Blockchain Empowered and Self-sovereign Access Control System

Hanif Tadjik
University of Stavanger
Stavanger, Norway
mohamad.h@hotmail.no

Jiahui Geng
University of Stavanger
Stavanger, Norway
jiahui.geng@uis.no

Martin Gilje Jaatun
University of Stavanger
Stavanger, Norway
martin.g.jaatun@uis.no

Chunming Rong
University of Stavanger
Stavanger, Norway
chunming.rong@uis.no

***Abstract*—Lack of trustworthiness, access policy flexibility, and user privacy preservation in centralized access control systems raise numerous security issues and reduce the collaboration maturity of global data sharing systems. In this paper, we propose a Self-Sovereign Identity-based, Decentralized, and Dynamic (SSIDD) access control system. SSIDD utilizes blockchain technologies to build trust for untrusted data sharing networks and ensures user privacy. Our access control provides high access policy flexibility and security for global inter-enterprise collaborations from a diverse industrial environment. SSIDD authenticates its users based on their Decentralized Identifiers (DID), which are under control of users and can be resolved into a DID document stored on the blockchain. Our data management technology keeps the data sharing systems safe against issues such as data breaches, identity thefts, and privacy violations. Besides, the authorization process of SSIDD is dynamic by adopting several smart contracts. The transparency of rules and agreements in smart contracts and the traceability of records on blockchain ledger provide a high level of security and trust. For proof of concept, we have developed and evaluated a prototype of SSIDD. Our evaluations show that the throughput and latency of our method are within an acceptable range.**

***Index Terms*—blockchain, access control, self-sovereign, DID, Web3.0**

I. Introduction

Nowadays, enterprises collaborate globally and exchange collections of data for different purposes, like scientific analysis and market research for providing better services to their customers. Conventional approaches for data exchange mainly rely on centralized access control systems, which have static and limited access policies and collect sensitive personal information for its authentication and authorization processes. Third-party resource (i.e. cloud) providers are often involved in handling collections of sensitive data. Moreover, because of a lack of accountability, transparency, and audibility in centralized access control systems, inter-organizational collaborations in untrusted networks are often complicated.

Non-flexible access policies lead to the immaturity of inter-enterprise collaboration in multinational or diverse industrial consortiums. Collecting sensitive personal information raises a potential risk of data leakage and privacy breaches. The intermediary computation resource involvements take the control of both personal and non-personal data from its owners, where the integrity of data is unsecured. Rules and agreements regarding data sharing systems are not transparent for all involved parties. Furthermore, with centralized access control systems, it is less likely to identify underlying issues for security flaws, and malicious behaviors are often left without being noticed [2] [6] [8] [16].

Our goal in this paper is to design and develop a decentralized access control system for inter-enterprise data sharing, in which the organizations involved are globally distributed, and share the data originated from diverse sources. Therefore, organizations must be able to decide on different access policies for resources they share in the network. Additionally, involved parties share sensitive data, which means that we need to prevent the data from any abuse. Furthermore, we require that our data sharing system preserve data privacy and avoid personal information leakage. More specifically the access controller must fulfill the following criteria:

- **Dynamic access policy**: Each organization must be able to dynamically customize its own access policy for resources shared in the network, aiming to promote the maturity in collaboration.
- **Self-sovereign**: To cope with data breaches and privacy-preserving violations, it must maintain data privacy by giving full control of data to its data owner.
- **Accountability and transparency**: All actions, rules, and agreements in the network, both previous and present, must be transparent for all parties. Additionally, the system must be able to trace organizations and their clients' interactions in the network, so that it overcomes the problem of trust between the parties.
- **Audibility, availability, and integrity**: it must track all attempts to access resources in the network and provide a robust and fail tolerant system. Additionally, the system must ensure that data is stored in a safe storage system where it cannot be modified maliciously.

We propose the SSIDD, a novel self-sovereign identity based, decentralized, and dynamic access control system. We have developed two main components for this system, which we call phase one and phase two, respectively. The overall aim of phase one is to ensure self-sovereignty, and phase two is to provide a dynamic access policy infrastructure.

Outline: In the following sections, we first present rel-

evant background knowledge of digital identities. Then, in Section III we introduce some of the existing access control systems used both in centralized and decentralized networks. Section IV describes the details of the design and architecture of SSIDD. Section V goes through the experimental evaluations of the smart contracts (SCs) of SSIDD. Section VI introduces some related research works and discusses our contributions. Finally, Section VII provides the conclusion of the paper and discusses future work.

II. Digital Identity

A. Tradition Digital Identities

A *digital identity* is an identification mechanism used to identify users on the internet. A digital identity can be a combination of username and password used to access a social media account, or in more complex cases, it can be a Norwegian bank ID [26] used to connect a user to her bank account. In the following, we discuss two different types of existing digital identity solutions [1]:

Centralized digital identities: Users can use their digital identities to authorize into a single application, where a social media account is an example. A centralized digital identity system manages the application and is controlled either by the owner of the social media application or its resource providers. In such systems, users can not use their accounts to authenticate into any other applications.

