STROM DAO

Distributed ledger technology
An introduction

Stefan Thon

SHA256 Hash

A Hash is a fingerprint of some digital data. The same data always yields the same hash.

Data:

Hallo Leipzig

Hash:

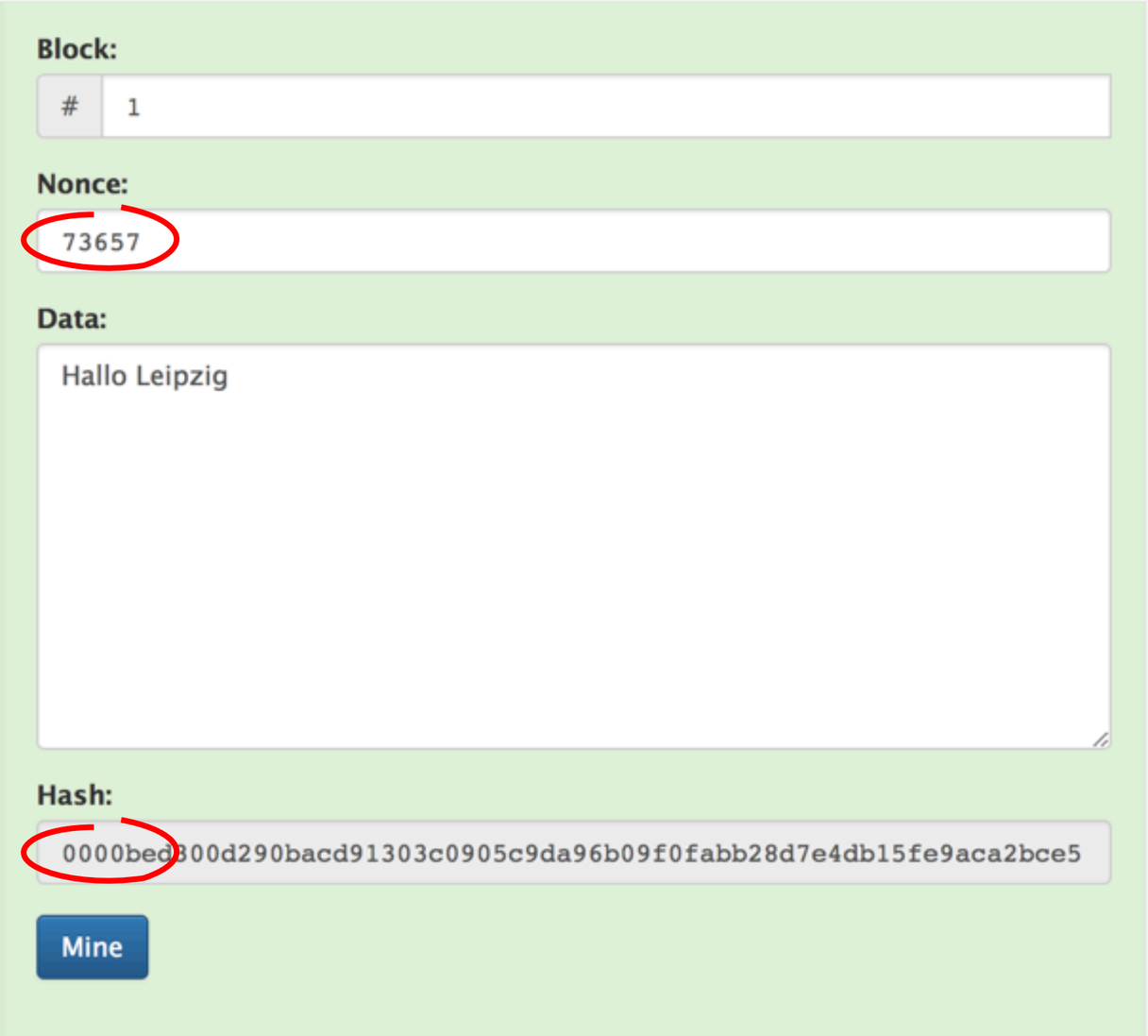
94b26fa1568455e3b35791344d46628947f02e9eba9c8439386033405a650a52

Block

A block is a structured set of digital data to be hashed. It contains a block number, a nonce and data.

There are requirements for a block hash to be valid. To find a valid hash the nonce must be computed.

