Trust Development for Blockchain Interoperability Using Self-sovereign Identity Integration

***Abstract*—Despite advances in developing various blockchain platforms and services, disparate blockchain networks have no direct connection, causing fragmentation and data and asset in silos. Blockchain interoperability is a solution that allows communication between different blockchain platforms to exchange data and assets. This study proposes a blockchain interoperability solution based on publish-subscribe architecture to share data between various blockchain platforms. We extend the solution by providing decentralized identifiers and verifiable credentials to the users sharing and accessing data through integration with Hyperledger Indy, Aries, and Ursa stack projects, blockchain services that offer decentralized identity. We also designed a testbed comprising multiple permissioned blockchain platforms integrated with decentralized identity services to evaluate the proposed system's security and integrity.**

***Index Terms*—Blockchain, Interoperability, Publish-subscribe, Self-sovereign identity, Hyperledger**

I. Introduction

Blockchain features such as immutable and transparent records, decentralization, traceability, and security have attracted significant attention from both industry and academia. Although a cryptocurrency application served as the first use of this technology, subsequent developments in blockchain platforms brought about smart contracts, which are expressed in various Turing-complete languages. Blockchain platforms can now support the complex structure of various corporate operations thanks to these automated programs, and numerous applications may now be built on top of the blockchain. As a result, several blockchain platforms have been created with unique services to offer and a specific goal to address. The creation of various incompatible blockchain platforms that cannot interact and share data with one another has caused fragmentation and blockchain-based solutions that operate in silos.

The capability to transfer assets and data from one blockchain to another is made possible by blockchain interoperability, a solution that bridges the gaps between various blockchain technologies. Interoperable blockchain infrastructure is a collection of independent blockchain networks, each of which can be represented as a distributed ledger of data. Data can be operated between these heterogeneous, unconnected blockchain networks, and the data ledger can be accessed by verified external data [1].

In Self-Sovereign Identity (SSI), the subjects control and own their digital identities and are the holders of their own identities. Identity providers issue and digitally sign the identities with the issuer's private key, making them tamper-proof and cryptographically verifiable [2]. In blockchain-based SSI, the issuer's public key materials are also stored in a blockchain ledger (tamper-resistant and highly available registry called verifiable data registry (VDR)), which is known to the verifier [3]. For example, in Hyperledger Indy [1], identity records that include Public Keys, Service Endpoints, Credential Schemas, and Credential Definitions are stored on the ledger. Hyperledger Indy, Aries, and Ursa are Hyperledger stack projects for decentralized identity. Hyperledger Ursa [4] is a modular cryptographic library. Hyperledger Indy [5] is the implementation of decentralized identity based on distributed ledger technology (DLT) and provides an application interface layer for applications to interact with the Indy ledger. Hyperledger Aries [6] is the agent (client) for managing verifiable digital credentials through cryptographic protocols provided by Ursa.

In this paper, we have implemented a blockchain interoperability solution based on the publish/subscribe architecture that enables a record of the data being transferred between disparate blockchain networks [7]. This interoperability solution has three blockchain components: a broker blockchain, publisher blockchains and subscribers blockchains. The broker blockchain functions as a messaging broker and keeps track of topics provided by other blockchain networks. Publisher blockchains transfer data by sending messages to the topics, and the subscriber blockchains access the shared data topics and are notified immediately of any changes in the topic through a received publish request. We provide a trust layer by integrating the interoperable system with blockchain services that offer decentralized identity. We have used Hyperledger stack projects of Indy, Aries, and Ursa for decentralized identity.

This paper presents the following contributions:

- To the best of our knowledge, this is the first time that SSI is used for blockchain interoperability
- We designed and implemented a trusted and robust authentication mechanism for users who share data between distinct blockchain platforms through publish-subscribe architecture
- Our evaluation result indicates the validity and integrity of our proposed system

The rest of the paper is organized as follows. Section II reviews blockchain interoperability and digital identity studies. Section III discusses system architecture. Section IV provides implementation details. Section V presents evaluation results, and finally, the conclusion and future work are presented in section VI.

