

AI: Internet Computing

Lecture 9 — Distributed Ledger Technology

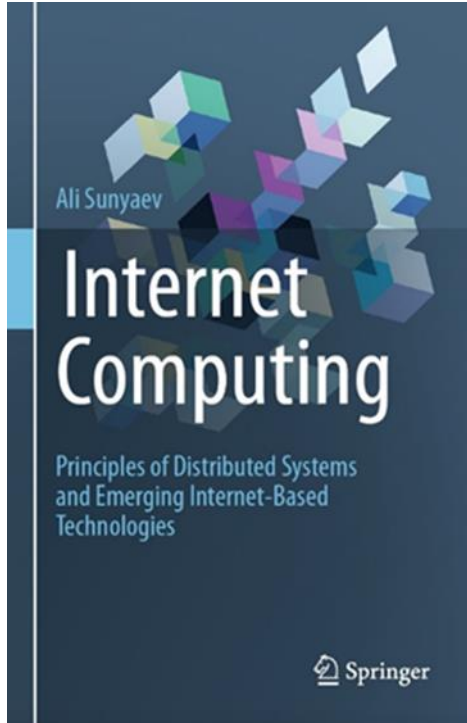


Lecture Slides for AI: Internet Computing © 2022 by [Dr. Ali Sunyaev](#) is licensed under [CC BY-NC-ND 4.0](#)

Learning Goals of the Lecture

- Gaining a basic understanding of Distributed Ledger Technology (DLT), its historical background, and associated possibilities of applications
- Understanding technical foundations and the functionality of DLT in general and of the blockchain
- Learning about the idea and the use of smart contracts
- Understanding the usefulness of DLT in real-world application areas and gaining the ability to explain the role of DLT in these use cases

Reference to the Teaching Material Provided



Chapter 9 Distributed Ledger Technology



Abstract

Distributed Ledger Technology (DLT) is one of the most promising innovations in the field of information technologies with the potential to change organization and collaboration in the economy, society, and industry. This chapter introduces the technical background and use cases of distributed ledger technology. It presents the major innovations originating from distributed ledger technology since the introduction of the blockchain concept. Furthermore, cryptocurrencies' historical background as a driver of fully decentralized distributed ledgers is outlined from their origins in the 1990s until the blockchain concept's introduction in 2009. DLT's technical principles are introduced to provide a sound understanding. Subsequently, the functioning of distributed ledger technology is illustrated by means of the Bitcoin blockchain example, which was the first fully decentralized cryptocurrency to not require a trusted authority (i.e., banks). Thereafter, smart contracts and the idea of decentralized applications are explained. Selected use cases for distributed ledger technology's application are subsequently discussed. This chapter concludes with a discussion of the prevailing challenges in the field of distributed ledger technology.

Learning Objectives of this Chapter

1. Understand the basic concepts of Distributed Ledger Technology (DLT)

The Distributed Ledger Technology

Deloitte's 2020 Global Blockchain Survey

Views of Organization Are Changing

It will be critical and in our top-five strategic priorities



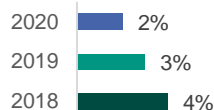
It will be important but not in our top-five strategic priorities



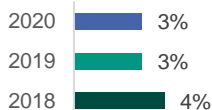
It will be relevant, but it's not a strategic priority



It will not be relevant



We haven't reached a conclusion



Source: Deloitte, 2020, Deloitte's 2020 Global Blockchain Survey – From promise to reality, https://www2.deloitte.com/content/dam/insights/us/articles/6608_2020-global-blockchain-survey/DI_CIR%202020%20global%20blockchain%20survey.pdf

Bitcoin Price History

Bitcoin to Euro Chart



Source: [Bitcoin price chart](#) by Coinmarketcap

Distributed Ledger Technology — A Game Changer

Definition

Distributed Ledger Technology (DLT) enables the realization and operation of **distributed ledgers**, where **benign nodes**, through a **shared consensus mechanism**, agree on an (almost) immutable record of transactions in the presence of **Byzantine failures** and eventually achieve consistency.



In order to fully understand this definition let's have a look at the **technical** and terminological foundations first!

Image source: [\[Questions\]](#) by Peggy and Marco Lachmann-Anke, November 4th 2015. [Pixabay License](#).

Centralized, Decentralized, Distributed (1/2)

Relational Databases

- Clearly defined tables
- Dependencies between those tables
- Four Operations: Create, Read, Update, Delete

Non Relational Databases

- No tabular scheme
- ...

Centralized Databases

- Reside on single storage device
- + Maintenance
- Availability
- Performance

Decentralized Databases

- No central storage
- Data is stored on multiple storage devices
- Devices are connected but located differently
- Hierarchical structure where one set of nodes communicates with a particular node

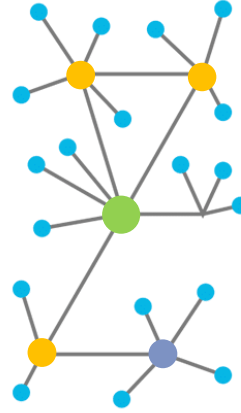
Distributed Databases

- + Increased availability
- + Higher performance
- Replications of the data are stored across multiple physically independent storage devices
 - In case a storage device of distributed database cannot be reached for a period other devices are still operating and can respond to open requests
- No hierarchical structure

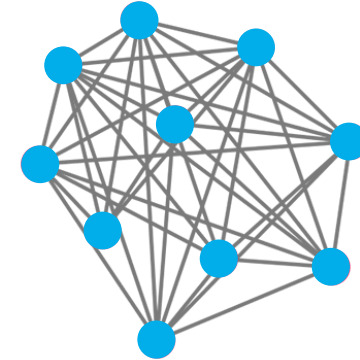
Centralized, Decentralized, Distributed (2/2)



Centralized Network
(e.g., Web-Services)



**Partially
(De-)Centralized Network**
(e.g., Domain Name Systems)



Decentralized Networks
(e.g., Peer-to-Peer Networks)

Degree of Decentralization



● Client ● Server (3rd Level) ● Server (2nd Level) ● Server (1st Level)

Figures align with Baran P. (1964) On Distributed Communications Networks. IEEE Transactions on Communications Systems 12(1):1–9.

Distributed Ledger Technology — Basics (1/4)

Definition

A **distributed database** is a type of database where data is stored on multiple storage devices (nodes).

- The **CRUD** operations should **always return the same results**, no matter on which node the operation is performed!
- To reach **consistency** the stored data must be **synchronized** between the nodes of a distributed database → nodes have to communicate with each other

CRUD: Create, Read, Update, Delete

Definition

A **consensus mechanism** is designed to achieve an agreement on a single state of a data value (e.g., the replication of stored data) among nodes of a distributed database under consideration of network failures.

- The algorithms and protocols that manage the synchronization between nodes are called **consensus mechanisms**.
 - **Probabilistic** consensus mechanism: current state is assumed and can be changed post hoc
 - **Finality-preserving** consensus mechanism: nodes agree on particular state which cannot be reverted

Definition

A **Byzantine fault** is a **condition** of a particularly distributed computer system, where components may fail and there is **imperfect information** on whether a component has failed.

- There are multiple forms of Byzantine faults (1)-(3)
 - 1) A node can be determined as **crashed** or **not reachable** over the network as the node does **not respond** anymore
 - 2) A monitoring system may **not be able to determine** the status of a certain node, which can occur when the **node crashes** or network failures result in **inconclusive responses** from the node.
 - 3) Nodes may follow **malicious intentions** such as trying to **store incorrect data** into the distributed database.

Distributed Ledger Technology — Basics (4/4)

Definition

A **distributed ledger** is a type of distributed database that assumes the presence of nodes, which have malicious intentions. A distributed ledger incorporates multiple replications of a ledger, where data can only be appended or read.

- Through the introduction of the Bitcoin blockchain a **central authority** is **not required** to administrate the content and distinguish between honest and malicious nodes
→ **Anyone** can contribute to the distributed ledger and assure that stored data is not corrupted!

Distributed Ledger Technology — A Game Changer

Definition

Distributed Ledger Technology (DLT) enables the realization and operation of **distributed ledgers**, where **benign nodes**, through a **consensus mechanism**, agree on an (almost) immutable record of transactions in the presence of **Byzantine faults** and eventually achieve consistency.

- The reliable **synchronization** of a **dynamically-changing** set of nodes of a distributed ledger in the presence of all types of Byzantine failures is one of the **main innovations** of **DLT**.

...but why do we need Distributed Ledger Technology!?

Distributed Ledger Technology — A Game Changer

- DLT renders a **trusted third party** obsolete for certain services:
 - **Cryptocurrencies** (e.g., Bitcoin, Ethereum) provide the ability to own **digital assets** without the need for institutions such as banks.
 - Data that go **beyond values of assets** can be stored in distributed ledgers, for example, **smart contract** that can be used to formalize business processes.
 - Other areas of use: identification, medical prescriptions, ...

...so, how did all this get started?

