Assessing Interoperability Solutions for Distributed Ledgers
Extended version

T. Koens[a], E.Poll[a]

[a]*Radboud University, Nijmegen, The Netherlands*

Abstract

Distributed ledgers (DLs), including blockchains, are gradually becoming an accepted technology. Almost all of these ledgers are currently siloed, and there is an increasing need for interaction between such ledgers. 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. We evaluate five solutions and identify several issues, of which some cannot be solved by the solution itself. Our approach can be used in practice to assess interoperability solutions, and can be used to determine which kind of solution fits best.

Keywords: Distributed ledger, Interoperability, Blockchain

1. Introduction

There exist 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].

Email addresses: tkoens@cs.ru.nl (T. Koens), E.Poll@cs.ru.nl (E.Poll)

There is, however, 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. In this work we aim to spark further DL interoperability research. In Section 2 we discuss DL interoperability. We provide the following contributions:

- We propose 12 key properties of interoperability solutions in Section 3. 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 of the kinds of solutions.

- Additionally, by using these properties, we describe and analyze five real-life solutions in Section 4. These are Polkadot, Cosmos, BTCRelay, Dogethereum, and hash-locking schemes in general.

- We discuss in detail the zone-spend attack in Section 5.1 which applies to all three kinds of solutions that interoperate with permissionless ledgers. The possibility of this attack, including the consequences of its mitigation, is critical for deciding on whether or not ledgers should interoperate.

- Furthermore, we evaluate these five solutions in Section 5, and we discuss interoperability issues of permissioned ledgers. We conclude that although

these three kind of solutions offer a different set of functionalities, there exists an overlap between issues of these solutions. This is useful for deciding on which interoperability to choose, and becoming aware of the issues that come with each solution.

In Section 6 we present future work, and we provide our conclusions in Section 7.

2. Background

DLs are in essence a database, with the exception that the order of transactions matters [4]. Similarly to databases, interoperability between DLs focuses on systems, data, and information [3]. Additionally, interoperability in DLs focuses on reading, observing, and acting on *state* and *events*. Here, state is the order of transactions at a given point in time. An event, typically a transaction, creates a new state of the ledger.

There are currently hundreds of permissionless ledger initiatives [5] and hundreds of permissioned ledger initiatives [6]. The main problem is that most of these ledgers are siloed and are not able to communicate with each other. For example, 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 centralize 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].

2.1. Kinds of Interoperability

In this section we describe *how* DL interoperability can be achieved. Based on the work by Buterin [1], current research [7, 8, 9] on DL interoperability distinguishes three kinds of DL interoperation. These kinds offer different functionalities, which we will discuss here:

1. **Notary schemes.** A group of parties together execute an action on ledger A when a specific event takes place on ledger B. A separate ledger always exists, which is managed by this group of parties to ensure interoperability between two (or more) DLs [9].

2. **Relay schemes.** Typically, 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 interface, 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 [10] 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 provide similar capabilities 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 token 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 significant 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.

2.2. 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 2.2. 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.

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

	Scheme kind			
Functionality	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

to further distinguish between these solutions. Third, one-way relay schemes offer the inverse functionalities of hash-locking schemes.

Buterin [1] makes the following important observation regarding hash-locking schemes. In hash-locking schemes the total amount of tokens on each ledger remains the same, hence hash-locking schemes cannot transfer tokens from one ledger to another. 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. Update function
7. Cost of a state change
8. Native token
9. Semantic interoperability
10. Syntactic interoperability
11. Technology development
12. Regulation

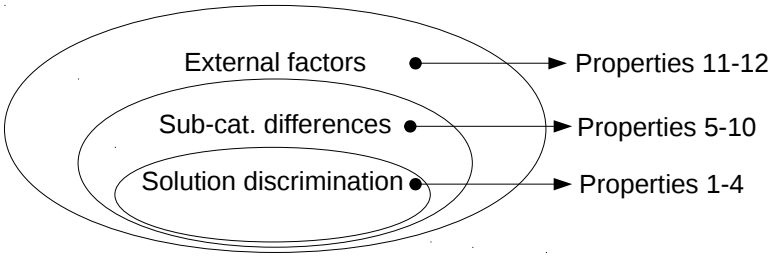

Figure 1: Ordering of the properties

We propose this order as shown in Figure 3, as properties 1 to 4 discriminate between the three kinds of solutions. Properties 5 to 10 are applicable to all three kinds, 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'.

3.1. Third party dependency

Distributed ledgers, and in particular blockchains, aim to be 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. For each interoperability kind we can determine third party dependency.

- Notary schemes. There exists a strong dependency on a third party, namely, the set of notaries.

- Relay schemes. Here there only exists the dependency between the ledgers that interoperate.
- Hash-locking. Here there only exists the dependency between participants of the ledgers that interoperate.

