Bridges Between Islands: Cross-Chain Technology for Distributed Ledger Technology

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

Michelle Pfister
Karlsruhe Institute of Technology
michelle.pfister@gmx.com

Malte Greulich
Karlsruhe Institute of Technology
malte.greulich@kit.edu

Sebastian Lins
Karlsruhe Institute of Technology
lins@kit.edu

Ali Sunyaev
Karlsruhe Institute of Technology
sunyaev@kit.edu

Abstract

Since the emergence of blockchain in 2008, we see a kaleidoscopic variety of applications built on distributed ledger technology (DLT), including applications for financial services, healthcare, or the Internet of Things. Each application comes with specific requirements for DLT characteristics (e.g., high throughput, scalability). However, trade-offs between DLT characteristics restrict the development of a DLT design (e.g., Ethereum, IOTA) that fits all use cases' requirements. Separated DLT designs emerged, each specialized to suite dedicated application requirements. To enable the development of more powerful applications on DLT, such DLT islands must be bridged. However, knowledge of cross-chain technology (CCT) is scattered across scientific and practical sources. Therefore, we examine this diverse body of knowledge and provide comprehensive insights into CCT by synthesizing its underlying characteristics, evolving patterns, and use cases. Our findings resolve contradictions in the literature and provide avenues for future research in an emerging scientific field.

1. Introduction

Distributed Ledger Technology (DLT) is one of the major technological innovations within the last decade [4, 37]. DLT enables the operation of a highly available and almost immutable shared, collaborative infrastructure for individuals and organizations [8]. Data is organized in an append-only ledger that is replicated across multiple storage devices, so-called nodes, that are synchronized by using a consensus mechanism. However, several challenges withhold the development of more powerful applications built on DLT. Developers face inherent trade-offs between DLT characteristics (e.g., *availability vs. consistency* [16, 26]), which is why the fulfillment of one DLT characteristic may impede others [26]. Consequently, while one DLT design can be well suited for a particular use case, it may not fit other use cases, ultimately inhibiting the development of a one-size-fits-all DLT design, and fueling the emergence of diverse DLT designs (e.g., Ethereum, IOTA, or Tezos) [16], each operating separately. To take advantage of individual DLT designs and to design more powerful applications, DLT designs must interoperate. Cross-Chain Technology (CCT), therefore, gained further attention among researchers and practitioners alike. For example, CCT could help extend the functionality of applications based on a private DLT design (e.g., Hyperledger Fabric) by allowing payments through a public DLT design (e.g., Ethereum). Thus, CCT is crucial to overcome the limitations of DLT designs and to prevent the emergence of separate 'islands' of distributed ledgers [14]. CCT enables different DLT designs to interoperate, thereby 'bridging islands' of distributed ledgers across industries and use contexts.

A growing number of CCT artifacts seek to enable cross-chain interoperability (e.g., Interledger or Polkadot). However, CCT artifacts pose different technical requirements (e.g., concerning locking or verification mechanisms) and non-technical requirements (e.g., concerning performance and security) on distributed ledgers. Consequently, developers must carefully compare CCT artifacts in order to choose an artifact that best suits their use case.

However, much of the current knowledge into CCT is not readily accessible to researchers and practitioners that would allow for such a comparison. We see three main reasons for this. First, existing comparisons of CCT artifacts (e.g., [18]) do not use a common set of characteristics, which limits the usefulness of such comparisons. Second, the literature into CCT proposes several patterns that describe the general functionality of CCT artifacts (e.g., sidechains [1]). Such patterns also inform the understanding of capabilities and limitations of such artifacts. Third, most of the current knowledge on CCT comes from blog posts (e.g., [4, 25]) or whitepapers (e.g., [34]). This scattered knowledge into CCT lacks transparency and rigor of (scientific) methods used to produce these insights. Likewise, the limited scientific research on CCT mainly refers to surveys for motivational purposes and seldom scrutinizes the claims made in blog posts (e.g., [4]) or whitepapers (e.g., [22]). Thus, there is a discrepancy between research and practice, which also hinders the scientific debate on DLT itself owing to the essential role of CCT for the diffusion of DLT at scale [14, 21].

In order to compare CCT artifacts more comprehensively concerning specific use case requirements, and to provide much-needed insights into CCT, a synthesis of characteristics and patterns of CCT artifacts is required. We thus seek to answer the following research question:

What are the characteristics and patterns of CCT?

By conducting a thorough review of the extant literature on CCT, we identified existing CCT artifacts and synthesized underlying characteristics. Then, we aggregated common functionalities across the identified artifacts into general patterns that enable easy and comprehensive comparison between artifacts. Finally, we compared these patterns with regard to particular use cases, which we identified during the literature review.

This work contributes to research and practice by defining common characteristics of CCT artifacts that allow researchers and developers to compare artifacts based on use case requirements. We provide a holistic comparison between patterns and their functioning, which offers generalized knowledge on CCT and which helps to overcome the scattered insights and prevailing disconnect between research and practice.

2. Theoretical background

2.1. Distributed ledger technology

DLT enables the operation of a distributed ledger, which is a special type of an append-only, distributed database that is particularly suited to the peculiarities of an untrustworthy environment [37]. DLT allows for the presence of Byzantine failures [20], which include arbitrarily crashed or unreachable nodes, network delays, or malicious behavior of nodes [20]. In DLT, new data is added to the ledger using transactions that are committed to each node's local replication. Owing to the use of cryptographic mechanisms (e.g., hashing), data that has been recorded in the distributed ledger typically cannot be removed or altered.

