

ChorSSI: A Model-Driven Framework for Self-Sovereign Identity Systems on Blockchain

Tommaso Cippitelli^a, Alessandro Marcelletti^{*,a}, Andrea Morichetta^a

^a University of Camerino, Camerino, Italy

Abstract. *The advent of the digital era has underscored the pivotal role of identity in online activities due to our increasing reliance on digital platforms. Consequently, various identity management systems have emerged, mainly relying on centralised infrastructures and exposed to security vulnerabilities. Self-Sovereign Identity (SSI) presents a promising alternative, as it allows individuals to manage their personal data autonomously and share it securely with others, eliminating the necessity for dependence on a central authority. In particular, the advent of blockchain technology has enabled the creation of novel SSI systems, thanks to its key characteristics of security, decentralisation and control over data. Nevertheless, the complexity and challenges associated with developing such systems and executing related operations pose significant barriers to the broader adoption of the SSI paradigm, particularly for users lacking expertise in the domain. To simplify the development and usability of SSI systems, in this work we propose ChorSSI, a model-driven BPMN-based framework that supports the modelling of an SSI system and the execution of the related interactions. The design relies on BPMN choreography diagrams, permitting the representation of SSI interactions between parties in a distributed manner. The proposed framework was implemented and the feasibility of the tool was assessed over the real Chromaway property transaction case study.*

Keywords. Self-Sovereign Identity • Blockchain • BPMN • Model-Driven Engineering

Communicated by Julius Köpke. Received 2024-01-13. Accepted on 2024-06-03.

1 Introduction

In the digital age, the growing interconnection of services and technologies has raised the significance of identity in online activities Baars (2016). As we progressively depend on digital platforms for communication, commerce, and various other engagements, the imperative for secure and dependable methods of identifying individuals and entities has become more important than ever Bokkem et al. (2019). To address this challenge, diverse identity management systems have been

proposed to oversee digital identities and their interconnections Tobin and Reed (2016). Nevertheless, many of these systems rely on a centralised infrastructure Bandara et al. (2021), necessitating the participation of trusted third parties Grüner et al. (2020). Indeed, such centralised solutions can be exposed to security vulnerabilities, such as data breaches and identity fraud Belchior et al. (2020). A promising alternative to traditional identity management systems is certainly Self-Sovereign Identity (SSI). This is a decentralised identity model that provides individuals control over their personal data and allows them to share data securely and selectively with other parties, without having to rely on a single central authority Der et al. (2017). This is possible thanks to blockchain, which is the underlying technology on which SSI systems are usually built on top of Ferdous et al.

* Corresponding author.

E-mail. alessand.marcelletti@unicam.it

This work has been funded by the European Union - Next-GenerationEU under the Italian Ministry of University and Research (MUR) Extended Partnership initiative on "Security and Rights in the Cyber Space – SERICS" (PE00000014 - CUP H73C22000880001)

(2019). The reason can be mainly identified in the properties that blockchain provides in line with the SSI principles. For example, blockchain provides a secure and tamper-proof distributed ledger to store and manage identity information Bokkem et al. (2019). Furthermore, the use of smart contracts can allow users to deploy immutable programs playing the role of identity provider and access control mechanisms. All these characteristics provide strong advantages considering immutability, provenance, distributed control, accountability and transparency. To enable information sharing on the blockchain, SSI systems rely on Verifiable Credentials (VCs), which are digital representations about an individual, organisation, or thing and they are issued by trusted parties and can be cryptographically verified Sedlmeir et al. (2021). Typical operations of SSI systems are related to the issuing, verification and revocation of VCs done by the actors that interact with an SSI system. Another key aspect of SSI is the confidentiality of interactions between actors, which is ensured by the Zero Knowledge Proof (ZKP) cryptographic protocol, allowing to prove the validity of a statement without sharing the underlying information Kulabukhova (2019). ZKP enhances user privacy while maintaining the necessary institutional trust for the correctness of digital interactions and represents one of the key benefits of SSI. Although the notion of Self-Sovereign Identity (SSI) is relatively recent, there has been an increasing interest in the creation and utilisation of SSI systems and applications.

However, several issues still limit the adoption of this new paradigm, representing challenges during the development and usage of SSI systems for both developers and end-users. The first aspect is related to the construction of SSI systems which poses a complexity for developers who must familiarise themselves with various concepts and technologies, often lacking sufficient support and reference tools due to the topic's novelty. The second aspect affects end-users or stakeholders that are typically involved during the execution of SSI operations. Indeed, the lack of established high-level methodologies and implementations

requires technical knowledge also for those actors who are more focused on the business aspects. All these factors collectively serve as barriers to the development and widespread adoption of the SSI model Manski (2020). To overcome these issues, the integration of low-code strategies can facilitate the development process of SSI systems and make related applications more accessible to a wider range of users.

For this reason, in this work, we propose ChorSSI, a model-driven framework based on the Business Process Model and Notation (BPMN) standard that supports the creation of an SSI system and guides the execution of the related operations. In particular, BPMN choreography diagrams are used to represent the SSI interactions between distributed parties from a high-level perspective. These diagrams provide a comprehensive representation of SSI interactions by determining the overall execution flow solely based on the definition of exchanged messages OMG (2011). In ChorSSI, each step in the choreography is designed to represent an SSI protocol. This protocol specifies for each choreography task the involved parties and the exchanged information (e.g., a set of credentials). This model is then used to build the underlying environment (e.g., network and agents) and guide the execution of the end-user, providing a high-level interface for interacting with the created components without the need for any technical knowledge. This is possible thanks to the supporting mechanisms provided by ChorSSI. Indeed, after the first design is done, ChorSSI automatically builds and connects the underlined software components which are abstracted to the end-user who can focus only on the use case rather than on the technicalities of the underlying technology.

With ChorSSI, we aim at providing an abstract representation of SSI systems and operations with BPMN choreographies, enabling the creation and usage of such systems also to non-expert users. To this purpose, we do not propose a brand new SSI system but we show the effectiveness of the proposed model-driven approach relying on the well-established Hyperledger Aries and Indy

frameworks. Indeed, ChorSSI can be potentially applied to many different technologies, thanks to its abstract representation of technical details. Finally, we validate the tool on the real-world Chromaway property transactions case study.

To resume, the contributions are the manifold:

- A model-driven approach for the design and execution of SSI systems;
- A conceptual mapping between the BPMN standard and SSI operations;
- A tool implementing the proposed approach;
- The approach is validated in a real-world case study.

This work builds on Cippitelli et al. 2023, which originally introduced the ChorSSI approach. We enhanced the approach by integrating a model-driven methodology fully based on BPMN choreographies. Specifically, we automated the ChorSSI phases, eliminating manual steps by deriving them directly from the choreography model. We validated the approach through performance evaluation in a real-world case study, comparing the model-driven execution to manual processes.

The rest of the work is organised as follows. Sect. 2 introduces the main concepts and technologies when developing an SSI system on blockchain. The section also introduced the considered case study. Sect. 3 described the ChorSSI framework and the phases for executing SSI interactions starting from a choreography. In Sect. 4 the ChorSSI prototype is described, highlighting the main phases and functionalities. Sect. 5 shows the feasibility of the approach in a real case study. Sect. 6 introduces relevant works while Sect. 7 concludes and provides an overview of future directions.