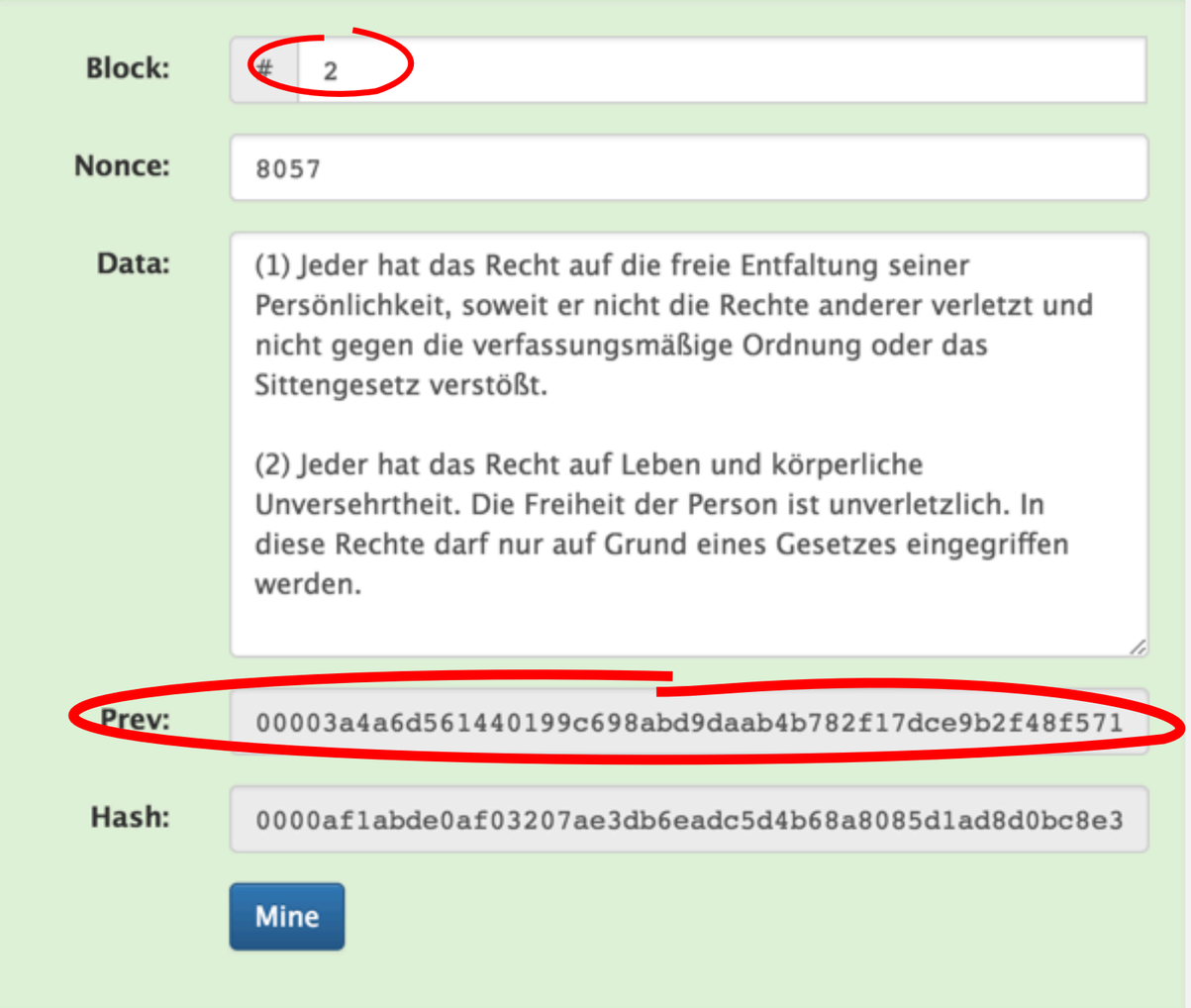
This process is called mining.



Block chain

A block chain is a series of blocks, whereby each block references the hash of the previous block.

This makes a block chains resist data mutations.



Distributed block chain

A block chain with many distributed copies.

This makes it easy to filter out invalid copies of the block chain. (The majority wins.)

Transactional data, not any kind of data!

A transaction is a transfer of value

A transferable value is an asset. A digitally transferable values is a Digital Asset.

Blockchain Technology is Value Technology rather than Information Technology.

$	10.00	From:	Anders	->	Sophia

A block is a list of transactions

There is a maximum number of transactions per block.

