DECENTRALIZED ACCESS CONTROL USING BLOCKCHAIN

A Thesis Submitted to the

College of Graduate and Postdoctoral Studies

In Partial Fulfillment of the Requirements

for the degree of Master of Science

In the Department of Computer Science

University of Saskatchewan

Saskatoon

By

Uurtsaikh Jamsrandorj

Permission to Use

ABSTRACT

To effectively participate in modern collaborations, member organizations should be able to share digital resources with various partners, while ensuring their digital resources are protected from inappropriate access. In the existing literature, a substantial amount of research on centralized access control in the context of a single organization has been carried out. However, research on decentralized access control in a collaborative environment is scarce. To advance research in this area, I implemented a prototype of a decentralized access control system, which supports transparency, auditability, immutability, and equality in a collaboration environment, using the Multichain blockchain platform, RESTful web services, and Java programming language. The prototype was developed to evaluate two primary metrics: *average response time* and *throughput*. To achieve this, I carried out a number of experiments to measure both metrics on a Local Area Network (LAN) and on Amazon Web Services (AWS) under 84 different experimental conditions. The LAN setup provides a baseline for system performance under optimal conditions, while the cloud infrastructure represents a real-world use case. With low-system loads (comprising one to thirty concurrent clients and a single server running the system), the LAN setup outperformed the AWS setup by a factor of 2.5 based on throughput. On the other hand, when the system load is significantly increased, with more servers running the system, the AWS and LAN setups showed only a marginal difference in their performance. This demonstrates the potential to horizontally scale the decentralized access control system using blockchain on cloud infrastructure.

ACKNOWLEDGEMENTS

I would like to thank all the people, who contributed in various ways to the research work described in this thesis.

First, I would like to express my deepest appreciation and genuine thanks to Dr. Ralph Deters, who has supported me during my research with his patient guidance, encouragement, and advice. His knowledge and continuous support were crucial for my master's study. Next, I would like to express my appreciation goes to graduate chair Dr. Julita Vassileva, who encouraged and supported me throughout my study. In addition, I would like to thank my committee members: Dr. Julita Vassileva, Dr. Jim Greer, and Dr. Anh Dinh. Also, I would like to thank Gwen Lancaster for her help and support during my study at the University of Saskatchewan. Furthermore, I will like to thank all students of the Multi-User Adaptive Distributed Mobile and Ubiquitous Computing (MADMUC) Lab for their friendship and selfless support.

On a personal level, I thank my entire family for their unconditional love and generosity. I would also like to thank my friend (Battushig Battseren) and his wife (Tselmeg Ulzii), who always encourage and support me.

Finally, I wish to express my love and gratitude to my wife (Ankhbayar Naidansuren) and my daughter (Khaliun Uurtsaikh).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABAC	Attribute-Based Access Control
ACL	Access Control List
AI	Artificial Intelligence
AWS	Amazon Web Services
DAC	Discretionary Access Control
DDoS	Distributed Denial of Service
HTTP	Hypertext Transfer Protocol
JSON	Java Script Object Notation
LAN	Local area network
MAC	Mandatory Access Control
NIST	National Institute of Standards and Technology
OAuth	Open Authorization
OrBAC	Organization-based Access Control
P2P	Peer-To-Peer
PBFT	Practical Byzantine Fault Tolerance
PoI	Proof of Importance
PoS	Proof of Stake
PoW	Proof of Work
RBAC	Role-based Access Control
REST	Representational State Transfer
SAML	Security Assertion Markup Language
SNS	Social networking sites
SPV	Simplified Payment Verification
TXO	Transaction Outputs
U of S	University of Saskatchewan
UTXO	Unspent Transaction Outputs
XACML	eXtensible Access Control Markup Language

CHAPTER 1

1. INTRODUCTION

Information systems are becoming complex, highly distributed, very dynamic systems that extend across organizational boundaries in the digital world. Organizations need to collaborate, share digital resources, and exchange information, allowing them to use techniques like data mining and analysis to support business decisions. Organizations require the ability to share some resources while also maintaining control and data protection over those resources. However, there is a conflict between data sharing and data protection. In an industrial information system, protecting data and information from cyber-attacks becomes one of the greatest challenges for organizations. As cyber-attacks evolve in new and more complicated ways, security organizations have difficulty keeping up. Cyber-attacks create threats to organizations' security, which costs billions of dollars each year [1]. FACC is an Austrian-based aerospace parts manufacturer working with Airbus and Boeing. They announced being a victim of hacking, and the cost of its implication was about 54.5 million U.S. dollars in January 2016 [2]. In 2016, Yahoo reported that they lost 500 million users' information, and it was the largest data breach in the history [3]. In addition, in 2012, 117 million emails and passwords were stolen by hackers from LinkedIn, and it became publicly available in May 2016 [4].

Thus, organizations need to have proper administration and security policies to maintain data security while allowing selective sharing of resources. The domain of access control provides solutions for the problems of authentication, authorization, and validation. Authentication proves the user's identity; authorization specifies and evaluates a set of access policies that define which user can access which resources and, validation securely verifies the authorized privileges and authenticated identities [5]. In the dynamic and collaborative environment, classical models of access control are not effective in protecting resources while allowing users to access to the resources within their privileges.

Blockchain is an underlying technology and closely linked to the digital currency Bitcoin, which was introduced in 2008 with the white paper *Bitcoin:A Peer-to-Peer Electronic Cash System* by an unknown author under the pseudonym Satoshi Nakamoto [6]. Although blockchain technology was initially conceptualized during the beginning of Bitcoin, it has been abstracted to refer to a distributed and decentralized public ledger that keeps all transactions or events which have been executed and shared among participants in a peer-to-peer network. Every transaction in the ledger is verified by a consensus mechanism in the system, and it is immutable in the blockchain. Additionally, blockchain technology is an emerging solution for decentralized data sharing allowing an extensive network of untrusted parties for sharing resources.

