Pervasive and Mobile Computing 59 (2019) 101079

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Assessing interoperability solutions for distributed ledgers

T. Koens *, E. Poll

Radboud University, Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 14 November 2018
Received in revised form 9 August 2019
Accepted 10 August 2019
Available online 16 August 2019

Keywords:
Distributed ledger
Interoperability
Blockchain

ABSTRACT

Distributed ledgers (DLs), including blockchains, are gradually becoming an accepted technology. Almost all of these DLs are currently siloed and there is an increasing need for interaction. To fulfill this need, interoperability solutions have been proposed. However, DL interoperability has been scarcely researched and it is not always clear which properties these solutions have, nor which particular issues exist in such solutions.

This paper describes a set of key properties with which interoperability solutions can be evaluated. Using these key properties, we systematically compare the two most common solutions and identify several issues, some of which cannot be solved by the solutions themselves. Finally, these properties can be used in practice to assess and compare interoperability solutions, and can be used to determine which kind of solution fits best.

1. Introduction

There are a multitude of distributed ledger (DL) technologies, most of which are siloed. It is unlikely that there will be "one ledger to rule them all" [1]. There is, however, sometimes a need for interoperability between DLs. For example, one may wish to exchange cryptocurrencies such as bitcoins for ether. Indeed, DL interoperability is gaining more attention now that DLs, including blockchains, become an accepted technology [2]. Although an extensive amount of research is dedicated to interoperability between databases [3], only some initial research exists on DL interoperability.

DL interoperability requires further research, as the effect of an interoperability solution on a DL is currently not clear. In this work we show that DL interoperability solutions do have an effect on ledgers. These effects are to such an extent that currently *not* achieving interoperability between DLs should be considered in some cases, as interoperation could impact the integrity of the DL or make that transacting becomes economically inviable.

In Section 2 we discuss DLs in general and DL interoperability. Then we continue with our three main contributions:

- We propose a new set of 12 key properties of interoperability solutions in Section 3. With these properties we can analyze the effect of a solution on a DL. For example, when interoperating with other DLs, security and scalability of a DL may decrease, whereas cost of a transaction may increase.
  Furthermore, these properties allow us to distinguish between interoperability solutions in three ways. First we can distinguish between kinds of interoperability solutions. There are three kinds of interoperability [1]: notary schemes, relay schemes, and hash-locking schemes. Second, we can distinguish between subcategories of these solutions. And third, we can distinguish between generic and specific issues with the kinds of solutions.
- Additionally, by using these 12 properties we describe the most important differences and similarities between two real-life solutions in Section 4. These are Cosmos and Polkadot.

* Corresponding author.
E-mail addresses: tkoens@cs.ru.nl (T. Koens), E.Poll@cs.ru.nl (E. Poll).

https://doi.org/10.1016/j.pmcj.2019.101079
1574-1192/

- Furthermore, with these 12 properties we can evaluate the effect of interoperability solutions on DLs in Section 5. In particular the property 'type of update function' shows that, if interoperability is achieved between two DLs, in some cases the integrity of a DL can be compromised by attacking the DL it interoperates with. Mitigating this attack can be done by simply waiting for the attack to become hard to execute. The consequence, however, is that the time for a transaction to be finally written to the ledger is significantly increased. The possibility of this attack, including the consequences of its mitigation, is critical for deciding on whether or not DLs should interoperate, as DL integrity is of key importance.

As interoperability solutions for DLs emerge, there is no or not much literature available. This is why we also rely on technical reports for the description of interoperability solutions in our study as opposed to peer-reviewed literature. Our work, therefore, also contributes to extending the scarce literature on DL interoperability. In Section 6 we present future work and we provide our conclusions in Section 7.

2. Background

We start by providing some background on distributed ledgers and follow up with providing background on distributed ledger interoperability.

2.1. Distributed ledgers

DLs are in essence databases, with the extra property that the order of transactions matters [4]. Each participant in the network holds a copy of the database. The main challenge here is to reach consensus amongst the participants on transactions and their order. Several proposals have been made to achieve such a consensus, for example RAFT [5] or the more recent honey badger protocol [6]. However, there exists a shortcoming in such protocols, namely, a leader (or set of leaders) determines the next state of the ledger. For particular ledgers, such as Corda or Hyperledger, this is a required feature as it allows a form of control over the ledger.

In 2008, Nakamoto [7] proposed a payment system called Bitcoin that allowed any number of unknown participants to collaborate on the new state of the ledger. The technology that underpins this payment system is called blockchain. The main idea of blockchain is that anyone can participate in the consensus process by solving a hard puzzle. The participant that solves this puzzle is allowed to propose a new ledger state. All nodes in the network are informed about this new state, after which a new puzzle needs to be solved.