Thus, in relay and hash-locking schemes there exists no third party dependency.

3.2. Reach

Refers to the number (one, two, *n*) of DL solutions to which interoperability is possible. This property includes the direction in which ledgers can operate, as some ledgers 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].

3.3. 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 on a single ledger was stored, 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 pre-image is stored on both ledgers.

3.4. 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.

3.5. Update function

An update function, also known as a consensus mechanism, determines how the network on which the ledger resides reaches consensus. 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 Practical BFT (PBFT) [13].

- **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 [14]. A typical example is Bitcoin's proof-of-work (PoW) [15].

In particular, we are interested in atomicity and consistency, as proposed by Reuter and Härder [16]. Atomicity is important because of the need of congruent valid ledger states between two interoperating ledgers. This property (of an update function) states that 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 Rosenfeld [17]. Whereas a single ledger typically has a fall-back mechanism (either the transaction updates the ledgers, 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 regarding the transaction that affects both ledgers. However, it is important to notice that currently no fall-back mechanism exists if two interoperating ledgers contain a different state related to the same transaction.

Furthermore, consistency [16] 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 the interoperating ledger.

Reuter and Härder [16] also propose isolation and durability. Here, isolation ensures that concurrent 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 protocol.

3.6. 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.

3.7. Native token

A native token is a token that represents some value on a particular ledger, and is inherent to that ledger. 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. Note that a ledger entry is made once a token is owned by a participant of that particular event. 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 between four possibilities:

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.

3. The *interoperating 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 [18]. 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. [19].

4. The *interoperating ledger* has no native token. Corda [20] 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 echange non-native tokens.

Note that the functionalities offered by an interoperability solution are not affected whether or not there are native tokens in either the interoperability solution or the interconnecting ledger.

3.8. Semantic interoperability

Semantic interoperability, also called semantic homogeneity [12], occurs when there is agreement about the meaning, interpretation or intended use of the same state. 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.

3.9. Syntactic interoperability

This relates to the ability of ledgers to communicate with each other based on specified data formats and communication protocols. Notary schemes and relay schemes ensure syntactic interoperability between two interoperating ledgers.

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 pre-image of the hash to that smart contract, another prerequisite here is that both ledgers verify the pre-image of the hash in the same way.

3.10. 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 in general change, from a technical point of view, over time. In case when there exists interoperability between two (or more) DLs, changes to one 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 average time of reaching consensus. A change in consensus mechanism on one DLs, which considerably increases the time in which consensus is achieved, affects interoperability. Namely, in some use cases the upgraded DLs still needs to wait for the other DLs to achieve consensus.

3.11. Regulation

Typically, permissioned DLs are deployed by corporate institutions, and therefore have to comply to 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. Use cases

In this section we discuss five interoperability solutions: Cosmos, Polkadot, BTCRelay, Dogethereum, and hashlocking schemes in general. We extensively describe Cosmos and BTCRelay, and compare these with the similar kind of solutions Polkadot and Dogethereum. We do this because there are clear similarities between the kinds of solutions, and some subtle differences. We summarize our findings in Table 1.

Table 1: Interoperability solutions and there specific properties

	Interoperability solution				
Property	**Cosmos**	**Polkadot**	**BTCRelay**	**Dogethereum**	**Hashlocking (general)**
Solution kind	Notary (3 main roles)	Notary (4 main roles)	Relay (one-way)	Relay (two-way)	Hashlocking
TP dependency	Yes	Yes	No	No	No
Reach	n (any ledger)	n (Currently only permissionless)	1	2	n
Scope	Entire ledger	Entire ledger	Headers only	Headers only	Hashes only
Scalability	low (tx/s only)	low (also considers amount of resources)	medium	medium	medium
Update function	Tendermint	BFT (exact algorithm unknown)	native	native	native
Cost state change	Increase	Increase (Transaction sets may decrease costs)	Increase. (Includes fees for relayers)	Increase. (Includes fees for submitters)	Remains the same
Native token	yes (photons)	yes (dots)	no	no	no
Semantic interoper.	Provided by solution	Provided by solution	Provided by single ledger	Provided by both ledgers	Provided by both ledgers
Syntactic interoper.	Provided by solution	Provided by solution	Provided by single ledger	Provided by both ledgers	Provided by both ledgers
Techn. developm.	unknown	unknown	New contract	New contract	New contract
Regulation	Not established yet. Uses also a permissioned ledger for FIAT-tokens	Not established yet. Focus is on permissionless ledgers	N/A	N/A	N/A

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 entity 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). To enable communication between the hub and its spokes, the Inter Blockchain Communication (IBC) communication protocol is used. 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 [21] (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. We discuss this ledger together with the property governance, and discuss permissioned ledgers in more detail in Section 5.5

At the edges of the hub and spoke model, peg zones validators allow the Cosmos hub to connect to a DLs, such as the Ethereum Mainnet. The peg zone role is to approve the interaction between two DLs; 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 Cosmus 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, as described in Section 2.2 can indeed be provided by Cosmos. Furthermore, Cosmos does not extend beyond these functionalities offered.

