Emerging Technologies for the Circular Economy

Lecture 7: Introduction to Blockchain Technology

Prof. Dr. Benjamin Leiding (Clausthal)
M.Sc. Arne Bochem (Göttingen)
M.Sc. Anant Sujatanagarjuna (Clausthal)

License

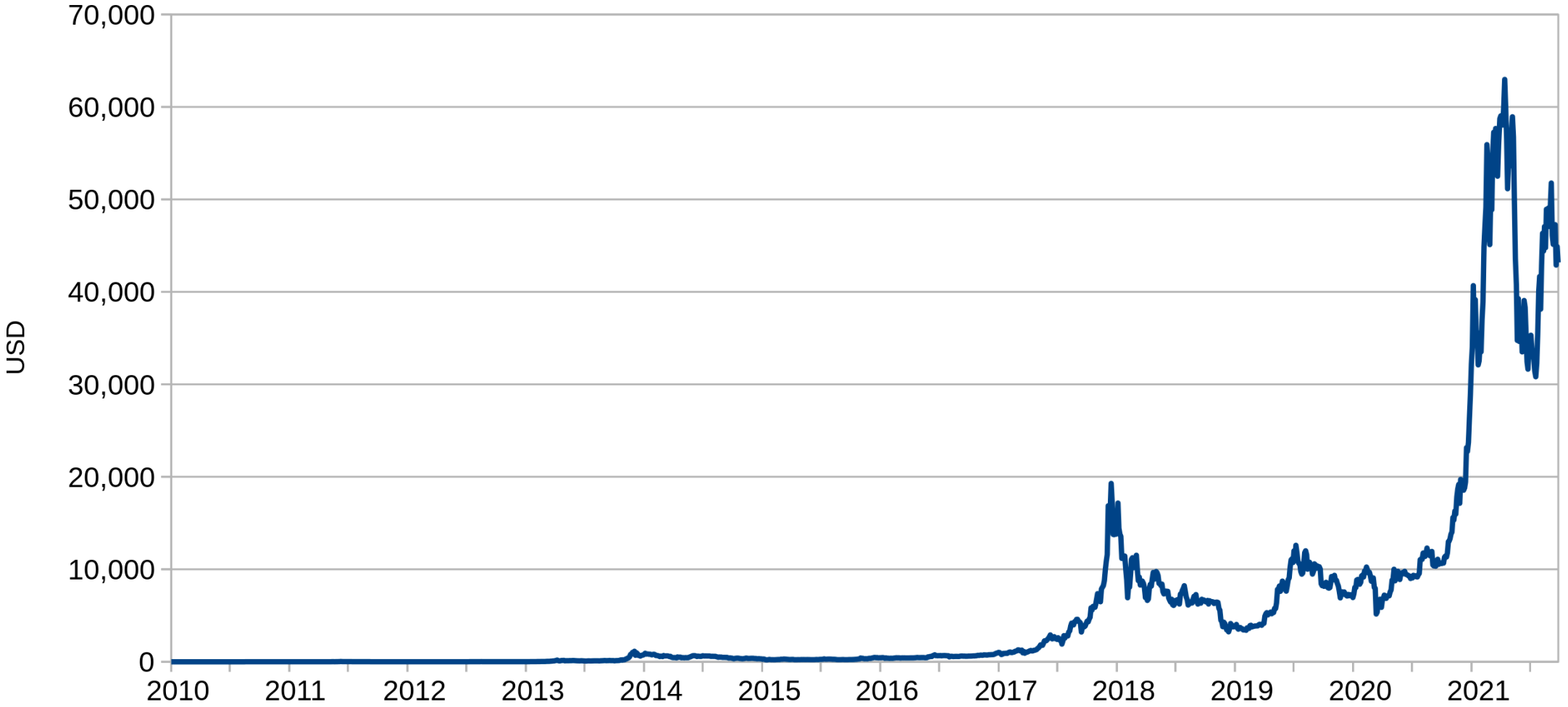
Disclaimer and Further Resources

- The lecture slides are based on the course *“Blockchain-based Systems Engineering”* from TU Munich
- All their slides, exercises and further information are available online – Link

U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

INTRODUCTION AND HISTORY

History



History

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” https://bitcoin.org/bitcoin.pdf

Blockchain Essentials

- Bitcoin ≠ Blockchain
  - Blockchain is the fundamental technology
  - Bitcoin is a specific implementation

Blockchain Essentials

Bitcoin ≠ Blockchain

What makes Bitcoin special?

Blockchain Essentials

Bitcoin ≠ Blockchain

What makes Bitcoin special?

- Decentralized digital currency → No central bank, no government
- No (trusted) third-party required to validate transaction in contrast to banks, Paypal, Western Union, etc.
- Bitcoins can be transferred but not duplicated
- Transparent
- Pseudonymous, not anonymous!
- Deflationary (max. 21 million coins)

BLOCKCHAIN TECHNOLOGY

Overview

Underlying concepts are not revolutionary, all of them have already been used for years, **BUT** mixing them all together was a revolutionary step.

In essence, blockchain is a:

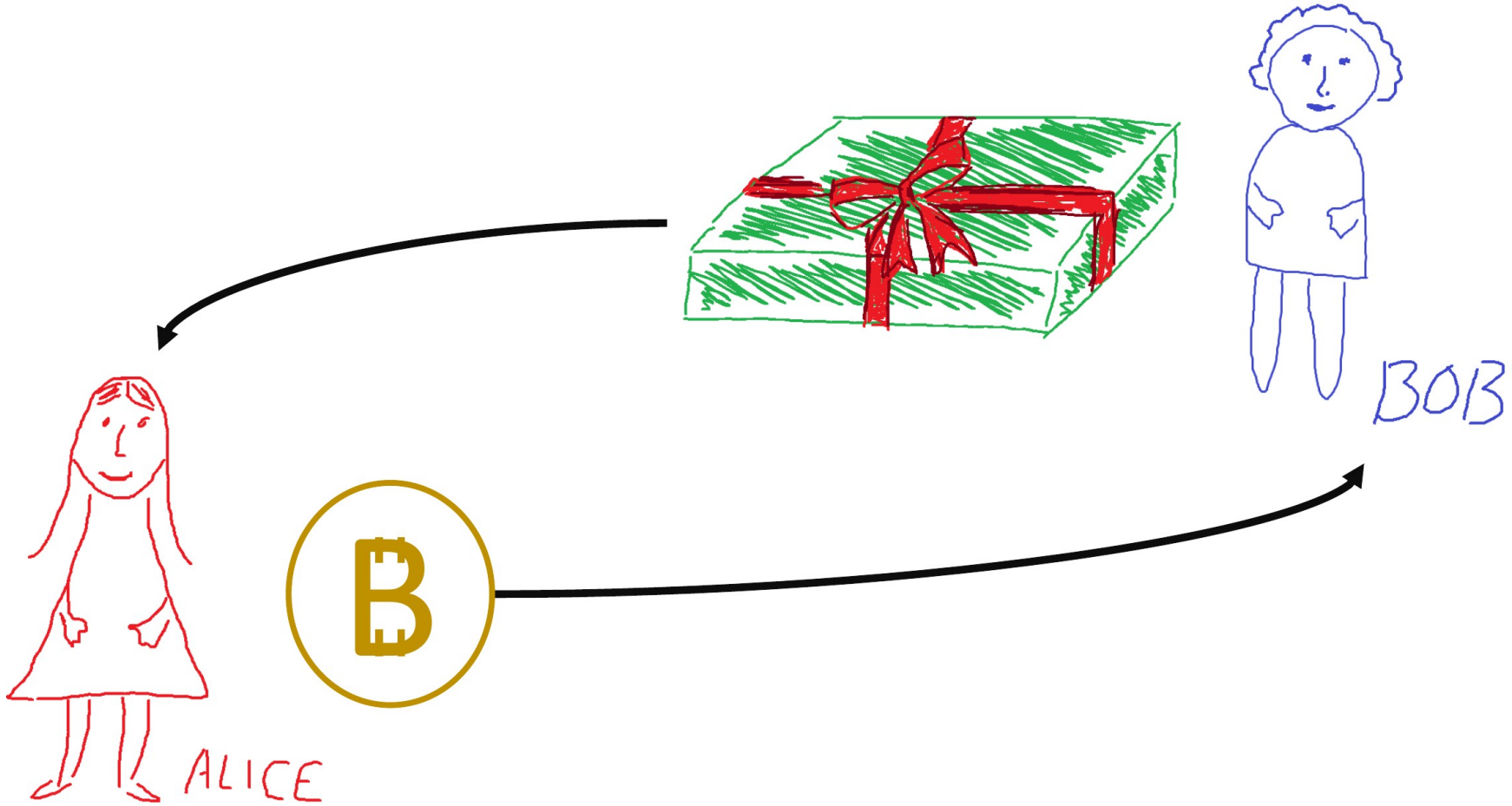
- distributed,
- secure,
- logfile.

Transaction Ledger

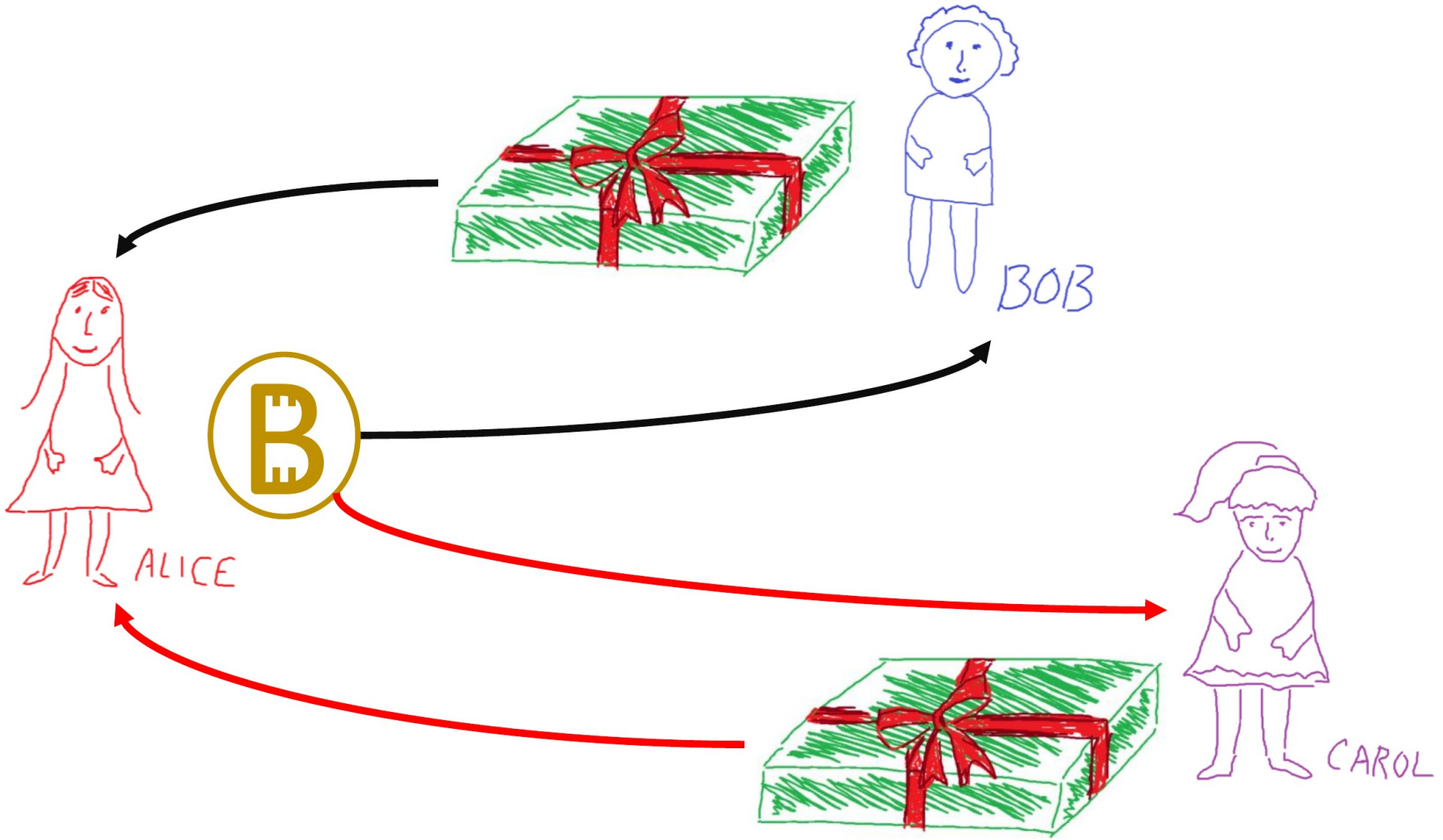
- Essentially a list of transactions
- e.g., Listing money going in/out
- But can also be used for other things, such as maintaining a record of all changes made to a house, all the owners, etc, or an “auto scheckheft”

A german ledger from 1828, by RaphaelQS licensed under CC0 1.0 Universal Public Domain

