Decentralized Access Control using Hyperledger Fabric

by

Jubayer Hossain
24341112
Mehedi Hasan Nabil
21101203
Farhan Labib Jahin
21101204

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Jubayer Hossain
24341112

Mehedi Hasan Nabil
21101203

Farhan Labib Jahin
21101204

Approval

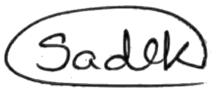
The thesis titled "Decentralized Access Control using Hyperledger Fabric" submitted by

1. Jubayer Hossain (24341112)
2. Mehedi Hasan Nabil (21101203)
3. Farhan Labib Jahin (21101204)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 22, 2024.

Examining Committee:

Supervisor:
(Member)



Dr. Md Sadek Ferdous

Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

In traditional access control systems, all the access control mechanisms are centrally managed which is seriously vulnerable. It is susceptible to a single point of failure due to its centralized architecture. As the system security breaks down due to the compromised central authority, it will be a huge risk, opening the door for data breaches, illegal access, and exploitation of private data. This research mitigates these risks by suggesting the decentralized control of access control systems using Extensible Access Control Markup Language (XACML). It is appropriate to use XACML for this task because XACML is flexible, open source, and works well in compliance with many access control models. This research focuses on decentralizing the four components of XACML: Policy Enforcement Point, Policy Decision Point, Policy Administration Point and Policy Information Point via the incorporation of Hyperledger Fabric (HF), a permissioned blockchain system. In the proposed architecture, the access control is distributed by smart contracts or chaincodes in multiple nodes of the network eliminating the single point of failure. To evaluate the feasibility of implementation, the development of the system following the proposed architecture is also done using chaincode. The results from the test evaluation show that decentralized implementation of the four XACML components with the Hyperledger Fabric eliminates single point of failure, scalability issues, and data integrity in distributed systems. The decentralization of the XACML components will help to create a secure and decentralized access control architecture. This research lays the foundation for future investigation of strategic blockchain-based decentralized access control systems.

Keywords: Access Control; Decentralization; Hyperledger Fabric; XACML; Blockchain; Security; Single Point of Failure (SPOF); Smart Contracts; Chaincodes

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor Dr. Md Sadek Ferdous for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$ABAC$ Attribute-Based Access Control

ACL Access Control Lists

ART Average Response Time

BP Business Process

$BRTA$ Bangladesh Road Transport Authority

$DDos$ Distributed Denial-of-Service

$DESCO$ Dhaka Electric Supply Company Limited

DoS Denial of Service

DSR Design Science Research

FR Functional Requirements

IdP Identity Provider

IoT Internet of Things

$NGAC$ Next Generation Access Control

PAP Policy Administration Point

PDP Policy Decision Point

PDP Policy Decision Point

PEP Policy Enforcement Point

PIP Policy Information Point

$RBAC$ Role-Based Access Control

RPC Remote Procedure Call

$SAML$ Security Assertion Markup Language

$SPOF$ Single Point of Failure

SR Security Requirements

SSO Single Sign-On

TEE Trusted Execution Environment

$XACML$ Extensible Access Control Markup Language

XML eXtensible Markup Language

XXE XML External Entitites

Chapter 1

Introduction

1.1 Motivation

In today's modern world, our daily life is becoming more and more inextricably connected to technology. Almost everything is connected and operating using some kind of technological development. We can not think of a day without taking help from it. For ensuring seamless operation of such systems, it needs a lot of data to process and generate very specific profiles for each individual which brings us to a very vital concern; security. Security issues include personal details about an individual, banking details, business details, and many more. Anyone with some particular set of skills and knowledge can try and cause security breaches on any available system, whether that be private or public. According to Deloitte's global survey, cybersecurity is the top concern among people using digital assets or services shown in the graph Figure 1.1 below [17]. These security breaches can be intentional attacks by anyone inside or outside the network. It can also be due to some unintentional vulnerabilities of any particular node with authorized access to the system. Either way, it is a violation of human rights as everyone has their own privacy, and putting trust in a system that is vulnerable to such things is also morally unethical.

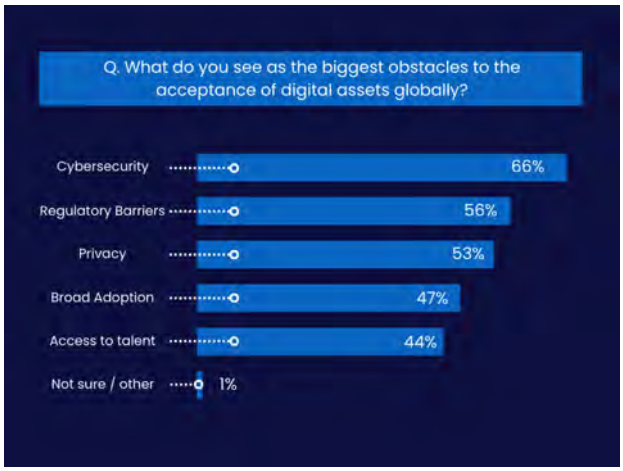


Figure 1.1: Percentage of people who feels these as obstacles when using digital assets globally

According to a news article by the Daily Star, last year in March alone, 4 different data breaches took place, including attacks on law enforcement, government agencies, and security forces [30]. These incidents leaked a massive amount of private data along with some confidential data too. Also last year in November, a huge amount of money was hacked from BRTA and DESCO where the hackers claimed to gain access to the servers without authorizing through the payment gateways and changed certain vehicle numbers illegally [29]. Hackers use different techniques to gain access to any system and it is even easier when there is a single point of failure. There are a lot of attacks that hackers can use in order to gain unauthorized access.

1.2 Problem Statement

Centralized access control models are commonly used in different types of systems and organizations. For various reasons using these models is very important but due to the centralization, it faces many challenges. A single point of failure can happen because of a centralized access control system. Access control might be completely blocked if the central authority is breached. Also, unwanted access can be granted because of this single authority. If the policy administration point (PAP), a component of XACML, is down for some reason, the central component policy decision point (PDP) will also be unresponsive which may break down the whole system. If the central authority is attacked, the organization will be at risk and there will be unwanted access which can lead to data leaks and more. Though single points of failure are one of the main problems of centralized models, there are other problems like scalability issues, performance issues, security problems, and dependency. Unauthorized accesses are not only the major problem here in some organizations but also authorized access can abuse their power and this can happen because of centralized structure. If the control was divided into multiple nodes instead of a central one, the attacker would have a much harder time breaching the system. However, we can solve this centralization problem by making it decentralized. A blockchain-based decentralized control system can solve this problem by implementing chaincode of the PEP, PDP, PAP and PIP which will spread access control all over the network and it will remove single points of failure. On the other hand, it may improve scalability, flexibility and transparency. We illustrated the theory in Figure 1.2. Though there are many benefits of using blockchain and access control models together, we have to face some challenges also. Consensus, private key management and scalability are the main challenges if we want to combine blockchain and access control models.

1.3 Research Objectives

In this contemporary era, data security has become exceptionally important and is considered the gold of this era. Despite access control models' dependability and significant contribution to the evolution of data security and authorized access, they are not impervious to the potential risks associated with a single point of failure (SPOF).

- **Research Objective 1 (RO1): Explaining the problem that arises from using a centralized system for access control and looking for**

an option that will decentralize this problem. The first objective is a critical analysis of the disadvantages of the centralized models and the threats including SPOF. From these vulnerabilities, the aim is to make a decentralized control system that will prevent those vulnerabilities.

- **Research Objective 2 (RO2): Propose a decentralized model that uses Hyperledger Fabric to solve security concerns like a single point of failure in conventional centralized access control systems.** The objective is to solve the security issues described in RO1, and it is achieved through decentralizing control and preventing any node from having all power.
- **Research Objective 3 (RO3): Securing and improving the visibility of access control policies by implementing chaincode.** The third goal is therefore to improve the dependability of the access control policies by integrating all the chaincodes of the PEP, the PDP, the PAP, and the PIP. These components should be disaggregated across these nodes.
- **Research Objective 4 (RO4): The performance and scalability analysis of actual implementations under different conditions for real-world applicability of the proposed architecture.** The last one is to evaluate the applicability and performance of this decentralized model and its scalability for big-scale applications.

While it may not guarantee a system that cannot be easily fooled, it is still advantages over any other current system. This research aims to achieve a new standard of access control by taking advantage of Hyperledger Fabric overcoming SPOF.

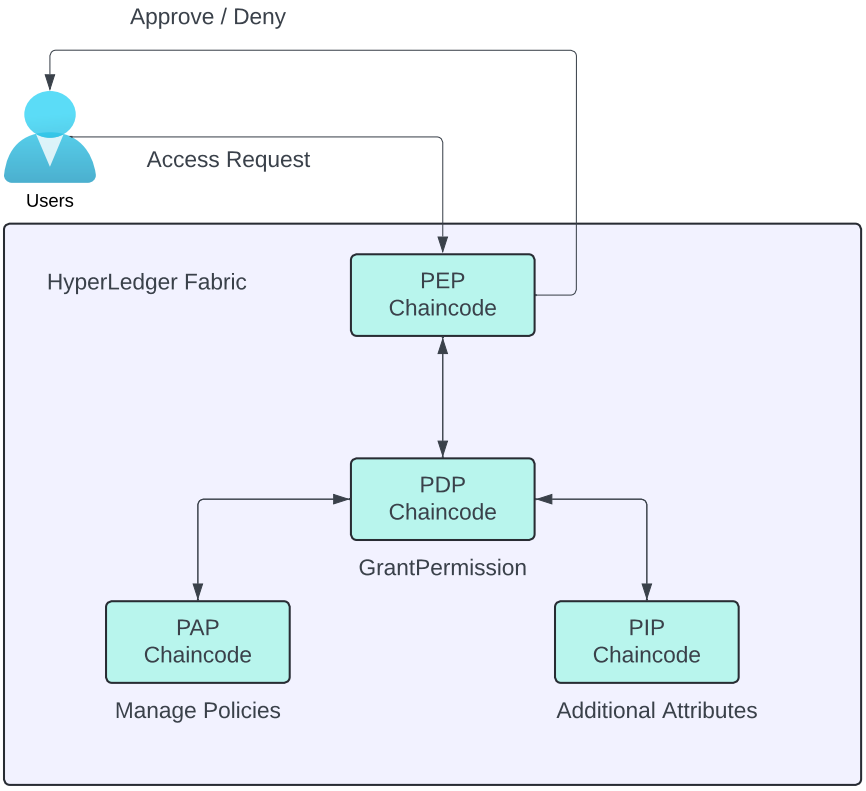


Figure 1.2: A decentralized XACML model

1.4 Report Structure

In this section, we will discuss about the structure of our report.

- **Chapter 1 : Introduction**
  This chapter provides an overview of the research motivation, problem statement, research objectives and report structure.
- **Chapter 2 : Background**
  In this chapter, there is a detailed explanation of different types of access control systems and different technologies related to this research.
- **Chapter 3 : Literature Review**
  The literature review describes previous research papers that are related to this topic.
- **Chapter 4 : Proposal**
  This chapter presents the system proposal, methodology, threat modelling and requirement analysis.
- **Chapter 5 : Architecture, Use-Case and Protocol Flow**
  This chapter contains the architecture of the proposed system along with the use-case and protocol flow.
- **Chapter 6 : Implementation and Performance Analysis**
  This chapter presents the process of implementing the proposed system along with its performance evaluation.
- **Chapter 7 : Discussion**
  In this chapter, the functional and security requirements are analysed. Research objectives are assessed along with the discussion of advantages, challenges or limitations and future works.
- **Chapter 8 : Conclusion**
  This chapter highlights the main outcomes and provides final thoughts on the significance of this research.

Chapter 2

Background

2.1 Access control and types

Access control is a procedure to check if a person should get access or not in a specific data or area. This whole procedure works depending on different systems and policies. A user can get access depending on different systems such as role, attribute, and rules. There are mainly four parts of this whole mechanism.

- **Identification:** Identifying the user responsible for performing actions
- **Subject Authentication:** Refers to checking the identity of the user beholding certain roles
- **Authorization:** Checking if the user has the given authority to perform certain actions related to their role
- **Access Decision:** Determines whether to accept or reject the request

When we make an access control system depending on the role called Role-Based Access Control (RBAC) which is a widely used system and there are also other access control systems like Attribute-Based Access Control (ABAC). There are a lot of access control systems around the world that are used based on their use cases and advantages.

2.1.1 RBAC

Role-Based Access Control is referred to as RBAC. It is a security model that limits authorized users' access to the system. According to RBAC, users are assigned to roles that correspond with their access permissions [1]. Rather than giving users explicit permissions, access is given according to a user's roles within an organization. So when a user tries to access any data the system will check if the user has that role which is mandatory to preview that data and if the user has that role the system will grant him permission otherwise it will reject his request. That's how RBAC works in different organizations. For RBAC there are three primary rules which are role assignment, role authorization, and permission authorization.

Role Assignment: A person can only exercise permission if they have been given the appropriate role.

Role Authorization: The person has to have authorization to play an active role. This rule, together with rule 1 above, guarantees that users can only assume roles for which they are authorized.

Permission Authorization: A person may only exercise permission if it is authorized for the person's active role. With rules 1 and 2, this rule guarantees that users can only exercise permissions for which they have been authorized.

RBAC also has a concept of hierarchies in which higher-level roles inherit the permissions of lower-level roles.