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 flow 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 other pubic 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 there 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 examples, 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 [22], 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. However, maximum transaction

throughput is always determined by the slowest ledger.

Update function. Cosmos uses the Tendermint consensus protocol [23]. According to the Cosmos white paper [21], forks cannot appear using this protocol, and the ledger created is immutable. As a solution for the zone fork (i.e. how to ensure transaction finality in probabilistic consensus algorithms, as discussed in Section 5.1), Cosmos states that "Peg zones are our solution" [24]. However, it is not possible for an interoperability solution (in this case the Cosmos peg zone) to ensure transaction finality for a connecting ledger, as the problem of the zone fork 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 that a transaction $tx1$ transfer of bitcoin to Ethereum takes place. This transaction is registered in the Cosmos hub using a deterministic consensus algorithm. However, Bitcoin uses a probabilistic consensus algorithm. As an example, we assume 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 send, whereas according to the Cosmos hub $tx1$ was send.

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 (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 [22], 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 euro-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. Polkadot uses a blockchain called 'relay chain' to register transactions between any DL. The relay chain is envisioned [25] to be a public permissionless ledger, although Polkadot's initial version contains a permissioned ledger. In Polkadot, DLs are referred to as parachains. The Polkadot bridge is therefore called parachain bridge.

A parachain bridge allows for a connection to a DL (e.g. Ethereum Mainnet), which is similar to a Cosmos peg zone. Wood [25] 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 [25]. Validators together aim to 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 will send proofs of illegal activity created by collators or validators. Once proof has been submitted of such illegal activity, the responsible collator or validator is penalized (i.e. slashed, punished for their behavior by taking away some tokens).

Similarities between Polkadot and Cosmos. 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 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.

Differences between Polkadot and Cosmos 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 update function. Whereas Cosmos uses Tendermint [23], in Polkadot this is based on Tendermint [23] and HoneyBadgerBFT [26]. It is not clear how exactly this new algorithm operates in Polkadot. 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 [25].

4.3. BTCRelay - A relay scheme

BTCRelay [10] in essence is a smart contract that resides on the Ethereum ledger. Its main purpose is to verify the validity of Bitcoin transactions. Additionally, these transactions can be relayed to any Ethereum smart contract. For example, BTCRelay can verify that a Bitcoin transaction is valid, and send it to a contract that sends a product to a participant, knowing that the amount of bitcoin has been paid. By storing Bitcoin block headers, BTCRelay builds a significant smaller version of the Bitcoin ledger. The block headers are used to verify that a transaction (to the BTCRelay smart contract) is legitimate, this is a similar technique as used in Bitcoin light clients (Simplified Payment Verification [27]). Relayers are participants that can read from the Bitcoin ledger and write to the BTCRelay contract. Relayers are incentived or submitting block headers by receiving fees from those that wish to have their transaction verified for that particular block. They constantly submit Bitcoin block headers to the BTCRelay smart contract. Note that

this verification only goes from Bitcoin to Ethereum, and not vice versa. This is because Bitcoin currently lacks the capabilities of implementing such smart contracts. In case of such a transfer, a transaction on the Bitcoin ledger sends its coins to an unrecoverable address, meaning that the tokens no longer can be used on the Bitcoin ledger.

4.3.1. BTCRelay properties

In this section we discuss BTCRelay and the 12 properties as proposed in Section 3.

Functionalities offered. BTCRelay offers three functionalities as stated in Table 2.2, being asset portability, cross-chain oracle, and cross-chain asset encumbrance. BTCRelay is a one-way relay, meaning that it cannot perform an atomic swap between two ledgers.

Third party dependency. In principle, relay schemes allow for interoperability without a third party. Although there exists a dependency on the relayers providing block headers to the Ethereum ledger, anyone can become a relayer. This is in contrast, for example, to the validator role in notary schemes, where a fixed set of participants are a validator.

Reach. The reach of BTCRelay is limited to two ledgers, namely Bitcoin and Ethereum. Furthermore, BTCRelay can only transfer tokens from the Bitcoin to the Ethereum ledger, as Bitcoin currently does not support this functionality.

Scope. BTCRelay stores Bitcoin blockheaders on the Ethereum ledger. This extends the scope of the Bitcoin ledger. However, since both ledgers are public the impact of extending its scope is minimal.