2 Background

This section presents the main concepts and technologies on which ChorSSI relies. In particular, a first introduction to SSI systems is given, introducing the actors, components and possible operations related to digital identities. Then, the

used Hyperledger technologies are presented with a final description of the used case study.

2.1 Self-Sovereign Identity Model

In an SSI system, individuals create and control their digital identities, which are stored on the blockchain thanks to the use of different concepts and technologies Mühle et al. (2018).

SSI Components

are the fundamental concepts that characterise self-sovereign identity systems and the elements required to comprehend and use an SSI infrastructure.

Verifiable Credential is a digitally signed data structure that contains a set of information about a subject that can be cryptographically verified. Verifiable Credentials are based on open standards, designed to be secure, privacy-preserving, and decentralised, and are a core component of the SSI model. A Verifiable Credential consists of three main components: the issuer, the subject, and the claims. The issuer is the entity that issues the credential, while the subject is the entity to which the claims apply. The claims are a set of statements about the subject that are digitally signed by the issuer and can include attributes such as name, address, date of birth, or any other type of information. The Verifiable Credential is stored on the subject's device or in a data store and can be shared with others as needed. When a Verifiable Credential is presented, it can be cryptographically verified using the issuer's public key and the underlying cryptography used to sign and verify the credential. The verifier can then determine if the claims are valid, tamper-evident, and properly issued by the trusted issuer.

Decentralised Identifiers (DIDs) are globally unique identifiers that are cryptographically verifiable and self-administered. DIDs are a critical component of the SSI model, enabling secure, decentralised, and interoperable management of digital identities and personal data. DIDs can be used to represent any entity that requires a unique identifier, such as individuals, organisations, things, or even abstract concepts. The DID

is associated with a public-private key pair, for verifying the associated digital signatures. The DID structure is reported in Fig. 1 and is composed of a method-specific identifier, a method-specific prefix that indicates the type of DID method being used, and a method-specific identifier suffix that is unique within the scope of the DID method. The DID is associated with a public-private key pair, and the public key is used to verify digital signatures associated with the DID.



Figure 1: Decentralised identifier

Wallet is a software application that enables an individual or an organisation to store, manage, and control their digital identities and VCs in a secure and privacy-preserving manner. A wallet in the SSI model typically consists of a set of software components and cryptographic keys, which enable the user to store and manage their digital identities and VCs. This includes the ability to receive and store VCs from issuers, manage the presentation of VCs to verifiers, and manage access to the data associated with their digital identities. The wallet is typically designed to be user-controlled, meaning that the user has full ownership and control over their digital identities and VCs. A wallet includes the ability to receive and store VCs, manage their presentation and access digital identity data. The cryptographic keys used to sign and verify digital identities and VCs are stored securely within the wallet, and can only be accessed by the user.

Zero-knowledge Proof is a cryptographic algorithm that enables the sharing of VCs or other sensitive information without revealing the actual content. Indeed, ZKP allows an entity to create digital proof about the ownership of a secret without actually revealing its value. ZKP algorithms are crucial in SSI systems, especially in situations where the sharing of information needs to be

privacy-preserving, such as in healthcare or financial contexts.

SSI Flow

is managed through a defined set of protocols and components. In this context, four key entities can be found: Issuers, Verifiers, Verifiable Data Registries (i. e., the blockchain) and Holders. The latter can also cover the role of Provers during the *Request Proof* operation. Indeed, the mentioned entities act according to the different procedures described below.

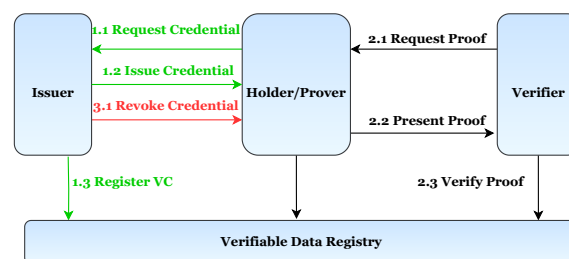


Figure 2: SSI entities and operations.

Issue Credential is the protocol that permits parties to exchange VCs between an Issuer and a Holder and it consists of a sequence of steps. The first one is to create a *credential definition*, which is a template that represents a given type of credential. The Issuer defines the credential schema, including the attributes to be considered (e. g., name, birth date, etc.) and the rules for their verification once the credential is emitted. The second step is the *credential offer*, where the Issuer sends an offer to the Holder, indicating the willingness to issue a credential. The offer includes the schema and any specific attribute values to include. The Holder can then choose to accept or reject the offer. In case of acceptance, the Holder sends a credential request to the Issuer through the *credential request* step. This includes any additional information needed to complete the credential, such as proof of identity. Finally, during the *issue credential*, the Issuer fills the credential, signing it with its private key and sending it to the Holder.

Request Proof is the protocol for requesting and receiving proofs from a Verifier in a secure,

and privacy-preserving manner. The first step is optional and corresponds to *propose presentation*, where the Prover sends a proof proposal message to the Verifier. Notice, the Prover is a credential Holder that during this phase assumes the role of the entity that wants to prove a certain claim. After this, in the *request proof* the Verifier describes what attributes or credentials have to be verified. The message contains the details of the proof request, including the requested attributes or credentials. The message also includes a nonce, which is a random number to ensure the uniqueness of the request. If the Prover decides to fulfil the received request, during the *proof presentation*, the proof is constructed and a confirmation message is sent back. This confirmation message includes both the requested proof and the proof presentation, which serves to demonstrate the validity of the exchanged proof. Subsequently, the Verifier can proceed to verify the proof presentation to ensure its authenticity and confirm that it fulfils the original proof request. If the proof is deemed valid, the Verifier can then use it for the intended purpose.

Revoke of a credential is the last protocol and is performed by an Issuer. In this case, the procedure is rather simple and consists of changing the reference of a valid certificate and its attributes. As a result, when a proof request is invoked, the credential no longer appears valid and is considered revoked.

2.2 Blockchain for Self-sovereign identity

Different identity management models have been developed over the years Ferdous et al. 2019; Pöhn and Hommel 2020. In the *Isolated User Identity* model, service providers have their identity domain providing the final service and the management and authentication of the identity. In this model, users authenticate in specific services performing isolated operations and having partial identities. This brings several issues since for each service, a user has to create a different identity on a different service provider, making the model unfeasible with the growing number of applications. Also from a security perspective,

having both the application and the identity of a single service provider is undesirable, since a single vulnerability leads to many issues. In the *Federated* model instead, an identity domain is used by more service providers and credentials released to the users are valid for all services providers attached to that domain. This enables cross-domain authentication, however, also in this case, the main challenges to consider are related to security and privacy. Finally, the *User-centric* model service providers share a single identity provider without trust. In the last years, the SSI model has grown in interest, especially with the emergence of blockchain which is recognised as a key enabler. Several requirements were identified as reference for SSI systems (Bokkem et al. (2019), Čučko et al. (2022), Ferdous et al. (2019), Nokhbeh Zaeem et al. (2021), Pöhn and Hommel (2020) and Satybaldy et al. (2020)). A fundamental requirement is the **security** of user information that has to be guaranteed by the system protecting identity data. Users' information should also be **transparent** permitting users to be aware of identity usage and interactions with the system while maintaining **ownership** over it. To this purpose, the avoidance of a trusted central authority **reduces the centralisation** of the system. Finally, **interoperability** ensures that services are not confined to a single domain, allowing identities to be used as widely as possible.