[1]https://www.hyperledger.org/use/hyperledger-indy

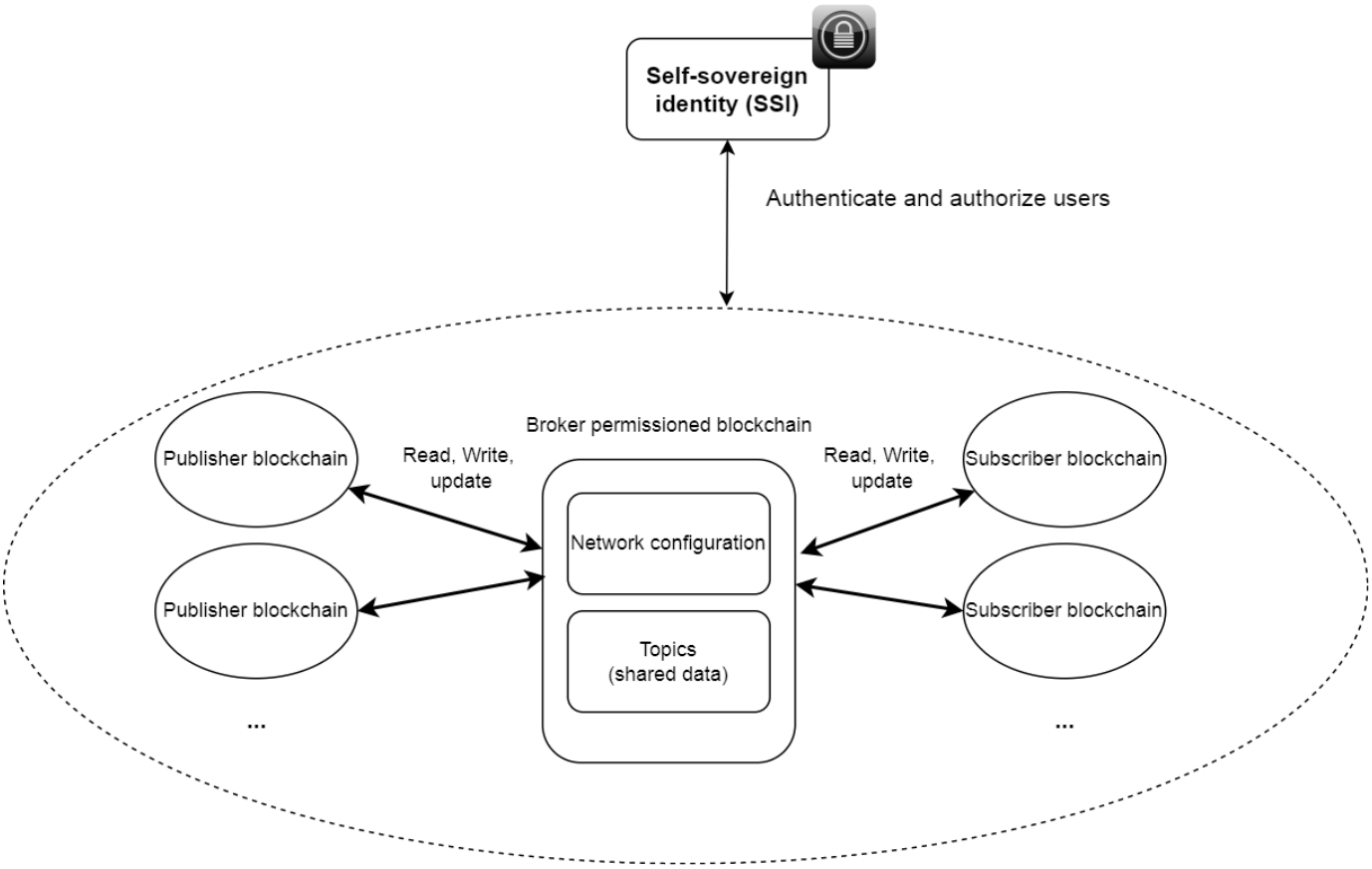

Fig. 1: publish-subscribe blockchain interoperability

II. Related works

The Internet of Blockchains is a network that connects homogeneous and heterogeneous blockchain systems, allowing them to communicate and facilitate cross-chain transactions by focusing on relationships between various blockchains [8].

Furthermore, network architecture and consensus techniques that combine and arrange blocks from several blockchains into ”meta-blocks” as well as research cross-blockchain transaction validation and administration, have all been explored in relation to blockchain interoperability [9]. The three primary categories for creating blockchain interoperability solutions are Blockchain of Blockchains (BoB), Public Connectors, and Hybrid solutions [10]. By offering reusable structures in several layers, such as data, network, consensus, and smart contracts, BoB creates specialized interoperable blockchains. Known platforms in this category include Polkadot [11] and Cosmos [12], which offer protocols for creating an interface for multiple state machines to connect and transmit messages and data. The public connector facilitates interoperability between public blockchains with a focus on cryptocurrency and digital asset trading [13], [14]. They employ different techniques to provide a secure and reliable interoperability solution, including sidechains, notary schemes, and hash locks [15].

