Blockchain, Smart Contract, Distributed Ledger Technology, Technische Grundlagen

Vorlesungsreihe Critical Legal Tech, Universität Luzern, 30. Oktober 2018

Jörn Erbguth, Consultant Legal Tech, Blockchain, Smart Contracts und Datenschutz

joern@erbguth.ch +41 787256027

KI & Deep Learning vs. Blockchain & Smart Contracts

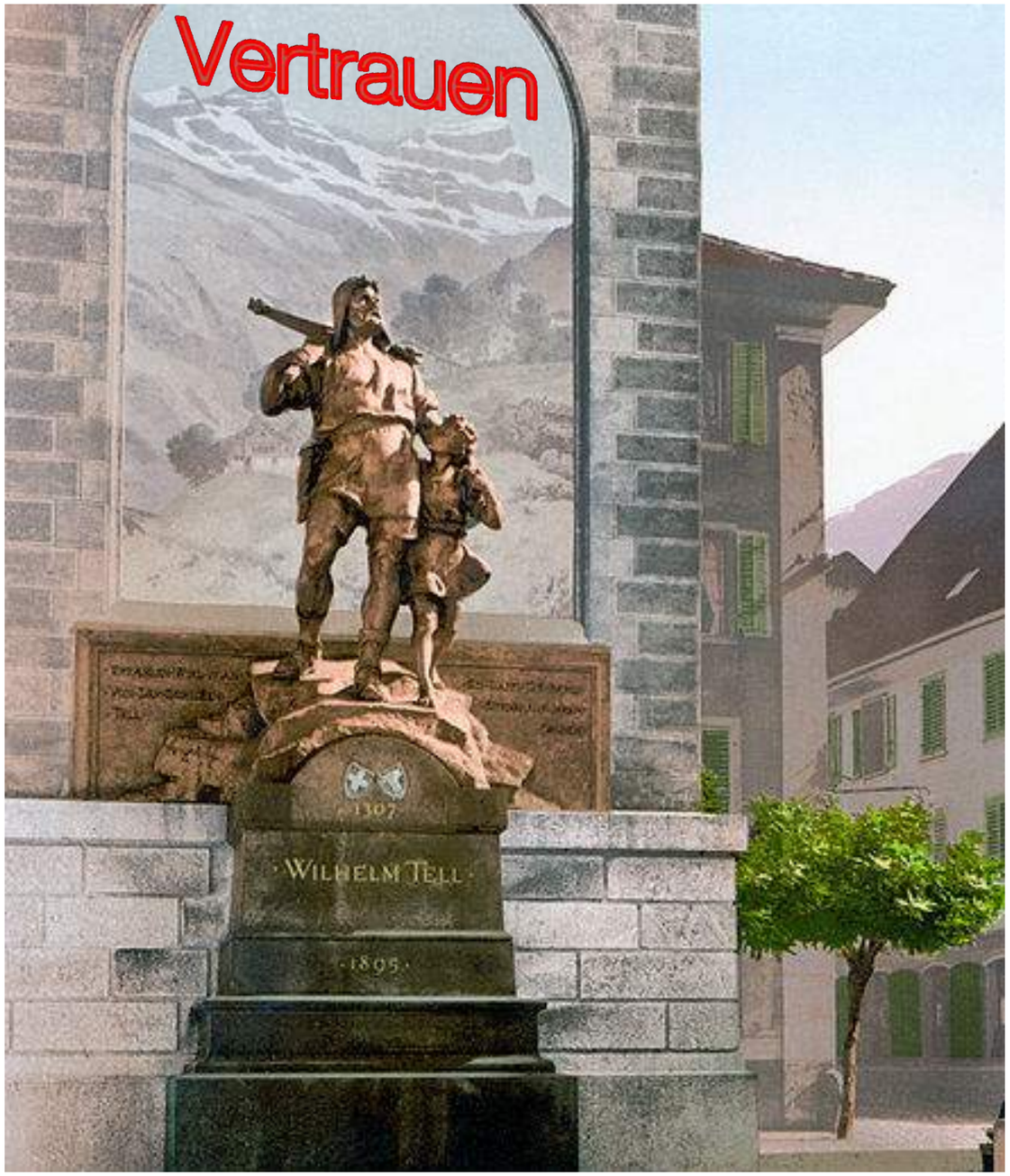
§§

Künstliche Intelligenz
Deep Learning

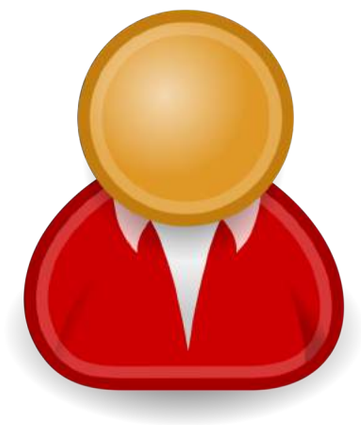
<code>

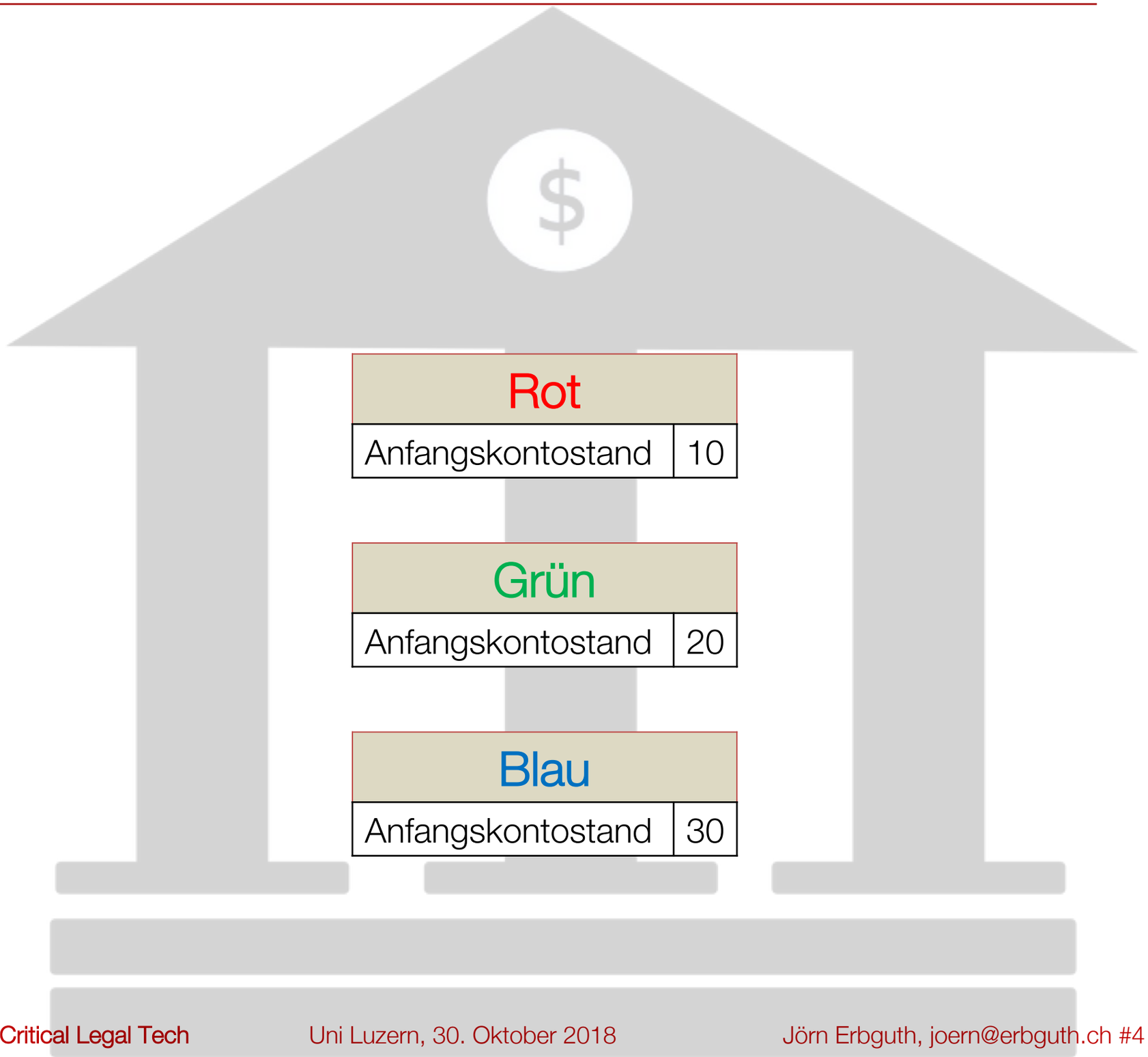
Blockchain
Smart Contracts

Warum Blockchain?

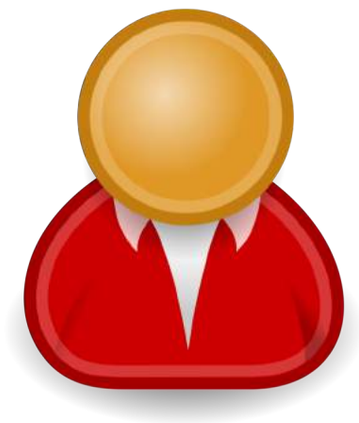


Zentralisierte Bank





Zentralisierte Bank



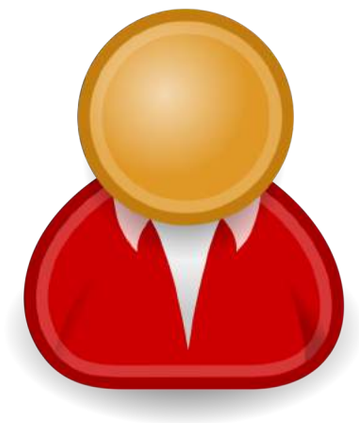
Transaktion:
5 Coins von Blau nach Rot

Rot	
Anfangskontostand	10
5 Coins von Blau	15

Grün	
Anfangskontostand	20

Blau	
Anfangskontostand	30
5 Coins zu Rot	25

