Gateway-based Interoperability for Distributed Ledger Technology

Guzmán Llambías[12], Bruno Bradach[1], Juan Nogueira[1], Laura González[1], Raúl Ruggia[1]
[1]Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
[2]Pyxis, Montevideo, Uruguay

{*gllambi, bruno.bradach, juan.nogueira, lauragon,ruggia*}*@fing.edu.uy*[1]
guzman.llambias@pyxis.tech[2]

Abstract

Blockchain is a distributed ledger technology (DLT) to manage data in a decentralised way. During the last years, interoperability has become one of the main challenges within blockchain research as blockchains increasingly require integration between each other. Indeed, blockchains work by design in silos of information as interoperability is not a native feature. The main efforts in the field are focused on blockchains, such as Bitcoin and Ethereum. However, interoperability in DLT remains an almost untouched area of work, as they introduce additional requirements focusing on privacy and identity. Although there are some interoperability solutions for DLT, they are either high-level design proposals that do not provide concrete implementations or focus on interoperability issues between business applications and blockchain platforms. In this paper, we propose a gateway-based platform-to-platform interoperability solution for DLT, which comprises a detailed solution design and a reference implementation. The proposal was assessed through the development of a social security case scenario and the evaluation through two interoperability frameworks. A reference implementation was built using two DLT: Hyperledger Fabric and Corda. Experimental results shows that it is possible to achieve technical interoperability between two heterogeneous DLT platforms using a gateway-based interoperability solution, relaxing decentralisation, data privacy, identity and authorisation management properties.

Keywords: distributed ledger technology, blockchain, interoperability, cross-chain transactions

1 Introduction

Blockchain is a distributed ledger technology (DLT) to manage data in a decentralised way. Bitcoin, its first application, had the purpose to provide a software infrastructure for decentralised payments on untrusted environments [1]. Bitcoin leaded the first generation of blockchains, which provides user anonymity, data immutability and transparency over a decentralised network of untrusted participants. Trust is achieved by following rules defined by consensus protocols that guarantee and define how transactions are reliably registered into the ledger, avoiding the misbehaviour of malicious participants. A second generation of blockchains, led by Ethereum [2], added support for Turing complete scripting capabilities and state management, that enabled and provided support for new use cases besides cryptocurrency exchange and transfer (e.g. art sale). More recently, organisations started building consortiums and using blockchain technology to improve their business processes, leading to enterprise blockchains [3]. These blockchains have additional requirements (e.g. data privacy and user identification) which were not supported by the original design features provided by Bitcoin and Ethereum. In fact, they needed to be closed and permissioned as opposed to the latter, which are open and unrestricted, allowing users to have full access to the ledger, participate on the consensus protocol, and record transactions. As a consequence, a new generation of distributed ledger technology has emerged to meet these requirements (e.g., Hyperledger Fabric [4] and Corda [5]). These new DLT use different ledger structures in opposition to pure blockchain ledgers. Furthermore, they require users to be identified and granted permissions in order to have access to the ledger or participate in the consensus protocol, unlike blockchains[1].

Besides all this, DLT work as information silos by design as interoperability is not a native feature. However, current scenarios require DLT to interoperate between each other [6–8]. Users started to require the exchange of assets (e.g. cryptocurrencies) and if they belong to different blockchains, this may not be a direct task. Furthermore, users may need to move their assets to other blockchains because they require new features or due to limitations in their blockchain. For instance, smart contracts is not a feature supported by Bitcoin and previous versions of Ethereum have limitations regarding low transaction throughput.

Initial proposals to enable blockchain interoperability focused on cryptocurrency exchange and transfer and many solutions were proposed [6, 9]. This is the case of Polygon PoS Bridge [10], an interoperability solution that enables the transfer of cryptocurrencies from Ethereum to Polygon blockchain and back. However, DLT interoperability remains a challenge, as it has specific requirements not covered by blockchain interoperability solutions. DLT are used by organisations to improve their business processes and their scenarios are more focused on data exchange, than cryptocurrency transfer or exchange [11, 12]. In this case, DLT interoperability is not a requirement in itself, it is an enabler for new opportunities to improve inefficiencies in such business process. The collaboration between HSBC and Banque de France is an example of DLT interoperability [13]. Both organisations carried out experiments to enable interoperability to exchange data between two DLT to achieve inter-bank settlement transactions and reduce the costs of their business process.

According to our previous work [9], only a few solutions where proposed for DLT interoperability [3, 14–16]. In general, these solutions focus on application-to-platform-interoperability, leaving out-of-scope platform-to-platform interoperability. Application-to-platform interoperability involves two or more DLT and an external third-party coordinator that coordinates independent local transactions on each DLT to enable interoperability. On the other hand, platform-to-platform interoperability involves two or more DLT that are coordinated between each other and execute local transactions using a cross-chain protocol without a third party that coordinates them. In that direction, Hardjono et al. [17] and Hardjono [18] proposed theoretical approaches for a gateway-based interoperability solution that promise platform-to-platform interoperability. However, the authors provide neither a detailed design or practical experimentation. To the best of our knowledge, no other work addressed these elements for gateway-based solutions for platform-to-platform interoperability between DLT.

This work is a substantially extended and thoroughly revised version of Bradach et al. [19]. Additional material includes a generalisation to DLT interoperability, the evaluation of the proposed solution using two interoperability frameworks, more details about the experimentation and an extended background.

This paper is positioned in the area of information systems integration, more precisely, on DLT interoperability. In particular, our paper focuses on the challenge suggested by Belchior et al. [6] with respect to shorten "the gap between theory and practice, including the lack of standardisation and implementations". This paper also provides a preliminary work to the challenge presented by Llambías et al. [9] to achieve interoperability between DLT.

The rest of the paper is organized as follows. Section 2 provides background concepts and Section 3 describes a social security interoperability scenario. Section 4 presents the detailed design of the proposal. Section 5 describes how the proposal was assessed through the implementation of a prototype, the development of a case scenario and the evaluation using two interoperability frameworks. Section 6 analyses related work. Finally, section 7 presents conclusions and future work.

2 Background

This section presents background concepts required for the comprehension of this work. This section was built based on our previous work, which included a literature review on blockchain interoperability [9].

2.1 Blockchain concepts

According to Xu et al. [20] a distributed ledger is a distributed storage across a network of machines (nodes) that stores transactions in an append-only mode. Once a transaction is registered into the ledger, it cannot be updated nor deleted. These authors also define blockchain as a distributed ledger structured as an ordered linked list of blocks, each one containing a set of transactions. Each block is linked to its predecessor block by cryptographic hashes that assure the security of the link (each block contains the hash of the previous block). Block data cannot be changed without breaking the link, which in practise provides immutability

[1]A blockchain is a distributed ledger and a distributed ledger is not necessary a blockchain. Despite this, the rest of the paper uses these terms interchangeably as they share common characteristics.

of data to the blockchain. The first block is called genesis block. Fig. 1 illustrates a general blockchain structure.

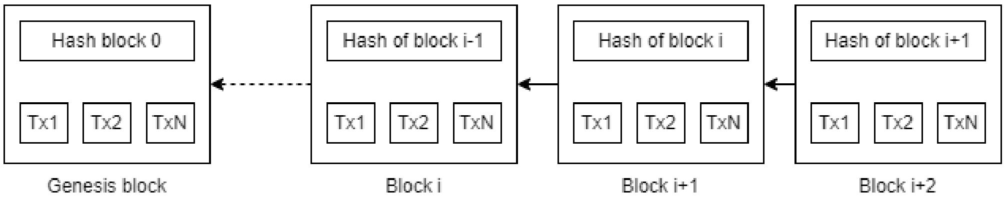

Figure 1: Blockchain data structure [21]

Distributed ledgers operate across a network of nodes without a central trusted authority to maintain the state of the ledger. Instead, they use consensus protocol rules to maintain its integrity and shared content [20]. These rules define how nodes agree to consider a transaction legitimate and be registered into the ledger. Consensus protocols may be probabilistic or deterministic. Probabilistic consensus protocols eventually guarantee the ledger's consistency with a high degree of probability. However, they do not assure transaction finality. There is always a probability that the transaction may be rejected. Proof-of-work is one of the most popular probabilistic consensus protocol and is used by Bitcoin. In Bitcoin, users assume a transaction to be final in case there are six new blocks after the transaction's block. They consider the probability being too low to be considered an illegitimate transaction. On the other hand, deterministic consensus protocols always ensure the finality of a transaction. After the transaction is registered on the ledger, the transaction is always considered final and correct. RAFT [22] is a deterministic protocol used by Hyperledger Fabric.

Distributed Ledger Technologies (DLT) provides the technological layer to operate and use distributed ledgers [23]. Blockchain platforms are a specialisation of DLT that enables the operation and usage of blockchains. They both provide the software and hardware required to run a node or any other software required to have access to the ledger. Bitcoin and Ethereum are two examples of blockchain platforms as they model the ledger as a blockchain. Corda and Hyperledger Fabric are two examples of DLT that use other ledger structures besides blockchain. In particular, Hyperledger Fabric's defines the concept of channel, where each channel has its own ledger that follows the blockchain structure. Users that participate on a channel, may not necessary participate on other channels. On the other hand, Corda follows a Direct Acyclic Graph (DAG) structure where each node of the graph is a transaction and each transaction depends on the output of one or more previous transactions. Unlike Hyperledger Fabric, transactions in Corda are not grouped in blocks. Fig. 2 depicts these examples.

Smart contracts are scripts deployed on the blockchain that are triggered by transactions on an autonomous way by any participant node. They can hold and transfer digital assets or invoke other smart contracts stored on the blockchain. Smart contracts only use data that is stored in the blockchain ledger or obtained from special external entities called Oracles. Smart contracts code is deterministic and immutable once deployed [20].

A blockchain provides the following non-functional properties once a transaction is registered in the ledger:

1. immutability: once data are registered on the ledger, they cannot be changed or deleted.
2. non-repudiation: every user must sign the transactions before sending them to the blockchain, which ensures the non-repudiation property.
3. decentralised: the blockchain storage and execution can be performed by any participant of the network.
4. provenance: blockchain users can trace data changes from its original state to its actual value and how it had changed over time.