2.1.2 ABAC

Attribute-based access Control also prevents unauthorized access but instead of using roles it uses different types of attributes to check permission. This model is very useful for a complex environment. different organizations can use this model selecting different attributes for their system and data [25]. ABAC can be customized using several attributes to establish access control policies in various environments. Unlike RBAC, which provides roles with defined privileges and topics, ABAC can describe complicated rule sets that evaluate multiple attributes.

2.2 XML

XML, also known as Extensible Markup Language, is a widely used markup language intended for presenting structured data in a format that is both human and machine-readable [34]. It was designed to be used as a standard for data interchange between humans and machines. XML is also a platform independent, it is that we can use it in any software and hardware environments. It is extensively used for transmitting data over the internet, specifically, in the development of Web services for which it provides an underlying basis of some of the protocols such as Simple Object Access Protocol. In addition, XML plays an important role within application as configuration data store, XML configuration files and data exchange format like,Office Application XML, RSS parsing and many more. Due to its ability of storing data in more than one dimension they are suitable for use in fields with basic as well as complex data structure such as financial and health.

2.3 XACML

Extensible Access Control Markup Language or XACML is a standard access control framework that specifies a language for expressing access requests in connection with policies as well as a processing model. It is intended especially for authorization and access control in computer systems and applications. There are four core components of XACML:

- **Policy Enforcement Point (PEP):** The interceptor between PDP and end user. This component Just receives request from the user and forward them to PDP.

- **Policy Decision Point (PDP):** The component decides whether a user should be allowed or denied based on the policies and attributes of PAP and PIP.
- **Policy Administration Point (PAP):** The authority that creates, manages, and updates the access control policies.
- **Policy Information Point (PIP):** The component that provides the necessary information, such as user attributes or roles to help the PDP make access decisions.

Figure 2.1 shows the Extensible Access Control Markup Language (XACML) architecture and data flow. The XACML model will work in the following steps:

i **Define Policies:** Policy Administration Point is responsible for managing policies and giving them to PDP when requested. These policies contain the decisions for a particular user according to the attributes and resources.

ii **User Request:** A user sends an access request to the PEP.

iii **Initiate Request:** PEP forwards the request to the Context Handler in native format. This format may include any attributes or the role, resource, action, or environment.

iv **Create Context & Send to PDP:** An XML format of the request is made by the context handler and sent to the PDP.

v **Request Additional Data:** The PDP asks for additional data or information from the context handler such as additional attributes, subjects, or actions.

vi **Retrieve Characteristics:** The context handler requests additional data from PIP.

vii **Retrieve Data:** PIP retrieves the additional information from subjects, resources and environment.

viii **Return Characteristics:** The data flows from PIP to the Context handler.

ix **Update Context:** Optionally, the context handler puts the information in the context.

x **Send to PDP:** The context handler sends the required characteristics asked by the PDP initially.

xi **Evaluate Policy, Return Decision:** The PDP reviews the policy and then returns the response along with the decision to the Context Handler.

xii **Convert Decision, Send to PEP:** The response from the PDP is converted back to the native response format and returned to the PEP.

xiii **Allow/Deny Access:** If access is allowed then the PEP allows access to the resource otherwise it denies access. The PEP usually fulfills the obligations, for both cases.

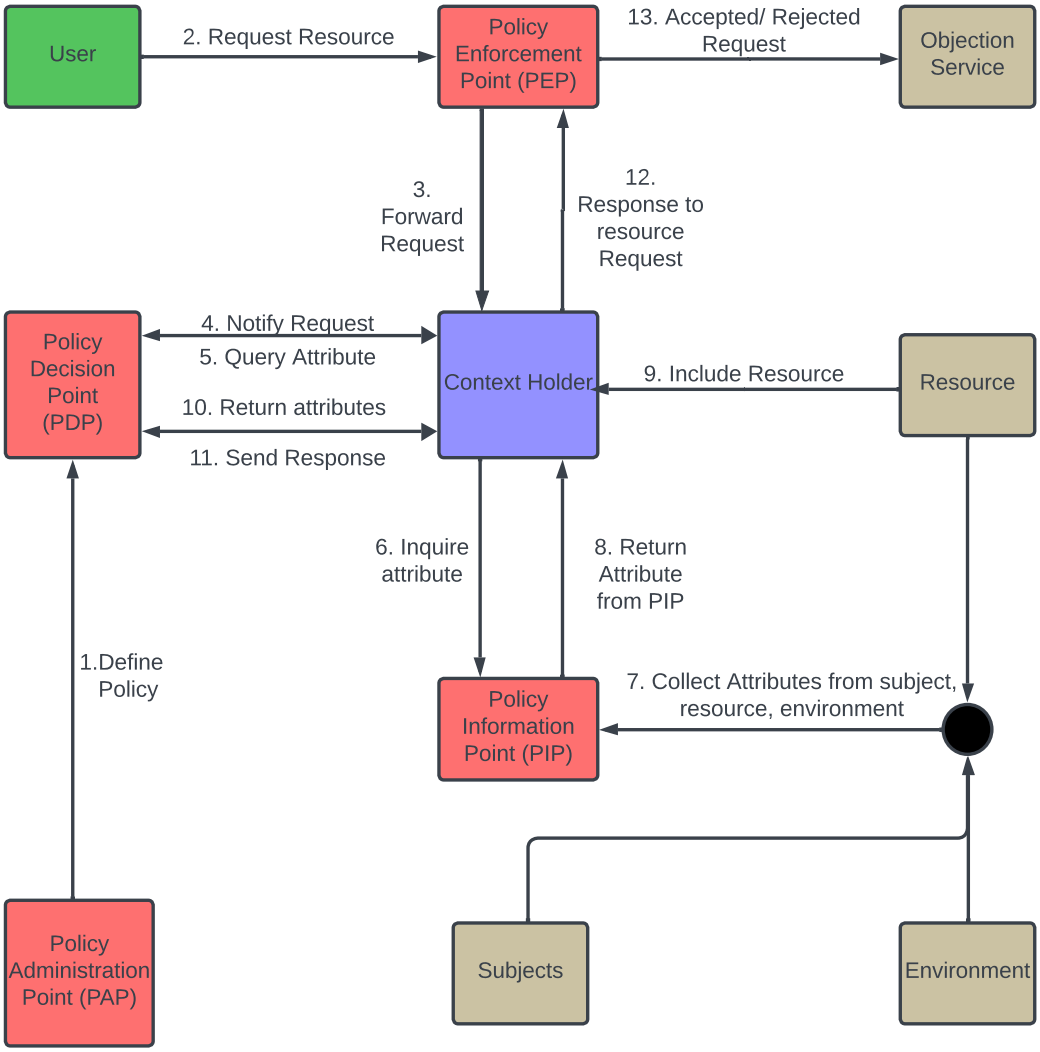


Figure 2.1: XACML Data Flow and Architecture

2.4 SAML

The Security Assertion Markup Language or SAML, is an open standard language that relies on Extensible Markup Language XML and is designed to facilitate a fluid yet secured authentication and authorization data among parties [32]. It works especially between the identity provider (IdP) and the service provider (SP). SAML uses SSO which allows the users to access multiple files or applications with a single sign-in.

- **Identity Provider (IDP):** The identity provider is responsible for verifying the identity and fetching the corresponding details of the providing services for a particular user [16].
- **Single Sign-On (SSO):** SAML uses SSO which enables users to sign in just a single time with the credentials and access multiple service providers without the necessity to log in for each application individually [33].

Figure 2.2 illustrates the architecture of SAML. This architecture model shows the steps in a SAML authentication flow. Here is a brief overview of each step

i **SAML Request:** The requestor/user sends the SAML request to the SAML Authority/IdP and this instantly initiates the requestor's authentication.

ii **SAML Response with Assertions:** The authority after validating the request, sends the requestor a SAML response containing an assertion.

iii **Send Assertions:** The Requestor/User sends the assertions to the web server.

iv **Validating the Assertions:** Then the web server forwards the assertion to the SAML authority/IdP for authenticating them

v **Confirmation/Rejection Assertion:** The authority sends back the authentication result to the web server. Before giving the user permission to access the web server, this statement will be taken into account

Based on the result the user will be granted access or rejected access in the web server. By doing this sequence of operations, SAML provides a strong foundation for secure interaction, allowing smooth authorization and authentication across different systems in a web-based SSO environment.

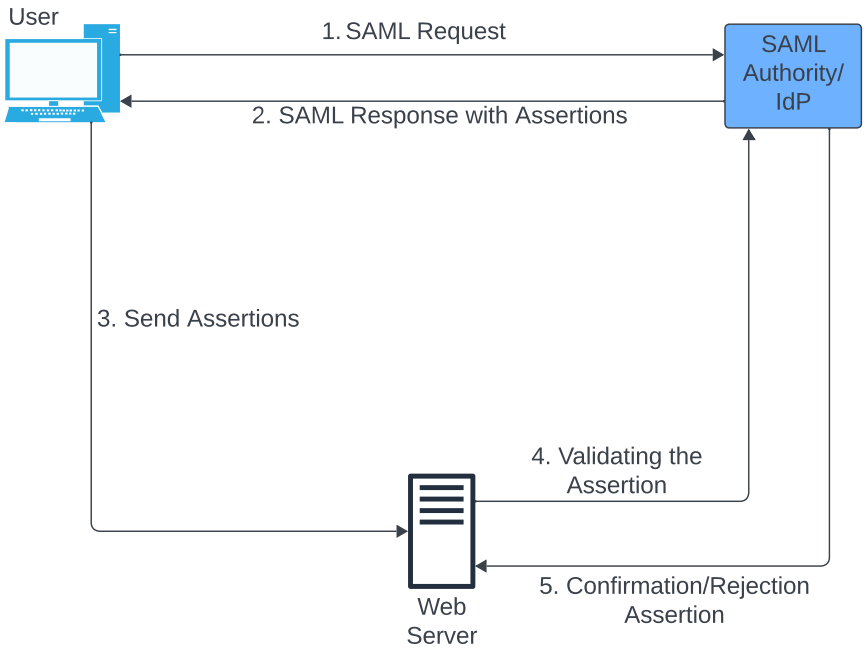


Figure 2.2: SAML Model

2.5 Blockchain

Blockchain is a distributed, decentralized ledger made up of blocks representing transactions [31]. Every member of a blockchain independently updates the distributed ledger. Every blockchain node is made up of miners and users. Users carry out transactions, and miners produce, verify, and reject blocks. On a blockchain, data ownership details, award histories, payment records, and contracts may be

kept. In a blockchain, every "block" provides a hash of the block before it. If a block is altered even slightly, the chain could be broken due to mismatched hashes. The longer chain is deemed to be the real one, and the shorter one is removed. Block hashes are generated using cryptographic hash functions. The function takes in input and outputs a distinct outcome.

2.5.1 Hyperledger Fabric

Hyperledger Fabric is basically a blockchain framework that is modular. It is utilized as the base foundation in any kind of blockchain based product or application development. It is open source distributed ledger technology launched by the Linux Foundation [8]. Later a newer version of this was released with features including faster transactions processing, better smart contract technology and more. Traditional blockchain networks usually don't allow private transactions or confidential contracts that are essential for private enterprises or businesses. Verifying identity being one of the primary requirements for such private industries, hyperledger fabric comes into play by being one step ahead of traditional blockchain in this regard. Hyperledger fabric supports authorization based on certain permissions. All the members trying to access the system must be identified by the security system beforehand in order to gain access. The modularity of this allows it to be used in different business sectors including healthcare sectors.

2.5.2 Smart Contract

A smart contract is an automatic contract that has its conditions written directly into code. It is a program that works on a blockchain and also fulfills different requirements and automatically upholds and brings out the terms of a contract. The main concepts of blockchain, including transparency, decentralization, and immutability, support the workings of smart contracts [22].

2.5.3 Chaincode

In Hyperledger Fabric, Chaincode prescribes how business transactions are to be enacted on a blockchain. It is either written in Go, Node.js or Java and it operates with the ledger for the purpose of executing logic. Chaincode, when deployed to function, executes within a docker container, making its operation independent and protected from other parts of the system [18]. The main function of chaincode is to receive proposals from the client, read or update the ledger, and provide the result of these activities to the clients.

2.5.4 Decentralization

Decentralization is a method of distributing power, control, and decisions within an organization instead of depending on a single authority. No one has full control over the network in a decentralized system. rather, different nodes or people share authority. There are many benefits in decentralization. First of all it prevents a single point of failure. On the other hand there are some challenges also. The main problem of decentralization is consensus.

Chapter 3

Literature Review

Cras et al. [26] explores the potential of implementing RBAC standards in a decentralized fashion enabled by blockchain. The traditional RBAC does not come up with the decentralization method of blockchain. It relies on a single authority called Policy Decision Point (PDP), posing a risk of single point of failure (SPOF). The research proposes a strategy that merges numerous roles and versatile RBAC policies within the smart contracts. This approach implies assigning each role some unique permission policy set. Altering those permissions needs the consent of selected groups which leads to decentralization. A prototype implementation of the Ethereum blockchain is also shown in the article. Even though the RBAC approach in this study is decentralized, problems with the policy decision point still exist because it is still vulnerable to a single point of failure.