The attention on Bitcoin by media, corporate institutions and the general public has led to an increased attention of distributed ledgers. New type of ledgers have been developed, both in the public space (e.g. permissionless ledgers such as Ethereum, DogeCoin), as in the private space (e.g. permissioned ledgers such as Corda, Hyperledger). However, as these ledgers mature, the need for interoperability between these ledgers arises.

2.2. Distributed ledger interoperability

As for databases, interoperability between DLs focuses on systems, data, and information [3]. Additionally, interoperability in DLs entails reading, observing, and acting on *state* and *events*. Here, state is the sequence of transactions at a given point in time. An event, typically a transaction, proposes a new state of the ledger.

There are currently hundreds of ledger initiatives [8]. A potential problem is that most of these ledgers are siloed and are not able to communicate with each other. For example, suppose Alice wants to swap ownership of her bitcoins for Bob's ether, a token stored on the Ethereum ledger. Here, a token represents a digital asset of some value. A possible solution is to use a centralized exchange, which manages the token swap between Alice and Bob. However, this typically goes against the grain of decentralization of these ledgers, as there now is a trusted third party (TTP) present. This TTP introduces the risk of not moving the asset to the Ethereum ledger, which basically transfers ownership of the asset to the TTP.

Distributed ledger interoperability can appear in the following ways:

1. Interoperability between a DL and legacy systems. For example, Bitcoin and a traditional banking payment system.
2. Interoperability between DL platforms. For example, between Bitcoin and DogeCoin (two permissionless ledgers) or Corda and Ethereum (a permissioned and permissionless ledger).
3. Interoperability between two smart contracts within a single ledger. For example between Corda apps, which are smart contracts on a single Corda ledger.

In this work we focus on the second type of interoperability. To ensure that decentralized ledgers are able to communicate with each other, three kinds of interoperability solutions have been proposed [1], as discussed below.

2.3. Kinds of interoperability

In this section we describe *how* DL interoperability can be achieved. Based on the work by Buterin [1], current research on DL interoperability [9,10] distinguishes three kinds of DL interoperation:

1. **Notary schemes.** A group of parties together execute an action on ledger A when a specific event takes place on ledger B. Collectively this group of parties perform the role of a TTP, as discussed in Section 2.1. A separate ledger always exists, which is managed by this group of parties to ensure interoperability between two (or more) DLs [10].
2. **Relay schemes.** Here a smart contract on a ledger can read, validate, and act upon state or events from another ledger. A smart contract can read from another ledger because a partial copy of the other ledger is stored on the ledger where the smart contracts resides. Note that there is no need for a third party, as two ledgers communicate with each other directly. There exist one-way and two-way relays.
   (a) **One-way relays.** In a one-way relay a ledger A can read from ledger B, but ledger B cannot read from ledger A. For example, BTCRelay [1] is a smart contract on the Ethereum ledger, that reads from the Bitcoin ledger. BTCRelay is a one-way relay as Bitcoin currently does not have the same smart contract capability as Ethereum.
   (b) **Two-way relays.** Here ledger A can read from ledger B and ledger B can read from ledger A. This allows, for example, to swap tokens from ledger A to ledger B where the tokens can be used to purchase goods. The merchant could swap tokens again from ledger B to ledger A.
3. **Hash-locking.** Operations on ledger A and ledger B have the same trigger, typically the revelation of the preimage of a particular hash. In contrast to relay schemes, where a partial copy of ledger A must be stored on ledger B, hash-locking schemes require only the exchange of a single hash between the two ledgers. Therefore, hash-locking schemes require significantly less information exchange compared to relay schemes to achieve interoperability. Here, the cryptographic proof generated on ledger A triggers an event on ledger B. The main purpose is to create an atomic swap between two ledgers without a third party, as is the case with a notary scheme.

2.4. Functionalities offered

This section describes *what* functionalities interoperability solutions offer. Buterin [1] proposes the following functions offered by the kinds of interoperability:

- Token portability. A token can be sent from ledger A to ledger B. Additionally, the token can be returned (by its new owner) from B to A.
- Atomic swap. A transfer between two parties is guaranteed to happen for both parties. If one of the parties does not deliver, then the transfer will be guaranteed not to occur.
- Cross-chain oracles. A smart contract on ledger B reads from ledger A and performs an action when a particular event or state is read.
- Cross-chain asset encumbrance. Tokens are locked on ledger A and locking conditions are dependent on events on ledger B.

We summarize this in Table 2.4. Buterin [1] furthermore specifies 'general contracts' as a fifth form of functionality offered by interoperability solutions. The example provided is that of paying dividends on ledger X to shareholders whose asset ownership is registered on ledger Y. However, we consider this example as a cross-chain oracle, as the action (paying dividends on ledger X) and the dependency (assets registered on ledger Y) are similar to that of a cross-chain oracle.