Transactions



Double Spending Problem



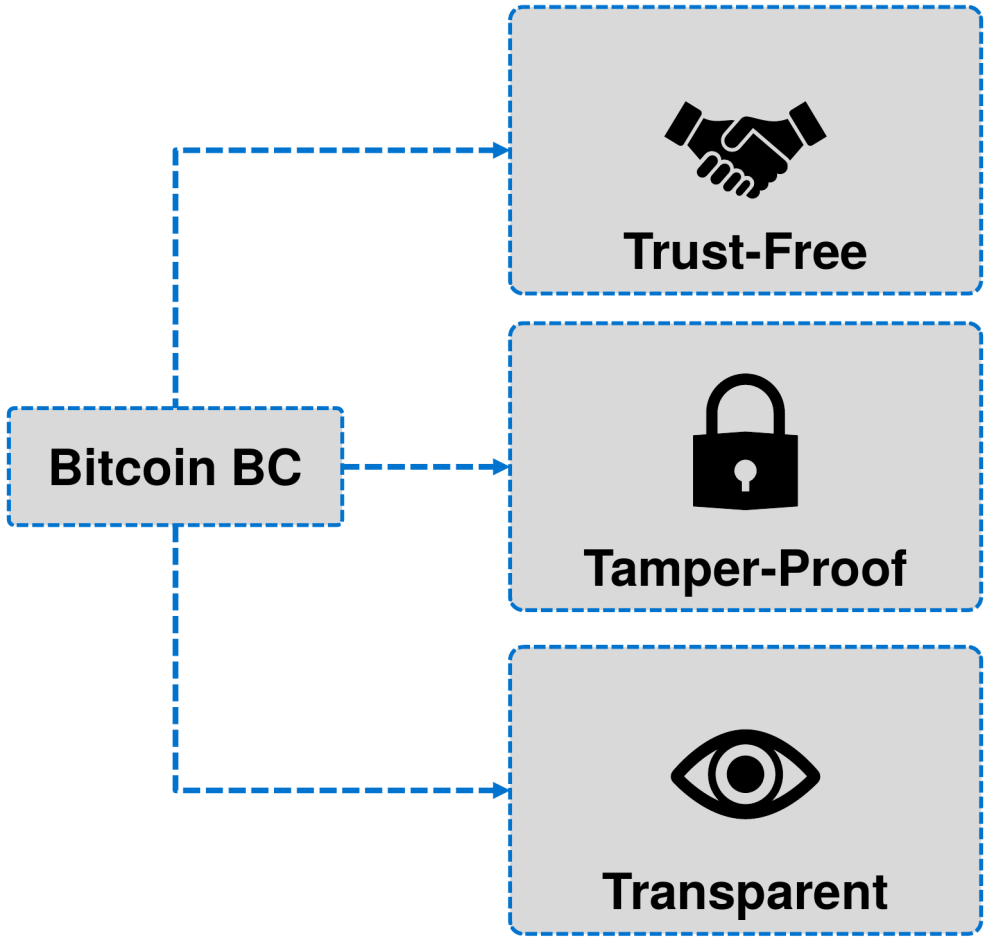
Double Spending Problem

The most common solution to the double spending problem:

- Banks and other intermediaries (Amazon, Ebay) → adds significant cost
- Main problem → trust in intermediaries:
  - The intermediary can tamper with the data
  - Difficult to do on a global scale while maintaining trust, accountability and transparency

Key Properties

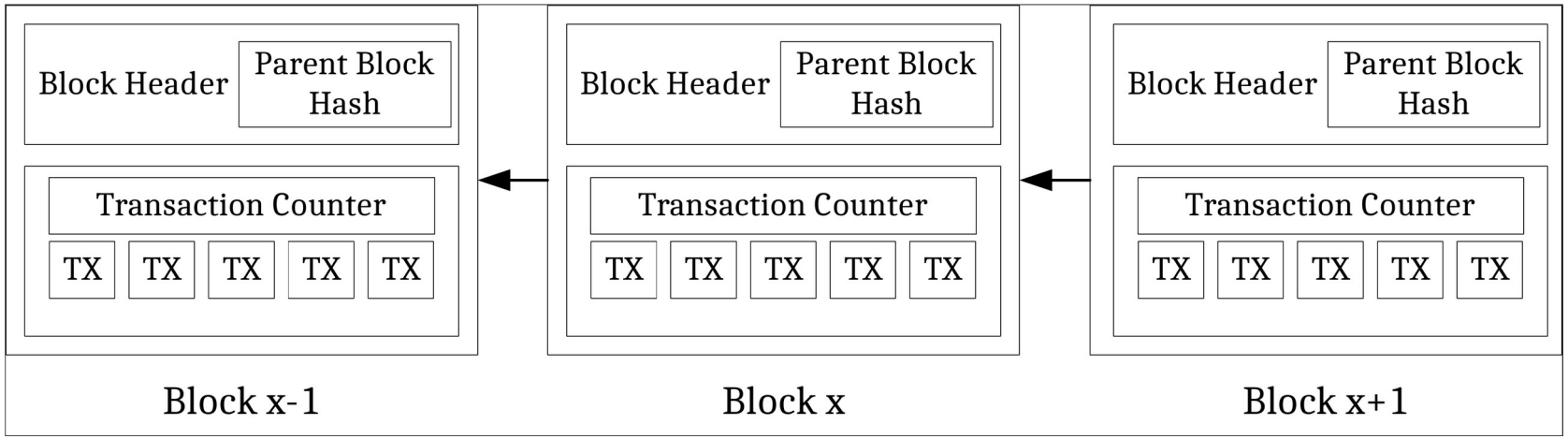
- **Trust-Free:** The system does not require a third party which controls or maintains the system.
- **Tamper-Proof:** The system is resistant to manipulation. The history of events cannot be changed.
- **Transparent:** Every participant of the system can read and validate all information and the current state.



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

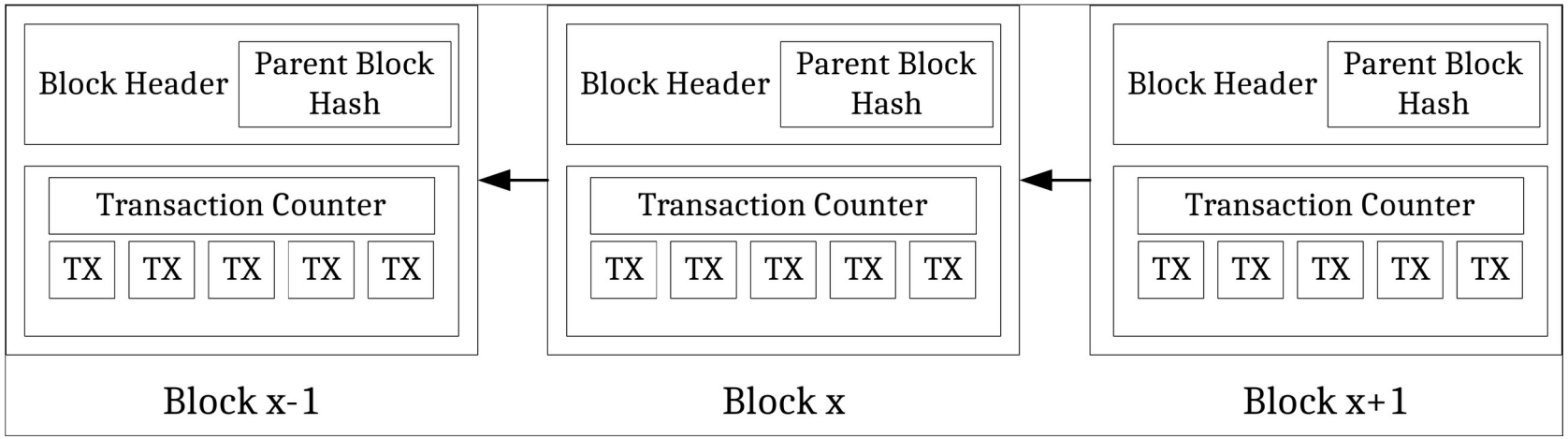
Blockchain

- Append-only sequential data structure (list)
- New blocks can only be added to the end of the chain
- Chaining blocks together → blockchain
- Changing a block requires recalculation of all subsequent blocks
- Massively duplicated (redundant) across the network (P2P)



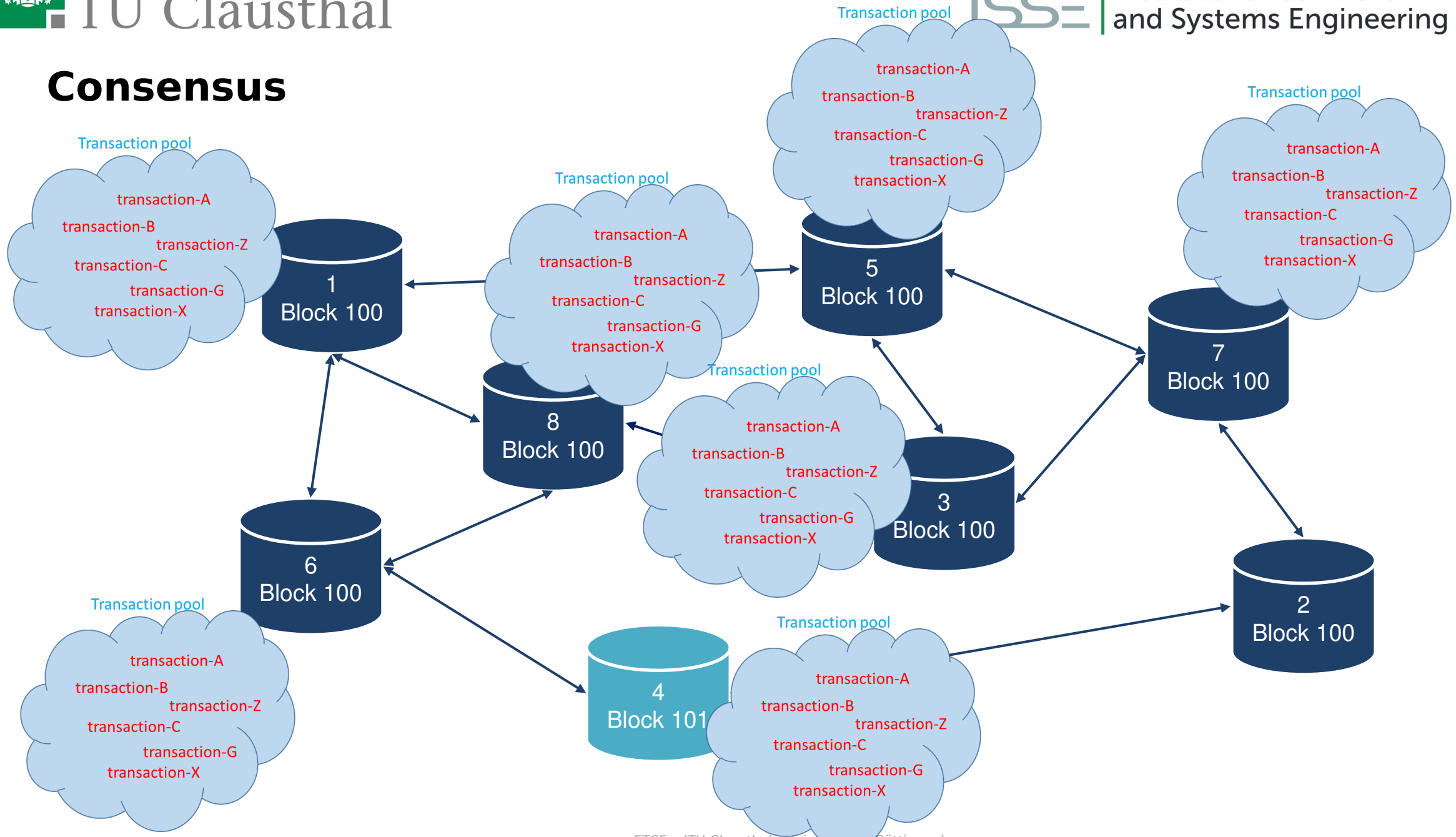
Blockchain

- Append-only sequential data structure (list)
- New blocks can only be added to the end of the chain
- Chaining blocks together → blockchain
- Changing a block requires recalculation of all subsequent blocks
- Massively duplicated (redundant) across the network (P2P)