Zentralisierte Bank



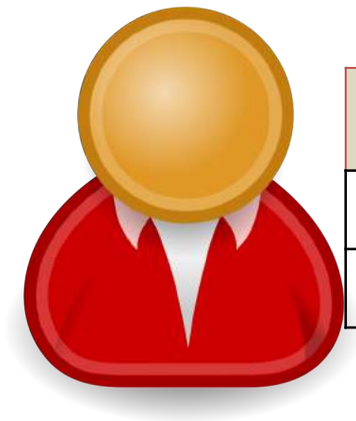
Transaktion:
20 Coins von Grün nach Blau

Rot	
Anfangskontostand	10
5 Coins von Blau	15

Grün	
Anfangskontostand	20
20 Coins zu Blau	0

Blau	
Anfangskontostand	30
5 Coins zu Rot	25
20 Coins von Grün	45

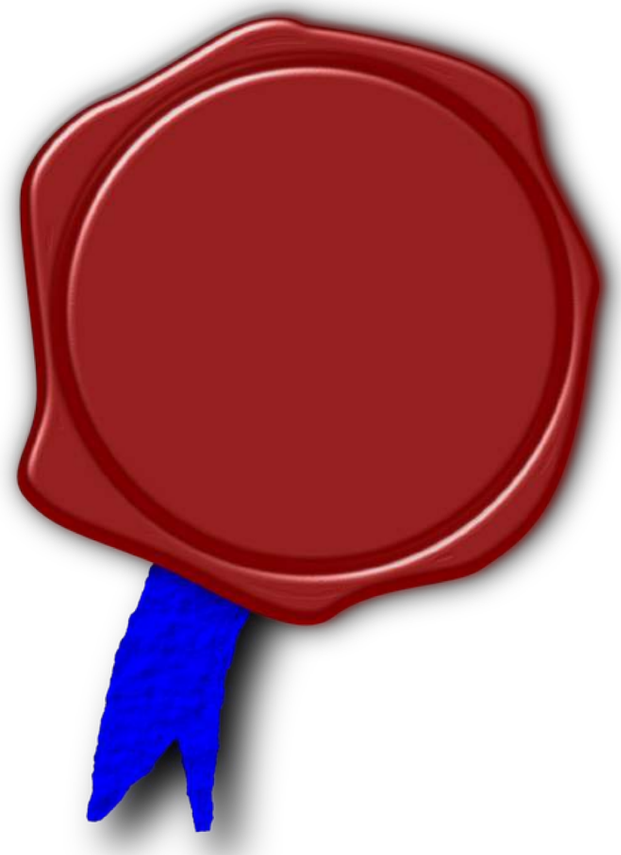
Wegfall der Zentrale



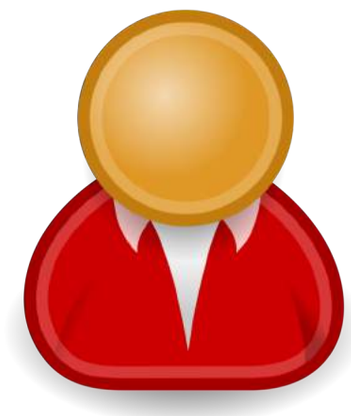
Rot	
Anfangskontostand	10
5 Coins von Blau	15

Grün	
Anfangskontostand	20
20 Coins zu Blau	0

Blau	
Anfangskontostand	30
5 Coins zu Rot	25
20 Coins von Grün	45



Jeder muss alle Transaktionen kennen



Ledger	
10	Anfangskontostand Rot
20	Anfangskontostand Grün
30	Anfangskontostand Blau
5	Coins von Blau zu Rot
20	Coins von Grün zu Blau

Ledger	
10	Anfangskontostand Rot
20	Anfangskontostand Grün
30	Anfangskontostand Blau
5	Coins von Blau zu Rot
20	Coins von Grün zu Blau