Scalability. Once a transaction has been included in a block on the Bitcoin

ledger, participants should wait for some time to be reasonably sure that the transaction is final on the Bitcoin ledger. Similar, once the transaction has been verified by BTCRelay and is relayed to another smart contract, again the participants have to wait some time to be reasonably sure that the transaction is final on the Ethereum ledger. This means that scalability is reduced to the time from which the transaction is send on the Bitcoin ledger, until it becomes final on the Ethereum ledger.

Update function. BTCRelay in itself does not introduce a new update function. Instead, it uses both the update functions of the Bitcoin ledger as well as the Ethereum ledger, which currently are both proof-of-work.

Cost of a state change. The total costs of a state change consist of three values. First there is the cost for the Bitcoin transaction, locking bitcoins on the Bitcoin ledger. Second, there are the fee costs for the relayer on the Ethereum ledger, for verifying the transaction. Third, there is the cost of an Ethereum transaction, where the verified transaction is, for example, being forwarded to another smart contract.

Native token. BTCRelay does not introduce any new native tokens. It simply uses the native tokens of the Ethereum ledger, called ether.

As BTCRelay is a smart contract residing on the Ethereum ledger, Ethereum ensures **syntactic** and **semantic** interoperability.

Technology development. In principle, Ethereum smart contracts cannot be altered [28]. In case of a technological development which causes BTCRelay to no longer function, a new smart contract could be deployed with the same functionalities as BTCRelay currently offers.

Regulation. Regulation on cryptocurrencies has not yet been firmly established. BTCRelay (and also Dogethereum, which we discuss in the next section) supports exchange of cryptocurrencies between ledgers. Therefore, we consider

that there is limited regulation for any interoperability solution which is used on a permissionless ledger.

4.4. Dogethereum - A relay scheme

General description. Dogecoin is a DL that uses a native token called Doge. Dogethereum is an interoperability solution that allows for moving Doges to the Ethereum ledger, and back.

From Dogecoin to Ethereum. To manage this process, Dogethereum introduces three roles, operators, submitters, and general participants. An operator resides on the Dogecoin network and serves as an intermediary for transferring tokens between the two ledgers. To be able to transfer Doge, a participant first locks the desired amount of Doge. This is achieved by creating a lock Doge transaction which sends the tokens to an address owned by the operator. This transaction then will be included in a block. A submitter creates a superblock by building a Merkle tree of several Dogecoin blocks containing transactions.

Once the participant's transaction is included in a superblock an Ethereum transaction is send to the DogeSuperblocks contract, which resides on the Ethereum ledger. This contract verifies the transaction and, if valid, sends the transaction to a DogeToken smart contract. If the transaction is valid according to the DogeToken contract, the contract sends the specified amount of ether to the user on the Ethereum ledger. Here, the ether will be assigned to the Ethereum address controlled by the private key that signed the initial dogecoin transaction.

From Ethereum to Dogecoin. A participant sends a transaction (which includes a unique identifier of an operator) to the DogeToken contract on the Ethereum ledger. If the transaction is valid, the operator will notice the unlock request. The

operator then creates an unlock transaction on the DogeCoin network, which unlocks the doge tokens. At this point the participant is again able to use the doge tokens on the DogeCoin ledger.

Similarities between Dogetehereum and BTCRelay. In Dogecoin there also exists no third party dependency, as anyone can propose superblocks. The reach of Dogetehreum is also limited to two ledgers. Also, a fee is required for each transaction made through Dogethereum [29]. Similar to BTCRelay, this increases the total transactions costs per transaction. Dogetehreum does not introduce a new native token. Instead, it relies on the already existing tokens of the Dogecoin ledger (doge) and Ethereum (ether). Similar to BTCRelay, Dogethereum ensures that semantic and syntactic interoperability exists between the two ledgers.

Differences between Dogetehereum and BTCRelay. The main difference is that Dogethereum allows for a two-way swap, whereas BTCRelay allows for a one-way swap only. This increases the functionality of Dogetehreum, as it allows for atomic swaps, see Table 2.2. Both relay schemes are dependent on the update function of the ledger they operate on.

Instead of storing blockcheaders on the Ethereum ledger as in done in BTCRelay, Dogetehreum solution uses superblocks [30]. A superblock is a collection of all Dogecoin block headers collected in the last hour, merged into a Merkle tree. This introduces a delay of an extra hour at most for transactions to be finalized. The benefit of proposing superblocks is that the costs of verifying blocks decreases. In contrast to BTCRelay, the scope of Dogetehreum is extended to both ledgers, as swaps can be made to both ledgers.

4.5. Hash-locking scheme

Hash-locking schemes, also called atomic swaps, are based on two users separately locking their tokens. These can be unlocked by the other user upon revealing the pre-image of a hash. On the Bitcoin forum [31] an atomic swap is proposed by a pseudonyms user called TierNolan, who often is associated with the hash-locking scheme [1]. Although this scheme is well-known in the distributed ledger community [32, 33], Herlihy [34] provides a first extensive analysis of the scheme, and Borkowski et al. [35] provide a survey on atomic swaps for distributed ledgers.