In this context, Blockchain technology is becoming a prominent solution for identity models providing new types of decentralised and secure infrastructures. This overcomes the need for a centralised trusted party while guaranteeing security and transparency over data. Despite blockchain representing a valid technological layer, a crucial aspect concerns the usability of the SSI system, which allows users to adopt such novel solutions and reduce complexity. Usability is crucial for an SSI system's success Čučko et al. (2022) and still represents a challenge in current established applications Satybaldy et al. (2020).

2.3 BPMN Choreography

The most relevant elements used in choreography diagrams are depicted in Fig. 3. On the left, the elements used for the business process's control flow are represented, while on the right the elements used for communication purposes. Typically, a choreography model is composed of different types of elements: events, sequence flows, and tasks. *Events* can be a *start event*, representing the starting point of the choreography, and an *end event*, raised when the choreography terminates. *Gateways* act as either join nodes (merging incoming edges) or split nodes (forking into outgoing edges). We can have different types of gateways. A *parallel gateway* (AND) in join mode has to wait to be reached by all its incoming edges to be activated, and subsequently, in the split case, all the outgoing edges are initiated simultaneously. An *exclusive gateway* (XOR) represents choices; it is initiated each time the gateway is reached in join mode, and it activates exactly one outgoing edge in split mode. In the *event-based gateway*, the outgoing branches activation depends on the reception of a message; the message events connected to the gateway are in race condition: the first one that is triggered activates the corresponding branch and disables the other ones. *Sequence Flows* are edges used to connect all the choreography elements, permitting the specification of the execution flow. *Tasks* are used to define the message exchanges between the parties involved in the choreography. They are represented as rectangles divided into three bands: the central one contains the task's name, while the others refer to the involved parties (the white band is the initiator, the grey one is the recipient). Messages can be sent either by one party (One-Way tasks) or by both parties (Two-Way tasks).

2.4 Hyperledger Aries & Indy

Hyperledger Aries is an open-source toolkit within the Hyperledger ecosystem that provides a set of protocols and tools for creating, transmitting, and storing verifiable digital credentials. *Aries Agents* are software components that enable secure communication and data exchange between different

entities in a decentralised system. These agents use cryptographic protocols and digital credentials to establish trust between parties and facilitate secure data sharing. Aries Agents can be categorised and configured based on the role they play in a decentralised system, including Issuer Agents, Holder Agents, and Verifier Agents.

Hyperledger Indy is a decentralised, open-source blockchain platform that provides the infrastructure for building and using DID solutions. It is specifically designed to address the unique requirements of decentralised identity systems, such as privacy, security, and interoperability. Hyperledger Indy provides several key features that are essential for decentralised identity systems, including the ability to create and manage DIDs, the ability to issue and verify verifiable credentials, and the ability to build privacy-preserving and secure interactions between entities.

A *von-network* is a pre-packaged Indy network built by the Government of British Columbia's Digital Identity and Trust team.¹ It provides an easy way to run a local sandbox Indy network using Docker containers with minimal effort. The network is designed to enable digital identity solutions that allow people, organisations, and things to prove their identities to each other in a secure and decentralized manner. The von-network offers several features to simplify the development of digital identity solutions. Von-network provides a web interface that allows developers to browse the transactions on the ledger and monitor the state of the network.

2.5 SSI Case Study

This section introduces the ChromaWay property transactions project reported in Fig. 4, a blockchain-based system for managing property transactions in Sweden. This project is a significant example of self-sovereign identity that has been highlighted also in Allessie et al. (2019) by the European Commission. This use case serves as an inspiration for exploring the potential of ChorSSI in real-world applications. Indeed, the real estate

¹ <https://github.com/bcgov/von-network>

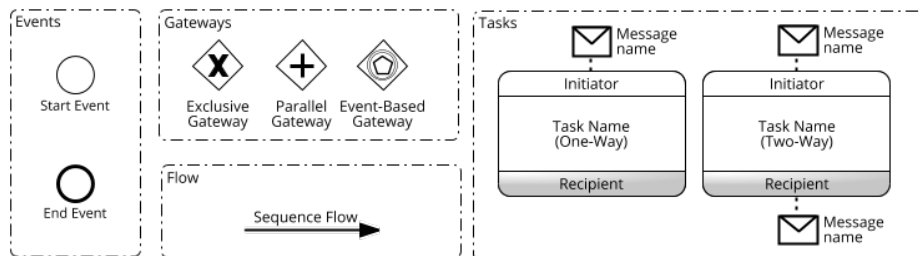


Figure 3: BPMN choreography elements.

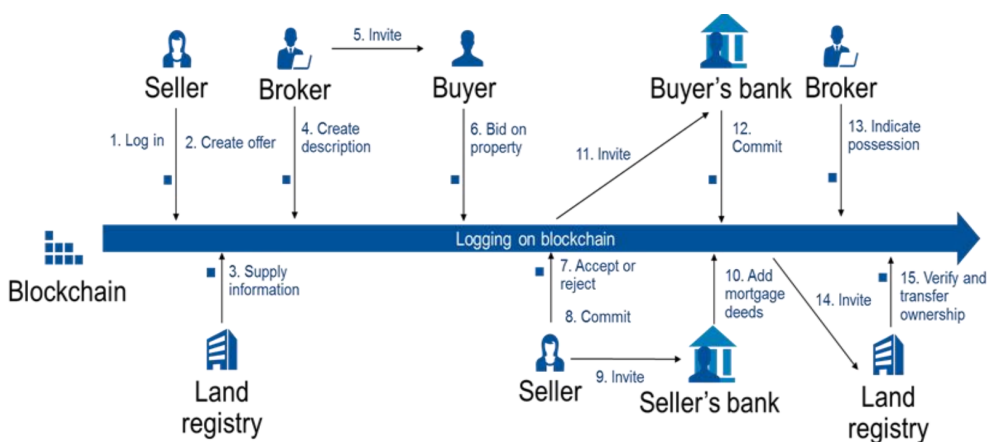


Figure 4: Chromaway case study from Alessie et al. (2019).

industry involves high-value transactions, making it crucial to ensure the security and transparency of property transfers. However, the current system for real estate transactions is slow, expensive, and prone to errors, including contested property deeds. To address these challenges, the Chromaway project was initiated in September 2016. The goal was to redefine real estate transactions and mortgage deeds, addressing the pain points of the current system, which include the lack of transparency, a slow registration system, and the complexity of agreements between buyers and sellers due to a lack of trust in the system and high transaction costs. The solution introduces a new blockchain-based workflow that streamlines and secures the process of transferring property titles. The Swedish Land Registry stores land titles while the blockchain stores the system’s state after each step in the workflow. The synchronisation of par-

ticipants involved in the transaction is guaranteed in this way.