Ledger	
10	Anfangskontostand Rot
20	Anfangskontostand Grün
30	Anfangskontostand Blau
5	Coins von Blau zu Rot
20	Coins von Grün zu Blau

- Alle müssen die gleiche Transaktionsliste haben
- Die Liste muss gegen nachträgliches Entfernen oder Zurücksetzen geschützt sein

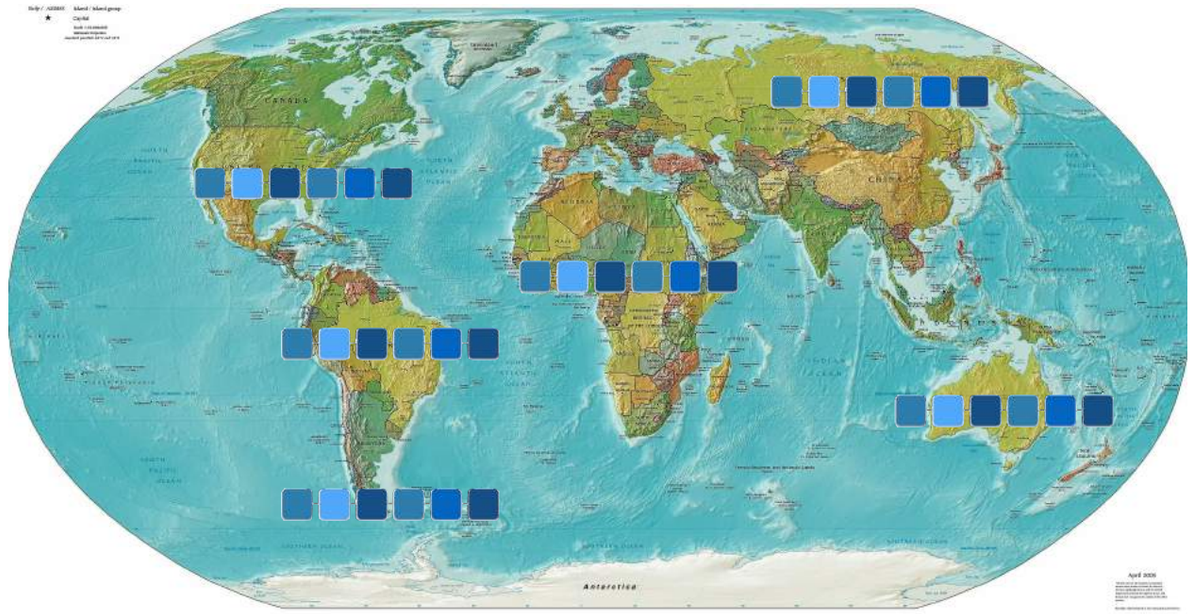
Kryptogeld sind keine Forderungen

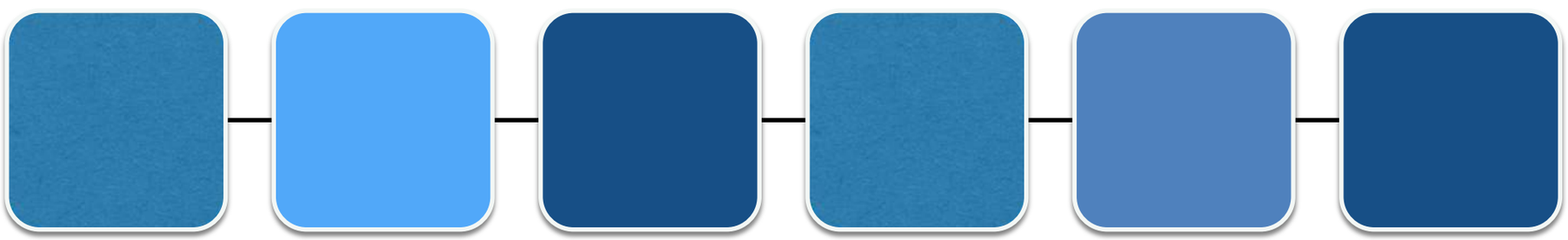
- Geld auf der Bank sind Forderungen gegen die Bank
- Bargeld ist aber keine Forderung gegen die Nationalbank
- Kryptogeld wie z.B. Bitcoin sind auch keine Forderungen

Daher reicht nicht der Nachweis, dass man einen Anspruch hat, sondern man hat das Kryptogeld nur dann, wenn man tatsächlich darüber verfügen kann – d.h. wenn man es weiter transferieren kann.

Bitcoin - Blockchain

- Ablage von Daten (Transaktionen)
- Zusammengefasst in Blöcken
- Unveränderbar
- Weltweit verteilt
- Regeln im Programmcode
- Regeln entscheiden über Zulässigkeit von Transaktionen





Wie stellen Blockchains Vertrauen her?

Kryptographie

+

Dezentralisierung

Hashfunktionen

+

Elektronische Signaturen

Kryptographische Hashfunktionen (1)

- Digitaler „Fingerabdruck“
- Praktisch eindeutig
- Gleiche Länge
- Für Objekte beliebiger Größe
- Kann (praktisch) nicht zurückgerechnet werden

Demo 2

Kryptographische Hash-Funktionen (2)

- “Rechtsanwalt“ ergibt
  13dd7a56194137a4aa844fb65bb119d82a401f2464c6c71cc34e762fa6350445
- „Rechtsanwalts“ ergibt
  c4a834b2cb0a7654efc03d37e1f6f3c1bf3d000a605b7be50001333e85b7a04d
- 
  ergibt
  c70032cb8270979b65ba543dd0e97c68d15d7306e379f377c056fea10ef65175

[Demo 2]

Kryptographische Hashfunktionen (3)

Anwendungen

- Checksumme
- Existenzbeweis
- Zeitstempel
- Basisfunktion für Blockchains und andere kryptographische Anwendungen

Demo 2

Hashfunktionen zur Verkettung von Blöcken

Wenn man einen Block ändert, muss man alle nachfolgenden Blöcke ebenfalls anpassen, damit die Kette gültig bleibt