Most of the proposed blockchain interoperability solutions rely on centralized trusted mediators, which does not align with the principle of decentralization [1]. In this study, we use a publish-subscribe architecture that uses a blockchain as a broker instead of a centralized third party to establish communication between different blockchain platforms [7], illustrated in the Figure 1. The broker blockchain acts as a messaging broker and keeps track of all the topics other blockchain networks provide. Publisher blockchains can send messages to the topics, and the subscriber blockchains are automatically notified immediately of any changes in the topic through a received publish request. Every component runs by a separate blockchain platform to support interoperability.

In addition, in the current blockchain interoperability solutions, there is no access control component that protects data transferred between multiple blockchain networks, so they mostly delegate it to the security model of participating blockchains while they might not be applicable out of the scope of their own network [1]. We use distributed digital identity known as a self-sovereign identity [16] to address this issue.

SSI is a new identity management scheme that removes the role of service providers as third parties and a centralized component for identification and verification of digital identities and offers an efficient solution as it prevents information duplication and allows users to control their personal identities

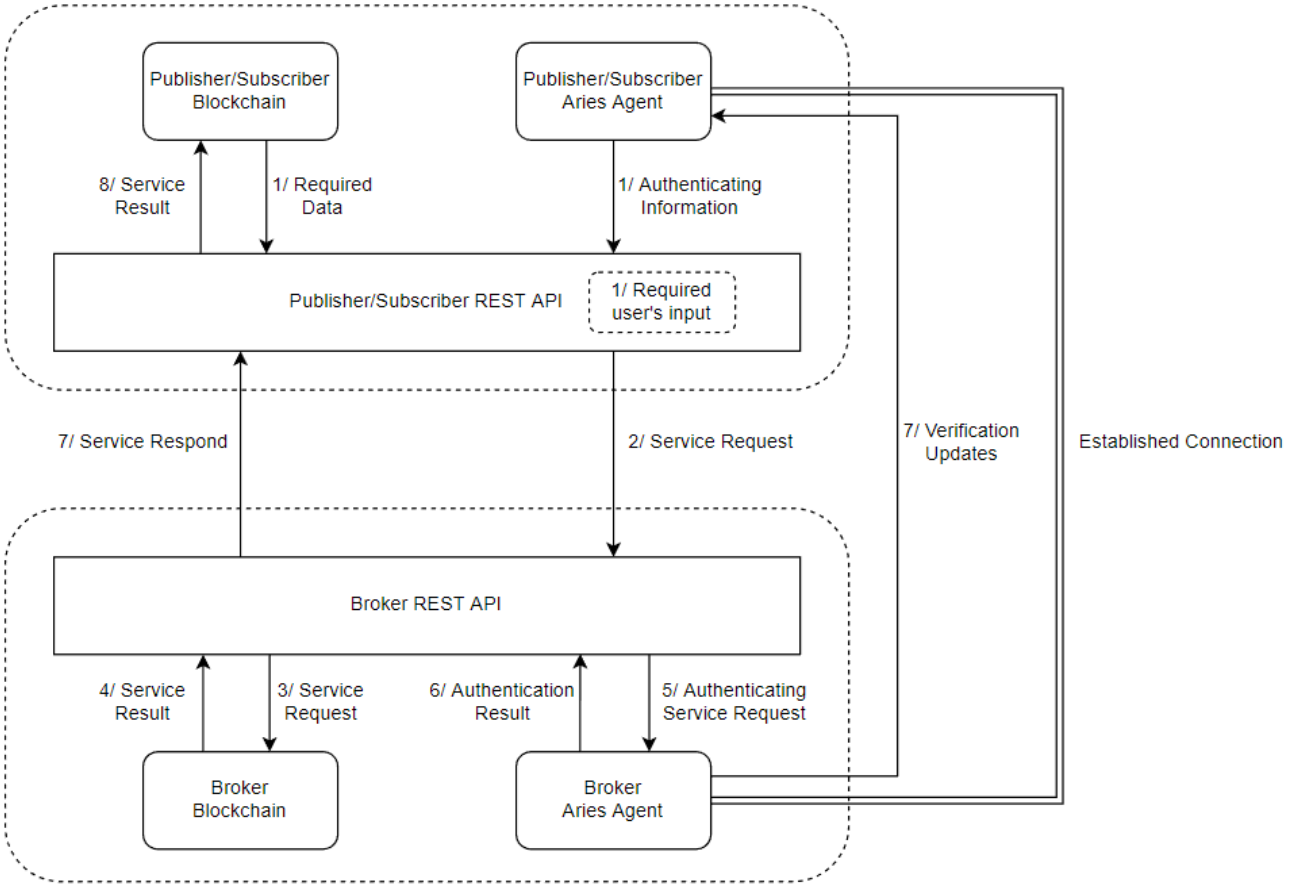

Fig. 2: System architecture

and their attributes without relying on centralized trusted authorities. Therefore, identity owners can store their identity data on their choice of storage, such as their local devices, and provide the required information to those who need it for validation purposes [17]. Blockchain is a promising infrastructure for developing SSI as it does not need a central authority to validate transactions and execute them as a distributed and autonomous system [18].

Self-sovereign identity has been investigated for various applications, such as Iot [19], [20] for protecting the identification of IoT devices and users as the owner of those devices and healthcare [21]–[24] for protecting patients' privacy.

III. System Architecture

Figure 2 presents the system architecture, including two main components, publisher/subscriber blockchains and the broker blockchain. The broker blockchain establishes communications network configuration between publisher and subscriber blockchains. The Aries agents are responsible for issuing and verifying credentials and authenticating requests.

The interaction of the proposed system is based on the REST API and the connection between Hyperledger Aries (HA) agents. REST API is the main tool for data transmitting where the pub/sub system can interact and use services from the broker system. Besides that, HA agents are used for authentication purposes. In order to request services from the broker system, the pub/sub system needs:

- Authentication information, which is obtained from its own HA agent.
- Required data for requesting service, which is obtained from its own blockchain or from the user's input.

