Do Not Be Fooled: Toward a Holistic Comparison of Distributed Ledger Technology Designs

Florian Gräbe
Karlsruhe Institute of Technology
florian.graebe@web.de

Niclas Kannengießer
Karlsruhe Institute of Technology
niclas.kannengiesser@kit.edu

Sebastian Lins
Karlsruhe Institute of Technology
lins@kit.edu

Ali Sunyaev
Karlsruhe Institute of Technology
sunyaev@kit.edu

Abstract

Distributed Ledger Technology (DLT) enables a new way of inter-organizational collaboration via a shared and distributed infrastructure. There are plenty of DLT designs (e.g., Ethereum, IOTA), which differ in their capabilities to meet use case requirements. A structured comparison of DLT designs is required to support the selection of an appropriate DLT design. However, existing criteria and processes are abstract or not suitable for an in-depth comparison of DLT designs. We select and operationalize DLT characteristics relevant for a comprehensive comparison of DLT designs. Furthermore, we propose a comparison process, which enables the structured comparison of a set of DLT designs according to application requirements. The proposed process is validated with a use case analysis of three use cases. We contribute to research and practice by introducing ways to operationalize DLT characteristics and generate a process to compare different DLT designs according to their suitability in a use case.

1. Introduction

Given the high potential of distributed ledger technology (DTL), numerous DLT designs have been developed during the past decade (e.g., Ethereum, IOTA, and Tezos). Such DLT designs specialize in fulfilling requirements of a particular set of applications on DLT in domains such as the Internet of Things (IoT), finance, and supply chain management. However, this specialization resulted in trade-offs between DLT characteristics that restrict the suitability of a DLT design to a particular set of applications [1, 2]. For example, a DLT design cannot provide high availability and a high degree of consistency simultaneously because a large number of nodes of the ledger, which is needed for high availability, requires more time and effort to be synchronized, thereby challenging consistency [1, 2]. Due to the prevalent trade-offs between DLT characteristics, the improvement of one DLT characteristic deters another, inhibiting a one-size-fits-all DLT design suiting all requirements of individual applications. Trade-offs thus require developers to choose the best fitting DLT design for their application [2]. Making careful and well-founded decisions in favor for an (appropriate) DLT design to develop viable applications on DLT becomes even more crucial because technical differences between DLT designs (e.g., different data structures and consensus mechanisms) impede the ex-post migration of data stored on one distributed ledger to another [3].

Developers, therefore, need to conduct a comprehensive comparison between prospective DLT designs to assess DLT designs' suitability for a particular application on DLT before starting the actual implementation. Such comparisons require the operationalization of DLT characteristics that is referred to as a process of defining the measurement of DLT characteristics to make them understandable, measurable, and comparable. However, it remains challenging for developers to compare DLT designs and operationalize DLT characteristics because DLT synthesizes multiple techniques of computer science (e.g., cryptography and distributed databases), which come with individual operationalizations for characteristics. In addition, DLT exhibits unique characteristics, such as stale block rate and smart contract support [1], requiring new operationalizations. The operationalization of DLT characteristics must first be clarified to enable developers conducting comparisons of DLT designs.

Research on DLT in computer science has already taken different perspectives on the analysis, operationalization, and benchmarking of DLT designs, for example, in regard to formal verification of consensus mechanisms (e.g., [4]) and the analysis of DLT designs in different configurations (e.g., [5, 6]). However, prior research predominantly considers DLT characteristics related to performance (e.g., [7, 8]) and, thus, neglects further important characteristics (i.e., flexibility, anonymity) and does not allow a holistic

URI: https://hdl.handle.net/10125/64512
978-0-9981331-3-3

comparison of DLT designs. Research on DLT in information systems deals with the specification of processes to support developers in their selection of an appropriate DLT design for an application (e.g., [9, 10]), among others. However, the presented processes are often too abstract to compare DLT designs. To conduct a comprehensive comparison of DLT designs it requires the synthesis of the prevalent research streams on the operationalization of DLT characteristics and benchmarking DLT designs, and decision support in the selection of an appropriate design. We ask the following research question (RQ):

RQ: How can DLT designs be compared according to application requirements prior to implementation?

To answer the RQ we follow two objectives: first the identification and operationalization of relevant DLT characteristics and, second, the development of a process for the comparison of DLT designs. For the first, we consolidated a set of DLT characteristics that we deem relevant for applications on DLT and synthesized existing research on benchmarks and operationalizations of DLT, distributed databases, and information systems. Second, we generated and validated a process for the comparison of DLT designs with three prominent use cases.

With our study, we contribute to practice as we enable a comprehensive comparison of DLT designs and, thus, provide decision support for the selection of a suitable DLT design for viable applications on DLT. We contribute to research because we synthesize existing approaches and generate new means for the operationalization of DLT characteristics. Thus, our results can serve as a foundation for research on the selection of suitable DLT designs for applications.

2. Related research

2.1. Distributed ledger technology

DLT serves as a shared, digital infrastructure for applications (e.g., financial transactions) because DLT enables the operation of an append-only database (referred to as ledger), which is distributed across multiple storage devices (so-called nodes) in an untrustworthy environment [11]. Each node maintains a local replication of the data stored on the ledger. An untrustworthy environment is characterized by the arbitrary occurrence of Byzantine failures [12] such as crashed or (temporarily) unreachable nodes, network delays, and malicious behavior of nodes (i.e., issuing wrong information). In DLT, data is appended to the ledger through transactions and is stored in a chronologically-ordered sequence. Each transaction contains meta-data (e.g., receiver address, timestamp) and a digital representation of certain, tokenized assets or program code of a smart contract. A tokenized asset is a digital representation of an asset (e.g., coins) in a structured data format (token), which can be transferred using transactions. When a node receives a new transaction, the transaction is validated. Valid transactions are forwarded to all adjacent nodes, which also validate and forward the transaction subsequently.