A variety of applications exist regarding hash-locking schemes, such as hash-locking and atomic swaps, hash-locking with payment channels, or hash-locking using a notary (or relay) scheme. Furthermore, Herlihy states that besides an exchange of tokens, atomic swaps can be used for sharding of ledgers and decentralized system upgrades. Here we focus on cross-ledger token swaps.

Variants of the atomic swap are possible (e.g. an atomic swap between three or more parties), but in principle are all based on the mechanics described next. To achieve an atomic swap, participant A creates a transaction which is locked (for time t) with the public key of participant B and the hash of a secret value. Note that, after time t, the tokens are refunded to A if the tokens are not claimed by B. Then, participant A shares the hash of a secret value with participant B. B then submits a transaction (sending tokens to A) which is locked (for time $t + n$) with A's public key and the secret value hash. A then claims the tokens send by B, by revealing the secret value. B then also knows the secret value, and can claim the tokens send by A. Here, the assumption is made that both t and $t + n$ are large enough for both parties to claim their tokens.

There exist many proposals to perform an atomic swap, for example [36, 37,

38, 34]. In principle, these hash-locking solutions are similar in how they work. A noticeable difference between these proposals is the number of steps that are used to describe an atomic swap, and the number of verifications done by each user. For example, some atomic swap schemes suggest that each user verifies the smart contract of its counter party for correctness [39].

A specific requirement for atomic swaps is that an out-of-band channel exists [36] to share the hash of the secret value between the two participants. However, the pre-image of the hash could be of value, and the sender may wish not to disclose the pre-image. Bowe and Hopwood [40] propose a zero-knowledge scheme, where one can proof to know the pre-image without revealing it. The receiving party then can use this proof to redeem the tokens send by the sender.

5. Discussion

In this section we discuss the issues and differences between the three kinds of interoperability solutions. We start with a discussion on the zone-spend attack in Section 5.1, as this attack applies to all schemes and may be critical in deciding whether or not interoperability should happen between two ledgers. In Section 5.2 we discuss the issues in notary schemes, in Section 5.3 the issues in relay schemes, and in Section 5.4 the issues in hash-locking schemes. We summarize our findings in Table 5.1. Additionally, we discuss in Section 5.5 interoperability issues of permissioned ledgers as these type of ledgers offer some unique challenges.

5.1. Zone-spend attack.

All three interoperability schemes allow for a new type of attack, which we call the zone-spend attack. In principle this is a double-spending attack made possible

due to interoperability between two ledgers. The main idea is as follows. Probabilistic consensus algorithms cannot guarantee finality of state. Therefore, when two ledgers A and B interoperate, ledger A may alter its history after interaction with ledger B. This new history may not include any interaction with the ledger B. However, ledger B then contains a state in which it interacted with ledger A. This causes an inconsistency of state between the two ledgers. Although the possibility of this attack has been raised in general as a concern [1, 25], we provide the first detailed description of this attack and determine the cost of this attack.

Imagine a participant K, who owns an account a with a value of 1 on DL A. Furthermore, there exists a DL B, on which an account b resides which holds no tokens, and is also owned by participant K. Note that the cumulative balance of a and b is 1. Now, K sends from $A\{a\}$ 1 token to $B\{b\}$ by transaction $tx1$. Once the tokens are transferred through the interoperability solution to $B\{a\}$, K ensures that the transaction is reversed by creating a longer chain on DL A which does not include $tx1$. This will set the balance of $A\{a\}$ to 1. As the balance of $B\{b\}$ is also 1, the cumulative balance of a and b is now 2.

Creating a longer chain in a probabilistic consensus mechanism may require a large amount of computational power. The number of tokens transferred, however, is not limited. This means that an attacker can easily calculate how many tokens (t) must be transferred to profit (p) from the attack (a):

$$p = t - a \tag{1}$$

To illustrate this in an example, assume the following:

- The bitcoin price is 6700 dollar (September 2018) [41].

- The total computational power of the Bitcoin network is 500 MegaWatt (MW) [42], which is an upper bound.
- To create a longer chain, an attacker needs 51% of the total network power, which is 255 MW [15].
- The physical hardware to deliver this capacity is located in China, as is currently the case for Bitcoin, where the price for electricity is 6.4 cents per kWh[43].

This leads to the price for an attack of a =(255.000*6.4)/6700 $\approx$ 244, meaning that a transfer of t =245 Bitcoin allows for a profit p when performing the attack. This is not an unlikely scenario, as in 2018 larger sums (e.g. 413 bitcoin [44]) of bitcoin have been transferred in the Bitcoin network.