$	25.00	From:	Darcy	->	Bingle
$	4.27	From:	Elizab	->	Jane
$	19.22	From:	Wickha	->	Lydia
$	106.44	From:	Lady C	->	Collin
$	6.42	From:	Charlo	->	Elizab

A block chain consists of multiple lists (or blocks) of transaction

Block:	#	1				
Tx:	$	25.00	From:	Darcy	->	Bingley
	$	4.27	From:	Elizabeth	->	Jane
	$	19.22	From:	Wickham	->	Lydia
	$	106.44	From:	Lady Cather	->	Collins
	$	6.42	From:	Charlotte	->	Elizabeth

Block:	#	2				
Tx:	$	97.67	From:	Ripley	->	Lambert
	$	48.61	From:	Kane	->	Ash
	$	6.15	From:	Parker	->	Dallas
	$	10.44	From:	Hicks	->	Newt
	$	88.32	From:	Bishop	->	Burke
	$	45.00	From:	Hudson	->	Gorman
	$	92.00	From:	Vasquez	->	Apone

Chaining blocks together

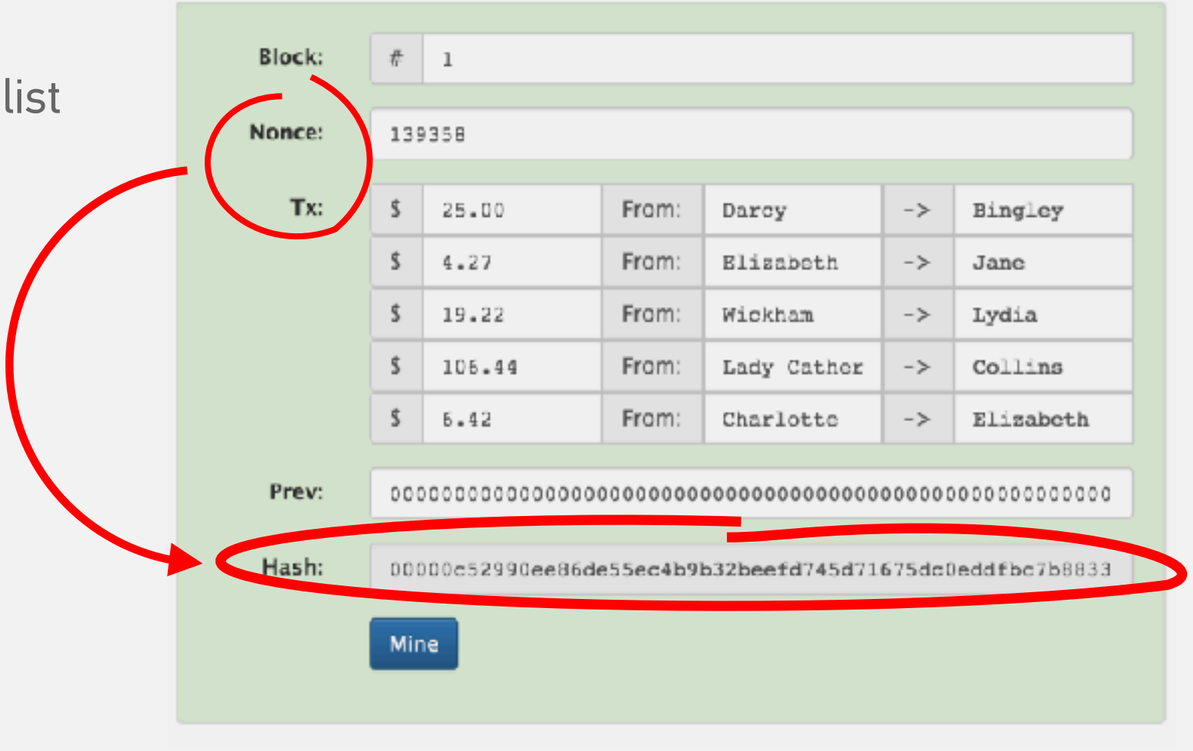
Transaction lists are chained together by each list referencing its previous list.

Block:	#	1			
Nonce:	139358				
Tx:	$	25.00	From:	Darcy	-> Bingley
	$	4.27	From:	Elizabeth	-> Jane
	$	19.22	From:	Wickham	-> Lydia
	$	106.44	From:	Lady Cather	-> Collins
	$	6.42	From:	Charlotte	-> Elizabeth
Prev:	0000000000000000000000000000000000000000000000000000000000000000				
Hash:	00000c52990ee86de55ec4b9b32beefd745d71675dc0eddfbc7b8833				

Block:	#	2			
Nonce:	39207				
Tx:	$	97.67	From:	Ripley	-> Lambert
	$	48.61	From:	Kane	-> Ash
	$	6.15	From:	Parker	-> Dallas
	$	10.44	From:	Hicks	-> Newt
	$	88.32	From:	Bishop	-> Burke
	$	45.00	From:	Hudson	-> Gorman
	$	92.00	From:	Vasquez	-> Apone
Prev:	00000c52990ee86de55ec4b9b32beefd745d71675dc0eddfbc7b8833				
Hash:	000078be183417844c14a9251ca246fb15df1074019873f5d85c1a6f				