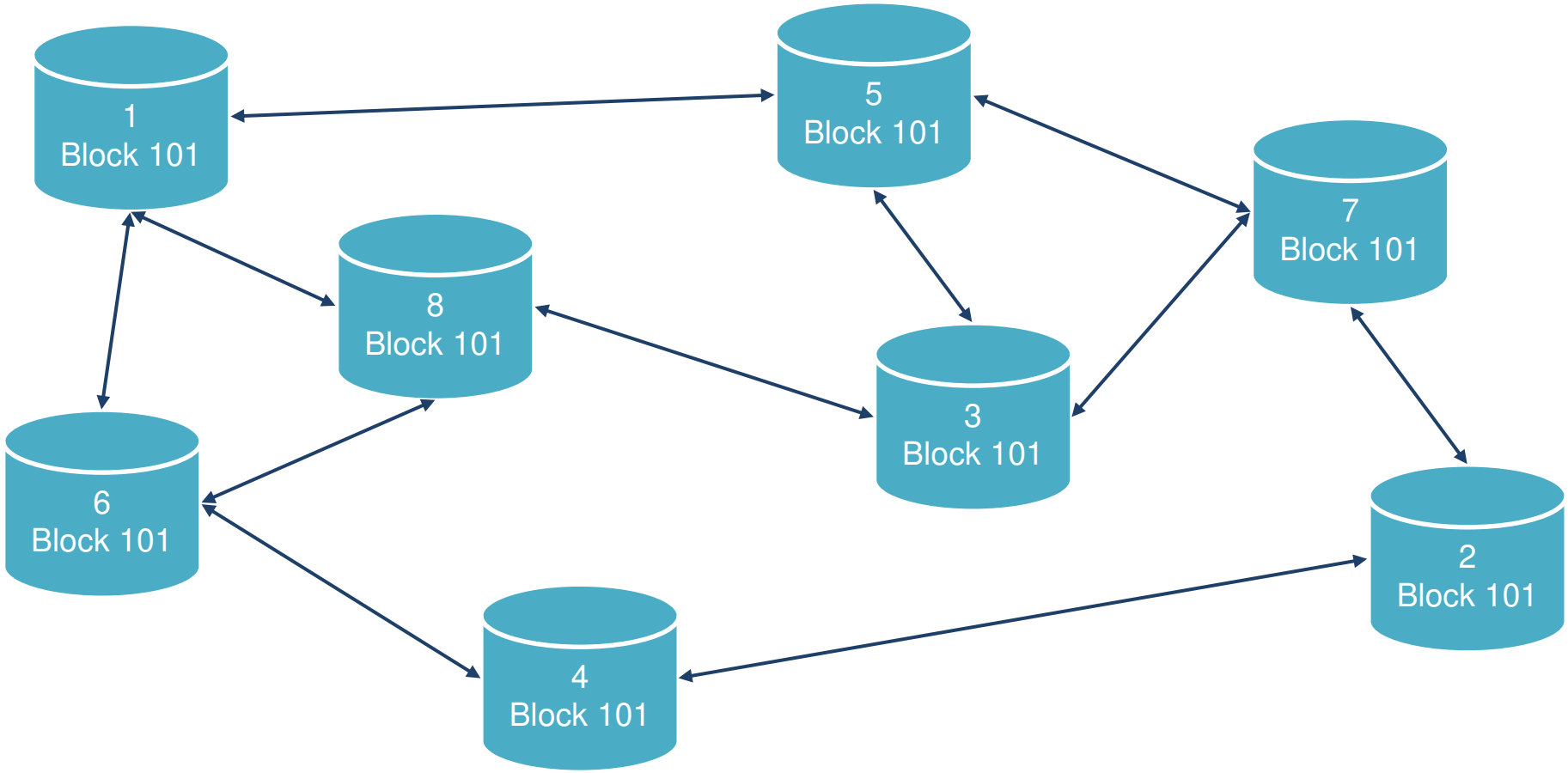


⇒ How to reach consensus on which transactions are part of the next block?

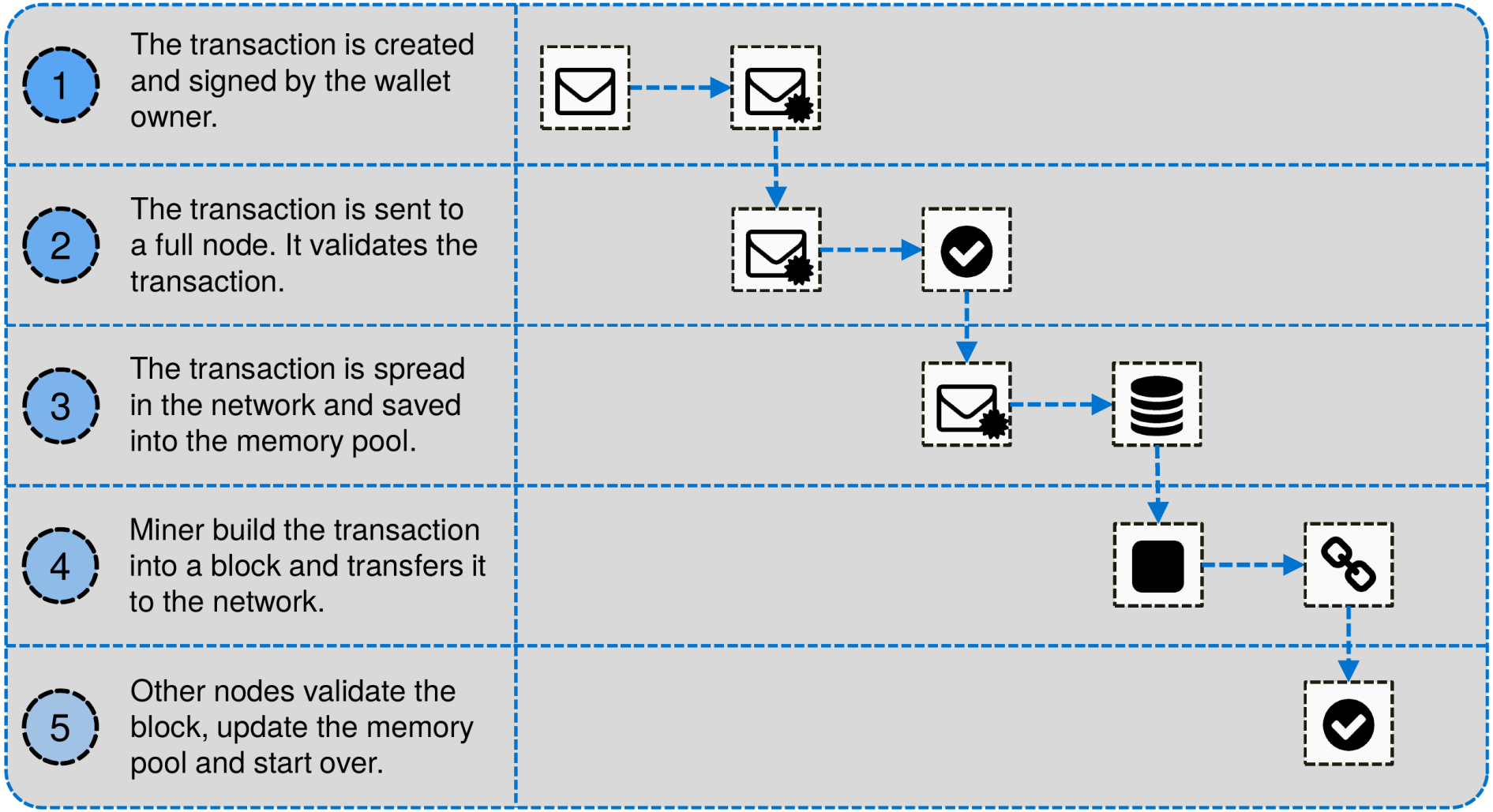
Consensus


Transaction pool
transaction-A
transaction-B
transaction-Z
transaction-C
transaction-G
transaction-X
1
Block 100
5
Block 100
7
Block 100
8
Block 100
3
Block 100
6
Block 100
4
Block 101
2
Block 100

Consensus


1
Block 101
5
Block 101
7
Block 101
8
Block 101
3
Block 101
6
Block 101
2
Block 101
4
Block 101

Propagating a Transaction



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Blockchain Platforms

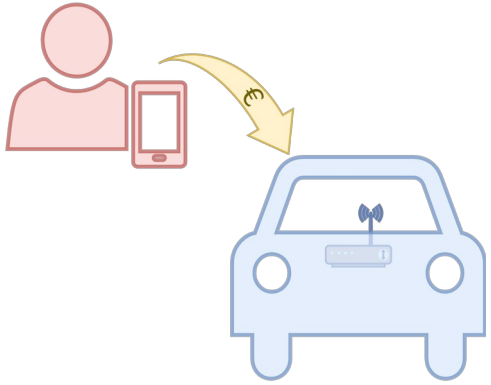
	Ethereum	Hyperledger Fabric	Ripple
Industry	Cross-industry	Cross-industry	Finance
Ledger Type	Public	Permissioned	Permissioned
Currency	Ether (ETH)	-	Ripple (XRP)
Concensus	Proof-of-Work (soon Proof-of-Stake)	Different modules	Federated Byzantine Agreement
Smart Contracts	Yes	Yes	No

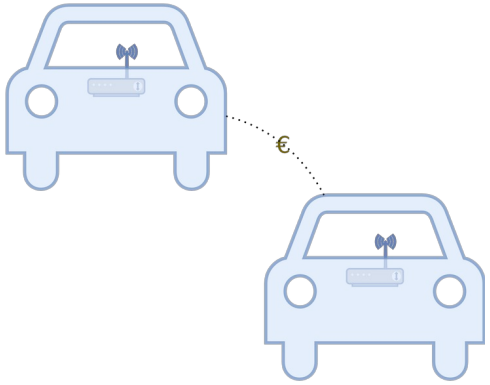
Smart Contracts

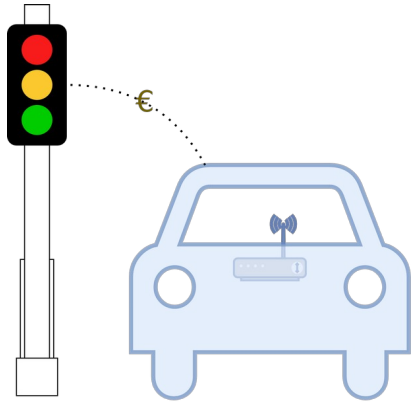
- Bitcoin has a very limited scripting languages to manage transactions → Smart-contracts are far more powerful
- Turing complete languages (e.g., Solidity)
- Contract:
  - Contract code compiled to byte code - e.g., for the Ethereum virtual machine (EVM)
  - Code execution (may) consumes cryptocurrency (tokens)
  - Autonomous execution
  - Every node runs every contract (deterministic)
  - Contracts can interact with world using so called ”oracles”
  - Smart contract capable blockchains → Ethereum,
- Corda, Qtum, etc.

APPLICATIONS

Machine-to-Everything (M2X) Economy







Machine-to-Human (M2H)
+
Machine-to-Machine (M2M)
+
Machine-to-Infrastructure (M2I)
=
Machine-to-Everything (M2X)

Blockchain-based Crowdfunding

- ICO = IPO on the blockchain.
- Fund development of your blockchain-based idea → Collect funding in exchange for tokens
- Product, team, marketing, social media, whitepaper, roadshow, etc.

Blockchain-based Crowdfunding

- **$28,366,035,804** raised between January 2014 and August 2018.

Some of the largest ICOs (in terms of how much they raised)

Coin	Total Funds raised	Year
EOS	$4.2 Billion	2018
Telegram	$1.7 Billion	February 2018
TaTaTu	$575 Million	June 2018
Dragon Coin	$420 Million	March 2018

Based on data sourced from: https://elementus.io/token-sales-history

Do I need a blockchain for that?