3 ChorSSI Framework

This section reports the ChorSSI framework, highlighting the main components and the different phases permitting the design of a choreography model representing SSI operations until its final execution and monitoring.

3.1 Components

The ChorSSI framework (Fig. 5) is a model-driven framework based on choreography diagrams. It enables the creation of SSI systems and the execution of the related interactions while abstracting from low-level and technical details.

Organisations willing to collaborate mutually agree on a choreography diagram, which is used to represent the communication between the parties.

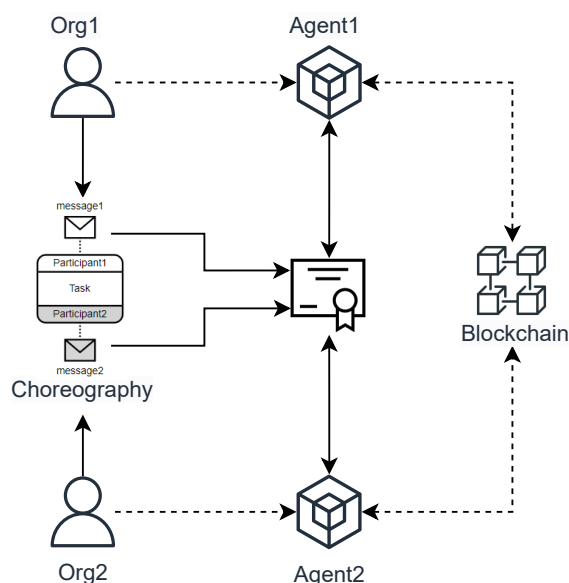


Figure 5: ChorSSI framework and components.

The interactions reported in the choreography represent the standard SSI operations involving the exchange and validation of credentials, coordinated by the specified control flow. In particular, ChorSSI supports the *issue credential*, *request proof* and *revoke* procedures. These procedures are concretely executed by software agents attached to each organisation and automatically generated by ChorSSI starting from the model. Agents support the exchange of data coming from the organisations and store relevant information on the blockchain.

The adoption of a model-driven methodology permits organisations to communicate on top of an infrastructure without the need for any technical knowledge. Furthermore, this facilitates the generation of the requested components and the invocations of their functionalities. Indeed, the role of organisations is limited to the insertion of relevant information during the execution. The primary responsibility of ChorSSI is to start the network and create the software components required to run and execute the SSI system. Essentially, a choreography represents the execution flow of an SSI system and it contains all the information

needed to generate the underlying infrastructure and to manage the data exchange.

3.2 Phases

The main phases of the ChorSSI framework are depicted in Fig. 6 and they are, namely, *Modelling*, *Execution* and *Monitoring*.

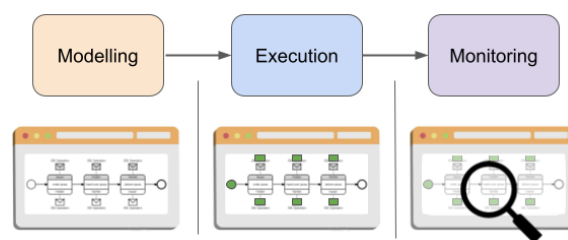


Figure 6: ChorSSI phases.




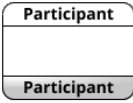

Modelling

The modelling phase is the initial stage of ChorSSI and involves the creation of the choreography model. This model contains all the elements representing the SSI scenario and the corresponding interactions between organisations. After the model specification, the SSI system enables the interaction and data exchange between organisations. To this purpose, ChorSSI first starts the blockchain infrastructure on which data is stored and then creates the agents according to the participants of the model. Tab. 1 reports the mapping between BPMN choreography elements and the SSI concepts in ChorSSI. The execution flow of the system is derived following the *sequence flows* (3) existing between the *start event* (1), and the *end event* (2) in the model. In particular, the *start event* indicates the first actions to perform and is also responsible for triggering the generation of the environment. The *end event* indicates the end of the process, thus terminating the execution. *Sequence flows* determine the sequence of messages to execute and the steps to complete. Other core elements are *choreography participants* (4), representing the involved organisations willing to cooperate. They are used to generate the software artefacts forming the underlying network and agents. Participants are also used to specify the

senders and receivers of data. Finally, the communication between the organisations is achieved through the exchange of *choreography messages* (5), which represents SSI operations. Each operation of the SSI protocol is built-in inside ChorSSI and, according to the message type, it is bonded to a particular message. In this way, during the execution, users only need to insert the required information and ChorSSI automatically invokes the underlying infrastructure. For each message in the model, ChorSSI permits the specification of the attributes forming a particular credential that can be used for issuing or proving activities.

During this phase, if an error is detected from an incorrect design, the choreography can be directly revised by adjusting the model as needed. This is possible because the generation of software artefacts occurs in a second moment, allowing the choreography to be modified without any restriction.

Table 1: Mapping of BPMN choreography elements to SSI operations.

BPMN element	SSI operation
1)  Start Event	<ul style="list-style-type: none"> Starts blockchain network Creates agents
2)  End Event	<ul style="list-style-type: none"> Indicates the process end
3)  Sequence Flow	<ul style="list-style-type: none"> Defines the execution flow
4)  Participant	<ul style="list-style-type: none"> Identifies SSI roles Defines agents Defines the exchange of data
5)  Message	<ul style="list-style-type: none"> Defines SSI operation Defines attributes Defines credentials

Execution

During the execution, the organisations perform the operations related to the SSI protocol following the choreography model. The model represents the execution flow of the implemented SSI system and it requires organisations to specify only the information needed at run-time. Thanks to the ChorSSI functionalities, the invocation and communication with the blockchain are automatically managed using the agents. Each organisation can thus focus on executing a specific task by simply providing the required information. In this way, ChorSSI reduces the amount of manual coding needed increasing productivity. Additionally, it allows final users to better focus on the complexity of the business process, thereby reducing the time required for manual development of the SSI-based use case.

In case an incorrect design is used to derive the low-level infrastructure, new modelling is necessary to correct it. Indeed, ChorSSI does not support the run-time update of the configurations and a new choreography model must be designed to address the errors. This revised model is then used to derive a new infrastructure, replacing the faulty one, and enabling participants to communicate effectively.

Monitoring

The last phase of ChorSSI enables organisations to observe the executed operations and monitor the current state of the process. This is done by controlling all the produced and exchanged data. In that way, each party can check whenever a certain object meets the initial expectations. The monitor permits to verify that the executed operations meet the requirements specified in the starting choreography. Additionally, it provides an opportunity to test and evaluate the SSI scenario, identifying uncovered issues or opportunities for improvement that may not have been recognised during the development process. Finally, it provides feedback that can be used for further refinement and improvement of the SSI scenario and the treated use case over time.

4 ChorSSI Implementation

This section shows the prototypical implementation of the ChorSSI framework.