Functionality	Scheme kind			
	Notary	Relay (2-way)	Relay (1-way)	Hash-lock
Asset portability	Yes	Yes	Yes	No
Atomic Swap	Yes	Yes	No	Yes
Cross-chain oracles	Yes	Yes	Yes	No
Cross-chain asset encumbrance	Yes	Yes	Yes	No

We make three observations. First, hash-locking schemes are only useful for atomic swaps. However, atomic swaps can also be accomplished by notary and relay schemes. Second, notary schemes and two-way relay schemes can provide the same functionalities. The properties in our work (see Section 3) can be used to further distinguish between these solutions. Third, one-way relay schemes and hash-locking schemes each offers the functionality that the other one does not.

Buterin [1] makes the following important observation regarding hash-locking schemes. Hash-locking schemes cannot transfer tokens from one ledger to another, hence in hash-locking schemes the total amount of tokens on each ledger remains the same. This is in contrast to notary schemes and relay schemes, where tokens *can* be transferred from one ledger to another.

3. Properties of interoperability solutions

We consider the following 12 properties of importance when assessing DL interoperability solutions:

1. Functionalities offered
2. Third party dependency
3. Reach
4. Scope
5. Scalability
6. Type of update function
7. Cost of a state change
8. Native token
9. Semantic interoperability
10. Syntactic interoperability
11. Technology development
12. Regulation

Properties 1 to 4 discriminate between the three kinds of solutions. Properties 5 to 10 are applicable to all three kinds of solutions and allow us to differentiate between the sub-categories of each kind from a technical perspective. Finally, Properties 11 and 12 are also applicable to all three kinds, but relate to external factors that may influence the interoperability between two ledgers. In the following sections we discuss these properties and where possible we make some general observations that apply to the three kinds of solutions. As we already discussed 'Functionalities offered' in Section 2, we start with 'Third party dependency'.

Third party dependency. Distributed ledgers, in particular blockchains, aim at being decentralized, meaning that there should be no dependency on a third party. Although recent work challenges the decentralization of permissionless ledgers [11], we assume that there exists no dependency on a third party in such ledgers. In notary schemes there is a strong dependency on a third party, namely, the set of notaries. In relay schemes and hash-locking schemes there exists no third party dependency.

Reach. This refers to the number (one, two, n) of DL solutions for which interoperability is achieved. This property also considers the direction in which ledgers can operate, as some solutions provide only one-way interoperability in contrast to two-way interoperability. For example, BTCrelay allows the Ethereum ledger to read from the Bitcoin ledger, but not vice versa, as the Bitcoin ledger lacks sufficient scripting capability [1].

Scope. Refers to where the state is stored [12] once interoperability has been achieved. For example, permissioned ledgers store state at a limited number of participants. The idea is that this state is not stored beyond the scope of ledger participants. Typically, this is done for privacy reasons. In contrast, in permissionless ledgers anyone can read the state from the ledger as the state is stored at all participants.

In notary schemes, state exchanged between two ledgers is stored on the two interoperating ledgers and the notary ledger. Whereas state was stored on a single ledger, it is now stored on three ledgers. In relay schemes, state is exchanged and stored only between the two interoperating ledgers. Hashlocking schemes only use a hash at first and later on its preimage is stored on both ledgers.

Scalability. In our work we consider scalability to be the number of transactions that can be processed per second. This may refer to the interoperability solution or to the ledger connecting to the solution.

Type of update function. The update function determines how the network reaches consensus on a new ledger state. This update function is also known as the consensus mechanism. There are, in principle, two types of update functions:

- **Deterministic consensus.** A group of nodes deterministically achieve consensus. Once a new ledger state has been determined, it can no longer be changed. Typically, the identity of the participants is known. These types of mechanisms typically are Byzantine Fault tolerant (BFT) mechanisms, for example HoneyBadgerBFT [6].
- **Probabilistic consensus.** A group of nodes reaches probabilistic consensus. Once a ledger state has been determined, over time, the probability of the ledger being changed becomes smaller. These types are called Nakamoto consensus [13]. A typical example is Bitcoin's proof-of-work (PoW) [7].

In particular, we are interested in atomicity and consistency, as proposed by Reuter and Härder [14]. Atomicity is important because of the need of congruent valid ledger states between two interoperating ledgers. An update function is atomic if and only if a transaction either succeeds (in updating a ledger) or fails completely. Atomicity can never be guaranteed in a single ledger that employs a probabilistic consensus mechanism, because the ledger could reach another state at any point in time. However, as time progresses, one can be reasonably sure that states no longer change; see for example the work by Rosenfeld [15]. Whereas a single ledger typically has a roll-back mechanism (either the transaction updates the ledger or it does not update the ledger), a single transaction may update ledger A, but not B after a change on ledger B. The only way, currently, to prevent this scenario from happening is to wait sufficient time on both ledgers

to be reasonably sure that no changes will take place regarding the transaction that affects both ledgers. However, it is important to notice that currently no roll-back mechanism exists for any interoperability solution if two interoperating ledgers contain a different state related to the same transaction. We will discuss this in more detail in Section 5.