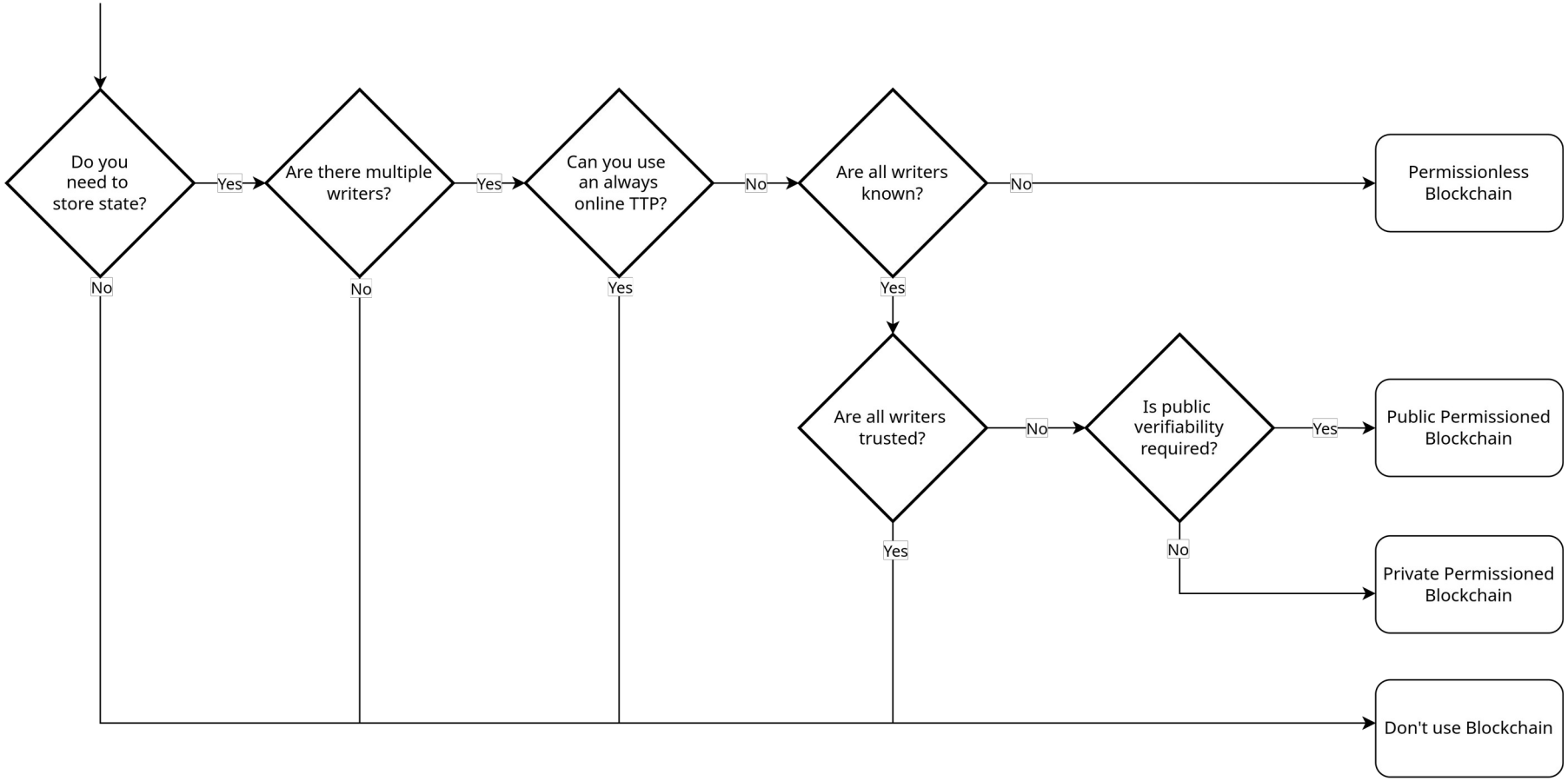


Image recreated from K. Wüst and A. Gervais, “Do you need a Blockchain?” IACR Cryptology ePrint Archive, vol. 2017, p. 375, 2017.

Mileage Logging

Connector in car sends mileage information to digital logbook → Data is distributed across many computers - blockchain ensures the authenticity and privacy of the data → When selling the car, authenticity of the odometer reading can be verified.

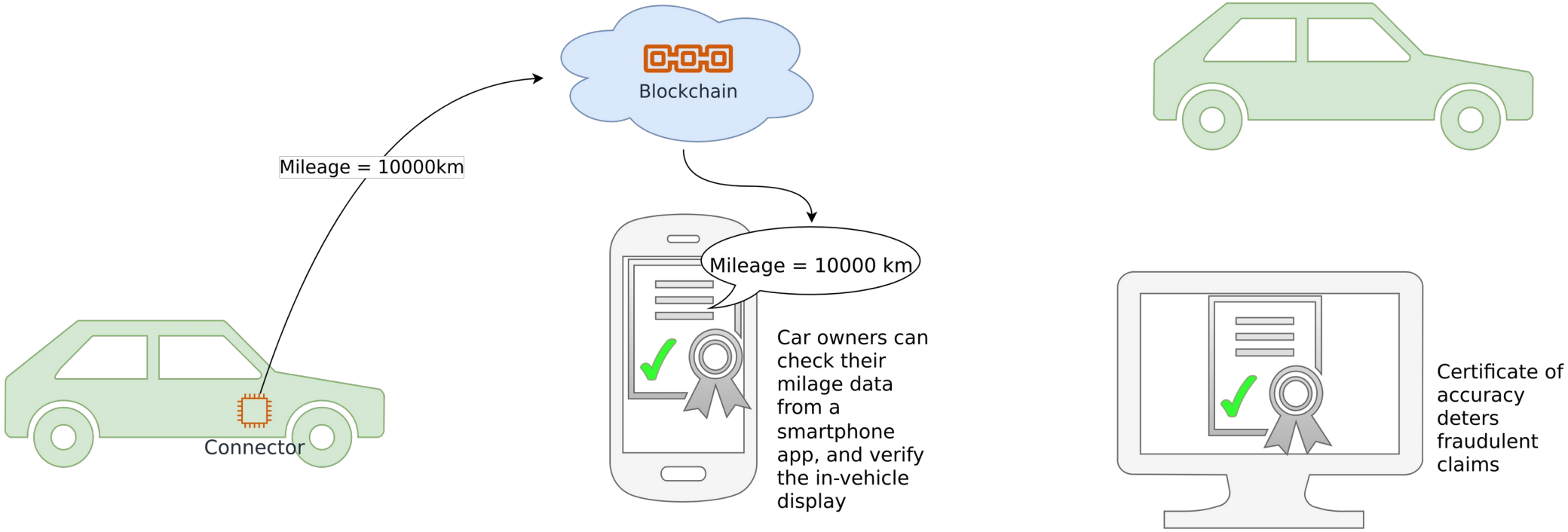


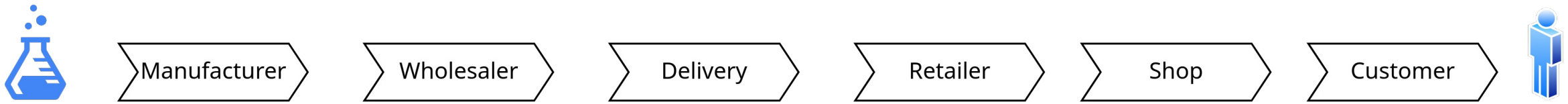
Image adapted from https://twitter.com/boschglobal/status/844131521426538497?lang=en

Access Management

- Not only buildings
- Sharing economy (cars, bikes, apartments, etc.)
- Also:
  - Health records
  - Insurance information (e.g., car insurance)
  - ⇒ Sharing of personal data in general

Supply-Chain Management

- Pharmaceutical supply chain management



Supply-Chain Management

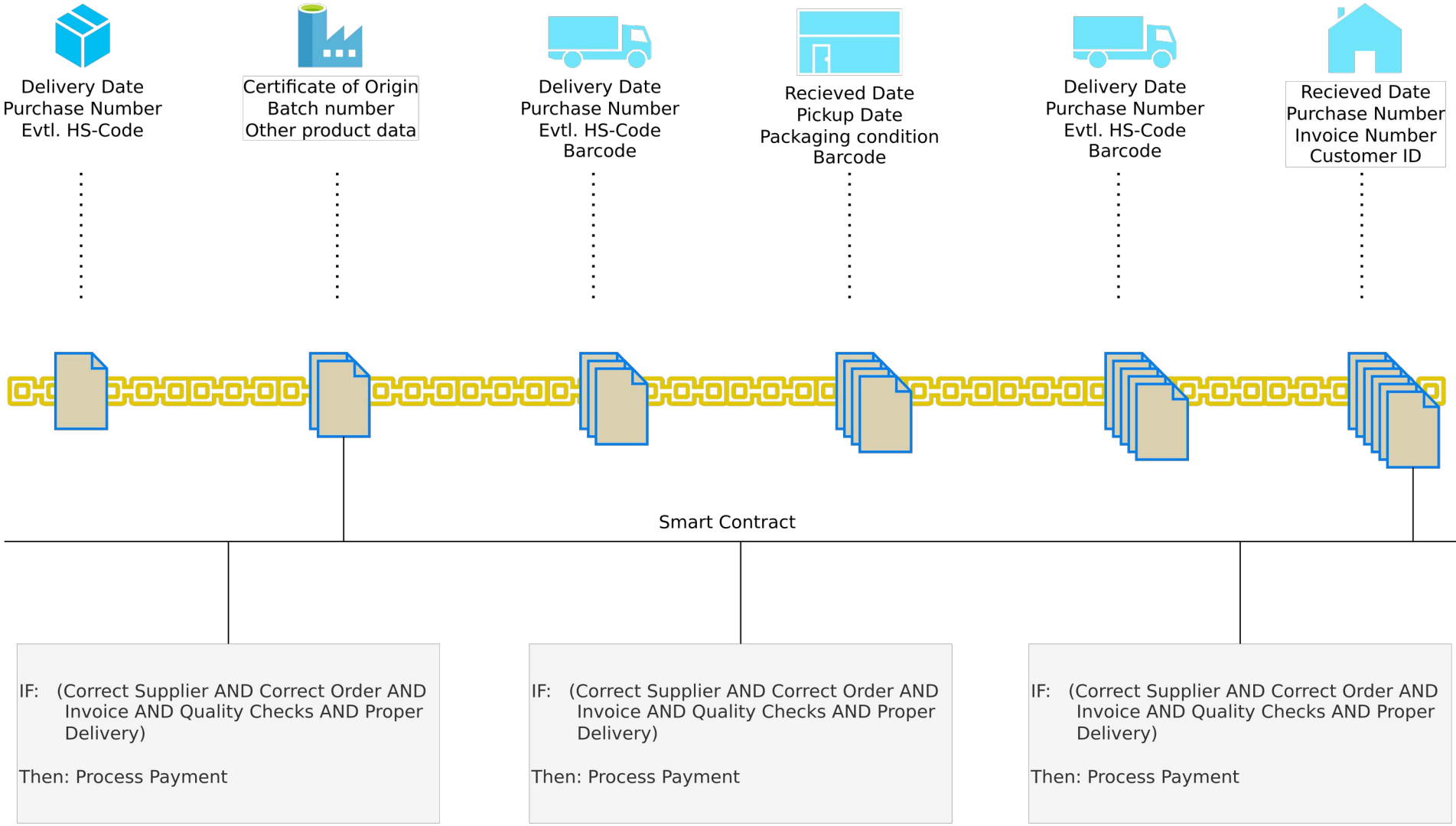


Image recreated from: BearingPoint – Blockchain als Treiber im modernen Supply Chain Management 4.0 (2018).

Asset Tokenization

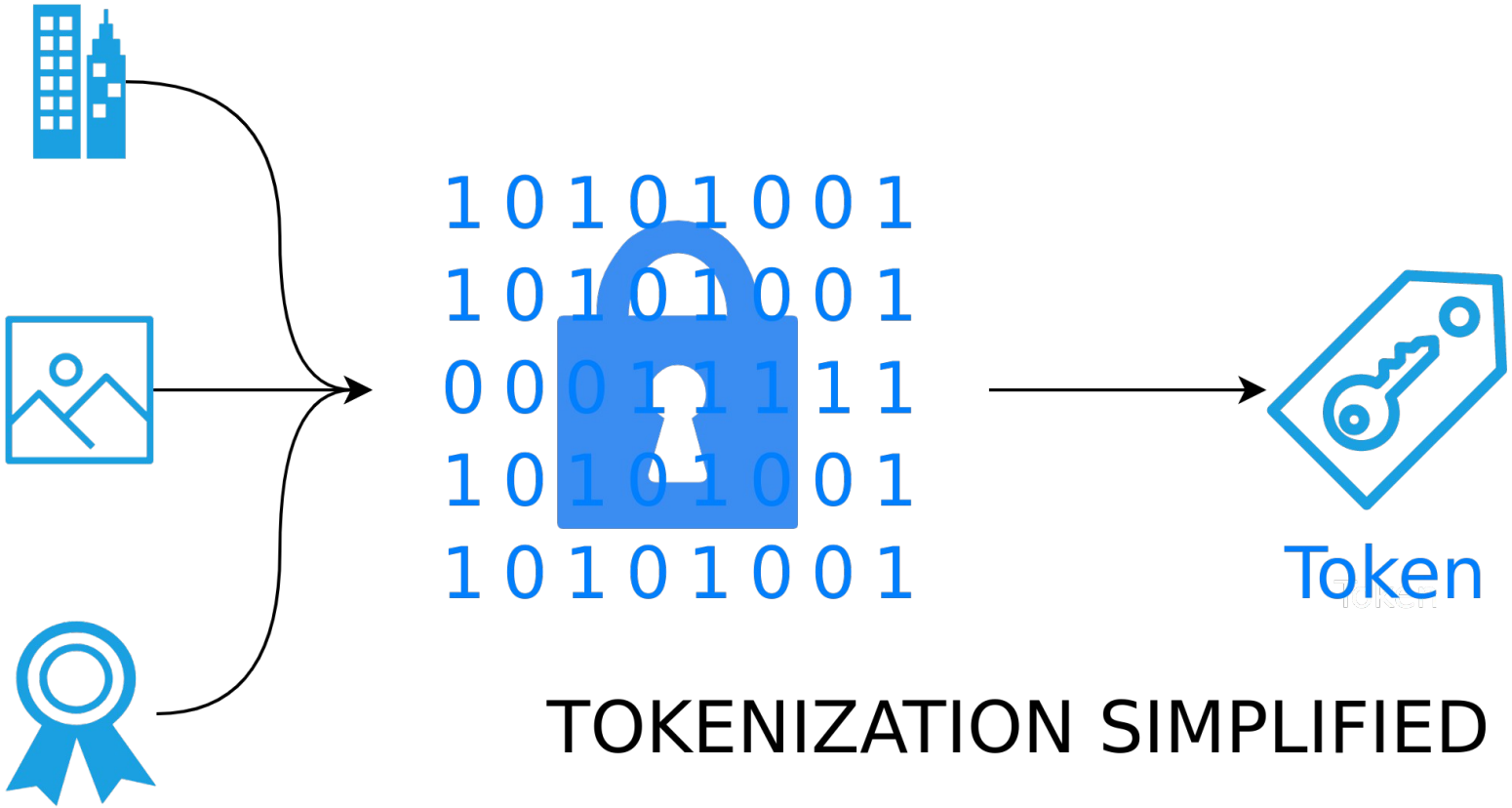


Image recreated based on blog.softwaremill.com

Asset Tokenization

Image recreated based on blog.softwaremill.com

Further Applications

- Identity solutions (Self-Sovereign Identities, DIDs, Authcoin, Unchained(X), Rechained, BlockVoke)
- Spotify on the blockchain
- Voting
- Land registry entries
- Frictionless payment solutions
- and so on ...