After receiving the required data for requesting service, the pub/sub system sends them to the broker system through the REST API. The broker system then performs the requested service and authenticates the pub/sub system. After performing the requested service and authenticating the pub/sub system, the broker system will respond to the requested service and update the credential for the pub/sub system's Hyperledger Aries agent. After receiving the response from the broker system, the blockchain state on the publisher and subscriber side will the ledger state will be updated to reflect the verification update.

IV. Implementation

In this section, we explain the detail of the authentication process between a pub/sub agent and broker agent. Agents

use DID communications to talk to each other. REST APIs are made for each system to send or receive actual data objects between the pub/sub system and broker system. The authentication part can be divided into two categories :

1) Initial connection authentication: This authentication happens when the pub/sub agent tries to connect to the broker agent. It begins when a pub/sub agent sends a connection invitation to the broker system. A connection invitation is an url to connect the receiver agent back to the sender agent. After the connection is established, both agents can communicate with each other. Now, the broker agent sends a proof request for the credential back to the pub/sub agent to check if this is the first time connection with this agent. It then waits for a proof response.
   If the pub/sub agent has the requested credential, it will accept the proof request and send the requested credential back to the broker agent as a proof response. Otherwise, the pub/sub agent declines the proof request, and that is sent as a proof response.
   If the proof request is accepted, then the pub/sub agent is verified, and the pub/sub system is ready to request services from the broker system. Otherwise, when the proof request is declined, the broker system assumes it has a new connection and the broker agent issues a new credential to the pub/sub agent. Every record that are exchanged between the pub/sub agent and the broker agent will be stored on both systems' Indy ledger. The process of Initial connection authentication is illustrated in Figure 3.

2) Service request authentication: This part of the authentication will be used after the initial connection step is completed, and it is used for verification during the pub/sub system using service from the broker system. Whenever the pub/sub system requests a service from the broker system, the pub/sub agent needs to send two parameters to the broker agent:
   - Connection DID
   - Credential thread ID

   These parameters are used to distinguish one pub/sub system from another when they connect to the broker system. The Connection DID is a parameter of the Connection Record, which is generated and stored on the Indy ledger whenever one Aries agent connects with another Aries agent. The credential thread ID is a parameter of the Credential Record, which is generated and stored on the Indy ledger whenever a new credential is issued to an agent. If there are multiple credentials stored in an agent, the latest credential thread ID will be sent to the broker agent. The broker agent also has a copy of these records on its own Indy ledger.
   Depending on the requested service, the pub/sub agent's credential might need to be updated, and whenever it happens, the broker agent will issue a new credential to the pub/sub agent. The process of Service request authentication is shown in Figure 4.