Federated centralized digital identities: Users can use one digital identity to authenticate to several somehow dependent but different systems. Either a single central system or a community of applications that uses the same identification mechanism manages users' identities. Taking the bank ID as the example, even though one centralized authority controls the Norwegian bank IDs, it enables users to authenticate into a few other bank and non-bank systems with the same identity provisioning requirements.

B. Limitations of Existing Digital Identities

User privacy: Users have no control over how and in what context a centralized system shares their personal identity information.

Central authority: User's personal information is controlled by centralized authorities that provide authorization services. Based on user-provided personal data, it denies or accepts access to systems, and decides how to use, share, and store data. Besides, most centralized authorities are not capable of evaluating fake identities, which is the source of many unethical behaviors on the internet nowadays.

Limited: Despite the more flexibility the federated identification provides to users, where issued digital identities give access to several platforms, it has many limitations. Firstly, the number of organizations that collaborate are just a few, are locally centralized, and provide highly dependent services. That is, a bank user in Norway can not authenticate for a bank system in the United State, or a social media authentication system is less likely to federate with a bank. Secondly, users still do not have control over their shared personal information. This becomes even worse if an application uses a third-party resource provider to store personal information. It will then not let the application have full control over its user data.

User experience: It is not obscure that the user experience of centralized digital identity systems is awful. Just thinking of how many passwords you need to remember to access your daily used applications. The trauma becomes even worse when you forget one password and need to go through the process again.

Hence, it is obvious that there is a need for a more robust and efficient digital identity mechanism that is proveable for any application on the internet. In the following we present a modern digital identity architecture that provides that flexibility.

C. Self-sovereign Identity

Since the emergence of web 3.0 [27], which started from the ideas behind the Ethereum protocol, research for digital identity systems has changed its direction toward decentralization. It is mostly because, the federation identity mechanisms, where third parties and central authorities are involved, will not result in a Self-sovereign Identity (SSI) system. An SSI system gives the control of personal data to its owner. It aims to preserve the security and privacy challenges that traditional digital identities undergo. To ensure user privacy and control over personal data, we need to eliminate centralization and third parties.

The goal of adopting Distributed Ledger Technology (DLT) systems is to not rely on any type of central authority to manage access or personal information of users. But with DLTs everything will be transparent for users, which gave them full control over how personal data is used, by who, and where. Data availability is another benefit of adapting DLT, where users get access to their data, modify them, and even remove them. The web 3.0 standards documentation provides a standard structure for the modern era digital identities, that is Decentralized Identifiers (DIDs). The structure of DID is as follows:

$$did :< method{-}name >:< method{-}specific{-}identifier >$$

The *method-name* is a unique domain for each organization, and the *method-specific-identifier* is a unique combination of numbers and letters within the method-name. The structure results in a globally unique DID. Moreover, it resolves to a DID document, which is stored on a DLT. Each DID document can include one or more Verifiable Credentials (VC). A VC is a sort of certificate that a user can be issued by an organization. For example, a diploma issued from a university can be registered as a VC. Every time a user claims that she holds a university diploma, she can refer to her VC stored on a public DLT.

The DID is a powerful mechanism to ensure self-sovereignty. However, in terms of the technical aspect, the number of challenges to obtaining its features are numerous. One of those challenges is having an efficient access control to a decentralized application that grants access based on DIDs while keeping both personal data and shared information secure [3] [13] [20].

III. Access Control

Access control (AC) mechanisms in traditional data sharing systems are often centralized and access to resources is usually managed through a third-party cloud provider. Despite many complex techniques and concepts used to regulate and restrict access to systems, relying on a central authority is both a reliability and security bottleneck in these kinds of systems. Attackers will have one target to attack, and in case of failure, an entire system can stop operating. However, since the evolution of adopting blockchain technologies in data sharing systems, we get closer and closer to realizing fully decentralized access control systems. In this section, we present and discuss some of the most commonly mentioned access control techniques.

1) Access Control List: An Access Control List (ACL) is the simplest and the most static access control technique. It consists of a list of network members along with their access permissions in a network. ACLs are static, which means that policies are defined as general purpose and do not specify any conditional cases. Moreover, ACL is used widely in centralized enterprise applications and IP networking systems as well as permissioned blockchain applications. However, as mentioned, a standard ACL mechanism has little flexibility for an environment with the need of dynamic access rules. Thus despite its simplicity, it will not be a good fit for environments that require more policy variability [14] [17].

2) Role Based Access Control: A Role-Based Access Control (RBAC) model controls access to a system by putting each *subject* (i.e. human, machines, etc.) in one or more predefined roles. Each role grants permission to access one or more specific network *objects* (i.e. resource(s)), which are predefined in a network's access policy. In RBAC models, only subjects that have been assigned a role have permission to access resources. Furthermore, the subject's role(s) is assigned based on their authorization level and their distinct role in the network. Finally, an RBAC system must make sure that subjects are granted permission to resources at the level of their authorization status [22]. RBAC is somehow more flexible than ACL in terms of access policies for organizational level access control systems. However, the technique does not provide any flexibility for inter-organizational systems.