To reach consistency between the replications stored on nodes of a distributed ledger, each DLT design employs a consensus mechanism. A consensus mechanism is an algorithm used to negotiate the current valid state of the ledger between different nodes of the distributed ledger. Due to inherent trade-offs that pertain to consensus finding in distributed systems, consensus mechanisms either reach total finality or probabilistic finality [11, 28]. Total finality is reached if all nodes agree on the same replication of the ledger. In probabilistic finality, it is only assumed that data is finalized to a certain probability, which depends on the number of successors after a transaction has been recorded. For example, such successors are blocks, which serve as a superordinate data structure, which includes transactions. The minimum number of successors that are required to assume stored data as finalized to a certain probability determines the so-called confirmation latency of a DLT design (e.g., 1 h in Bitcoin to append 6 blocks).

In general, there are two types of DLT designs that differ in their hierarchical structure: public and permissioned DLT designs. In public DLT designs, anyone can join the distributed ledger with an own node, which has equal permissions concerning reading from and writing to the distributed ledger. In contrast, permissioned DLT designs exclude nodes from operations (e.g., transaction validation, joining the distributed ledger) as they employ a permission model. Such permission models strongly influence the applicability of consensus mechanisms due to trade-offs between characteristics that pertain to distributed systems [11, 28]. For example, public DLT designs predominantly employ consensus mechanisms that only reach probabilistic finality (e.g., Nakamoto consensus). In contrast, consensus mechanisms of most permissioned DLT designs reach total finality and, thus, do not fork or only fork for a short period but can only include a limited number of nodes into consensus finding.

2.2. Smart contracts

Several DLT designs enable the deployment and the execution of customized software programs, so-called smart contracts. Smart contracts allow for the expression of formalized conditions in program code for the issuance of transactions [5]. Initially, smart contacts were limited to (un-)locking of assets stored on a distributed ledger (e.g., using hash locks, time locks, and multi-signatures) [13]. To increase developers' flexibility for implementing more expressive smart contracts, an environment for the execution of Turing-complete smart contracts was developed [5]. Today, smart contracts can store assets and issue transactions once the smart contract's formalized conditions are met. Such conditions can relate to data stored on the same distributed ledger (on-chain) as well as data from external sources (off-chain), so-called oracles [36]. Smart contracts are also crucial to enable atomic communication between distributed ledgers (e.g., asset transfers) and, thus, cross-chain interoperability.

2.3. Cross-chain technology

Interoperability of a DLT design refers to its ability to retrieve data from or exchange data with external systems. CCT helps to achieve interoperability by enabling data exchange between DLT designs or with external systems. Such data exchanges can increase a DLT designs' flexibility (e.g., [14, 32]), overcome performance issues (e.g., [24, 30]), and increase security of DLT designs (e.g., [1, 29]). For example, poor scalability and low throughput can be tackled using *sharding* [24]. In *sharding*, a distributed ledger is split into smaller chunks, which can be managed independently and enables parallel processing of transactions to increase throughput and scalability [24].

The development of CCT artifacts is mainly driven by organizations that publish the functioning of their own-developed artifact in whitepapers (e.g., Interledger, Wanchain). Several articles present a classification of such artifacts into patterns (e.g., [4, 22]), which are abstract descriptions of the functioning of assigned artifacts. However, the proposed patterns lack empirical evidence, which makes the results hard to reproduce.

3. Methodology

We applied a three-step research approach. First, we conducted a literature review to extract CCT artifacts, their characteristics, existing patterns, and use cases. Our descriptive literature review was guided by recommendations for literature reviews in the information systems field [3, 17, 33]. Second, we derived patterns from the identified artifacts. Third, we compared the derived patterns and explain dependencies between characteristics inherent to artifacts and patterns.

3.1. Literature search

To identify publications addressing CCT, we searched scientific databases that we deemed representative as they cover the top computer science and information systems conferences and journals: *EBSCOhost*, *IEEE Xplore*, *AIS eLibrary*, *ScienceDirect*, *ACM Digital Library*, *ProQuest*, and *Springer*. To cover a broad set of publications, we searched each database with the following string in title, abstracts, and keywords: *('Distributed Ledger Technolog*' OR 'DLT*' OR 'Blockchain*') AND ('Interoperab*' OR 'Cross-chain*' OR 'Sidechain*' OR 'Multi-chain*' OR 'Interconnect*' OR 'Connect*')*. As whitepapers in this particular context are a crucial source of knowledge on CCT, we additionally applied the search string to the search engine DuckDuckGo.

We identified 611 articles in this initial search as of May 2019. To identify and filter articles, we first checked the relevance of each article by analyzing title, abstract, and keywords. If any indication for relevance appeared, the article was marked for further analysis. We excluded articles that were non-English articles (5), grey literature (i.e., books, news articles; 2), duplicate articles (31), and off-topic (504). This first relevancy assessment resulted in a sample of 69 articles deemed to be potentially relevant. Afterward, a fine-grained relevance validation was made by reading the articles in detail, resulting in a sample of 38 relevant articles on which we applied a forward and backward search. For each of these articles, we applied the exclusion criteria, which resulted in a final set of 81 relevant articles.