The tool was implemented as a web application based on JavaScript programming language through the use of Node-Js and connected to the von-network which is a pre-packaged blockchain network of Hyperledger Indy.²

4.1 Modelling

In the modelling phase, the end-user can design the Choreography diagram for creating the SSI system and executing SSI-related operations. The model contains all those interactions related to SSI, including the issuance of an identity and the verification of a credential. The modeller is available on the ChorSSI home page and is based on the chor-js component Ladleif et al. (2019) supporting the BPMN choreography elements. This makes it possible to create choreography diagrams in a standard manner, by using drag-and-drop functionalities according to the supported elements presented in Tab. 1. The modeller is reported in Fig. 7 and was extended with a new SSI panel accessible by selecting a BPMN element within the model.

This panel allows the dynamic configuration of SSI-related data objects to be exchanged during each phase of the execution, as well as the possibility to set their content. During the message creation, the user can select the corresponding SSI operation type in the SSI panel. The admissible message types include *offer*, *accept*, *proof request*, and *present proof*. These are indeed the types that support the various SSI protocols in ChorSSI as described in Sect. 2 and Sect. 3.

In the case the design of a message exchange represents an issue credential protocol, the *Offer Credential* message requires the definition of the credential by the setting of the name of the schema and the list of attributes as demonstrated in Fig. 7. When instead a proof request is selected, the user has to insert only the attributes that are required to be verified during the execution.

² <https://github.com/bcgov/von-network>

Once the messages and the participants are designed and configured, ChorSSI creates the underlying SSI infrastructure which includes the network and the related agents. This is done by first clicking on the start event of the choreography and then on the *connect* button in the SSI panel. As the first step, ChorSSI builds the Indy network and starts the related nodes. Then, it automatically registers a DID for each agent representing an organisation. This is done by extracting the choreography participants from the model and using their names and IDs to form the DID and define the various agents. At this point, the Indy Tails Server is started. This is a specialised file server used to receive, store, and distribute Hyperledger Indy Tails files. Those contain the information required for credential holders to produce a zero-knowledge proof when responding to a proof request. Finally, ChorSSI starts the registered agents corresponding to the organisations that participate in the choreography and connects them forming dedicated channels.

Once the network is initiated and the agents are registered and operational, the credential definitions of each message are created starting from the previously established schema. This is done by issuer agents which are invoked by ChorSSI to write the schemes, specified in the offer messages, to the public ledger. The generated credential definitions are created and written on the ledger.

4.2 Execution

In the execution phase, all the choreography participants can take part in the workflow execution by exchanging data. This communication is concretely supported by determining credentials and attribute values which are then sent to the participants for issuing or proving activities. For each message in the model containing a credential, the SSI panel shows a table in which the user can insert the value of the previously defined attributes and send them as depicted in Fig. 8. This action triggers ChorSSI that automatically invokes the agents of the participants involved in the task

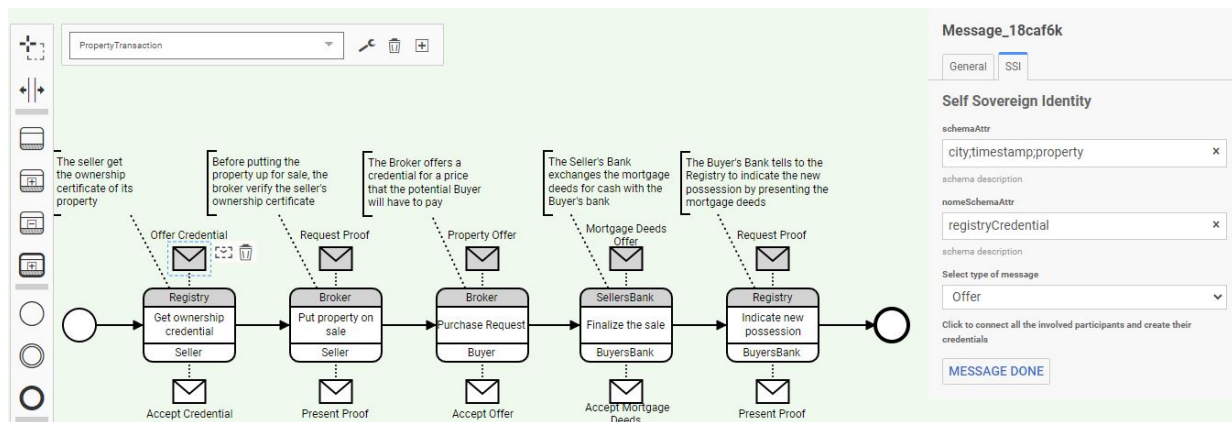


Figure 7: Offer message definition.

leading to subsequent interaction with the blockchain. Depending on the SSI protocol, different execution flows are produced in ChorSSI.

In the case of an *issue credential* interaction, the credential object is created and sent. Here the connection identifier of the counterpart and the credential definition previously created are automatically set by ChorSSI. Once the credential information is set, the user can send it and ChorSSI automatically manages the agent execution. At this point, the counterpart receives the credential, becoming the holder responsible for accepting it. This time, only the view of the received credentials is provided. If the issued credential satisfies the receiver organisation, the latter can accept it by executing the accept message. This triggers the related agent that stores the issued credential in the Holder's wallet and registers it on the blockchain. Finally, the completed messages become green indicating the successful completion of the task.

The second possible operation of the choreography diagram represents the *request proof* protocol. During this phase, a proof request message is sent to the participant acting as the Prover. Also, in this case, the selection of the Prover is automatically done by ChorSSI exploiting the choreography. Here the SSI panel contains the request proof object containing all the attributes the Verifier intends to verify. The transmission of the proof request to the counterpart is then managed by ChorSSI invoking the underlying agent. Once

the proof request message is received, the Prover has to respond with a valid credential satisfying it. This is made by the Prover which responds with a proof presentation message containing the credential attributes to present. This triggers the agent, concluding the execution of the protocol.

The last admitted SSI-related operation is the *revoke* of an issued credential. This can be done by an Issuer clicking on the button on the page used also for monitoring purposes. Subsequently, the credential will be marked as *Revoked* in the Holder wallet. This can still be used for presenting proof but the Verifier will notice that the presented credential has been revoked.

4.3 Monitoring

The last phase of the ChorSSI methodology is the monitoring of the various operations performed during the choreography execution. For checking the content of the responses each user can navigate to the 'Profile' section and select the related page. On the Profile page, we can find a status bar containing all the agents. From this, we can select the agent correlated to the participant of the diagram of which we want to observe information details (coloured in green). The shown information will depend on the object that the selected agent has generated and the role played in the SSI system. If the selected agent is an issuer, the visible information includes the created credential