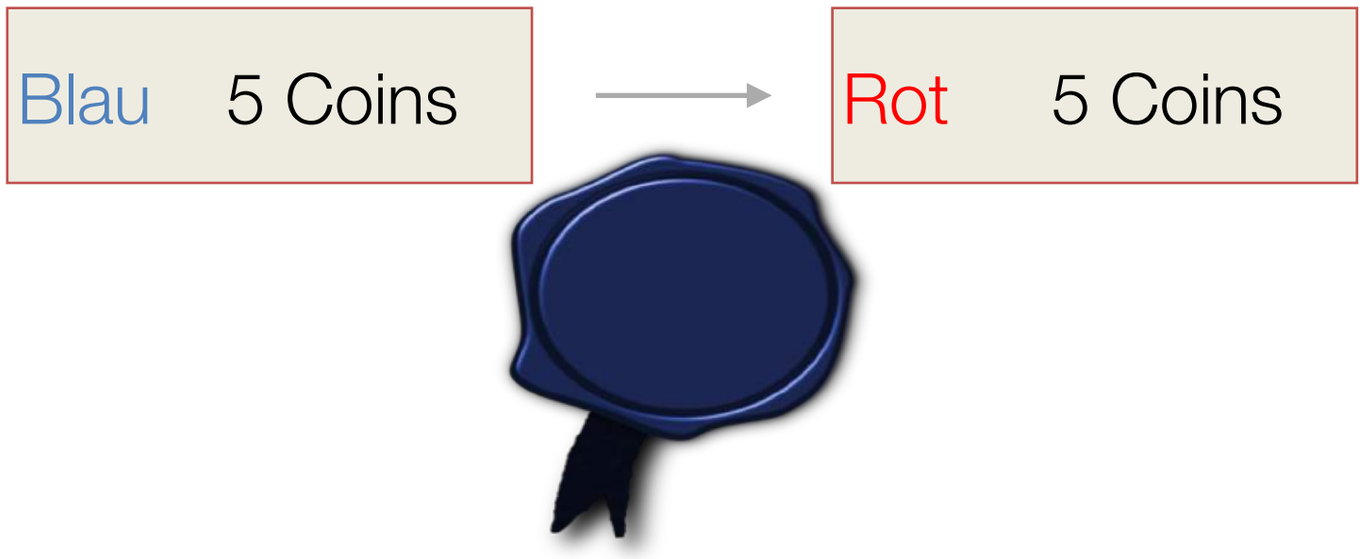
Ablauf einer Transaktion

1. Signatur der Transaktion

Ledger	
10	Anfangskontostand Rot
20	Anfangskontostand Grün
30	Anfangskontostand Blau
5	Coins von Blau zu Rot
20	Coins von Grün zu Blau