Distributed Ledger Technology — History (1/4)


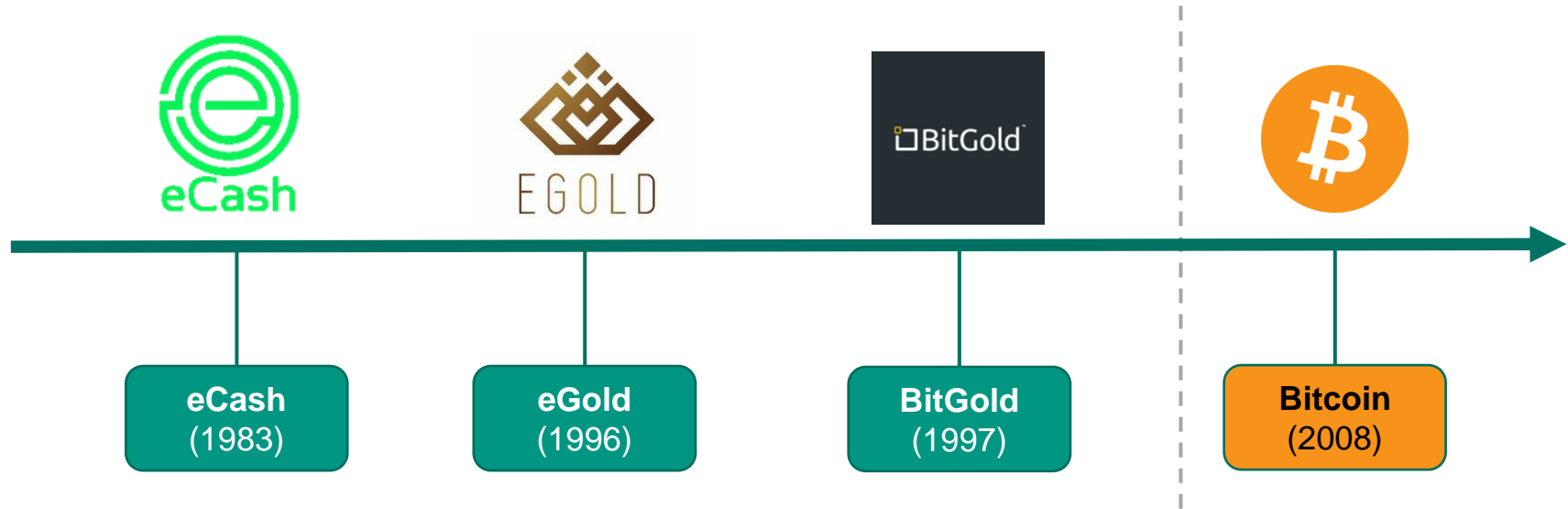
- Distributed Ledger Technologies (DLT) became popular with the public due to the rise of the cryptocurrency **Bitcoin** (2008) 
- However, inspired by the idea of an anonymous, digital voting system **Dr. David Chaum** came up with the idea of cryptocurrencies in the 1980's
 - David Chaum invented the so-called **blind signatures**
 - David Chaum founded the company DigiCash in 1989 and created the cryptocurrency **eCash**

Image source: [\[Bitcoin\]](#) by WikimediaImages, September 7th 2015. [Pixabay License](#).

Distributed Ledger Technology — History (2/4)



Distributed Ledger Technology — History (3/4)

- **Bitcoin** (2008) – considered as the DLT generation 1.0
 - Build upon a distributed ledger (underlying infrastructure is architecturally distributed and logically centrally)
- **Ethereum** (2015) – allows for the execution of Turing-complete smart contracts on a DLT design
 - Not limited to be used as a cryptocurrency but enables the development of applications
 - Introduction of Initial Coin Offerings (ICOs)
 - Popular example: the game CryptoKitties
- DLT designs are meanwhile applicable for domains such as **access management, identity management, and logging**

Distributed Ledger Technology — History (4/4)

Blockchain 1.0

- Bitcoin Blockchain
- Solution of the *Double-Spending* problem
- Focus on digital currencies

[1]



Blockchain 3.0

- Use of directed, acyclic graphs (DAG)
- Improvements in performance, security, and confidentiality
- Development of applications on DLT



Blockchain 2.0

- Turing complete Smart Contracts
- Additional flexibility
- Initial Coin Offerings (ICOs) are possible



Blockchain-as-a-Service

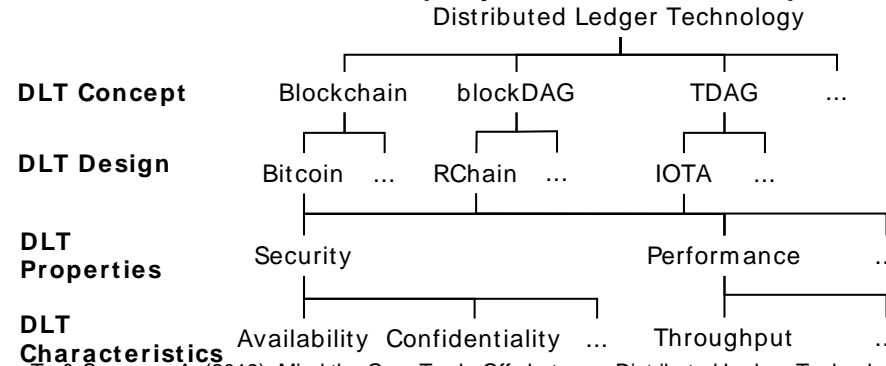
- DLT designs provided as a cloud service (SAP Leonardo, IBM Watson IoT™, Microsoft Azure, HyperLedger Fabric)



Image source [1]: [\[Bitcoin\]](#) by WikimediaImages, September 7th 2015. [Pixabay License](#).
Image source [2]: [\[Ethereum\]](#) by Muhammad Salman, May 24th 2021. [Pixabay License](#).

Terminology in DLT (1/3)

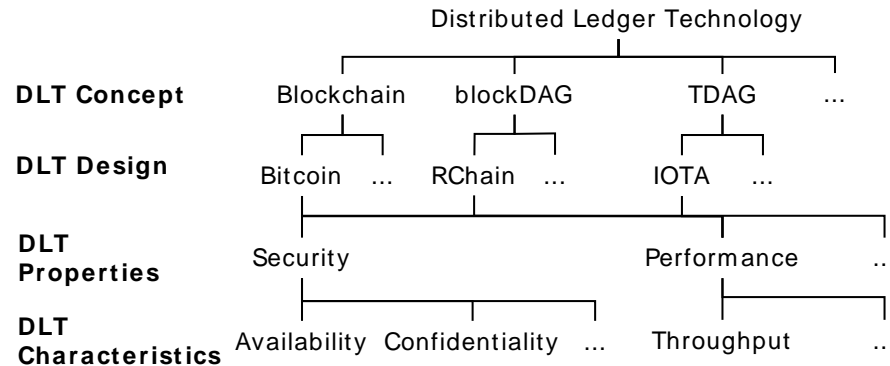
- A **DLT concept** provides description of the architecture of a DLT and the organization of transactions
 - **Block-based** directed acyclic graphs (blockDAG)
 - **Transaction-based** directed acyclic graphs (TDAG)
- A **DLT design** is the concrete implementation of a DLT concept
 - For example: Bitcoin and Ethereum employ the DLT concept blockchain



Source: Kannengießer, N., Lins, S., Dehling, T., & Sunyaev, A. (2019). Mind the Gap: Trade-Offs between Distributed Ledger Technology Characteristics. ArXiv:1906.00861 [Cs]. Retrieved from <http://arxiv.org/abs/1906.00861>

Terminology in DLT (2/3)

- A **DLT property** forms a group of related DLT characteristics, which is featured by all DLT designs
- A **DLT characteristic** is a granular component of a DLT property, which can be used to define application requirements for DLT



Source: Kannengießner, N., Lins, S., Dehling, T., & Sunyaev, A. (2020). Trade-offs between distributed ledger technology characteristics. *ACM Computing Surveys (CSUR)*, 53(2), 1-37.

Terminology in DLT (3/3)

DLT Properties:

| |
|--|
| Community |
| A group of individuals who have a common interest in using and/or maintaining a DLT design. |
| Flexibility |
| The degree of technical freedom to customize a DLT design and to deploy applications on a DLT design. |
| Law & Regulation |
| The ability of authorities to enforce compliance of a DLT design with legal and regulatory requirements |
| Transparency |
| The perception of an individual of being informed about the relevant actions and characteristics of another party who uses the DLT design. |
| Performance |
| The accomplishment of a given task on a distrusted ledger measured against targets for accuracy, completeness, cost, and speed. |
| Security |
| The preservation of confidentiality, integrity, and availability of data stored on a distributed ledger. |
| Usability |
| The extent to which DLT design users can achieve their goals with respect to effectiveness, efficiency, and satisfaction in their use contexts |

Exemplary DLT Characteristics of the DLT Property Performance:

- Block creation interval
- Confirmation latency
- Scalability
- Throughput
- ...

Before we introduce the functioning of a traditional blockchain, we need to understand certain...