3.2. Data analysis

We analyzed the relevant articles by applying the coding rules proposed by Lacity et al. [19]. First, we carefully read and analyzed the relevant articles independently to identify and code artifacts, corresponding characteristics, use cases, and patterns in the context of CCT. For each extracted code, we recorded a name, a description, and the original source [19] and came up with a preliminary set of 57 artifacts, 139 characteristics, 19 patterns, and 15 use cases for CCT. Since different people name things differently, it is crucial to avoid semantic ambiguities for the validity of qualitative analysis (e.g., different terminology for the same characteristics) [19]. Therefore, we aggregated the codes into master codes (e.g., aggregating throughput and transaction volume into the master code throughput). If an identified code fitted into an existing master code, we assigned it accordingly; otherwise, we created a new master code. We repeated the aggregation of master codes two times and discussed and validated the generated master codes with two additional researchers, who are knowledgeable in the domain of DLT. The goals were to agree on a set of master codes for artifacts, characteristics, patterns, and use cases and to reach theoretical saturation, in order to stabilize the list of master codes. We applied the same coding approach for artifacts, characteristics, patterns, and use cases. We came up with a final set of 57 distinct artifacts, 37 distinct characteristics, 3 distinct patterns, 1 hybrid pattern, and 4 use cases for CCT. We assume to have reached theoretical saturation because no new master codes were identified for the last 20 articles. Please find details on the literature search and coding in the supplementary online material (*https://bit.ly/2lUWprE*).

Second, we used an inductive grouping approach to classify master codes for characteristics into superordinate properties. We generated the properties under consideration of the characteristics' predominant classification in extant literature. For example, we assigned the master code *throughput* to the property *performance*.

Third, we evaluated the coded patterns by considering the identified artifacts. We compared the artifacts with regards to their values for the particular characteristics and assigned the artifacts accordingly to the coded patterns. Artifacts that use a similar approach to achieve cross-chain interoperability were assigned to the same pattern (e.g., Dogeethereum and InfiniteChain were assigned to the sidechain pattern).

4. Results

4.1. Properties and characteristics of CCT

For parsimoniousness, we briefly introduce the properties of CCT artifacts in Table 1 and refer to Table 2 for a description of the associated characteristics. Please see a full list of the coding and the identified artifacts (e.g., BTCRelay, Polkadot, and Wanchain) in the online material (*https://bit.ly/2lUWprE*).

4.2. Patterns of cross-chain technology

Our analysis of the artifacts and their characteristics (cf. Table 1) revealed three distinct patterns: *manual asset exchange (MAE)*, *notary schemes*, and *relays*. Additionally, we identified a fourth, *hybrid* pattern, which includes the three distinct patterns. By and large, the patterns differ in terms of characteristics that are related to the *networking* property (e.g., communication). The *networking* property strongly influences other properties. For example, the *procedure* characteristic (e.g., atomic cross-chain swap [13], three phase commit [15]) influences transaction speed as it determines the number of verifications and requests between the distributed ledgers. We describe these implications in more detail in Section 4.4.

Table 1. Properties of CCT artifacts

Property	**Description**
Administration	Characteristics concerned with the management and assignment of responsibilities for the operation and maintenance of an artifact.
Flexibility	Characteristics concerned with the developers' freedom of customizability of the artifact, the connected DLT designs, and applications on these distributed ledgers or artifacts.
Performance	Characteristics referring to the effectiveness of a data exchange measured by accuracy, completeness, cost, and speed.
Security	Characteristics concerned with the preservation of confidentiality, integrity, and availability of data stored on a distributed ledger.
Networking	Characteristics concerned with the structure and processes that enable the exchange of data between distributed ledgers.

4.2.1. Manual Asset Exchange. A MAE is the simplest CCT pattern. MAEs follow the typical lifecycle of financial transactions: settlement, order matching, and clearance. In the first stage, *A* settles a new asset exchange order as *A* locks assets on the corresponding distributed ledgers using a certain secret (e.g., pre-image of a hash value, a private key for an account). In the second stage, *A* must find a corresponding exchange partner *B* to eventually agree upon the asset exchange rate for the respective orders (e.g., 1 Bitcoin in reward for 32.5 Ether). In MAEs, such order matching is conducted off-chain, for example, by a third party or through personal interaction. After *A* and *B* agreed on the exchange, *B* locks assets on the corresponding distributed ledger. In the third stage, the order clearance, the actual assets exchange takes place. Therefore, *A* and *B* exchange their secrets to unlock the locked assets respectively. MAEs have certain drawbacks and can only be applied for asset exchanges (as opposed to the other patterns). Typically, MAEs do not employ automated order matching and do not require an artifact because the asset exchange can solely base on a personal agreement of parties. However, MAEs can be vulnerable to fraud. If *A* receives *B*'s secret first and there is no mechanism to unlock *A*'s secret in return, A could use *B*'s secret to unlock *B*'s assets without transferring its own assets to *B*. *A* would thus not complete the exchange correctly. In order to prevent exchange partners from carrying out fraudulent activities that could lead to financial losses, atomicity is crucial for the exchange of assets [13]. The most prominent protocol for MAEs to achieve atomicity is the atomic cross-chain swap protocol [13], which is based on hashed time-locked contracts [13, 25]. *A* hashed time-lock contract locks assets using a hash value s_h during a period t. Assets are unlocked when a pre-image s is provided within a period t with a hash value equal to s_h.

4.2.2. Notary Schemes. In notary schemes, a trusted third party establishes the connection between distributed ledgers [22]. The notary scheme provides an infrastructure (e.g., multi-miner) and related services (e.g., order matching) to facilitate asset transfers or similar actions (e.g., execution of a smart contract). Before an action on a distributed ledger is executed, a notary must first agree that a certain event (e.g., commitment of a transaction) on another distributed ledger took place. For instance, consider the case of a cryptocurrency exchange. Here, a notary must first verify that a transaction was completed successfully on distributed ledger *A*, before it issues the corresponding transaction to distributed ledger *B*. Thus, the data exchange between distributed ledgers is completely managed by notaries. Notary schemes follow a centralized architecture to achieve cross-chain interoperability [14, 22].