3) Attribute Based Access Control: A role-based mechanism for access control does not fit the need of systems with more complex and dynamic access rules. Thus, based on this idea, Shamir [23] proposed an Identity-Based Encryption (IBE) method. In this technique, a user will be supplied by secret keys from a central authority, where each secret key is mapped directly to the identity of the user. The key is then used to encrypt data that a user wants to share. Other network members that hold a public key associated with the secret key can then decrypt the data. This method is robust in such a way that it ensures both data security and access control. But, still, it does not provide a totally fine-grained access control where other conditions can be applied to access a system or a set of data. Additionally, it leans on a trusted central authority system that authorizes users, which can be challenging in untrusted and decentralized applications.

Thus, an Attribute-Based Encryption (ABE) method is proposed by Sahai [21]. It is basically an extension of IBE where it takes steps further toward a more dynamic access control mechanism and replaces identities with attributes. In ABE, an attribute can be anything that characterizes a network participant, a set of data, or an environment. It provides high flexibility and dynamicity to specify access rules. Besides, the Attribute-Based Access Control (ABAC), which is an access control technique made of ABE, also takes care of security of data by its encryption techniques. Hence, it has wide use cases in both centralized and decentralized applications that require more complex and involved access rules [9].

However, even though an ABAC does not need a central authority for authorization of identities as in IBE, it still requires a central key management system that controls and provides cryptographic keys based on access policies, attributes, etc. Additionally, ABAC relies on many detailed operations and can add lots of overhead, which can worsen the performance of any system, including the more performance-critical systems, such as blockchain [5] [12] [28].

A. eXtensible Access Control Markup Language

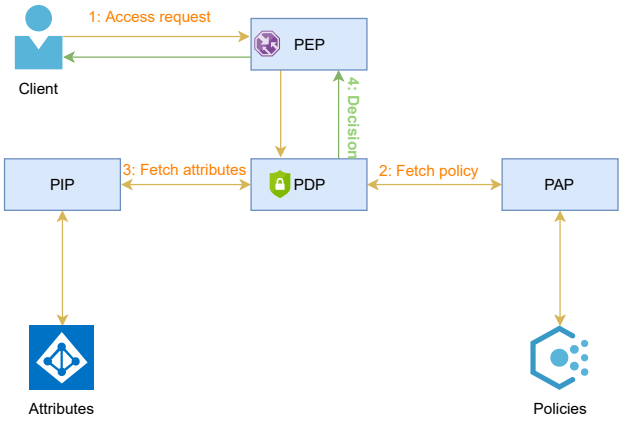


Fig. 1: An overview of the architecture of XACML.

ABAC consists of several different components and techniques that can be difficult to manage and have low flexibility to environmental changes. To overcome this issue, the eXtensible Access Control Markup Language (XACML) [24] technique was proposed. It is basically a decoupled ABAC system, where it separates policies from resource attributes in an organized manner. Figure 1 shows an example of an XACML access control architecture. Each component has the following responsibilities:

Policy Administration Point (PAP): One or more policies can be assigned to the system and managed through this component. Policies are represented as XML markups and contain policy ID, attributes, and rules fields. The attribute field defines a set of user and resource attributes to be determined for the access decision, and the rules assign the required state of each attribute.

Policy Information Point (PIP): Subject and object (i.e. users and resources, respectively) attributes are also stored in XML format. The PAP component manages these attributes and maps them to their associated policies.

Policy Decision Point (PDP): This is the main component of the XACML technique. Here relevant policy and attributes are fetched from PIP and PAP. Then based on the rules of the policy and corresponding subject and object attributes, the PDP can decide on an access request. The PDP forwards its decision to the gatekeeper (i.e. PEP) component.

Policy Enforcement Point (PEP): This component is the gatekeeper of the system. First, it redirects access requests from clients to PDP. Then, after the PDP component has made a decision, the PEP allows or denies access to requested resources.

By utilizing the XACML technique, the access control system will have high flexibility for integration into any type of application, both centralized and decentralized. The unique feature of XACML for decentralized applications is its decoupling of the access control mechanisms into several components where we can execute each part in a smart contract. Besides, it ensures fully dynamic access control for any single resource in the system by making it possible to assign a unique policy rule to each resource [4].

IV. Proposed Approach

We aim to build a self-sovereign and dynamic access control system by utilizing DLTs for untrusted networks. We have designed a novel solution that fulfills security and trustworthiness requirements in global data sharing systems. It consists of several components and technologies that work in parallel to provide an efficient and powerful access control system.