...Technical Foundations

Technical Foundations: Permissioned & Permissionless DLT Designs

i Private and public DLT designs usually require different consensus mechanisms

| | Permissioned | Permissionless |
|---------|---|--|
| Public | <ul style="list-style-type: none">■ Public network■ Only few trusted nodes validate transactions■ Anybody can join the DLT design | <ul style="list-style-type: none">■ Public network■ Anybody can join the DLT design■ Anybody can participate in mining, and validation |
| Private | <ul style="list-style-type: none">■ Private network■ Foreign parties cannot access the DLT design■ Only few trusted nodes validate transactions | <ul style="list-style-type: none">■ Private network■ Any included device is allowed to access (read, write), and validate transactions |

Technical Foundations

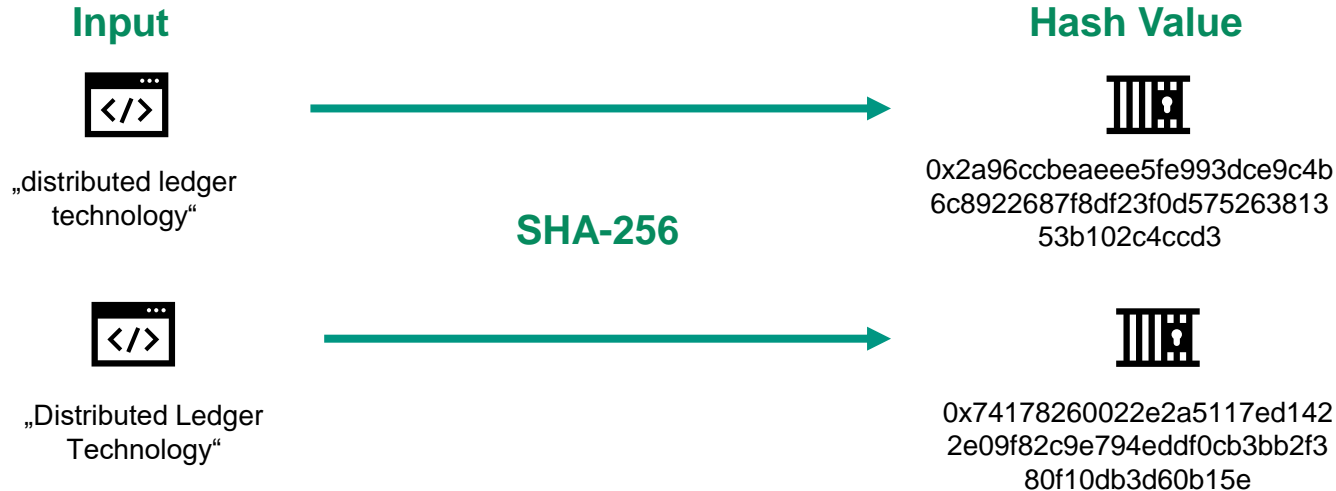
- DLT incorporates multiple **techniques** from **computer science** such as
 - Hashing
 - Public key cryptography
 - Distributed systems

Technical Foundations: Hash Functions

- **Hash functions** are injective functions that are used to map data blocks of a **arbitrary** size to data blocks of a **defined** size (e.g., 160 bit) – so called **hash values**
 - Hash functions enable the use of **digital signatures** (e.g., **proof ownership** of a certain amount of assets or **verify authenticity** of digital messages)
 - Hash functions need to be **deterministic** and the probability for each hash value is ideally **uniformly distributed**
 - Hash functions must not reveal the original data
 - $h = \text{hash}(d)$ reconstructing the data value **d** has to be extremely difficult
- The harder it is to reconstruct the original message **d** (pre-image) the more secure is the hashed data (collision resistant)

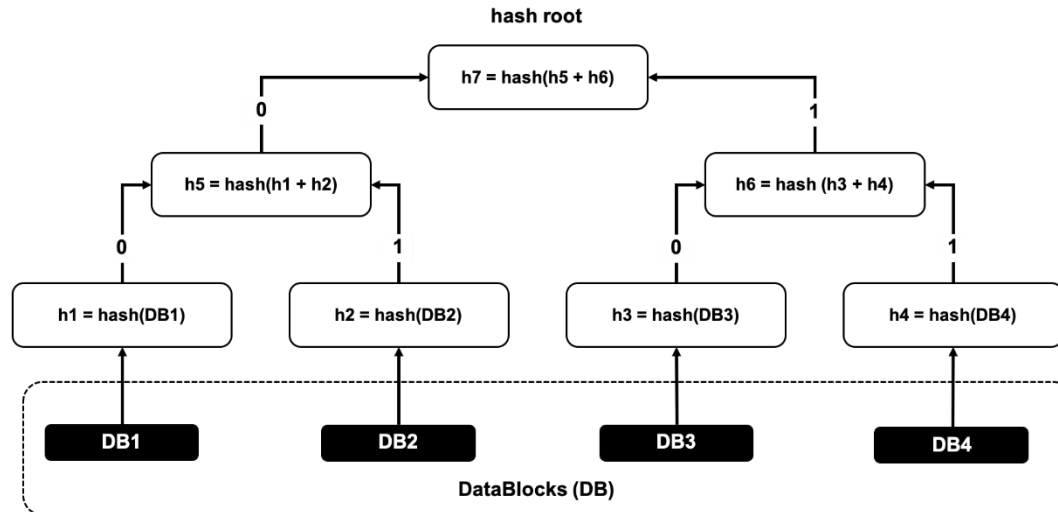
Technical Foundations: Secure Hash Algorithms

- Multiple **Secure Hash Algorithms (SHA)** have been developed (e.g., SHA-1, SHA-2 and SHA-3)
 - Each SHA makes use of **different** hash functions
 - The most used hash algorithms are **SHA-256** and **SHA-512**



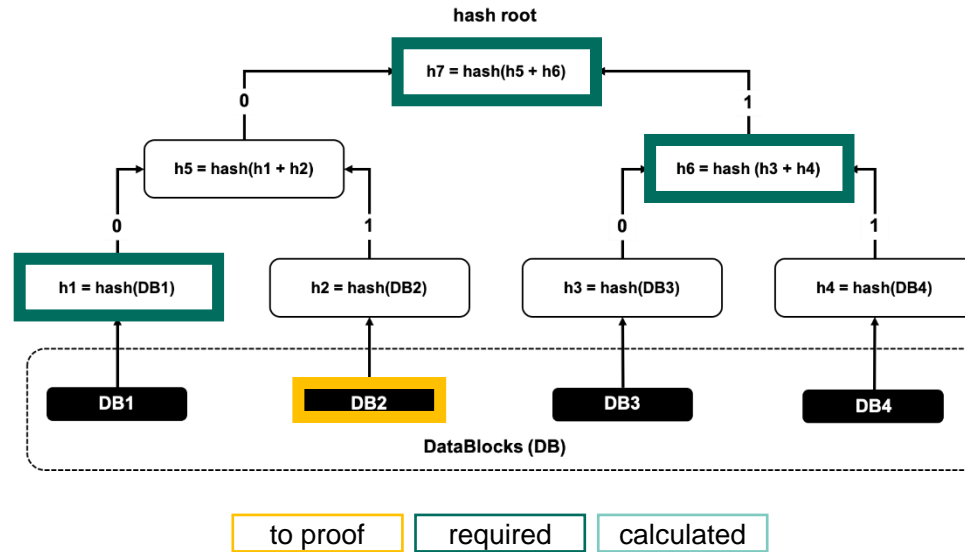
Technical Foundations: Merkle Tree

- **Merkle trees** can be used for **efficient verification** of **integrity** of data blocks
 - The structure of Merkle trees is based upon hash values
 - Merkle trees are used in **peer-to-peer networks** such as Bitcoin and Tor network
 - Advantage: It is not necessary to know the entire Merkle tree to verify a data block's integrity



Technical Foundations: Merkle Tree Example (1/2)

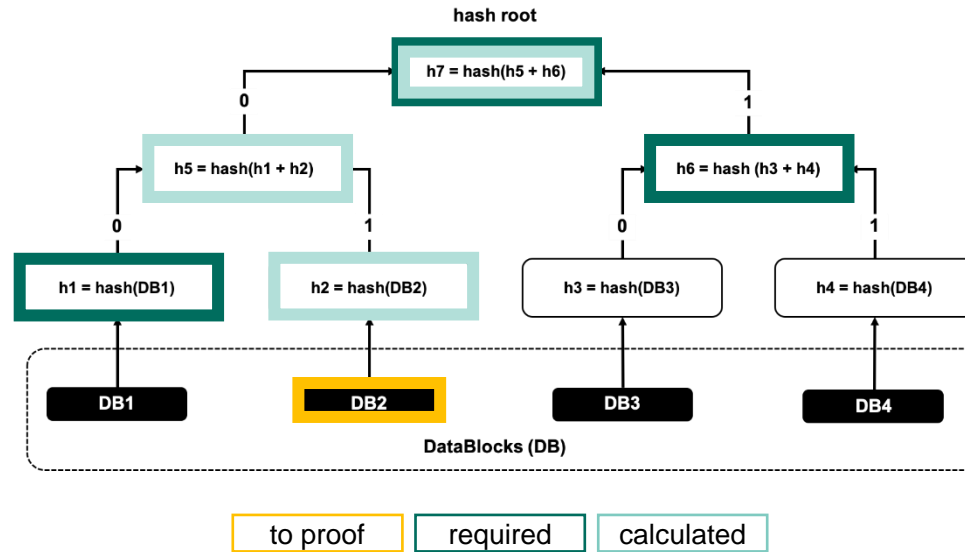
- **Merkle trees** can be used for **efficient verification** of **integrity** of data blocks
 - Example: Proof of integrity of *DB2* by comparing the calculated hash root with the original hash root *H7*



Technical Foundations:

Merkle Tree Example (2/2)

- **Merkle trees** can be used for **efficient verification** of **integrity** of data blocks
 - Example: Proof of integrity of *DB2* by comparing the calculated hash root with the original hash root *H7*

