msg.value;
9  }
10 function withdrawBalance(){
11  if (!(msg.sender.call.value(
12    userBalances[msg.sender])())) { throw; }
13  userBalances[msg.sender] = 0;
14  }}
```

Code-Snippet 4 - Verwendung eines Block-Zeitstempels als Seed

Paper 2 - Quantitative Analyse

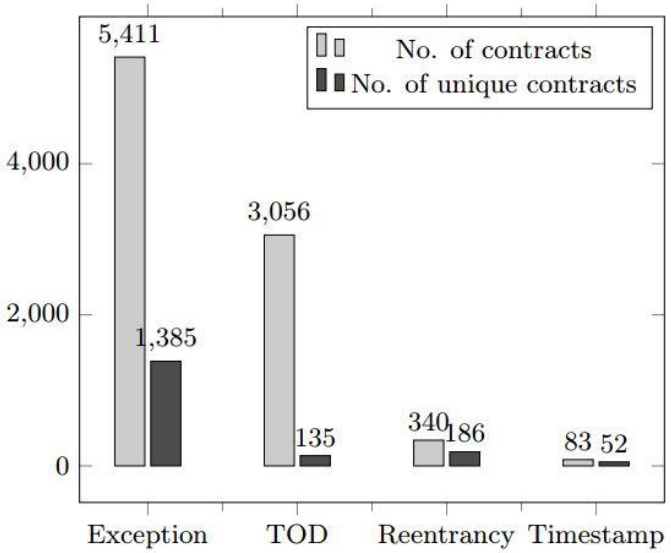


Abb. 11 - Quantitative Analyse von Sicherheitslücken in Bytecode von Ethereum-Contracts

Paper 2 - Quantitative Analyse

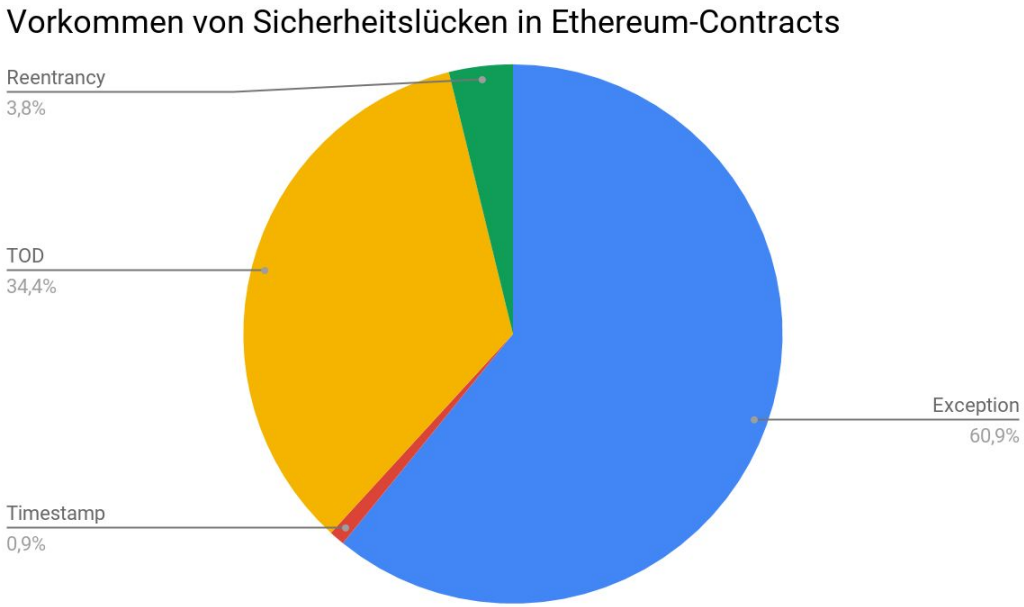


Abb. 12 - Quantitative Analyse von Sicherheitslücken in Bytecode von Ethereum-Contracts

Paper 2 - Vorschläge

- Solidity-Updates
- Pre-Deployment-Tool
  - Best-Practice durchsetzen

Konferenzen

- BCC 2017, 2 - 6 April, Abu Dhabi, UAE
  http://asiaccs2017.com/
- IEEE Blockchain Summit, 4 April, Vancouver, CA
  http://blockchain.ieee.org/2017-blockchain-summit/
- Annual Blockchain Conference, July 28, Washington, DC, USA
  https://www.blockchaindc.com
- Blockchain & BTC Conference, September 7, Stockholm, Sweden
  https://stockholm.blockchainconf.world/en

Blockchain im Master

- **Grundseminar:**
  Kartographieren des aktuellen Stands

- **Grundprojekt-Ideen:**
  Software Engineering für öff. Blockchains und Contracts, Alternative Verifizierung von Blöcken

- **Hauptseminar/-projekt:** ?

Referenzen

- https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Fachartikel/2016/fa_bj_1602_blockchain_en.html (Aufruf: 04.06.2017)
- https://ronanquillevere.github.io/2016/06/23/blockchain.html (Aufruf: 04.06.2017)
- https://www.cs.helsinki.fi/u/jakangas/Teaching/DistSys/DistSys-08f-1.pdf (Aufruf: 11.06.2017)
- https://www.bafin.de/SharedDocs/Veroeffentlichungen/EN/Fachartikel/2016/fa_bj_1602_blockchain_en.html (Aufruf: 11.06.2017)

Weitere Paper

- Underwood, Sarah. "Blockchain beyond bitcoin." *Communications of the ACM* 59.11 (2016): 15-17.

- Yli-Huumo, Jesse, et al. "Where Is Current Research on Blockchain Technology?—A Systematic Review." *PloS one* 11.10 (2016): e0163477.

Abbildungen

- https://ronanquillevere.github.io/2016/06/23/blockchain.html
  Abb. 5 - Replikation (Aufruf 04.06.2017)

- https://chrispacia.wordpress.com/2013/09/02/bitcoin-mining-explained-like-youre-five-part-2-mechanics/
  Abb.8 - Root-Hash in einem Merkle-Tree (Aufruf 10.06.2017)

- https://bitcoinpaperwallet.com/
  Abb.10 - Bitcoin Paper-Wallet (Aufruf 14.06.2017)