Possible countermeasure to the zone-spend attack. Cosmos proposes to wait for 100 blocks [24], wheras Polkadot proposes to wait 120 blocks (both when connected to Ethereum) [44]. Indeed, the probability of the Ethereum ledger forking after 120 blocks is very small. However, this takes 120*14 seconds = 1800 seconds (30 minutes) before 'finality' is achieved on the Ethereum ledger. Furthermore, if there is an exchange between Ethereum and a similar platform, the 'finality' is increased to 2*1800 = 60 minutes. The downside of this solution is that it significantly throttles the transaction finality between two ledgers.

5.2. Notary schemes

We observe that Polkadot and Cosmos differ in architecture such as the number of roles, and in technological choices such as update function. We expect that this

[1]Only applies when permissionless ledgers are involved

Table 2: Interoperability issues per kind of solution

Issue	**Notary**	**Relay**	**Hashlock**	**Prop.**
Out-of-band channel required	-	-	x	1
TP dependency	x	-	-	2
Single point of failure	x	-	-	
Limited reach	-	Only in one-way relay schemes	-	3
Increased scope	In particular when permissioned ledger is involved			4
Decrease tx scalability	Always depends on the slowest ledger			5
Zone-spend [1]	x	x	x	6
Financial DoS	Only icm hashlock		x	
Decrease tx finality	x	x	x	
Transaction cost increase	Depends how tx are managed	Yes, tx cost for both ledgers		7
Reaching consensus	Limited possibilities when no native token is present			8
Cartel forming	-	Yes	-	9
Early stages of development	x	x	x	11
Compliance to regulation	In particular when permissioned ledger is involved			12

applies to other notary schemes. This is a subject for future research. We consider the property 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.

Compared to relay and hash-locking schemes, notary schemes seem to increase the cost of transactions most 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 [25] 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, and how different interoperable DLs (e.g. Bitcoin and Ethereum) manage such bundled transactions.

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.

5.3. Relay schemes

Relay schemes introduce more complexity compared to hash-locking schemes. Namely, participants (i.e. owners of tokens) must interact with operators (see Section 4.4), and operators with smart contracts. This does introduce new issues. For

example, operators may collaborate to raise the price of fees for every lock/unlock action they receive. Operators could delay adding information to the smart contract, which effectively could delay transaction verification. This may be an issue in use cases where fast transaction finality is required, for example in high frequency stock trading. The zone spend attack, described in Section 5.1, also applies in relay schemes, with the only difference that there is no third party involved.

5.4. Hash-locking schemes

Hash-locking schemes (aim to) guarantee an atomic swap, under the assumption that none of the parties involved deviates from the scheme. If one of the parties does deviate in a hash-locking scheme, the atomic swap is either guaranteed not to happen, or the party that deviates from the scheme is disadvantaged. Therefore, honest participants can safely assume that all parties will stick to the scheme.

The term atomic is, however, somewhat misleading in probabilistic consensus algorithms, as the atomic swap can be reversed on one of the interoperating ledgers. Only with increased probability will there be no ledger change that influences the swap. Therefore, a better term would be *probabilistic swap* when a probabilistic consensus algorithm is involved in the swap.

Note that notary schemes and relay schemes may provide an atomic swap by using a hash-locking scheme. However, this requires either a third party when using notary scheme, or an exchange of ledger information in case of a relay scheme.

A possible downside of hash locking schemes is the financial denial of service, due to time-locking conditions. This occurs when one of the swapping parties goes back on their end of the swap. Te other party has to wait at most the maximum of the time-out at which the funds are returned. In principle, this issue can be dealt

with by initiating an atomic swap using a notary scheme or relay scheme. Finally, hash-locking requires an out-of-band channel in a censorship-resistant manner to exchange the hash which is being used for the swap. [9].

5.5. Permissioned ledger interoperability

Most research on ledger interoperability is focused on interconnecting permissionless ledgers. Currently, there appears to be even even less research on permissioned ledger interoperability. We can however, to some extent, reason about permissioned ledger interoperability.

A distributed ledger is permissioned if a limited set of known participants determines the next state of that ledger. Whereas permissionless ledgers use Nakamoto-type consensus algorithms (a probabilistic algorithm) to prevent Sybil attacks, this is not required by permissioned ledgers, as its participants determining the next state are known and fixed. Typically, Byzantine Fault Tolerant (BFT) consensus algorithms are used to create consensus in permissioned distributed ledgers. These algorithms cannot fork, unlike Nakamoto-type consensus algorithms, and transactions stored are final, hence it is called deterministic consensus.

We use the properties proposed in Section 3 to discuss interoperability issues with permissioned ledgers, and make some observations related to the properties.