We have decoupled the workflow of SSIDD into two phases to make it more scalable and flexible. An SSI-based phase and a trust-oriented and dynamic phase. Thereby, we ensure that it is scalable for managing resources. Depending on the arising performance bottlenecks, and flexibility for using different platforms and frameworks in each of the two phases. Accordingly, based on use cases and interests and by keeping SSIDD's design as a baseline, it should be straightforward to apply different architectures to our SSIDD system. Furthermore, security is the other concept that we have carefully considered in our design. The SSIDD is an access control system that ensures user privacy, data integrity, availability, and accountability. We ensure the security from the time a client requests access until SSIDD makes an access decision. Figure 2 shows an overview of the architecture of SSIDD.

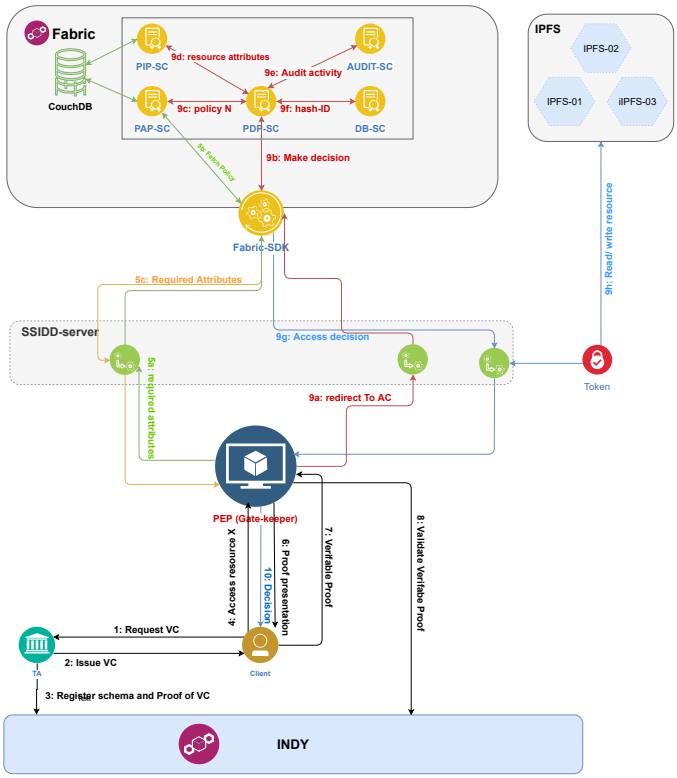


Fig. 2: An overview of the architecture of SSIDD. Note that the arrows presents workflow of an access request to **read** some resource X from the Inter Planetary File storge System (IPFS) .

It is a complex system that consists of several components. However, in the rest of this section, we will clarify the following: The idea behind the design, how the components interact with others, the purpose of having each component in each phase, and our security considerations in the design of SSIDD.

A. Phase One

In the first phase, we have four main components. An HLI ledger, a *Trust Anchor (TA)*, a *client*, and a *gatekeeper*. We use HLI as our DID platform. The TA is a component that we use to issue VCs to a client. The client component models an HLA agent that provides DIDComm services for users. In a practical situation, a TA can be an administrator of an organization of a data sharing network, and a client component can serve as a chief executive officer, an employee, or even an administrator of the organization. Moreover, the gatekeeper presents the PEP component of an XACML-based system, with the exception that it also operates as an HLA agent.

As presented in Fig. 3, the workflow of phase one starts from a TA offer and issues a VC to a client. Then, the client communicates with the gatekeeper and requests access to a resource. The gatekeeper receives the client's request and verifies that the client holds enough credentials to get permission to the network. Finally, after successful verification

of the required VC, the gatekeeper redirects the client further to the second phase of our access control system, where several communication steps are involved.

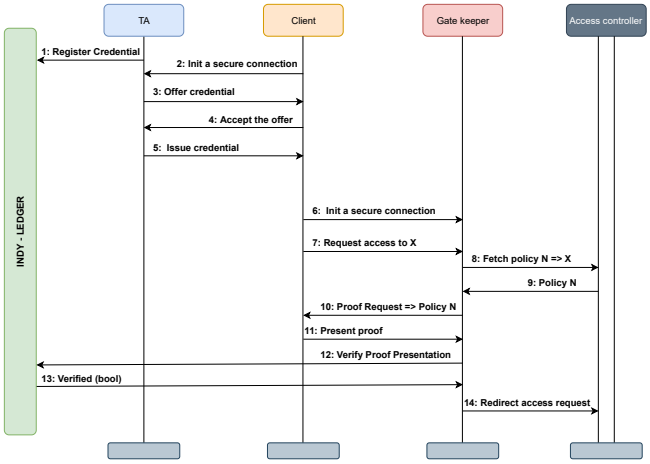


Fig. 3: First phase communication process of a read request.