According to R. Bagchi [6] the strategies for secure control of IoT devices in a non-centralized fashion, utilizing blockchain technology to avoid dependence on main entities. It demonstrates the practicality of applying IoT device control on a worldwide scale via blockchain, highlighting its ability to confirm device identity, safeguard data, manage access, and conduct transactions in a distributed method. The benefits of using blockchain in IoT encompass equal participation by users, transparency, privacy, non-rejection, integrity, non-centralization, trustlessness, and unchangeability. The study emphasizes the importance of the clear yet confidential characteristic of blockchain, guaranteeing secure transactions and data reliability. Nevertheless, issues like scalability, power consumption, and storage scalability are recognized, with suggested remedies including Simplified Payment Verification nodes, block pruning, and sharding. Scalable blockchain procedures such as BitcoinNG, Lightning Network, and BigchainDB are brought forward as potential solutions. In conclusion, this study offers crucial perspectives on decentralized IoT management, underlining the game-changing capacity of blockchain technology and offering remedies to confront associated difficulties.

Similar to this, Cruz J.P. et al. [9] presents RBAC-SC which is a platform that addresses cross-organizational Role-Based Access Control issues by using Ethereum's smart contract. Through its use of blockchain technology and smart contracts, RBAC-SC aims to protect RBAC architectures from malicious exploitation and allow small organizations to get involved by utilizing a challenge-response verification method. The study performs a performance analysis and investigates the RBAC-

SC framework which includes the use of smart contracts and the challenge-response method. Their implemented application is publicly available and their goal is to increase security and flexibility in cross-organizational RBAC scenarios. It is implemented on Ethereum's Testnet blockchain.

In Business Process (BP) management Viriyasitavat et al. [28] shows how blockchain is used to make the system more modern and efficient. Blockchain records everything in a way that someone can not change it easily and this system makes it more trustworthy in Business Process operations. Blockchain also decrease different conflicts and make their operations more reliable. This study talked about different benefits and challenges that can be achieved and faced if blockchain is used in Business Process (BP) management. it talked about the difficulties because of working with different technology together. In Short blockchain can make the whole operation more smoothly but there are some points which should be solved to make it work in every situation. Importantly, the review centers on methodological frameworks and technologies for Blockchain's adoption in business cycles, underscoring the need for a thorough investigation of challenges in infrastructure, standards, models, methods and algorithms.

Li et al. [21] talked about how permissions are given to users according to their roles throughout the organization in the traditional RBAC access control model. It assures effective control of access rights, decreasing the possibility of unauthorized entry while upholding an organized security system. But because of the complexities and vulnerabilities of modern networks, researchers are looking into decentralized options for RBAC. In this paper they talked about a smart way to solve all the problems and keep all information safe in the internet of things. To protect data and privacy this paper suggests combining two technologies which are blockchain and Role based access control. where RBAC can manage who can access the system and who can not and blockchain can make the whole system decentralized. To make the privacy more strong this paper suggests a tool which is a trusted execution environment (TEE). With the help of this tool transaction information can be kept more safely and it can protect different details like roles.

Vasishta et al. [25] introduces a decentralized structure for controlling access in multi-user online service applications by exploiting Hyperledger Fabric and Attribute Based Access Control (ABAC). An all-inclusive system design is presented by the authors, involving essential operations such as subject incorporation, object addition, rule establishment, and access request management. Interestingly, the plan's intent is to manage situations where the access verification required attributes are spread amongst numerous authorities. Hyperledger Fabric is used for its capabilities to offer transparent permission, unchangeable credentials and secure authority interactions. With a prototype implementation featured, the authors verify the system's performance under diverse parameters. This study adds value to blockchain-based access management, posing a possible solution to decentralized authorization related issues. Still, the article would gain from a more straightforward presentation, explicit contrasts with current solutions, and extra analysis of practical applications. Overall, it sheds meaningful light on integrating Hyperledger Fabric with ABAC for decentralized access control in multiple authority situations.

Hu [27] talks about Blockchain which is a distributed ledger system also very reliable

and safe. Blockchain is a good choice to make ABAC decentralized because it has distributed, transparent and immutable features. If blockchain is used ro make ABAC decentralized not only the drawbacks of centralized ABAC will be solved but also we will get many more benefits and it will improve the whole ABAC system. This paper basically shows different features of blockchain that can be helpful in the ABAC system. They use XACML and NGAC to the system. mainly they focused on XACML and they discussed how the elements of XACML can be converted into blockchain access control.

Steichen M. et al. [10] here the authors stated that on a blockchain, large files cannot be stored efficiently. Within the blockchain network, where data needs to be propagated, the blockchain becomes distended, and for sharing files, a lot of space is required. IPFS is a system where people can share their files more efficiently. Even large files can also be shared through this system. It depends on the cryptographic hashes. This system doesn't permit users to share files with selected parties, which is necessary for personal or sensitive data. This paper showed an updated version of the InterPlanetary Filesystem (IPFS) that was built using Ethereum smart contracts to provide file sharing that is access-controlled. That smart contract was used to observe the list of access controls. So to maintain access control perfectly, whenever a file is uploaded, downloaded, or transferred, it interacts with the smart contracts.

Rashid A. et al. [23] shows RC-AAM as a decentralized method for role-based authentication and access control, which is very important to enhance security and reduce administrative work. The system has been made without depending on a single authority, and they used different types of security features that are provided by the blockchain and cryptography. In their system, they have role verification and also role validation. They have implemented their RC-AAM prototype on the Ethereum test network. It was challenging to make an efficient system that would be centralized for distributed applications, but after their implementation, they found that their method could handle access control effectively and make distributed environments better. Their performance is also better than most of the advanced role-based access control systems that are available.

Xu R. et al. [11] presents BlendCAC, which is a decentralized capability-based access control for IOT security that is achieved by blockchain technology. BlendCAC aims to deliver an effective access control system for large-scale Internet of Things systems. A capability transfer system was suggested for the propagation of access control based on the blockchain network. In their system, they have a capability for token management that is identity-based. They used smart contracts to allow authorization, propagation, registration, and revocation. Instead of the central authority, IOT devices can manage their resources with the help of the BlendCAC schema. They conducted an experiment on a local blockchain network and showed it offers a decentralized and fine-grained access control solution to IOT systems.

Ihle et al. [14] examines if distributed ledger-based authorization systems can assist corporate systems. They take advantage of the blockchain system to implement their smart contract-based role management system. In their system, they didn't sacrifice the advantages of central authorization methods. According to the paper on a decentralized prototype, one of the main advantages is that it can be applied as a basis for additional decentralized company advancements. The main topic of

this paper is the validation and implementation of blockchain-based access control systems for different types of decentralized applications. They used RBAC, which is a role-based access control system, for their examination. Using a distributed ledger platform they prove this is a chain solution for RBAC.

Li Q. et al. [2] present a unique role-based access control model for decentralized and distributed systems. They also propose a decentralized management mode as one of the main efforts to address management problems in traditional RBAC systems. Their model can be applied to flexible tasks in collaborative applications, where the flexible tasks are usually managed by local administrators. This allows multiple administrative duties for many applications to be allocated among many local administrators, leading to the development of a smooth RBAC administration model based on different types of local administrative policies. According to their paper, they developed a secure spread prototype to serve as evidence of the concept method based on their proposed model and to show its feasibility in real-world applications.

Tamassia R. et al. [3] introduce an independently verified delegation mechanism that enables delegation authentication to be verified without involving domain administrators. They proposed a simple and efficient protocol for cross-domain transfer of authority that they call role-based cascaded delegation (RBCD). According to their model, a role member can make delegations via RBCD, considering the current requirements and needs of collaboration. So anyone may verify an authorization chain without involving the role administrators. They also showed a practical implementation of role-based cascaded delegation with aggregate signatures.

Markus I. et al. [15] present a new decentralized ledger access control system that is cryptography-based. This new system is compatible and adaptable with a variety of storage systems with different types of high-level privacy. According to the authors, this new system has a verification system for end users to locate malicious nodes in the decentralized ledger. They implemented the schema with Hyperledger Fabric. They also analyze its performance to show its usefulness in real-world situations.

According to Jamsrandorj U. [7], a substantial amount of research has been done and is available in much of the literature on centralized access control within the framework of a single organization. But there is a shortage of research on decentralized access control in collaborative settings. They developed a prototype for a decentralized access control system using Java, RESTful web services, and multichain blockchain, and the system supports auditability, immutability, equality, and transparency in a collaborative setting for more research in this field. The mean response time and efficiency were the main two outcomes that the prototype was created to evaluate. They executed many experiments in which they measured the metrics on both local area networks (LANs) and Amazon Web services. There was a minor performance difference between the LAN and AWS when more servers were running.

Zhang Y. et al. [24] suggests a decentralized and reliable access control system for smart cities by creating an attribute-based access control (ABAC) model combined with the smart contract technology of blockchain. In this system, they use one contract to handle different ABAC policies, one contract for the attributes of the subject and one for the attributes of objects, and one contract to manage ac-

cess control. They implemented a local system with Ethereum blockchain and four smart contracts to evaluate the financial cost and compare the proposed system with the existing systems. The experimental results showed that the proposed schema requires a higher initial investment than the ACL-based schema and pays lower expenses during system administration, especially for large-scale smart cities.

3.1 Comparative Analysis

In Table 3.1, we have compared ten research works based on some key points, which are their decentralization level, access control model, performance, scalability, security measurement, implementation, protocol flow, and threat modeling. Analyzing those papers, we found that RBAC-SC Paper (Cruz J. P. et al.) [9] and Attribute-Based Access Control for Smart Cities (Zhang Y. et al.) [24] have a high decentralization level. In the first paper, they used RBAC as their access control model, while the second paper used ABAC as their access control model, but there is no threat modeling or protocol flow in their paper. Between Smart Contract-Based Role Management on the Blockchain (Ihle et al.) [14] and DAcc (Markus I. et al.) [15], the first paper has a high decentralization level but DAcc is partially decentralized. DAcc used a protocol flow in their paper. Another paper uses a protocol flow called independently verifiable decentralized role-based delegation (Tamassia R. et al.) [3]. They also implemented their research but they didn't mention any metric to mention their decentralization level. After analyzing carefully, we find out that most of the papers use RBAC as their access control. If we talk about the performance analysis, most of the papers have done their performance analysis based on different matrices. In the table, we have a security measure field by which we can analyze the security measurement of their systems. Most of the papers didn't give any security measurements but some have very high security, like blockchain security mechanisms, Java security, BLS short signatures, MIRACL libraries, digital signatures, etc. After analyzing the whole table, we can say most of them do not have any threat modeling or protocol flow but most of them have a high decentralization level and performance analysis. But then again, most of them do not have security measures. If we talk about the implementation, most of them implemented their proposed ideas. If we compare the four papers that used Ethereum or Hyperledger Fabric, we find that all of them use RBAC or ABAC as their access control models. So we can say ABAC and RBAC are the most famous access control models. In those four papers, three of them have a high decentralization level. In our paper we are using RBAC as our Access Control Model with a high level of decentralization using Hyperledger fabric. To evaluate the feasibility of our research, we implemented our idea in a real-life application and also ran different tests to analyze the performance of our proposed model. We find out the system is scalable enough and also we have blockchain security. We also include a protocol flow and threat model to understand the system and its security requirements better.

Table 3.1: Comparative Analysis

Research Works	Decentralization Level	Access Control Model	Performance Analysis	Scalability	Security Measures	Implementation	Threat Model	Protocol Flow
RBAC-SC (Cruz J.P. et al.) [9]	High	RBAC	✓	-	-	✓	✗	✗
Blockchain-Based, Decentralized Access Control for IPFS (Steichen M. et al.) [10]	High	ACL	✗	Depends on Ethereum network scalability	-	✓	✗	✗
(RC-AAM (Rashid A. et al.) [23]	-	Decentralized role-based authentication and access management using blockchain	✓	Scalable (P2P network)	Digital signature	✓	✗	✗
BlendCAC (Xu R. et al.) [11]	High	Capability-based	✓	Scalable (device-to-device communication)	Blockchain security	✓	✗	✗
Smart Contract-Based Role Management on the Blockchain (Ihle et al.) [14]	High	RBAC	-	-	Smart Contracts security	✓	✗	✗
Towards a group based RBAC Model and Decentralized user-role administration (Li Q. et al.) [2]	-	Group-Based RBAC (GB-RBAC)	✓	Limited	Session Management, Access Control Policies	✓	✗	✗
Independently verifiable decentralized role-based delegation (Tamassia R. et al.) [3]	-	RBAC	✓	Limited	Java Security, BLS Short Signature, MIRACL Library	✓	✗	✓
DAcc (Markus I. et al.) [15]	Partial	RBAC augmented with decentralized ledger	-	Scalable	Cryptography	✓	✗	✓
Decentralized Access Control Using Blockchain (Jamsrandorj U.) [7]	High	-	✓	Scalable	-	-	✗	✗
Attribute-Based Access Control for Smart Cities (Zhang Y. et al.) [24]	High	RBAC	✓	-	-	-	✗	✗
Our Work	High	RBAC	✓	Scalable	Blockchain Security	✓	✓	✓

Chapter 4

Proposal

4.1 Proposed Model

The most significant drawback of having a centralized system in operation is the existence of central authority that creates a dependency along with the risk of single point of failure. To prevent this, our research paper proposes a decentralized access control solution that applies the principles of blockchain technology more precisely Hyperledger Fabric for the access control. Decentralizing all of the four components of XACML will distribute the decision making authority to every available node in the system resulting in a more robust and secure system. As demonstrated in the system architecture, each of the nodes will have all four components of the XACML in chaincodes. This way even if one or more nodes get compromised, as long as even one node is active, the system will respond with accurate decisions according to the policies and attributes. Along with our authorization, our research paper also proposes a SAML authentication for identifying users using SAML Assertion. This will allow users to share their security credentials across all available services under a domain more widely known as Single Sign-On (SSO). According to I. Alom et al., identity provider can be decentralized using hyperledger fabric [19]. If our proposed model is combined with a decentralized SAML identity provider, it will result in a complete decentralized access control system but our research focuses solely on decentralizing authorization. The proposed solution of a decentralized system is not bound to any specific industry making it scalable and possibly further usable across all environments.