Furthermore, consistency [14] is of interest. This ensures that a ledger can be brought only from one valid state to another valid state. Again, in probabilistic consensus algorithms it is possible that two valid, conflicting ledger states are temporarily present. Ultimately, this dual state will return to a single valid state. As long as this dual valid state occurs, the interoperating ledger cannot be sure which of the two states will be valid and it has to wait until the conflict is solved. Although the underlying ledger protocol, ultimately, ensures consistency, dual valid states have impact on the time in which transactions can be processed by any interoperating ledger.

Reuter and Härder [14] also propose the notions of isolation and durability. Here, isolation ensures that concurrently executed transactions leave the ledger in the same state as when the same transactions were executed sequentially. Durability ensures that once a transaction is committed, it remains committed even in the case of a node failure, such as a node crash. We do not further discuss isolation and durability as we assume that these are guaranteed by the underlying ledger protocols.

Cost. Transactions may be processed for a certain fee. These fees may differ between DLs. In fact, the difference in fees may be so high that one may choose not to use the DLs that charges high fees. For example, a micro-payment could be sent from ledger A to ledger B. However, the processing of the transaction on ledger B incurs a fee that could be higher then the actual value of tokens sent. Thus, the total cost of the transaction is higher then the actual payment. This may lead to that users of ledger A will not use the services of ledger B, because the price of sending a transaction is too high.

Native token. A native token is a token that represents some value on a particular ledger and is inherent to that ledger, e.g. bitcoin or ether. In contrast, non-native tokens are a representation of any asset that is not inherent to the ledger, for example, a token can represent a house, FIAT money, or the carbon emission a company produces yearly. Typically, in permissionless ledgers participants that propose new ledger states receive native tokens. As an example, a miner on the Ethereum network may propose a valid block. When the block is accepted by all participants, this miner receives a reward in tokens called ether. This ether represent a certain value, which in September 2018 is 230 US dollars. In contrast, a permissioned ledger such as Corda does not contain a native token. We distinguish two possibilities for interoperability solutions:

1. The *interoperability solution* may have one or more native tokens. These tokens may be used as a standard currency to exchange interconnecting ledger tokens.
2. The *interoperability solution* may not have native tokens. It may have, however, non-native tokens. This may be the case where two permissioned ledgers interoperate. Permissioned ledgers typically use a consensus mechanism that does not require a native token. As such permissioned ledgers do not have a native token. However, non-native tokens representing assets can be exchanged between these ledgers.

In addition, each of the interoperating ledgers have (or, do not have) a native token. Note that (not) having a native token does not affect technical interoperability. From a legal perspective, however, interoperability may not be desired when native tokens (such as bitcoins) are traded for non-native tokens (such as bonds), as it becomes hard for companies trading the bond to comply to Know Your Customer (KYC) regulation.

1. The *distributed ledger* has a native token. Typical examples are Bitcoin, Ethereum, and Ripple. Native ledger tokens may be used as a means of exchange between another ledger. These tokens also can be used as stake to reach consensus, for example in the consensus algorithm Proof-of-Stake [16]. The main idea here is that the stake represents a basic value that increases with time. This stake allows its owner to vote on the next ledger state. The participant with the most stake is allowed to propose the next ledger state. Once the ledger has been updated, the value of the stake of the participant whose state update was accepted is set to its basic value. For a more detailed explanation on consensus mechanisms, including proof-of-stake, we refer to the work of Bano et al. [17].
2. The *distributed ledger* has no native token. Corda [18] is an example of a ledger that does not contain native tokens. Ledgers that have no native token do not offer this functionality and are solely used to store and exchange non-native tokens.

Semantic interoperability. Semantic interoperability, also called semantic homogeneity [12], occurs when there is agreement about the meaning, interpretation or intended use of the same transaction. In contrast, semantic heterogeneity occurs when there exists no such agreement. As an example, consider a ledger B that will trigger an event when a token of 1M Satoshi (one Satoshi being the smallest unit of bitcoin currency) is locked on ledger A. Here, locking a unit means that the unit can no longer be used on ledger A, until its becomes unlocked, which is typically triggered from ledger B. Ledger A, however, locks 0.01 bitcoin, equivalent in value to 1M Satoshi. Even though these tokens represent the same value, locking the tokens on ledger A will not trigger the event on ledger B. Namely, ledger B expects an amount of 1, but the amount represented is 0.01. Here semantic heterogeneity exists due to differences in state values of related attributes.

Syntactic interoperability. This relates to the ability of ledgers to communicate with each other based on specified data formats and communication protocols.

Notary schemes ensure syntactic interoperability between two ledgers.

In relay schemes syntactic interoperability is achieved by each interoperating ledger interpreting the (copy of) the ledger it interoperates with.

In hash-locking schemes, syntactic interoperability between ledgers is limited to the way the hash is stored in the smart contracts and the type of hashing algorithm used. As the two participants claim their tokens based on revealing the preimage of the hash to that smart contract, another prerequisite here is that both ledgers verify the preimage of the hash in the same way.