Ablauf einer Transaktion

1. Signatur der Transaktion



Von der Verschlüsselung zur elektronischen Signatur

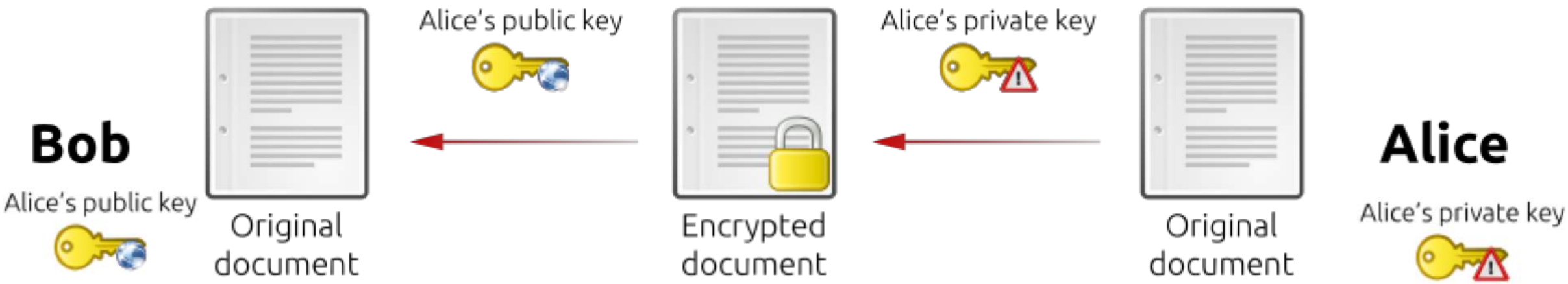
Symmetrische Verschlüsselung

Asymmetrische Verschlüsselung

Von der Verschlüsselung zur elektronischen Signatur

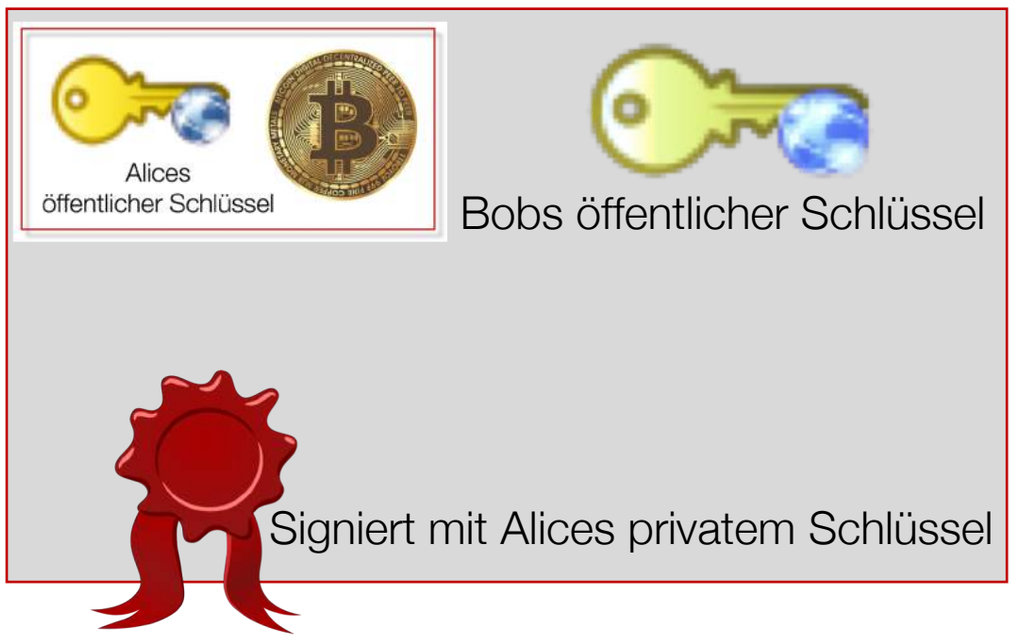
Asymmetrische Verschlüsselung

Elektronische Signatur

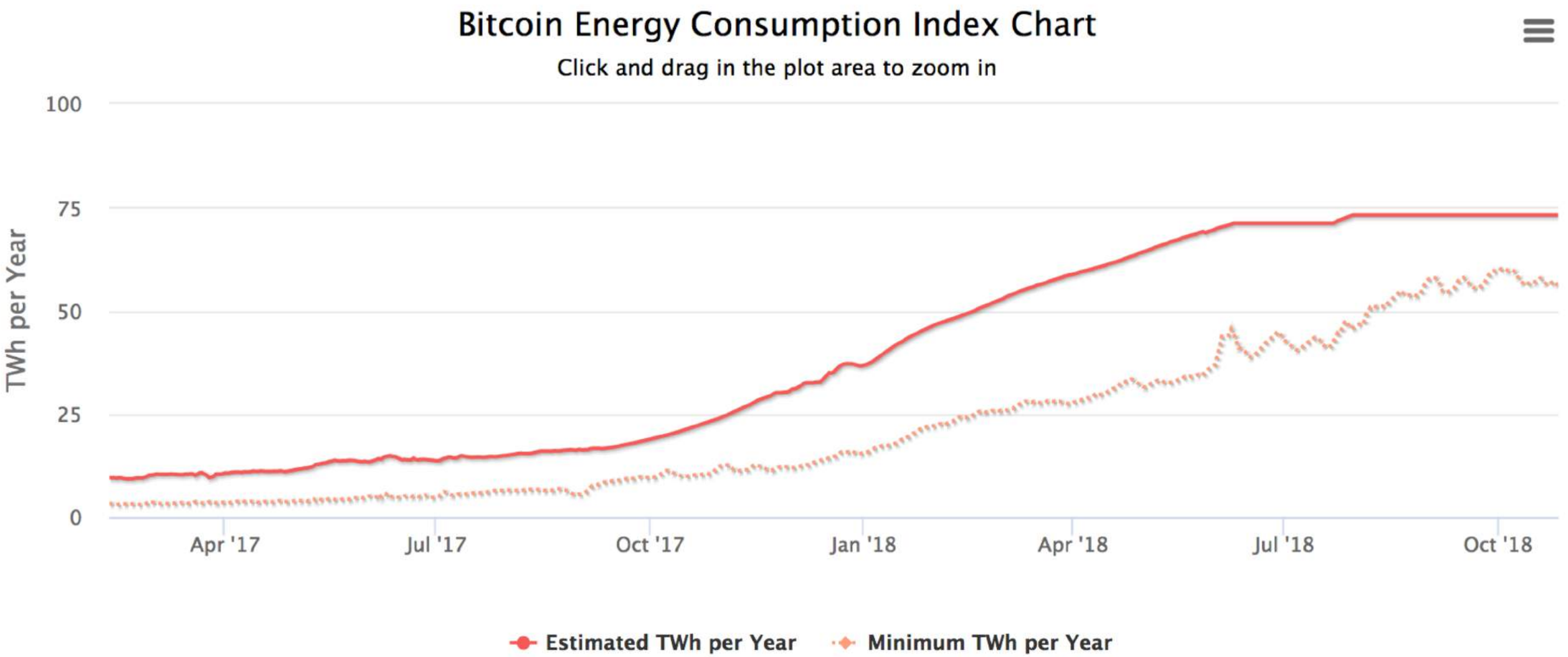


Nutzung der elektronischen Signatur in Blockchains

- Assets sind mindestens einem öffentlichen Schlüssel zugeordnet
- Assets werden bei einer Transaktion einem anderen öffentlichen Schlüssel zugeordnet
- Transaktionen müssen mit dem privaten Schlüssel des bisherigen öffentlichen Schlüssel elektronisch signiert werden



Energieverschwendung des Proof of Work



Source: Digiconomist

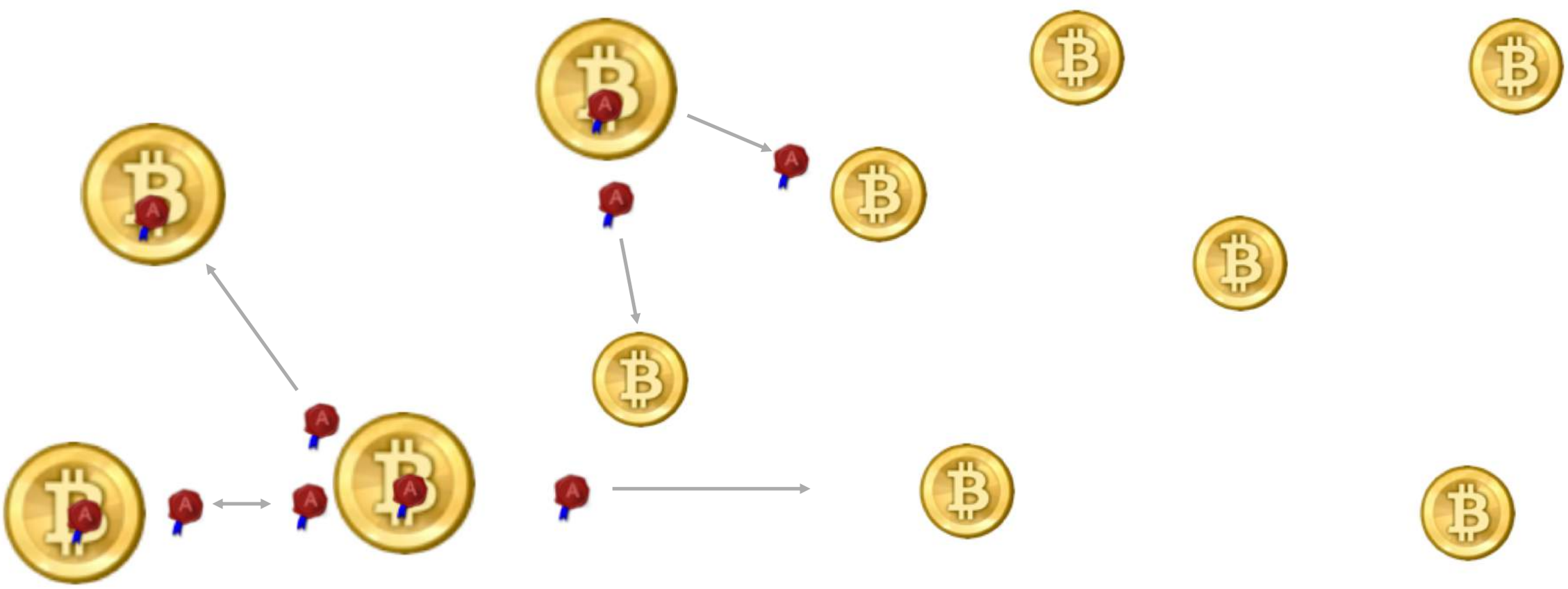
Keine Transaktion soll entfernt werden können

Keine Transaktion soll entfernt werden können

- Verkettung bedeutet, dass bei einer Änderung alle Folgeblöcke angepasst werden müssen
- Um das Nachberechnen vieler Blöcke zu verhindern, muss das Erstellen neuer Blöcke schwierig sein