The objective of this research is to create a decentralized access control system using a private Blockchain to facilitate data sharing across organizational boundaries. The rest of the thesis is organized as following chapters:

- Chapter 2 introduces and defines the problems
- Chapter 3 reviews the related technologies and research
- Chapter 4 presents the architectural design and implementation
- Chapter 5 describes the evaluation of experiments and discusses the results
- Chapter 6 concludes the contributions of this thesis and discusses future work

CHAPTER 2

2. PROBLEM DEFINITION

In today's digitalized world, systems are usually distributed, not managed by a single organization, and require collaboration of multiple participants who provide data and offer services or computational resources. This dynamic collaborative environment requires suitable access control systems that enable participants to define who can access their data and services. These security requirements dynamically change in a distributed system in different organizations, as each organization needs to be able to manage their own access control independently [7]. For instance, universities already have their own access control system within the organization, and when they need to collaborate among each other, the collaboration will require an additional security layer or access control system.

Organizations usually store and manage their access control data in centralized servers when they use distributed systems. A centralized access control system manages the permissions of users to access the centralized data storage [8]. Therefore, every organization manages its independent access control system. In centralized access control, all access requests go through a central authority, which decides whether grant or deny. This approach simplifies administration of access control because all configuration and management can be managed only by a single entity. However, a centralized approach has the following drawbacks:

a. **Single point of failure** - if the centralized access control fails then no one can access the entire system
b. **Central authority** - all requests must be approved by the central authority, which can cause performance problems
c. **Limited transparency** - Since there is centralized control in a collaborative environment, transparency becomes another challenging issue

Traditional access control models DAC (Discretionary Access Control) and MAC (Mandatory Access Control) are not readily adapted to distributed and heterogeneous systems. RBAC (Role-based Access Control) is a more flexible and widely used access control model. Roles are assigned to users, and permissions are assigned to the roles. RBAC also supports hierarchical structuring of the roles. Similarly, the OrBAC (Organization-based Access Control) model is an extension of

RBAC that defines permissions independently from implementation and helps organizations to define their security policies. Also, the OrBAC model does not adapt to a multi-organizational collaborative environment, when there are large-scale independent organizations with decentralized systems, as each organization keeps its own access control information [9], [10]. Therefore, these classical access control models are well suited and purposed to heterogeneous and distributed systems.

There is a lack of suitable security models for the collaboration of participants. Although some research has been done on decentralized access control [11]–[13], most approaches do not support fine-grained access control policies nor transferring the access rights. To follow the principle of least privilege, permissions should be defined in a fine-grained way. Also, in decentralized systems, the access policies should be self-descriptive.

Thus, building an access control for cross-boundary organization is an important and challenging issue.

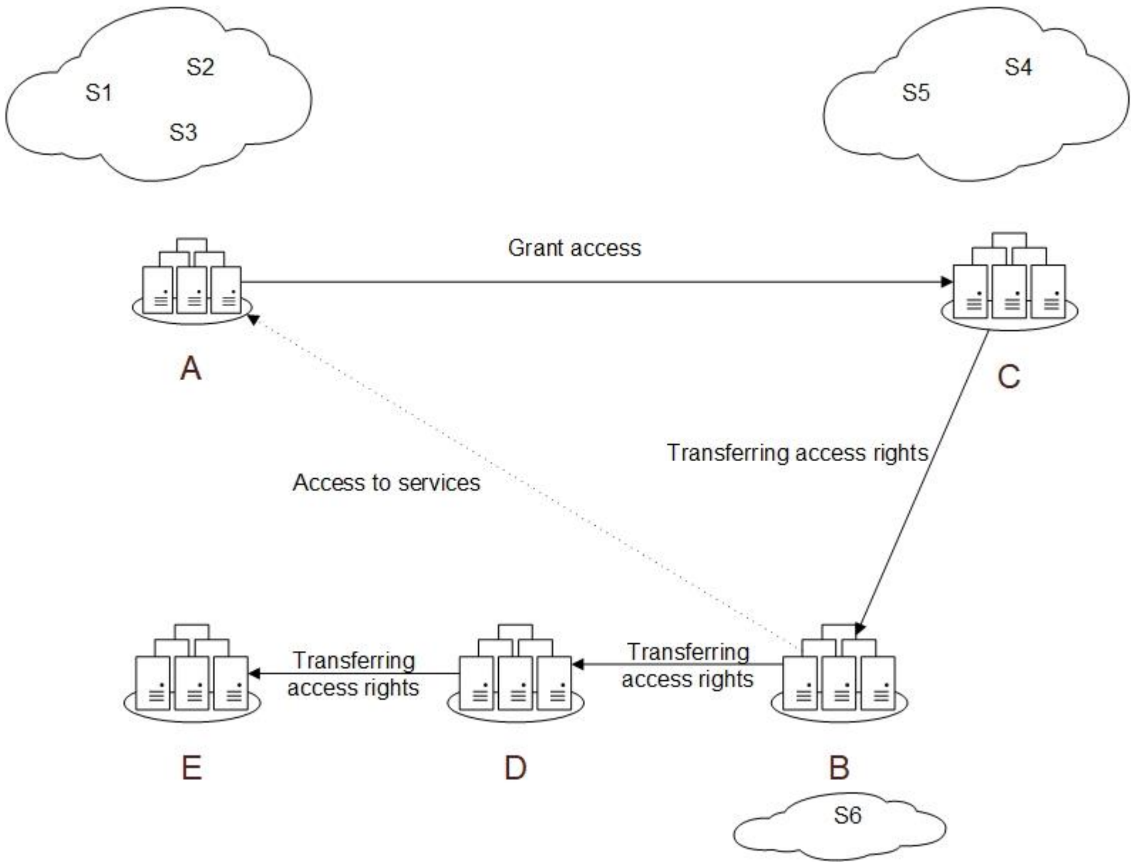

S1
S2
S3
S5
S4
Grant access
A
C
Transferring access rights
Access to services
Transferring access rights
Transferring access rights
E
D
B
S6

Figure 2-1: Problem definition

Figure 2-1 describes a scenario where organizations want to share digital resources and transfer access rights from one organization to another organization. For example, Organization A grants the access to Organization C to give access to its micro services, which are S1, S2, and S3. Then, if Organization C would like to transfer its access permissions to Organization B, the question is how organization A can grant access to Organization B based on transferred access rights.