B. Phase Two

Phase two of SSIDD is where the logic of our dynamic access control is located. As in the first phase, here we also have several components that communicate across different platforms. One of the core components of phase two is the SSIDD-server. In our case it is a Google Remote Procedure Call (gRPC) [11] server that communicates with the gatekeeper, HLF network(Hyperledger Fabric), and an off-chain database. It wraps client requests into a required format and forwards those to the HLF. Our SCs follows the XACML technique, which decouples the access control components and provides a dynamic system. However, our design has the following exceptions from an XACML model:

- We have a decentralized system; our access control components are SCs that are implemented in a transparent and decentralized network.
- A self-sovereign access control system; we do not store any sensitive client information in the system. We only store clients DID to enable auditing clients' activity in the network and ensure the security of the SSIDD. We simply count the number of times a user is banned and block her from making more requests. By doing this we avoid a Denial Of Service (DOS) attack where a system is halted by setting huge traffic against the network.
- We have additional components; as illustrated in Figure 4, we have an Audit-SC which is not a part of standard XACML. Besides, we have the database SC (DB-SC), which is isolated from the public network and invoked by the PDPSC to retrieve a Hash-ID that is associated with the off-chain database.

More specifically, we have five SCs that communicate with each other before making an access decision, which are as follows:

PIPSC: manages resource attributes.
PAPSC: manages policy administration.
DBSC: keeps track of transaction hashes offchain. Its functionalities are isolated from outside of network.
AuditSC: records invalid access request attempts.
PDPSC: makes access decisions based on information it retrieves from the other four contracts.

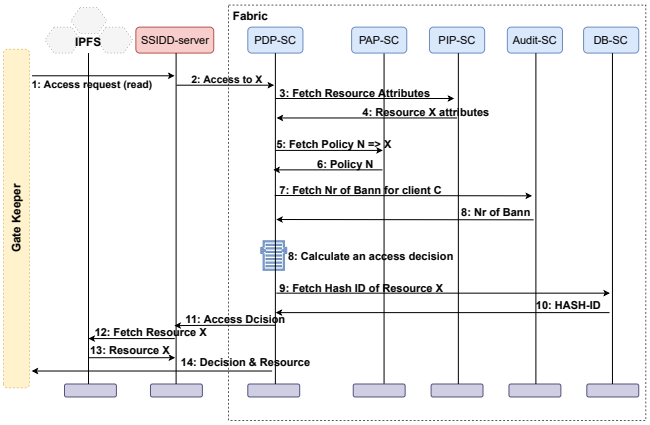


Fig. 4: Second phase communication process.

The downside of this design is that it will have reduced throughput and higher latency in comparison with centralized access control systems. But, the upside of our design is that by decoupling access control components, we are able to develop a self-sovereign and fully dynamic access control system. Moreover, we ensure transparency, trust, security, and accountability through our decentralized network.

It is worth mentioning that in this section, we have only presented the transaction flow for a resource read request. In our design, we have considered full Create, Read, Update, and Delete (CRUD) operations for accessing resources in addition to CRUD operations for managing policy objects and adding or removing organizations to the network. However, the design of all access requests will differ in small details. Thus, in general, a read request can present the whole idea of our SSIDD system.

V. Evaluation

In our experimental evaluations, we develop a Hyperledger Caliper [15] benchmark module for testing and benchmarking our chaincode. We test the performance of SCs with one *xlarge* VM, which runs in the local data center. In Table I we present details of our testing environment.

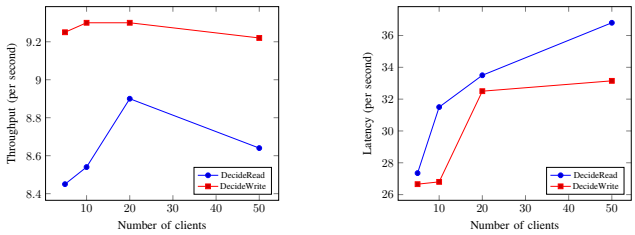
OS	Ubuntu bionic 18.04.6 LTS
8 x CPU	Intel(R) Xeon(R) CPU E5-2640 v4 2.40GHz
RAM	16 GiB
Docker	20.10.14
Go	1.17.9 linux/amd64
Fabric	2.4
Caliper	v0.5.0

TABLE I: Specification of our test environment.

Our HLF network consists of three peer nodes (each represents one organization), one orderer node, and a Fabric CA node. In the following, we present our benchmarks for

the DecideRead and DecideWrite methods from the PDPSC contract. We measure throughput and latency by changing the number of clients, block size, and batch timeout configurations. The first method makes an access decision on a read request. It reads a resource, policy, audit, and database transaction from the ledger's global state by invoking their respective contracts internally. Additionally, it evaluates policy rules against a user's provided attributes. Furthermore, the DecideWrite method makes an access decision on a write request, if a user wants to share a set of data or add a policy to the network. The method goes through the same processes, except that it does not invoke the PIPSC and DBSC contracts to read resources and database transactions from the global state. Thus, it goes through fewer steps than the DecideRead method.