Other non-functional properties may be available depending on the type of the blockchain: public, private or a consortium.

Public blockchains are a type of blockchain where users have full access to the ledger, participate in the consensus protocol and record transactions without restrictions. Users are pseudo-anonymous (participants use a pseudonym instead of their true identity) and there are no regulation rules to be followed. These blockchains are open and all participants may have a copy of the ledger. Bitcoin and Ethereum are two examples of public blockchains [21].

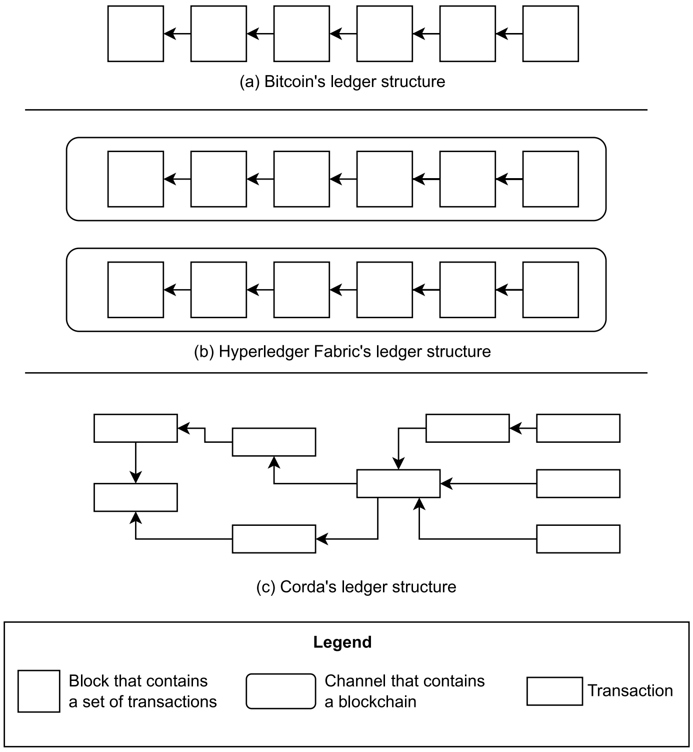

Figure 2: Distributed ledger structures

Private blockchains on the other hand are closed, used by a single organisation and only the organisation's users have access to the ledger, participate in the consensus protocol or record transactions. They have a high degree of trust in their participants, who need to be identified in order to participate in the network. Transactions carried out in the blockchain may be confidential and restricted for non-desired users. This more trusted environment enables these blockchains to use less reliable consensus protocols than public blockchains and provides them with higher transaction throughput [21].

Consortium blockchains are very similar to private blockchains, except that they are governed by a set of organisations [21].

Blockchains can be permissioned or permissionless. On permissionless blockchains users have full access to the ledger, participate in the consensus protocol and record transactions without restrictions. On the other hand, on permissioned blockchains users are restricted and need permissions to have access to the ledger as well as to participate on the consensus protocol. Access to the ledger may also be restricted to read or write access [21]. Generally, public blockchains are permissionless and private/consortium blockchains are permissioned. However, there are some particular cases (e.g. Sovrin[2]) which are public and permissioned.

2.2 Blockchain interoperability

According to Belchior et al. [6], blockchain interoperability involves a source blockchain network that initiates a transaction on its local ledger which must be executed on a target blockchain. Transactions that span the domain of a blockchain platform into another blockchain platform are called cross-chain transactions. Cross-chain transactions use cross-chain communication protocols to communicate blockchains. In certain cases, middleware software enables this cross-chain communication. On the other hand, other authors consider that blockchain interoperability involves an external third party that executes coordinated local transactions on each blockchain [3,14,16]. We consider Belchior et al. approach to be a platform-to-platform interoperability and the other authors approaches to be application-to-platform interoperability. Polygon PoS Bridge is an example of platform-to-platform interoperability, while Hyperledger Cactus is an example of application-to-platform interoperability. Polygon PoS Bridge acts as a middleware and listens to local transactions initiated on Ethereum to triggers local transactions to be executed on Polygon blockchain. Hyperledger Cactus executes two independent transactions on each blockchain and provides business logic plugins to coordinate its execution [15]. Fig. 3 illustrates these concepts.

Blockchain interoperability may involve homogeneous blockchains, that share the same constructs and

[2]https://sovrin.org/

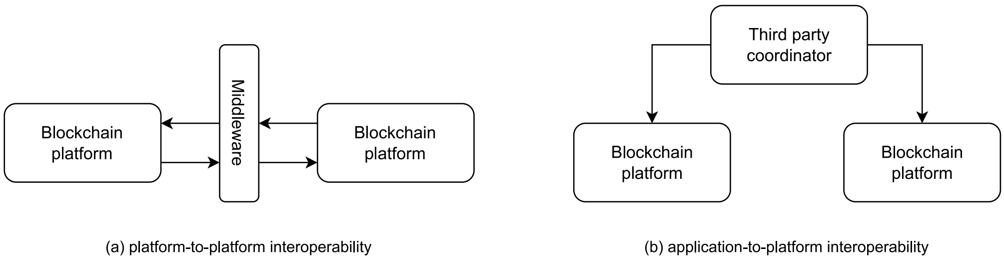

Figure 3: Blockchain interoperability types

technological primitives (e.g. blockchains based on the Ethereum Virtual Machine) or heterogeneous blockchains, which are completely different in structure, behaviour and technical primitives (e.g. Ethereum and Bitcoin). As blockchains can be permissionless or permissioned, different interoperability requirements need to be fulfilled regarding identity, authorisation, confidentiality and governance.

Blockchain networks are silos of information self governed not designed with interoperability in mind. In a blockchain context, interoperability defers from traditional interoperability requirements of information systems, where message format and communication protocols are usually enough to achieve interoperability (e.g. using SOAP web services). Semantic understanding, verification and validation of the exchanged data are requirements in this context [8]. Abebe et al. [3] provides a suitable definition of blockchain interoperability including technical and semantic requirements. They define blockchain interoperability "as the semantic dependence between distinct ledgers for the purpose of transferring or exchanging data or value, with assurances of validity or verifiability".

As aforementioned, our work focuses on platform-to-platform blockchain interoperability and in next sections the term blockchain interoperability is used to refer to this type of interoperability.

2.3 Blockchain interoperability modes

Blockchain interoperability may involve the exchange and transfer of two types of artefacts: data and assets [12]. Data are raw bytes formatted as strings that represent a piece of information. Data artefacts work at the technical layer and can be copied from a source blockchain to a target blockchain. On the other hand, assets are a type of artefact that have a technical representation, but also have a semantic value. Assets may be fungible or non-fungible. Fungible assets share the same type and value between each other and are interchangeable. Criptocurrencies like Bitcoin, are an example of fungible asset. Non-fungible assets (also known as non-fungible tokens) are unique, indivisible, have an owner, are not interchangeable and it is possible to prove their scarcity. Non-fungible assets may or may not be linked to a physical entity. In case they are, they may be called a digital twin. Considering their properties, non-fungible assets should not be copied between blockchains. Instead, non-fungible assets are transferred or exchanged. Two popular examples of non-fungible assets from the art domain are Crypto Punks[3] and Bored Ape Yacht Club[4].

Considering the previous definitions, blockchain interoperability may involve three modes: data transfer, asset exchange and asset transfer [12].

The data transfer mode is presented graphically in Fig. 4, where data from a source blockchain is copied to a target blockchain.

Asset transfer involves an asset that belongs to a user and must be moved from a source blockchain to a target blockchain. The asset must be burned or locked on the source blockchain and a semantic equivalent asset must be generated or minted and assigned to a new user on the target blockchain. The target blockchain must assure that the asset was burned/locked on the source blockchain before minting the new asset. An example of asset transfer is provided by Polygon PoS Bridge to transfer assets from Ethereum blockchain to Polygon blockchain. For example, Ethers[5] are locked on Ethereum and a semantic equivalent value of MATIC[6] is minted on Polygon. Fig. 5 depicts the asset transfer mode.

Finally, the asset exchange mode involves two users U_1 and U_2 that want to exchange two assets they hold on two different blockchains. Both users participate in both blockchains. This mode requires two operations to be atomically made on each blockchain. The first operation must do a local transfer of asset A on blockchain A from user U_1 to user U_2. The second operation must do a local transfer of asset B on blockchain B from user U_2 to user U_1. If these two operations are not executed atomically, only one transfer

[3]https://www.larvalabs.com/cryptopunks
[4]https://boredapeyachtclub.com/
[5]Ethereum cryptocurrency
[6]Polygon cryptocurrency

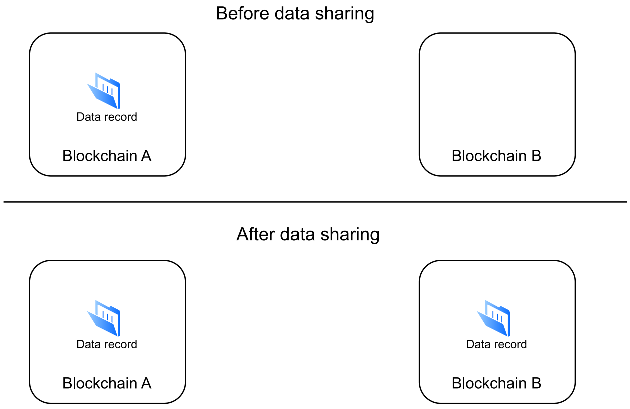

Figure 4: Blockchain interoperability mode: data transfer

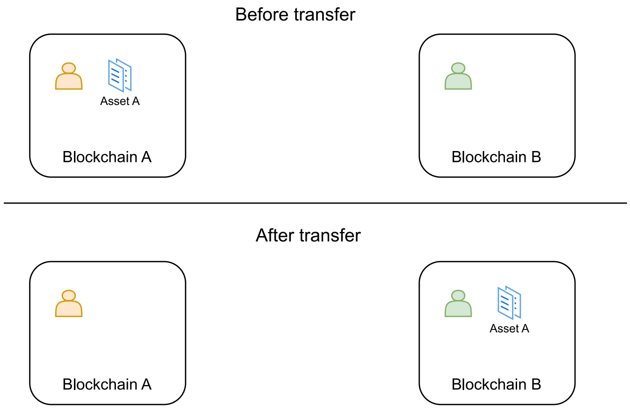

Figure 5: Blockchain interoperability mode: asset transfer

may occur, leaving one user with both assets. This mode is depicted in Fig. 6. An example of asset exchange is the exchange of criptocurrencies allowed by decentralised exchanges like the Komodo Platform[7].