4.2 Methodology

To complete our research in a structured approach we adopted Design Science Research (DSR) methodology to develop and evaluate our Hyperledger Fabric-based decentralized access control system. This is one of the methodologies that provides an approach to developing and proving solutions. That is beneficial to the researcher in constructing solutions and hypotheses in areas of interest such as business, science and technology [5]. Every phase of the approach made sure we have complete knowledge of the topic and development of the system, with options for iterative approach depending on results and problems faced at every level:

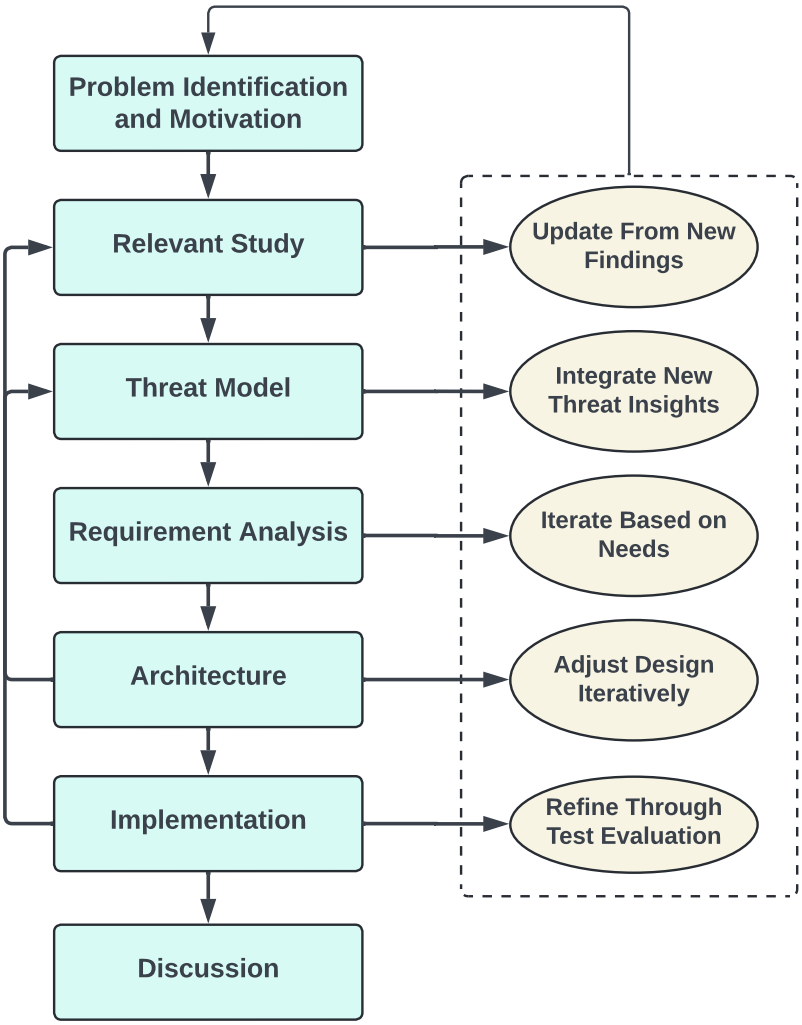


Figure 4.1: Methodology Flow Diagram

1. **Problem Identification and Motivation:** In this phase, we determined the fundamental problems with current access control systems in terms of security. After observing numerous security breaches, we found our motivation to come up with a solution.
2. **Relevant Study in Depth:** In order to have a better understanding of current security on access control systems, a comprehensive review of related research has been done to have an idea of what should or should not be done. The results of this review have helped identify current technology gaps and serve as the basis for system design.
3. **Threat Model:** In this phase a security model is chosen to follow which is enough to address the security risks inherent to the access control systems. Other than the model we also considered various attacks such as internal threats and external breaches to make sure the proposed system is secure.
4. **Requirements Analysis:** Here the functional and security requirements analysis has been done in great detail. Reviews of previous research are useful

to identify the requirements to make the system secure and scalable.

5. **Architecture:** The architecture of the proposed system is flexible and scalable because it uses blockchain technology in order to decentralize the access control system. The architecture is clear and explains chaincodes, blockchain networks, and interfaces. Because of a clear structure, changes are easy while maintaining good performance.

6. **Implementation:** In the implementation phase, the system employs Hyperledger Fabric to build chaincode that manages the proposed extended access control mechanism. This stage involves writing the chaincode and configuration of the blockchain such that when deployed it operates to the optimum and securely. Also, the system is integrated with the current IT environment to understand the real-world scenario. This detailed process allows the creation of a safe and stable solution that will meet certain access control requirements.

7. **Discussion:** In this phase a discussion and assessment of the previous requirements have been done. Fulfillment of the functional and security requirements has been identified along with research objectives.

In each step of the proposed methodology shown in Figure 4.1, if there are issues that indicate the need to go back to the previous level, then looping back to any previous level is possible. This looping back makes it an iterative process that enables continuous improvement and problem-solving of the system. For example, if during the discussion phase risks are found, the system may have to go back to the Architecture or Implementation phase for corrections. Then, it can be done. So, new requirements that might be discovered during usability tests could refer to Requirement Analysis or even Relevant Study to cover the needs of making this research successful.

4.3 Threat Modelling

In this section, we try to identify the possible security and privacy-compromising scenarios that can take place while implementing an XACML based system. Each of these parameters can be taken care of individually while implementing the system. To be very specific, we will be focusing on an established threat model known as STRIDE [4]. STRIDE is a security threats which are given below:

T1. **Spoofing Identity:** This threat is the consequence of an intruder pretending to be someone else for the purpose of gaining access to the network by getting the user's information during verification. An attacker might pretend to be the real user in an XACML access control system. This system can be bypassed by using this information to get sensitive information without permission.

T2. **Tampering with Data:** By impersonating a verified attribute, a hacker can alter data that is very important to the system or network. Also, a hacker with elevated authority can alter features, roles, and even authorization in an XACML security control system. After alteration, a hacker can change the system's decisions regarding access control and get unauthorized access to important information.

T3. **Repudiation:** Refers to the capability of developed entities, in keeping with regulations, to reject specific acts that are illegal. Someone can try to change an action in the XACML access control system and then state that they were not involved there. They might achieve this by taking advantage of holes in the implementation of policies and not enough policies. As needed by the XACML policies, it will become harder to perfectly link actions to specific entities.

T4. **Information Disclosure:** The unintentional disclosure of confidential data by an aspiring hacker is information disclosure. For instance, a mistake in establishing control may inadvertently provide broad access to distinct roles. and these mistakes might enable unauthorized outsiders to get confidential information, which may lead to data leaks and dangers.

T5. **Denial of Service (DoS):** A DoS attack can be utilized to accomplish an unauthorized set of services. A hacker may launch a DoS attack by rushing the servers with excessive requests and eliciting information at once, making them uneffective. and thus this can render the system unstable, leaving the valuable assets exposed to unauthorized entry and prohibiting them from fully imposing authorization policies.

T6. **Elevation of Privilege:** The utilization of further actions to boost one's likelihood of obtaining access to system or network resources is the elevation of privilege. For example, if the attacker gains initial access via a low-privileged user, they can potentially take advantage of the weaknesses of the system to increase their privileges by obtaining around XACML constraints and gaining unauthorized access to the resources.

Additionally, an additional risk, which is very essential for the security of the services, has been addressed.

T7. **Replay Attack:** Replay attacks require an attacker trying to capture the data packets that users give to the server throughout the session. Afterwards, he tries to send those packets again to fool the server into thinking that the requests are authentic. In cases where private information is sent, it might give rise to data manipulation and unauthorized access.

T8. **XML External Entities (XXE):** A web security flaw that allows an attacker to manipulate how it processes its XML Data and even inject additional XML data. An attacker might inject malicious XML data including external entities. Through this manipulation, an attacker can easily access sensitive data such as passwords, emails, and configuration files stored in the server by changing policies. This might compromise the entire system.

4.4 Requirement Analysis

In this section, we specify multiple functional, security, and privacy requirements. Functional Requirements describe the basic tasks that the system must perform. On the other hand, security and privacy requirements will ensure the mitigation of possible threats to the system.

Functional Requirements (FR): The functional requirements required for out report are presented below:

FR1. The system must have the ability to define and enforce access control policies and, the capability to manage user attributes and permissions.

FR2. A blockchain-based access control system should be integrated to ensure decentralized access control.

FR3. The system must make the right access control decision through chaincodes.

FR4. The system should work with a legacy system to show its applicability

FR5. The system must be scalable to support increasing user bases and access control demands.

Security Requirements (SR): The security requirements for the report are presented below:

SR1. A reliable authentication system must be implemented to mitigate the T1 threat.

SR2. Different types of hashing mechanisms must be utilized to protect data integrity and prevent tampering which will mitigate the T2 threat

SR3. Digital signatures must be used to ensure the non-repudiation of actions and mitigation of threat T3

SR4. Implementation of access control measures to prevent unauthorized information disclosure Also the data must be stored and transmitted in an encrypted manner through networks. It might mitigate the threat T4

SR5. The system must implement different measures to identify and mitigate Denial of Service (DoS) attacks

SR6. To mitigate the T6 danger, the system will use access control mechanisms such as RBAC [1] to prevent attackers from elevating their privileges.

SR7. The system must take protective measures such as signature-based authentication, token, and data encryption to prevent T7.

SR8. Implementation of input validation and safe parsing algorithms to avoid XML External Entities (XXE) attacks and mitigate T8.

Chapter 5

Architecture, Use-Case and Protocol Flow

5.1 Architecture Design

The architecture of the decentralized access control system using Hyperledger Fabric is intended to mitigate the vulnerabilities that are inherited from centralized access control systems. This system aims to lower the risks related to unauthorized access. To meet our research objective, the architecture must be configured in a way that the resource control is distributed among the network nodes, which implies redundancy, fault tolerance, and consensus-based trust. To make sure all the nodes have the necessary features, blockchain technology will be integrated. It should use Smart Contracts or chaincodes to encode access control policies making them transparent and secure at the same time. Furthermore, the architecture must allow for the deployment of multiple controlling nodes. Each node will be independent of managing access control decisions. It will improve the scalability, and flexibility, and ensure the transparency of the structure. The decentralized nature of the architecture is to prioritize the visibility and immutability of access rights. Such features can be achieved by utilizing the features of Hyperledger Fabric to ensure a secure as well as resilient access control system.

XACML and SAML cooperate beautifully. While SAML makes it certain that the person is who they claim to be, it allows them to interchange data. This often merges into B2B applications because SAML will allow single sign-on, while XACML will allow more fine-grained access control. At least one Identity Provider (IdP) needs to be available that supports SAML. Figure 5.1 presents a compatibility architecture of SAML and XACML.

In the proposed architecture, we intend to use the Hyperledger Fabric Blockchain technology to make sure to strengthen our access control mechanism. We need to implement the chaincodes of PIP, PAP, PEP and PDP into the blockchain. The aim is to attach these components to the blockchain to guarantee the transparency, immutability, and decentralized governance of the access control. This approach fits very well with the decentralized nature of blockchain technology, bringing greater security and trust into our decentralized applications.