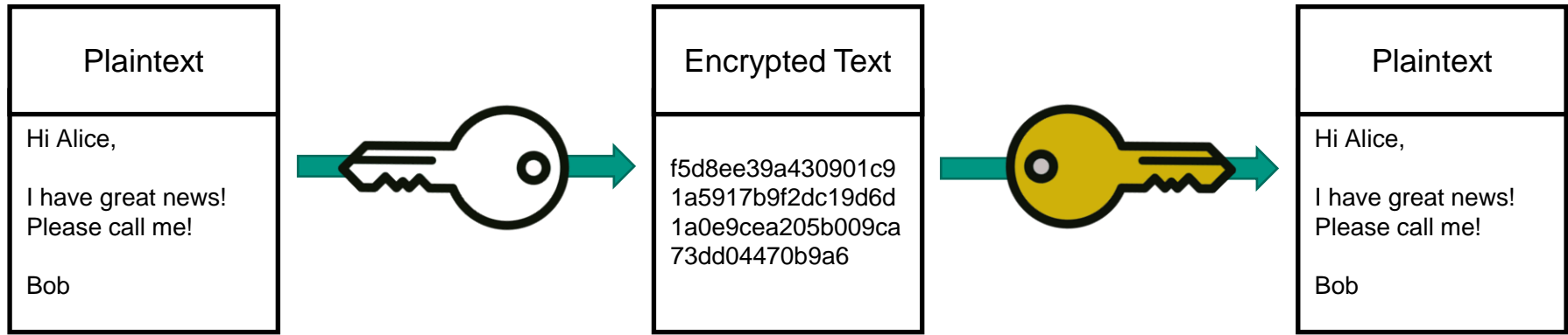


Technical Foundations: Public Key Cryptography (1/5)


- **Public Key Cryptography** can be used to encrypt data **before** the data is sent over the network
 - Data is **encrypted** using the data receiver's **public key** which is retrieved from the repository of the PKI
 - The encrypted data can only be **decrypted** using the **private key** matching the public key that was used for encryption

$$\text{encrypt}(\text{dataBlock}, \text{publicKey}_{\text{receiver}}) \rightarrow \{\text{encryptedDataBlock}\}$$
$$\text{decrypt}(\text{encryptedDataBlock}, \text{privateKey}_{\text{receiver}}) \rightarrow \{\text{dataBlock}\}$$

Technical Foundations: Public Key Cryptography (2/5)



 Alice's Private Key

 Alice's Public Key

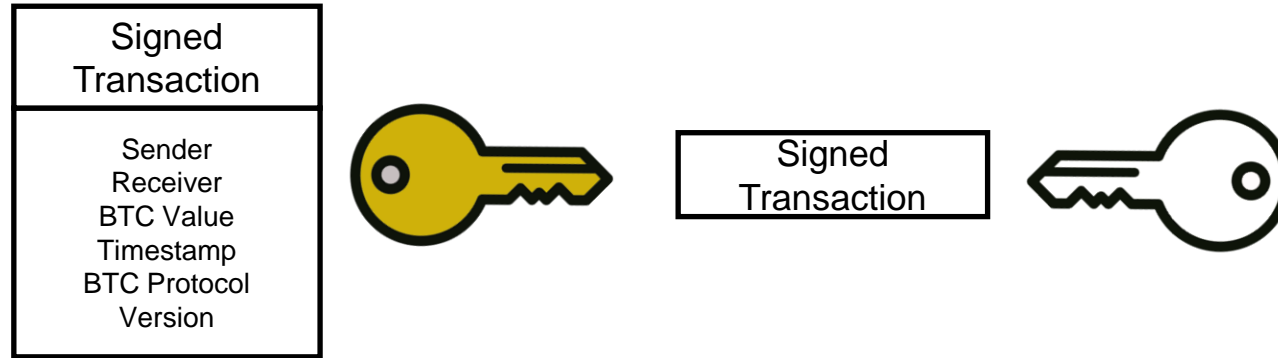
Technical Foundations: Public Key Cryptography (3/5)

- **Digital Signatures** can be used to deliver a proof that a certain entity has sent a particular data block
 - Digital signatures are also used in, for example, emails to proof the identity of the sender

$sign(dataBlock, privateKey) \rightarrow \{signature\}$

$verify(message, publicKey, signature) \rightarrow \{true, false\}$

Technical Foundations: Public Key Cryptography (4/5)

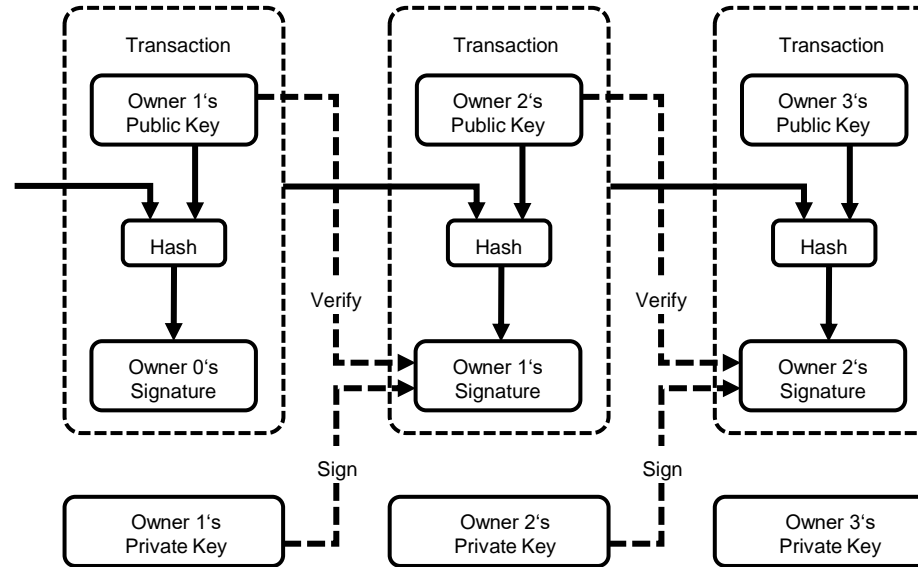


 Private Key

 Public Key

Technical Foundations: Public Key Cryptography (5/5)

i In DLT, digital signatures are used to authorize and link transactions.

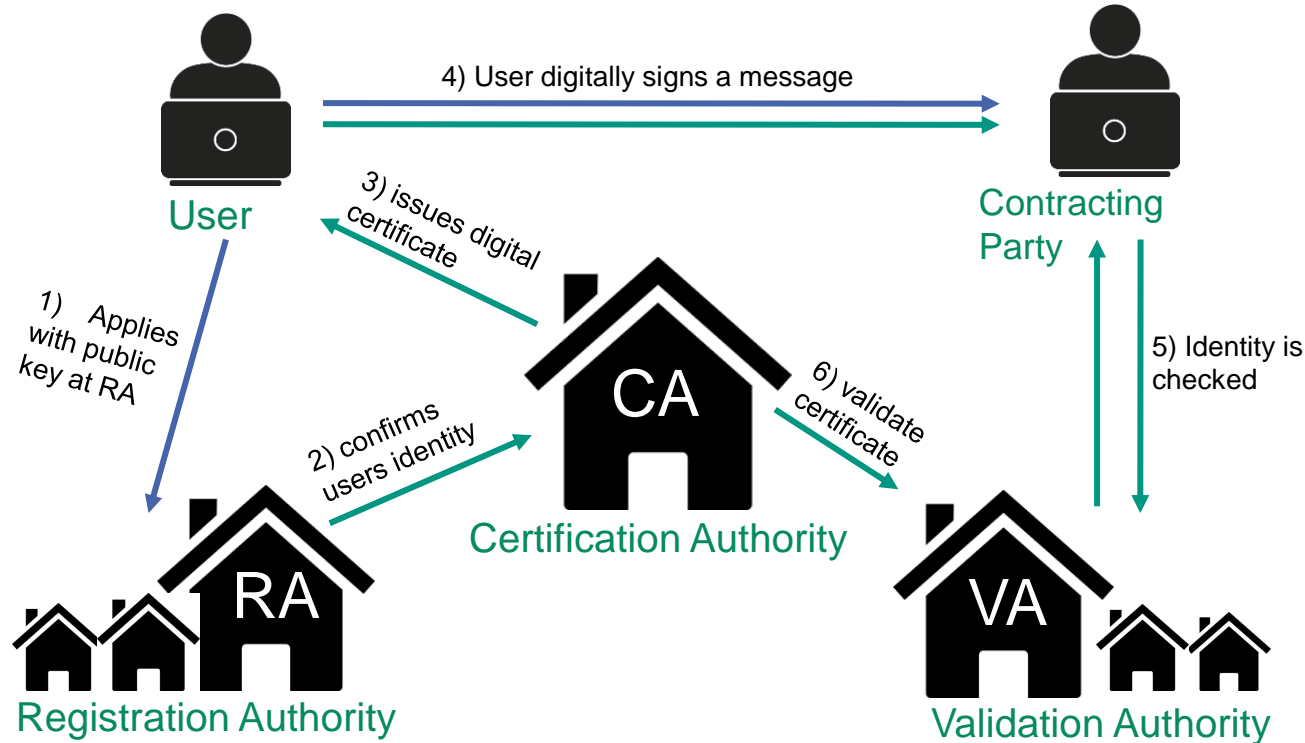


Source: Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Received from <https://bitcoin.org/bitcoin.pdf>

How are the keys distributed?

- Several permissioned DLT Designs draw from the traditional concept of a **Public Key Infrastructure (PKI)** (e.g., Hyperledger Fabric)
- Unpermissioned DLT designs predominantly apply a distributed PKI
- A **PKI** comprises hardware, software, policies, procedures and roles used for the **secure electronic transfer** of data over an **insecure network** such as the Internet
 - A PKI manages the creation, distribution, and revocation of digital certificates which is required for the use of public key cryptography

Technical Foundations: Public Key Infrastructure for Digital Signatures



Technical Foundations: Consensus Mechanisms in DLT

Definition

A **consensus mechanism** is designed to achieve an agreement on a single state of a data value (e.g., the replication of stored data) among nodes of a distributed database under consideration of network failures.