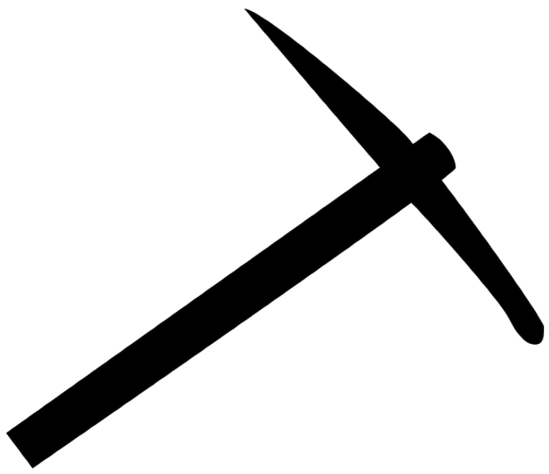
Ablauf einer Transaktion

1. Signieren der Transaktion
2. Verteilung an die Mining-Knoten

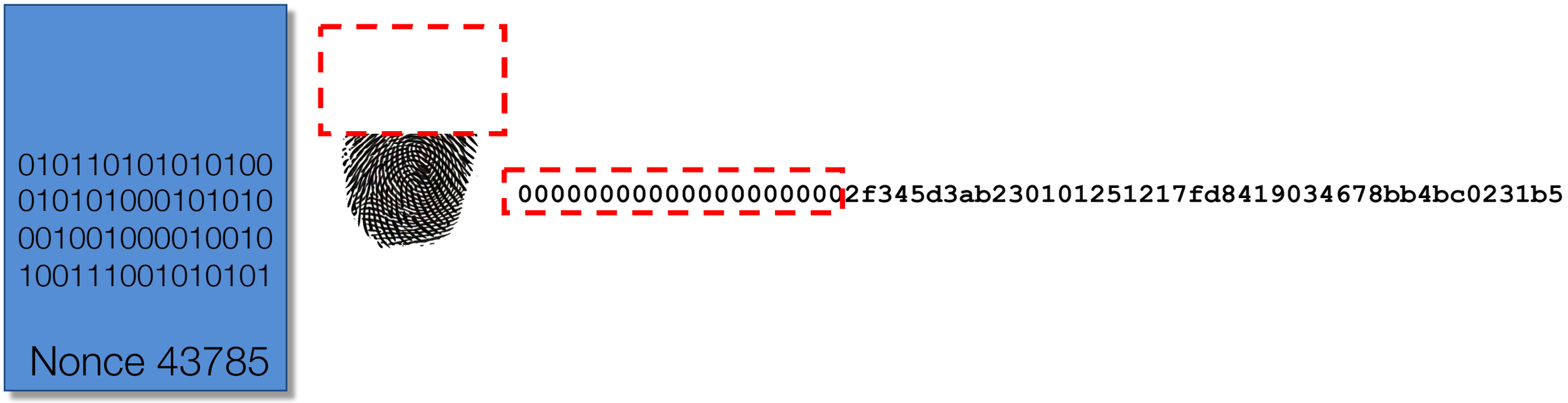


Ablauf einer Transaktion

1. Signieren der Transaktion
2. Verteilung an die Mining-Knoten
3. Jeder Mineur sammelt Transaktionen für einen neuen Block
4. Jeder Mineur versucht einen passenden „Nonce“-Wert zu finden

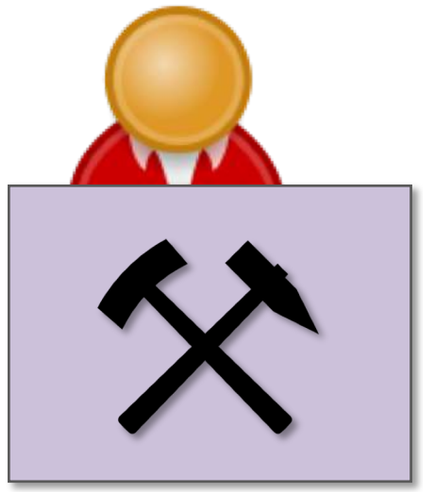


Mining

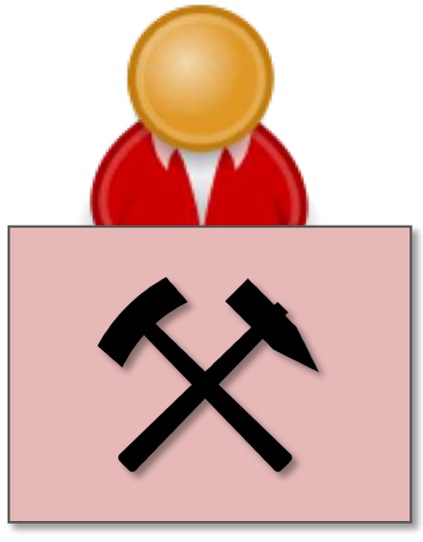


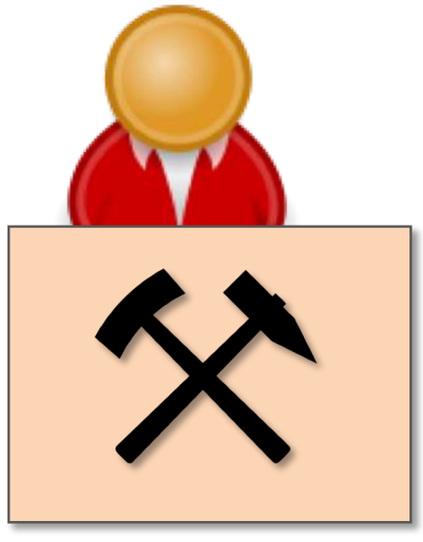
Ein neuer Block erfordert Zeit und Rechenleistung

Mining – Proof of Work

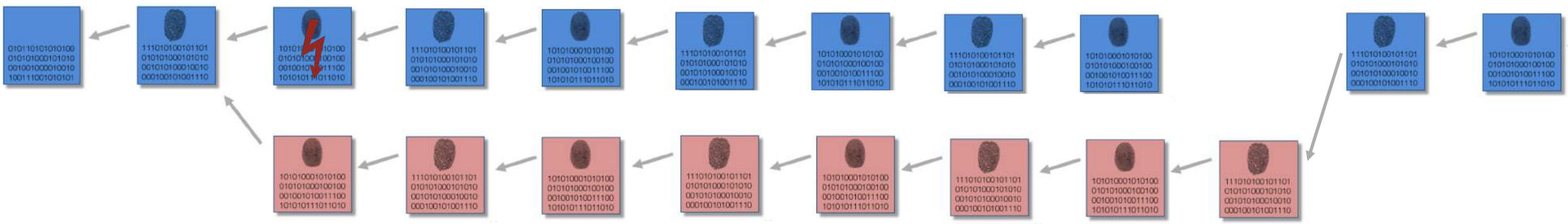


Transaktion Tra
nsaktion Transa
ktion Tr
n Trans





51% Attacke

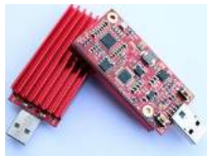


Wer Transaktionen aus einem alten Block entfernen will, muss

- alle nachfolgenden Blöcke ändern und
- dann die übrigen Mineure überholen, bis er die längere Kette hat.

Danach werden die anderen Mineure auf der längeren Blockchain weiterarbeiten.