Technology development. DLs are a relative new type of databases. Current research and development continues to propose changes to existing DLs. This makes that DLs change over time, from a technical point of view. In case when there exists interoperability between two (or more) DLs, changes to one of these DLs may affect the interoperability. Both DLs may have to adapt to ensure that certain interoperability still is possible. For example, assume that two DLs have an equal time of reaching consensus. A change in consensus mechanism on one DL, which considerably increases the time in which consensus is achieved, affects interoperability. Namely, in some use cases the DL that now has a faster consensus algorithm still needs to wait for the other DL to achieve consensus.

Regulation. Typically, permissioned DLs are deployed by corporate institutions and therefore have to comply with regulation and are subject to legislation. In contrast, there exists little regulation or even legislation for permissionless DLs. This contrast may be a reason for permissioned DLs not connecting to permissionless DLs.

4. Comparing Cosmos and Polkadot

In this section we discuss two interoperability solutions: Cosmos and Polkadot. We choose for a description of these schemes for four reasons. First, these schemes are the most common solutions for DL interoperability. Second, analyses of two similar schemes provides a fair comparison to determine the subtle differences between two similar solution types. Third, out of all schemes, notary schemes are the most complex schemes. Finally, there is literature available on these schemes that we can draw from.

4.1. Cosmos — A notary scheme

General description. Cosmos is a network of nodes that acts as an intermediary between DLs. The Cosmos architecture consist of a hub and spoke model. Here, a central hub is responsible for relaying messages to its spokes. Spokes are separate entities that facilitate as a bridge between an external distributed ledger (e.g. Ethereum). At its center, the Cosmos hub consists of a known set of nodes that reach consensus on the transactions between peg zones. This state is stored in a blockchain [19] (here, a permissioned ledger), meaning that interaction between any two peg zones is recorded. Cosmos also manages another blockchain used for exchanging FIAT-currencies, which can be considered a permissioned ledger.

At the edges of the hub and spoke model, peg zone validators allow the Cosmos hub to connect to a DL, such as Ethereum. The peg zone role is to approve the interaction between two DLs, namely, the DL connecting to Cosmos and the ledger residing in the Cosmos hub. An example, on the Ethereum DL ether could be locked in a smart contract, which would in turn create new tokens (called photons) in the peg zone. These photons can be transferred within the Cosmos network. Peg zones, similar to the Cosmos hub, consists of a set of nodes that reach consensus by the Tendermint protocol.

Furthermore, peg zones ensure semantic interoperability between two ledgers (i.e. two zones). For example, imagine a transaction which aims to transfer 1 BTC from the Bitcoin network to the Ethereum network. The peg-zone that connects the Bitcoin network to the Cosmos hub then:

1. Interprets the transaction.
2. Determines its intended use — i.e. the transfer of 1 BTC to Ethereum.
3. It allows its validators to understand the meaning of the transaction. Namely upon execution of the transactions, 1 BTC is locked on the Bitcoin network and the Cosmos network is requested to further deal with the transaction request.

4.1.1. Cosmos properties

In this section we use the properties proposed in Section 3 to analyze Cosmos.

Functionalities offered. The four functionalities described in Section 2.4 can indeed be provided by Cosmos, and Cosmos does not provide functionality beyond this.

Third party dependency. DLs connecting to Cosmos are dependent on their peg zone, as well as the Cosmos hub. Both the peg zone and the Cosmos hub validate transactions, on which the connected DL depends for transactions being processed. This means that there is a strong dependency on a set of third parties when two DLs become interoperable. In fact, the set of parties obtains significant control over the transaction between two DLs. When the Cosmos network fails, interoperability will no longer be possible between two ledgers. This means that Cosmos can be considered a single point of failure. As a result, tokens transferred between two DLs could be locked up indefinitely.

Reach. Cosmos allows, in principle, interoperability between any DL. However, in case of permissioned DLs, interaction with permissionless DLs is currently not preferred or even allowed by these permissioned DLs. Consider a permissioned DL

held by financial institutions. Currently, regulation discourages interaction with permissionless networks such as Bitcoin, as complying to KYC (know your customer) and AML (anti-money laundering) standards then becomes very hard. This may call for an interoperability solution that, for example, only connects permissioned DLs from certain industries.

Scope. Cosmos requires state to be stored in four places:

- The original source of the state (say, ledger A, from which tokens are transferred to ledger B).
- The receiving ledger, to which the tokens are send (ledger B).
- The Cosmos ledger, in particular the validators, who must agree on the state of the tokens exchanged.
- The peg zones.

The purpose for storing the state at all validators is that these must be able to verify the current state of the interoperating ledgers independently.