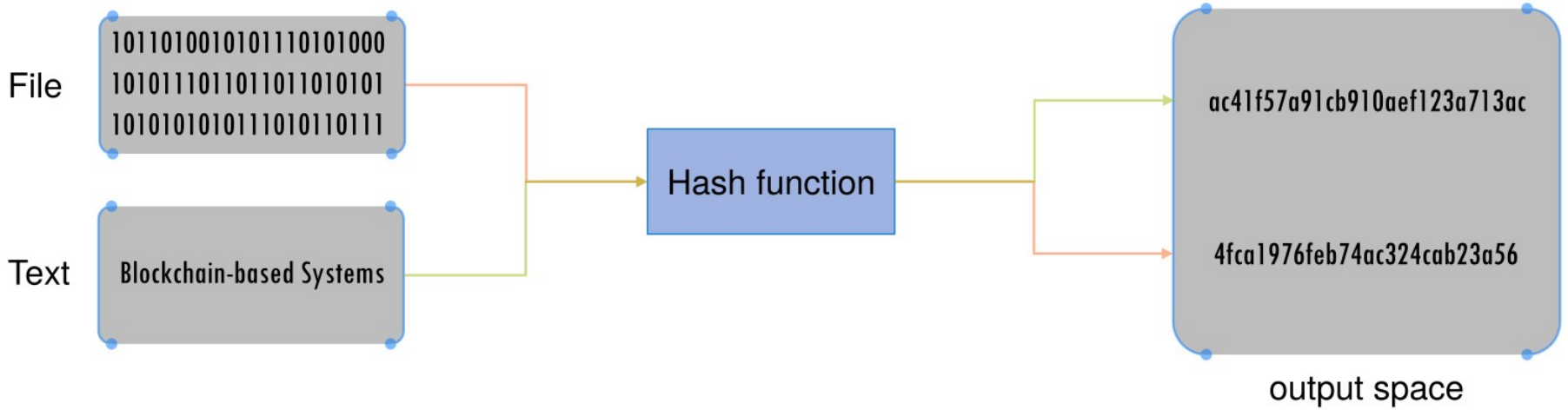
CRYPTOGRAPHY FOR BLOCKCHAIN TECHNOLOGY

Quick Note

- While cryptocurrencies use cryptographic techniques such as cryptographic hash functions and signature schemes, there is usually no encryption or decryption involved.

What Are Hash Functions?

- **Definition:**
  - A function *h* is called a hash function if it has the following properties:
  - Compression: *h* maps an input *x* of arbitrary finite bit length to an output *h(x)* of fixed bit length *n*:
    - $h : 0, 1^* \rightarrow 0, 1^n$
  - Ease of computation: Given *h* and *x* it is easy to compute *h(x)*



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Additional Properties of Cryptographic Hash Functions

- Cryptographic hash functions have certain properties in addition,compared to regular hash functions.
- First two definitions that will be useful for the remainder of this section.

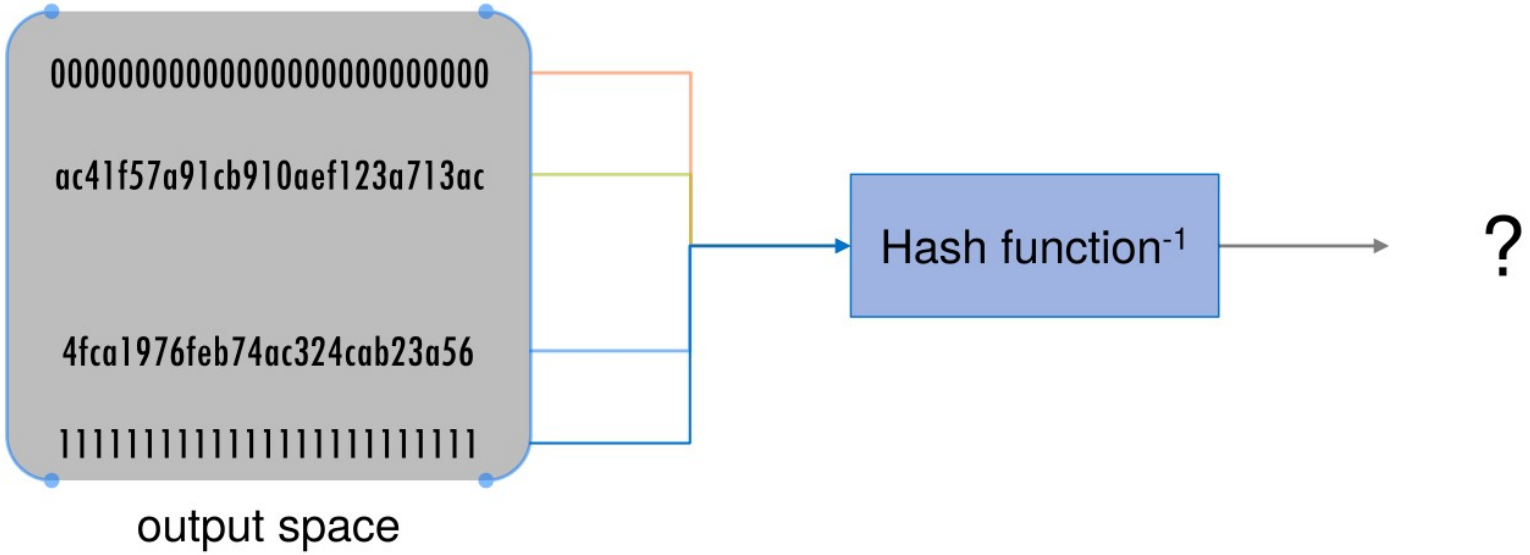
Definition
If a function h is a *one-way function*, then a function h^{-1} does not exist. It is therefore computationally infeasible to find the input to an output of h.

- **Definition**
  A problem that is *computationally infeasible* is e.g. a ”hard” instance of an NP-hard problem of sufficient size. ”Hard” means that there is no more efficient way of solving the problem than trying all possible solutions (”bruteforce”). Sufficient means that the size is large enough that it can be considered not computable with classical computers, e.g. a search space of size 2^{256}.

Additional Properties of Cryptographic Hash Functions

Definition
Preimage resistance of hash function *h*:

- For essentially all pre-specified outputs *y*, it is *computationally infeasible* to find an *x* such that *h(x) = y*
- *h* is also called a one-way function

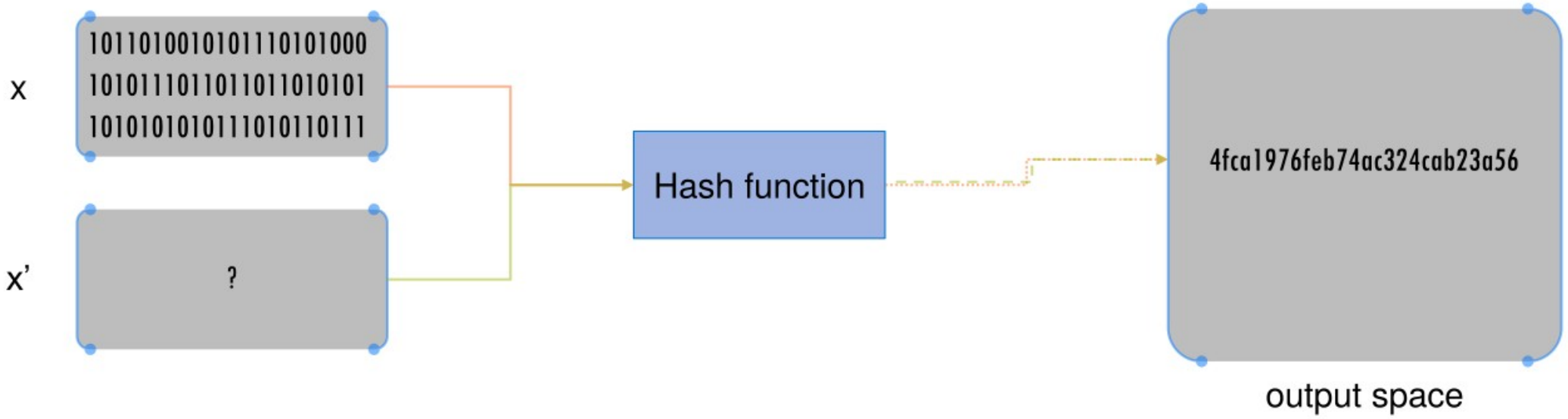


U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Additional Properties of Cryptographic Hash Functions

Definition

- Second preimage resistance is the property of a hash function h that for a given *x* it is computationally infeasible to find any second input *x'* with *x != x'* such that *h(x) = h(x')*. In other words, it is computationally infeasible to find a different input *x'* that results in the same output.

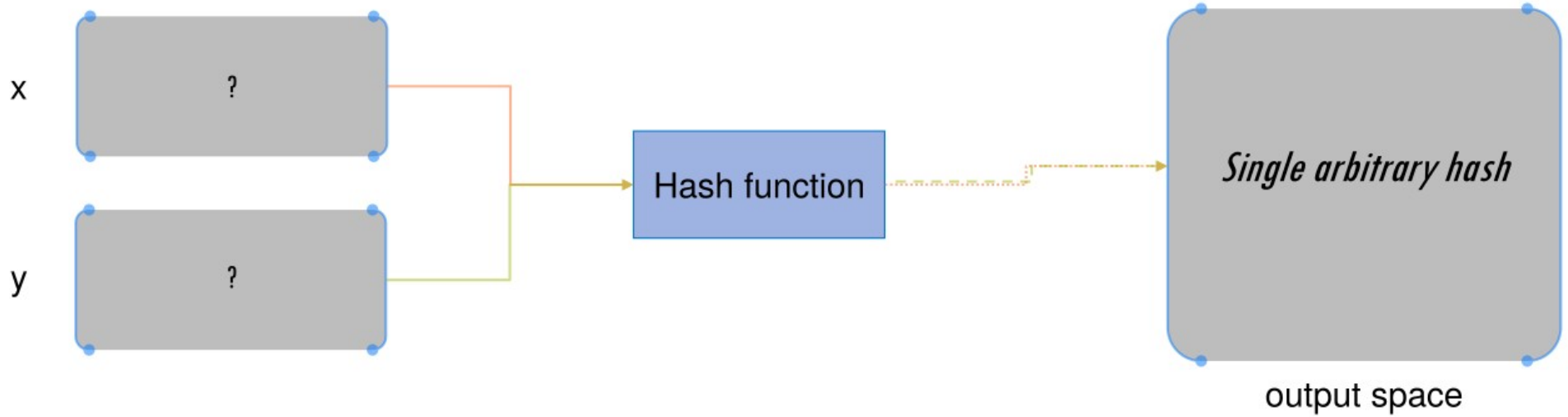


U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Additional Properties of Cryptographic Hash Functions

Definition

- A hash function *h* is said to be collision resistant if it is computationally infeasible to find two values *x* and *y* such that *x != y* , but *h(x) = h(y)*. In other words, it is *computationally infeasible* to find two values that hash to the same output. This is different from second preimage resistance insofar as here no fixed *x* or output is given and both inputs may be changed.