Proof of Work – Abstimmung mit Rechenleistung



Proof of Stake – Abstimmung nach Anteilen






Proof of Authority – Abstimmung Auserwählter

Wallets - Geldbörsen

Verwahren der privaten Schlüssel

- Paper-Wallet

Wallets - Geldbörsen

Verwahren der privaten Schlüssel

- Paper-Wallet
- Software-Wallet
- Hardware-Wallet
- Wallet-Services

Smart Contracts

Smart Contracts aus technischer Sicht

- Programmierbare Blockchain
- Regeln sind programmierbar
- Gesetz = feststehende Regeln
- Vertrag = zwischen zwei Parteien definierbare Regeln

Aspekte der Smart Contracts auf einer Blockchain

1. Formulierung der Vertragsbedingungen als Computerprogramm

```
pragma solidity ^0.4.2;

contract Handel {
    string public Ware;
    uint public Preis;
    address public Verkaeufer;
    address public Kaeufer;
    bool public bezahlt;
    bool public geliefert;
    bool public abgewickelt;
    address constant post=0x1234567

function Angebot(string iWare, uint
{
    Verkaeufer=msg.sender;
    Ware=iWare;
    Preis=iPreis;
}

function Annahme() payable
{
    if(msg.value>=Preis)
    {
```

2. Automatische Vertragsausführung

3. Vereinbarung und sichere Ausführung auf einer Blockchain

Formulierung eines Vertrages als Computerprogramm (1)

- ✓ Berechnungen
- ✓ Abläufe
- ✓ Fristen
- ✓ Rechtsfolgen
- ✗ Unbestimmte Rechtsbegriffe (z.B. *Fahrlässigkeit)*

```
pragma solidity ^0.4.2;

contract Handel {
    string public Ware;
    uint public Preis;
    address public Verkaeufer;
    address public Kaeufer;
    bool public bezahlt;
    bool public geliefert;
    bool public abgewickelt;
    address constant post=0x1234567

function Angebot(string iWare, uint
{
    Verkaeufer=msg.sender;
    Ware=iWare;
    Preis=iPreis;
}

function Annahme() payable
{
    if(msg.value>=Preis)
    {
```

Formulierung eines Vertrages als Computerprogramm (2)

```
pragma solidity ^0.4.2;

contract Handel {
    string public Ware;
    uint public Preis;
    address public Verkaeufer;
    address public Kaeufer;
    bool public bezahlt;
    bool public geliefert;
    bool public abgewickelt;
    address constant post=0x1234567

function Angebot(string iWare, uint
{
    Verkaeufer=msg.sender;
    Ware=iWare;
    Preis=iPreis;
}

function Annahme() payable
{
    if(msg.value>=Preis)
    {
```

Rechtswirksamkeit ?

- B2B
- B2C
- Formerfordernisse (z.B. Schriftform)
- Akzeptanz vor Gericht

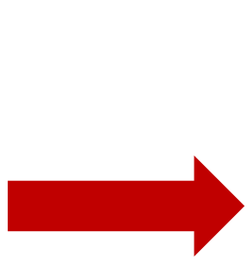
Formulierung eines Vertrages als Computerprogramm (3)

```
pragma solidity ^0.4.2;

contract Handel {
    string public Ware;
    uint public Preis;
    address public Verkaeufer;
    address public Kaeufer;
    bool public bezahlt;
    bool public geliefert;
    bool public abgewickelt;
    address constant post=0x1234567

function Angebot(string iWare, uint
{
    Verkaeufer=msg.sender;
    Ware=iWare;
    Preis=iPreis;
}

function Annahme() payable
{
    if(msg.value>=Preis)
    {
```

Programmcode ≠ juristischer Vertrag ≠ Vertrag in Textform

- Es gilt, was die Parteien wollten (Art. 18 OR)
- Zwingendes Recht
- Bugs

Formulierung eines Vertrages als Computerprogramm (4)



```
pragma solidity ^0.4.2;

contract Handel {
    string public Ware;
    uint public Preis;
    address public Verkaeufer;
    address public Kaeufer;
    bool public bezahlt;
    bool public geliefert;
    bool public abgewickelt;
    address constant post=0x1234567

function Angebot(string iWare, uint
{
    Verkaeufer=msg.sender;
    Ware=iWare;
    Preis=iPreis;
}

function Annahme() payable
{
    if(msg.value>=Preis)
    {
```

Vorteile

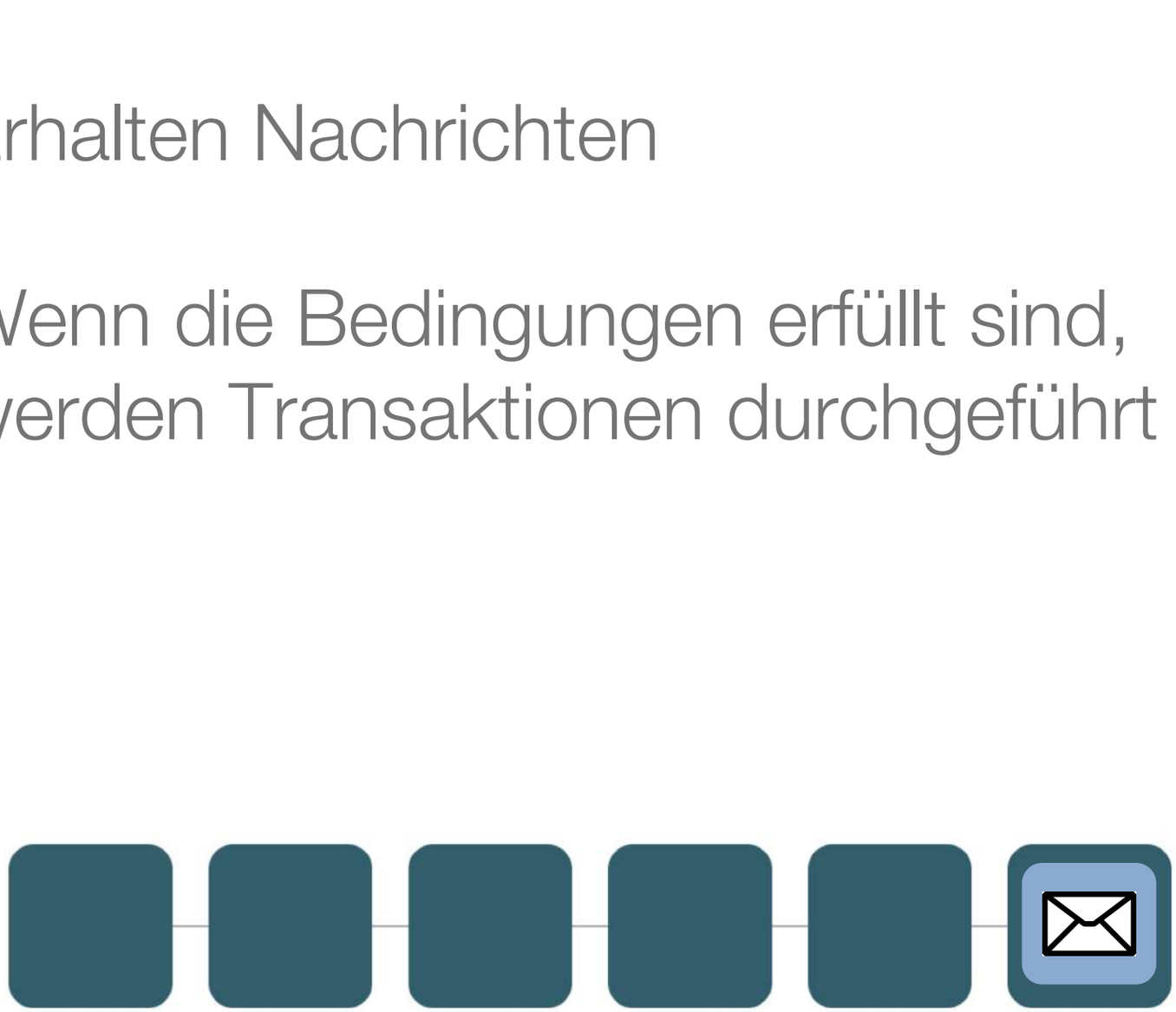
- Einsatz von Software-Entwicklungstools
- Automatisches Evaluieren einer grossen Anzahl von Verträgen
  - Unternehmensübernahmen / Due Diligence
  - Evaluierung der Auswirkungen von Urteilen oder Gesetzesvorhaben
  - Simulation von Handlungsoptionen