A notary scheme can correspond to either a single notary (centralized notary scheme or centralized exchange), or a consortium of notaries (decentralized notary scheme or decentralized exchange) [31]. In centralized notary schemes, a single notary may set up and operate a node per connected distributed ledger. For example, when a

Table 2. Characteristics of cross-chain technology artifacts

Prop-erty	Characteristic	Description
Admin-istration	Accountability	The presence of a party responsible and liable for the correct operation of the artifact.
	Auditability	The degree to which the artifact behavior can be analyzed (e.g., examination and traceability of stored transactions).
	Compliance	The ability of an artifact to adhere to laws and regulations.
	Governance	The process and assignment of responsibilities for deciding to change the artifact and connected distributed ledgers.
Flexibility	Development Support	The degree to which the community or organization that maintains an artifact provides developers with documentations, guidelines, and additional features to facilitate the setup and maintenance of the connection to the artifact.
	Feature Scope	The artifact's feature richness and appropriateness for use cases.
	Maintainability	The ease with which developers can update the inter-connected distributed ledgers, the applied artifact, or the associated applications on DLT.
	Openness	The ease with which an artifact enables developers to independently register or unregister distributed ledgers in the multi-chain network.
	Supported DLT Designs	The variety of DLT designs the artifact can support.
Networking	Communication Approach	The direction of the data flow, the direct or indirect communication, the used data structures (e.g., tokens), and the necessary events (e.g., successful transaction commit).
	Locking Mechanism	A mechanism that securely locks assets on one ledger and unlocks transactions on another (e.g., hash locking, time locking, hash-time lock-contracts).
	Procedure	The procedure determined by the artifact to reliably fulfil its feature scope.
	Routing	The management of forwarding data sent over the network via particular entities.
	Topology	The structural organization of the inter-connection of distributed ledgers.
	Verification Mechanism	The applied mechanism to verify that transactions on one distributed ledger have been finalized.
Performance	Cost	The total sum of (monetary) cost to execute a transaction, including, for example, transaction fees and collaterals.
	Resource Consumption	The required amount of computational resources (e.g., bandwidth or storage space) to operate an artifact.
	Scalability	The ability of the distributed ledger to appropriately adapt to increasing and decreasing workload.
	Throughput	The number of transactions that can be included in the ledger within a given period.
	Transaction Speed	The period between a cross-chain transaction is issued and its (probabilistic) finalization at the distributed ledger(s).
Security	Atomicity	The ability of an artifact to guarantee that each (cross-chain) transaction either succeeds completely or fails completely.
	Availability	The probability to which an artifact is in a functioning state at a random point in time.
	Censorship-Resistance	The probability of transactions being intentionally delayed or even dropped.
	Confidentiality	The degree to which unauthorized access to stored data is prevented.
	Consistency	The homogeneity of data stored by all nodes participating in a DLT design without contradictions.
	Durability	The degree of durability that ensures transactions are saved permanently and do not accidentally disappear or get erased, even during a database crash.
	Finality	The type of finality (e.g., probabilistic or total) and time that is needed to undergo a threshold to consider a transaction finalized.
	Fault Tolerance	The degree to which the artifact is able to cope with Byzantine failures (e.g., node failure, network delays).
	Isolation	The degree to which the operations on a distributed ledger can affect or are affected by operations in concurrent transactions.
	Incentive Mechanism	The applied reward mechanism that motivates maintainers of an artifact to act honestly and to share computational resources (e.g., mining or merge mining).
	Integrity	The degree to which the artifact protects exchanged data from being modified without consent of the data owner.
	Ledger Independence	The degree to which at least two distributed ledgers, which are connected via an artifact, remain autonomous.
	Liveness	The ability of concurrent systems to make progress despite concurrently executing processes.
	Non-Repudiation	The ease of proving participation in transactions.
	Reliability	The period of time during which a distributed ledger is correctly functioning.
	Safety	The degree to which the artifact is able to avert or not cause (economic) loss (e.g., loss of assets).
	Transparency	The degree to which identities and their activities are visible or traceable to others.

notary decides to enable assets transfers from Bitcoin to Ethereum, the notary sets up a Bitcoin and Ethereum node to manage the receive and the issuance of transactions on both distributed ledgers. The notary alone confirms if an event occurred (e.g., transaction reception) and triggers the corresponding event (e.g., transaction issuance). To democratize the confirmation of events among a consortium of notaries, to increase transparency, and to increase the availability of the exchange service decentralized notary schemes have been introduced [31]. To issue a corresponding transaction for an event in decentralized notary schemes, trusted third parties often share, for instance, a distributed private key [22, 23] or employ a multi-signature wallet [35]. Only if a certain number of notaries confirms the event (e.g., locking assets on distributed ledger *A*), a corresponding event is executed (e.g., unlocking assets on distributed ledger *B*).

4.2.3. Sidechains (relays). In general, sidechains are subordinate distributed ledgers that are connected to a central distributed ledger (main chain), such as Bitcoin or Ethereum. Sidechains are technically independent of the main chain and, thus, can have their own consensus mechanism, tokens, and miners. Initially, sidechains were developed to enhance the extensibility of existing distributed ledgers through asset transfers. Sidechains can read and verify data from the main chain [1], which is important, for example, to transfer assets from the main chain to the sidechain (one-way peg [1]). In such asset transfers, several assets (e.g., coins) are locked on the main chain. The locking of assets is confirmed by the verification mechanism of the destination sidechain, which eventually unlocks (or generates) a corresponding amount of native tokens [1, 5].