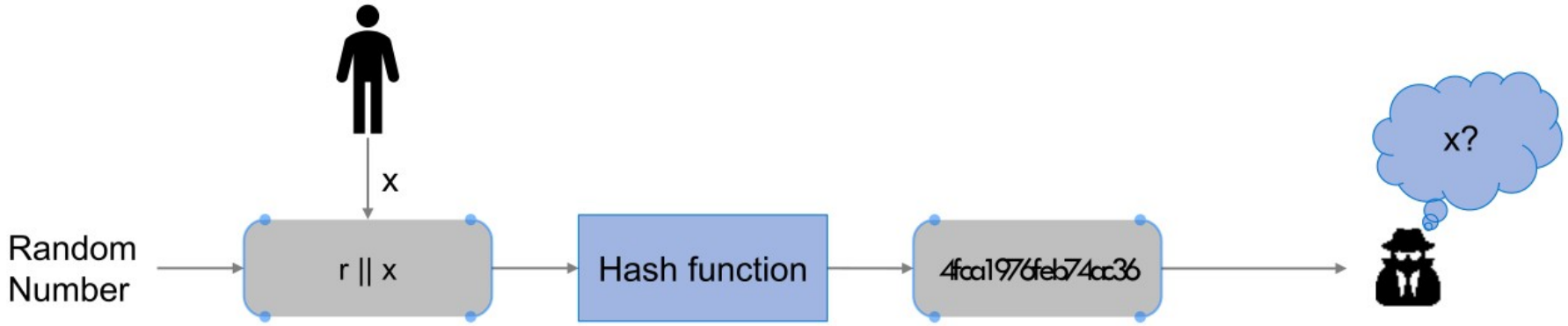


U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Additional Properties of Cryptographic Hash Functions

Definition

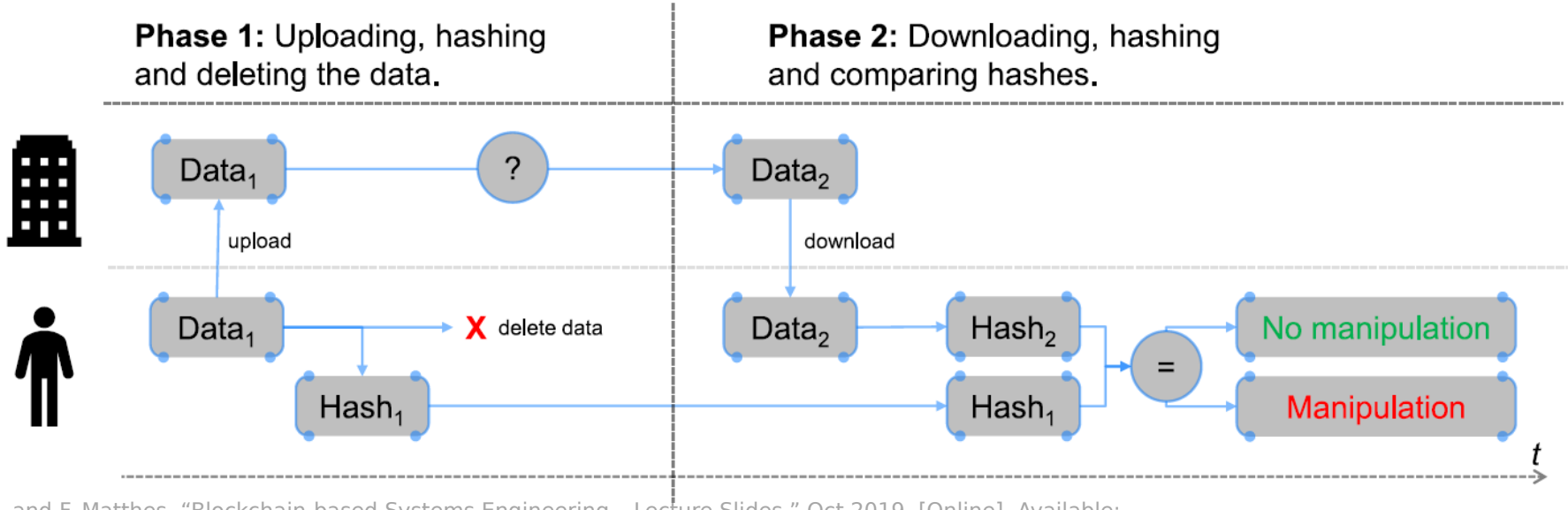
- A hash function *h* is said to be hiding if, for a secret, randomly chosen value *r* of sufficient size, given *h(r||x)* it is *computationally infeasible* to find *x*. Sufficient size, again, refers to the search space. A size of 256 bit can usually be considered sufficient.
- This means that the hash function in combination with a random number can protect the value *x* contained in the hash, even if is from a small set of values that otherwise could easily be found through bruteforce.



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Application: Message Digests

Suppose you want to store information on an external hosting service. After a successful upload on the external service you want to free up space by deleting that information from your hard drive. You plan to download the data later. However, you want to make sure that the external party cannot modify your content in the mean time without you knowing about the manipulation. How do you proceed? As of the property second preimage resistance of the cryptographic hash function it is not possible to generate the same hash with different contents. Therefore, if the external service manipulates your data, the hash changes. With that, manipulation can be detected if you store the hash instead of the data itself.



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

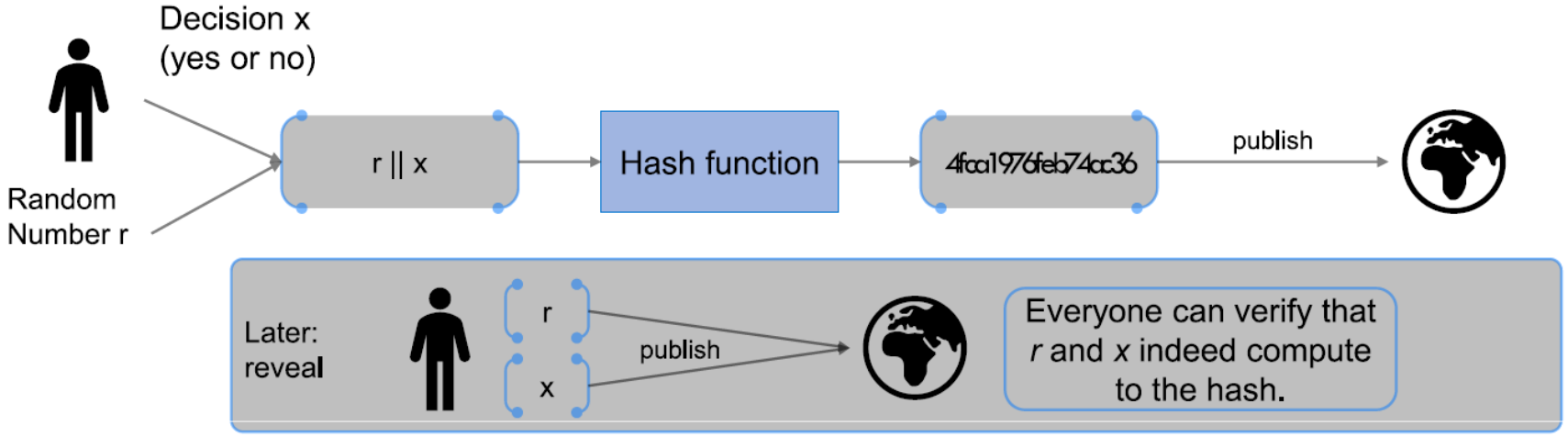
Application: Commitments

A person can commit to a value without revealing it immediately by using a hiding cryptographic hash function.

Definition

A commitment scheme consists of two algorithms:

- *com := commit(msg; nonce)* where msg is the message and nonce is the random number. The hash of the concatenation is returned.
- *verification := verify(com; msg; nonce)* where msg is the result of checking whether a given (msg; nonce) pair produces the same result as com.



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Application: Search Puzzle

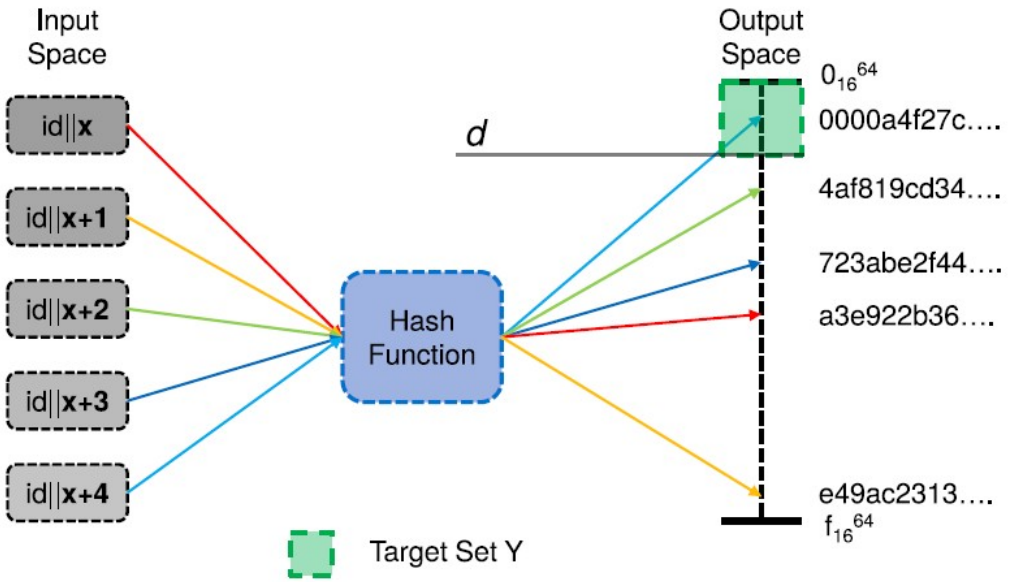
A search puzzle is a mathematical problem which requires searching a very large space in order to find a solution. In particular, there are no shortcuts in finding the solution. *h* has an *n* bit output, therefore it can take any of 2^n values. Solving the puzzle requires finding an input so that the output falls within the set *Y*. Depending on the size of the set *Y,* the puzzle can be more or less difficult, e.g., if the set contains all *n* bit strings it is trivial, if it contains only one string, it is maximally hard.

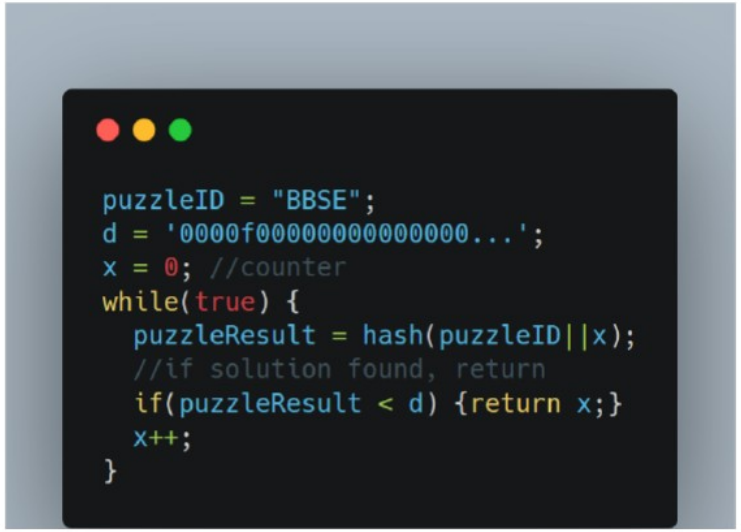
Definition

- A search puzzle consists of:
  - Cryptographic hash function *h*, used to compute the puzzle results
  - A value *id*, a puzzle-ID used to make the puzzle unique
  - A target set *Y*, which contains all valid solutions
  - Computation: *id* is concatenated with a value *x* and hashed. *x* changes until the puzzle result is in *Y.*

Application: Search Puzzle Visualized

While For simplicity we define the target set *Y* as *0, 1, …, d*, therefore we only have to check if the result of the hash function is smaller than the difficulty *d*. For our example, we use a puzzle-ID of "BSSE", but in practice it must not be known in advance as otherwise solutions can be recomputed The puzzle-ID needs to be different for each instance of the puzzle.





```
puzzleID = "BBSE";
d = '0000f00000000000000...';
x = 0; //counter
while(true) {
  puzzleResult = hash(puzzleID||x);
  //if solution found, return
  if(puzzleResult < d) {return x;}
  x++;
}
```

U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Cryptographic Hash Functions – SHA Family

There are many different cryptographic hash algorithms:

- Message Digest 4 / 5 (MD4 / MD5) Considered broken!
- Secure Hash Algorithm 1 (SHA-1) Considered broken!
- Secure Hash Algorithm 2 / 3 (SHA-2 / SHA-3) At the moment safe to use, favor SHA-3 over SHA-2.