These three interoperability modes are hard to achieve, as stated on the data accept and access problem presented by Pillai et al. [24]. This problem defines technical and practical limitations to accept or access data between blockchains. Technical limitations arise as blockchains are isolated systems that follows consensus protocols between their participants to define the current state of the ledger. Assets and data are created and exist only within the blockchain system. Therefore, accepting data or assets from a source blockchain, implies that the target blockchain must achieve consensus with the source blockchain. In other words, the target blockchain must have access to the source blockchain ledger and validate its content to accept the incoming data/asset. On the other hand, practical limitations arises to access data from external sources. The decentralised nature of blockchains requires that all participants reach consensus about the state of the ledger. If participants need to fetch data from an external source, secure integration protocols must be used to prevent receiving different results (given the dynamic nature of external data) that may interfere with the consensus protocol.

2.4 Blockchain interoperability solutions

This section presents blockchain interoperability solutions following the classification presented by Belchior et al. [6].

2.4.1 Notary scheme

Notary scheme is a trusted third party solution that comprises an entity called notary, which monitors multiple blockchains and reacts upon events triggering cross-chain transactions on other blockchain [6]. Blockchain networks trust the third party and do not require proofs or further cross-chain validation. Central exchanges are the most popular type of Notary Scheme and provide users a way to exchange cryptocurrencies between different blockchain platforms.

[7]https://komodoplatform.com/en/

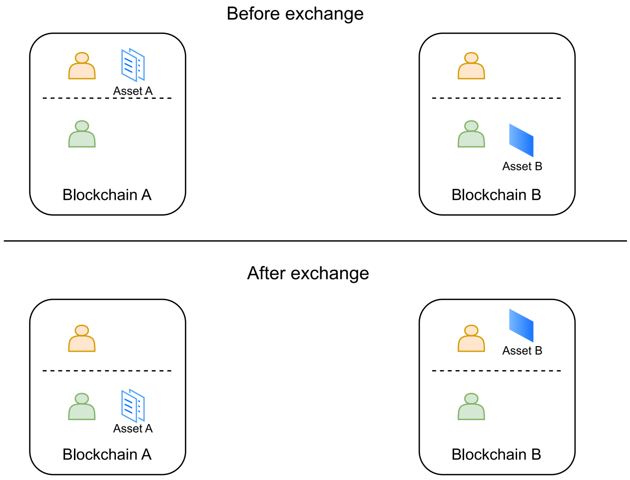

Figure 6: Blockchain interoperability mode: asset exchange

2.4.2 Sidechain/relays

Sidechains enable interoperability between two blockchains, a main chain and a secondary chain (or sidechain), which is an extension of the first one. Communication between the main chain and sidechain can be one-way (i.e. the main chain sends information to the sidechain) or bi-directional, two-way peg [6] (i.e. both blockchains communicate with each other). An example of a two-way peg is the Liquid Network: a sidechain of Bitcoin [25].

Sidechains communications require a cross-chain protocol and a relay. The relay is a piece of software hosted on the sidechain that verifies the validity of cross-chain transactions that are received by the sidechain following the consensus rules of the main chain. The BTC Relay is an example of relay [26].

2.4.3 Atomic Swaps/Hash Time Lock Contracts

Hash Time Lock Contracts is the combination of time locks [27] and hash locks [28] used to implement atomic exchanges between two different blockchains. Atomic swaps is the most reliable solution for asset exchange between parties with zero trust. However, one drawback is the long wait needed in order to release the timelocks. In some scenarios like the cryptocurrency exchange, these waits may be an issue, considering the fast volatility of cryptocurrency value [7].

2.4.4 Trusted relays

Trusted relays are trusted parties hosted aside each blockchain platform that enable interoperability between them. Requests received at the source blockchain's trusted relay are redirected to the target blockchain's trusted relay and after validation, redirected to the target blockchain. In case the operation is a request-response interaction, response messages are sent to the target trusted relay that secures the response and redirects it to the source trusted relay, which validates the message according to the target blockchain consensus protocol [6]. Arbitrary end-user business logic can be implemented inside of the trusted relay in order to build complex orchestrations among multiple blockchain platforms. Hyperledger Cactus is an example of trusted relay [15].

2.4.5 Blockchain of blockchains

Blockchain of Blockchains is a generalisation of the Sidechain/Relay solution, comprising a main chain and multiple sidechains collaborating with each other. It provides a framework for reusable data, networking, consensus, incentive and contract layers to integrate the sidechains between each other using the main chain [6] [7]. When a sidechain needs to send a transaction to another sidechain, it uses the main chain as a routing mechanism and as the consensus layer. Blockchain of blockchains is a trusteless solution but at the cost of a high complexity. Cosmos [29] and Polkadot [30] are two examples of this type of solution.

2.5 Levels of Conceptual Interoperability Model for Blockchain

Tolk and Muguira proposed in 2003 a framework called Levels of Conceptual Interoperability Model (LCIM) [31] to evaluate the levels of interoperability software systems can achieve. This model was later on extended and nowadays has seven levels of interoperability. This model has been applied to several domains (e.g. health, multi-agent systems systems) [32, 33], where each level defines the capabilities a software must have and which type of interoperability achieves. Fig. 7 depicts a graphical representation of the LCIM model. The lowest levels of the model refer to none or little interoperability, while the top levels refer to the highest level of interoperability a software system may achieve. Level 1 of the model requires software systems to exchange raw data between them (e.g. bits over the network). Level 2 states that systems exchange data with structure where a message format is used (e.g. SOAP message format [34]). In Level 3, software systems exchange messages and they understand the meaning of them. In Level 4, software systems are aware of the purpose and context of why messages are exchanged. At level 5, interoperating software systems share a common state model that defines the behaviour of each system. A change in the model may trigger changes in the behaviour of one or more systems. Finally, in level 6, software systems have a complete understanding of data models, concepts and assumptions while they are exchanging messages. At this level, process were standardised and software systems share a common goal. As a result, organisations are conceptually interoperable.

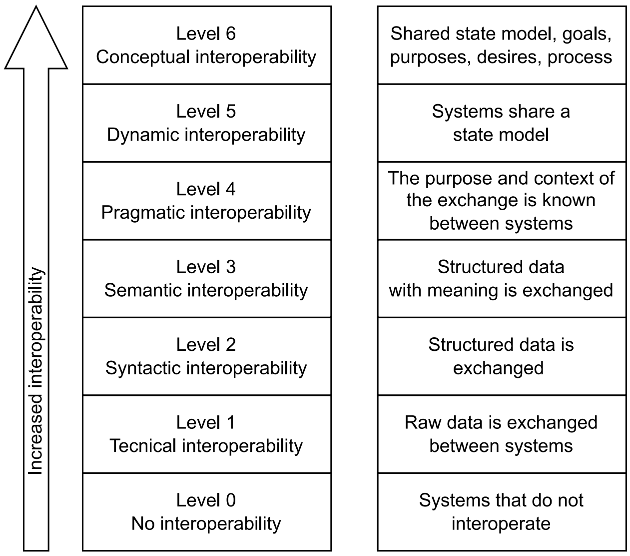

Figure 7: Levels of Conceptual Interoperability Model (adapted from [33])

Later on, Pillai et al. [35] extended the LCIM model and proposed an extension for blockchain based systems. The proposed extension considered only five levels. Level 6 was discarded as the authors stated that is not achievable that one blockchain dynamically exchange messages with another blockchain, as this behaviour may interfere with the consensus process and arrive at different results. The extended model defines Level 1 at the technical level, where blockchains platforms share network communication protocols. Level 2 specifies that blockchain platforms share the same data structure (e.g. blocks and transaction format), although the value they carry is not necessary the same. For example, the value of Ethereum assets will not be recognised by Ethereum Classic [36] and vice versa. At these two levels, blockchain platforms relies on third-party entities to enable the exchange of messages between them, as they do not provide native capabilities to achieve this task. The protocol used by this exchange needs to provide solutions to the data access and acceptance problem described in Section 2.3. The following upper levels relies on integration services (native to the blockchain platform) that enables asset transfer, asset exchange and data sharing following the consensus protocol rules of each blockchain. At level 3, blockchain platforms share an understanding of the value they are exchanging. Level 4 is achieved when multiple blockchain platforms are involved after the execution of a function on another blockchain platform. For example, to allow different blockchains to exchange message through a gateway. Finally, at Level 5, changes on a source blockchain platform triggers changes on a target blockchain platform, where blockchain platforms are aware and understand state changes between them.

2.6 Cross-blockchain integration design decision framework

Pillai et al. proposed a cross-blockchain integration design decision framework (CBIDD) to choose the most suitable interoperability solution considering security assumptions [24]. The proposed framework is depicted in Fig. 8, has five stages and was designed to help stakeholders to design the most suitable interoperability solution. The first stage requires the stakeholders to identify the value type of the integration. The value type can be a crypto-coin (e.g. Bitcoin), a crypto asset (e.g. fungible asset) or data (e.g. social security data of a citizen). Stage 2 defines the integration goal and can be one of the interoperability modes presented in Section 2.3: data exchange, asset transfer or asset exchange. The crypto-coin value type can only be used with the asset exchange interoperability mode, while the data value type can only be used with data exchange. Crypto-assets can be combined with asset transfer and exchange. Step 3 identifies the integration approach that can be centralised or decentralised. On the centralised approach only one entity operates the integration process, while on the decentralised approach there exists a set of entities that controls it. Stage 4 identifies the integration mode to apply, where the stakeholder can choose between direct, third party, bridge, connector and others. Direct integration relies on the Relays described in Section 2.4. Third party mode relies on the Notary Scheme solution. The Bridge integration mode includes Trusted Relays, while the connector mode refers to the Blockchain of Blockchains interoperability solution. Other modes include Oracles and APIs as an integration mode. Finally, stage 5 defines the integration protocol to address the data accept problem presented in Section 2.3. The integration protocols available are atomic swaps, lock/unlock, burn/mint and others. Atomic swaps is described in Section 2.4. Lock/unlock protocol requires a temporal asset transfer from a source blockchain to a target blockchain, with the assurance that the asset can return back to the source blockchain. During the temporal transfer, the asset is locked on the source blockchain and no changes can be applied to it. The burn/mint protocol defines a permanent transfer of value from a source blockchain to a target blockchain. As a result, the asset is destroyed (burned) on the source blockchain. Other options include custom protocols for message exchange, like IBC [37].