Because all nodes of a distributed ledger maintain a local replication of the ledger, all nodes must be synchronized and agree on a common state of the distributed ledger to reach consistency. For this purpose, a consensus mechanism is employed [13]. A consensus mechanism is used to manage the negotiation between nodes, which eventually agree on a common replication of the ledger. Once appended, data can hardly be altered or removed anymore.

2.2. Comparison of DLT designs

The comparison of DLT designs requires taking both a technical perspective considering DLT characteristics, such as fault tolerance or throughput, and an economical perspective considering costs and time for software development. Therefore, we first present related research in benchmarking DLT characteristics. Second, we present related research on supporting developers to select a suitable DLT design in order to avoid, for example, high switching costs. Benchmarking forms a measurement of a (non-productive test) system at a specific point in time [14] and allows for the comparison of systems in artificially created scenarios. To date, DLT benchmarking predominantly focuses on DLT characteristics related to performance and security (e.g., throughput, network partitioning) [5, 15], specific use cases [16], or private DLT designs [17]. Still, such approaches do not allow for holistic benchmarking of DLT designs because they do not consider DLT characteristics such as availability or cost. Since DLT incorporates multiple domains of computer science (i.e., databases, cryptography, information systems), DLT benchmarking must also consider extant research in these disciplines.

Database benchmarks include characteristics such as transaction speed and consistency. Meanwhile, industry standards have been established for database benchmarks with a focus on transaction execution, performance, and scalability (e.g., TPC-H). While these operationalizations might be used for certain DLT characteristics, database benchmarks do not cover several unique DLT characteristics (e.g., confirmation latency, fault tolerance).

Extant research in cryptography already provides operationalizations for characteristics related to hashing or public-key encryption, which are substantial for DLT. From the performance perspective, for example,

time complexity of such encryption algorithms is an important criterion to assess. From the security perspective, the collision resistance of hashing algorithms is a common criterion to validate a hashing algorithm against pre-image attacks [18]. Although the operationalization of cryptographic procedures is useful to measure in DLT, these operationalizations serve only one (research) domain DLT draws from.

Related research on decision support to find an appropriate DLT design is still in its infancy. There are already approaches that support the decision of using DLT or not (e.g., [10]) and several classifications of DLT design have been proposed to make DLT and its characteristics better comprehensible for developers (e.g., [19]). However, such processes do not consider the operationalization of DLT designs and, thus, are only applicable to DLT on a limited scale.

3. Method

We applied a four-step research approach. First, we reviewed the extant literature on DLT to generate a preliminary set of characteristics, which are relevant for DLT design comparison (e.g., [1, 22]). Second, we searched operationalizations of the respective DLT characteristics in extant literature. Third, we evaluated the suitability of DLT characteristics and corresponding operationalizations for the comparison process following well-known IT benchmarking requirements and quality criteria for metrics (e.g., [14, 20, 21, 23]; cf. Table 1 and Table 2). Finally, we generated a DLT comparison process and evaluated its usefulness using three use cases and five DLT designs (i.e., Ethereum, Hyperledger Indy, IOTA, Tezos and Hashgraph).

3.1. Selecting suitable characteristics for the comparison of DLT designs

To identify candidate DLT characteristics for our comparison process, we built on our prior research on identifying and clustering DLT characteristics [1, 24, 25], leading to a set of 37 DLT characteristics. Moreover, we reviewed further research articles and white-papers on DLT characteristics (e.g., [1, 19, 26]) and on DLT benchmarking (e.g., [22]), which led to a final set of 50 DLT characteristics as candidates for the comparison process. We classified these DLT characteristics into qualitative and quantitative sub-groups to conduct a more structured search for operationalization approaches. As a next step, we performed a focused literature search for each DLT characteristic to identify potential operationalizations in DLT research and related research streams (i.e., databases, cryptography, information systems). In particular, we searched in scientific databases, including IEEE Xplore, EBSCOHost, ACM Digital Library, and Proquest. For each DLT characteristic, we created a unique search string comprising the name of the DLT characteristic (i.e., *scalability*) and synonyms for operationalization (i.e., *benchmarking*, *metrics*, *measurement*). We reviewed the resulting research articles and noted proposed operationalizations for each characteristic. The identified operationalizations can be classified into three different categories: First, operationalizations found in the (non-) scientific literature on DLT that could directly be inherited into the comparison process (e.g., operationalizations for consistency or confirmation latency). Second, operationalizations found in related domains that needed to be adapted to fit the DLT context, such as operationalizations for developer support, scalability, or availability. Lastly, we were left with four DLT characteristics (i.e., traceability, transaction content transparency), where no applicable operationalizations were found. For these DLT characteristics, we developed new metrics complying with the criteria from Table 2.

To assess whether identified DLT characteristics are suitable for DLT design comparison, we evaluated whether they fulfill the requirements for IT benchmarks (cf. Table 1) and whether corresponding operationalizations comply with requirements for quality metrics (cf. Table 2). For more information see "Exclusion Methodology" in the supplementary online material (*https://bit.ly/2klPK9v*). First, we investigated if identified operationalizations are consistent.