Scope. The implications of connecting both permissioned and permissionless ledgers are typically not considered by interoperability solutions. The scope increases when permissioned ledgers interconnect with permissionless ledgers, and may not always be desired as the state shared between the two ledgers now becomes publicly available.

Scalability. In general, transaction throughput is much higher in permissioned

ledgers than the transaction throughput in permissionless ledgers. Typically consensus algorithms in permissioned ledgers are deterministic, meaning that the update function (see next property) can ensure quick finality. Therefore, interoperability between permissioned ledgers allows for a higher scalability than when interoperability has to be achieved with a probabilistic ledger.

Update function. Permissioned-to-permissioned ledger interoperability does not suffer from the zone-spend attack, as transactions are final once written to the ledger.

Permissioned-to-permissionless interoperability decreases the transaction finality to that of the ledger that is slowest in processing transactions. This means that, as permissionless ledgers have typically a low transaction throughput, transaction throughput from a permissioned ledger to a permissionless ledger is throttled to that of the permissionless ledger.

Cost of a state change. In contrast to permissionless ledgers, permissioned ledgers participants that propose state changes typically need no financial incentive to propose new state changes. Therefore, transaction costs are much lower than transactions that occur in a permissionless ledger.

Native token. Following the previous property, as there typically is no financial incentive needed in a permissioned ledger, native tokens are usually not present. This may limit the use cases in which permissioned ledgers can be connected to permissionless ledgers. For example, transferring a token from a permissionless ledger to a permissioned ledger requires some form of token on both ledgers.

Similar to permissionless ledgers, interoperability solutions provide **semantic** and **syntactic interoperability** for permissioned ledgers.

Technology development. Permissioned ledgers are governed by a small set

of participants. This is in contrast to permissionless ledgers where, given its decentralized nature, all participants are part of its governing body. Smaller governing bodies reach consensus more easy than that of entire decentralized communities [45]. Therefore, we assume that permissioned ledgers can better adapt to a changing environment than permissionless ledgers.

Regulation. Permissioned ledgers, typically used in corporations, are subject to regulation. As there exists control by the ledger participants on who can read and write to the ledger, complying to regulation can be achieved. Corporations would want interoperability solutions also to comply to regulation. This is, however, a topic rarely addressed by interoperability solutions.

6. Future Work

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 deter-

mine 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 with DL interoperability, such as the introduction of a third party (in notary schemes), state finality between probabilistic and deterministic ledgers, state distribution between permissioned and permissionless ledgers, and in particular the zone spend attack. Proposing solutions for these issues may be part of future work on DL interoperability.

7. Conclusion

Despite the technical differences of interoperability solutions for distributed ledgers (DL), the functionalities offered by the sub-categories of these solutions are similar. Therefore, the classification of interoperability solutions, as proposed by Buterin [1], is a proper classification to distinguish between such solutions.

We proposed an approach to further distinguish interoperability solutions and their sub-categories, based on the key properties of these solutions. We have shown that it is possible to describe, analyze, and evaluate DL interoperability solutions with these properties.

We identified several critical issues, some of which are inherent to distributed ledgers. In particular, we described in detail the zero-spend attack, which is applicable to all interoperability solutions that include a DL with a probabilistic consensus algorithm. In fact, we consider this attack, including the consequences of its mitigation measure, to be critical for deciding on whether or not ledgers should interoperate. A successful attack would lead to an invalid state between immutable

ledgers. The current solution to this attack, i.e. wait for a sufficient time (e.g. one hour), significantly reduces 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 documentation, and most projects on interoperability have shown only early proof-of-concepts.

Finally, we conclude that many interoperability topics require further attention, such as the zone-spending attack, 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.

References

[1] V. Buterin, Chain interoperability, 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, 2015. 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] Coinmarketcap, All Cryptocurrencies, 2018. https://coinmarketcap.com/all/views/all/.

[6] R. Beck, C. Müller-Bloch, Blockchain as radical innovation: A framework for engaging with distributed ledgers as incumbent organization, in: Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS 2017); Waikoloa Village, Hawaii, USA.

[7] Chain Interoperability: R3 Viewpoint, 2016. https://www.r3cev.com/s/Chain-Interoperability-R3-Viewpoint-2.pdf.

[8] H. Jin, X. Dai, J. Xiao, Towards a novel architecture for enabling interoperability amongst multiple blockchains, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp. 1203–1211.

[9] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, W. J. Knottenbelt, Xclaim: Interoperability with cryptocurrency-backed tokens, Technical Report, 2018. https://eprint.iacr.org/2018/643.

[10] BTCRelay. a bridge between the Bitcoin blockchain & Ethereum smart contracts, 2018. http://btcrelay.org/.

[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 Computing Surveys (CSUR) 22 (1990) 183–236.

[13] M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery, ACM Transactions on Computer Systems (TOCS) 20 (2002) 398–461.