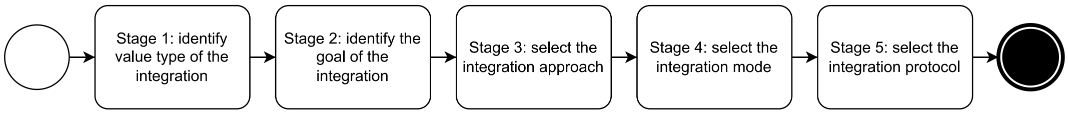

Figure 8: Cross-blockchain integration design decision framework (adapted from [24])

3 Interoperability motivational scenario

This section presents a motivational social security scenario, considering two organisations that already use blockchain and need to interoperate between each other using the data transfer mode. This motivation scenario is inspired by the pilot carried out by the International Social Security Association (ISSA) to assess the technical feasibility to use blockchain technology for social security data exchange among social security organisations [38].

3.1 General description

Social security organisations from different countries frequently exchange information about citizens regarding working years or life status. This information exchange is required in situations where people have worked in one country but decided to receive their retirement income in another country.

When a country needs social security information, it must generate a request to the corresponding country and wait for the response. According to agreements between countries, information requests must be fulfilled before a specified due date. In this context, it is common to misunderstand the exact due date in which responses need to be sent. In order to avoid this issue, blockchain seems a suitable technology to use. Blockchain technology provides the accountability features required to avoid misunderstandings between a consortium of organisations. Furthermore, it enables high transparency for all parties involved and clearly states to everyone when the information was requested and when it needs to be answered. In addition, its immutability feature provides reliability to every party, as opposed to a central database that needs to be managed by a organisation that is trusted among all participants. In this setting, social security agencies of the European Union may develop a consortium blockchain to deal with this issue. This may be replicated by other grouped countries, like Mercosur for example. As there is a strong cultural relationship between some countries of the European Union and Mercosur, it seems necessary to exchange social security data between these two consortia. For example, Uruguay and Spain have signed an international social security agreement that may need this exchange [39]. The two regional organisations need to interoperate their blockchains in order to have a unique source of truth regarding due dates.

Fig. 9 shows a graphic representation of the scenario, considering Mercosur and the European Union as two consortia with their own blockchains. Every time a country from Mercosur sends an off-chain request of data to an European country, it generates an on-chain transaction and saves the requestID with its request date and message hash on its own blockchain (Mercosur). This on-chain transaction generates a cross-chain transaction to the other country's blockchain saving the requestID, message hash and request date. The target country sends an off-chain response and generates an on-chain transaction on its own (European Union) blockchain that saves the requestID, response date and response message hash. This on-chain transaction generates a cross-chain transaction having the Mercosur blockchain as destination. After receiving the cross-chain transaction, the Mercosur blockchain saves the cross-chain transaction data (requestID, message hash) on its own ledger.

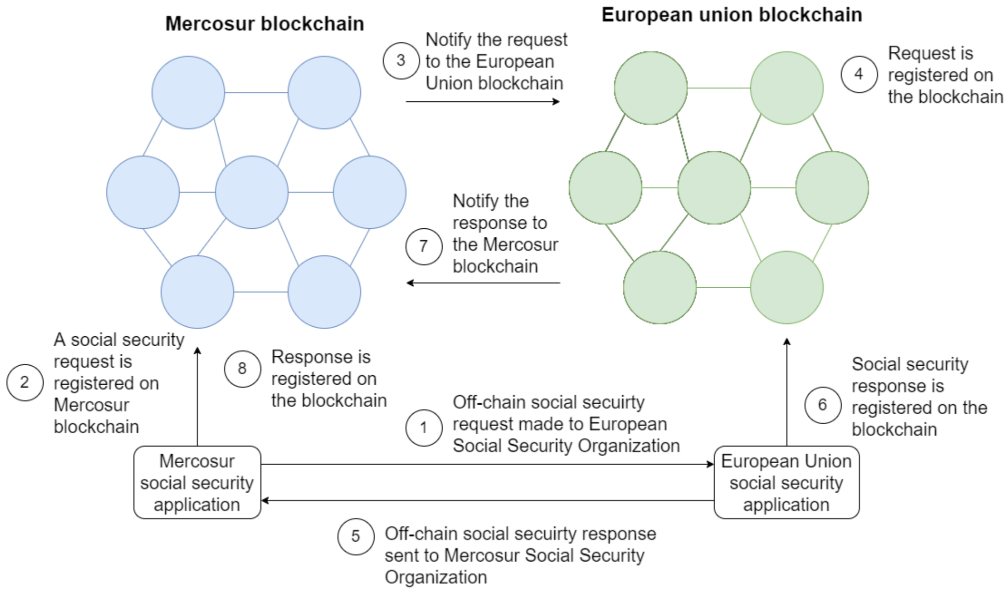

Figure 9: DLT interoperability in a social security scenario

3.2 Interoperability requirements

This section presents the functional requirements of the motivational scenario.

The business application of each country can submit a request for information to another country (off-chain transaction). For each request of information, a blockchain transaction must be registered on each blockchain with the following data: Global Unique Identifier (GUID) of the request, country that makes the request, destination country of the request, date of the submitted request, expected date of the response and hash of the off-chain message request.

The business application of each country can submit a response to another country (off-chain transaction). For each response of information, a blockchain transaction must be registered on each blockchain with the following data: GUID of the correlated request message, date of the submitted response, hash of the off-chain message response.

Finally, every on-chain transaction related to a request/response of information registered on one blockchain, must have the corresponding on-chain transaction registered on the other blockchain.

4 DLT interoperability gateway

This section describes the proposal, including its guiding design principles, a high-level description as well as details of its main components, interactions and design principles compliance.

4.1 Design principles

Inspired by the work of Abebe et al. [3], the proposed DLT interoperability solution is guided by the following design principles:

- DLT heterogeneity: As DLT may have different architectures and are independent of each other, the solution design must support different types of integration alternatives.

- DLT independence: DLT may evolve independently of each other to provide new capabilities or improve them. Therefore, the solution must evolve as DLT evolves, but only involving DLT integration components (not others).

- Non-invasive: Changing the source code of a DLT is not an easy task and may imply soft (e.g. Taproot on Bitcoin) or hard forks (e.g. Ethereum Classic fork), that need an intensive testing and quality assurance process. Therefore, the solution must avoid hard fork at all costs and is desirable to avoid soft forks as they must go through a governance process. In particular, to apply our solution in an immediate way, changes to the DLT must be avoided.

- Technology agnostic: In order to ease the integration, the solution must be agnostic of its underlying implementation technology (e.g. by using standard technologies and communications, as much as possible).

In addition, there are other relevant design principles for DLT interoperability that were considered to be part of the work, but were left out of the scope:

- Preserve DLT properties: The solution must keep the properties of the DLTs that integrates. Namely, it is not appropriated to provide a solution that does not keep data confidentiality when integrating DLT. Therefore, the solution must keep user identity management, data authorisation and data privacy.

- Decentralisation: The solution must avoid centralised services or trusted third parties in order to be a full decentralised solution. Having centralised or trusted parties does not comply with DLT decentralised nature.

4.2 General description

The proposed solution focuses on solving the technical interoperability between two DLT, following a gateway architecture inspired by the work presented by Hardjono et al. [17]. In particular, it is classified as a trusted relay solution. The gateway acts as a middleware between the DLT and translates the messages received from the source DLT to the target DLT. Its responsibility is to adapt communication protocols and apply message data format transformation.

As illustrated in Fig. 10, the gateway-based solution is composed of a router component and one connector for each of the involved DLT. Connectors communicate with the Router using a common data format and common communication protocols. Connectors communicate with the DLT using the native DLT data format and communication protocols. Common data format is later on presented in Section 4.5 in Table 1 and Table 2.

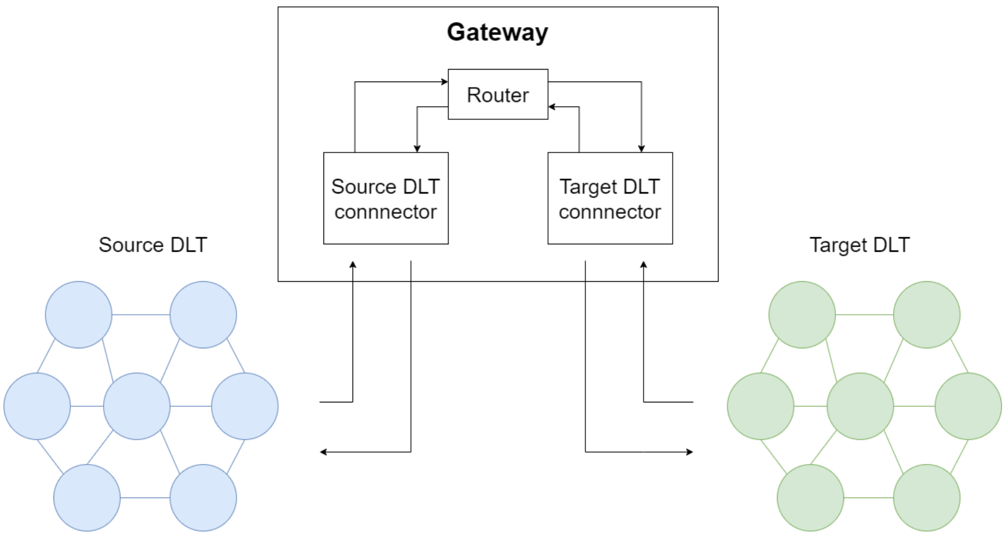

Figure 10: Gateway-based interoperability solution

When a source DLT needs to send a cross-chain transaction to a destination DLT, it must generate a message and send it to the connector component. This component transforms the message to a common data format and redirects it to the router, that checks its destination and routes it to the corresponding connector of the target DLT. The target connector receives the message, transforms the message from the common data format to the target DLT specific data format. After this transformation, the connector sends the transformed message to the target DLT using its native communication protocol.