Table 1. Summary of requirements for IT benchmarking [14, 20, 21]

Requirement	Description
Economic Efficiency	It must be economically affordable to run the benchmark.
Fairness	Fairness means that a benchmark should treat every system under test fairly and equally and should not make assumptions on the system's features.
Portability	It should be easy to implement the benchmark on many different systems and architectures.
Relevance	The benchmark should focus on typical operations within that problem domain.
Reproducibility	Reproducibility implies that running the same benchmark multiple times will yield similar results, meaning that a certain degree of determinism is required.
Simplicity	The benchmark must be understandable, otherwise, it will lack credibility. A key aspect is the presence of meaningful and expressive metrics.
Scalability	The benchmark should apply to small and large computer systems. It should be possible to scale the benchmark up to larger systems and to parallel computer systems as computer performance and architecture evolve.

Table 2. Summary of requirements for quality metrics [14, 23]

Requirement	Description
Consistency	The output of the metric should be consistent. A monotonic function is often required.
Correlation	There should be a (not necessarily linear) correlation between the dimension under observation and the metric output.
Discriminative Power	The metric should accurately display differences of parameter levels and especially differentiate between high and low parameter levels.
High resolution	The metric should have a large number of possible output values and should avoid unnecessary aggregation.
Tracking	The metric should build on the current state of the system.

We excluded 17 DLT characteristics (i.e., governance, non-repudiation, interoperability) because a consistent output of identified operationalizations could not be guaranteed as, for instance, the operationalizations require a subjective view on qualitative DLT characteristics. For example, we excluded the DLT characteristic *compliance* because a metric that operationalizes *compliance* would have a strongly varying output depending on the particular regulations, thereby threatening consistency. As a next step, we excluded five DLT characteristics (e.g., incentive mechanism) that did not follow the high-resolution criteria of quality metrics (cf. Table 2). The remaining DLT characteristics fulfill the requirements for quality metrics, including correlation, discriminative power, and tracking. As the last step, we examined if the final set of DLT characteristics fulfills all requirements for IT benchmarking (cf. Table 1). We removed eight DLT characteristics that did not comply with the relevance criteria, leading to a final set of 20 relevant DLT characteristics for the comparison process. For example, *block creation interval* was excluded because it is expressed in *throughput* and *stale block rate* [27], which can be directly mapped to application requirements.

3.2. Comparison process development

We grounded the development of a comparison process on previous research of benchmarking processes in the IT field (e.g., [28, 29]). We also drew from extant research on benchmark process development (e.g., [29]) and adapted it to the field of DLT under consideration of the requirements depicted in Table 1. We used [29] as a basic approach for a benchmarking process and adapted it in two discussion rounds with researchers to fit the DLT context while complying with the requirements from Table 1. During this adaptation, the important core steps of the comparison process were identified, adapted and enhanced to our use case. The individual steps were then renamed to keep consistent terminology. We showed the usefulness of the comparison process by evaluating the process in three prominent use cases in the field of DLT: *cryptocurrency*, *IoT*, and *identity management*. We selected these use cases because they form a high percentage of identified DLT use cases in research [30] and currently worked on by large companies [31]. For each use case, we applied the comparison process and included five strongly different DLT designs, which have been developed for different purposes: Ethereum as a general-purpose blockchain, Hashgraph as a public multi-purpose DLT design, IOTA with a focus on IoT, Hyperledger Indy for identity management, and Tezos for strong governance to ensure a wide variety of different ledgers and to show the usefulness of the process for different DLT designs. We chose these DLT designs because they strongly differ in the used data structures (e.g., Hashgraph and IOTA follow the concept of transaction-based, directed acyclic graph instead of a linked chain of blocks like in Ethereum, Hyperledger, and Tezos), the applied consensus mechanisms (e.g., IOTA uses the Tangle and Ethereum relies on Proof-of-Work), and in their permissioning (Ethereum as a permissionless DLT designs and Hyperledger Indy as permissioned DLT design). Thus, we show that the developed operationalizations are applicable to a variety of DLT designs and reflect their individual strengths.

4. Results

4.1. DLT characteristics for the comparison

4.1.1. Flexibility. Flexibility incorporates the possibilities offered by a DLT design for maintenance and further development [1]. The *purpose of tokens* (F1) can be classified into achieving three different objectives: payment tokens (e.g., Bitcoin), utility tokens (e.g., Filecoin), and security tokens (e.g., tZERO). *Smart contract support* (F2) is a qualitative characteristic that can be mapped to a nominal scale: either a DLT design offers Turing-complete, Turing-incomplete, or no smart contracts.

4.1.2. Institutionalization. Institutionalization describes the embedding of concepts and artifacts (here DLT) in social structures. *Developer support* (I1) is mainly driven by the size of the developer community currently dealing with a particular DLT design. Therefore, we operationalize *developer support* as the number of active developers. A developer is considered active if she has at least one commit to a DLT design core (e.g., Ethereum Protocol) or a project related to a

Table 3. DLT characteristics and their operationalization following [1, 8, 22]