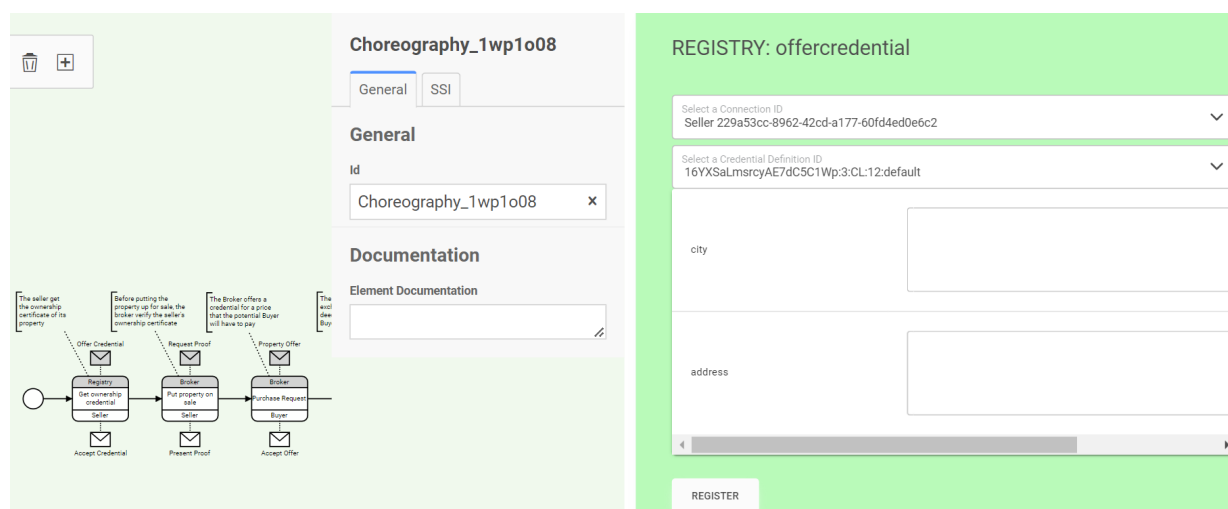


Figure 8: SSI panel during message execution.

definition, the linked schema object, and a reference to the issued credential. Additionally, the option to revoke a credential is available. If the agent is a Holder the issued credential stored in its wallet is visible. Lastly, if the agent is a verifier, the objects related to the verification of the proof presented by holders are visible. It's worth noting that an agent could act as both a verifier and an issuer as well as a holder without constraints, this depends on the specific use case model. While the monitor phase highlights the execution of SSI operations, the underlined infrastructure generated by ChorSSI can be inspected using the dedicated agents' APIs and network explorer. In particular, they expose low-level functionalities that monitor running nodes, executed transactions, agent registration, and more.

5 Validation

This section provides the validation of ChorSSI³ on the Chromaway case study presented in Sect. 2.5. The choreography model of the scenario was reported in Fig. 9. The Seller starts by obtaining the ownership certificate of its property from the Land Registry, which is represented as an issue credential protocol. Before putting the property

up for sale, the Broker verifies the Seller's ownership certificate by employing the request proof protocol. The Broker then offers the property in the form of a credential and establishes a price for potential Buyers. This involves issuing a credential about the property offer to the Buyer. At this point, the Seller's Bank exchanges the mortgage deeds for cash with the Buyer's bank by issuing the credential. Finally, the Buyer's Bank informs the Registry of the new ownership by presenting the mortgage deeds thus, completing the process. This last step involves the Buyer's Bank presenting the proof of the possession which is verified by the registry in the final task. The aforementioned Chromaway case study was designed using the ChorSSI tool. As a result, the model is composed of 5 tasks and comprises 6 participants with a total of 10 messages, divided into 6 issue credential types and 4 request proof types.

5.1 Performance analysis

In this section, we aim to assess the tangible benefits of ChorSSI in the context of the presented case study. To achieve this, we conducted a comparative analysis of the time required for initialisation and execution operations in the Chromaway system using ChorSSI versus a manual setup. Measuring this impact proves to be a complex task, given the many inherent factors that must

³ The tool source code together with the conducted tests are available at <https://pros.unicam.it/chorssi/>

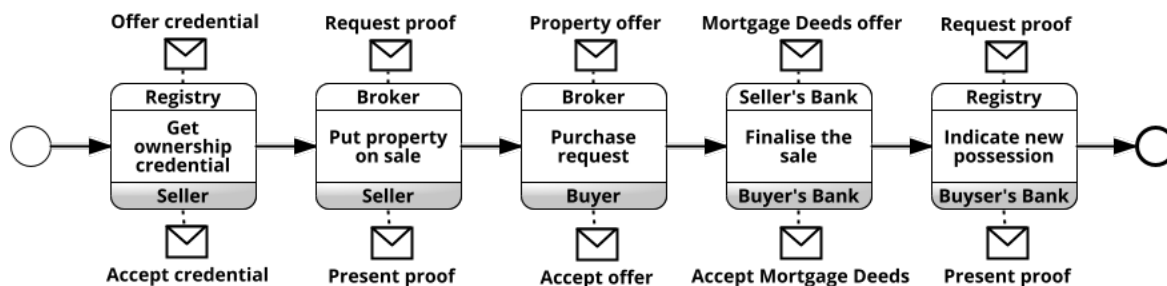


Figure 9: Chromaway case study choreography diagram.

be considered. These factors include hardware specifications, the expertise of the developers, and their skill level. To mitigate potential inaccuracies, we maintained a consistent hardware environment for both scenarios. This approach ensures that any time disparities during the different operations cannot be attributed specifically to the computing infrastructure. Additionally, we sought to standardise the expertise level of the developer involved in the manual measurement. For this purpose, we selected a developer with more than one year of SSI systems experience and presented the Chromaway case study. This decision aims to establish a baseline level of competence and familiarity with the case study, facilitating a fair and meaningful comparison between ChorSSI and an experienced developer. It is important to recognise that despite these precautions, other variables may still impact the results and that in the measurement we did not consider the time used for the developer to implement the system, but only the time for its initialisation.

From the presented scenario, we measured the performance in terms of *time* required from ChorSSI to complete all the low-level *operations*. To assess the validity of the model-driven methodology, we compare the results obtained above with the manual initialisation and execution of the same scenario performed by the developer invoking directly the low-level commands. Also in this case, we measured the time required to write and execute the operations, comparing it with the ChorSSI version. Furthermore, in the

comparison, we maintained the same *parameters*, corresponding to the technical information that the user has to provide to execute the system and how this is abstracted in ChorSSI. For both experiments, we identified the common activities that have to be carried out to start and execute an SSI system. The first corresponds to the initialisation of the infrastructure, which includes the start of the network, the tails server and the registration of agents. Then, the agents are connected to establish the communication channels required for data exchange. Following this, the creation of credentials schema and definition becomes essential for representing the run-time object to be exchanged. Finally, the issue credentials and verify proof protocols are executed, representing the run-time operations of the SSI system. Each of these operations requires a set of parameters. In particular, for the operations related to networks and agents, several configuration options have to be defined, requiring technical expertise. Similarly, during execution operations, intricate details about agents and credential references are essential to specify senders, receivers, and the data to be transmitted. These requirements demand a profound knowledge of the SSI system and how it works, representing a barrier to its wider usage, especially when non-expert users are involved.

Tab. 2 reports the performance results of the manual experiment. This requires technical knowledge from the developer about prerequisites and technicalities to manually start and execute the low-level SSI system. Since no tool is used, all

the configurations and message exchanges have to be done manually, requiring technical expertise. Here, it is possible to notice the high time required for the operations, especially for initialising the SSI system. This is primarily attributed to the length of the commands that need to be manually written and the multiple parameters to be configured. It is important to mention that we considered the most basic scenario, where a developer writes and executes each command. In this context, we did not consider the potential time required to identify any necessary IDs and data; instead, we assumed that this information was already known.