4.2.1 Connector component

The connector component has the responsibility of communicating the DLT with the router and solving two tasks: i) receive a DLT message, transform it to a common data format and redirect the common data format message to the Router component, and ii) receive a common data format message from the Router and send it to the DLT in its native data format and following its communication protocol.

Given the diversity of DLT, the solution allows to include one connector for each DLT. The connector is tightly coupled with the DLT and implements the integration logic to communicate with it. It provides a common interface to the Router, decoupling it from the specific DLT implementation details.

Connectors can be integrated with the DLT following two modes: passive and active mode [40]. In passive mode, the connector monitors the DLT and generates the necessary events. In active mode, the DLT explicitly sends messages to the connector component.

4.2.2 Router component

The Router component has the responsibility of routing messages received from a connector to the corresponding destination (also a connector component), without changing its content. In order to perform this task, the Router follows the Content Based Routing pattern (of the Enterprise Integration Patterns [41]) and uses a static DLT discovery mechanism [11]. Whenever a message arrives to the Router, it will check the target DLT name provided on the message and with this value, the Router will query its local registry and get the connector address. With this address the Router will redirect the message to the target connector's address. Section 5 will explain the details regarding this behaviour and an example of the registry and message is provided on Listing 1 and Listing 8.

4.3 Main interactions

Interoperability between a source DLT and a target DLT using the proposed solution is achieved through the following interactions:

1. The source DLT registers a transaction and sends an event addressed to its connector. This event includes information of the target DLT and business data.
2. The source DLT's connector receives the event and sends it to the router.
3. The router receives the message, gets the name of the target DLT from the message, checks the DLT connectors registry and sends it to the target DLT's connector.
4. The target DLT connector receives the message and sends it to the target DLT.
5. The target DLT receives the message and invokes a smart contract on the target DLT.

4.4 Compliance with design principles

In order to comply with the design principles described in Section 4.1, a key decision was to design the router and connectors as independent components, that communicate using HTTP endpoints:

- DLT heterogeneity: DLT may be implemented with different technologies. Designing the three components as a monolith would have limited DLT integrations. The connector is tightly coupled with the DLT and provides an abstraction to the Router and the rest of the Connectors components. This design decision enables different types of DLT behaviours, as the Connector adapts to its own DLT and standardize the interactions with the Router.
- DLT independence: Connectors may evolve as the DLT evolves. As this behaviour is encapsulated in the Connector, it does not affect the overall solution. By having the connector as an independent piece of software, it can evolve independently of the rest of the components as long as it follows the interface defined in Section 4.
- Non-invasive: The proposal does not require modifications on the DLT source code, as Connectors and Routers are external components to the DLT. The unique restriction is that DLT have to be able to emit events that can be listened by the Connectors and have support for smart contracts to receive messages.
- Technology agnostic: The use of standard protocols to implement cross-chain interactions, enables the development of Connectors using the most suitable technology according to the corresponding DLT.

As already mentioned in Section 4.1, decentralisation and preservation of DLT properties are design principles that were left out of scope of this work. In particular, the proposal follows a centralised approach and does not tackle identity, authorisation and data privacy requirements of permissioned blockchains.

4.5 Connectors and router interfaces

The solution defines interfaces for connectors and the router, which are described in the following subsections.

4.5.1 Connectors interface

Connector components must expose an interface to receive messages from the Router. In particular, connectors must expose an HTTP endpoint to receive requests using the POST method, containing a payload with the structure described in Table 1.

Table 1: Connector's common data format

Field	Type	Description
targetContract	String	Contains the name of the target smart contract on the target DLT.
data	Any	Contains the payload message with the data to be sent to the smart contract on the target DLT (e.g. its input parameters).

4.5.2 Router interface

The Router component must expose an interface to receive messages from connectors.

In particular, the router must expose an HTTP endpoint to receive requests using the POST method, containing a payload with the structure described in Table 2

Table 2: Router's common data format

Field	Type	Description
targetBlockchain	String	Contains the name of the target DLT to which the message must be redirected to.
targetContract	String	Contains the name of the smart contract on the target DLT.
data	Any	Contains the payload message with the data to be sent to the smart contract.

5 Implementation and assessment

The proposed solution was assessed through the implementation of a prototype, the development of a case scenario and the validation against two interoperability frameworks: LCIM and CBIDD. The authors of this work selected both frameworks for convenience in terms of knowledge and availability. Each assessment method provided different perspectives to validate the proposal. The goal of the implementation prototype was to analyse the technical feasibility of the proposal and provide a reference implementation. The goal of the case scenario was to illustrate its functional behaviour. The goal of the LCIM assessment was to determine the level of interoperability achieved. The goal of the CBIDD assessment was to verify the gateway-based solution served as a suitable approach to the social security scenario presented in Section 3. These three approaches, the prototype, development of the case scenario and the two interoperability assessments, provide a step forward towards the assessment of the proposal.

Section 5.1 presents the implementation of the prototype. Section 5.2 presents the functional assessment. Section 5.3 presents the LCIM assessment. Section 5.4 presents the CBIDD assessment. Finally, Section 5.5 presents a discussion of the obtained results.

5.1 Reference implementation prototype

This section describes implementation details of the prototype, which provides a reference implementation for the solution. The goal of this prototype was to validate the technical feasibility of the proposal. Further details of the implementation as well as its source code are available online[8].

[8]https://gitlab.fing.edu.uy/open-lins/blockchain-interoperability

Fig. 11 presents the architecture of the prototype. In this case, the Gateway is composed of: i) the Router, ii) the Hyperledger Fabric Connector, and iii) the Corda Connector.

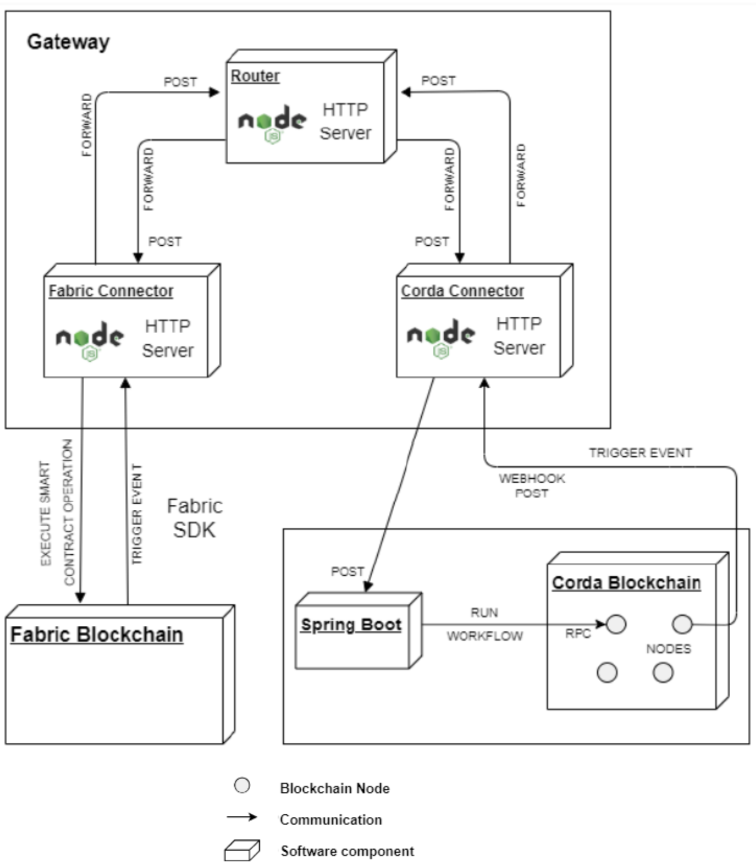

Figure 11: Prototype architecture [19]

The Router component is implemented using Node.js and follows the specification defined in Section 4.5. It has a static blockchain registry implemented as a configuration file (similar to the one presented in Listing 1), that enables the routing of messages to the Connectors. The Router inspects messages and redirect them (without modifications) to the target blockchain specified in their payload.

The Hyperledger Fabric connector is implemented using Node.js and Hyperleder Fabric SDK. The SDK is used to listen to the DLT events and to send messages to a specific smart contract on the DLT. Listing 2 presents the code of the connector that listens events and generates a message for the router component. On the other hand, Listing 3 depicts the code of the connector that receives the Router's messages and invokes the social security smart contract.

The Corda connector is also implemented with Node.js. As Corda does not provide an SDK, the prototype uses the Spring Boot[9] server provided by Corda with the purpose of interacting with this platform. This server provides HTTP endpoints that listen to messages, process them and send them to Corda through RPC messages. To receive Corda events, the connector exposes a webhook that Corda Flows invokes. Listing 5 depicts the code of the connector that receives the messages from the Spring Boot component and sends them to the Router component, while Listing 5 presents how the connector manages the Router messages and sends them to Corda.

To conclude, the development of the prototype allowed us to confirm it was technically feasible to develop a gateway-based interoperability solution following the design presented in Section 4. In particular, to enable interoperability between two DLT: Hyperledger Fabric and Corda. It also confirmed that interoperability can be enabled by using built-in features of both DLT, not requiring new features nor modifications of their source code.