Proof of Work (PoW):

- Originally invented to reduce spamming (Jakobsson & Juels 1999)
- Forms a foundation for consensus mechanisms applied to most unpermissioned DLT designs (e.g., Bitcoin, Ethereum)
 - Each node must solve a computationally **hard challenge** before new transactions are included in the ledger
 - The challenge is to find a random word (nonce) that is linked to the block to be published and whose hash value is less than a given target t
 - In Bitcoin, t defines the minimum number of preceding zeros such hash value has to provide



PoW is energy consuming and inefficient! Therefore, alternative consensus mechanisms have been developed, which have a shorter confirmation latency and are more resource efficient.

Source: Jakobsson, M., & Juels, A. (1999). Proofs of Work and Bread Pudding Protocols. Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security, 258–272. Dordrecht, The Netherlands: Kluwer, B.V.

Technical Foundations: Energy Consumption of Proof of Work



Image source: Brandt, M. (2017, February 22nd). Blockchains: Supercomputer-Leistung für die Sicherheit [Digitales Bild]. Last view 3rd of July 2018, from <https://de.statista.com/infografik/8210/daten-zu-bitcoin-und-blockchains>

Technical Foundations: Alternative Consensus Mechanisms (1/2)

Proof of Stake (PoS)

- In PoS-based cryptocurrencies the creator of the next block is chosen via various combinations of **random selection** and **wealth or age of stake**
- Less energy consuming substitute for PoW
- Nodes can mine a new block depending on their stake
 - How **many coins they hold and how long** they already held the coins



PoS consensus mechanisms are more centralized compared to PoW consensus mechanisms and are vulnerable to *nothing at stake* attacks, where nodes randomly publish blocks to corrupt the distributed ledger.

Technical Foundations: Alternative Consensus Mechanisms (2/2)

Practical Byzantine Fault Tolerance (PBFT)

- Was presented by Castro & Liskov (1999)
- Is used in **private, permissioned** DLT designs (unlike PoW and PoS)
- Enables the implementation of high-performance Byzantine fault-tolerant replicated state machines (RSM).
- Handles f Byzantine faults in a system with **$3f+1$ nodes**

What is a Byzantine fault?

A Byzantine fault f is a **condition** of a particularly distributed computer system, where components may fail and there is **imperfect information** on whether a component has failed.



PBFT does not scale horizontally and is only applicable for a small set of nodes.

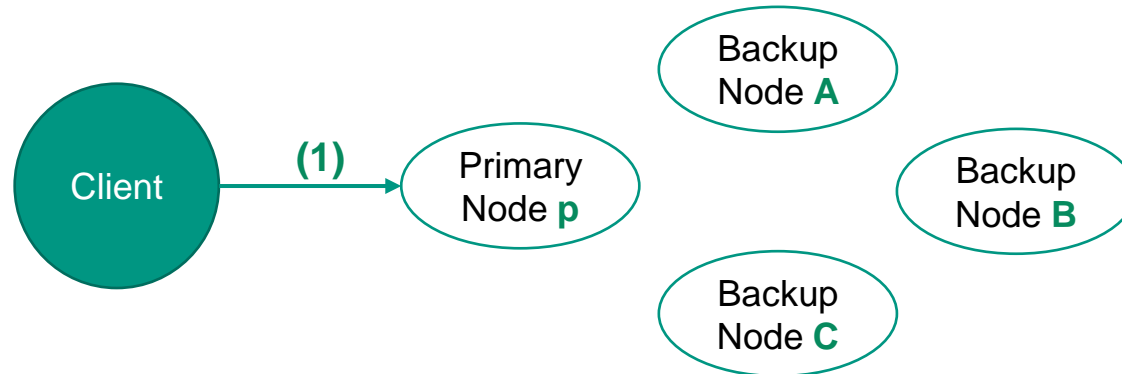
Source: Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation, 173–186. Berkeley, CA, USA: USENIX Association.

Applied Consensus Mechanisms (PBFT)

Practical Byzantine Fault Tolerance (PBFT)

- All nodes in the PBFT model are arranged in order
- There is one **primary node p** and **other backup nodes**
- Each round of PBFT consensus comprises four phases which proceed sequentially

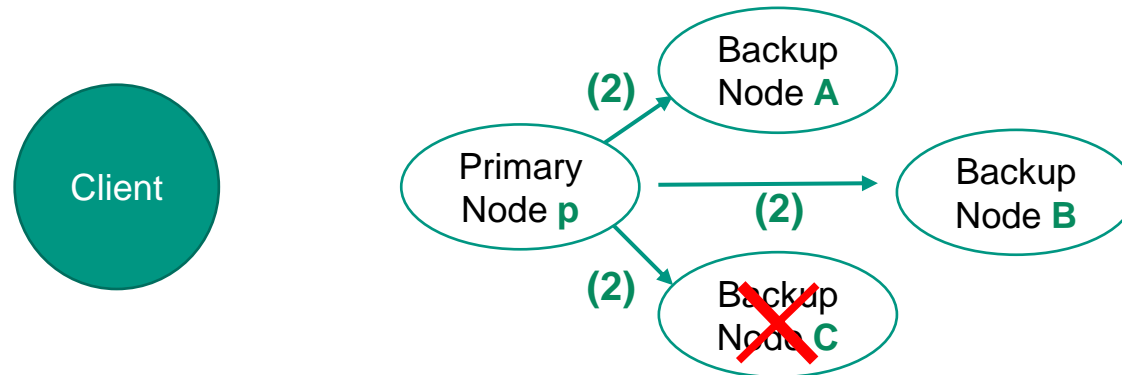
(1) *A client sends a request to p to invoke a service operation.*



Applied Consensus Mechanisms (PBFT)

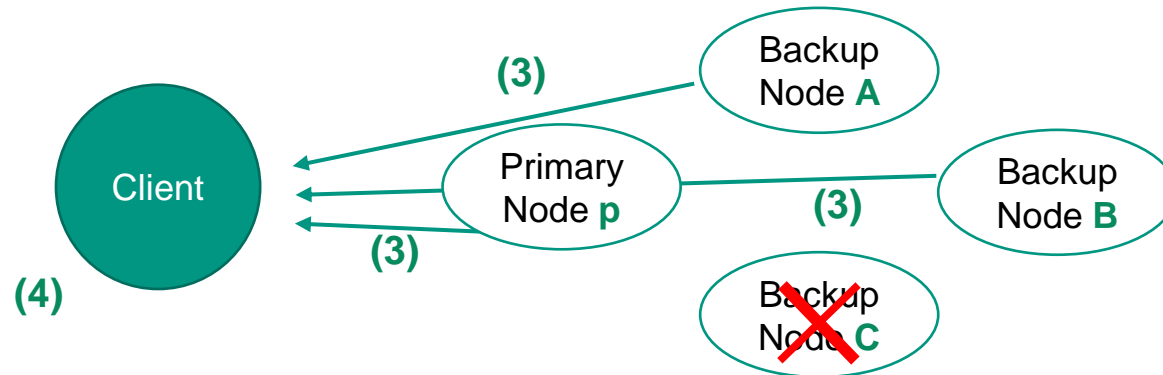
(2) *p multicasts the request to the backup nodes*

Assumption: Backup Node C drops out due to poor internet connection
Recall: $3f+1$ nodes are needed to handle f byzantine faults
 $3*1+1 = 4 = |p,A,B,C|$



Applied Consensus Mechanisms (PBFT)

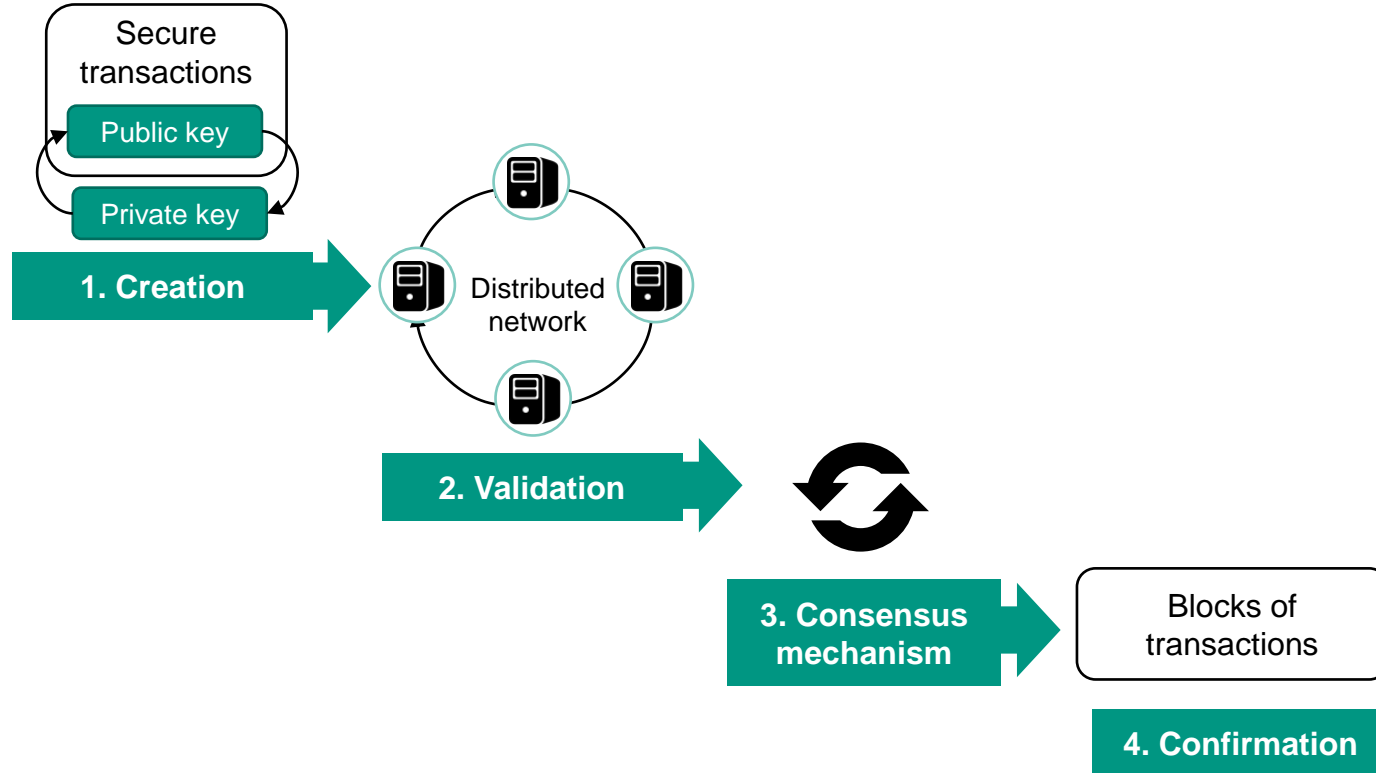
- (3) *The backup nodes execute the request and then send a reply to the client.*
- (4) *The client awaits $f + 1$ replies from different nodes with the same result. This result is the result of the operation.*