2.1 Research Goals

The primary goal of the thesis is to establish a decentralized access control system, which provides the opportunity to share digital resources and transfer defined access rights from one organization to another organization without having a central authority across organizational boundaries. In order to be able to answer this broad question, the following issues must be addressed:

a) How to create a decentralized access control system?

In a decentralized and collaborative environment, centralized access control systems are not a suitable solution, so developing decentralized access control mechanism is crucial for decentralized systems.

b) How can organizations grant or transfer access rights of micro services from one organization to another?

Traditional access control systems are mostly centralized, so there is a single place to store access permissions. When the access control systems are centralized, granting and transferring access rights becomes a challenge. With decentralized systems in a collaborative environment, access control systems need to grant access rights in a decentralized architecture. Also, transferring the access rights is an essential feature for a decentralized access control system.

c) How can consumer organization directly access the provider organization's micro services?

When the resource provider organization grants access rights for its own microservices, the challenge is how the consumer organization can directly access the microservices without having additional third party services.

CHAPTER 3

3. LITERATURE REVIEW

This chapter presents a review of the work related to decentralized access control system and Blockchain technology. It also presents introductions to several technical areas which will be helpful to understand the remainder of this thesis. First, I introduce the concept of access control, a summary of existing access control models in the literature, centralized and decentralized access control, temporality in access control, and open authentication. Second, I provide a discussion of Blockchain technology including public and private blockchains, disintermediation and equality in Blockchain, and Multichain Blockchain.

3.1. Access control

Samarati and de Vimercati [14] defined Access Control as "*the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied*".

The goal of access control is to check and restrict the actions that authorized users can perform within a computer system. Access control restrictions include what a user could perform directly as well as what programs can be executed on behalf of the users who are permitted to execute those programs. Access controls depend on cooperation with other existing security services [15]. Chapple et al. defined access control as *"the formalization of those rules for allowing or denying access. Access controls define the allowable interactions between subjects and objects. It is based on the granting of rights, or privileges to a subject with respect to an object"* [16]. The following concepts are commonly used in the access control community.

- **Object:** Resource to which the subject requests access (file, database, network resource, cloud service, and device, etc.)
- **Subject:** in the form of user, device, process, or application requesting to access an object
- **Operations:** A process initiated by a subject
- **Permission(privilege):** The rules that govern who has access to which resources (For example, a manager may have permissions to see the dashboards while an accountant may keep track of the transactions on the general ledger)

- **Access Control List (ACL):** Access control lists are a list of permissions that are associated with an object. It specifies who or which processes are granted access to the given object. For example, if Alice has read and write permissions, and Michael has to delete permission a given file, an access control list representation might be (Alice: read, write; Michael: delete).
- **Access control matrix:** The access control matrix is a two-dimensional table, which captures subjects as columns and objects as rows. Each cell shows the permission of the corresponding column(subject) and row(object).

The development of access control systems is typically a multi-stage based on the following concepts:

- **Security policy:** A set of high-level rules, which access control should be governed. (For example, managers can see to the dashboards)
- **Security model:** A formal representation specified and enforced by security policies. It may establish a formal model of access rights, a model of computation, or model of distributed computing.
- **Security mechanism:** A low-level implementation based on the security policy and model. It can be hardware or software.

Access control models

Three traditional access control models are mandatory access control (MAC), discretionary access control (DAC), and role-based access control (RBAC). Additionally, another commonly used and more logical access control model is attribute-based access control(ABAC).

3.1.1 Mandatory access control (MAC)

The mandatory access control policy grants access based on subjects and objects, which will be assigned security labels. These security labels are the combination of hierarchical classification levels and non-hierarchical compartments. The hierarchical order specifies the sensitivity of the data. For instance, an organization might have sensitivity levels Top Secret, Secret, Confidential, and Unclassified. Every object is associated with a set of compartments (nuclear, weapon, aircraft weapon, etc.) [17]. As an example, if an object is related to the {aircraft weapon, nuclear} it may be accessible by subjects, who need to know both about aircraft weapons and nuclear. In practice, a document might have the label (Top Secret, {aircraft weapon, nuclear weapon}) if it contains extremely sensitive information regarding aircraft weapon and nuclear weapon. Additionally, each paragraph of this document could be assigned a set of sensitivity levels and compartments, so the classification of this entire document would be the most restrictive classification given to each paragraph in the document. One of the well-known implementations of MAC is multi-level security that is commonly used on information technology systems deployed at highly sensitive government organizations such as the U.S. Department of Defense [18]. It is used in these scenarios as all security policies are centrally controlled by a security policy administrator, so users cannot override the security policies. In this case, MAC provides a high degree of security in MLS systems. Also, the underlying philosophy of MAC is that information belongs to an organization, so the organization should control the security policies of its information [19].

3.1.2 Discretionary access control (DAC)

Discretionary access control (DAC) provides the possibility for data owners to control their data accessed by themselves rather than by a central authority. DAC gives the possibility to users to grant and revoke access to objects, which are under their control. DAC is based on the notion that users are “owner” of objects; therefore, they will have complete control over their objects. Ownership usually refers to the creator of the object. DAC policies grant users access to information based on user’s identification and authorizations that state the user is allowed or denied access to information. Each user request will be assessed by these authorization rules [13]. A mechanism of DAC policy implementation needs to answer the following question: does a subject (S) have access right (AR) for an object (O)? More practically, the presentation of this policy can be represented as an Access control matrix. Each cell of the access control matrix includes a set of access rights. For example:

	Folder 1	File 1
Alice	rw	r
Bob	rw	w

Table 3-1: Access control matrix

3.1.3 Role-based access control (RBAC)

Role-based access control (RBAC) was formally introduced by Sandhu et al. in 2000 [20] and standardized by ANSI [21]. RBAC was initially proposed for enterprises where permissions will be granted to roles rather than individual users. A role is a combination of a set of users and set permissions. An organization can create roles based on various job functions and responsibilities, and users can be assigned these roles. The concept of roles logically maps to an organization's structure.