[9]https://spring.io/projects/spring-boot

```
{  "blockchains": {
    "fabric": {
        "connectorHost": "localhost",
        "connectorPort": "3001"},
    "corda": {
        "connectorHost": "localhost",
        "connectorPort": "3002"}
  }
}
```

Listing 1: Router blockchain registry

```
async listenBlockchainEvents() {
    // listen events on fabric blockchain
    const network = await this.fabricGateway.getNetwork(
            config.blockchains.fabric.fabricChannel);
    const contract = network.getContract(
            config.blockchains.fabric.fabricContract);
    const remoteEvent = config.blockchains.fabric.fabricRemoteRequestEvent;

    await contract.addContractListener('listener', remoteEvent,
        this.eventReceivedFromFabric.bind(this));
}

eventReceivedFromFabric(err, event, blockNumber, transactionId, status) {
    this.printLine('Event received from fabric blockchain');

    if (err) {
        console.error(err);
    } else if (status == 'VALID') {
        const eventData = this.getDataFromFabricEvent(event);
        console.log(eventData);

        this.sendEventToRouter(eventData);
    }
}

sendEventToRouter(event) {
    const options = {
        host: config.router.endpoint,
        path: '/',
        port: config.router.port,
        method: 'POST'
    }

    this.callendpoint(options, JSON.stringify(event))
        .then(() => {
            console.log('\nEvent sent to router properly');
        })
        .catch(err => {
            console.error('\nAn error occurred while trying to send the event to the router');
            console.error(err);
        });
}
```

Listing 2: Hyperledger Fabric connector listening to events and sending them to Router component.

```
eventReceivedFromRouter(requestBody, response) {
    let event = JSON.parse(requestBody);
    let resp = {
        success: true
    }

    response.writeHead(200, {"Content-Type": "application/json"});
    response.write(JSON.stringify(resp));
    response.end();
    this.sendEventToFabricBlockchain(event);
}

async sendEventToFabricBlockchain(event) {
    // notifies fabric blockchain about the received event
    const network = await this.fabricGateway.getNetwork(config.blockchains.fabric.fabricChannel);
    const contract = network.getContract(event.targetContract);
    const receiveEvtOper = config.blockchains.fabric.fabricReceiveEventOperation;

    await contract.submitTransaction(receiveEvtOper, JSON.stringify(event.data));
}
```

Listing 3: Hyperledger Fabric connector listening to messages from the Router and sending them to the DLT.

```
eventReceivedFromCorda(event) {
    console.log(event);
    this.sendEventToRouter(event);
}

async sendEventToRouter(event) {
    try {
        const response = await axios.post(config.router.endpoint, event);
        console.log('\nEvent sent to router properly');
    } catch(error) {
        console.error('\nAn error occurred while trying to send the event to the router\n');
    }
}
```

Listing 4: Corda connector listening to messages from the Spring Boot component and sends them to the Router.

```
eventReceivedFromRouter(event) {
    this.printLine('Event received from router');
    console.log(event);
    this.sendEventToCorda(event);
}

async sendEventToCorda(event) {
    try {
        const response = await axios.post(config.blockchains.corda.cordaEndpoint, event);
    } catch(error) {
        console.error('\nAn error occurred while trying to send the event to Corda\n', error);
    }
}
```

Listing 5: Corda connector listening to messages from the Router and sends them to the Spring Boot component.

5.2 Development of case scenario

This section describes how the proposed solution was assessed through an illustrate scenario such as the social security scenario presented in Section 3. The objective of this assessment was to show the functional behaviour of the gateway and the technical details for its implementation.

The development of the case scenario involved the implementation of smart contracts on each of the DLT, and perform requests/response messages with the exchanged information.

5.2.1 Smart contracts

Listing 6 shows the smart contract that was implemented in Hyperledger Fabric to send events to the connector.

```
async notifyInformationExchange(ctx, informationExchange) {
    // notifies corda blockchain about an information exchange between a country from fabric blockchain
        and another one from corda
    let remoteEvent = new RemoteEvent()

    informationExchange.sourceBlockchain = 'fabric'
    informationExchange.sourceContract =
        'socialSecurityExchange'

    remoteEvent.setTargetBlockchain('corda')
    remoteEvent.setTargetContract('SocialSecurityExchange')
    remoteEvent.setEventData(informationExchange)

    this.sendRemoteEvent(ctx, remoteEvent)
}
```

Listing 6: Hyperledger Fabric smart contract: send event.

Regarding Corda, RemoteFlow and SSFlow were used for integration. RemoteFlow was used to process incoming messages to Corda and SSFlow was used to process cross-chain transactions initiated in Corda. Listing 7 shows how Corda SSFlow notifies the transaction (sends an event) after it is confirmed to the Corda Connector.

5.2.2 Request / Response of information

Listing 8 and Table 3 present an example of a query request message sent from the Hyperledeger Fabric connector to the Router. This same message is redirected from the Router to the Corda connector. The destination of this query request message is specified on the target targetBlockchain and targetContract fields. In this example, is the SocialSecurityExchange smart contract on the Corda blockchain platform.

```
// Signing the transaction.
SignedTransaction signedTx = serviceHub.signInitialTransaction(txBuilder);

try {
    getLogger().info("Forwarding transaction to peers");
    subFlow(new FinalityFlow(signedTx, sessions));
    getLogger().info("Transaction signed by all peers");
} catch (FlowException e) {
    //Process exception
}

getLogger().info("Triggering cross-chain event");
RemoteSSTxService service = serviceHub.cordaService(RemoteSSTxService.class);
service.notifyRemoteTx(outputState, signedTx);
```

Listing 7: Corda SSFlow notifying the confirmed transaction.

```
{
    "targetBlockchain": "corda",
    "targetContract": "SocialSecurityExchange",
        "data": {
        "requestID": "nj23r23js",
        "senderInstitution": 'ESP",
        "receiverInstitution": "URU",
        "requestType": "WorkedPeriods",
        "requestDate": "Sun Jun 27 2021 14:51:52 GMT+0000 (UTC)",
        "expectedReplyDate": "15/07/2021",
        "status": "pending",
        "messageType": "request",
        "packageHash": "323zczAI",
        "sourceBlockchain": "fabric",
        "sourceContract": "socialSecurityExchange"
    }
}
```

Listing 8: Query operation: request message

Table 3: Query request message fields

Field	Description
requestID	ID of the request message.
senderInstitution	It is the country that makes the request.
receiverInstitution	It is the destination of the request.
requestType	Specifies the type of the message request.
requestDate	Specifies the date the request is made.
expectedReplyDate	Specifies the expected response due date.
messageType	Specifies if the message is a request or response.
packageHash	Specifies the hash related to the off-chain request made by the countries.
sourceBlockchain	Specifies the source blockchain that made the request.
sourceContract	Specifies the source smart contract name on the source blockchain.

In addition, Listing 9 and Table 4 present an example of response message of the query operation.

```
{
    "targetBlockchain": "corda",
    "targetContract": "SocialSecurityExchange",
    "data": {
        "messageType": "response",
        "requestID": "nj23r23js",
        "packageHash": "sd923CSACA",
        "responseDate": "Sun Jun 27 2021 15:13:02 GMT+0000 (UTC)",
        "sourceBlockchain": "fabric",
        "sourceContract": "socialSecurityExchange"
    }
}
```

Listing 9: Query operation: response message

Table 4: Query response message fields

Field	Description
requestID	This is the ID of the request message that this response is correlated to.
packageHash	Specifies the hash related to the off-chain response made by the country.
responseDate	Specifies the date when the response is sent.
messageType	Specifies if the message is a request or response.

5.3 Interoperability analysis

The proposed solution was analysed using the LCIM model presented in Section 2.5. The goal of this assessment was to determine the level of interoperability achieved by the proposal.

After the analysis, this work found that the extension proposed by Pillai et al. [35] did not consider distributed ledger technologies such as Hyperledger Fabric or Corda. For example, these blockchains do not share the same ledger structure and could not achieve further interoperability than level 2. However, the third party involved in this level (that enables the exchange of messages) can provide transformation capabilities (as proposed in this work), to solve this heterogeneity. Considering this, this work proposes to relax the definition of level 2 and provide a new definition to consider distributed ledgers. At Level 2, distributed ledger technologies may share the same data structure or there exists a function f_1 that allows to transform a transaction from the source distributed ledger format to target distributed ledger format. On the other hand, f^{-1} allows to transform a transaction from the target distributed ledger format to the source distributed ledger format.

Considering the aforementioned modification to the model, the proposed solution achieved level 4 of interoperability. The gateway fulfilled level 1, as DLT emit events to send messages between them and they provide smart contracts as interfaces to listen to them. Hyperledger Fabric uses native protocols to emit events and send messages to smart contracts, while Corda relies on http calls for these tasks. Level 2 is satisfied besides Hyperledger Fabric and Corda have different syntax for blocks and transactions. The gateway provides the transformation capabilities to adapt the message heterogeneity's between them. The reader may note that the gateway did not provide proofs nor a verification process for incoming transactions and solve the data access and acceptance problem. However, as the gateway is a trusted software and considered for private or consortium scenarios, it is considered acceptable. Level 3 is achieved as smart contracts on each DLT provide the semantic capabilities to understand the meaning of the exchanged messages. Level 4 is reached as the proposed gateway allows a source DLT to invoke smart contracts (functions) on a target DLT. Level 5 is not achieved as DLT are not aware about state changes between them.

To conclude, the proposed gateway achieved level 4 of our modified version of Pillai et al. LCIM model. These means, our proposal enabled Hyperledger Fabric to execute Corda functions and understand the semantics of the exchanged data and vice-versa.

5.4 Interoperability design decision analysis

This section describes the application of the CBIDD assessment framework presented in Section 2.6 to the proposed gateway. As it was proposed after the development of this work, it could not be used at the initial stages. However, it served as a postmortem assessment to validate the suitability of the proposed solution to the applied scenario. The goal of this assessment was to verify that the gateway-based solution served as a suitable approach for the social security scenario presented in Section 3. In particular, we considered that our proposal was validated as long as we could arrive to a high-level design of an alike interoperability solution, by following the stages of the CBIDD framework.