[14] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, E. W. Felten, Sok: Research perspectives and challenges for bitcoin and cryptocurrencies, in: Security and Privacy (SP), 2015 IEEE Symposium on, IEEE, pp. 104–121.

[15] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008). https://bitcoin.org/bitcoin.pdf.

[16] T. Haerder, A. Reuter, Principles of transaction-oriented database recovery, ACM Computing Surveys (CSUR) 15 (1983) 287–317.

[17] M. Rosenfeld, Analysis of hashrate-based double spending (2014). https://arxiv.org/pdf/1402.2009.

[18] S. King, S. Nadal, Ppcoin: Peer-to-peer crypto-currency with proof-of-stake (2012). https://pdfs.semanticscholar.org/0db3/8d32069f3341d34c35085dc009a85ba13c13.pdf.

[19] 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.

[20] 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.

[21] The Cosmos Hub, 2018. https://cosmos.network/docs/introduction/cosmos-hub.html.

[22] Cosmos Whitepaper, 2018. https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md#horizontal-scaling.

[23] J. Kwon, Tendermint: Consensus without mining (2014). https://cdn.relayto.com/media/files/LPgoWO18TCeMIggJVakt_tendermint.pdf.

[24] C. Unchained, The technicals of interoperability. Introducing the Ethereum peg zone, 2018. https://blog.cosmos.network/the-internet-of-blockchains-how-cosmos-does-interoperability-starting-with-the-ethereum-peg-zone-8744d4d2bc3f.

[25] G. Wood, Polkadot: Vision for a heterogeneous multi-chain framework (2016). https://icowhitepapers.co/wp-content/uploads/PolkaDot-Whitepaper.pdf.

[26] A. Miller, Y. Xia, K. Croman, E. Shi, D. Song, The honey badger of BFT protocols, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, pp. 31–42.

[27] Bitcoin Wiki. Simplified Payment Verification, 2018. https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification.

[28] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on Ethereum smart contracts (sok), in: Principles of Security and Trust, Springer, 2017, pp. 164–186.

[29] J. Teutsch, M. Straka, D. Boneh, Retrofitting a two-way peg between blockchains (2018). https://people.cs.uchicago.edu/~teutsch/papers/dogethereum.pdf.

[30] I. Bejarano, O. Guinzberg, Using superblocks to bridge Dogecoin to Ethereum (2018). https://github.com/dogethereum/docs/blob/master/superblocks/superblocks-white-paper.pdf.

[31] Bitcoin Forum. Alt chains and atomic transfers, 2013. https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949.

[32] Komodo. An advanced blockchain technology, focused on freedom, 2018. https://komodoplatform.com/wp-content/uploads/2018/06/Komodo-Whitepaper-June-3.pdf.

[33] G. Zyskind, C. Kisagun, C. Fromknecht, Enigma catalyst: A machine-based investing platform and infrastructure for crypto-assets, 2018. https://enigma.co/enigma_catalyst.pdf.

[34] M. Herlihy, Atomic cross-chain swaps (2018). https://arxiv.org/pdf/1801.09515.

[35] M. Borkowski, D. McDonald, C. Ritzer, S. Schulte, Towards atomic cross-chain token transfers: State of the art and open questions

within TAST (2018). http://dsg.tuwien.ac.at/staff/mborkowski/pub/tast/tast-white-paper-1.pdf.

[36] Decred-compatible cross-chain atomic swapping, 2018. https://github.com/decred/atomicswap.

[37] Atomic cross-chain trading, 2018. https://en.bitcoin.it/wiki/Atomic_cross-chain_trading.

[38] P. McCorry, E. Heilman, A. Miller, Atomically trading with Roger: Gambling on the success of a hardfork, in: Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer, 2017, pp. 334–353.

[39] P. Bennink, L. v. Gijtenbeek, O. v. Deventer, M. Everts, An analysis of atomic swaps on and between Ethereum blockchains using smart contracts (2018). https://homepages.staff.os3.nl/~delaat/rp/2017-2018/p42/report.pdf.

[40] bitcoin/bips/bip-0199.mediawiki, 2017. https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki.

[41] Bitcoin (USD) Price, 2018. https://www.coindesk.com/price/.

[42] H. Vranken, Sustainability of bitcoin and blockchains, Current opinion in environmental sustainability 28 (2017) 1–9.

[43] CEIC China Electricity Price, 2018. https://www.ceicdata.com/en/china/electricity-price.

[44] Blockchain.com. View information about a Bitcoin transaction, 2018. https://www.blockchain.com/nl/btc/tx/0bbcf2486bad550d43beca51c2686ab94124c3f53777f9b352954f2b62ffe4f8.

[45] G. Desanctis, R. B. Gallupe, A foundation for the study of group decision support systems, Management science 33 (1987) 589–609.