Automatische Vertragsausführung

- Jeder Verkaufsautomat führt automatisiert Verträge aus
- Programmierung verborgen
- Verkäufer kann Programm unbemerkt manipulieren
- Keine sichere Protokollierung der Transaktion

Smart Contracts auf der Blockchain

- Kleine Programme
- Erhalten Nachrichten
- Wenn die Bedingungen erfüllt sind, werden Transaktionen durchgeführt



Smart Contracts – kleine Programme auf der Blockchain

- Nicht jedes Smart Contract Programm auf der Blockchain hat etwas mit juristischen Verträgen zu tun

- Der Autor eines Smart Contract Programms ist häufig nicht Vertragspartner

- Ein Smart Contract Programm kann ggf. viele Verträge zwischen zwei oder mehr Parteien vermitteln

Smart Contracts – warum auf der Blockchain?

- Auf der Blockchain gespeichert
- Auf der Blockchain ausgeführt
- Transparent
- Manuell nicht beeinflussbar
- Agiert wie ein Treuhänder

Smart Contracts und Krypto-Währungen

Smart Contracts können

- Krypto-Geld erhalten
- Krypto-Geld halten
- Krypto-Geld transferieren

Smart contract – Beispiel Gebrauchtsoftwarehandel

Smart Contract
Entwickler

ABC AG
Ersterwerber

Software	XYZ Software V 1.2
Serial Number	123456789
Original Purchaser	ABC AG
Original Purchase Date	1.8.2017
Verification	1.2.2018
Offer	10 Ether
Acceptance	DEF AG
Date	1.5.2018
Offer	8 Ether
Acceptance	GHI AG
Date	1.8.2018

Lizenz
Verifikation

DEF AG
Zweiterwerber

GHI AG
Dritterwerber

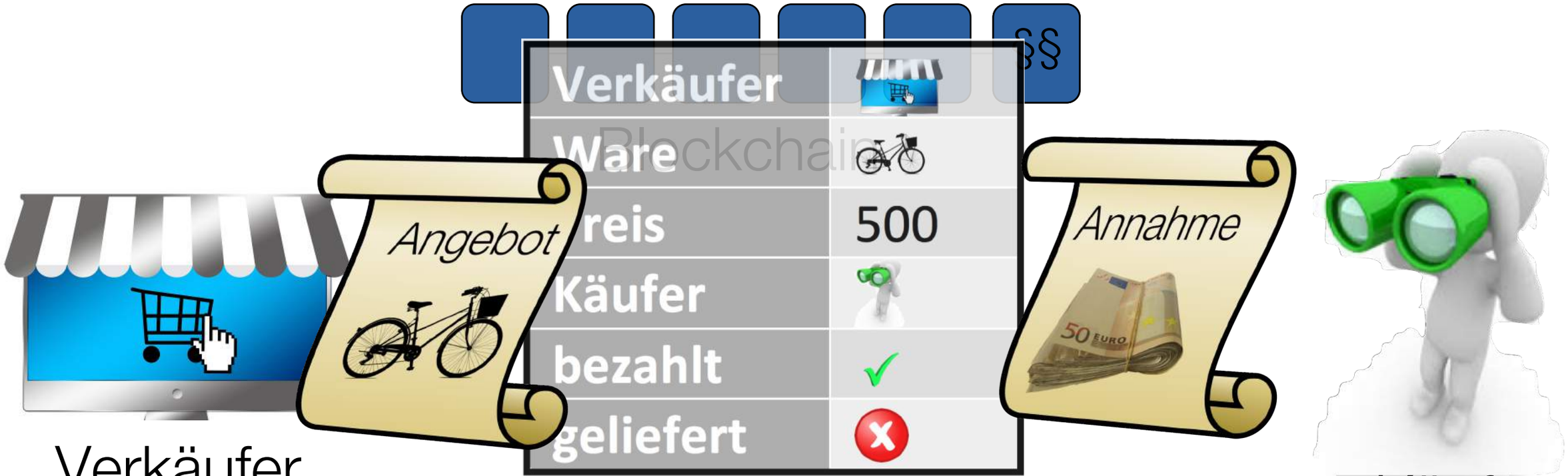
Smart Contract – Beispiel

Blockchain basierte Handelsplattform

- Transparente Regeln
- Komplett automatisiert
- Ohne manuelle Eingriffsmöglichkeit

Smart Contract - Beispiel Handelsplattform

Smart Contract
Entwickler



Verkäufer

Käufer

Smart Contract - Beispiel Code

```
1  pragma solidity ^0.4.2;
2
3  contract Handel {
4      string public Ware;
5      uint public Preis;
6      address public Verkaeufer;
7      address public Kaeufer;
8      bool public bezahlt;
9      bool public geliefert;
10     bool public abgewickelt;
11     address constant post=0x1234567890abcdef;
12
13 function Angebot(string iWare, uint iPreis)
14 {
15     Verkaeufer=msg.sender;
16     Ware=iWare;
17     Preis=iPreis;
18 }
19
20 function Annahme() payable
21 {
22     if(msg.value>=Preis)
23     {
24         Kaeufer=msg.sender;
25         bezahlt=true;
26     }
27 }
28
29 function Lieferung()
30 {
31     if(msg.sender==post && geliefert==false)
32     {
33         geliefert=true;
34         abgewickelt=Verkaeufer.send(Preis);
35     }
36 }
37 }
```

Vielen Dank für Ihre Aufmerksamkeit!

Themen in der nächsten Vorlesung (27.11.):

- Anwendungsbereiche von Blockchains
- Initial Coin Offerings (ICOs)
- Smart Contracts rechtlich gesehen
- Permissioned Blockchains, Sidechains
- Weiterentwicklungen z.B. Hashgraph, Sharding
- Governance und Dispute Resolution
- Datenschutz und Blockchains
- Rechtliche und gesellschaftliche Perspektiven der Distributed Ledger Technologie (DLT)