The first stage of the framework requires the identification and selection of the value type of the scenario. This value type is data, as two social security organisations needs to exchange social security data between them. The second stage requires the identification of the integration goal which, in this case, is data exchange. The third stage requires identification of the integration approach. The proposed solution follows a centralised approach, as there is only one entity that controls the integration process. The gateway is responsible for listening to events on the source blockchain and invoke smart contracts in the target blockchain. Since social security organisations trust each other, it is suitable to choose a centralised approach. The fourth step requires the selection of the integration mode. The proposed solution uses the bridge mode proposed by the assessment framework as there are gateway nodes that monitor the blockchains and perform computations on them based on this monitoring. Finally, the fifth step requires the selection of the integration protocol, where a custom protocol is proposed and used.

To conclude, it was possible to follow all the stages proposed by the CBIDD framework. This result allowed us to conclude that our proposal can be selected as a suitable interoperability solution for the proposed social security scenario.

5.5 Discussion

This section presented the implementation and assessment of the proposal. A prototype was developed to evaluate its technical feasibility. An illustrative case scenario showed its functional behaviour. The application of the LCIM model determined the level of interoperability provided by the proposal. The CBIDD framework allowed to verify the suitability of the proposal applied to the social security scenario. These validations constitute a step forward towards the assessment of the gateway-based interoperability solution and its evolution.

The prototype constitutes a reference implementation for the proposal, providing the necessary information to understand and evolve the solution. The prototype confirms the technical feasibility to develop a platform-to-platform gateway-based interoperability solution between two DLT. In particular, between Hyperledger Fabric and Corda. Finally, it confirms that the solution can be implemented using built-in features of both DLT, not requiring new features nor modifications on their source code.

The application of the proposal on a social security scenario enabled its functional validation and a progress in the technical feasibility evaluation. Regarding the functional validation, it was possible to create requests and responses of information from both consortia. With respect to the technical feasibility, we confirmed that the solution can be applied in a specific scenario, only requiring the development of smart contracts to send events and receive requests of information.

The LCIM evaluation framework allowed us to characterise our gateway solution according to the level of interoperability it may enable between these two DLT. The gateway enabled a high level of interoperability, reaching level 4 of the LCIM model. A higher level of interoperability (level 5) requires sharing the state model between blockchains, that our gateway-based solution does not support yet. On the other hand, the CBIDD framework allowed us to make a postmortem assessment of the interoperability approach, showing the suitability of the proposal.

The experimentation has some limitations as it does not consider identity management, data privacy nor authorisation management, which are main properties of DLT, as presented in Section 2.1. Furthermore, the proposed solution assumes trust in the gateway, which is tolerable on a consortium or private interoperability scenario. Furthermore, our gateway solution is not a reusable solution and must be generated for each pair of DLT within a specific scenario. Finally, the LCIM model is suitable for blockchain platforms, but it could not be applied directly on distributed ledgers technology. We adapted the model to be applied for Corda and Hyperledger Fabric, but more analysis needs to be performed to consider other DLT and provide a more exhaustive evaluation.

To sum up, we confirm that it is possible to achieve platform-to-platform interoperability between Hyperledger Fabric and Corda blockchains using a gateway-based interoperability solution on a specific case scenario, without considering identity management, data privacy and authorisation management properties of these DLT.

6 Related work

Our previous work performed a literature review that served as the basis to this section [9].

Hardono et al. [17] presented a theoretical blockchain interoperability reference architecture based on the internet architecture design. The work proposed interoperability between blockchain platforms through blockchain gateways and trust domains. It defines intra-domain and inter-domain nodes, being the first ones required for the proper operation of the blockchain (e.g. full nodes, miners), while the second ones enable cross-chain transactions. Our work is inspired by this architecture and goes a step forward, providing a practical implementation with a detailed design and source code for DLT interoperability.

Abebe et al. [3] proposed a trusted relay solution to provide interoperability between two homogeneous DLT implemented with Hyperledger Fabric. The solution is based on system smart contracts (i.e. Configuration Management, Data Exposure and Data Verification) for data transfer cross-chain transactions. DLT discovery is accomplished by the Configuration Management smart contract, while the Data Exposure smart contract enforces authorisation policies and defines which data can be queried on the ledger and by whom. The Data Verification smart contract is used to validate incoming data from an external source, by verifying the source DLT consensus protocol proofs. End-to-end data encryption is applied in order to avoid the trusted relay to have access to confidential data, which is decrypted on arrival on the target DLT. Business applications use the trusted relays to query data of other DLT and save them after proofs validation. The main difference with our work is that our proposal provides a platform-to-platform blockchain interoperability solution, instead of application-to-platform interoperability solution. The source of our cross-chain transactions are smart contracts inside a DLT and not external business applications. Additionally, our proposal provides a solution for heterogeneous DLT (e.g. Hyperledger Fabric and Corda), not only homogeneous permissioned interoperability. Our work can benefit from this paper with respect to the approaches for DLT discovery, authorisation and verification of cross-chain transactions.

Abdullah et al. [42] proposed the Chain-Net framework inspired on the internet to enable interoperability across heterogeneous blockchains. The proposed framework was based on gateway modules that are considered part of the blockchain architecture. The authors presented a prototype based on Ethereum Virtual Machines blockchains that served as proof of concept. The prototype was analysed regarding security, scalability and costs. Our approach shares similarities with this work, as it is based on gateways to enable DLT interoperability, although, our work follows a different design and considers gateways as an external component of the DLT. Despite the Chain-Net framework was proposed for heterogeneous blockchains, permissioned blockchain properties are not discussed nor considered (e.g. identity, authorisation). Our approach explicitly left them out of scope as they required further analysis. We consider our work can be improved by considering security and scalability analysis as the ones presented in this work.

Hyperledger Cactus is a framework that enables business applications to achieve interoperability with heterogeneous DLT [15]. Through its extensible plugin architecture, Cactus enables the integration of different DLT and nowadays supports Hyperledger Fabric, Hyperledger Besu, Corda and Quorum. Interoperability validators enable Cactus to validate cross-chain transactions and its business logic plugin enables developers to add custom business logic to develop coordinated operations. Cactus is an advanced trusted relay solution for application-to- platform interoperability. We follow a different approach trying to achieve platform-to-platform interoperability.

Scheid et al. proposed Bifröst [16] which provides an API that enables business applications to communicate with heterogeneous DLT on a standardised way. Its functionality is focused on connectivity and integration, leaving out of scope cross-chain transaction verification and identity management, among others. Bifröst supports a wide variety of DLT: EOS, Corda, Hyperledger Fabric, Stellar, Bitcoin, Ethereum, IOTA and Multichain. Compared to our proposal, we understand Bifröst is an application-to-platform interoperability solution.

Falazi et al work on SCIP [43] defined a message specification and provided a prototype implemented over Hyperledger Fabric and Ethereum, supporting connectivity and integration of both DLT. Cross-chain transaction verification, identity and authorisation is not tackled and the work supported the following operation types: invocation, subscription, unsubscription, query and callback. Compared to our proposal, we understand that SCIP is an application-to-platform interoperability solution.

Cosmos [29] and Polkadot [30] are two blockchain of blockchains interoperability solutions that follow similar approaches. In Cosmos, each blockchain is called a Cosmos Zone and the first connected blockchain is called Cosmos Hub. The Hub connects to other zones using the IBC protocol and tracks the different token types and records the number of tokens existing in each zone. Cosmos uses a DPoS consensus protocol. On the other hand, in Polkadot, blockchains are called parachains. Each parachain is connected to each other by the Polkadot Relay Chain: a chain that is responsible for the security of the network, for applying the consensus protocol and for blockchain interoperability. Bridges enable external blockchain networks (e.g. Bitcoin, Ethereum) to be connected to Polkadot and enable communication with the parachains. Both approaches provide a platform-to-platform interoperability solution, but follow a different approach than our work. Blockchain of blockchains solutions require changes on the blockchain source code. Our work follows a non-invasive approach and tries to achieve the same results with a more simple solution.

Polygon PoS Bridge [10] is an interoperability solution based on sidechain/relays, that allows to move assets from Ethereum to Polygon blockchain and vice-versa. Polygon PoS Bridge considers the atomic and consistency properties of ACID to move assets from one blockchain to the other. Besides this, Polygon PoS Bridge is a specific interoperability solution for asset transfer mode and does not support other DLT. Our proposal is specific for DLT interoperability using data transfer mode.

Finally, Hardjono [18] extends the work of Hardjono et al. [17] and presented a reference architecture for a gateway interoperability solution. He defined the design principles and properties required for cross-chain transaction support and proposes an atomic unidirectional gateway protocol. It also discussed the challenges that this type of solution needs to solve regarding gateway identity, gateway crash recovery, gateway discoverability as well as extinguish and regeneration of assets. Unfortunately, this work lacks of a prototype that enable the validation of the approach. Our work may be considered as a simplified version of this approach, as a first step and basis to extend and apply it.

7 Conclusions and future work

In recent years, organisations started building consortium blockchains to improve their business processes. They consider blockchain as a suitable technology that guarantees immutability, confidentiality, and availability of a common data source to every party in the consortium. Although there have been significant developments improving blockchain platforms, they still operate independently from each other and cannot be integrated easily, leading to information silos.

Interoperability is not a built-in feature of blockchain platforms. Nevertheless, some initial blockchain interoperability solutions have been developed in cryptocurrency scenarios. However, they do not directly apply to every distributed ledger as they do not share the same properties on ledger treatment. Although some work has been done, DLT interoperability still needs further research.

This paper proposes a gateway-based approach to address the challenges on DLT interoperability. Existing related work focus on achieving interoperability between business applications and multiple blockchain platforms (application-to-platform interoperability) instead of achieving interoperability between two or more DLT (platform-to-platform interoperability). Some theoretical approaches are defined but, to the best of our knowledge, none of them provides a detailed design and a reference implementation as our work.