*	DLT Chars.**	Definition	Operationalization
Flexibility	Purpose of the Tokens (F1)	The purpose and flexibility in usage of the provided tokens of a DLT design.	$\begin{cases} Equity\ Token \\ Utility\ Token \\ Security\ Token \end{cases}$
	Smart Contract Support (F2)	The level of how well smart contracts are supported by a DLT design.	$\begin{cases} Turing-Complete \\ Turing-Incomplete \\ None \end{cases}$
Institutional.	Developer Support (I1)	The existence of documentation, interaction platforms such as forums, or direct contact to the team developing the distributed ledger for questions and issues related to the development of applications integrating DLT as well as programming tools and interfaces.	$Number\ of\ active\ developers$
	Liability (I2)	The existence of an organization that is responsible for the maintenance of a DLT design.	$\begin{cases} Liable\ organisation\ exists \\ No\ liable\ organisation \end{cases}$
Anonymity	Node verification (LA1)	The extent to which new nodes can join the distributed ledger without being verified.	$\begin{cases} Public, unpermisisoned\ DLT\ design \\ Public, permissioned\ DLT\ design \\ Private, permissioned\ DLT\ design \\ Private, unpermissioned\ DLT\ design \end{cases}$
	Traceability (LA2)	The level to which the transfer of an asset can be traced chronologically on a distributed ledger.	$\begin{cases} Transfers\ are\ publicly\ traceable \\ Transfers\ are\ obfuscated\ and\ hardly\ traceable \\ Transfers\ cannot\ be\ traced \end{cases}$
	Transaction Content Transparency (LA3)	The ability to publicly view an account's holdings and transactions' contents on a distributed ledger.	$\begin{cases} Plain\ Content \\ Encrypted\ Content \end{cases}$
Performance	Confirmation Latency (P1)	The average time until enough blocks (or transactions) are added to the distributed ledger so that the likelihood of tampering of a previously added block or transaction is below a certain threshold.	$Security\ Confirmations * Block\ Creation\ Interval$
	Scalability (P2)	The capability of a DLT design to handle an increasing amount of workload or its potential to be enlarged to accommodate that growth.	$\frac{\frac{Throughput(k_2)}{MTL(k_2)}}{\frac{Throughput(k_1)}{MTL(k_1)}}\ MTL > 0, k_1 < k_2$
	Throughput (P3)	The number of transactions validated and appended to the ledger in a given time interval.	$\frac{completed\ transactions}{time\ interval}$

** DLT Property ** DLT Characteristics*

DLT design (e.g., an application using the DLT design) on GitHub during the last three months. *Liability* (I2) is classified as a qualitative characteristic that we map to a nominal scale with binary values existence of or non-existence of an enterprise organization for the purpose of operationalization.

4.1.3. Anonymity. Anonymity describes the degree to which individuals are not identifiable within a set of users [1]. *Node verification* (LA1) is predominantly concerned with granting or revoking permissions for nodes. Thus, we mapped *node verification* to the DLT design types public or private to consider read permissions. For write permissions, we distinguish between permissioned and permissionless DLT designs.

For *traceability* (LA2), we chose three distinct values: publicly-viewable transfers (e.g., in Bitcoin); obfuscated transfers (e.g., using mixing [32]); not traceable transfers (e.g., in ZCash). For *transaction content transparency* (LA3) we chose a binary value that distinguishes between data stored in plain text or encrypted.

4.1.4. Performance. Performance includes DLT characteristics regarding the accomplishment of a given task measured against standards of accuracy, completeness, costs, and speed [1] such as *confirmation latency*, *throughput*, and *scalability*. *Confirmation latency* (P1) refers to the period required to append a minimum number of blocks to the distributed ledger that must at least succeed a certain block *b* to assure that *b* cannot be altered anymore (e.g., [33]). We operationalize *confirmation latency* (P1) as duration until a transaction is seen as finalized.W

For *scalability* (P2), we focus on horizontal scalability (cf. [17]), which is operationalized by changes to *throughput* (P3) and *mean transaction latency* (MTL) when the number of validating nodes (de-) increases. For operationalization, we use p-scalability [34] (cf. formula 1), which compares two differently scaled system levels k_1 and k_2 according to the power metric (cf. formula 2) and scaling cost [34]. We relay this concept to DLT by excluding the cost factor and replacing the mean delay with the mean transaction latency (MTL) (cf. [17]).

(1) $$p - Scalability = \frac{\frac{Throughput(k_2)}{MTL(k_2) * ScalingCost(k_2)}}{\frac{Throughput(k_1)}{MTL(k_1) * ScalingCost(k_1)}}$$

(2) $$power = \frac{Throughput}{MTL}$$

In (1), throughput behaves proportionally to scalability, while transaction latency behaves anti-proportionally to scalability. For example, if a DLT scales up well, one would expect throughput to increase and transaction latency to decreases. This behavior is represented in our metric by placing throughput in the numerator of the metric and the transaction latency in the denominator. To form a score for scalability, we compare two different horizontal scaling levels k_1 and k_2 with different numbers of participating nodes as our metric. We assume MTL to never converging to zero. Usually, a scalability coefficient equal to or larger than 1 reflects good scalability, while a scalability coefficient close to zero indicates bad scalability. The quotient of the data from the actual period is compared to the performance data quotient from the previous period to form the scalability score which is then computed into a final *scalability* score (P2).

4.1.5. Security. Security refers to the preservation of confidentiality, integrity, and availability of information. *Availability* (S1) is operationalized as probability and considers the Meant Time To Failure (MTTF) and Mean Time To Repair (MTTR) [35]. MTTF is the period a distributed ledger is expected to correctly operate. MTTR is the required period to recover the distributed ledger from a failure. This sum is equal to the Mean Time Before Failure (MTBF) [35]. In the field of DLT, we refer MTTR and MTTF to the full distributed ledger instead of particular nodes.

Consistency (S2) refers to storing the identical replications of the ledger on each node at the same time [36]. In Hyperledger Caliper, transaction latency is a metric to measure consistency as the period for a transaction's effect to be available on all nodes [8]. We adopt this interpretation and measure the period between transaction issuance and transaction confirmation. We include it as a criterion in form of an average of *N* measurements at different times and states of the distributed ledger. *Fault tolerance (S3)* is the ability of a distributed ledger to correctly operate in the presence of failures [35]. We operationalize fault tolerance as the changes in throughput and transaction latency (TL) during node failure (cf. [17]). Node failure is the stopping of a node (crash failure), network delay, or random responses due to corrupted messages [17].