The most important rule in cryptography: Never do your own crypto!
Use established libraries! (libsodium is pretty nice)

The SHA family

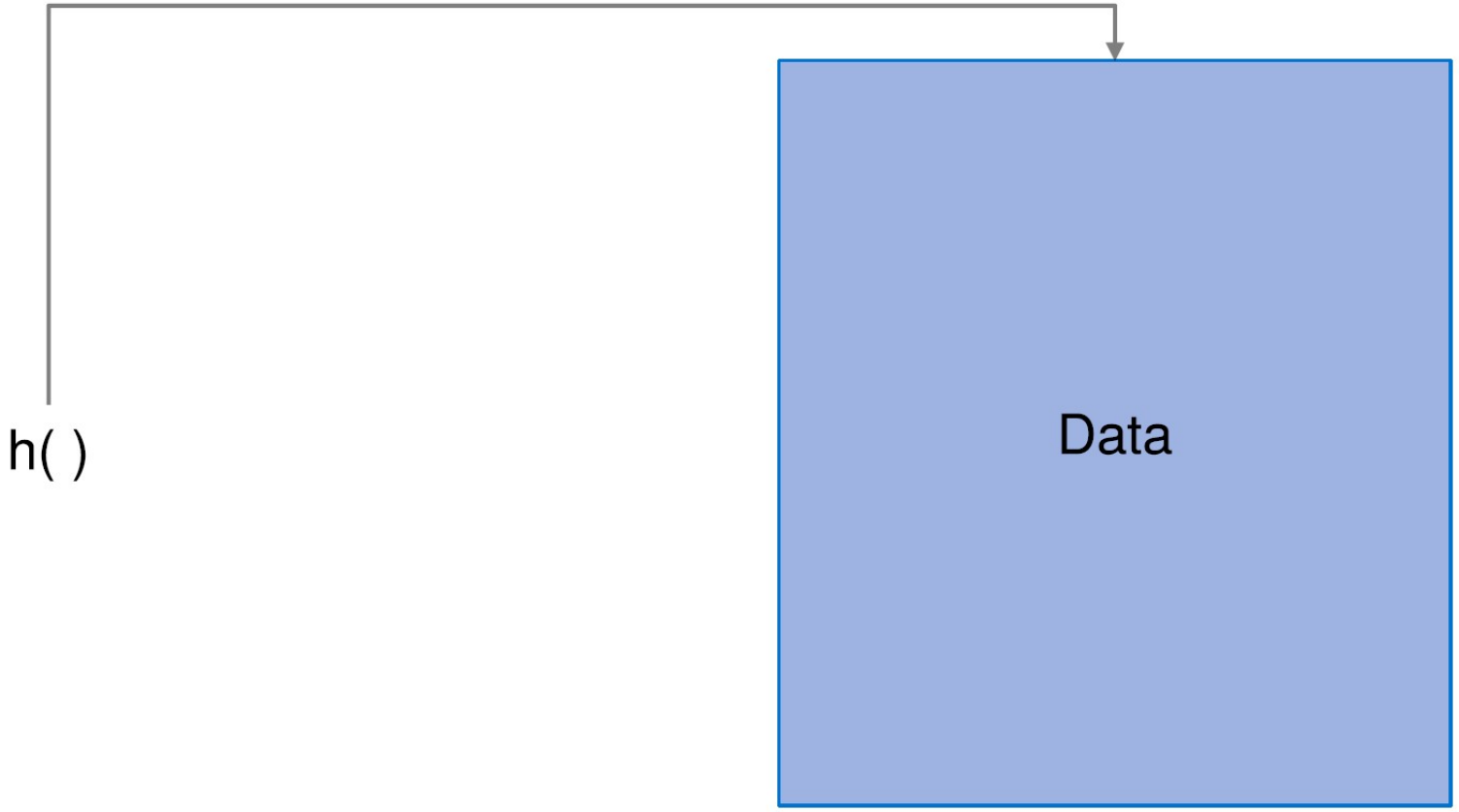
The SHA family describes a group of hash functions standardized by the National Institute for Standards and Technology (NIST). The SHA-1 and SHA-2 algorithms were developed by the NIST and the National
Security Agency (NSA). After the rst attacks on SHA-1 in 2004, the NIST started a competition to nd a new, more secure algorithm to become SHA-3. In 2012, Keccak was announced as the SHA-3 standard.
Keccak itself is not a single hash algorithm, but a family of hash algorithms with different parameters.

Data Structures

Hashes can be used to build different data structures. In the following, a few of them will be introduced.

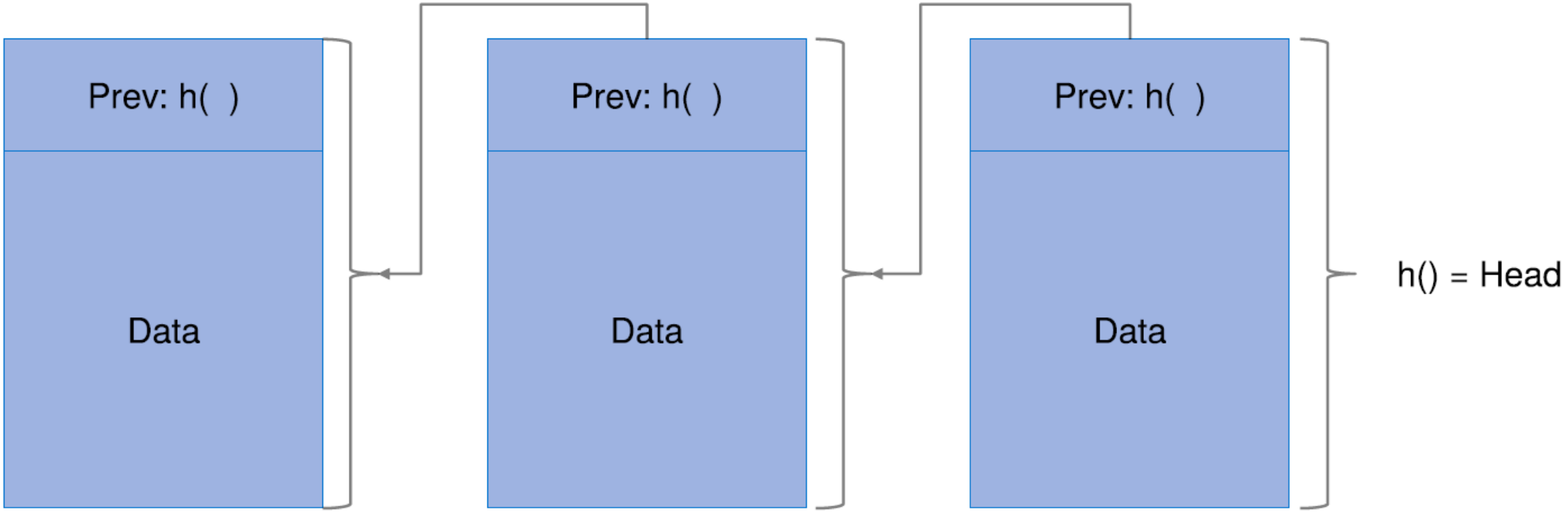
Hash Pointers

A hash pointer contains information about the location of data enriched with a cryptographic hash of it. The difference between a regular pointer and a hash pointer is, that a hash pointer allows verifying that the data has not changed.



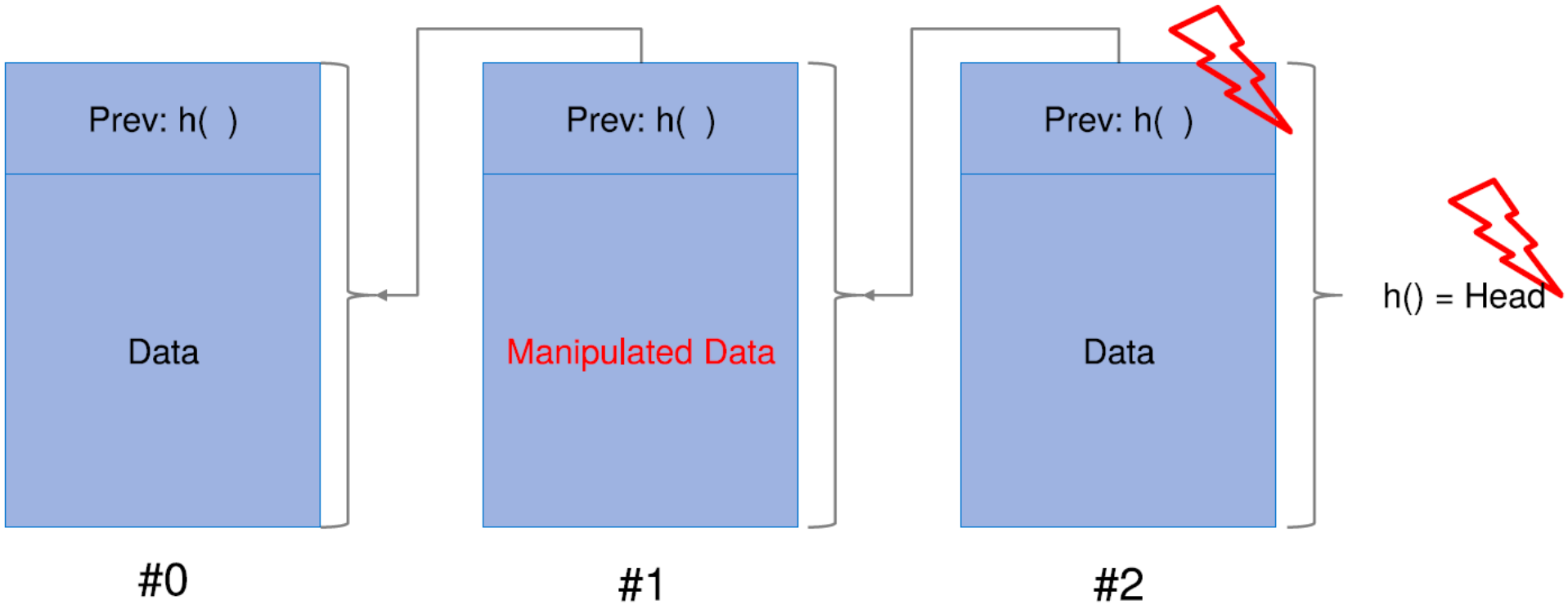
Blockchain

Here a linked list of hash pointers is shown. This is typically referred to as a blockchain. Instead of normal pointers, hash pointers are used. With this, the integrity of the complete blockchain can be ensured. If the information in any block is manipulated, the block's hash will change and no longer match the hash given in the block pointing to it. This means that just knowing the hash of the head of the blockchain is enough to verify the integrity of the entire chain.



Blockchain (cont.)

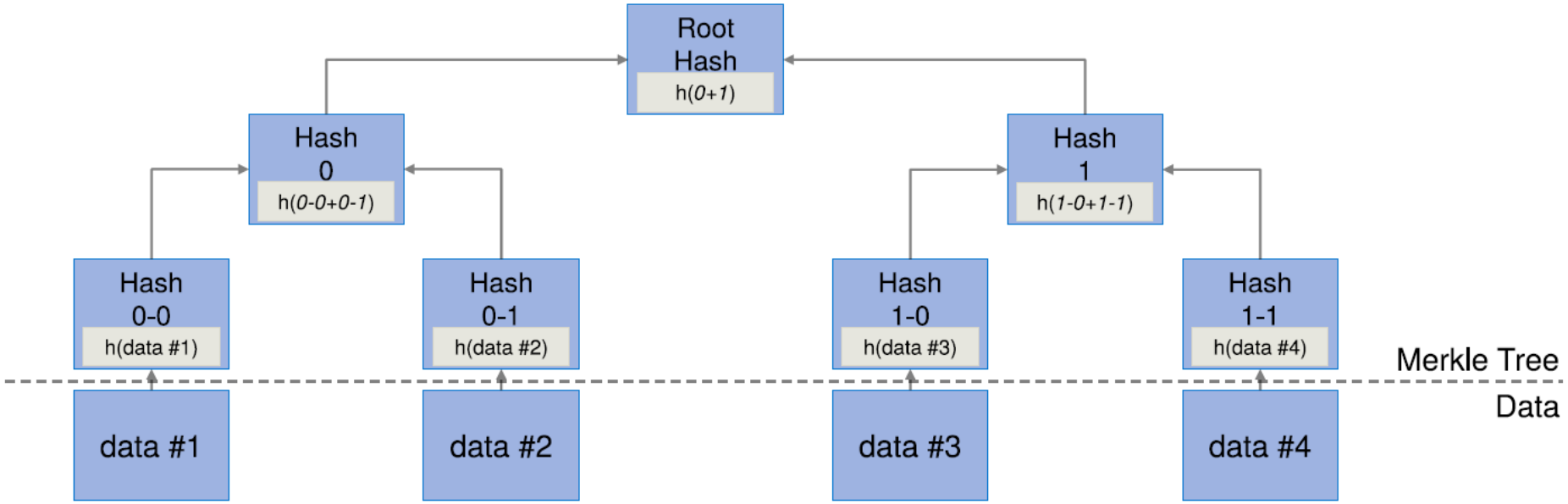
Here is an example of how a manipulation of block #1's data would be detected:



Merkle Trees

Merkle trees (sometimes referred to as hash trees) are a data structure using cryptographic hashes. They are used as an efficient and secure way to verify the integrity of large datastructures and provide short (O(log N)) proofs that certain nodes are indeed part of the tree (*proof of membership*).

In the case of sorted Merkle Trees, it can also be proven that certain nodes are not part of the Merkle tree *(proof of non-membership)*.