The idea of grouping privileges was initially proposed by Baldwin [22]. The core notion of RBAC is that users and permissions can be associated with roles. The idea of roles greatly simplifies the management of permissions because system administrators do not need to manage individual users and their permissions. Roles can be granted and revoked permissions to users of certain applications and systems as necessary [21]. Specifically, when role and permissions' relationships are predefined, assigning users to the predefined roles will be easy for the system administrators [23]. RBAC defines the following three main elements:

a) User - person, who interacts with a computer system
b) Role - job function or responsibility within an organization
c) Permission – an operation on an object

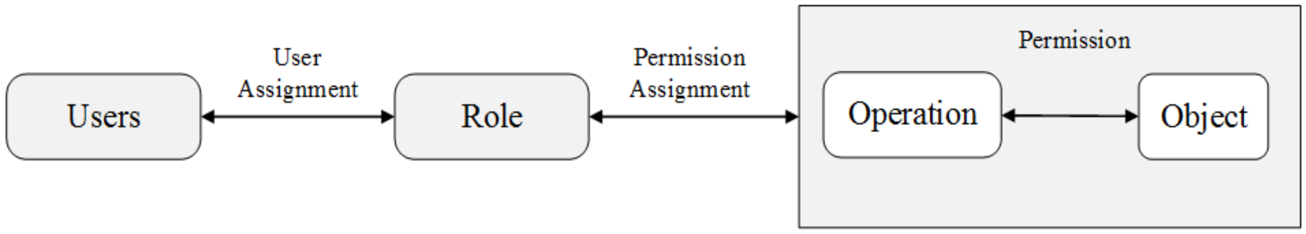


Figure 3-1: Basic elements of role-based access control [24]

One of the most important principles of access control is **least privilege**. This principle describes that only mandatory permissions, which are required by the user to execute tasks in a certain role, will be assigned to the role. The system should guarantee that the users will have only those permissions and no more [25]. Thus, it protects from the problem of users having the ability to perform unnecessary and potentially harmful actions. In some cases, a sensitive task may require two or more individuals to complete. A common example of these systems is a payment that requires two roles, which are payment initiator and payment authorizer. In this situation, RBAC applies the concept of **separation of duties** to enforce conflict of interest and other separation rules using RBAC elements. Also, RBAC uses the data abstraction concept. Instead of using the default operating system's permissions such as read, write, and execute, data abstraction allows the definition of abstract permissions [21].

3.1.4 Organization-based access control (OrBAC)

Organization-based access control (OrBAC) is an access control model proposed to define security policies at an organizational level. The central concept of OrBAC is the organization; an organization is an entity which is responsible for managing a security policy. The goal of OrBAC is to define a security policy at the organization level that is independent of its implementation. The OrBAC model proposes to use roles that subject, objects, and actions play within the organization rather than modelling the security policy based on the concepts of subject, object, and action. OrBAC uses eight main set of entities, which are Org (organizations), A (actions), S (subjects), O (objects), R (roles), *A*(activities), V (views), and C (contexts). OrBAC connects abstract entities (e.g. roles, activities, views) and real entities (e.g. empower, consider, use, hold, and obligation).

Each organization can specify roles that have the permissions, obligations, or prohibitions to do some activities in a given context. The context is used to express dynamic rules in OrBAC, and security rules are characterized by 5 predicates:

- permissions(Org,R, *A*, V, C) – means that the role R has the permission to perform activity A on the view V when the context C is true
- prohibition and obligations are similar to permissions, but differ in is that they define security requirements as well. The prohibition predicates express that a role cannot perform some activities on some views if a given context is not true. The obligation predicate defines that a role must execute some activities on some views when the related context is true.
- consider (Org, A, *A*) – states that an organization Org, action A implements activity *A*
- empower(Org, S, R) – states that subject S is empowered in the role R in the organization Org
- Use (Org, O, V) – states that object O is used in view V in the organization Org [26]

3.1.5 Attributed-based access control (ABAC)

In the National Institute of Standard and Technology(NIST), attribute-based access control(ABAC) is defined by a logical model that evaluates access rights using policies and conditions against attributes of the subject, object, and environment relevant to a request. **Attributes** are characterized by entities (subjects and objects), environmental conditions, and requested actions [27], [28]. Traditional access controls primarily consider static authorization decisions based on the subject's privileges on the target resources [29]. For example, RBAC decides the access right based on a subject's role that is associated with the permissions. The roles tend to be assigned to more static functions and positions within the organization. RBAC does not easily support complex access control decisions. For instance, if the decision needs to evaluate factors including user's physical location, the current time, the client type (PC, smartphone, etc.), and the client's IP address, RBAC will have challenges making this dynamic access control decision. If the organization tries to implement these types of access control system, it needs to create many roles that will lead to "role explosion" [28]. On the other hand, ABAC supports complex authorization process and constraints that are defined by the relationship between attributes of physical location, behavior, time, resource types and other additional information associated with access. In ABAC, attributes are variable and dynamic, while policies are relatively static. Therefore, ABAC can support cases, where users are dynamically changing, and ABAC can allow an access control without having the predefined information a subject [30].

3.1.6 Centralized and decentralized access controls

Access control administration can be centralized, decentralized, and hybrid, which uses both approaches. **A centralized access control** is a responsibility for granting or revoking an authorization to users, and it provides unified identity and authorization management across computer systems inside the organization. Centralized access controls simplify and help to manage the organization's security systems. One disadvantage of centralized access control is a single point of failure. **Decentralized access control** is when multiple places can grant access permissions for the users. This approach can be more stable than centralized access control, but it will require more administration and maintenance to protect several locations [30].