Now, we put the parts together and describe the basic functioning of...

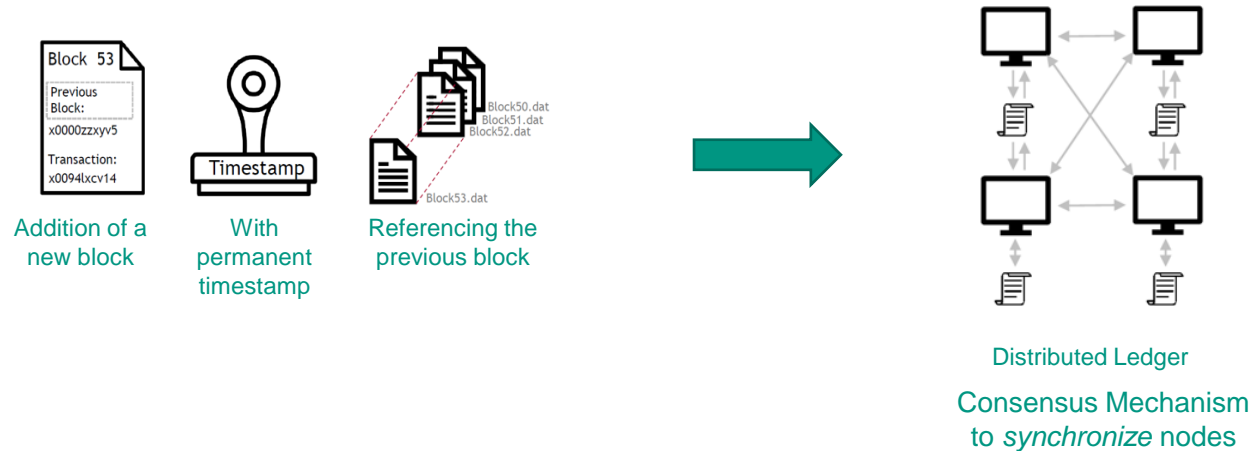
...the (Bitcoin) Blockchain

Blockchain — Putting the Parts Together (1/3)



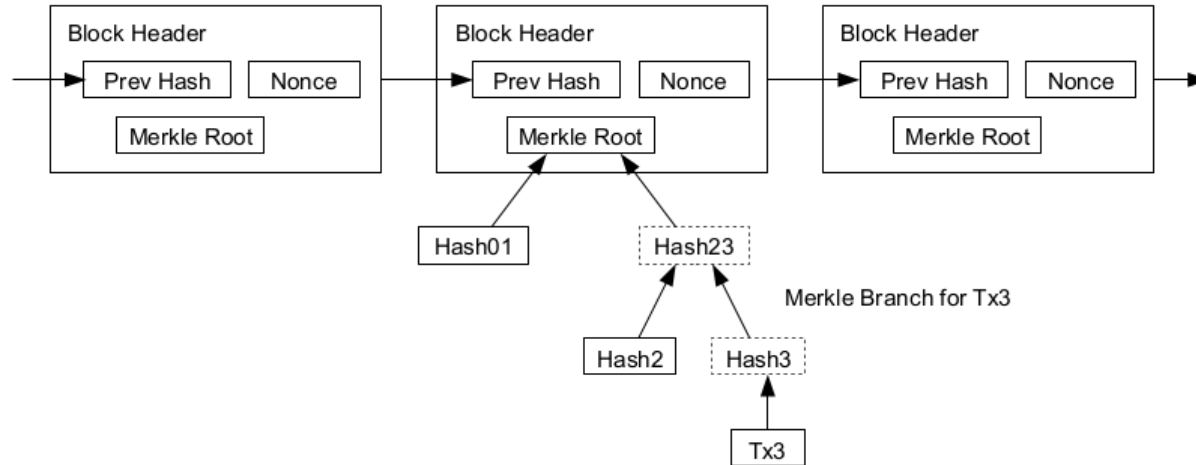
Blockchain — Putting the Parts Together (2/3)

- Updated, chronologically ordered, decentrally stored and publicly accessible register with information about **ownership** and **transactions**.
- Cryptographic principles enable **retroactive immutability** of the entries.



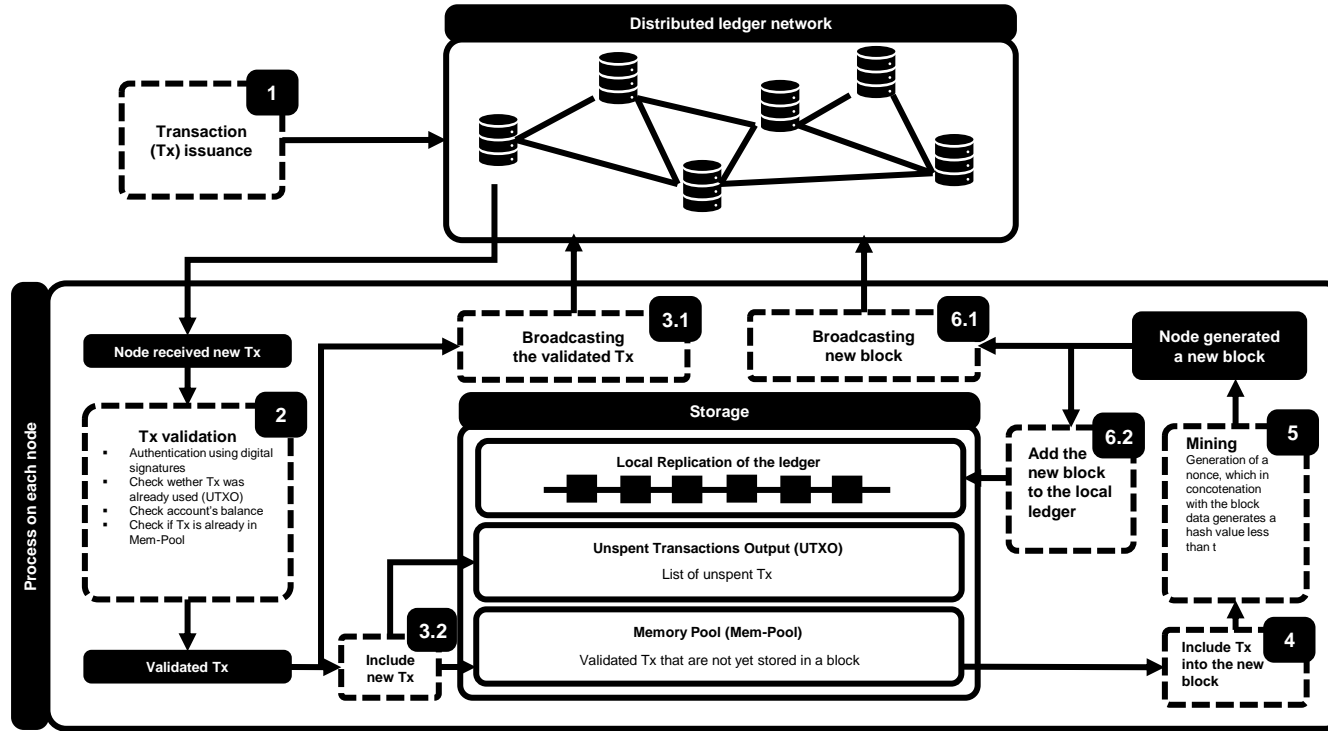
Blockchain — Putting the Parts Together (3/3)

- Each new block contains a unique link with its predecessor (and thus with its predecessors)
- Creation timestamp of the block determines order of the blocks



Source: Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from <https://bitcoin.org/bitcoin.pdf>

The Bitcoin Blockchain

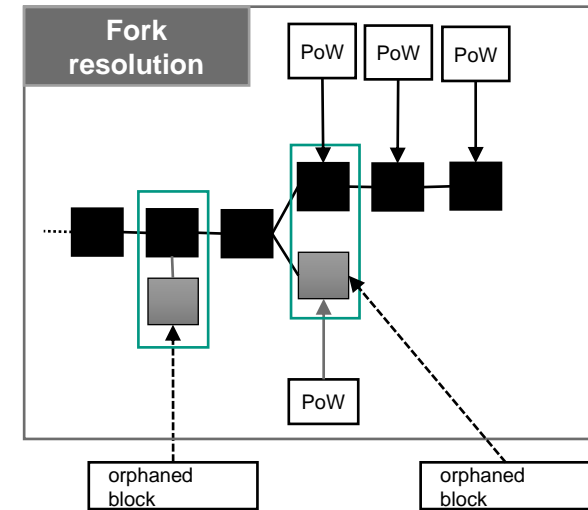


What Multiple Nodes Mine a Block at the Same Time?