Several articles refer sidechains almost exclusively to the use in combination with the Bitcoin blockchain [6, 22]. For example, BTC Relay extends the Bitcoin blockchain with support for smart contracts [10]. The original sidechains only allow for asset transfers or forwarding information in one direction: from the main chain to the sidechain (cf. one-way peg [2]). Meanwhile, the concept of sidechains has been enhanced and implemented in artifacts that allow for bidirectional communication of distributed ledgers. This subordinate pattern of sidechains in which distributed ledgers communicate bidirectionally is called a two-way peg [1]. To be able to transfer the assets back to the main chain or to another sidechain, the main chain must also be able to verify data on the relay, which decreases the number of supported DLT designs.

4.2.4. Hybrid solutions. There are also hybrid solutions, which combine certain aspects of the previously explained patterns. Hybrid solutions, for example, help to set up a two-way pegged sidechain although one of the DLT designs does not support an appropriate verification mechanism (e.g., Simple Payment Verification (SPV) [12]). In such situations, a notary scheme replaces the ability of the particular relay to recognize and validate included transactions and let trusted notaries provide this information [1]. For instance, such a federated pegged relay is implemented in Rootstock [29]. Rootstock sets up a two-way peg with the Bitcoin blockchain although the Bitcoin blockchain is not able to perform light client proofs on other distributed ledgers [29]. Furthermore, Rootstock implements a mechanism to empower the notaries to sign valid transactions as the overall hashing power in the Rootstock chain is below 5 % of the Bitcoin hashing power to prevent double-spending.

4.3. Use cases for cross-chain technology

The literature review revealed four use cases: *asset transfers*, *cross-chain oracles*, and *cross-chain smart contracts*. In *asset transfers*, assets are moved from one distributed ledger to another. As a special form of asset transfer, we identified *asset exchanges*, which allow users to spend assets of one distributed ledger in return of assets from another distributed ledger (e.g., trading of cryptocurrencies or other assets). Asset exchanges pose a requirement for atomicity to prevent financial loss.

Rather than moving assets, *cross-chain oracles*, in contrast, provide information from one distributed ledger to another [5]. Thus, cross-chain oracles can be employed to verify that certain events (e.g., a transaction) occurred on another distributed ledger (e.g., SPV [12]) enabling, for example, the migration of data from one distributed ledger to another or the interaction of distributed ledger in supply chain management (SCM). In SCM, one distributed ledger for payments could request the current state of a shipment on another distributed ledger for tracking to execute conditional payments. Cross-chain oracles are of particular importance for asset encumbrance (cf. [4, 18]), which is the ability of a ledger to lock assets and unlocking the locked assets if a certain predefined event on another distributed ledger occurs, which is of particular importance for the migration of a ledger from its DLT design to another.

Cross-chain smart contracts describe the ability to trigger the execution of a smart contract on another distributed ledger, which can increase the level of automation in the previous SCM example [6]. In contrast to *cross-chain oracles*, the execution of cross-chain smart contracts requires the issuance of transactions on the destination chain, which causes a change of state of the distributed ledger.

According to literature, there are also other use cases for CCT such as sharding (e.g., [4, 30]). We argue that

Table 3. Pattern comparison with regards to the networking property

		MAE[A]	**Notary Scheme**		**Sidechain (Relay)**		**Hybrid Solution**
			Centralized	Decentralized	One-Way Peg	Two-Way Peg	
Networking	Procedure	Atomic cross-chain swap Non-atomic swap	Single notary confirms events	Consortium of notaries confirm events	Sidechain verifies locking of assets on the main chain	Sidechain verifies locking of assets on other chain	Relays or notaries verify locking of assets on other relays
	Communication	None Bidirectional Off-Chain	Indirect Bidirectional Off-Chain	Indirect Bidirectional On-Chain or Off-Chain	Direct Unidirectional On-Chain	Direct Bidirectional On-Chain	Direct or indirect[B] Bidirectional On-Chain and partially Off-Chain
	Locking Mechanism(excerpt)	Hash-lock Hashed time-lock contract	Own Accounts	Distributed Private Key Multi-Signature Wallet	Smart Contract[C]	Smart Contract[C]	Hash-lock Hashed time-lock
	Routing	Off-Chain	One Central Multi-Node	Connector Node(s) Consortium of Notaries	Main Chain Smart Contract Sidechain	Main Chain Smart Contract Sidechain	Connector Nodes Relays
	Topology[E]	$N \leftrightarrow N$	$N \leftrightarrow C \leftrightarrow N$	$N \leftrightarrow C \leftrightarrow N$	$1 \rightarrow 1$	$N \leftrightarrow N$	$1 \rightarrow C \rightarrow 1$[D] $1 \leftarrow 1$[D]
	Verification Mechanism (excerpt)	Manual verification via block headers	Verification by a single notary (e.g., SPV[F])	Verification by a group of notaries (e.g., SPV[F])	SPV[C,F]	SPV[C,F]	SPV[C,F] and notary observations
	Artifacts (excerpt)	An artifact is not required	Binance Coinbase Kraken	Polkadot Interledger InterChain	BTC Relay	Dodgethereum InfiniteChain	Rootstock

A: *Manual Asset Exchange*
B: *Depends on the transfer direction*
C: *Automated through smart contracts*
D: *DLT design not natively supporting the*
E: *Topology uses the following notation: C represents a connector entity (e.g., a notary), N represents an arbitrary number of distributed ledgers, 1 represents one distributed ledger*
F: *Simple payment verification [1]*

such use cases are combinations of the previously presented use cases and, thus, should not be grouped into an additional class of use cases. For example, sharding can be assigned to the *cross-chain oracle* use case.