3.1.7 OAuth 2.0

Social networking sites(SNS) are the phenomenon that started in the 90’s. SNS including Facebook, Twitter, Google, and LinkedIn have become essential communication tools used by many people in their daily lives. Authenticating with these social network accounts is convenient for users rather than creating new accounts for each website. This is one of the reasons that OAuth (Open Authorization) has become common and useful technology [31]. OAuth 2.0 is an authorization framework, which allows users to grants third-party applications access to their resource without giving their security identity and passwords. OAuth defines the following four roles:

1) **Resource owner:** An entity that has the power to grant access permissions to other users to access its own resources.
2) **Resource Server:** The server that hosts the protected resources that are owned by the resource owners, and the server that can respond to access requests to resources based on access tokens.
3) **Client:** An application which makes a request to protected resources on behalf of the resource owner after obtaining authorization.
4) **Authorization server:** The server that issues access tokens to the clients after the resource owner successfully authenticates and gains authorization [32].

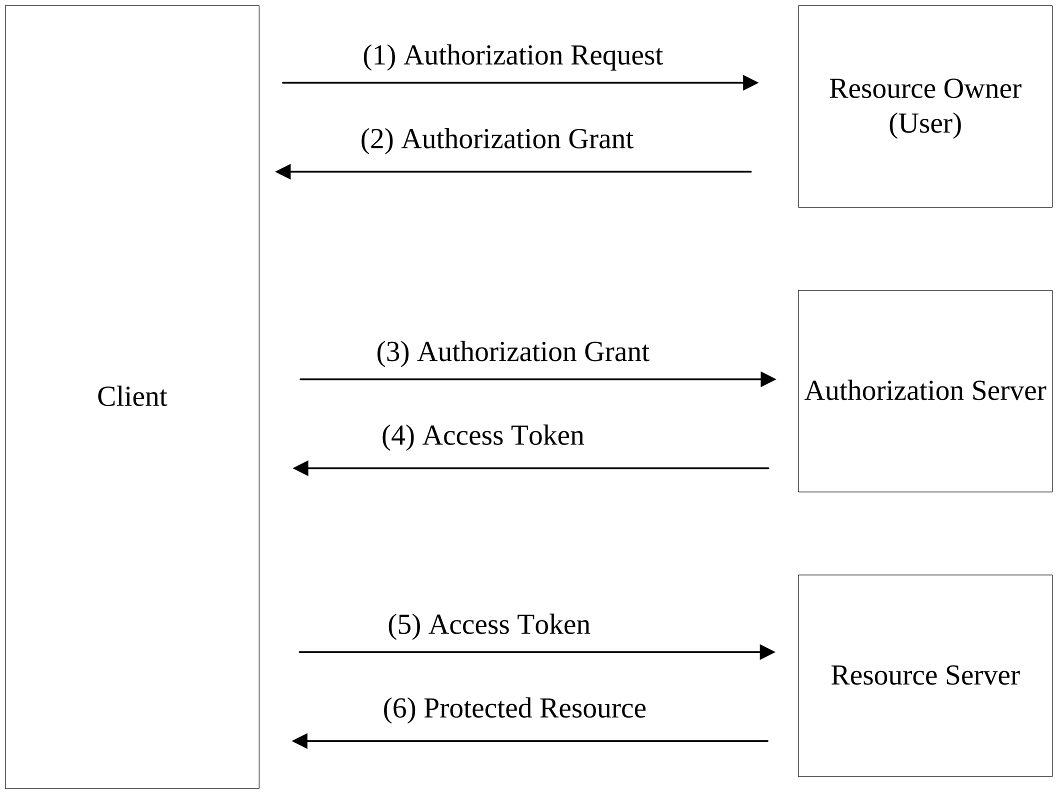


Figure 3-2: The abstract OAuth 2.0 flow

A) The application requests authorization to access the resource from the resource owner.

B) If the resource owner authorizes the request, the application receives the authorization grant.

C) The client sends an access token request using the authorization to the authorization server.

D) If the identity is authenticated and the authorization grant is valid, the authorization issues an access token and, it is sent to the application.

E) The application makes a request to the resource from the resource server using the access token.

F) If the access is valid, the resource server provides the requested resource.

3.2. Blockchain

Introduction

Blockchain technology is linked to the digital currency Bitcoin. In 2008, an individual or group of people under the name of Satoshi Nakamoto published a paper "Bitcoin: A Peer-To-Peer (P2P) Electronic Cash System", which described that Bitcoin is a digital currency, that can allow peer-to-peer transactions without having a trusted intermediary [6]. Bitcoin is a system that operates electronic money without any central authority, so it should have mechanisms to prevent falsification of data, payment duplication, and any kinds of attacks from malicious users. To achieve this goal, Bitcoin uses several different underlying technologies including hashing, public-key cryptography, digital signatures, P2P, and Proof of work. The following sections describe the core technologies of the blockchain.

3.2.1. Bitcoin: A peer-to-peer Electronic Cash System

Bitcoin is a cryptocurrency, which allows the transfer of digital assets without the need of trusted third parties, in the peer-to-peer network. The Bitcoin cryptocurrency is the most widely used and highly valued digital currency in the world [6], [33]. In order to control the ownership of each coin, a complete order of blocks is crucial for Bitcoin. For this reason, each block includes a pointer that points to the previous valid block in the chain. The point is a hash of the preceding block; therefore, the blockchain has the structure of the linked list.

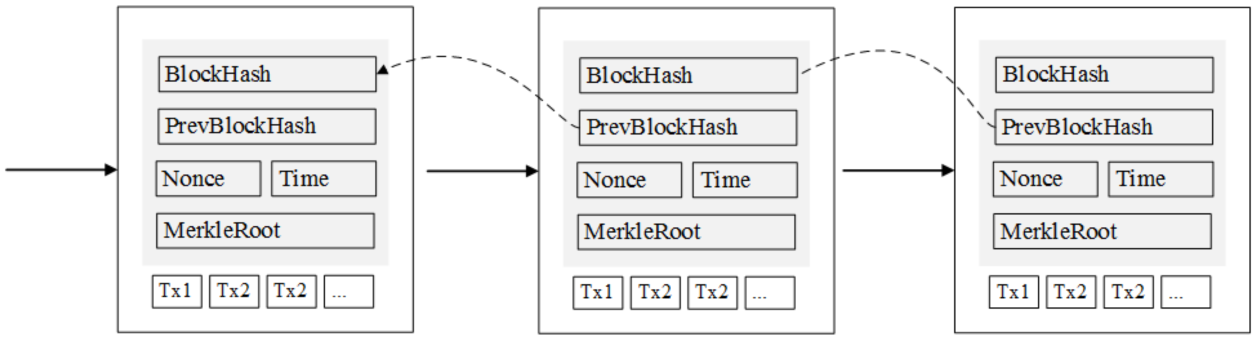


Figure 3-3: Simplified block chain

The number of blocks from beginning to end of the blockchain is called a *block height.* In general, the blockchain constantly grows, and this continuous increase will require an enormous amount of computational power for the validation process. To keep the size and computational effort reasonable, Bitcoin offers a simplified payment verification (SPV) based on Merkle trees [6].