- If a node receives multiple candidate blocks that could be appended to the blockchain, the node must decide for one block using a fork resolution rule

Fork resolution rule: A defined rule to identify the main chain when it came to forks. A fork is the existence of multiple branches on the main chain.

- Fork Resolution Rule in Bitcoin:
The longest chain is chosen by the nodes under the assumption that the longest chain has required the most work
- Blocks that are not included into the blockchain are called stale blocks
- Stale blocks correspond to network partitioning are a cause for inconsistencies between nodes
- Forks only occur in DLT designs that apply probabilistic consensus mechanism



Distributed ledgers are not only databases!

Smart Contracts

Smart Contracts

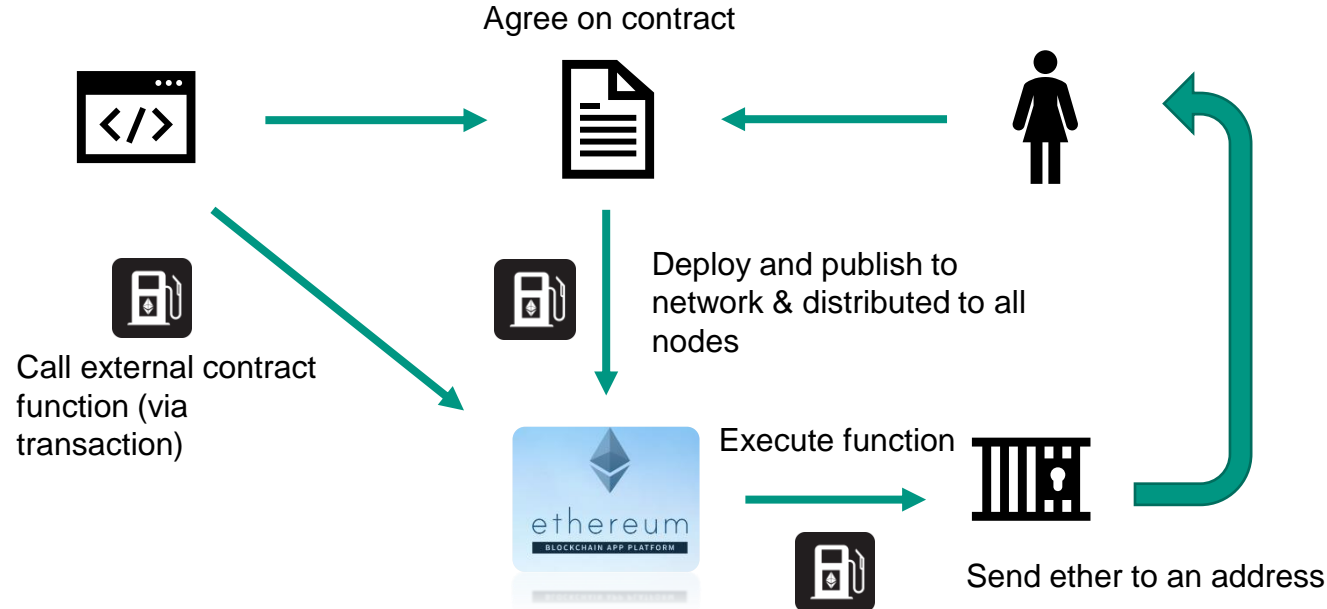
“...allow untrusted parties to manifest contract terms in program code and thus eliminate the need for a trusted third party.”

(Wöhler, 2018)



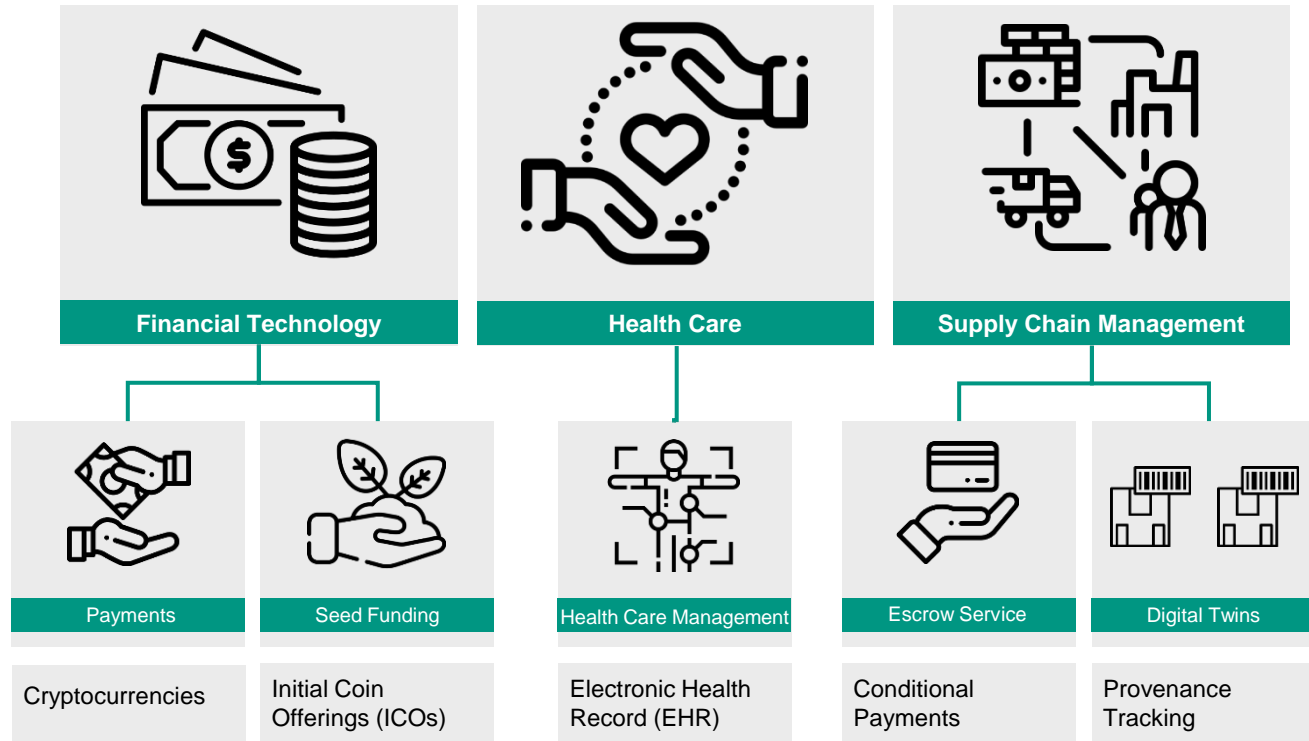
Image source: [\[Smart Contracts\]](#) by Pngwing, n.d. CC BY-NC.

Introduction to Smart Contracts



Exemplary Applications for DLT

Exemplary Applications of Distributed Ledger Technology



DLT for Open Access: BISE Student (1/2)

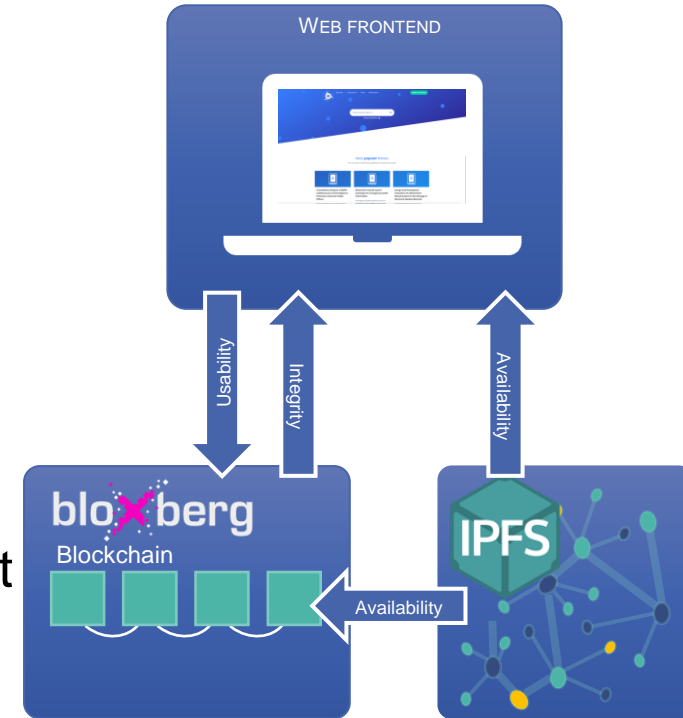
- Open Access platform for student theses
- Decentralized system through bloxberg blockchain
 - Fast and effective publication process
 - Openness and transparency
 - High integrity of data and processes
 - High system availability and scalability

Overall goal of the platform is to increase the **usefulness and impact** of high-quality student theses.