V. Evaluation

In this section, we will discuss the evaluation of our project. We used the Google Cloud Platform to run multiple Virtual Machine (VM) instances and test our application and evaluate the validity and integrity of the system. To do so, we simulated different case scenarios. As a result, we created 7 instances, which can be described as follows:

- Broker instances: One instance hosted a broker system (Broker blockchain + Aries agent) and responded to requests sent by publisher and subscriber systems. Here, we will call it broker1.
- Publisher instances: Two instances hosted publisher systems (Publisher blockchain + Aries agent) and created and updated topics. Here, we will call them pub1 and pub2
- Subscriber instances: Four instances hosted subscriber systems(Subscriber blockchain + Aries agent) and queried subscribed topics. Here we will call them sub1, sub2, sub3, and sub4.

All the GCP instances had the following configurations :

- Machine type: E2-medium (2 vCPU, 4 GB memory)
- OS: Debian GNU/Linux (bullseye)
- Boot disk: Balanced persistent disk
- Disk size: 80 GB
- Installed applications: Indy SDK, Aries Javascript framework

A. Setup

We created a setup before requesting any service from the system, as shown in Table I.

B. Testing

We tested various scenarios applied to the above general setup. Table II presents publish to topic tested operations and table III presents subscribe to a topic and query a topic tested operations. As the results indicate, the credentials are issued and confirmed for authorized requests to access the shared topics, and all transactions are completed successfully. For unauthorized access requests also agent did not permit any publish, update or access on a topic.