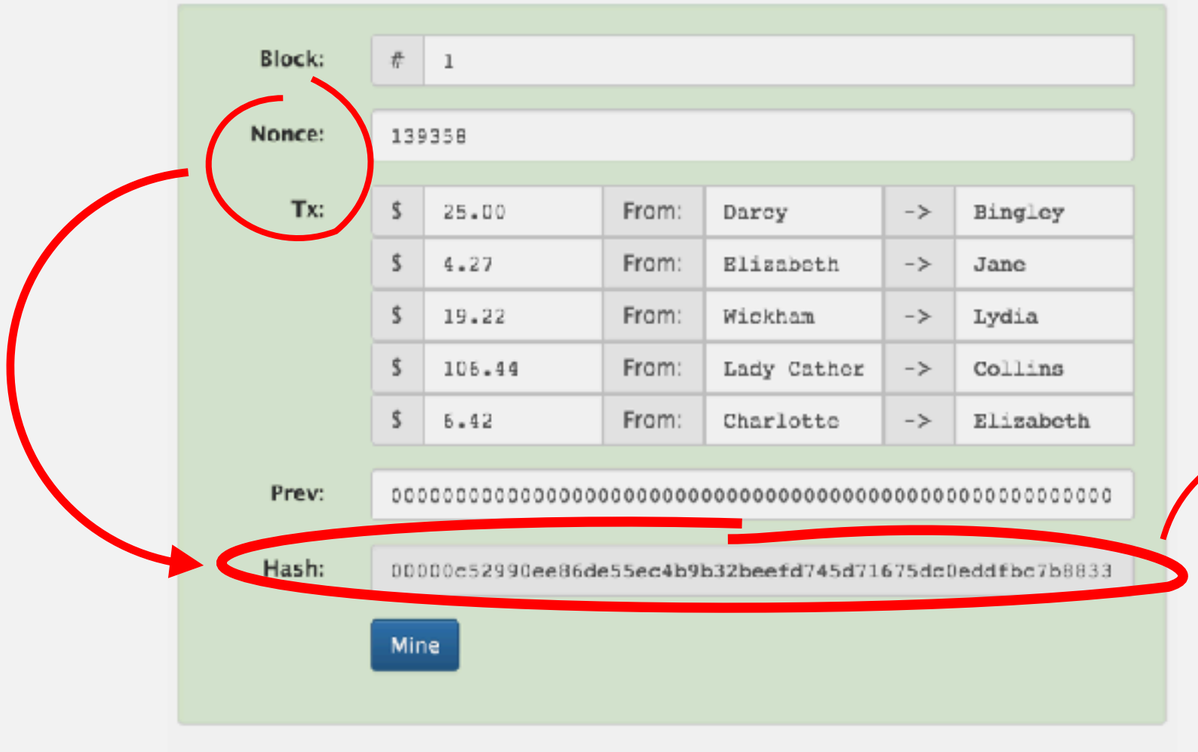
Chaining blocks together

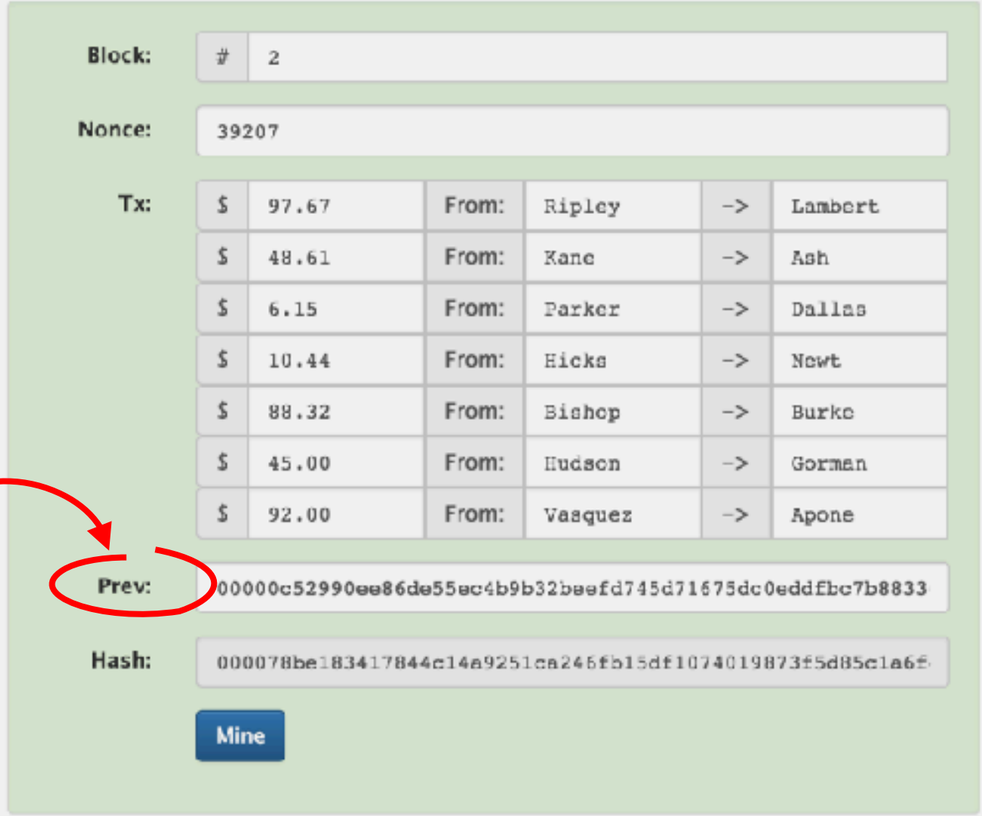
The process of validating and referencing the previous list is called mining. It's crypto magic.