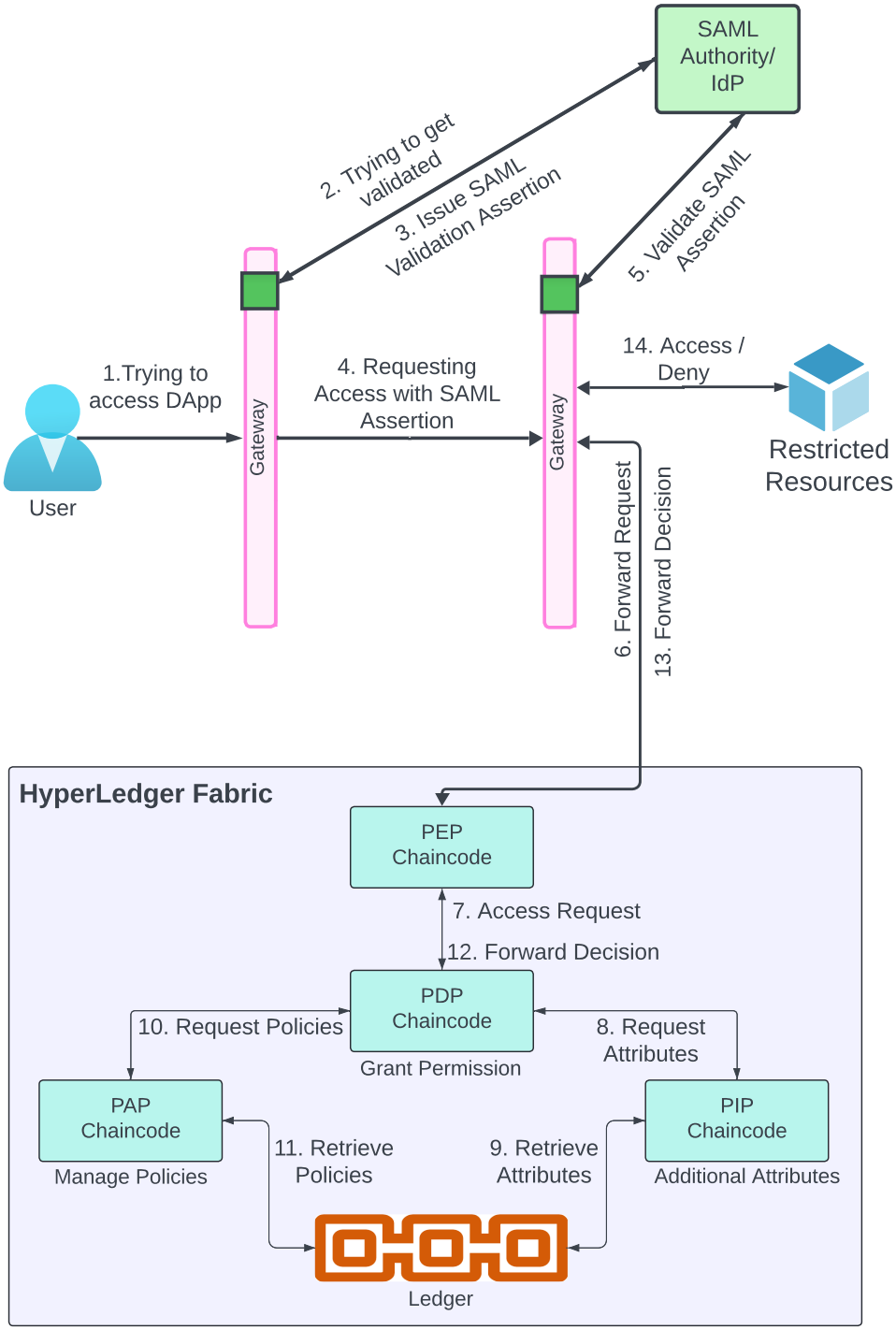


Figure 5.1: Proposed Architecture

The flow of this architecture is given below:

i **Trying to access DApp:** A user is trying to access a decentralized application.

ii **Trying to get validated:** The user is seeking validation from the SAML Authority or IdP.

iii **Issue SAML Assertion:** The SAML Authority such as IdP issues SAML Assertion to the user. An assertion is an XML document that is signed using the private key of the IdP and contains information of the user.

iv **Request Access with SAML Assertion:** The user requests access to the DApp with the SAML assertion.

v **Validate SAML Assertion:** Before making the decision regarding access the DApp decrypts the assertion to validate the IdP to validate it.

vi **Forward Request:** Based on the authentication, if allowed access, users can request resources or services to PEP who is responsible for authorization decisions.

vii **Access Request:** The PEP forwards the access request to the PDP which is responsible for evaluating the access control policies and make decisions.

viii **Retrieving attribute from PIP:** The PDP requests and receives attributes from the PIP which is stored in the ledger of hyperledger fabric.

ix **Retrieving policies from PAP:** The PDP requests and receives policies from the PAP which is also stored in the ledger of hyperledger fabric.

x **Access Response:** After evaluating, PDP sends an Access Response to PEP.

xi **Permit/Deny:** The PEP forwards the decision to the system.

This architecture describes a way to combine SAML and XACML to achieve secure access control for decentralized services. SAML is used for authentication and XACML for authorization.

5.2 Use-Case and Protocol Flow

In this part of the paper, we will present a use-case along with protocol flow to show how hyperledger fabric helps us decentralize the access control system using XACML. At first we present our mathematical notations shown in Table 5.1 which is self-explanatory. Secondly, our data model is shown in Table 5.2 and a clear overview of this table is explained Data Model section.

Data Model: All the user action in the system can be combined as a request, denoted as *req* in Table 5.2. A *req* can be of 5 types: $spMetaURL$, $idpMetaURL$, type, data, and SAML. These requests consist of asking for data the function needs to do its functional jobs. In the algorithm section, we will talk more about why the function needs those data. The responses are denoted as resp, and there are 3 types of responses: $HTTP$, URL, and $assertion$. Just like *req*, *resp* is also responsible for outputting a specific data type which might be needed for other functionalities or features of the system. Next, we will use $loginData$, It will be a combination of username and password. Email can also be used instead of username but data data type have to be unique here. And the login credentials need to be in pairs. Then we have redirect, denoted as redirect in the table. It works as a function and it takes two parameters. The first one is the URL to redirect to and the second one is what it will take. Assertion could be a good example of with because we will be judging the assertion as the identity card of the U, so it needs to be taken with us wherever

Table 5.1: Cryptographic Notations

Notations	Description
SP	Service Provider
U	User
K_U^{SP}	Public key of U for SP
K_{SP}^U	Public key of SP for U
$K_{SP}^{-1/U}$	Private key of SP to be used for U
$K_U^{-1/SP}$	Private key of U to be used for SP
N_i	A fresh nonce
CC	Chaincode
CC_{PEP}	Chaincode of PEP
CC_{PDP}	Chaincode of PDP
CC_{PIP}	Chaincode of PIP
CC_{PAP}	Chaincode of PAP
$H(M)$	SHA-256 hashing operation of message M
$K_{-1/IdP}$	Private key of IdP
$[...]_{https}$	Communication over secure https channel
$[...]_{RPC}$	Communication over a RPC channel
$\{\}_{K_{SP}^U}$	Encryption operation using public key of SP for U

we go in the system. Then we have a certificate, which we will get from IdP and its call the X509 Certificate. After that we have some data denoted as DATA in the respective table. Then we have the sign method. It takes two parameters: document and key. We sign the document with the key to avoid its forging. Then at last we have decrypt which takes M, Message, and a key to decrypt with.

Table 5.2: Data Model

req $\triangleq$ ⟨ SPMetaURL, IdPMetaURL, type, data, SAML ⟩
resp $\triangleq \langle HTTP, url, assertion \rangle$
loginData $\triangleq$ ⟨ username, password ⟩
redirect $\triangleq$ ⟨ url, with ⟩
certificate $\triangleq$ ⟨ X509 Certificate ⟩
DATA $\triangleq$ ⟨ MetaData ⟩
sign $\triangleq$ ⟨ document, key ⟩
decrypt $\triangleq$ ⟨ M. key ⟩

Algorithms: Firstly we need to initialize and configure the IdP and SP for data transaction. If a system decides to rely on a IdP for the authentication of the users, the SP and IdP need to exchange the metadata. Metadata is an XML format document that contains many things such as where to redirect after redirect, what to do if authentication fails, what to do for an unauthorized or authorized user, and some other requirements. After the metadata exchange between IdP and SP, they both use this metadata for future communication.

Both SP and IdP has cryptographic trust. Additionally included in the transmitted information are the public and certificated keys for signing and encrypting. All message transit between SP and IdP is thereafter secured using these.

When the SP sends the users and the authentication request to the IdP jointly. It generally autographs it. It does so for the matching key in the metadata using its private key. The IdP checks the signature with the public key in the metadata that the SP previously delivered when it receives the used authentication request.

Algorithm 1: Algorithm snippet for Setting Up IdP

Input: *req*
Output: None

```
1  Start
2  function exchangeMetadata(req)
3      SPMetadata := fetchMetadata(req, SPMetaURL);
4      IdPMetadata := fetchMetadata(req, IdPMetaURL);
5      storeMetadata('SPMetadata', SPMetadata);
6      storeMetadata('IdPMetadata', IdPMetadata);
7      return;
8  function fetchMetadata(url)
9      resp := HTTP.getURL;
10     return resp;
11 function parseMetadata(req)
12     DATA := { entityID := req.data.entityID,
13     .
14     .
15     certificate }
16 function storeMetadata(DATA)
17     database.store(DATA);
18     return;
19 End
```

In Algorithm 1, we go through how we handled the metadata exchange between SP and IdP which includes the fetching of the metadata, parsing it storing the necessary information from the metadata. The main function takes two URLs over a secure HTTP channel one is idpMetadataURL and the other one is spMetadataURL. In exchangeMetadata we first fetch the metadata of IdP and SP using a secure HTTPS channel. Then we parse the metadata and extract all the necessary information or data we need to make sure a secure a safe communication between IdP and

SP. The necessary data includes the certificates and public keys of both IdP and SP. Information also includes data like spEntityID, acsURL, sloURL, spCertificate. storeMetadata function is used to store the metadata and parsed information from the metadata. It saves the information as an object that holds information like entityID, ssoURL, sloURL, and certificate. SP will store the metadata of IdP and IdP will store the metadata of the SP. Certificates are very important key of trust between IdP and SP. By exchanging the certificates a trust is built within the IdP and SP and this trust will be used in the future to authenticate the users.

After setting up the metadata for both IdP and SP. Now we will see how the assertions are exchanged between these two entities while both of the entities have each other's public keys. Now we will see how the assertion is exchanged between the user, IdP and SP.

Algorithm 2: Algorithm snippet for Sending SAML Assertion

Input: $req \rightarrow recievedSAMLRequest$
Output: $resp \rightarrow SAMLAssertion$

```
1  Start
2  function processSAMLRequest(req)
3      valid := validateSAMLReq(req.SAML);
4      if valid then
5          assertion := sign(req.SAML. K^{-1/IdP});
6          redirect(U.url, {assertion}_{K_U^{SP}};
7      else
8          sendErrorResponse("Invalid SAML Request");
9      return
10 function SendAssertion(resp)
11     resp := resp.assertion;
12     redirect(SP.url, resp.assertion);
13     return;
14 End
```

In Algorithm 2, the SAML request is validated and the proceedings are done to create, sign, and send the assertion if they are validated. This function is activated when the SP redirects the user to the IdP for authentication. This function will take loginData which is generally username/email and password. In this function we take validateSAMLRequest functions help to function properly. The function takes the loginData as its parameter and checks if the user is who he is claiming to be. In that function, we will have to put a proper logic in order to build our criteria of a valid SAML assertion. Otherwise the attackers might bypass the authentication process. Generally, they come up with verified certification like X509Certificate or it may

have to sign with proper entities public keys or private keys. We will talk about it more in the implementation section. After that, IdP will generate the assertion for the user. This generated assertion carries the user information like username, full name, age, and other necessary information required by the service providers. Then the next step would be signing the assertion with IdP's private key. After validating and signing the assertion, we redirect the user to the service provider along with the assertion. The user carries this assertion with him until a certain time passes. There is a Time Limit Exceed (TLE) parameter in the assertion, after that particular time user needs to get validated again. Generally the assertion stays in the browser cookies.

Now, the SP knows that this assertion is from the trusted IdP depends on the signature. In the next algorithm, we show how the SP verifies it.

Algorithm 3: Algorithm snippet for SAML Assertion Validation

Input: $req \rightarrow receivedSAMLAssertion$
Output: $validationStatus \rightarrow$ "$Valid$" or "$Invalid$"

```
1  Start
2  function validateSAMLAssertion(req)
3      assertion := req.assertion;
4      valid := verifySignature(assertion);
5      if valid then
6          return "Valid";
7      else
8          return "Invalid";
9  function verifySignature(assertion)
10     M := assertion;
11     DecryptResp := decrypt(M, K_SP^{-1/SP});
12     IdPSign := DecryptResp.Sign;
13     if IdPSign == K_IdP then
14         return true;
15     else
16         return false;
17 End
```

Here in Algorithm 3, we validate the SAML assertion inside the SP so that the SP would know if the assertion is valid and which resources is this U authorized to see and use. Here we decrypt the SAML response using the private key of the SP then extract the assertion. Then we verify the signature of the assertion. We will decrypt the response in the decryptResponse function using the SP private key. Because it was encrypted using the public key of the SP by IdP. Then from the decrypted assertion we extract the signature then using the verifySignature function we verify it. the verifySignature function is defined to check the validity of the signature. It loads the IdP's public key and then verifies the hash of the assertion data against the extracted signature using the public key of the IdP. This pseudocode gives an overview of how the SP retrieves, validates, and decrypts the SAML response to guarantee integrity and validity.