3.2.2. Bitcoin transactions

A Bitcoin transaction is a movement of Bitcoin value which is broadcasted to the network and collected into blocks. Users send and receive bitcoins by sending digitally signed messages to the bitcoin network using a software called a bitcoin wallet. In another definition, **a bitcoin wallet** is a software tool with a built-in feature that can maintain a collection of addresses with corresponding private keys. The wallet enables users to issue outgoing transactions and verify whether the transaction is confirmed or rejected by the participants in the network, and to participate in peer voting [34]. The transactions are stored in a distributed, replicated public database (Blockchain) with the consensus mechanism accomplished by the PoW system called mining. In a blockchain, each transaction consumes one or more transaction outputs (TXO) within the sum of spendable bitcoins. These unspent sums are called "Unspent Transaction Outputs" (UTXO) [38]. A single transaction can contain one or more inputs and one or more outputs.

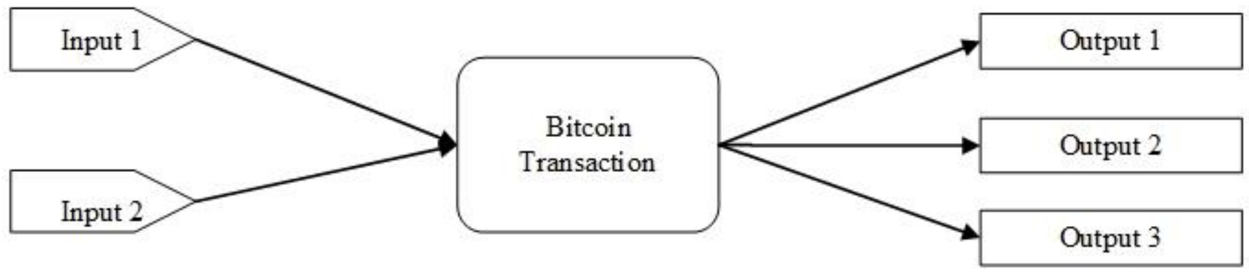


Figure 3-4: A single Bitcoin transaction allows for multiple inputs and outputs

A transaction includes a small fee attached to the transaction to incentivize miners to mine blocks. When miners create a block in the blockchain, they receive a small amount of this fee as well as a block reward that they can credit to their own wallets.

3.2.3. Hash function

A hash function *H* takes an input data *I* and returns a fixed-size string, which is called the hash value *h*. This mechanism provides that only the same input can result in the same hash value. If the input has a slight change, the hash value will be entirely different. Inferring the original data from the hash value is extremely difficult. The Bitcoin uses the calculation of hash values mechanism to guarantee the detection of data falsification and check continuation of blockchain data.

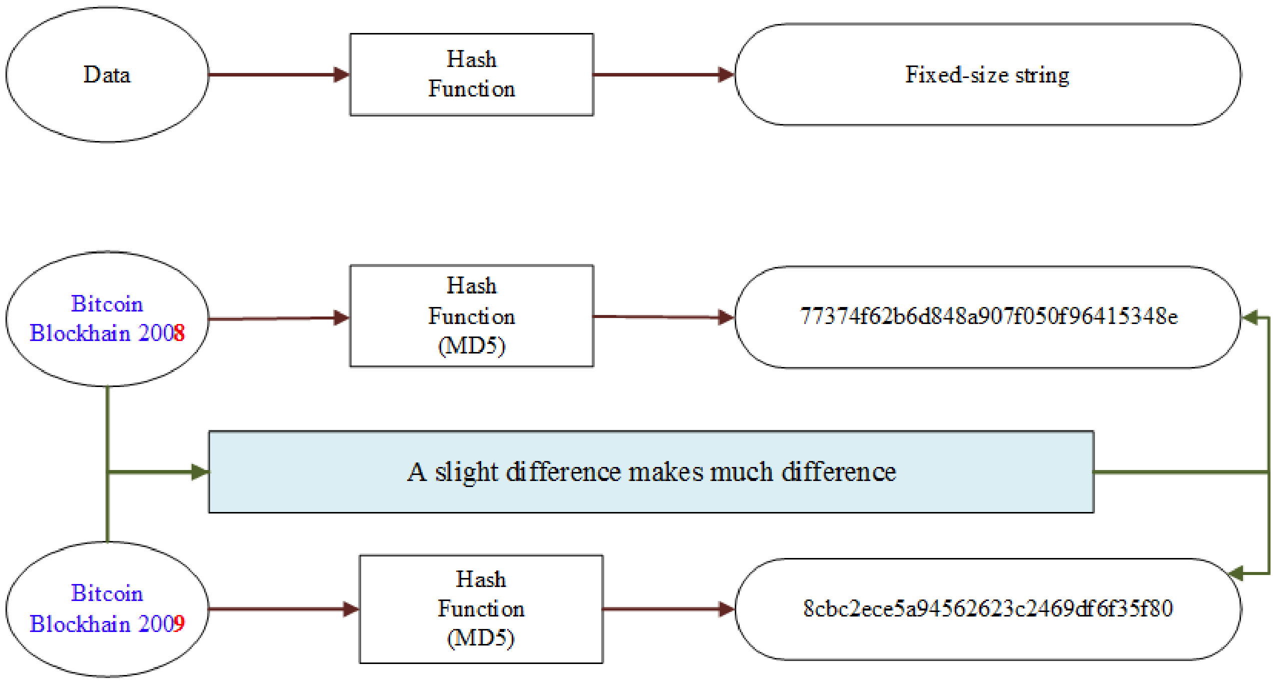


Figure 3-5: Hash mechanism

3.2.4. Public-key cryptography and digital signature

One of the most important parts of a blockchain system is the keys (public and private) of ownership to transfer the digital currency.