Chaining blocks together

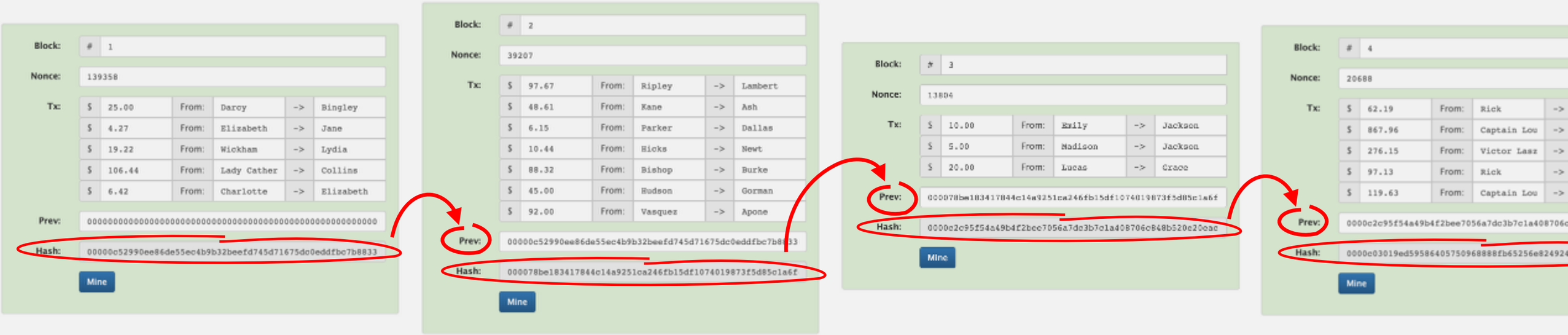
The process of validating and referencing the previous list is called mining. It's crypto magic.