Level of decentralization (LoD) (S4) expresses the ratio of the number of independent validating nodes (VNs) and the number of validating node operators (VNOs). While VNs represent validating nodes, VNOs represent an individual or organization, who maintains the VNs (e.g., a mining pool). To include permissioned DLT designs in our operationalization,

Table 3 cont. DLT characteristics and their operationalization following [1, 8, 22]

*	DLT Chars.**	Definition	Operationalization
Security	Availability (S1)	The probability that a system is operating correctly at an arbitrary point in time.	$\frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}$
	Consistency (S2)	The state that all nodes store the same data in their ledger at the same time.	$\emptyset\, TL = \frac{1}{N} * \sum_{t=1}^{N} Confirmation\ Time(t) - Submit\ Time(t)$
	Fault Tolerance (S3)	The ability of a distributed ledger to correctly operate in the presence of (hardware or software) failures.	$\begin{cases} \Delta\, Throughput\ during\ node\ failure \\ \Delta\, Transaction\ Latency\ during\ node\ failure \end{cases}$
	Level of Decentralization (S4)	The number of independent node controllers participating in transaction validation and consensus finding.	$\frac{number\ of\ VNOs}{number\ of\ VNs} *\ number\ of\ VNOs$
	Network Size (S5)	The number of validating nodes of a distributed ledger that keep a full replication of the ledger.	$Number\ of\ full\ nodes$
	Reliability (S6)	The period of time during which a system is correctly functioning.	$Reliability(t) = e^{(-\frac{t}{MTTF})}$
	Stale Block Rate (S7)	The number of blocks in a defined period of time that has been mined but not added to the distributed ledger. A stale block forms a fork until it is resolved by the DLT design's fork resolution rule.	$\frac{mined\ but\ not\ added\ blocks}{last\ 1000\ blocks}$
	Strength of Encryption (S8)	The level of security of the applied cryptographic approach.	$Collision\ resistance\ of\ used\ hashing\ algorithm$
Usability	Censorship Resistance (U1)	The equal right of any user of the distributed ledger to submit transactions that are not altered or dropped by a third party.	$\frac{LoD}{VNOs} = \frac{number\ of\ VNOs}{number\ of\ VNs}$
	Cost (U2)	Costs related to the implementation and usage of a DLT design, including software development and operational costs.	$\frac{\emptyset\ H\&E\ Cost}{Month} + \emptyset T - cost * \frac{\emptyset\ Transactions}{Month} + \frac{\emptyset\ M\&S\ Cost}{Month}$

** DLT Property ** DLT Characteristics VN: Validating Node VNO: Validating Node Operator*

we multiply the ratio of VNs to VNOs with the number of VNs to correctly scale it to the network size and to differentiate between permissioned DLT designs with a lower LoD score and permissionless networks with a higher LoD score due to a higher number of VNOs.

Network size (S5) describes the number of nodes in a distributed ledger that stores a full replication of the ledger, considering only full network nodes that keep a full replication of the ledger [1]. Thus, we operationalize network size as the number of full nodes in a distributed ledger.

Reliability (S6) refers to the period during which a distributed ledger is correctly functioning. In DLT, the component that may fail refers to the complete distributed ledger [35]. The operationalization yields a probability that the system produces a correct output up to a time *t* [35]. For our benchmarking process, we introduce the estimated project duration *t* for all DLT designs and require a certain probability. Due to the little occurrences of system failures of DLT designs, this metric is assumed to be close to one, even for larger *t*.

Stale blocks impact security and performance because they lead to inconsistency between nodes through forks. The *stale block rate* (S7) is operationalized as a percentage of the mined but not included blocks measured over the last 1000 blocks [7]. Non-forking consensus mechanisms (e.g., Practical Byzantine Fault Tolerance) get the score of zero.

The strength of encryption refers to the level of security concerning the application of authentication-related cryptographic primitives (e.g., hashing algorithm). For the benchmarking process, we use the collision resistance of the applied hashing algorithm in the DLT design, which refers to the ease in guessing a pre-image for a hash value. A Birthday attack is a probabilistic approach of guessing pre-images exploiting the fixed degree of permutations [37]. A Birthday attack evaluates a hash function with *n* input bits and *m* output bits for randomly-selected inputs until two matching outputs are found. The number of pairs (p) in-between inputs which may yield to a collision grows quadratically with the number of trials *l* in a Birthday attack (cf. formula 3). As every pair of inputs is a chance for a matching output, finding a collision becomes more and more likely. Using a Birthday attack, it is possible to find a collision of two different inputs with Formula 4 or better time attack (t_{attack}) [37] leading to Formula 5 as the security in (output-) bits (s_{bit}) of *h*.

$$(3)\quad p = \frac{l * (l-1)}{2} \qquad (4) t_{attack} = 2^{\frac{m}{2}} \qquad (5)\quad s_{bit} = \frac{m}{2}$$

4.1.6. Usability. Usability refers to the extent to which a DLT design can be used by specified users to achieve specified goals with respect to effectiveness, efficiency, and satisfaction in a context of use. *Censorship resistance* (U1) describes the probability that a party can strongly influence the acceptance or refusal of transactions with a reasonable effort. Little censorship resistance comes from a low LoD [1]. Therefore, we operationalize censorship resistance as a probability dependent on LoD. A high LoD represents high censorship resistance, which the metric expresses with a score close to *1.0*.