4.4. Comparison of patterns

In this section, we outline differences between the three patterns presented in section 4.2. Since the *networking* property mainly influences the other properties, we now explain the implications of the *networking* property on the others. For the sake of comprehensibility, we summarized the differences between patterns with regards to the *networking* property in Table 3.

Administration. Concerning the administration of an artifact, MAEs appear as an easy to setup pattern because there are no external dependencies compared to notaries or sidechains. In notary schemes, due to the fact that governance of notaries is distributed among only a few notaries of a consortium (or even only one notary), corresponding artifacts are easier to govern than relays. In relays, a public DLT design such as Bitcoin may form the main chain, which comes with considerable challenges owing to its high level of decentralization.

Flexibility. MAEs do not come with high technical requirements towards DLT designs because basic locking mechanisms can be employed (e.g., hashed time-lock contracts). Due to the low technical requirements, maintaining a MAE is technically easier compared to notary schemes, which initially require to set up an infrastructure and its maintenance during operation. Notary schemes are more flexible than relays because they can be easier extended by new distributed ledgers and pose almost no technical requirements towards distributed ledgers that should be connected. Notaries can easily add or dispend distributed ledgers by setting up a respective connector node, while it is harder to block certain distributed ledgers from the artifact in sidechains.

Performance. In MAEs, order matching can be challenging because there are no mechanisms to automate the process of finding an exchange partner. Furthermore, the involved parties are responsible for the verification of the relevant transaction locking. In contrast, notary schemes accelerate cross-chain transactions due to the engagement of a trusted third party, which manages order matching, transaction verification and which is responsible for the governance and maintenance of the artifact. Thus, notaries are often liable for potential faults in the exchange. However, openness is sacrificed because notaries decide on which DLT designs is supported, which impedes censorship resistance. In all patterns, the transaction speed in MAEs strongly depends on the respective confirmation latency of the distributed ledgers, which must hold to ensure atomicity of asset exchanges [13]. Thus, overall performance is impacted by the supported DLT designs (public or permissioned) and their applied consensus mechanisms.

Security. In MAEs, safety depends on the trust model of the connected distributed ledgers, rigor in following the proposed *procedures*, and the type of finality the consensus mechanism provides (probabilistic or total). Atomicity is of high importance in all derived patterns, especially, when it comes to the transfer of assets (e.g., coins of a cryptocurrency), where, for example, financial loss may occur. However, the examined artifacts achieve atomicity through different procedures (e.g., atomic cross-chain swaps [13] or additional consensus mechanisms [27]), which strongly influences transaction speed. In DLT designs that employ probabilistic finality, *t* in atomic cross-chain swaps should be chosen under consideration of the estimated confirmation latency of the involved DLT designs. Otherwise, atomicity of the exchange cannot be guaranteed because transactions may not be included in the main branch due to forks [18]. This is a particular challenge when establishing interoperability with a permissionless distributed ledger, which typically comes with long confirmation latency and only probabilistic finality (e.g., Bitcoin or Ethereum). In terms of decentralization, MAEs and sidechains currently appear as the most decentralized pattern because the two distributed ledgers can communicate directly with each other. Due to the higher level of decentralization in MAEs and relays compared to notary schemes, censorship resistance appears more likely in these patterns than in notaries.

The applied *routing* influences *availability* of the artifact, *reliability* of cross-chain transactions, and *censorship resistance*. For example, there are artifacts (e.g., Blockchain Router) that employ particular connector nodes for the routing of messaged between distributed ledgers. Consequently, the routing comes with less redundancy than if all nodes of the distributed ledgers were employed into the routing (e.g., BTC Relay), which is why availability and reliability of the connection are decreased.

5. Discussion

Although the 57 CCT artifacts have been developed for different use cases to bridge applications from different industries, all CCT artifacts aim to solve common challenges such as high performance and high security. For example, all artifacts require that the data transfer from one distributed ledger to another is atomic [4]. This is predominantly ensured by waiting a sufficient period until finality can be assumed or is reached in fact [1]. Since atomicity strongly depends on the consensus mechanism of the connected DLT designs, the type of DLT design must be considered when deciding for a CCT artifact in the first place. As public blockchains more often employ a consensus algorithm with slow or only probabilistic finality (e.g., Proof of Work), this endangers the atomicity of transactions [14]. At the same time waiting periods become longer and the CCT artifact less performant. Even though slow finality is more likely to be linked to public blockchains [14], there are also solutions for public blockchains that ensure rapid finality, and, therefore, do not result in weakened security or less performance. The type of a distributed ledger also impacts the applicability of the presented patterns. The applicability of certain identified CCT patterns for combinations of public and private distributed ledgers has been explicitly stated in literature (e.g., notary schemes [23, 34] or hybrid solutions [4]). For example, the notary schemes InterChain and Wanchain can connect public distributed ledgers with private or public ones [9]. Rootstock as a hybrid solution still connects only public distributed ledgers. Generally, hybrid solutions such as the federated pegged sidechains are also applicable to private distributed ledgers [4]. MAEs and sidechains appear applicable to public and private distributed ledgers under certain conditions. MAEs require the involved parties to own accounts on both distributed ledgers [13]. Thus, MAEs are only applicable if both parties have access to both distributed ledgers, which limits the applicability of MAEs. For example, users of a private distributed ledger cannot exchange assets with users of a public distributed ledger without additional mechanisms. In sidechains, the distributed ledgers need access to the destination distributed ledger in order to verify the transaction by using SPV [1, 37]. Usually, relayers provide the required data to proceed with a SPV from the original distributed ledger to the destination distributed ledger [4, 10]. Therefore, if a private distributed ledger is involved the relayers must have access to the private distributed ledger and the information necessary to validate a transaction must be insensitive enough to be shared on a public ledger.