Addressing design issues constitutes one of the main contributions of this paper, which provides a detailed design of the gateway as well as the specification of the interactions and interfaces of components required to achieve platform-to-platform interoperability. We also developed a data transfer prototype in the social security area to validate the approach, achieving interoperability between two DLT: Corda and Hyperledger Fabric. The proposal was evaluated using two interoperability frameworks: LCIM and CBIDD. The evaluation through the LCIM model resulted in a partially achievement of level four. These means, our proposal enabled Hyperledger Fabric to execute Corda functions and understand the semantics of the exchanged data and vice-versa.The CBIDD evaluation confirmed that that our proposal can be selected as a suitable interoperability solution for the proposed social security scenario.

The gateway solution presented in this paper is still preliminary and supports certain degree of interoperability based on the LCIM model evaluation. However, verification and validation of DLT transactions is still pending. This preliminary results constitutes a step forward to implement platform-based interoperability using DLT.

Future work includes its evolution to complete the design principles not covered in this work: preserving blockchain properties and decentralisation. In particular, we plan to address transaction validations, data privacy, identity and authorisation management. Data consistency between cross-blockchain transactions is another future work. Furthermore, model driven development may be applied for the generation of the gateway for two pair of given blockchains.

Acknowledgment

Guzmán Llambías was supported by Pyxis. The research that gives rise to the results presented in this publication received funding from the Agencia Nacional de Investigación e Innovación under the code POS_NAC_2022_4_174476.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009, (accessed Aug. 2023). [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] V. Buterin, “Ethereum: A next-generation smart contract and decentralized application platform,” 2014, (accessed Aug. 2022). [Online]. Available: https://ethereum.org/en/whitepaper/

[3] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola, “Enabling enterprise blockchain interoperability with trusted data transfer (industry track),” in *Proceedings of the 20th International Middleware Conference Industrial Track*, Davis, CA, USA, Dec. 2019, p. 29–35. [Online]. Available: https://doi.org/10.1145/3366626.3368129

[4] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, “Hyperledger fabric: A distributed operating system for permissioned blockchains,” in *Proc. 13th EuroSys Conf.*, Porto, Portugal, April 2018, pp. 1–15. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[5] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: An introduction," 2016, (accessed Aug. 2022). [Online]. Available: https://docs.r3.com/en/pdf/corda-introductory-whitepaper.pdf

[6] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–41, Oct. 2021. [Online]. Available: https://doi.org/10.1145/3471140

[7] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, Jul. 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2926880

[8] T. Koens and E. Poll, "Assessing interoperability solutions for distributed ledgers," *Pervasive and Mobile Computing*, vol. 59, p. 101079, Oct. 2019. [Online]. Available: https://doi.org/10.1016/j.pmcj.2019.101079

[9] G. Llambías, L. González, and R. Ruggia, "Blockchain interoperability: a feature-based classification framework and challenges ahead," *CLEI Electron. J.*, vol. 25, no. 3, Mar. 2023. [Online]. Available: https://doi.org/10.19153/cleiej.25.3.4

[10] Polygon Team, "Introduction to polygon pos," (accessed Aug. 2023). [Online]. Available: https://wiki.polygon.technology/docs/develop/getting-started

[11] H. Tam Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, and M. Mohania, "Internet of blockchains: Techniques and challenges ahead," in *2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, Halifax, NS, Canada, Aug. 2018, pp. 1574–1581. [Online]. Available: https://doi.org/10.1109/Cybermatics_2018.2018.00264

[12] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do you need a distributed ledger technology interoperability solution?" *Distrib. Ledger Technol.*, vol. 2, no. 1, pp. 1–37, Mar. 2023. [Online]. Available: https://doi.org/10.1145/3564532

[13] HSBC Bank, "The interoperability of cbdcs across networks and currencies," Dic. 2022, (accessed Aug. 2023). [Online]. Available: https://www.gbm.hsbc.com/en-gb/feed/innovation/the-interoperability-of-cbdcs-across-networks-and-currencies

[14] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "Hermes: Fault-tolerant middleware for blockchain interoperability," *Future Generation Computer Systems*, vol. 129, pp. 236–251, Apr. 2022. [Online]. Available: https://doi.org/10.1016/j.future.2021.11.004

[15] H. Montgomery *et al.*, "Hyperledger cactus whitepaper," 2022, (accessed Aug. 2023). [Online]. Available: https://github.com/hyperledger/cactus/blob/main/whitepaper/whitepaper.md

[16] E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller, "Bifröst: a modular blockchain interoperability api," in *IEEE 44th Conference Local Computer Networks*, Osnabrueck, Germany, Feb. 2019, pp. 332–339. [Online]. Available: https://doi.org/10.1109/LCN44214.2019.8990860

[17] T. Hardjono, A. Lipton, and A. Pentland, "Toward an interoperability architecture for blockchain autonomous systems," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1298–1309, Nov. 2020. [Online]. Available: https://doi.org/10.1109/TEM.2019.2920154

[18] T. Hardjono, "Blockchain gateways, bridges and delegated hash-locks," 2021, (accessed Aug. 2023). [Online]. Available: https://doi.org/10.48550/arXiv.2102.03933

[19] B. Bradach, J. Nogueira, G. Llambías, L. González, and R. Ruggia, "A gateway-based interoperability solution for permissioned blockchains," in *2022 XVLIII Latin American Computer Conference (CLEI)*, Armenia, Colombia, Oct. 2022, pp. 1–10. [Online]. Available: https://doi.org/10.1109/CLEI56649.2022.9959907

[20] X. Xu, I. Weber, and M. Staples, *Architecture for blockchain applications.* New York, NY, USA: Springer Cham., 2019.

[21] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, pp. 352–375, Jan. 2018. [Online]. Available: https://doi.org/10.1504/IJWGS.2018.095647

[22] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX annual technical conference (USENIX ATC 14)*, Philadelphia, PA, USA, Jun. 2014, pp. 305–319.

[23] "Iso 22739:2020. blockchain and distributed ledger technologies — vocabulary," (accessed Aug. 2023). [Online]. Available: https://www.iso.org/standard/73771.html

[24] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, "Cross-blockchain technology: Integration framework and security assumptions," *IEEE Access*, vol. 10, pp. 41 239–41 259, Apr. 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3167172

[25] J. Nick, A. Poelstra, and G. Sanders, "Liquid: A bitcoin sidechain," May 2020, (accessed Aug. 2023). [Online]. Available: https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf

[26] BTC Relay, "A bridge between the bitcoin blockchain & ethereum smart contracts," (accessed Aug. 2023). [Online]. Available: http://btcrelay.org/

[27] Bitcoin wiki, "Timelock," (accessed Aug. 2023). [Online]. Available: https://en.bitcoin.it/wiki/Timelock

[28] Bitcoin Wiki, "Hashlock," (accessed Aug. 2023). [Online]. Available: https://en.bitcoin.it/wiki/Hashlock

[29] J. Kwon and E. Buchman, "Cosmos whitepaper," 2022, (accessed Aug. 2023). [Online]. Available: https://v1.cosmos.network/resources/whitepaper

[30] Polkadot, "An introduction to polkadot," 2020, (accessed Aug. 2023). [Online]. Available: https://polkadot.network/Polkadot-lightpaper.pdf

[31] A. Tolk and J. A. Muguira, "The levels of conceptual interoperability model," in *Proceedings of the 2003 fall simulation interoperability workshop*, vol. 7, Orlando, FL, USA, Sep. 2003, pp. 1–11.

[32] M. Robkin, S. Weininger, B. Preciado, and J. Goldman, "Levels of conceptual interoperability model for healthcare framework for safe medical device interoperability," in *2015 IEEE Symposium on Product Compliance Engineering (ISPCE)*, Chicago, IL, USA, May 2015, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ISPCE.2015.7138703

[33] E. Wassermann and A. Fay, "Interoperability rules for heterogenous multi-agent systems: Levels of conceptual interoperability model applied for multi-agent systems," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, Emden, Germany, Jul. 2017, pp. 89–95. [Online]. Available: https://doi.org/10.1109/INDIN.2017.8104752

[34] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple object access protocol (soap) 1.1," May 2000, (accessed Aug. 2023). [Online]. Available: https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[35] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, "Level of conceptual interoperability model for blockchain based systems," in *2022 IEEE Crosschain Workshop (ICBC-CROSS)*, Shanghai, China, May 2022, pp. 1–7. [Online]. Available: 10.1109/ICBC-CROSS54895.2022.9793328

[36] Ethereum Classic, "Ethereum classic," (accessed Aug. 2023). [Online]. Available: https://ethereumclassic.org/

[37] C. Goes, "The interblockchain communication protocol: An overview," Jun. 2020, (accessed Aug. 2023). [Online]. Available: https://doi.org/10.48550/arXiv.2006.15918

[38] International Social Security Association (ISSA), "Issa webinar: Blockchain in social security," Nov. 2020, (accessed Aug. 2023). [Online]. Available: https://ww1.issa.int/webinar-2020-11-18

[39] Banco de Previsión Social, "Convenios internacionales," Jun. 1999, (accessed Aug. 2023). [Online]. Available: https://www.bps.gub.uy/12610/convenios-internacionales.html?idFiltro1=12643

[40] H. Jin, X. Dai, and J. Xiao, "Towards a novel architecture for enabling interoperability amongst multiple blockchains," in *2018 IEEE 38th Int. Conf. Dist. Comp. Sys.*, Vienna, Austria, Jul. 2018, pp. 1203–1211. [Online]. Available: https://doi.org/10.1109/ICDCS.2018.00120

[41] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Boston, MA, USA: Addison-Wesley Professional, 2003.

[42] S. Abdullah, J. Arshad, and M. Alsadi, "Chain-net: An internet-inspired framework for interoperable blockchains," *Distrib. Ledger Technol.*, vol. 1, no. 2, Dec. 2022. [Online]. Available: https://doi.org/10.1145/3554761

[43] G. Falazi, U. Breitenbücher, F. Daniel, A. Lamparelli, F. Leymann, and V. Yussupov, "Smart contract invocation protocol (scip): A protocol for the uniform integration of heterogeneous blockchain smart contracts," in *Advanced Information Systems Engineering*, Grenoble, France, Jun. 2020, pp. 134–149. [Online]. Available: https://doi.org/10.1007/978-3-030-49435-3_9