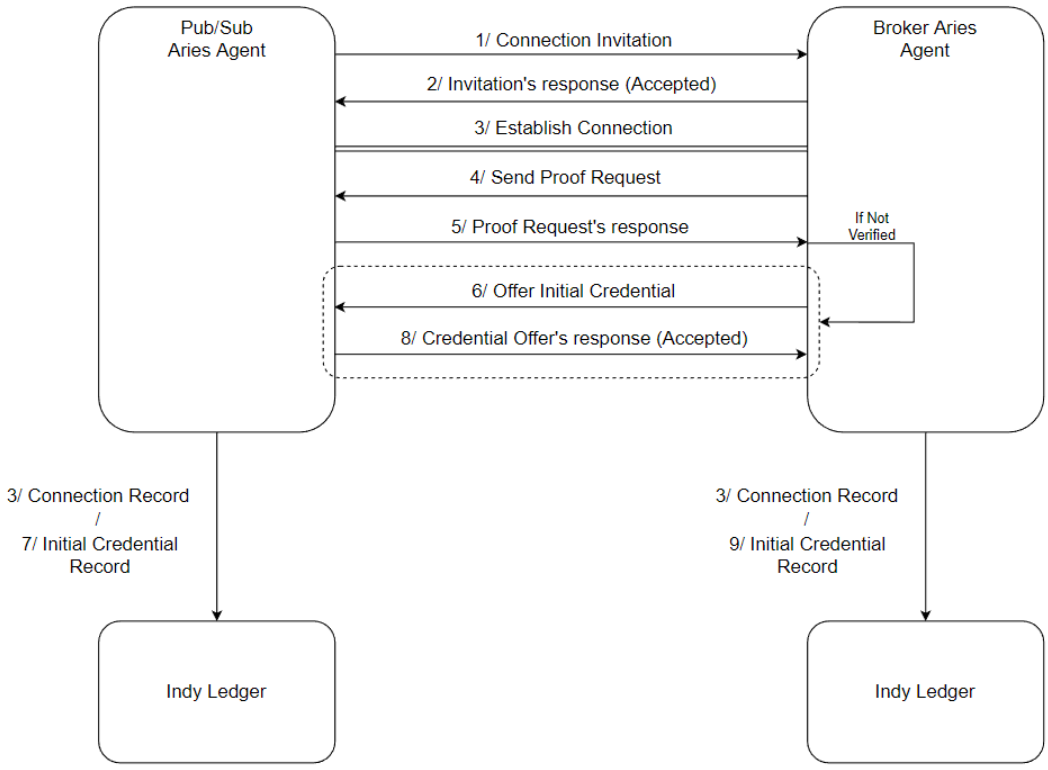

Fig. 3: Initial connection authentication flow

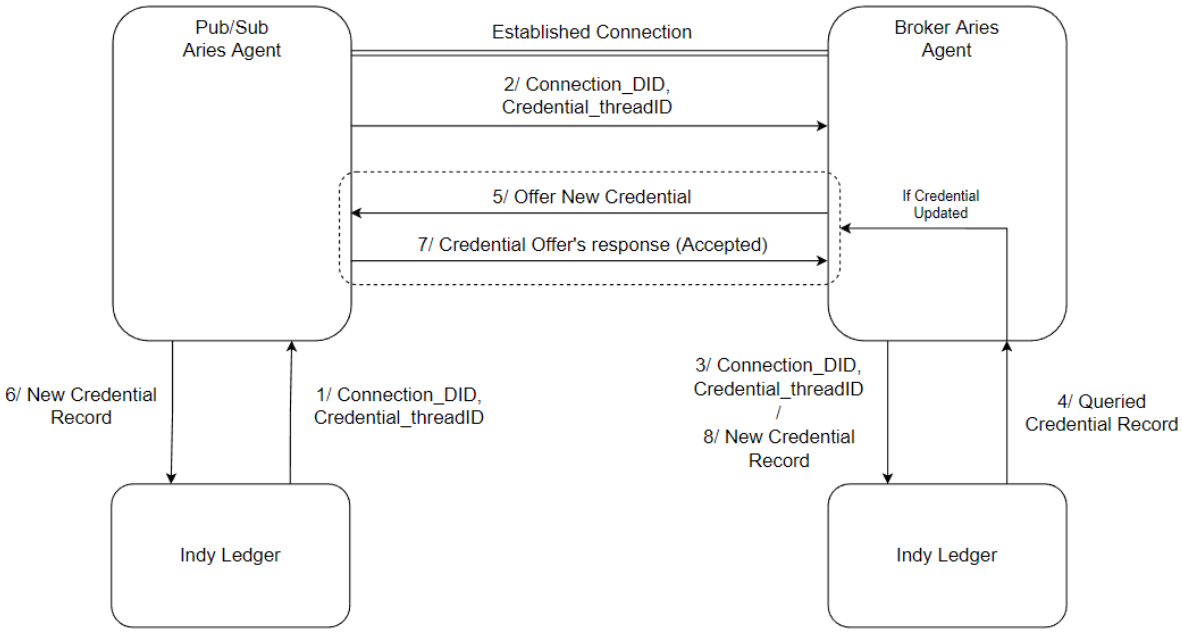

Fig. 4: Service request authentication flow

TABLE I: Test-bed setup

All topics in the broker	**Action**	**Subscription details (Topic: subscribers)**	**Expected result**	**Actual result**	**Comments**
None	pub1 creates topic ‘pub1topic1’ with message ‘p1t1m1’	None	‘pub1topic1’ is created	‘pub1topic1’ is created	New credentials were issued to pub1 to give access to the newly created topic
pub1topic1	pub2 creates topic ‘pub2topic1’ with message ‘p2t1m1’	None	‘pub2topic1’ is created	‘pub2topic1’ is created	New credentials were issued to pub2 to give access to the newly created topic
pub1topic1, pub2topic1	sub1 subscribes to topic ‘pub1topic1’	pub1topic1: sub1, pub2topic1: None	Successfully subscribe to ‘pub1topic1’	Successfully subscribe to ‘pub1topic1’	sub1 subscribed to a topic to get updates from that topic
pub1topic1, pub2topic1	sub2 subscribes to topic ‘pub2topic1’	pub1topic1: sub1, pub2topic1: sub2	Successfully subscribe to ‘pub2topic1’	Successfully subscribe to ‘pub2topic1’	sub2 subscribed to a topic to get updates from that topic
pub1topic1, pub2topic1	sub3 subscribes to topic pub1topic1’ and ‘pub2topic1’	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	Successfully subscribe to ‘pub1topic1’, Successfully subscribe to ‘pub2topic1’	Successfully subscribe to ‘pub1topic1’, Successfully subscribe to ‘pub2topic1’	Sub3 subscribed to all the topics available in broker1