During our review of the literature, we recognized that the discussion on CCT parallels the general discussion on DLT: both literature streams are strongly driven by innovations from practice. Progress made by practitioners is largely documented in more practical publications, which lack a scientific methodology. Thus, such results are hardly reproducible. We recognized that practitioners and researchers predominantly rely on own developed metrics to compare CCT artifacts or concepts (e.g., [5]), which points out that the topic of CCT is still in its infancy and no common understanding of CCT patterns and their characteristics has become widely adopted. The scientific debate about CCT and how it will affect DLT in general is still in its infancy. It also became apparent that only a little progress has been made in the development of new patterns, which could overcome prevalent disadvantages of the relay pattern (see Section 4.2.2). Instead, artifacts became more efficient in terms of routing and communication and offer even more flexibility for developers. For example, artifacts such as OneLedger, Ontology, and Wanchain provide an infrastructure to deploy smart contracts and, thus, enable customization of the cross-chain protocol. Some of these artifacts already offer Software Development Kits for the deployment of a developed smart contract on distributed ledgers connected to the artifact (e.g., OneLedger). However, most of such feature-rich artifacts come at the cost of openness as they are maintained by notaries (e.g., Polkadot).

Our results contradict the wide-spread classification of patterns in CCT because we identified slightly different patterns than proposed in extant literature: hash locking, notary scheme, relays (e.g., [4, 18]). Although hash locking is predominantly mentioned as a pattern next to relays and notary schemes (e.g., [5, 7, 22]), we assigned hash locking to the *locking* characteristic because hash locking can be applied in all of the identified patterns. Therefore, the generated overview of patterns and their characteristics contributes to a common understanding of CCT, which was previously fragmented and empirically not supported.

5.1. Implications

Our work has several implications for research and practice. The presented characteristics of CCT artifacts support practitioners when comparing artifacts in CCT with each other. Furthermore, the aggregation of artifacts into patterns helps to obtain a first impression of the functioning of artifacts assigned to a particular pattern, their advantages, and disadvantages. To facilitate the selection of an artifact for a particular use case, we provide insights into three classes of use cases, their individual requirements, and present recommendations for what pattern could fit best. Furthermore, artifacts can be comprehensively compared by considering the identified characteristics, which supports the development of new artifacts or even new patterns.

We contribute to research through the revision of the dominant classification of patterns, use cases, and characteristics in CCT. Our comprehensive and systematic review of the literature offers researchers a solid basis for a more profound discourse on CCT and its implications for the emergence of DLT in general. Furthermore, we contribute to research in requirements engineering because the generated overview of characteristics can be adapted to models in the requirements analysis in software engineering.

5.2. Limitations and future research

The presented results are only generalizable on a limited scale because they are merely derived from scientific literature but mostly from whitepapers issued by practitioners in the domain of DLT and CCT. Furthermore, several articles are proposals and the respective implementations have not been developed, yet. Thus, there is no evidence of how well the proposed architecture operates. All of the identified artifacts refer to the DLT concept blockchain. Thus, alternative DLT concepts (e.g., transaction-based Directed Acyclic Graphs) are not included in our study.

In order to validate the presented results, the coding should be evaluated, for example, by using natural text analysis (NTA) or focus groups. Since the investigated CCT artifacts are not in use or not even developed for the most part (e.g., [30, 34]), the applicability of these artifacts can hardly be proven. Thus, we aim to investigate their potentials and constraints for various industrial use cases in future work.

6. Conclusion

Extant research on CCT predominantly builds upon findings in practice, which is why only scattered insight into CCT has been provided so far. By synthesizing prior literature and research and defining CCT characteristics and patterns, our study provides a foundation for a common understanding of CCT and should enliven the discussion on CCT in research and practice. As the number of CCT artifacts will probably continue to increase, it becomes apparent that developers will need decision support when choosing an artifact. Thus, future research should investigate how such decision support should be designed to communicate potential drawbacks for the particular application DLT. Therefore, the characteristics of CCT and DLT should be operationalized to conduct quantitative studies on CCT artifacts.

7. References

[1] Back, A., M. Corallo, L. Dashjr, et al., “Enabling blockchain innovations with pegged sidechains”, *https://www.blockstream.com/sidechains.pdf*, 2014.

[2] Back, A., M. Corallo, L. Dashjr, et al., “Enabling blockchain innovations with pegged sidechains”, 2014. https://www.blockstream.com/sidechains.pdf

[3] vom Brocke, J., A. Simons, K. Riemer, B. Niehaves, and R. Platfaut, “Standing on the Shoulders of Giants: Challenges and Recommendations of Literature Search in Information Systems Research.”, *Communications of the AIS 37*(9), 2015, pp. 205–224.