Cost (U2) relates to the implementation and use of a DLT design. To operationalize *cost* we draw from a Total Cost of Ownership (TCO) approach and adjust it to DLT [38]. Our TCO metric employs the sum of average Hardware and Electricity (H&E) cost, the average cost for transactions (e.g., transaction fees), and the average Maintenance and Servicing (M&S) cost, all computed per month. Costs for research and human resources are excluded to keep the calculation feasible and because they are assumed to be similar for each DLT design since research and hiring of quality personal needs to be done for all DLT designs. H&E costs include the cost for all needed hardware to set up all necessary components to participate in the distributed ledger to the extent required by the use case, as well as the accumulated energy and resource costs of mining (if mining is performed). M&S costs cover all maintenance costs to keep the network operational and all services used within or outside the distributed ledger.

4.2. Comparison process for DLT designs

We propose a seven-step process for the holistic comparison process of DLT designs: *use case definition*, *requirements definition*, *DLT design selection*, *boundary condition examination*, *data collection*, *analysis*, and *decision making* (cf. Figure 1). In the following, we illustrate the usefulness of the comparison process for an exemplary use case in IoT. For parsimoniousness, we only consider the DLT characteristics *consistency* (S2) and *throughput* (P3), which are consolidated in Table 4. Please refer to "Use Cases" in the supplementary online material for the full comparison (*https://bit.ly/2klPK9v*).

First, we define the use case and functional requirements for the application during the *use case definition*. To assess the usefulness of DLT we include the process of [9] into our comparison process, which we adapt to DLT in general. The process serves as a pre-filter to prevent developers from choosing an unsuitable technology (e.g., DLT vs. centralized database) beforehand. If DLT has been found suitable for the use case, the process continues.

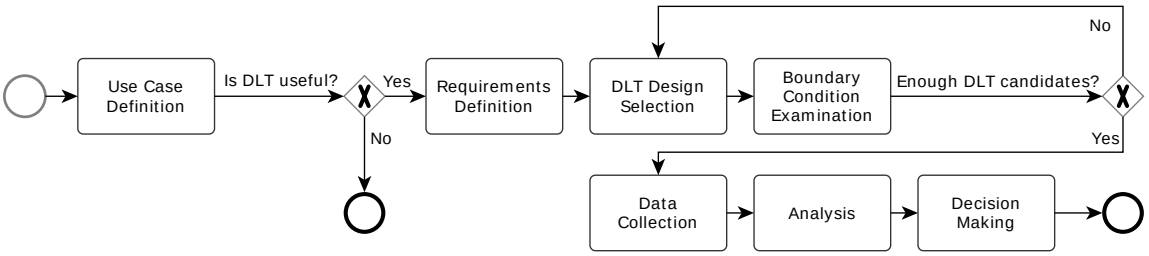

Figure 1. Sequential steps of the process to compare DLT designs

Second, the pursued values of the individual metrics are ascertained as forming criteria of the process in *requirements definition*. In the IoT use case, *consistency* (S2) should be fairly low (< 2 s) because many IoT networks share and handle real-time sensor data. IoT networks usually incorporate a large number of devices and sensors. Therefore, we require *throughput* (P3) to be at least 1,000 tps.

Third, the *DLT design selection* requires to generate a set of possibly suitable DLT designs for the use case. A survey on the application of DLT designs in similar use cases should lead to a set of possible DLT designs for the following evaluation process. This set should have a cardinality of at least two DLT designs but is not restricted by an upper limit. For our exemplary use case, we select a set of five DLT designs: Ethereum, Hyperledger Indy, Tezos, and IOTA, Hashgraph.

Fourth, during the *boundary condition examination* potential compliance, legal, and maturity risks of members of the previously defined set of possible DLT designs are considered. For example, newly developed DLT designs with an error-prone code base that are hardly ready to use may be examined and then excluded from the set of candidates. In this step, an evaluation of previously excluded qualitative characteristics can be made. All DLT designs, which do not comply with these factors should be excluded from the set. If too many ledgers are being excluded in this step, a step back to *DLT design selection* towards more DLT design candidates may be necessary. The topics ease of use, dependencies on third parties (e.g., an enterprise, a foundation), auditability restrictions, or data ownership considerations can be examined to exclude additional DLT design candidates if they are considered important for the use case. After reviewing possible legal, compliance and maturity risks, no hindering boundary conditions (e.g., prohibiting data-security legislation, a non-mature code framework) were found. Therefore, we deem all DLT design candidates mature and suitable for the exemplary IoT use case.

Fifth, after all factors are included and a final set of DLT design candidates is generated, we conduct a *data collection* to use the developed operationalizations and to calculate the corresponding metrics. For our example, we gathered data on transaction latency and transactions per second for every DLT design out of the set of DLT design candidates from monitoring websites (e.g., [39]) and existing studies (e.g., [17, 40]) to calculate the values for consistency (S2) and throughput (P3).

Sixth, we compose a table to summarize the found results in the *analysis* step. The table includes all criteria in the columns and lists all DLT design candidates in the rows. For each DLT design candidate, every criterion is evaluated on whether the application requirement is fulfilled by the respective value of a DLT design candidate. If the calculated value of the metric fulfills the particular application requirement, we rate it *1*. If a criterion is only partially fulfilled or the application requirement is very close to the actual value of the DLT design, we rate it *0.5*. Otherwise, the table entry is rated *0*.

Finally, the *decision making* is performed by summing up the ratings for each DLT design candidate, leading to a total score. The DLT design with the highest score represents the assumed best suitable DLT design for the application on DLT. In the exemplary IoT use case, Hyperledger Indy scored best (cf. Table 4).