Public-key cryptography (asymmetric cryptography) is a cryptographic methodology that uses key pairs for encryption and decryption. A public key is publicly available to the world, and a private key is known only to the owner. In the public-key cryptography, anyone can encrypt and send a message using the public key of the message receiver, and the message can be decrypted by only the receiver's private key [35].

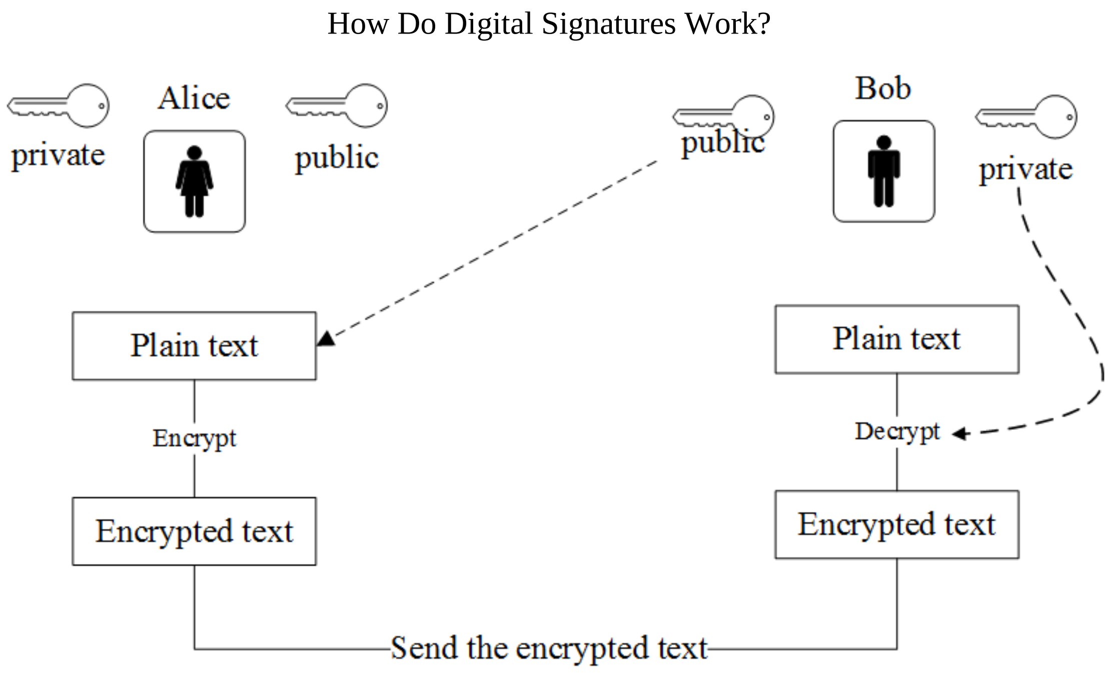


Figure 3-6: Digital signature

A digital signature is a mechanism to check the authenticity of digital messages and documents using public-key cryptography. The digital signature provides the following abilities:

- authentication: the message receiver can check the message sent by the sender
- non-repudiation: the sender cannot deny that the message was sent from herself or himself
- integrity: the message was not falsified or changed during communication

Public-key cryptography and digital signature are used for validating a creator of the bitcoin transaction as well as validating addresses of the bitcoin wallet in the bitcoin system [36].

3.2.5. Peer-to-Peer (P2P)

A Peer-to-Peer network (P2P) is a collection of various distributed resources that are connected by a network. From the application point of view, P2P is the opposite of Client/Server architecture. A distributed network architecture might be called a P2P network when the participants share a portion of their resources (processors, storage, network capacity, etc.). These resources should provide the service and content, which are accessible by other peers directly without going through intermediary entities, by the network [37]. Bitcoin uses the P2P network that has contributed and has developed a complete distributed network to overcome the single point of failure. Some of the most successful applications of P2P technologies are file sharing platforms such as Napster and BitTorrent.

3.2.6. Timestamp Server

Bitcoin proposed that a timestamp server takes a hash of a block of transactions to be timestamped and publishes the hash broadly. The timestamp proves the connection sequence of transactions, and each timestamp is included in the previous timestamp in its hash [6].

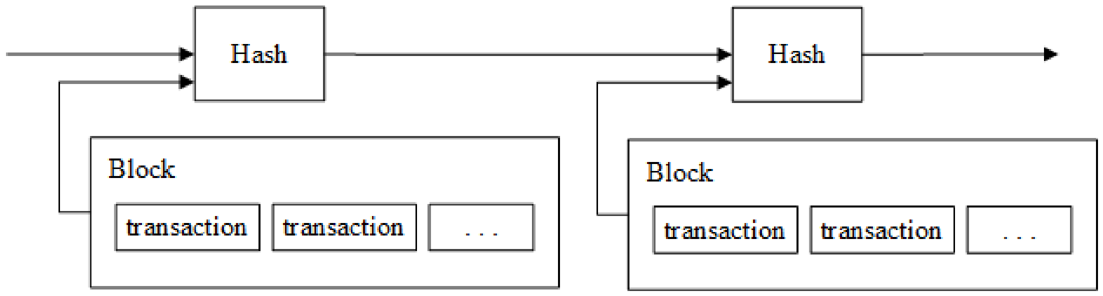


Figure 3-7: Timestamp Server

The timestamp server is designed to prove that the data must have existed at the time.