Scalability. Cosmos distinguishes between *horizontal* scalability and *vertical* scalability. Vertical scalability refers to the number of transactions a single ledger can process per second. Cosmos claims it is able to process thousands of transactions using the Tendermint protocol [19], which is used to reach consensus between the validators on the Cosmos ledger. However, at some point even a single ledger will reach its maximum number throughput. Horizontal scaling aims at using multiple parallel ledgers, which would allow for a further increase in transaction throughput on a platform, such as Cosmos.

Type of update function. Cosmos uses the Tendermint consensus protocol [20]. The security assumption made in Tendermint is that at least two-third of the voters does not act maliciously [19]. However, even when this assumption holds, it is not possible for an interoperability solution (in this case the Cosmos peg zone) to ensure transaction finality for a connecting ledger. Namely, the problem of transaction finality lies within the connecting ledger's consensus algorithm. Note that this issues applies to any interoperability solution which involves a probabilistic DL. For example, assume a transaction $tx1$ that transfers bitcoin to Ethereum. This transaction is registered in the Cosmos hub using a deterministic consensus algorithm. However, Bitcoin uses a probabilistic consensus algorithm. We assume in this example that the Bitcoin ledger forks after $tx1$ was included in the Bitcoin ledger and the new Bitcoin ledger does not contain $tx1$. This would create a difference in state between the Bitcoin ledger and immutable Cosmos hub. Namely, according to the Bitcoin ledger $tx1$ was never sent, whereas according to the Cosmos hub $tx1$ was sent.

Cost of a state change. It is unclear what exactly the cost of a state change (i.e. a single transaction) will be. However, we expect this cost will increase, as where transactions used to be processed on a single ledger, now transactions have to be processed on two (or more) different DLs. Transactions within a single DL already require a fee to be processed. Here, the transaction triggers a single state change, namely that of the DL itself. However, in case of a notary scheme, three state changes have to be initiated, 1. that of the sending DL, 2. that of the interoperability solution, and 3. that of the receiving DL. Cosmos currently does not seem to address this issue.

Cosmos has a **native token** called photons and Cosmos ensures **semantic** and **syntactic** interoperability.

Technology development. It s not clear how Cosmos adapts to changes in other DLs that are connected to the Cosmos network.

Regulation. Cosmos aims to support FIAT-tokens (i.e. a currency with intrinsic value, such as the dollar or euro). However, most current regulations do not allow FIAT-tokens to be exchanged on permissionless ledgers. Therefore, Cosmos introduces a second, permissioned ledger. This permissioned ledger is managed by a private authority [19], which can be considered a central hub. There may be legislative issues in this scenario. The euro, for example, is used in multiple countries, each having different legislation. This would imply that a single permissioned ledger for each European country should be present. Furthermore, regulating instances (such as the US Federal reserve and the European Central Bank) likely will want to have full insight in the transactions of each of these permissioned ledgers. Given these constraints, a blockchain here seems not to be needed [4].

4.2. Polkadot — A notary scheme

General description. Polkadot can be considered a set of third parties on which DLs are dependent when they wish to achieve interoperability, similar to Cosmos. Polkadot uses a blockchain called 'relay chain' to register transactions between any DL, similar to the Cosmos blockchain. The relay chain is envisioned [21] to be a permissionless ledger, although Polkadot's initial version contains a permissioned ledger. In Polkadot, DLs are referred to as parachains. A parachain bridge allows for a connection to a DL (e.g. Ethereum Mainnet), which is similar to a Cosmos peg zone. Wood [21] considers the network protocol, i.e. how the parachain bridges communicate with the relay chain, an important feature of Polkadot but currently out of scope of Polkadots documentation.

The Polkadot network consists of four roles: validators, nominators, collators and fishermen [21]. Validators together aim at reach consensus (similar to Cosmos validators) on the new state of the relay chain. Validators run a full node and verify every transaction. Nominators do not run a full node, but may vote on blocks or may transfer their 'dots' to a trustworthy validator (similar to delegators in Cosmos). A collator stores the full database of a parachain and handles the task of storing the parachain and produce unsealed blocks. A collator provides these blocks to a group of validators. Collators thus assist validators in producing blocks, by taking some of the workload such as validity and verification.

Fishermen can be considered as independent bounty hunters. They send proof of illegal activity created by collators or validators. Once such proof has been submitted, the responsible collator or validator is penalized (i.e. slashed, punished for their behavior by taking away some tokens).

4.3. A comparison of Polkadot and Cosmos

Instead of listing all Polkadot properties, we group the 12 properties (highlighted in italics) by discussing the differences and similarities between Cosmos and Polkadot.

Similarities. There are several similarities between Cosmos and Polkadot. They offer the same *functionalities*, there exists a *third party dependency*, and the *reach* is aimed at any ledger. State is stored (*scope*) in four places in both solutions; in the ledger sending the transaction, the bridge, the central ledger, and the receiving ledger. Polkadot also uses a native token, called 'dots'. Validators ensure that there exists semantic and syntactic interoperability; thus, similar to Cosmos, the interoperability solution ensures semantic and syntactic interoperability. Furthermore, also similar to Cosmos, it is not clear how Polkadot will adapt to changes to that of connecting ledgers (*technology development*).