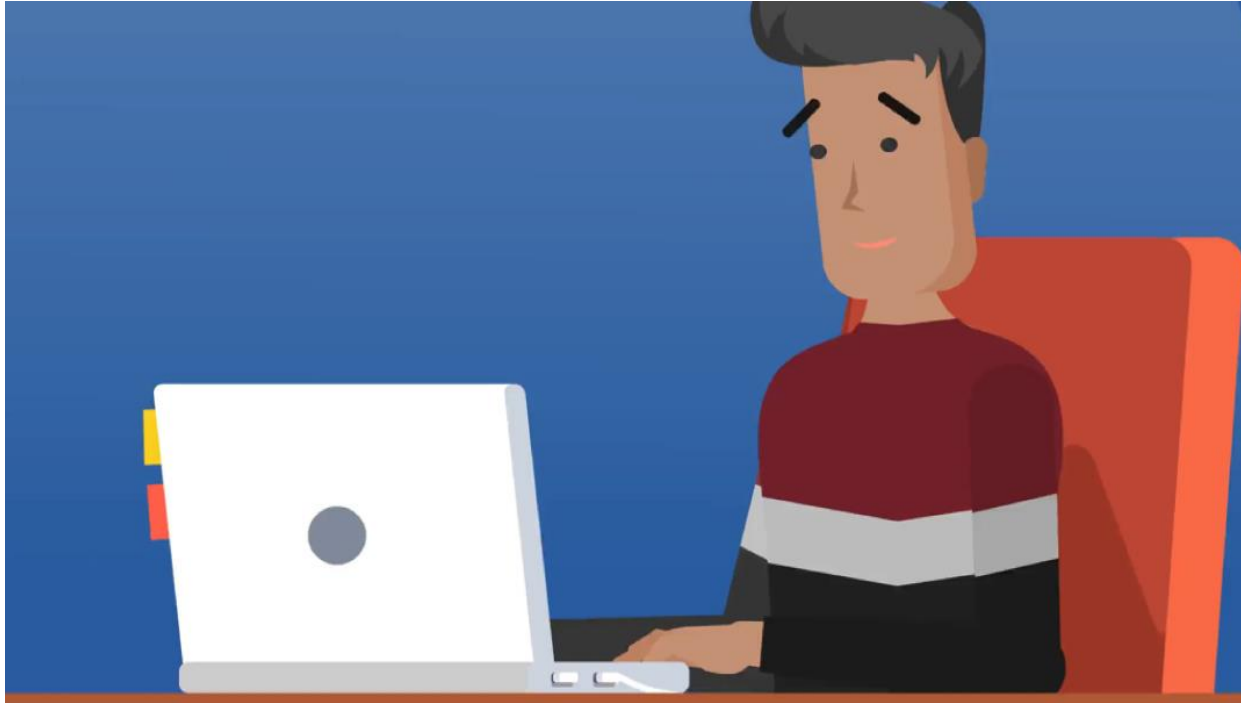
Powered by



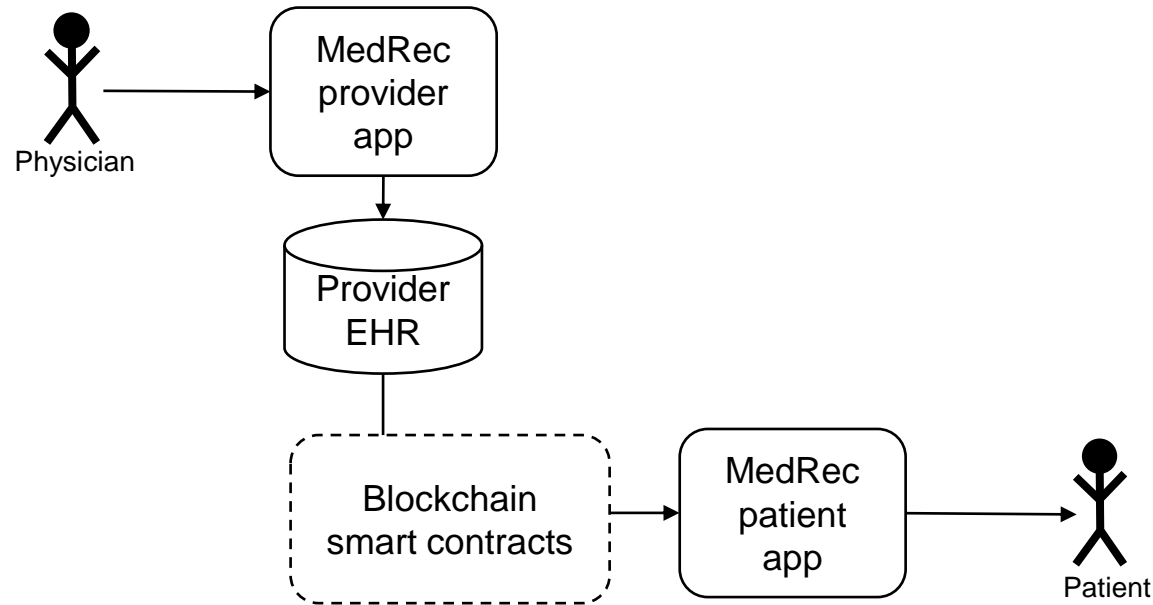
BUSINESS & INFORMATION
SYSTEMS ENGINEERING



DLT for Open Access: BISE Student (2/2)

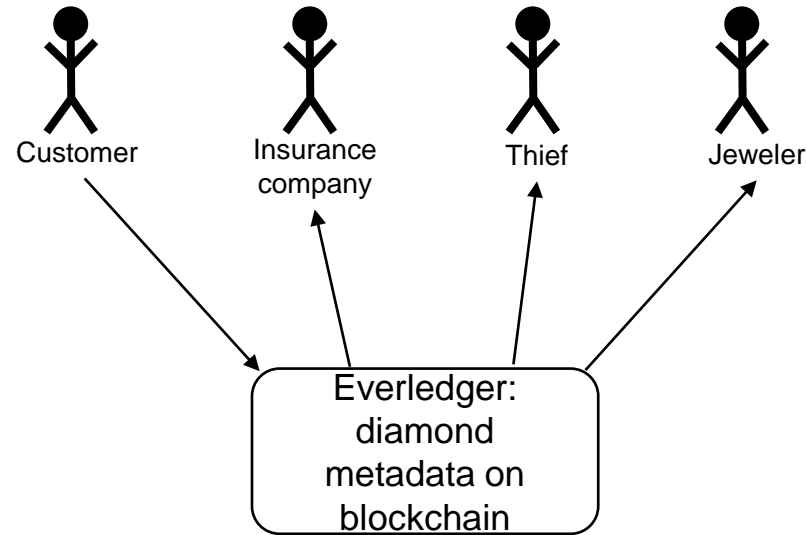


Managing Electronic Medical Records with MedRec



Source: Azaria, Ekblaw, Vieira, & Lippman. Medrec: Using blockchain for medical data access and permission management. In Open and Big Data (OBD), International Conference, IEEE, 25-30, 2016.

Verifying the Origin of Diamonds with Everledger



References

- Antonopoulos, A. M. (2017). Mastering Bitcoin: Programming the open blockchain, O'Reilly Media, Inc.
- Buterin, V. (2014). A next-generation smart contract and decentralized application platform.
- Kannengießer, N., Lins, S., Dehling, T., & Sunyaev, A. (2019). Trade-Offs between Distributed Ledger Technology Characteristics. *ACM Computing Surveys*, 53, 2.
- Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.
- Yeow, K., Gani, A., Ahmad, R. W., Rodrigues, J. J. P. C., & Ko, K. (2018). Decentralized Consensus for Edge-Centric Internet of Things: A Review, Taxonomy, and Research Issues. *IEEE Access*, 6, 1513–1524.
- Vallois, V., & Guenane, F. A. (2017). Bitcoin transaction: From the creation to validation, a protocol overview. In *Cyber Security in Networking Conference*, (pp. 1-7).
- Vukolić, M. (2017). Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts* (pp. 3-7).

Questions

Questions

1. What are key properties of distributed ledgers?
2. What are advantages and disadvantages of distributed ledgers compared to commonly used architectures?
3. What are hash functions and why are they important for DLT?
4. Which consensus mechanisms do you know and how do they work?
5. What are possible applications on DLT?

Further Reading

- Buterin V (2018). Ethereum whitepaper. <https://ethereum.org/en/whitepaper/>. Accessed 15 June 2021
- Glaser F, Bezzenberger L (2015). Beyond cryptocurrencies – a taxonomy of decentralized consensus systems. 23rd European conference on information systems, Münster, pp 1–18
- Göbel J, Krzesinski AE (2017). Increased block size and Bitcoin blockchain dynamics. 27th international telecommunication networks and applications conference, Auckland, 2017, pp 1–6
- Kannengießer N, Lins S, Dehling T, Sunyaev A (2019). What does not fit can be made to fit! Tradeoffs in distributed ledger technology designs. Paper presented at the 52nd Hawaii international conference on system sciences, Maui, HI, 8–11 Jan 2019
- Kannengießer N, Pfister M, Greulich M, Lins S, Sunyaev A (2020). Bridges between islands: crosschain technology for distributed ledger technology. Presented at the 53rd Hawaii international conference on system sciences, Waikoloa, Hawaii
- Nakamoto S (2008). Bitcoin: a peer-to-peer electronic cash system. <https://bitcoin.org/bitcoin.pdf>. Accessed 15 June 2021
- Yeow K, Gani A, Ahmad RW, Rodrigues JJPC, Ko K (2018). Decentralized consensus for edgecentric internet of things: a review, taxonomy, and research issues. IEEE Access 6:1513–1524