Merkle Trees – Proof of Membership

The proof of membership is what is most commonly used in blockchain applications. The problem solved by this is as follows:

We want to verify that a certain block of data is contained in the Merkle tree without hashing or even having all the data belonging to the tree. One scenario where this is useful is so-called Simple Payment

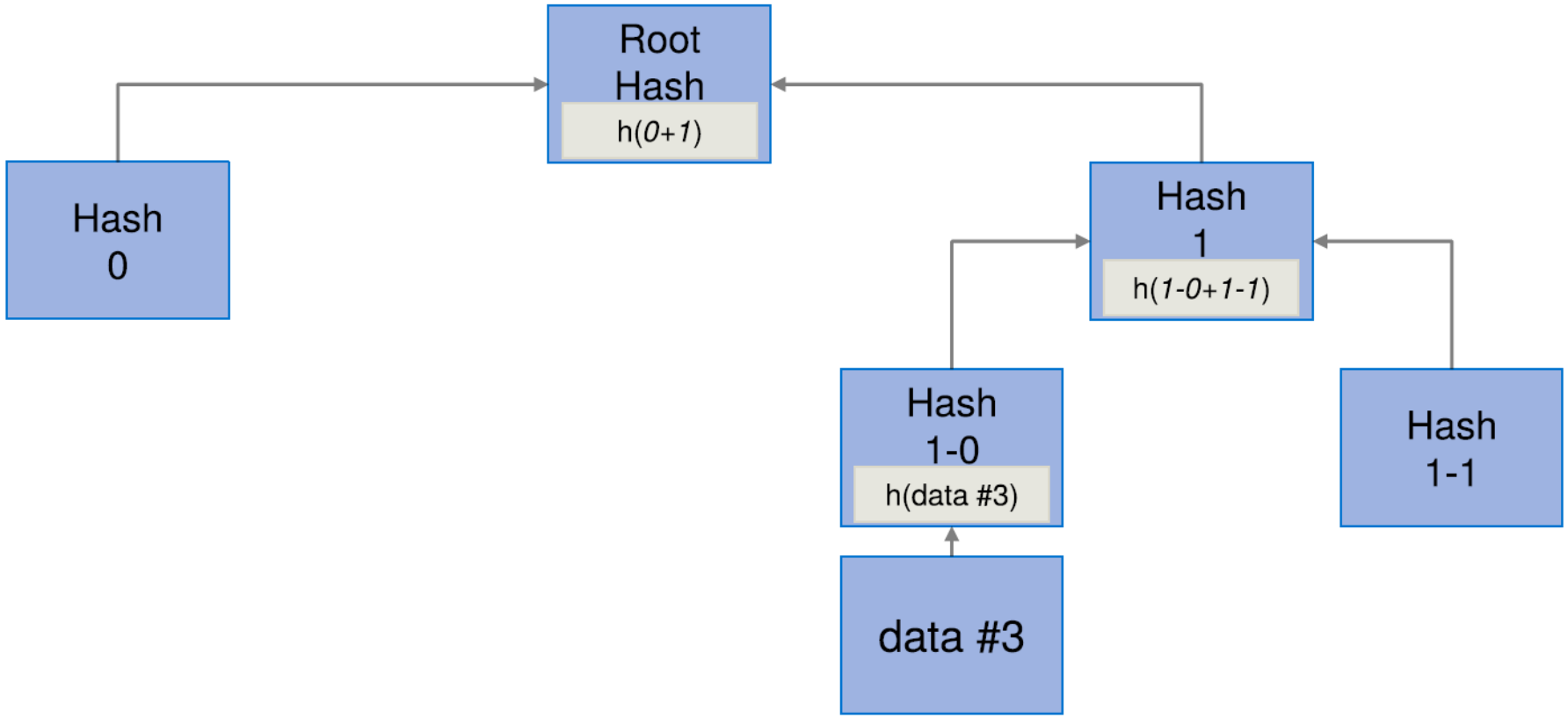
Verification (SPV) clients in the Bitcoin network. These clients only download block headers and know the hashes of the Merkle tree roots of each block. Using a proof of membership, these, often resource

constrained, light clients can validate the existence of a transaction on the blockchain.

We can do this by only verifying the hashes of nodes along the path from the node to be verified up to the root. This can be done win O(log N) time.

Merkle Trees – Proof of Membership (Example)

In this example, we can see how for a Merkle tree with four leaf nodes, the presence of node #3 can be verified using the hashes 0 and 1-1.



Bloom Filters

Bloom Filters do not have any direct relationship to cryptography or blockchains, but they can be useful for the implementation of blockchain related software. They allow heuristic checks on whether an element is in a given set S. Given a possible element e, the bloom filter will return one of two values:

- false or 0: The element e is definitely not in the set.
- true or 1: The element e might be in the set.

Due to the possibility of false positives, bloom filters cannot be used in every scenario.
Bloom filters can be implemented similar to hash tables. Rather than storing the elements however, it suffices to store a single bit per element. All bits are initialized as 0. For each element in S, the bit at
the corresponding position in the hash table will be set to 1.
To query the bloom filter, the bit at the location of the given element's hash is returned. As multiple elements may hash to the same value false positives may occur, but false positives are impossible.

Digital Signatures – Overview

Digital signatures are based on asymmetric cryptographic algorithms such as RSA or ECC. There are two properties that need to hold for digital signatures. First, only the entity holding the private key is able to create a signature, but everyone can verify it. Secondly, the signature is tied to a certain piece of data which gets signed. It cannot be used as a signature for different data.

Definition

A digital signature scheme consists of three algorithms:

- (sk; pk) := generateKeys(keysize) where sk is the secret key used to sign messages and pk is the public key distributed to everyone and used to verify the signature.
- sig := sign(sk; message) where sk is the secret key and message the message to be signed and sig is the output signature for the given message.
- isValid := verify(pk; message; sig) where pk is the public key corresponding to the secret key sk that signed the message. If sig is indeed a signature generated using sk for the message message, isValid is true, otherwise it is false.
  - In addition, the following conditions must hold:
    - verify(pk; message; sign(sk; message)) == true
    - Signatures are unforgeable

Unforgeability

Definition

Signatures are called unforgeable if under the following conditions, an attacker still cannot create a valid signature for any unseen message:

- The attacker knows the corresponding public key pk
- The attacker sees signatures sig signed for pk for an arbitrary amount of messages

Digital Signatures – Algorithms

Two major digital signature schemes are available:

- RSA based schemes such as RSA-PSS were invented in 1997 by Rivest, Shamir and Adleman and are based on the assumption that the factorization of large prime numbers after multiplication is very hard, but easy with additional information (so called trapdoor one-way functions)
- ECC based schemes such as ECDSA were suggested independently by Neal Koblitz and Victor S. Miller in 1985 and are based on discrete logarithms. Another variant called EdDSA with favorable properties for secure implementations and based on Edwards curves was proposed by Daniel J. Bernstein et al. in 2012[1].

Key sizes of at least 2048 bit for RSA and at least 256 bit for ECDSA are recommended.

Due to higher performance and smaller key sizes in ECC compared to RSA, Bitcoin uses ECDSA.

1: D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-speed high-security signatures," J. Cryptographic Engineering, vol. 2, no. 2, pp. 77{89, 2012. [Online]. Available: https://doi.org/10.1007/s13389-012-0027-1

Digital Signatures – Practical Concerns

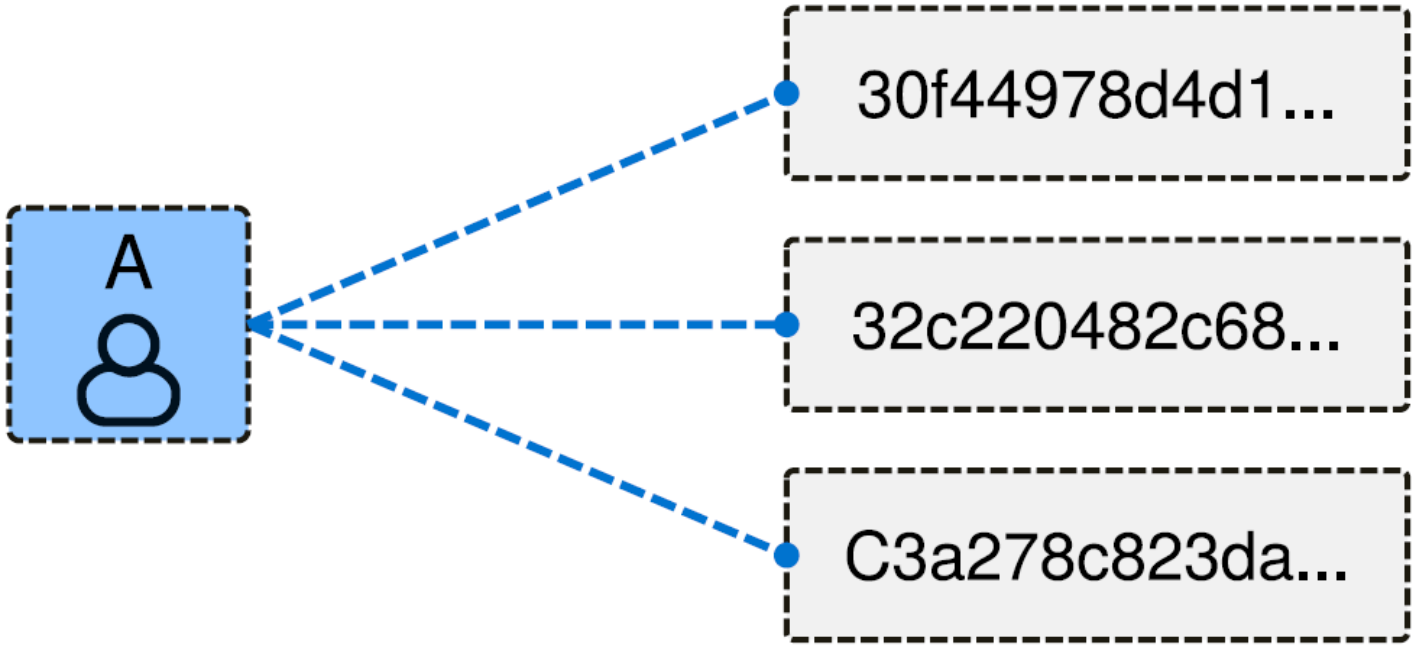
- Many signature algorithms are based on entropy. Without a strong source of entropy, private keys may be leaked.
- Digital signatures can only sign a small amount of data. However, as long as a secure cryptographic hash function is used, signing the hash of the message is sufficient.
- Private keys are not recoverable. If they are lost, it becomes impossible to act as the entity represented by the key. This may result in lost money, assets, etc.
- An appropriate key length should be used. If the key length is too short, the private key might get computed in the future.

Digital Signatures For Identity Creation

- Digital signature schemes can be used as identity systems where the public key acts as an identity and the private key is the "password" used to act on behalf of this identity.
- New identities can be generated at will using the generateKeys function and are at first not connected to any real-world identity and therefore pseudonymous, although acting under the identity may intentionally or unintentionally leak information about its connection to the real-world identity behind it.
- Hashing public keys of identity may provide certain benefits, such as reducing their size and reducing their exposure to possible quantum computing attacks.
- To validate a statement for a hashed public key, the public key has to be attached to the statement. The verifier can then validate that the public key hashes to that of the identity and that the message signature verifies with respect to the provided public key.

Decentralized Identity Management

- This approach allows a decentralized identity management without a need for registering identities at a central authority, allowing each user to create an arbitrary amount of identities while being simple to verify
- All cryptocurrencies / blockchain based systems handle it this way
- The address is (in Bitcoin and Ethereum) the hash of a public key



U. Gallersdörfer, P. Holl, and F. Matthes, “Blockchain-based Systems Engineering – Lecture Slides,” Oct 2019. [Online]. Available: https://github.com/sebischair/bbse

Digression: Quantum Resistance of Signature Schemes and Hash Functions

Digital signature schemes based on the integer factorization problem, the discrete logarithm problem or the elliptic curve discrete logarithm problem can be solved with Shor's algorithm on sufficiently powerful quantum computers[1].

Cryptographic hash functions are considered to be relatively secure against quantum computers[2], but hash based signature schemes suffer from slow speeds and big signature sizes.

What are the implications?

- Can decentralized identity management work in a post-quantum World?
- Can Bitcoins be stolen using quantum computers?
- How can we prevent this?

1: D. J. Bernstein, "Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete," SHARCS, vol. 9, p. 105, 2009. [Online]. Available: https://cr.yp.to/hash/collisioncost-20090517.pdf
2: P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM J. Comput., vol. 26, no. 5, p. 1484{1509, Oct. 1997. [Online]. Available: https://doi.org/10.1137/S0097539795293172

Can decentralized identity management work in a post-quantum world?

The fact that Bitcoin addresses are hashed public keys gives them some degree of resistance against quantum computer attacks.

- Hashing itself is not broken by quantum computers
- As long as only the hashed public key of a bitcoin address is known, it is computationally infeasible to calculate the private key and nothing can be stolen
- As long as quantum computers take longer to break a public key than it takes for a bitcoin transaction to be confirmed, the system remains usable, as long as public keys are never reused

In today's Bitcoin, address reuse is considered bad hygiene. In a post-quantum world, it will get your funds stolen.

Further Resources

- Bitcoin Whitepaper – Link
- Ethereum Whitepaper – Link
- Blockchain für Entwickler – Grundlagen, Programmierung, Anwendung – Link
- Architecture for Blockchain Applications – Link
- The Path to Self-Sovereign Identity – Link
- A Review on Consensus Algorithm of Blockchain – Link
- Blockchain Consensus Protocols in the Wild – Link
- Ethereum for Developers – Link
- Blockchain-based Systems Engineering – TU Munich – Link

Questions?