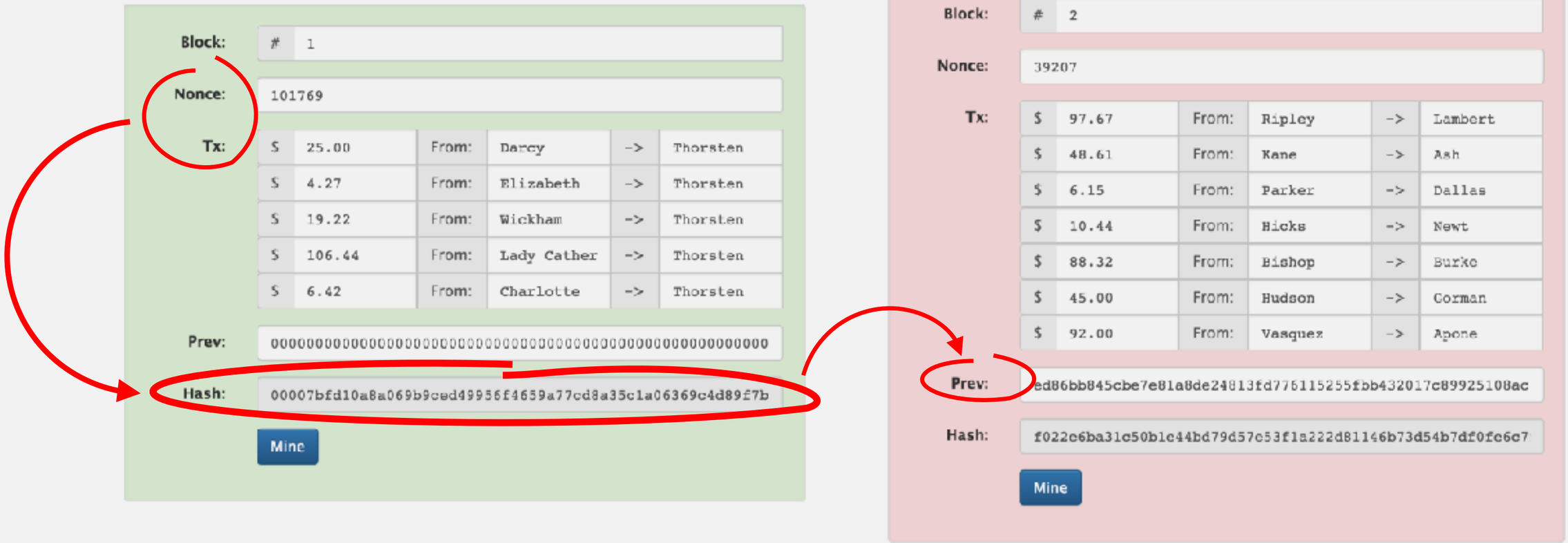
Chaining blocks together

This allows to store an unlimited number of Transactions within a chain structure.



Data consistency

Once validated and chained, transactions can not be manipulated easily.



Data redundancy

Each user (peer) works with her own copy of the full chain.

Peer A

Negotiating consensus

Each user (peer) works with her own copy of the full chain.

The validity of transactions is negotiated simultaneously across all peers.

Peer A

Peer B

Peer C

Peer D

Negotiating consensus

Each user (peer) works with her own copy of the full chain.

The validity of transactions is negotiated simultaneously across all peers.

This increases the consistency guarantee.

Peer A

Peer B

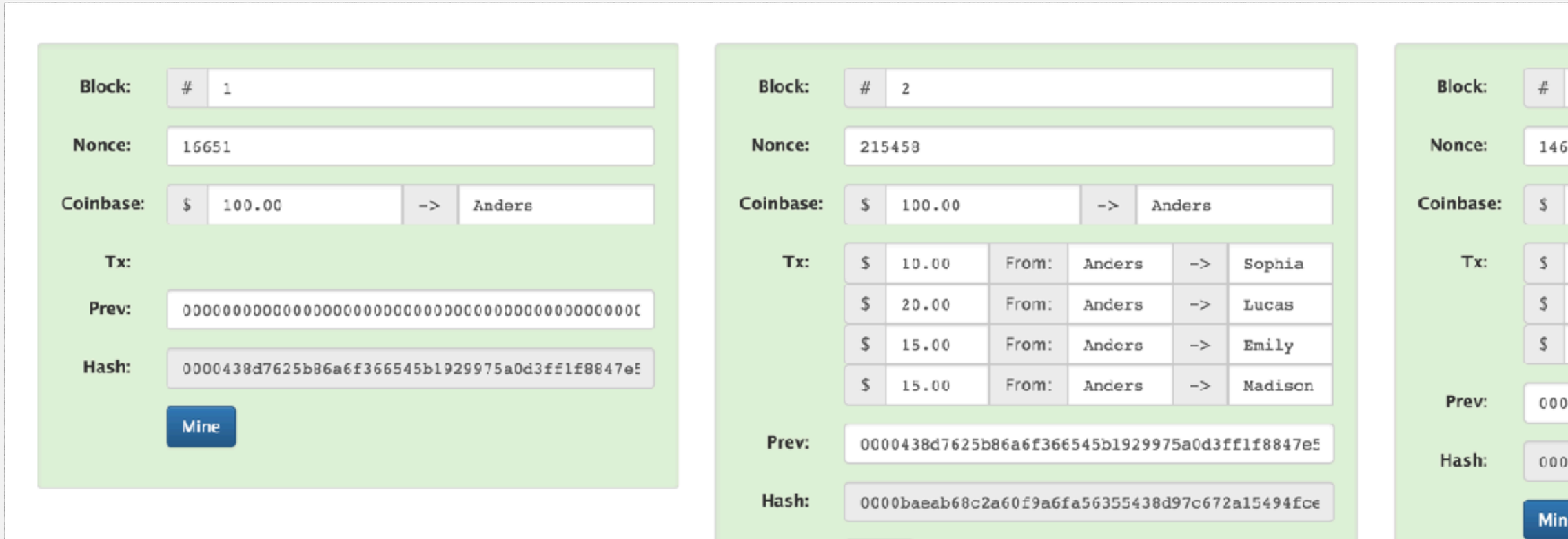
Peer C

Peer D

Controlling asset circulation with coinbase transactions

Initial asset creation

Creation and dissemination of is central to any consensus system.