Table 4. Evaluation of consistency and throughput inside the process to compare DLT designs

	Consistency (< 2 s)		**Throughput** (> 1000 tps)		
DLT Design Candidates	**Value [s]**	**Score**	**Value [tps]**	**Score**	**Σ**
Ethereum	<15	0	20	0	0
Hashgraph	n.A.[2]	0	> 10,000	1	1
Hyperledger Indy	<2[1]	1	>3,500	1	2
Tezos	<60	0	40	0	0
IOTA	n.A.[2]	0	500-800	0.5	0.5

1: *Transaction latency is taken from a Hyperledger Fabric evaluation. Since both DLT designs employ a similar PBFT consensus algorithm with low latencies the data is comparable.*
2: *Transaction confirmation is non-deterministic*

5. Discussion

In this work, we present a comprehensive overview of DLT characteristics and their corresponding operationalization that can be used as criteria for a holistic comparison. Furthermore, we generated a first approach for a structured comparison process including research from DLT, distributed databases, and computer science. The validation of the generated process for comparison indicates its validity because the calculated suitability ranking is coherent with the intended purposes of the included DLT design candidates. For each use case, the process proposed a suitable DLT design for which proof of concepts in the respective fields have already been developed.

We regard the proposed operationalization as a first approach for the operationalization of DLT designs for a holistic comparison. Due to the strong focus of the included criteria for performance and security, the process will probably perform sufficiently for DLT designs that are designed for a similar purpose.

5.1. Implications

The presented operationalization of DLT characteristics supports a better comparison of DLT designs and helps to quantify their advantages. Using the operationalizations, the presented process supports the selection of a suitable DLT design for applications on DLT. The process facilitates and structures decision making for choosing a DLT design, which avoids unnecessary overhead and improves decision making.

The developed process synthesizes extant research on DLT from research in computer science as we evaluated different operationalizations for DLT characteristics and present a set of applicable operationalizations for selected criteria. Through the selection of operationalizations applicable to DLT characteristics, we support research on the identification of a suitable DLT design in information systems. The presented operationalizations of DLT characteristics support the quantification of dependencies between DLT characteristics, which broadens the scope for a comprehensive analysis of DLT designs [33] and helps to reveal the weaknesses of current DLT designs.

5.2. Limitations and future research

As the selection and operationalization of DLT characteristics are based primarily on literature and the presented process has only been evaluated in three use cases, we cannot generalize the presented process without limitation. As not all DLT characteristics have a corresponding characteristic in research on database or distributed systems monitoring (e.g., stale block rate), we developed own operationalizations. However, these measures have not been evaluated in the field, yet. Some operationalization concepts are restricted to or have a higher significance with blockchain-based DLT designs. During the data collection for the validation of the process, it became obvious that DLT and especially some of the chosen DLT designs form a fairly new research topic, thus discovering a need for additional research and practical measurement of DLT characteristics. This work relies on previous research on DLT characteristics [1, 24]. Thus, we also used the DLT property *usability* according to the examined literature.

In future research, the presented comparison process should be applied and evaluated in the field to assess its validity and overcome potential challenges in its usefulness. Additionally, the scoring model and the use of weights for the respective importance of DLT characteristics should be investigated. Since several DLT characteristics are not yet operationalizable, it is of high interest to generate new operationalizations for such DLT characteristics to obtain a holistic view of DLT designs and their benefits and potential constraints. Since the importance of cross-chain technology in DLT increases [25], additional metrics and operationalizations should be investigated and developed in order to make different cross-chain technology approaches comparable with each other.

6. References

[1] Kannengießer, N., S. Lins, T. Dehling, and A. Sunyaev, "What Does Not Fit Can be Made to Fit! Trade-Offs in Distributed Ledger Technology Designs", *52nd Hawaii International Conference on System Sciences*, (2019).

[2] O'Donoghue, O., A.A. Vazirani, D. Brindley, and E. Meinert, "Design Choices and Trade-Offs in Health Care Blockchain Implementations: Systematic Review", *Journal of Medical Internet Research 21*(5), 2019, pp. e12426.

[3] Nelson, J., M. Ali, R. Shea, and M.J. Freedman, "Extending Existing Blockchains with Virtualchain", 2016. https://www.zurich.ibm.com/dccl/papers/nelson_dccl.pdf

[4] Pass, R., and E. Shi, "The Sleepy Model of Consensus", *Advances in Cryptology – ASIACRYPT 2017*, (2017), 380–409.

[5] Eyal, I., and E.G. Sirer, "Majority is not Enough: Bitcoin Mining is Vulnerable", In *Financial Cryptography and Data Security*. 2014.

[6] Weber, I., V. Gramoli, A. Ponomarev, et al., "On Availability for Blockchain-Based Systems", *2017 IEEE 36th Symposium on Reliable Distributed Systems*, (2017), 64–73.

[7] Gervais, A., G.O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains", *Proceedings of the 2016*

ACM SIGSAC Conference on Computer and Communications Security, (2016), 3–16.

[8] Hyperledger Performance and Scale Working Group, "Hyperledger Blockchain Performance Metrics", 2018. https://www.hyperledger.org/wp-content/uploads/2018/10/HL_Whitepaper_Metrics_PDF_V1.01.pdf

[9] Lo, S.K., X. Xu, Y.K. Chiam, and Q. Lu, "Evaluating Suitability of Applying Blockchain", *2017 22nd International Conference on Engineering of Complex Computer Systems*, (2017), 158–161.