Furthermore, we ran each of our experiments in three rounds. In the first round, we warm up the Caliper with 200 transactions. In each of the second and third rounds, we submit 2000 transactions with a transaction rate of 500 per second (tps). The benchmark result that we present in this section is the average of the two last rounds.

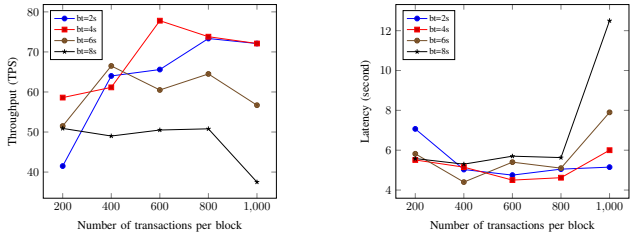


(a) (b)

Fig. 5: Throughput and latency of DecideRead and DecideWrite of PDPSC contract in terms of number of clients with 1000 tps.

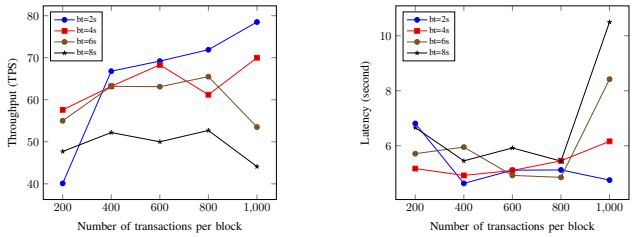
Figure 5 shows that when the number of clients increases, we observe that throughput also increases. The increase contiues until we have 20 clients. After that, it starts to degrade. The latency does also show an increase by increasing the number of clients. The overall performance is acceptable for a blockchain application. But, for this set of tests, we have used default values for block size and batch timeout configurations of HLF, which has a maximum of 10 transactions for block size and 2 seconds for the batch timeout. These metrics have key roles in the performance of a blockchain network. In the following, we present our result of changing the block sizes and batch timeouts.

Finding an appropriate block size is critical for any permissioned blockchain network, and HLF is no exception. Larger blocks take more channel bandwidth, require more computational resources, and transactions need to wait longer to be confirmed. Batch timeouts also have somehow the same role. As shown in Figure 6 and Figure 7, we have tested several block sizes and batch timeout configurations to fine-tune the performance of our blockchain network. In the first test, we jump from the default HLF configuration of 10 transactions per block to 200 transactions. The performance shows a significant improvement compared with our result in Figure 5. However, for the DecideRead method, we reach the global maxima with a block size of 600 and a batch timeout of 4 seconds. The DecideWrite method goes through fewer steps and the 2 second batch timeout seems to be the best option. However, as the latency of this method is lower, the throughput still increases even after 1000 transactions per block. Besides, the latency also does not increase extensively.



(a) (b)

Fig. 6: Throughput and latency of DecideRead in terms of block size and batch timeout (bt).



(a) (b)

Fig. 7: Throughput and latency of DecideWrite in terms of block size and batch timeout (bt).

A. *Security Evaluation*

Starting from the phase one component of the architecture of SSIDD, we use DID based communications to ensure user privacy. But we do also ensure that SSIDD is secure against:

Data breach and information leakage: With our efficient data management technique, we do not collect personal information on our servers. Accordingly, we will not be a target for attackers aiming to steal personal information.

Privacy violations: There is no way that we violate someone's privacy since the personal data is immediately removed from our servers after the SSIDD makes an access decision.

Identity theft: Each client has a unique DID store in the HLI ledger that we can recover if it is lost or faked by anyone else. Besides, clients' identity attributes and credentials are held secured by cryptographic keys,

which makes it infeasible for someone to access DID and credentials without holding the keys.

Moreover, in phase two, the SSIDD-server communications with other components must happen through secure protocols, such as Transport Layer Secuirty (TLS) [25]. With TLS, communications are encrypted after an efficient handshake mechanism. Thus, it ensures that information transmitted from the different components is secure and immutable.

Moving further to our smart contracts and HLF, here we ensure several potential security vulnerabilities that can occur with any other centralized system, among others:

Accountability and auditability: With the traceability of the HLF ledger, where all state changes are recorded, we ensure that every action in the network is recorded and transparent for all parties. Thus, any malicious behavior can easily be tracked.

Availability and single point of failure: The system relies on a CFT consensus protocol. Therefore, in case of any failures, we still have available servers to provide services. Besides, we cope with the single point of failure problem, which is one of the drawbacks of most centralized networks. However, this depends on the computing resources available.

Data integrity: Records on the HLF ledger are immutable. Besides, for the offchain storage, we ensure integrity by using IPFS, which uses a blockchain based data structure to store data. Additionally, we submit a hash of offchain data to our HLF ledger, which can also ensure the integrity of data in case anything happens to IPFS or if replacing it with any other storage system.

Finally, we want to add that the transparency of our smart contracts and the two blockchain ledgers, adds a unique level of security and trustworthiness in SSIDD.