Differences. In contrast, Polkadot divides *scalability* in the number of transactions that can be processed and the total amount of resources that are needed to process a single transaction. Also different is its *type of update function*. Whereas Cosmos uses Tendermint [20], in Polkadot this is based on Tendermint [20] and HoneyBadgerBFT [6]. It is not clear how exactly this new algorithm operates in Polkadot. The *cost of a state change* are discussed in both Cosmos and Polkadot. However, the difference here is that Cosmos discusses fees per transaction, whereas Polkadot addresses the issue of the transaction cost increase. Finally, Polkadot currently focuses on connecting permissionless ledgers which are currently not subject to regulation. This is in contrast with Cosmos (which supports FIAT-tokens), as Polkadot considers connecting to permissioned ledgers open for discussion [21].

5. Discussion

In the previous section we used the key properties to systematically compare two specific DL interoperability solutions. In this section we draw general conclusions on notary schemes using four of the key properties from Section 3, which are: third party dependency, scope, type of update function, and cost of a state change. The remaining eight properties have shown that there are no further significant differences between Cosmos and Polkadot.

Third party dependency. A notary scheme is a third party that creates a dependency between two interoperating ledgers. This goes against the grain of decentralized ledgers.

Scope. The scope of storing data is extended ad transactions are stored in 3 networks: in the network sending the transaction, in the network of the notary scheme, and in the transaction receiving network. This may affect privacy, in particular for private ledgers.

Type of update function. Networks such as Bitcoin and Ethereum currently use probabilistic consensus mechanisms, which allows for the reversal of transactions [4]. In contrast, the Cosmos hub uses a deterministic consensus mechanism. Although peg zones may wait some time to be reasonably sure that the interconnecting DL will not reverse a transaction, peg zones can never be sure. In fact, it may happen that a DL reverses a transaction which was validated in the immutable ledger of Cosmos. This would lead to an inconsistency of state between the two interconnecting ledgers. Also, it leads to an inconsistency between the Cosmos ledger and the interconnecting ledger on which transactions are reversed.

We observe that Polkadot and Cosmos differ in architecture such as the number of roles and in technological choices such as type of update function. We consider the property 'type of update function'to be an important difference between Polkadot and Cosmos. Whereas the Cosmos peg zones are responsible for verifying blocks of their respective zones, in Polkadot the validators validate all blocks of all ledgers. In Cosmos there is a dependency on the peg zones by the Cosmos hub. Since the security assumption is that the majority of the nodes of each peg zone is honest, the Cosmos hub must make this assumption for every peg zone which weakens its security. In contrast, only the majority of the Polkadot validators must be honest. However, this introduces a single point of failure for all DLs connected.

Cost of a state change. Notary schemes seem to increase the cost of transactions because for each ledger update costs are charged. Typically, in notary schemes three ledgers have to be updated, namely two interconnecting ledgers and the ledger of the interoperability solution. Wood [21] proposes to bundle transactions that can be processed as a set, which would reduce the cost of a transaction. There exist, however, almost no research on transaction bundling in interoperable DLs to reduce costs. Many topics are open for further study, such as the optimal bundling of transactions or how different DLs (e.g. Bitcoin and Ethereum) should bundle such transactions.

6. Future work

In Section 4 we assessed two notary schemes by using 12 key properties of interoperability solutions. Similar notary schemes can also be assessed and compared to our work. Also, other types of interoperability solutions should be assessed. First by identifying specific differences between similar solutions and second by identifying general issues in interoperability solutions.

Current interoperability solutions focus on connecting either two permissionless ledgers or as many (both permissionless and permissioned) ledgers as possible to create a network of interconnected ledgers. However, permissioned ledgers may not wish to share their state in a permissionless ledger. Therefore, further research on interoperability between permissionless and permissioned ledgers is needed to determine how these ledgers can connect and maintain their privacy properties.

With our set of properties interoperability solutions in general can be assessed. Some of these properties, however, can be extended for specific scenarios. For example, the property scalability could include the storage requirements for the size of the ledger, which may be of importance in an Internet-of-Things scenario where devices typically have limited storage capacity.

The goal of most interoperability solutions currently focuses on *technically* connecting two or more ledgers. However, another approach to this is to determine stakeholder requirements for connecting to another ledger. This may reveal new, non-technical properties (such as maturity of the solution), which allows for building new interoperability solutions.

Finally, we identified multiple issues in notary schemes, such as the introduction of a third party, state finality between probabilistic and deterministic ledgers, and state distribution between permissioned and permissionless ledgers. Proposing solutions for these issues may be part of future work on DL interoperability.