Tab. 3 reports the results obtained with the ChorSSI tool. Differently from the previous experiment, this time modelling knowledge is required. This corresponds to the knowledge about the BPMN standard and the representation of the SSI system interactions needed to design the choreography model used by ChorSSI. Indeed, thanks to the model-driven approach, the initialisation of the SSI system, agent connection and credential definition are grouped under a single operation. This requires the choreography model to be designed and the messages to be filled with the types and credential data as well as schema and attributes. To provide a realistic setting, we also include the time required to create the BPMN choreography model and set the needed data. Similarly to the previous analysis, we assume that the elements to include are already known. The result shows a total time of 720 seconds which is significantly lower compared to the manual execution. Indeed, thanks to the automatic setting of the parameters, ChorSSI reduces the complexity required by the end user for creating SSI infrastructures. Also in the case of protocol operations in ChorSSI, the resulting time is lower compared to the manual execution with a total time of 239 seconds. As for the previous operation, the avoidance of low-level parameter definition facilitates the execution of SSI interactions. Indeed, while the execution inside agents can be measured in terms of milliseconds, the main impacting aspect is the parameter insertion.

5.2 Discussion

Analysing the ChorSSI approach and the obtained observations, several considerations arise. The primary benefit consists of executing SSI operations without manually implementing infrastructure and individual interactions. This is facilitated by the model-driven approach supported in ChorSSI and its integrated functionalities. Indeed, the only effort resides in the initial modelling and in messages configuration in which the objects to be exchanged need to be defined. Indeed, the model-driven experiment demonstrated a substantial advantage compared to the manual one conducted by the expert user, especially considering that we have provided all the essential commands to replicate the same experiment for comparability. Naturally, this gap can only increase if we consider not offering any guidance to the developer who has to start from scratch. Furthermore, if we consider standard users, this gap could be insurmountable, due to the requisite knowledge to implement an SSI system. Using ChorSSI instead, users with any level of knowledge can produce an SSI system without any apparent difference in performance since the framework manages all aspects of the process automatically. Secondly, the use of a choreography model abstracts the technicalities of SSI systems, requiring users only to input the necessary information. Here the choreography plays a pivotal role by defining the execution flow and is the means for automatically generating software artefacts along with invocations to their low-level capabilities. A further benefit to consider is the support for monitoring purposes in an easy manner, eliminating the need for external instruments. Indeed, ChorSSI captures and presents all exchanged information, providing a user-friendly view.

6 Related Works

In recent years, the field of self-sovereign identity has gained significant attention as it aims at giving individuals more control over their data and identities by allowing them to create and manage their

Table 2: Time and technical knowledge required to manually execute the Chromaway SSI system.

Operations	Required parameters	Times (s)
Initialise SSI system (von network, tails server, register agents, start agents)	agent seed, agent network, endpoint wallet-type, seed, wallet-key, wallet-name, genesis-url, inbound-transport, outbound transport, admin, admin-insecure-mode, endpoint, auto-provision, auto-accept-invites, auto-accept-requests, tails-server-base-url, preserve-exchange-records, auto-ping-connections, auto-verify-presentation	2040
Connect agents	agent port, receiver did, receiver endpoint	600
Create Schema and Credential Definition	agent port schema id	180
Issue credential	agent port, schema id, credential definition id, credential exchange id	720
Verify Proof	agent port, presentation exchange id, credential definition id	480

Table 3: Time and modelling knowledge required to execute the Chromaway SSI system in ChorSSI.

Operations	Required parameters	Times (s)
Initialise SSI system (von network, tails server, register agents, start agents, connect agents, create schema, credential definition)	Choreography model (participants, messages, control flow elements), message types, schema name, credential attributes	720
Issue credential	Attributes value	124
Verify Proof	Attributes to verify	115

own digital identities without relying on centralised authorities. While some solutions currently exist such as cloud-based ones Mätäsniemi (2023) and Siddiqui et al. (2021), in this work we focus on blockchain technology, which marked a significant shift in this field, establishing it as a key enabler for the SSI model.

In this context, the creation of supporting mechanisms for the development and execution of such systems is a relevant topic. This can be achieved through the use of visual models, which allow individuals with domain knowledge, but limited technical capabilities, to understand the content of each layer of the SSI ecosystem. This is beneficial as it allows for a wider range of stakeholders to participate in the development process and contribute their expertise Sroor et al. (2022).

However, while many works faced the development of blockchain-based systems using high-level models Corradini et al. (2021, 2022, 2023), López-Pintado et al. (2019) and Tran et al. (2018) exploiting model-driven techniques, the integration in the SSI domain is a relatively new and evolving field of research and few approaches are currently available. In Alboaie and Cosovan (2017), the work proposes a decentralised approach to private data storage and sharing, called the Private Data System (PDS), to address privacy issues arising from the centralised architectures used by governments and large companies to store and share massive amounts of personal data. PDS is composed of nodes spread across the Internet that communicate through executable choreographies, enabling self-sovereign storage and sharing of private data. While in PDS choreographies are

used for achieving the communication between network nodes, in ChorSSI they are used at a higher level of abstraction, to act as a standard interface for end-users. Differently from ChorSSI, PDS is intended to address the self-sovereignty elements that arise from privacy regulations, rules, and principles. In Alboaie et al. (2019) the authors present a platform that uses swarm communication and executable choreographies to implement smart contracts and a generic architecture for blockchain-based systems. The authors propose novel concepts, including secret smart contracts and the near-chain approach, a storage strategy to achieve data self-sovereignty. In this case, the proposed approach exploits a model-driven technique to provide a trusted environment for the creation and execution of smart contracts. In addition, the concept of self-sovereignty is persecuted with the use of Cloud Safe Boxes (CSB), an encrypted folder that can be seen as a sort of offline blockchain with special access rules about who can update or even delete them. With the introduction of choreographic smart contracts, CSB is exploring transaction validation without running the code in all nodes. Despite the required privacy of an SSI system is guaranteed and the performance is improved, the main difference with ChorSSI is the use of the choreography diagram to separate the technical complexity of an SSI system and to make it usable to non-technical organisations.

Although it is known that the use of visual approaches enables the easier development of SSI systems, concrete proposals supporting the modelling and execution through models are still lacking. For these reasons, ChorSSI introduces novelty by enabling the execution of SSI systems through choreographies, even for non-technical organisations.

7 Conclusions & Future Works

The continuous interconnection between people, things, and services has raised the need for systems capable of storing and managing digital identities. To this aim, over the years several identity models have been proposed, though they

are usually centralised and rely on trusted third parties. A prominent alternative to those models is Self-sovereign Identity in which individuals are responsible for their own data. This has been enabled by blockchain, which provides a secure and decentralised infrastructure for implementing SSI systems. However, the development and execution of such software require technical knowledge and skills in different technologies from both developers and stakeholders, hindering the adoption of the SSI model.

For this reason, in this work, we propose ChorSSI, a model-driven and BPMN-based framework for supporting the development and guiding the execution of Self-Sovereign Identity systems and functionalities. The proposed framework enables individuals and organisations to have control over their digital identity and to selectively disclose verifiable claims to other parties, without the need for intermediaries.