[10] Wüst, K., and A. Gervais, "Do you need a Blockchain?", 2017. https://eprint.iacr.org/2017/375.pdf

[11] Zhang, K., and H.-A. Jacobsen, "Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains", *2018 IEEE 38th International Conference on Distributed Computing Systems*, (2018), 1337–1346.

[12] Lamport, L., R. Shostak, and M. Pease, "The Byzantine Generals Problem", *ACM Transactions on Programming Languages and Systems 4*(3), 1982, pp. 382–401.

[13] Natoli, C., and V. Gramoli, "The Blockchain Anomaly", *2016 IEEE 15th International Symposium on Network Computing and Applications*, (2016), 310–317.

[14] Bermbach, D., E. Wittern, and S. Tai, *Cloud Service Benchmarking: Measuring Quality of Cloud Services from a Client Perspective*, 2017.

[15] Decker, C., and R. Wattenhofer, "Information propagation in the Bitcoin network", *IEEE P2P 2013 Proceedings*, (2013), 1–10.

[16] Bott, J., and U. Milkau, "Towards a framework for the evaluation and design of distributed ledger technologies in banking and payments.", *Journal of Payments Strategy & Systems 10*(2), 2016, pp. 153–171.

[17] Dinh, T.T.A., J. Wang, G. Chen, R. Liu, B.C. Ooi, and K.-L. Tan, "BLOCKBENCH: A Framework for Analyzing Private Blockchains", *ACM International Conference on Management of Data*, (2017), 1085–1100.

[18] Konheim, A.G., *Hashing in computer science: fifty years of slicing and dicing*, John Wiley & Sons, Hoboken, N.J, 2010.

[19] Glaser, F., and L. Bezzenberger, "Beyond Cryptocurrencies - A Taxonomy of Decentralized Consensus Systems", *23rd European Conference on Information Systems*, (2015), 1–18.

[20] Gray, J., ed., *The Benchmark handbook: for database and transaction processing systems*, M. Kaufmann Publishers, San Mateo, Calif, 1991.

[21] Huppler, K., "The Art of Building a Good Benchmark", *Performance Evaluation and Benchmarking*, (2009), 18–30.

[22] Hyperledger Foundation, "Hyperledger Caliper", 2018. https://github.com/hyperledger/caliper

[23] Kaner, C., S. Member, and W.P. Bond, "Software Engineering Metrics: What Do They Measure and How Do We Know?", *In METRICS 2004. IEEE CS*, (2004).

[24] Kannengießer, N., S. Lins, T. Dehling, and A. Sunyaev, "Mind the Gap: Trade-Offs between Distributed Ledger Technology Characteristics", *arXiv:1906.00861 [cs]*, 2019.

[25] Kannengießer, N., M. Pfister, M. Greulich, S. Lins, and A. Sunyaev, "Bridges Between Islands: Cross-Chain Technology for Distributed Ledger Technology", *53rd Hawaii International Conference on System Sciences*, (2020).

[26] Yli-Huumo, J., D. Ko, S. Choi, S. Park, and K. Smolander, "Where Is Current Research on Blockchain Technology?", *PLOS ONE 11*(10), 2016, pp. e0163477.

[27] Göbel, J., and A.E. Krzesinski, "Increased block size and Bitcoin blockchain dynamics", *2017 27th International Telecommunication Networks and Applications Conference*, (2017), 1–6.

[28] Ahmed, P.K., and M. Rafiq, "Integrated benchmarking: a holistic examination of select techniques for benchmarking analysis", *Benchmarking for Quality Management & Technology 5*(3), 1998, pp. 225–242.

[29] Doll, W.J., X. Deng, and J.A. Scazzero, "A process for post-implementation IT benchmarking", *Information & Management 41*(2), 2003, pp. 199–212.

[30] Hileman, G., and M. Rauchs, "2017 Global Blockchain Benchmarking Study", *SSRN Electronic Journal*, 2017.

[31] Deloitte, "Breaking blockchain open", 2018. https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/financial-services/cz-2018-deloitte-global-blockchain-survey.pdf

[32] Liu, Y., X. Liu, C. Tang, J. Wang, and L. Zhang, "Unlinkable Coin Mixing Scheme For Transaction Privacy Enhancement of Bitcoin", *IEEE Access 6*, 2018, pp. 1–1.

[33] Anceaume, E., T. Lajoie-Mazenc, R. Ludinard, and B. Sericola, "Safety analysis of Bitcoin improvement proposals", *2016 IEEE 15th International Symposium on Network Computing and Applications*, (2016), 318–325.

[34] Jogalekar, P.P., and C.M. Woodside, "A scalability metric for distributed computing applications in telecommunications", In *Teletraffic Contributions for the Information Age*. 1997, 101–110.

[35] McClusky, E.J., and S. Mitra, "Fault Tolerance", In *Computer Science Handbook*. Chapman & Hall/CRC, Boca Raton, Fla, 2004.

[36] Haerder, T., and A. Reuter, "Principles of transaction-oriented database recovery", *ACM Computing Surveys 15*(4), 1983, pp. 287–317.

[37] Goldwasser, S., and M. Bellare, *Lecture Notes on Cryptography*, Cambridge, Massachusetts, USA, 2008.

[38] Ellram, L.M., "Total cost of ownership: an analysis approach for purchasing", *International Journal of Physical Distribution & Logistics Management 25*(8), 1995, pp. 4–23.

[39] Etherscan, "Etherscan", https://etherscan.io/

[40] Nasir, Q., I.A. Qasse, M. Abu Talib, and A.B. Nassif, "Performance Analysis of Hyperledger Fabric Platforms", *Security and Communication Networks 2018*, 2018, pp. 1–14