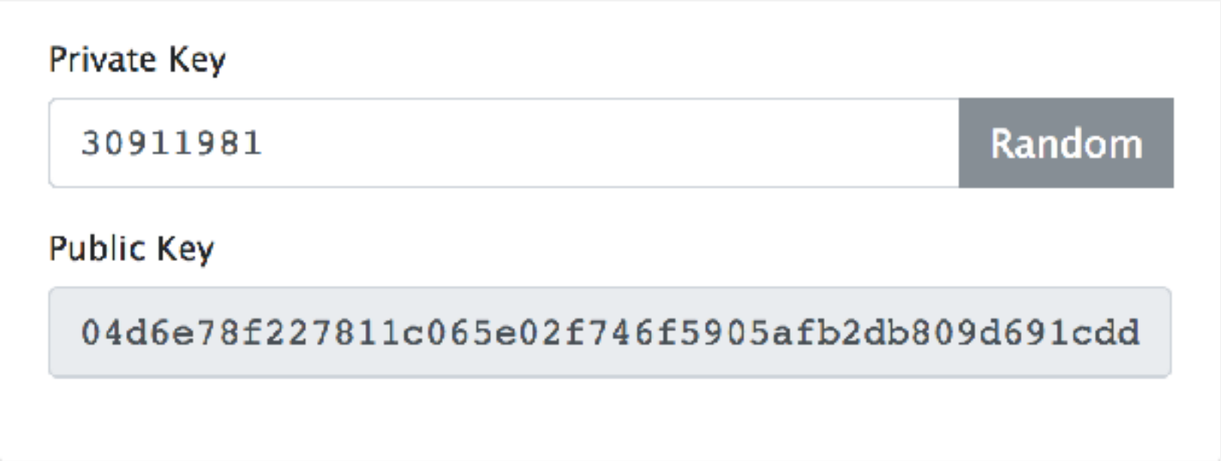


Authorising transactions

Public / private key pairs

A public key is data that is computed from a private key.

There is no way to derive from a public key what the private key is that relates to it.



Signing a messages with a private key

Signing data with a private key yields a unique signature that can be verified against the public key of the signee.

Message

Hallo Leipzig

Private Key

30911981

Sign

Message Signature

3046022100d93586571ec12aa6d59dee74f6343cedceea

Verifying a message against a public key

Verifying a signature validates that the person who signed the data in question had access to the private key behind the public key.