Algorithm 4: Algorithm snippet for Chaincode PDP

Input: *req* → *jsonObject*

Output: *Decision* → "*Permit*" *or* "*Deny*"

```
1  Start
2  function evaluate(req)
3      req := Parse(req);
4      subject := req.subject;
5      action := req.action;
6      resource := req.resource;
7      policies := invokeChaincode('PAPChaincode', 'getAllPolicies');
8      policies := Parse(policies);
9      decision := 'Deny';
10     for policy of policies do
11         parsedPolicy := Parse(policy);
12         result := evaluatePolicy(parsedPolicy, req);
13         if result := 'Permit' then
14             decision := 'Permit';
15             break;
16     return decision;
17 function evaluatePolicy(policy, req)
18     rules := policy.Policy.Rule;
19     for rule of rules do
20         if matchRule(policy.Policy, rule, req) then
21             return rule.$.Effect;
22     return 'Deny';
23 function matchRule(policy, rule, req)
24     subject := req.subject;
25     action := req.action;
26     resource := req.resource;
27     roles := invokeChaincode('ChaincodePIP', 'getRoles');
28     subjectMatches := rule.Target[0].Subjects[0].Subject[0].SubjectMatch;
29     actionMatches := rule.Target[0].Actions[0].Action[0].ActionMatch;
30     resourceMatches := rule.Target[0].Resources[0].Resource[0].ResourceMatch;
31     subjectMatch := subjectMatches.some(sm →
         roles.includes(sm.AttributeValue[0]._));
32     actionMatch := actionMatches.some(am → action === am.AttributeValue[0]._);
33     resourceMatch := resourceMatches.some(rm → resource ===
         rm.AttributeValue[0]._);
34     if (not subjectMatch) or (not actionMatch) or (not resourceMatch) then
35         return False;
36     return rule.$.Effect == 'Permit';
37 End
```

Algorithm 4 specifies whether or not a particular user is permitted to use a service or to see a resource in our system. The 'evaluate' function is the heart of the algorithm that takes JSON requests which should contain information about the subject (the username), the action the user wants to take, and the resource on which the action is to be taken. The system then calls for all the policies by invoking a chaincode

known as "PAPChaincode". It stores it on a variable "policies", and then the system parses it to do later operations. Initially, it sets the decision variable to "Deny," and then it goes through all the policies through a loop, for each policy it calls a function "evaluatePolicy()" which in turn evaluates each policy in detail. It takes each policy at a time and again parses it and passes it to a function named "matchRule". This function compares the given request with the requirement of a particular rule in order to see if the rule can be implemented. It retrieves the subject, action, and resource from the request, and the roles of the subject by invoking the PIPChaincode. After that, it checks if the subject, action, and resource are present in the rule using SubjectMatch, ActionMatch, and ResourceMatch. This is done by making a comparison of the value with the values in the rule set against the values in the request. If the rule was matched to the request by the user, the action and resource of this function will return True. If the rules do not find any matches, the function returns False. The boolean return will then propagate back to the evaluate function. Depending on the result, the user gets either a "Permit" or "Deny".

Protocol Flow: In the protocol flow we have shown how IdP authenticates a user and the functionalities of XACML in Table 5.3 and Table 5.4 respectively.

Table 5.3: Data Flow Table 1

$M1$	$U \rightarrow SP : [N_1, serviceReq(url_{sp})]_{https}$
$M2$	$SP \rightarrow U : [N_1, redirect(url_{idp})]_{https}$
$M3$	$U \rightarrow IdP : [N_2, serviceReq(url, IdP)]_{https}$
$M4$	$IdP \rightarrow U : [N_2, CredentialRequest]_{https}$
$M5$	$U \rightarrow IdP : [N_2, LoginData]_{https}$
$M6$	$IdP \rightarrow U : [N_2, redirect(url_{sp}, [SAMLAssertion]_{k-1/IdP})]_{https}$
$M7$	$U \rightarrow SP : [N_3, serviceReq(url_{sp}, [SAMLAssertion]_{k/IdP})]_{https}$

i The user submits a request to SP and try to use its services at url_{SP}.

ii SP returns Welcome Page to the user and redirects the U to url_{IdP} for authentication as a U via M2 in Table 5.3.

iii As the U is redirected to the IdP, it asks for login credentials to validate the U

iv The user submits LoginData to initiate the login process with the IdP.

v IdP will verify the user and generate an SAML Assertion. The assertion contains many information or attributes like entityId, username etc. It also contains the certificates, authentication status and some additional attributes such as roles, permissions, email etc. Finally the assertion is digitally signed by Private key of the IdP ($K^{-1/IdP}$), step M5 of Table 5.3.

vi IdP redirects the U to url_{SP}, The U carries the SAML Assertion in its browser.

vii After getting to the SP with the assertion, the SP validates the assertion by decrypting the assertion using the public key of IdP and checking the certification of the assertion.

Table 5.4: Data Flow Table 2

$M1$	$U \rightarrow CC_{PEP} : [N_3, DATAReq]_{https}$
$M2$	$CC_{PEP} \rightarrow CC_{PDP} : [N_3, CC_{PDP}(DATAReq)]_{RPC}$
$M3$	$CC_{PDP} \rightarrow CC_{PIP} : [N_3, attrReq_U]_{RPC}$
$M4$	$CC_{PIP} \rightarrow CC_{PDP} : [N_3, attrResp_U]_{RPC}$
$M5$	$CC_{PDP} \rightarrow CC_{PAP} : [N_3, policyReq_U]_{RPC}$
$M6$	$CC_{PAP} \rightarrow CC_{PDP} : [N_3, policyResp_U]_{RPC}$
$M7$	$CC_{PDP} \rightarrow CC_{PEP} : [N_3, resp]_{RPC}$
$M8$	$CC_{PEP} \rightarrow SP : [N_4, resp]_{https}$
$M9$	$SP \rightarrow U : [N_5, DATA]_{https}$

Table 5.4 represents the data flow of the functionalities of XACML. The data flow explanation is discussed below:

i After all the validation, the U then finally be able to request any DATA or Resources.

ii The request have to go through the PEP. The CC_{PEP} handles the request and forward it to the PDP through a RPC channel.

iii After receiving the request CC_{PDP} requests for additional information on the user to verify if the U is authorized to have or use the DATA to PIP through the RPC channel.

iv CC_{PIP} responds with that request and sends all attributes and optional data of that user to PDP using a RPC channel.

v CC_{PDP} then checks all the policies and verifies if the U is authorized or not

vi PDP then forwards the decision of the authorization of the U which is denoted as $resp$.

vii Finally if the validation id successful and the U is authorized to have the DATA, SP sends the DATA to the U.

The protocol flow that is presented effectively illustrates how SAML-based SSO allows safe and easy user authentication across many services. Then PEP, PDP, and PIP access control techniques are included into the system to guarantee that users not only authenticate successfully but also get the right access to resources depending on specified policies and user roles.

Chapter 6

Implementation and Performance Analysis

The approach that we are proposing in this paper utilizes XACML, SAML, and Hyperledger Fabric for decentralized access control. In our case, we chose RBAC as our access control model. We opted to implement the whole idea into an enterprise-like web service. The purpose of this implementation is to decentralize the management of access control, remove single-point points of failure, and increase security by applying hyperledger blockchain technology. We built an Event management web service where there will be multiple roles, each role will have access to certain resources, and every time the resources are tried to access, the evaluation will happen in the chain codes. This ensures that any resources that are locked up by this system can only be opened by users with certain privileges, such as an administrator or an admin.

6.1 Environment Setup

To implement the proposed system in a real-world enterprise like a web service, the following environment was set up:

- **OS:** Ubuntu 24.04 LTS was selected as the operating system because of its reliability, stability, security, and compatibility with the tools and libraries
- **Blockchain:** Hyperledger Fabric version 2.5.9, used for distributed ledger management and chaincode execution. It is the core component which decentralizes the whole access control.
- **SAML SSO:** WSO2 Identity Server (WSO2 IS) version 7.0.0 is used for SAML-based authentication and single sign-on (SSO) features. It also provides the necessary information for a user which helps to manage users efficiently [32].
- **Node.js:** Version 16.x is used to build the front-end part of the application and for the interactions with the Hyperledger Fabric. The Node.js environment also supports chaincodes and API that allows decentralized access to the blockchain system [40].

- **Go:** Version 1.16.x was used because it is one of the prerequisites of hyperledger fabric technology [37].
- **cURL:** During application setup commonly used for downloading binaries and interacting with different REST APIs cURL was installed for downloading the required scripts and packages including Hyperledger Fabric binaries from official repositories [36].
- **Git:** When it comes to version control, Git was used for cloning repositories and chaincode and other Hyperledger Fabric related artifacts' source code and binaries. Git also guarantees that the development process is both orderly and coordinated [13].
- **Python:** Version 2.7 was installed because certain pieces of Fabric tools depend on Python for scripts and automation through the setup process. Though Python 3.9 is the latest one but we went for the older version because in practice hyperledger fabric is not updated to support newer versions of Python [41].

6.2 Tools and Technologies

To facilitate the system integration and just the way it works overall, we used several tools:

- **WSO2 Identity Server (IS):** For SAML-based authentication and identity service [35].
- **Hyperledger Fabric:** For the distributed ledger management and chaincode to be able to make authorization decisions prospects. It is very much suitable for enterprises who want to pursue blockchain as their security environment [38]
- **Blockchain Explorer:** Version 2.0.0 to visualize and observe the network of Hyperledger Fabric. It helped with the monitoring of transactions and blocks [20]
- **Docker and Docker Compose:** Versions 20.10.7 and 1.29.2 respectively for containerization and orchestration of services. It is also necessary for hyperledger fabric because Services were deployed in Docker containers to enforce the isolation of execution environments of peers, orderers, and CAs. Docker Compose was used to solve the problem of running multiple services at once [12].
- **XACML:** For defining the access control policies and to measure the access control requests.
- **Visual Studio Code:** The Development Environment that was used in the development process in writing, editing, designing, compiling both the chaincodes and front-end backend codes, and testing the system [42]
- **JMeter:** Version 5.6.3 to conduct load testing and evaluate the performance and capacity of the system. It helped us determine if our system was able to support large enterprise loads without being slow [39].

6.3 System Overview

We are implementing a solution that utilizes blockchain technology more specifically Hyperledger Fabric for the access control.

6.3.1 Development

In order to build a simplified decentralized secure system, we have implemented two peers, and two organizations with one orderer and one channel network architecture using Hyperledger Fabric. Each of the peers is part of a different organization, sharing the same distributed ledger and executing chaincodes for access control. The orderer also guarantees that the transaction is ordered properly and propagated to the nodes of the network. In this architecture, the system attains decentralization because both peers share the control of the system. If one peer is offline, the other peer keeps on handling requests and confirming the transaction to ensure uninterrupted access control. Also, a round-robin algorithm was implemented that alternates between all the peers distributing the load equally among them. This round-robin technique ensures that every peer participates in each task so that the system gets better performance. The resilience and fault tolerance of the system increases with the increasing number of peers engaged in the network to ensure the continuous availability of access control services. This integration is essential to avoid any possibilities of SPOF. Since no one peer or organization can dominate the system, there is no single point where all the access decisions are made. Both peers are active members of the network and all of the access control policies are mirrored about them. As long as there is at least one peer available, the system is operational, can validate the user's requests, and make the access decision based on the XACML policies stored in the blockchain which improves the security and decentralization of the access control system.

The system architecture uses the WSO2 Identity Server as the primary SAML-based Identity Provider (IdP) to handle all the authentication and authorization needs, while Hyperledger Fabric handles both the distributed ledger and chaincode operation for managing access control on the distributed ledger. SSO is also supported through SAML 2.0 assertions which means that once a user has logged in to an application, he can access all the other related resources without having to enter a username and password again. Concurrently, Hyperledger fabric keeps access control policies decentralized and only allows the user to access them by invoking chaincodes. Access control decisions are decentralized across the blockchain network to mitigate SPOF. WSO2 Identity Server provides necessary information of a user which the SP needs but the roles of each user are saved on the distributed ledger through the chaincode of PIP of Hyperledger fabric for enhanced security. The policies will also be stored in the distributed ledger through the chaincode of PAP, as policies are the most sensitive security concern of access management. As mentioned in the paper, the proposed architecture decentralizes the access control model, in our case RBAC, and improves security, trust, and robustness by using blockchain to make authorization choices throughout the network.

6.4 Steps to Implementation

The implementation of the system was executed in four key phases: WSO2 Identity Server (WSO2 IS) configuration, Hyperledger Fabric configuration, XACML and SAML Configuration as well as chaincode implementation. Initially, WSO2 IS was deployed on a Ubuntu 24.04 LTS and a new SP was created to use SAML for authentication and SSO features. The Assertion Consumer Service (ACS) URLs and metadata were then set to allow secure connections between the client application and the identity server. Following this, a Hyperledger Fabric network was set up through Docker and Docker Compose for the transaction ordering service, peer nodes, as well as the CA to handle network identity. Then chain codes of PEP, PDP, PAP, and PIP were developed to decentralize the access control management. In the next phase, role based access control was implemented by combining XACML and SAML technologies. SAML was used to authenticate users and PIP to get the user's roles while the PDP was made responsible for deciding the access for a certain resource based on policies given stored in the PAP of the XACML. Then, chaincode development was initiated and done very carefully as it will be the pillar of the access control system. The PEP is the interceptor between the users and the PDP service and it is responsible for sending the access requests to the PDP and giving access results back to the users based on the evaluation of the PDP. The PEP was to intercept access requests which were to be processed by the PDP based on the roles and policies that were stored in PIP and PAP respectively. The PIP and PAP retrieved the role attributes and policies required from the blockchain, to provide decentralized, secure management of roles and policies. This implementation did not require a specific database to store the user roles and permissions, instead, it leveraged the Hyperledger Fabric blockchain distributed ledger to manage user roles and policies securely.

6.5 Performance Analysis

To compare the results under varying loads and stress we have recorded the total sample count, average response time, error, throughput and latency after each test cycle. And after the completion of each test cycle, the data derived from different configurations were aggregated in order to get an assumption of the system performance under different circumstances. Consequently, the results were analyzed thoroughly to understand how the performances were varying. Figure 6.1, Figure 6.2 and Figure 6.3 show the comparative analysis of the test scenarios. Device Configuration:

- **Processor:** AMD Ryzen 9
- **Memory:** 16GB RAM
- **Graphics:** NVIDIA GeForce RTX 3060
- **Storage:** 1TB SSD

6.6 Testing Scenarios

In this testing process, two key metrics are used to evaluate the performance of the 'enforce' function in different scenarios. Both metrics call for the 'enforce' function, which is the system's most complicated and core function of our research. The function calls the PEP of the chaincode which then makes an inter-chaincode call to PAP and PIP for policies and roles of the user.