7. Conclusion

We proposed an approach to further distinguish DL interoperability solutions and their sub-categories, based on the key properties of these solutions. We have shown that these properties, introduced in Section 3, are useful for describing, analyzing, and evaluating DL interoperability solutions. Our evaluation based on these properties, shows that the update function is the main difference between the notary schemes Cosmos and Polkadot, as discussed in Section 5. The update function of both notary schemes weaken security. Cosmos makes the assumption that the majority of participants in multiple, different sets are honest, whereas Polkadot introduces a single point of failure.

We also identified several critical issues, some of which are inherent to distributed ledgers. In particular, the property *type of update function* showed that it is possible to breach the integrity of a ledger by attacking the ledger it interoperates with, as discussed in Section 5. A successful attack would lead to an invalid state between immutable ledgers. This attack is applicable to all interoperability solutions that include a DL with a probabilistic consensus algorithm. In fact, we consider such an attack, including the consequences of its mitigation measure, to be critical for deciding on whether or not ledgers should interoperate. The current solution to this attack, i.e. wait for a sufficient time (e.g. one hour), significantly increases transaction finality between ledgers.

We conclude furthermore that the topic of distributed ledger interoperability is almost a greenfield area of study. There is a lack of literature and most projects on interoperability have shown only early proof-of-concepts.

Finally, we conclude that many interoperability topics require further attention, such as ensuring finality when interoperability exists between a probabilistic ledger, interoperability between permissionless ledgers and permissioned ledgers, interoperability solely for permissioned ledgers, and increasing transaction costs. This is, of course, an opportunity to study and develop distributed ledger interoperability further.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] V. Buterin, Chain interoperability, in: R3 Research Paper, 2016, https://static1.squarespace.com/static/55f73743e4b051cfcc0b02cf/t/5886800ecd0f68de303349b1/1485209617040/Chain+Interoperability.pdf.
[2] T. Swanson, Consensus-as-a-Service: A Brief Report on the Emergence of Permissioned, Distributed Ledger Systems, Report, 2015, Available online, Apr. https://allquantor.at/blockchainbib/pdf/swanson2015consensus.pdf.
[3] A.P. Sheth, Changing focus on interoperability in information systems: From system, syntax, structure to semantics, in: Interoperating Geographic Information Systems, Springer, 1999, pp. 5–29.
[4] T. Koens, E. Poll, What blockchain alternative do you need? in: Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer, 2018, pp. 113–129.
[5] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: 2014 USENIX Annual Technical Conference, ATC 14, 2014, pp. 305–319.
[6] A. Miller, Y. Xia, K. Croman, E. Shi, D. Song, The honey badger of BFT protocols, CCS'16, ACM, 2016, pp. 31–42.
[7] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system, 2008, https://bitcoin.org/bitcoin.pdf.
[8] R. Beck, C. Müller-Bloch, Blockchain as radical innovation: A framework for engaging with distributed ledgers as incumbent organization, in: Proceedings of the 50th HICSS (2017), Hawaii, USA, 2017.
[9] H. Jin, X. Dai, J. Xiao, Towards a novel architecture for enabling interoperability amongst multiple blockchains, in: 38th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2018, pp. 1203–1211.
[10] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, W. Knottenbelt, Xclaim: Trustless, interoperable, cryptocurrency-backed assets, in: S&P, IEEE, 2019.
[11] A.E. Gencer, S. Basu, I. Eyal, R. van Renesse, E.G. Sirer, Decentralization in Bitcoin and Ethereum networks, 2018, https://arxiv.org/pdf/1801.03998.

[12] A.P. Sheth, J.A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Comput. Surv. 22 (3) (1990) 183–236.
[13] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, E.W. Felten, Sok: Research perspectives and challenges for bitcoin and cryptocurrencies, in: Symposium on Security and Privacy, SP, IEEE, 2015, pp. 104–121.
[14] T. Haerder, A. Reuter, Principles of transaction-oriented database recovery, ACM Comput. Surv. 15 (4) (1983) 287–317.
[15] M. Rosenfeld, Analysis of hashrate-based double spending, 2014, https://arxiv.org/pdf/1402.2009.
[16] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, 2012, Self-published paper, August 19.
[17] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, G. Danezis, Consensus in the age of blockchains, 2017, https://arxiv.org/pdf/1711.03936.
[18] R.G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: An introduction, 2016, https://docs.corda.net/releases/release-M7.0/_static/corda-introductory-whitepaper.pdf.
[19] J. Kwon, E. Buchman, Cosmos whitepaper, 2018, https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md#horizontal-scaling.
[20] J. Kwon, Tendermint: Consensus without mining, 2014, https://cdn.relayto.com/media/files/LPgoWO18TCeMIggJVakt_tendermint.pdf.
[21] G. Wood, Polkadot: Vision for a heterogeneous multi-chain framework, 2016, https://icowhitepapers.co/wp-content/uploads/PolkaDot-Whitepaper.pdf.