TABLE II: Publisher test

All topics in the broker	**Action**	**Expected result**	**Actual result**	**Comments**
pub1topic1, pub2topic1	pub1 publishes a new message (‘p1t1m2”) to topic ‘pub1topic1’	Transaction submitted	Transaction submitted	pub1 should be able to update ‘pub1topic1’ as it has access to it.
pub1topic1, pub2topic1	pub1 tries to publish a new message (‘p1t1m2”) to topic ‘pub2topic1’	The agent is not permitted to publish on this topic	The agent is not permitted to publish on this topic	pub1 should not be able to update topic ‘pub2topic1’ as pub2 owns this topic .
pub1topic1, pub2topic1	pub2 publishes a new message (‘p2t1m2”) to topic ‘pub2topic1’	Transaction submitted	Transaction submitted	pub2 should be able to update ‘pub2topic1’ as it has access to it.
pub1topic1, pub2topic1	pub2 tries to publish a new message (‘p2t1m2”) to topic ‘pub1topic1’	The agent is not permitted to publish on this topic	The agent is not permitted to publish on this topic	pub2 should not be able to update the topic ‘pub1topic1’ as pub1 owns this topic.
pub1topic1, pub2topic1	pub1 tries to publish to topic ‘NoPubTopic’	‘NoPubTopic’ does not exist	‘NoPubTopic’ does not exist	Since the topic ‘NoPubTopic’ doesn’t exist, no publisher should be able to publish it.

TABLE III: Subscription and topic query testing

All topics in the broker	**Action**	**Subscription details (Topic: subscribers)**	**Expected result**	**Actual result**	**Comments**
pub1topic1, pub2topic1	sub1 queries topic 'pub1topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	Message: 'p1t1m2'	Message: 'p1t1m2'	The latest message on the topic should be received as a result.
pub1topic1, pub2topic1	sub2 queries topic 'pub2topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	Message: 'p2t1m2'	Message: 'p2t1m2'	The latest message on the topic should be received as a result.
pub1topic1, pub2topic1	sub1 queries topic 'pub2topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	The agent is not permitted to query this topic	The agent is not permitted to query this topic	As sub1 is not subscribed to 'pub2topic1', so it should not be able to query it.
pub1topic1, pub2topic1	sub2 queries topic 'pub1topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	The agent is not permitted to query this topic	The agent is not permitted to query this topic	As sub2 is not subscribed to 'pub1topic1', so it should not be able to query it.
pub1topic1, pub2topic1	sub3 queries topic 'pub1topic1' and 'pub2topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	Message: 'p1t1m2' , Message: 'p2t1m2'	Message: 'p1t1m2' , Message: 'p2t1m2'	As sub3 is subscribed to both topics, it should be able to query both topics, which shows query as many topics as we want as long as we have subscribed to it.
pub1topic1, pub2topic1	sub4 queries topic 'pub1topic1' or 'pub2topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	The agent is not permitted to query this topic	The agent is not permitted to query this topic	As sub4 is not subscribed to any topic, it should not be able to query any topic.
pub1topic1, pub2topic1	sub1 queries topic 'NoTopic'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	'NoTopic' doesn't exist	'NoTopic' doesn't exist	Since 'NoTopic' does not exist, it should not be queried.
pub1topic1, pub2topic1	sub1 subscribes topic 'NoTopic'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	'NoTopic' doesn't exist	'NoTopic' doesn't exist	Since 'NoTopic' does not exist, it should not be subscribed.
pub1topic1, pub2topic1	sub1 subscribes to topic 'pub1topic1'	pub1topic1: sub1,sub3, pub2topic1: sub2,sub3	Already subscribed to 'pub1topic1'	Already subscribed to 'pub1topic1'	Since sub1 already subscribed to 'pub1topic1', it should not be resubscribed by the same agent.

VI. Conclusion

This study presents a trust development component for a fully decentralized blockchain interoperability solution based on publish/subscribe architecture, compromising three main blockchain-based components: broker blockchain, publisher blockchains and subscriber blockchains. We integrated the system with SSI using Hyperledger digital identity stack projects to issue and verify credentials for users sharing data through publishing topics and updating existing topics and users accessing the data through subscribing to topics.

The proposed solution promotes a secure infrastructure to prevent illegal information flow among connected blockchains that are run by a network of nodes with various trust levels. We evaluated our system by developing a testbed compromising multiple blockchain networks communicating through publish/subscribe architecture and restricting access to the topics through SSI components by distributedly issuing and verifying temper-proof credentials. In the presented system prototype, only authenticated and authorized participants can publish topics and subscribe to existing topics.