- **One-Peer Fallback:** In this case, one peer is in charge of the incoming requests and forwards the responses. In case that peer fails or falls the other peer will take responsibility for the incoming requests. This is decentralized in the sense that there is a fallback to restore control over the system. If we ignore the fallback section, this metric can also be compared to a centralized system as only one peer is working.
- **Round-Robin Load balancing:** In this setup, both peers are equally responsible for the access requests. Every peer takes turns taking requests to ensure that each peer does not carry the burden of many requests beyond a particular limit. It helps to prevent any peer from being overloaded. Also, they will work much faster as they divide the requests between nodes.

6.7 Test Plan

To evaluate the performance we executed the following test plan

- Using 100 threads to start and step up by 100 to 2000 threads to understand how the system performs when the number of concurrent users increases and to have an insight at what point the system begins to struggle.
- Implemented a ramp-up time of 5 seconds and loop count of 5 for each test to allow controlled testing with less traffic to more traffic by time. The sudden traffic also gives us an insight into how the system handles sudden spikes of concurrent users.
- Used 5 https requests all to the same endpoint but with different json body where 4 of them were expected permit and 1 deny to test both access control decisions as the system might encounter them in its practical application. In our assumption, eighty percent of requests will be permitted while the other twenty will be unauthorized or denied.
- Every test case was run three times and then the average was taken for more accuracy. This ensures that the sudden changes in behaviors were taken into account and the average provided more reliable and consistent results.
- Tried to identify the saturation point where the performance starts to decrease as the load increases. This is to understand the limitations of the system and identify how much load it can handle without falling apart.

6.8 Metrics Evaluation

In our tests we used some of the output metrics to judge the performance capability of our system. They are briefly explained down below:

i **Throughput:** Within a second the number of successful queries on the ledger can be handled. And, much better performance can be achieved by a higher throughput mostly if there is a high load. Throughput was chosen because it describes the system's capacity to grow and manage many transactions at once, which is crucial in distributed systems with high loads.

ii **Latency:** The significant time that is required to complete whole queries and operations. We chose latency because it gives the level of performance that a system can offer. Less latency means a higher processing power which is essential in quick decision making in a decentralized network.

iii **Error Rate:** The percentage of unsuccessful transactions and queries. A low error rate denotes that the system is more trustworthy and is capable of managing more requests. The reason why error rate was chosen is because it illustrates the stability and reliability of a system. A small error rate is important for raising transaction volumes to go through the system without leading to system failure while high error rates do the opposite.

iv **Response Time:** This test evaluates time taken for the system to give a response to the request from the time it was sent. The reason it was chosen because it directly affects the user experience. A lower response time means a quicker and more efficient system, which enhances a better or improved user experience with the system.

6.9 Observations

After testing thoroughly, we analyzed the results, and came up with the following observations:

- **Throughput Comparison:** Figure 6.1 compares the throughput of both one-peer fallback and Round-Robin scenarios. The Round-Robin consistently outperforms One-Peers fallback with the throughput increasing steadily from around 150 to nearly 180 requests per second as the number of loads increases. On the other hand, the One Peer Fallback configuration begins at a comparable point and trends up to about 150-160 requests per second after which it hardly scales up regardless of the amount of load it receives. This shows that the Round Robin which distributes the load equally among peers, is much more efficient at handling large loads than One Peer Fallback which sacrifices performance for stability. That's why Round Robin is more suitable for scenarios that require high scalability.

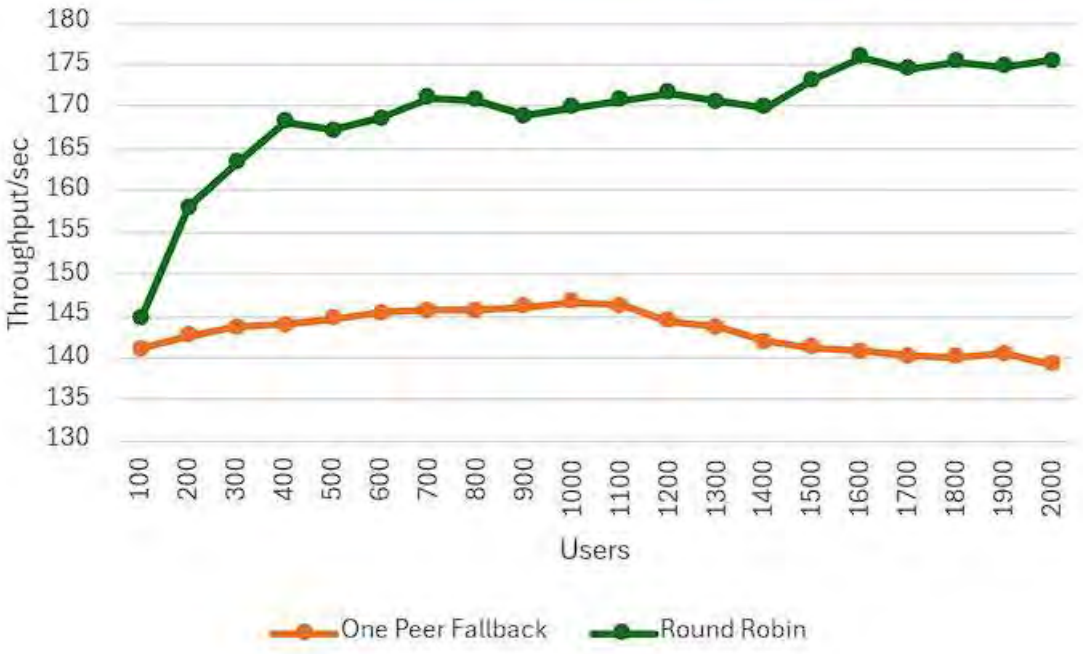


Figure 6.1: Throughput

- **ART Comparison:** Figure 6.2 compares the response time of both one-peer fallback and Round-Robin scenarios. In both scenarios, the response time grows as the number of users increase. The One Peer Fallback shows reliably higher response times than the Round Robin and the difference increases with each additional user level. This graph indicates that the One Peer Fallback performs less efficiently under heavier loads than the Round Robin does and shows lower than average response times throughout its request handling.

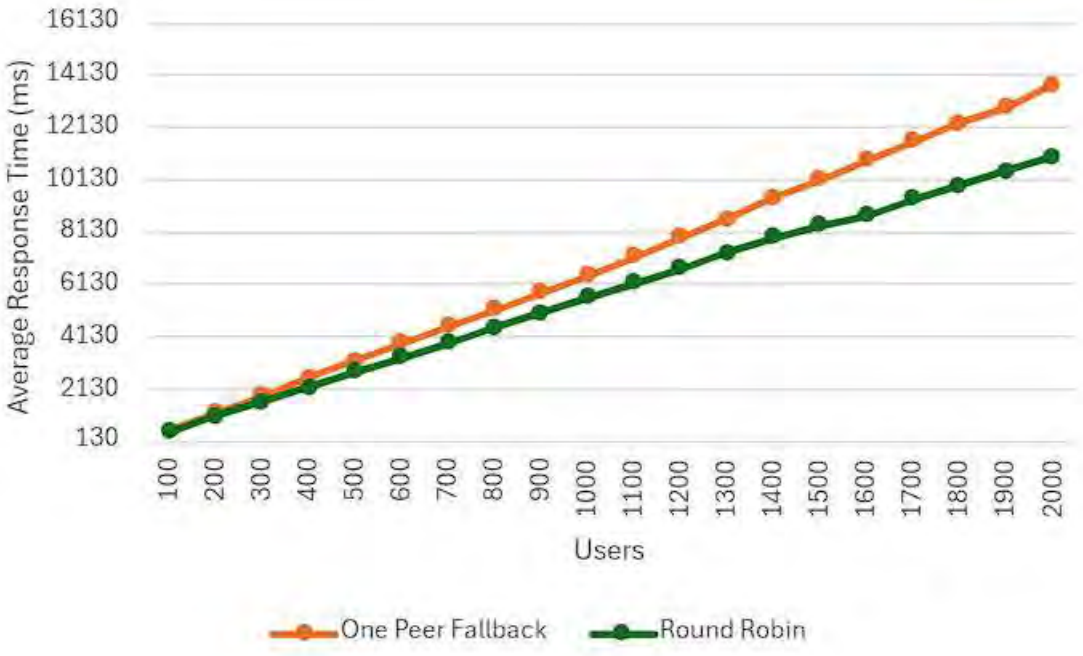


Figure 6.2: Average Response Time

- **Latency Comparison:** Figure 6.3 illustrates the average latency between One-Peer fallback and Round-Robin scenarios where both of them have high-level latency as the number of users grows with increasing number of requests. The Round-Robin shows lower latency than one peer fallback in all scenarios.

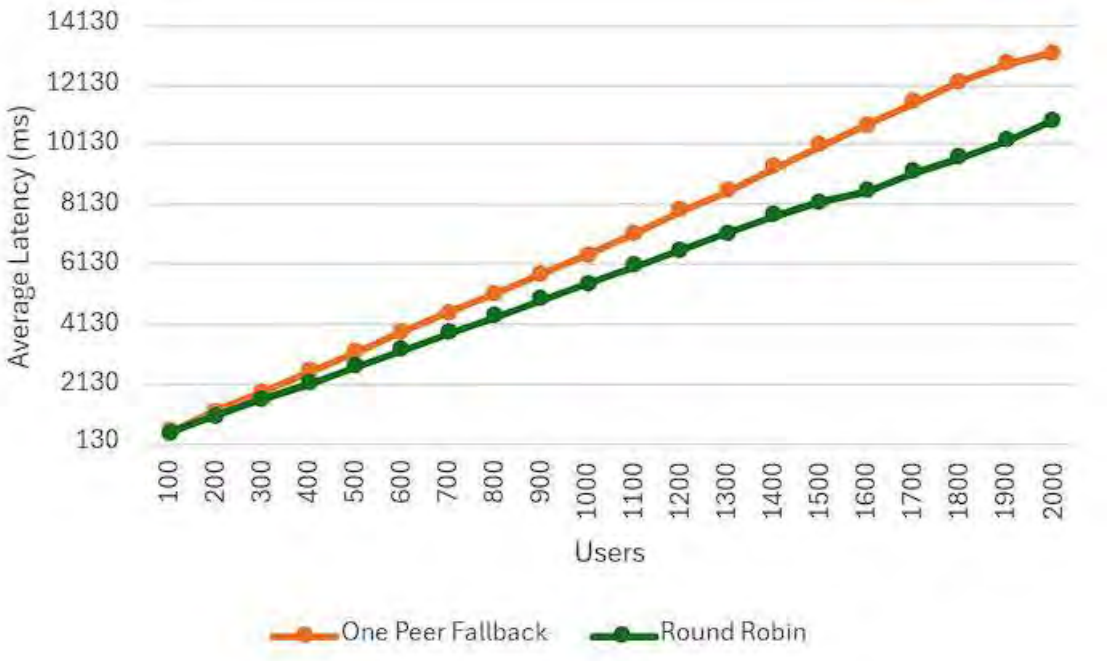


Figure 6.3: Average Latency

- **Overall Performance:** The response time, throughput, and latency rates show that the Round Robin configuration is better than the One Peer Fallback in the higher request volume. Balancing load between multiple peers across the network to distribute the load can mitigate bottlenecks and vulnerability to DoS attacks. This scalability and redundancy allow for the management of traffic loads to be more efficient which in turn makes Round Robin more capable of handling the effects of DoS attacks.

Chapter 7

Discussion

In this chapter, the functional and security requirements are analyzed. Research objectives are assessed along with the discussion of advantages, challenges or limitations and future works.

7.1 Functional Requirement Analysis

This section evaluates how the implemented decentralized access control system meets each of the defined functional requirements:

FR1. The system successfully fulfills the overall objective of integrating chaincodes in defining the access control policies and also to easily enforce them in the blockchain network. This keeps policy application consistent across the system while eliminating the need for a control center to monitor the access controls, thus satisfying the concerns of needing strong access control mechanisms.

FR2. This requirement was also fulfilled as Hyperledger fabric establishes a highly decentralized model of decision-making on access control by distributing the responsibilities among multiple nodes. Besides decentralizing the access control, the integration of this process also uses blockchain security characteristics that improve the overall stability of the system.

FR3. We have developed chaincodes for all the XACML components for decision-making in access control to be accurate and deployed them inside the Hyperledger Fabric network. This capability enables the organization to meet automatic and accurate access control decisions.

FR4. We implemented our theory system in an event management service which proves its viability in legacy systems. This practical implementation demonstrates the function of the system and how decentralized access control can be easily integrated into existing organizational structures.

FR5. Due to the design of the architecture, the scalability can be increased with increasing the amount of peers involved. The modular scalability approach fulfills this requirement. It also proves that the system's performance and reliability improves with the network size which makes it suitable for many organizations.

7.2 Security Requirement Analysis:

In this section we assessed how the implemented decentralized access control system satisfies each of the security requirements that have been outlined:

SR1. Authentication in this system is made secure by SAML authentication using the WSO2 Identity Server which uses digital certificates for identification. The use of this setup allows for proper and secure validation of user identities within the network to meet the condition of strong authentication.

SR2. The integration of the WSO2 Identity Server fulfills this requirement, as it applies hashing mechanisms for securely storing user data.