The use of BPMN choreography diagrams to depict SSI protocols not only streamlines the implementation process but also guides the correct execution flow, making it more accessible for non-expert users. In particular, ChorSSI permits users to create and manage digital identities, issue and verify credentials, and share verifiable claims with other parties. The framework was evaluated through the development of a prototype and the execution of a real-world case study involving Chromaway property transactions.

In future work, we intend to enrich the ChorSSI modelling phase to support the implementation of multi-tenancy agents, which makes it possible to represent multiple organisations for a single role. This makes it possible to support a wider range of BPMN elements such as multiple instance ones.

References

Alboaie S., Cosovan D. (2017) Private data system enabling self-sovereign storage managed by executable choreographies. In: International Conf. on Distributed Applications and Interoperable Systems. Springer, pp. 83–98

- Alboaie S., Alboaie L., Pritzker Z., Iftene A. (2019) Secret smart contracts in hierarchical blockchains. In: Information Systems Development: Information Systems Beyond 2020. ISEN Yncréa Méditerranée
- Allessie D., Sobolewski M., Vaccari L., Pignatelli F. (2019) Blockchain for digital government. In: Luxembourg: Publications Office of the European Union, pp. 8–10
- Baars D. (2016) Towards self-sovereign identity using blockchain technology. <http://essay.utwente.nl/71274/>
- Bandara E., Liang X., Foytik P., Shetty S., De Zoysa K. (2021) A blockchain and self-sovereign identity empowered digital identity platform. In: 2021 International Conf. on Computer Communications and Networks. IEEE, pp. 1–7
- Belchior R., Putz B., Pernul G., Correia M., Vasconcelos A., Guerreiro S. (2020) SSIBAC: self-sovereign identity based access control. In: International Conf. on Trust, Security and Privacy in Computing and Communications. IEEE, pp. 1935–1943
- van Bokkem D., Hageman R., Koning G., Nguyen L., Zarin N. (2019) Self-Sovereign Identity Solutions: The Necessity of Blockchain Technology. In: arXiv preprint arXiv:1904.12816
- Cippitelli T., Marcelletti A., Morichetta A. (2023) Chorssi: A bpmn-based execution framework for self-sovereign identity systems on blockchain. In: Business Process Management: Blockchain, Robotic Process Automation and Educators Forum. LNBIP. Springer, pp. 5–20
- Corradini F., Marcelletti A., Morichetta A., Polini A., Re B., Scala E., Tiezzi F. (2021) Model-driven engineering for multi-party business processes on multiple blockchains. In: Blockchain: Research and Applications 2(3), p. 100018
- Corradini F., Marcelletti A., Morichetta A., Polini A., Re B., Tiezzi F. (2022) Engineering Trustable and Auditible Choreography-based Systems Using Blockchain. In: ACM Trans. Manag. Inf. Syst. 13(3), 31:1–31:53
- Corradini F., Marcelletti A., Morichetta A., Polini A., Re B., Tiezzi F. (2023) A flexible approach to multi-party business process execution on blockchain. In: Future Generation Computer Systems 147, pp. 219–234
- Čučko Š., Bećirović Š., Kamišalić A., Mrdović S., Turkanović M. (2022) Towards the classification of self-sovereign identity properties. In: Ieee Access 10, pp. 88306–88329
- Der U., Jähnichen S., Sürmeli J. (2017) Self-sovereign identity – opportunities and challenges for the digital revolution. In: arXiv preprint arXiv:1712.01767
- Ferdous M. S., Chowdhury F., Alassafi M. O. (2019) In search of self-sovereign identity leveraging blockchain technology. In: IEEE access 7, pp. 103059–103079
- Grüner A., Mühle A., Gayvoronskaya T., Meinel C. (2020) A comparative analysis of trust requirements in decentralized identity management. In: International Conf. on Advanced Information Networking and Applications. Springer, pp. 200–213
- Kulabukhova N. (2019) Zero-knowledge proof in self-sovereign identity. In: CEUR Workshop Proceedings Vol. 2507, pp. 381–385
- Ladleif J., von Weltzien A., Weske M. (2019) chor-js: A Modeling Framework for BPMN 2.0 Choreography Diagrams.. In: ER Forum/Posters/-Demos. CEUR-WS.org, pp. 113–117
- López-Pintado O., García-Bañuelos L., Dumas M., Weber I., Ponomarev A. (2019) Caterpillar: A business process execution engine on the Ethereum blockchain. In: Softw. Pract. Exp. 49(7), pp. 1162–1193
- Manski S. (2020) Distributed ledger technologies, value accounting, and the self sovereign identity. In: Frontiers in Blockchain 3, p. 29
- Mätäsniemi K. (2023) Solving Inter-Organizational Data Sharing Challenges With Gaia-X

Mühle A., Grüner A., Gayvoronskaya T., Meinel C. (2018) A survey on essential components of a self-sovereign identity. In: *Computer Science Rev.* 30, pp. 80–86

Nokhbeh Zaeem R., Chang K. C., Huang T.-C., Liao D., Song W., Tyagi A., Khalil M., Lamison M., Pandey S., Barber K. S. (2021) Blockchain-based self-sovereign identity: Survey, requirements, use-cases, and comparative study. In: *International Conf. on Web Intelligence and Intelligent Agent Technology.* ACM, pp. 128–135

OMG (2011) Business Process Model and Notation (BPMN). [url=https://www.omg.org/spec/BPMN/2.0/PDF/](https://www.omg.org/spec/BPMN/2.0/PDF/)

Pöhn D., Hommel W. (2020) An overview of limitations and approaches in identity management. In: *International Conf. on Availability, Reliability and Security*, pp. 1–10

Satybaldy A., Nowostawski M., Ellingsen J. (2020) Self-sovereign identity systems: Evaluation framework. In: *Privacy and Identity Management. Data for Better Living: AI and Privacy*, pp. 447–461

Sedlmeir J., Smethurst R., Rieger A., Fridgen G. (2021) Digital identities and verifiable credentials. In: *Business & Information Systems Engineering* 63(5), pp. 603–613

Siddiqui H., Idrees M., Gudymenko I., Quoc D. L., Fetzer C. (2021) Credentials as a Service Providing Self Sovereign Identity as a Cloud Service Using Trusted Execution Environments. In: *International Conf. on Cloud Engineering*, pp. 210–216

Sroor M., Hickman N., Kolehmainen T., Laatikainen G., Abrahamsson P. (2022) How modeling helps in developing self-sovereign identity governance framework: An experience report. In: *Procedia Computer Science 204 International Conf. on Industry Sciences and Computer Science Innovation*, pp. 267–277

Tobin A., Reed D. (2016) The inevitable rise of self-sovereign identity. In: *The Sovrin Foundation* 29(2016), p. 18

Tran A. B., Lu Q., Weber I. (2018) Lorikeet: A Model-Driven Engineering Tool for Blockchain-Based Business Process Execution and Asset Management. In: *Dissertation Award, Demonstration, and Industrial Track at BPM. CEUR Workshop Proceedings Vol. 2196.* CEUR-WS.org, pp. 56–60