References

[1] A. Lohachab, S. Garg, B. Kang, M. B. Amin, J. Lee, S. Chen, X. Xu, Towards interconnected blockchains: a comprehensive review of the role of interoperability among disparate blockchains, ACM Computing Surveys (CSUR) 54 (7) (2021) 1–39.

[2] H. Yildiz, A. Küpper, D. Thatmann, S. Göndör, P. Herbke, A tutorial on the interoperability of self-sovereign identities, arXiv preprint arXiv:2208.04692 (2022).

[3] M. S. Ferdous, F. Chowdhury, M. O. Alassafi, In search of self-sovereign identity leveraging blockchain technology, IEEE Access 7 (2019) 103059–103079.

[4] H. Ursa, " hyperledger ursa, https://www.hyperledger.org/category/hyperledger-ursa (accessed 2022-07-29.

[5] H. Indy, " hyperledger indy, https://www.hyperledger.org/use/hyperledger-indy (accessed 2022-07-29.

[6] H. Aries, " hyperledger aries, https://www.hyperledger.org/use/aries (accessed 2022-07-29.

[7] S. Ghaemi, S. Rouhani, R. Belchior, R. S. Cruz, H. Khazaei, P. Musilek, A pub-sub architecture to promote blockchain interoperability, arXiv preprint arXiv:2101.12331 (2021).

[8] H. T. Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, M. Mohania, Internet of blockchains: Techniques and challenges ahead, in: 2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), IEEE, 2018, pp. 1574–1581.

[9] Y. Pang, A new consensus protocol for blockchain interoperability architecture, IEEE Access 8 (2020) 153719–153730.

[10] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A survey on blockchain interoperability: Past, present, and future trends, ACM Computing Surveys (CSUR) 54 (8) (2021) 1–41.

[11] G. Wood, Polkadot: Vision for a heterogeneous multi-chain framework, White Paper 21 (2016) 2327–4662.

[12] J. Kwon, E. Buchman, Cosmos whitepaper, A Netw. Distrib. Ledgers (2019).

[13] M. Herlihy, Atomic cross-chain swaps, in: Proceedings of the 2018 ACM symposium on principles of distributed computing, 2018, pp. 245–254.

[14] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, A. Juels, Tesseract: Real-time cryptocurrency exchange using trusted hardware, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1521–1538.

[15] B. Dai, S. Jiang, M. Zhu, M. Lu, D. Li, C. Li, Research and implementation of cross-chain transaction model based on improved hash-locking, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2020, pp. 218–230.

[16] A. Mühle, A. Grüner, T. Gayvoronskaya, C. Meinel, A survey on essential components of a self-sovereign identity, Computer Science Review 30 (2018) 80–86.

[17] U. Der, S. Jähnichen, J. Sürmeli, Self-sovereign identity − opportunities and challenges for the digital revolution, arXiv preprint arXiv:1712.01767 (2017).

[18] D. Van Bokkem, R. Hageman, G. Koning, L. Nguyen, N. Zarin, Self-sovereign identity solutions: The necessity of blockchain technology, arXiv preprint arXiv:1904.12816 (2019).

[19] G. Fedrecheski, J. M. Rabaey, L. C. Costa, P. C. C. Ccori, W. T. Pereira, M. K. Zuffo, Self-sovereign identity for iot environments: a perspective, in: 2020 Global Internet of Things Summit (GIoTS), IEEE, 2020, pp. 1–6.

[20] N. Kulabukhova, A. Ivashchenko, I. Tipikin, I. Minin, Self-sovereign identity for iot devices, in: International Conference on Computational Science and Its Applications, Springer, 2019, pp. 472–484.

[21] B. Houtan, A. S. Hafid, D. Makrakis, A survey on blockchain-based self-sovereign patient identity in healthcare, IEEE Access 8 (2020) 90478–90494.

[22] M. Shuaib, S. Alam, M. S. Alam, M. S. Nasir, Self-sovereign identity for healthcare using blockchain, Materials Today: Proceedings (2021).

[23] A. Siqueira, A. F. Da Conceicao, V. Rocha, Blockchains and self-sovereign identities applied to healthcare solutions: A systematic review, arXiv preprint arXiv:2104.12298 (2021).

[24] M. Shuaib, S. Alam, M. S. Nasir, M. S. Alam, Immunity credentials using self-sovereign identity for combating covid-19 pandemic, Materials Today: Proceedings (2021).