SR3. Non-repudiation is achieved with the help of the digital signatures used by each peer of Hyperledger Fabric. This setup makes it possible for actions that take place within the system to be proven and no one can deny it after the event.

SR4. Chaincode-based access control decisions and transaction confirmation mean that the system restricts access to the data and prevents unauthorized information leakage.

SR5. Since the system is decentralized, it will be relatively easier to defend against DoS attacks. This security can be improved by adding more peers as attacking one peer will not affect the whole system. More peers will give more security.

SR6. Our access control mechanism is strong enough to prevent attackers from elevating their privileges. One would have to bypass many security layers such as the digital certificates of peers and WSO2IS to gain this control.

SR7. Handled effectively by the digital signature mechanisms within the WSO2 Identity Server.

SR8. In this research, we are not focused on counteracting S8 as it is out of our research scope. S8 refers to an XML External Entity, which deals with vulnerabilities in how XML parsers process external entities and our research is solely on decentralizing authorization engine.

7.3 Research Objective Analysis:

In this section, we analyzed if the implemented decentralized access control system fulfills our research objectives:

RO1. The objective was fulfilled as the system proposed to decentralize the access control model to distribute the access decision and policy management across multiple nodes in a blockchain network after analyzing the vulnerabilities of the centralized access control system.

RO2. It was also successfully fulfilled as in the Architecture section we have proposed a decentralized model for XACML which uses Hyperledger Fabric which later in implementation proved to solve SPOF.

RO3. Fulfilled this objective by developing chaincodes of PEP, PDP, PAP, and PIP and deployed them in Hyperledger Fabric to propagate it among multiple peers to achieve decentralization.

RO4. Based on performance results obtained under higher loads, the system can be scalable so this objective was also achieved. Also according to our evaluation, the scalability can be increased by adding more nodes to the network.

7.4 Advantages

Incorporating decentralized access control system in access control models provide a number of advantages which are discussed below:

- In this system, the decision-making process is distributed among multiple peers. Thus it does not depend on any single node. So if any nodes go offline other nodes will make up. So it eliminates the multiple threats.
- With the implementation of Hyperledger Fabric, the access requests can be propagated across several peers. This is helpful in terms of handling more requests simultaneously, which makes the response time better.
- We have deployed the XACML components (PIP, PDP, PEP and PAP) as chaincode in Hyperledger Fabric. Once these components are deployed their functionality cannot be modified which ensures the security and reliability of the access control decisions. While the chaincode can be upgraded if necessary, the access control logic stays consistent and temper-proof after each deployment layer. Thus, the integrity of access control decisions is served within the system.
- Since we are working with Hyperledger Fabric, the roles and policies are set within the distributed ledger. If there is a modification, it will be transparent within the distributed ledger.
- We have also added SAML for authentication as an extra layer of security and SSO functionality. Using this also allowed having other necessary information for each user.

7.5 Challenges And Limitations

Along with all its advantages decentralized access control systems also come with many challenges and limitations on its own that are discussed next:

- Since the number of users and policies increases gradually, the management of such users and policies becomes challenging, and this slows down policy evaluation and decision-making processes.
- The performance of the system decreases with high loads due to the time required for the inter-chaincode communication. This is especially felt during load testing.
- To get better scalability, we have to increase the number of peers, which makes it more expensive in real-life scenarios.

- The use of WSO2 for authentication creates a new dependency external to the system. Any problem related to WSO2 could result in compromising the performance or the availability of the system.

7.6 Future Works

In future, we intend to work on the following:

1. We intend to work on dynamic node management. For decentralizing access control, multiple nodes might need to be managed according to the needs of organizations. Handling those nodes dynamically would make the system less dependent on raw manpower making it more efficient.
2. We intend to optimize the chaincode more for better scalability. In our limited research scope, we developed the chaincodes to serve the functionality but we felt it could be more optimized for better scalability. Thus, Scalability considerations are taken into account.
3. We would like to combine our implementation with other access control systems like ABAC, ACL, and some others. Analyzing how these systems can take advantage of the decentralization of the blockchain could reveal opportunities for managing security in a more adaptive, sensitive manner.

Chapter 8

Conclusion

Authorization in almost every model is mostly done using a centralized system which makes it vulnerable to many security threats. Centralized authorization models constantly leave enterprises open to large hacks and data branches. Anyone who is putting their trust in technology with little to no knowledge is being harassed without even knowing the mistake. This research highlights that blockchain can transform the current access control system from a centralized to a decentralized model. This also not only has a positive impact on security by eliminating the SPOF but also on scale and transparency. In this paper, we have discussed how XACML can be completely decentralized using Hyperledger Fabric and shown that blockchain offers a reliable and fault-tolerant approach for the management of access control that can be applied widely. Our findings and theories are also implemented in a real-world enterprise-like service which confirms the practical application and effectiveness of our decentralized access control. As our system is decentralized the control is distributed throughout the network, which makes the system more redundant and safe from the risks of a centralized model. The practical implementation of our proposed system for handling dynamic access control has provided evidence that it works effectively and keeps integrity and confidentiality while producing correct authorization decisions. In the future, We intend to work on the efficiency of our system. Additionally, Future studies could be about how a more sophisticated consensus mechanism could be implemented into the system to enhance performance and security. Furthermore, we can also research how it can be implemented in emerging technologies such as AI, machine learning, and the Internet of Things and how our findings could help them to have stronger security and faster responses.

Thus, our work can be regarded as a valuable contribution to the further enhancement of decentralized applications as it offers a scalable and secure access control solution.

Bibliography

[1] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996. DOI: 10.1109/2.485845.

[2] Q. Li, M. Xu, and X. Zhang, "Towards a group-based rbac model and decentralized user-role administration," in *2008 The 28th International Conference on Distributed Computing Systems Workshops*, 2008, pp. 441–446. DOI: 10.1109/ICDCS.Workshops.2008.26.

[3] R. Tamassia, D. Yao, and W. H. Winsborough, "Independently verifiable decentralized role-based delegation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 6, pp. 1206–1219, 2010. DOI: 10.1109/TSMCA.2010.2045118.

[4] A. Shostack, *Threat Modeling: Designing for Security*. Wiley, 2014, ISBN: 9781118809990. [Online]. Available: https://books.google.com.bd/books?id=asPDAgAAQBAJ.

[5] A. Dresch, D. P. Lacerda, and J. A. V. Antunes Jr, *Design Science Research*. Cham: Springer International Publishing, 2015, ISBN: 9783319073736. DOI: https://doi.org/10.1007/978-3-319-07374-3.

[6] R. Bagchi. May 2017. [Online]. Available: https://core.ac.uk/reader/132491552.

[7] U. Jamsrandorj, "Decentralized access control using blockchain," Ph.D. dissertation, University of Saskatchewan, 2017.

[8] E. Androulaki, A. Barger, V. Bortnikov, *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18, Porto, Portugal: Association for Computing Machinery, 2018, ISBN: 9781450355841. DOI: 10.1145/3190508.3190538. [Online]. Available: https://doi.org/10.1145/3190508.3190538.

[9] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract," *IEEE Access*, vol. PP, pp. 1–1, Mar. 2018. DOI: 10.1109/ACCESS.2018.2812844.

[10] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1499–1506. DOI: 10.1109/Cybermatics_2018.2018.00253.

[11] R. Xu, Y. Chen, E. Blasch, and G. Chen, "Blendcac: A blockchain-enabled decentralized capability-based access control for iots," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1027–1034. DOI: 10.1109/Cybermatics_2018.2018.00191.

[12] Docker, *Docker documentation*, Oct. 2019. [Online]. Available: https://docs.docker.com/.

[13] Git, *Git - documentation*, 2019. [Online]. Available: https://git-scm.com/doc.

[14] C. Ihle and O. Sanchez, "Smart contract-based role management on the blockchain," in *Business Information Systems Workshops*, W. Abramowicz and A. Paschke, Eds., Cham: Springer International Publishing, 2019, pp. 335–343, ISBN: 978-3-030-04849-5. DOI: 10.1007/978-3-030-04849-5_30.

[15] I. Markus, L. Xu, I. Subhod, and N. Nayab, "Dacc: Decentralized ledger based access control for enterprise applications," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 345–351. DOI: 10.1109/BLOC.2019.8751479.

[16] S. Michael and Z. J. Anna, "An identity provider as a service platform for the edugain research and education community," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 739–740.

[17] L. Pawczuk, B. Hansen, R. Massey, and J. Holdowsky, *Deloitte's 2020 global blockchain survey*, 2020. [Online]. Available: https://www2.deloitte.com/content/dam/insights/us/articles/6608_2020-global-blockchain-survey/DI_CIR%202020%20global%20blockchain%20survey.pdf.

[18] M. Shen, L. Zhu, and K. Xu, "Blockchain and data sharing," in *Blockchain: Empowering Secure Data Sharing*. Singapore: Springer Singapore, 2020, pp. 15–27, ISBN: 978-981-15-5939-6. DOI: 10.1007/978-981-15-5939-6_2. [Online]. Available: https://doi.org/10.1007/978-981-15-5939-6_2.

[19] I. Alom, R. M. Eshita, A. Ibna Harun, *et al.*, "Dynamic management of identity federations using blockchain," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–9. DOI: 10.1109/ICBC51069.2021.9461128.

[20] *Hyperledger explorer screens — hyperledger explorer documentation*$_2$021, 2021. [Online]. Available: https://blockchain-explorer.readthedocs.io/en/main/presentation/index.html.

[21] X. Li, J. Yang, S. Gao, Z. Shi, J. Li, and X. Fu, "Dbs: Blockchain-based privacy-preserving rbac in iot," in Nov. 2021, pp. 94–110, ISBN: 978-3-030-91423-3. DOI: 10.1007/978-3-030-91424-0_6.

[22] P. Mukherjee and C. Pradhan, "Blockchain 1.0 to blockchain 4.0—the evolutionary transformation of blockchain technology," in May 2021, pp. 29–49, ISBN: 978-3-030-69394-7. DOI: 10.1007/978-3-030-69395-4_3.

[23] A. Rashid, A. Masood, and A. u. R. Khan, "Rc-aam: Blockchain-enabled decentralized role-centric authentication and access management for distributed organizations," *Cluster Computing*, vol. 24, no. 4, pp. 3551–3571, Dec. 2021, ISSN: 1386-7857. DOI: 10.1007/s10586-021-03352-x. [Online]. Available: https://doi.org/10.1007/s10586-021-03352-x.

[24] Y. Zhang, M. Yutaka, M. Sasabe, and S. Kasahara, "Attribute-based access control for smart cities: A smart-contract-driven framework," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6372–6384, 2021. DOI: 10.1109/JIOT.2020.3033434.

[25] M. V. Akhil Vasishta, B. Palanisamy, and S. Sural, "Decentralized authorization using hyperledger fabric," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 238–243. DOI: 10.1109/Blockchain55522.2022.00040.

[26] S. Craß, A. Lackner, N. Begic, S. A. M. Mirhosseini, and N. Kirchmayr, "Collaborative administration of role-based access control in smart contracts," in *2022 4th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2022, pp. 87–94. DOI: 10.1109/BRAINS55737.2022.9909116.

[27] V. C. Hu, *Blockchain for access control systems*, en, May 2022. DOI: https://doi.org/10.6028/NIST.IR.8403. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934417.

[28] W. Viriyasitavat, L. D. Xu, D. Niyato, Z. Bi, and D. Hoonsopon, "Applications of blockchain in business processes: A comprehensive review," *IEEE Access*, vol. 10, pp. 118 900–118 925, 2022. DOI: 10.1109/ACCESS.2022.3217794.

[29] *bdnews24*, May 2023. [Online]. Available: https://bdnews24.com/bangladesh/dnc2b2cvhp.

[30] Z. Islam, S. Rahman, and Z. Faiaz, "Hackers feast on government sites," *The Daily Star*, Jul. 2023. [Online]. Available: https://www.thedailystar.net/news/bangladesh/crime-justice/news/hackers-feast-government-sites-3364261.

[31] Wikipedia, *Blockchain — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1195541539, [Online; accessed 24-January-2024], 2024.

[32] Wikipedia, *Security Assertion Markup Language — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Security%20Assertion%20Markup%20Language&oldid=1189552494, [Online; accessed 24-January-2024], 2024.

[33] Wikipedia, *Single sign-on — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Single%20sign-on&oldid=1193816139, [Online; accessed 24-January-2024], 2024.

[34] Wikipedia, *XML — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=XML&oldid=1196753148, [Online; accessed 24-January-2024], 2024.

[35] WSO2, *Home - wso2 identity server*, 2024. [Online]. Available: https://is.docs.wso2.com/en/latest/.

[36] *Curl - documentation overview*. [Online]. Available: https://curl.se/docs/.

[37] *Documentation - the go programming language*. [Online]. Available: https://go.dev/doc/.

[38] *A blockchain platform for the enterprise — hyperledger-fabricdocs main documentation*. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/.

[39] *Apache jmeter-user's manual*$_2$019, 2019. [Online]. Available: https://jmeter.apache.org/usermanual/index.html.

[40] *Introduction to node.js*. [Online]. Available: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs.

[41] P. S. Foundation, *Welcome to python.org*, 2019. [Online]. Available: https://www.python.org/doc/.

[42] V. S. Code, *Documentation for visual studio code*, 2023. [Online]. Available: https://code.visualstudio.com/docs.