Message

Hallo Leipzig

Public Key

04d6e78f227811c065e02f746f5905afb2db809d691cdd

Signature

3046022100d93586571ec12aa6d59dee74f6343cedceea

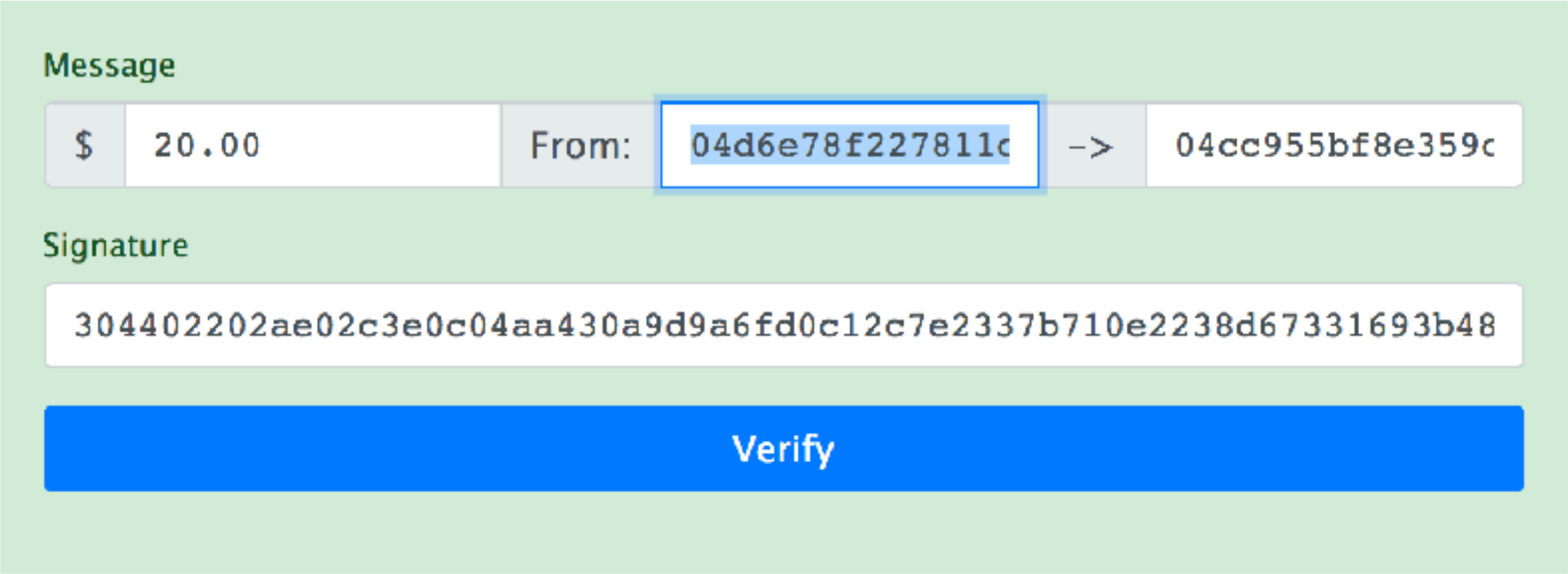
Verify

Signing a transaction

In a block chain context, every transactions must be signed by the actor who initiates the transaction.

Verifying a transaction

Verifying signed transactional data against the public key of the sender, validates that the sender did initiate said transaction with his or her private key.



Block chain transactions

Transactions in a block chain consist of public keys for sender & recipient and have to be signed by the sender.

This protects each transaction independent from the mining / block validation process.

Block:

#	2

Nonce:

7054

Coinbase:

$	100.00	->	04fe1be031bc7a54d900ff

Tx:

$	10.00	From:	04fe1be031b	->	04cc17dc129
Sig:	3046022100bcfe74e2ee8972367dda52a8f90008800ad10fb				

$	20.01	From:	04fe1be031b	->	04997ac426a
Sig:	304502210089cbf8f4bc854fb010c3bb7747f8c4c010fd029				

$	15.00	From:	04fe1be031b	->	042222d7af3
Sig:	3045022036cfd31dbdc400993a612bf9ba9c897a75b4578c8				

$	15.00	From:	04fe1be031b	->	041c377677b
Sig:	3045022036cfd31dbdc400993a612bf9ba9c897a75b4578c8				

Prev:

00006908f507a101e89544498978e9bd2e35462b91d86ef1351068

Hash:

0000b1df512e06962d2d177bb6c0fd0e5513d1cdeffdc52a79cc49

Takeaways

1. **Immutability**: History can not be changed.
2. **Openness**: Everyone and everything with a key pair can participate.

A fault tolerant, highly redundant transaction store…

...to establish & maintain irrevocable consensus among market participants.

A value machine

Blockchain technology can be utilised to facilitate and enshrine any kind of transaction between market participants.

Backup

Blockchain-Akteure & Adressen.

Jedes Objekt, das eine Transaktion in der Blockchain vornehmen kann, hat eine eineindeutige Adresse.

Rückschlüsse, ob es sich dabei um einen Marktakteur, SmartContract, oder Token handelt, sind nicht möglich.

Adressen sind (unveränderlich). Ein Smart Contract mit einer bestimmten Adresse kann nicht mehr nachträglich in seiner Funktion verändert werden.

Transfers of value must adhere to rules to be effective

A transfer of value requires irrevocable and verifiable proof of the transfer itself and of its necessary preconditions.

Transaction rules may be enshrined in (Smart) Contracts. Contracts effect the transfer of value.

```
pragma solidity 0.4.18;
contract SimpleMultiSig {

  uint public nonce;                // (only) mutable state
  uint public threshold;            // immutable state
  mapping (address => bool) isOwner; // immutable state
  address[] public ownersArr;        // immutable state

  function SimpleMultiSig(uint threshold_, address[] owners_) public {
    require(owners_.length <= 10 && threshold_ <= owners_.length && threshold_ != 0);

    address lastAdd = address(0);
    for (uint i=0; i<owners_.length; i++) {
      require(owners_[i] > lastAdd);
      isOwner[owners_[i]] = true;
      lastAdd = owners_[i];
    }
    ownersArr = owners_;
    threshold = threshold_;
  }

  // Note that address recovered from signatures must be strictly increasing
  function execute(uint8[] sigV, bytes32[] sigR, bytes32[] sigS, address destination, uint
value, bytes data) public {
    require(sigR.length == threshold);
    require(sigR.length == sigS.length && sigR.length == sigV.length);

    // Follows ERC191 signature scheme: https://github.com/ethereum/EIPs/issues/191
    bytes32 txHash = keccak256(byte(0x19), byte(0), address(this), destination, value, data,
nonce);

    address lastAdd = address(0); // cannot have address(0) as an owner
    for (uint i = 0; i < threshold; i++) {
        address recovered = ecrecover(txHash, sigV[i], sigR[i], sigS[i]);
        require(recovered > lastAdd && isOwner[recovered]);
        lastAdd = recovered;
    }

    // If we make it here all signatures are accounted for
    nonce = nonce + 1;
    require(destination.call.value(value)(data));
  }

  function () public payable {}
]
```