VI. Related Work

Deters et al. [19] propose a distributed ABAC system based on XACML architecture for a permissioned blockchain network. The proposed system utilizes the decoupling feature of the XACML method and Smart Contract (SC) to develop a robust and dynamic access control. The three main SCs are PIP, PAP, and PDP, which manage resource and user attributes, policies, and a final decision, respectively. They use the Hyperledger Fabric (HLF) as their permissioned blockchain system, and store policies and attributes onchain in the form of JSON objects. The blockchain is used as an access control for an off-chain organization. Upon an access request from a client, the offchain organization routes the request to the HLF network. Then after the PDP has made an access decision, it redirects its decision to the offchain platform. The evaluation results [19] show that the proposed technique is efficient both in terms of security and performance. However, user attributes are stored in the blockchain, while users have no control over them. Collecting user data, specifically onchain, is the most critical part of this approach.

Our solution is similar to this approach in such a way that we also use the XACML approach to ensure dynamicity, and use blockchain technology to ensure transparency, auditability, and trust. However, our approach is different in several cases. First, we ensure the integrity of resources stored offchain by implementing an additional smart contract that traces the offchain transactions. Next, we also ensure to block users from making more requests if a limited number of illegal access requests is reached. We manage this functionality by implementing a smart contract that records users' illegal activities. Lastly, the most important difference is that we ensure user privacy and do not collect personal information either onchain or offchain. Our access control system consists of two parts. The first part ensures policy dynamicity, data security, and trust, and the other part to ensures user privacy and preserves systems from potential security vulnerabilities that arise because of collecting user data.

SISBAC is an access control system based on SSI and XACML proposed by Belchior et al. [4]. The system uses HLI and Hyperledger Aries (HLA) for VC creation and approval, and runs the XACML components in a centralized system. It focuses on user privacy and control of data by adopting an SSI based access control system. Thus, users can control their personal sensitive data in their DID wallets. Upon a user's request for access to resources, the system provides a requirement schema in form of a Verifiable Presentation (VP) request and the user can decide what to share. The system executes a set of operations in centralized XACML components and decides on access requests. The system is highly efficient in terms of ensuring user privacy, user control over personal data, and the dynamicity of the access control policies. The performance bottleneck of the system lies in the process of connection between the user and a verifier component which runs as HLA agents. Otherwise, it shows high throughput and an acceptable range of latency.

The most important difference between SISBAC and our approach is that we utilize decentralized technologies for the implementation of XACML. e ensure that:

1) Our access control system is adaptable for any untrusted network.
2) By tracing a record of all activities in a decentralized ledger, we ensure that users are accountable for their behaviors in the network.
3) We guarantee that policy rules and agreements are transparent for all parties.
4) In decentralized systems, based on Crash Failure Tolerance (CFT) and Byzantine FailureTolerance (BFT) consensus algorithms, if one server fails, the rest of the network can continue operating, i.e., there is no single point of failure problem in our proposed system.
5) We ensure integrity of our resources stored on- and off-chain.

The other noteworthy difference is our additional components in form of SCs, which ensures data integrity and preserves unnecessary and maliciously high workloads against the network.

Ding et al. [7] propose an RBAC method for federated data sharing systems running in permission decentralized environ-

ments. The proposed technique is similar to an ABAC method in such a way that they use the concept of attributes but in a unique manner. In their federated system, a user has acquired a set of dynamic attributes represented in binary upon her role in an organization. After requesting the access to the data-sharing system, SCs evaluate the attributes and the level of authority of the user, and if valid, she gets access to the resources she is allowed to. The difference between this method and an ABAC is that any cryptographic key mechanism and consequently, any symmetric encryption method is not used. Removing the cryptographic operations can indeed increase the performance of the network. Additionally, the proposed technique, they do not rely on a central key management system, which has significant importance in decentralized systems. However, the difference between this approach and SSIDD is that it does not ensure user privacy. Besides, the access control is not flexible for inter-organizational systems.

Rong et al. proposed an approach named OpenIaC, which aims to provide services based on the principles of Zero Trust Architecture (ZTA) among the federation of connected resources based on Decentralized Identity (DID) [18]. Our proposed approach complies with the principles of OpenIaC, and supports fine-grained access control for shared resources managed by blockchain. Geng [10] proposed the use of blockchain as an access management infrastructure for federated learning systems.

VII. Conclusion and Future Work

In this paper, we have proposed the SSIDD, a blockchain empowered and self-sovereign access control system. Our systems rely on two DLTs, where for our proof-of-concept we used the Hyperledger Indy (HLI) and HLF, respectively. We used HLI to ensure user privacy and avoid personal data security issues such as privacy breaches. To authenticate the users, we do not collect personal information on our system. We utilized the HLF to provide a secure, trustable, and flexible access policy architecture, where we develop several SCs each with a unique purpose. Additionally, the decoupled access policy infrastructure that we developed in our SCs, eases the management of the dynamicity of the SSIDD. Our experimental evaluations show that we can increase the performance of our SCs by reconfiguring the HLF network's block size and batch timeout configurations. However, it should be based on the expected workload of an application that utilizes our access control system.