[4] Buterin, V., “Chain Interoperability [white paper]”, 2016. http://www.r3cev.com/s/Chain-Interoperability-8g6f.pdf

[5] Buterin, V., “Ethereum Whitepaper”, 2018. https://github.com/ethereum/wiki/wiki/White-Paper/f18902f4e7fb21dc92b37e8a0963eec4b3f4793a

[6] Chen, S., H. Wang, L.-J. Zhang, et al., eds., “InfiniteChain: A Multi-chain Architecture with Distributed Auditing of Sidechains for Public Blockchains”, Springer International Publishing (2018).

[7] Deng, L., H. Chen, J. Zeng, and L.-J. Zhang, “Research on Cross-Chain Technology Based on Sidechain and Hash-Locking”, *Edge Computing*, (2018), 144–151.

[8] Di Ciccio, C., A. Cecconi, M. Dumas, et al., “Blockchain Support for Collaborative Business Processes”, *Informatik Spektrum*, 2019.

[9] Ding, D., T. Duan, L. Jia, K. Li, Z. Li, and Y. Sun, “InterChain: A Framework to Support BlockchainInteroperability”, (2018).

[10] Ethereum, “BTC Relay”, *GitHub*, 2016. https://github.com/ethereum/btcrelay/tree/master

[11] Fischer, M.J., N.A. Lynch, and M.S. Paterson, “Impossibility of distributed consensus with one faulty process”, *Journal of the ACM 32*(2), 1985, pp. 374–382.

[12] Frey, D., M.X. Makkes, P.-L. Roman, F. Taïani, and S. Voulgaris, “Bringing Secure Bitcoin Transactions to Your Smartphone”, *15th International Workshop on Adaptive and Reflective Middleware*, (2016), 1–6.

[13] Herlihy, M., “Atomic Cross-Chain Swaps”, *2018 ACM Symposium on Principles of Distributed Computing*, (2018), 245–254.

[14] Jin, H., J. Xiao, and X. Dai, eds., “Towards a Novel Architecture for Enabling Interoperability amongst Multiple Blockchains”, *2018 IEEE 38th International Conference on Distributed Computing Systems*, (2018).

[15] Kan, L., Y. Wei, A.H. Muhammad, W. Siyuan, G. Linchao, and H. Kai, eds., “A Multiple Blockchains Architecture on Inter-Blockchain Communication”, *2018 IEEE International Conference on Software Quality, Reliability and Security Companion*, (2018).

[16] 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).

[17] Kitchenham, B., O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic Literature Reviews in Software Engineering”, *Information and Software Technology 51*(1), 2009, pp. 7–15.

[18] Koens, T., and E. Poll, “Assessing Interoperability Solutions for Distributed Ledgers Extended version”, 2018. https://www.ingwb.com/media/2667864/assessing-interoperability-solutions-for-distributed-ledgers.pdf

[19] Lacity, M.C., S. Khan, A. Yan, and L.P. Willcocks, “A review of the IT outsourcing empirical literature and future research directions”, *Journal of Information Technology 25*(4), 2010, pp. 395–433.

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

[21] Lima, C., “Developing Open and Interoperable DLT/Blockchain Standards”, *Computer 51*(11), 2018, pp. 106–111.

[22] Liu, S., B. Tekinerdogan, M. Aoyama, et al., eds., “Research on Cross-Chain Technology Based on Sidechain and Hash-Locking”, (2018).

[23] Lu, J., B. Yang, Z. Liang, et al., “Wanchain”, 2017. https://wanchain.org/files/Wanchain-Whitepaper-EN-version.pdf

[24] Luu, L., V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A Secure Sharding Protocol for Open Blockchains”, *ACM SIGSAC Conference on Computer and Communications Security*, (2016), 17–30.

[25] Meyden, R. van der, “On the specification and verification of atomic swap smart contracts”, *CoRR abs/1811.06099*, 2018.

[26] 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.

[27] OneLedger, “OneLedger Public Blockchain White Paper”, 2017. https://oneledger.io/whitepaper/oneledger-whitepaper.en.pdf

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

[29] Rootstock, “Rootstock Platform”, 2015. https://uploads.strikinglycdn.com/files/90847694-70f0-4668-ba7f-dd0c6b0b00a1/RootstockWhitePaperv9-Overview.pdf

[30] The Elrond Team, “Elrond - A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake”, 2018. https://elrond.com/files/Elrond_Whitepaper_EN.pdf

[31] Wang, D., J. Zhou, and A. Wang, “Loopring: A Decentralized Token Exchange Protocol”, 2018. https://github.com/Loopring/whitepaper/blob/master/en_whitepaper.pdf

[32] Wang, H., Y. Cen, and X. Li, “Blockchain Router: A Cross-Chain Communication Protocol”, *6th International Conference on Informatics, Environment, Energy and Applications*, (2017), 94–97.

[33] Webster, J., and R.T. Watson, “Analyzing the Past to Prepare for the Future: Writing a Literature Review”, *MIS Q. 26*(2), 2002, pp. 13–23.

[34] Wood, G., “Polkadot”, 2016. https://icowhitepapers.co/wp-content/uploads/PolkaDot-Whitepaper.pdf

[35] Yang, S., H. Wang, W. Li, W. Liu, and X. Fu, “CVEM: A Cross-chain Value Exchange Mechanism”, *2018 International Conference on Cloud Computing and Internet of Things*, (2018), 80–85.

[36] Zhang, F., E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town Crier: An Authenticated Data Feed for Smart Contracts”, *2016 ACM SIGSAC Conference on Computer and Communications Security*, (2016), 270–282.

[37] 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.