References

[1] Christopher Allen. The path to self-sovereign identity. *Life with Alacrity*, 2016.

[2] Fayez Hussain Alqahtani. Developing an information security policy: A case study approach. *Procedia Computer Science*, 124:691–697, 2017.

[3] Oscar Avellaneda, Alan Bachmann, Abbie Barbir, Joni Brenan, Pamela Dingle, Kim Hamilton Duffy, Eve Maler, Drummond Reed, and Manu Sporny. Decentralized identity: Where did it come from and where is it going? *IEEE Communications Standards Magazine*, 3(4):10–13, 2019.

[4] Rafael Belchior, Benedikt Putz, Guenther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. Ssibac: self-sovereign identity based access control. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1935–1943. IEEE, 2020.

[5] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 321–334. IEEE Computer Society, 2007.

[6] Luc Dandurand and Oscar Serrano Serrano. Towards improved cyber security information sharing. In Karlis Podins, Jan Stinissen, and Markus Maybaum, editors, *5th International Conference on Cyber Conflict, CyCon 2013, Tallinn, Estonia, June 4-7, 2013*, pages 1–16. IEEE, 2013.

[7] Yan Ding, Liaoliao Feng, Ying Qin, Chenlin Huang, Pan Dong, Long Gao, and Yusong Tan. Blockchain-based access control mechanism of federated data sharing system. In *2020 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 277–284, 2020.

[8] Josep Domingo-Ferrer, Oriol Farràs, Jordi Ribes-González, and David Sánchez. Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Comput. Commun.*, 140-141:38–60, 2019.

[9] Chun-I Fan, Yi-Fan Tseng, and Chih-Wen Lin. Attribute-based encryption from identity-based encryption. *IACR Cryptol. ePrint Arch.*, page 219, 2017.

[10] Jiahui Geng, Neel Kanwal, Martin Gilje Jaatun, and Chunming Rong. Did-efed: Facilitating federated learning as a service with decentralized identities. In *Evaluation and Assessment in Software Engineering*, pages 329–335. 2021.

[11] Introduction to grpc. [online]. https://grpc.io/docs/what-is-grpc/introduction/, 2022. Accessed: 2022-02-16.

[12] Lifeng Guo, Xiaoli Yang, and Wei-Chuen Yau. TABE-DAC: efficient traceable attribute-based encryption scheme with dynamic access control based on blockchain. *IEEE Access*, 9:8479–8490, 2021.

[13] Jim Hendler. Web 3.0 emerging. *Computer*, 42(1):111–113, 2009.

[14] Access control lists (acl). [online]. https://hyperledger-fabric.readthedocs.io/en/release-2.2/access_control.html, 2022. Accessed: 2022-04-25.

[15] Measuring blockchain performance with hyperledger caliper. [online]. https://www.hyperledger.org/blog/2018/03/19/measuring-blockchain-performance-with-hyperledger-caliper, 2018. Accessed: 2022-04-22.

[16] Siani Pearson and Azzedine Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing, Second International Conference, CloudCom 2010, November 30 - December 3, 2010, Indianapolis, Indiana, USA, Proceedings*, pages 693–702. IEEE Computer Society, 2010.

[17] Jiang Qian, Susan Hinrichs, and Klara Nahrstedt. Acla: A framework for access control list (acl) analysis and optimization. In *Communications and Multimedia Security Issues of the New Century*, pages 197–211. Springer, 2001.

[18] Chunming Rong, Jiahui Geng, Thomas J Hacker, Haakon Bryhni, and Martin Gilje Jaatun. OpenIaC: open infrastructure as code – the network is my computer. *Journal of Cloud Computing*, 11(1):1–13, 2022.

[19] Sara Rouhani, Rafael Belchior, Rui S Cruz, and Ralph Deters. Distributed attribute-based access control system using permissioned blockchain. *World Wide Web*, 24(5):1617–1644, 2021.

[20] Riaan Rudman and Rikus Bruwer. Defining web 3.0: opportunities and challenges. *The Electronic Library*, 2016.

[21] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 457–473. Springer, 2005.

[22] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[23] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[24] OASIS Standard. extensible access control markup language (xacml) version 3.0. *A:(22 January 2013). URl: http://docs. oasis-open. org/xacml/3.0/xacml-3.0-core-spec-os-en. html*, 2013.

[25] Sean Turner. Transport layer security. *IEEE Internet Computing*, 18(6):60–63, 2014.

[26] YOU. DIGITALLY - bank id. [online]. https://www.bankid.no/en/private/, 2022. Accessed: 2022-04-22.

[27] Standards. [online]. https://www.w3.org/standards/, 2021. Accessed: 2022-04-25.

[28] Shangping Wang, Yinglong Zhang, and Yaling Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, 6:38437–38450, 2018.