3.2.7. Merkle trees

Each block of bitcoin includes the summary of all transactions in the block using merkle trees. A merkle tree is a data structure to summarize and verify the integrity of large sets of data. A merkle tree is similar to a binary tree, as each node will have a maximum of two child nodes [38]. In Bitcoin, Merkle trees are used to summarize all transactions in a block and to provide an efficient process to validate whether a transaction is included in the block or not. To create a Merkle tree, the hash function runs on a pair of nodes recursively until there is only one hash known as Merkle root. The cryptographic hash algorithm of Bitcoin's Merkle tree applies SHA256 twice, and the Merkle tree is constructed bottom-to-up. Transactions will not be stored in the Merkle tree; rather, their data will be hashed, and the result of hash will be in each child node. A Merkle tree requires an even number of child nodes because it is a binary tree. Therefore, if there is an odd number of transactions, the last transaction will be duplicated to create an even number of child nodes [39]. Merkle tree gives a possibility to discard unnecessary transactions without breaking a block’s hash because the interior hashes are not stored.

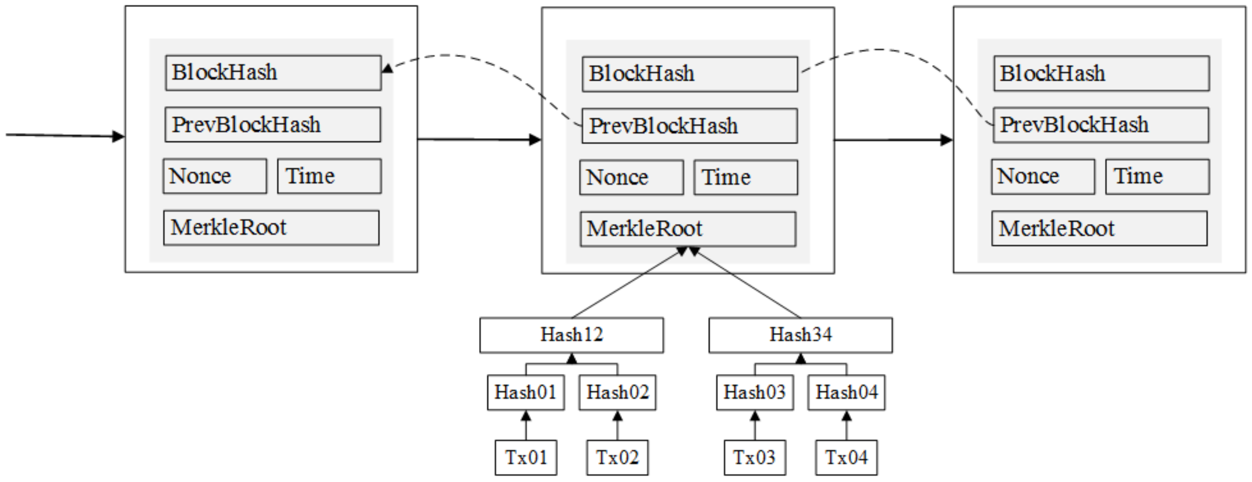


Figure 3-8: Merkle tree

Using Merkle trees, **simplified payment verification (SPV)** is possible to validate payment without running a whole network node. A user only needs to store a copy of the block headers of the longest PoW chain. The user will not be able to check a transaction by herself or himself, but the user can prove that the network accepted it, and further blocks have already been confirmed by the network. This verification process can be reliable as long as the network is under honest nodes' control.

The blockchain network is a peer-to-peer network which operates on cryptographic protocol and consists of a network of nodes. The network nodes handle communications and verify transactions, and the nodes also compete for profit (transaction fee) by generating a valid block through a process of receiving transactions and solving a resource incentive task (PoW). The bitcoin network runs the following steps:

1. Digitally signed new transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. Each node computes a difficult proof-of-work for its block
4. When a node finds a proof-of-work, it broadcasts the block to all nodes
5. Nodes accept the block only if all transactions in it are valid and not already spent
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash [6]

From the perspective of the node, the longest chain is the correct one. When nodes broadcast a different version of the next block at the same time, the other nodes may receive different blocks.

In this case, the nodes will work on the received one, but they will save the longest one. When the next valid PoW is found, and it becomes the longer chain, the node will switch to the longer one.

3.2.8. Proof of work

A proof of work is a piece of data that is hard to generate, but it is easy to verify and check certain requirements [40]. Generating the PoW is a random process with low chances, and the process goes through a lot of trial and error until it becomes a valid proof-of-work. The PoW normally refers to a mechanism to confirm that a person does certain work (to discourage to act wrongly). In this chapter, we will describe the proof of work in bitcoin. Bitcoin introduced a digital currency that can eliminate a central bank. The question is how Bitcoin can achieve this goal. Bitcoin proposed an efficient solution that everyone can be a bank, so every participant keeps a copy of all transactions similar to central banks. A distributed ledger can store all transactions and ownerships. Bitcoin uses blockchain technology as a distributed ledger. However, distributed copies of blockchain data can create new chances for the users to cheat. For instance, Alice can send two different transaction to her two friends (Bob and Charlie) without having sufficient coins. This problem is called *double spending*. If Alice's friends validate and accept the transactions separately, it can lead to an inconsistent state of the block chain. If Charlie accepts the transaction and informs everybody in the network before Bob accepts his transaction, Bob would be able to figure the double spending out. So, a synchronized distributed ledger is viable when there is a synchronized and a not-jammable broadcast channel. Therefore, we should deal with the undesirable period of time between issuing a transaction and having every participant be informed about the transaction to solve a distributed consensus problem. Bitcoin solves the double spending problem by requiring the entire network to validate the legitimacy of the transactions so that other participants can notice the double spending problem. If the majority of participants confirms that the transaction is valid, the receiver should accept it. Otherwise, the transaction will be rejected. However, the majority of participants are also questionable when there are many false identities. This kind of attack is called a Sybil attack in the computer community [41]. For example, Alice could configure many instances that constitute the majority, which all confirm the transaction, even it is a double spend. In this case, Charlie would accept the transaction. Bitcoin uses the proof of work to prevent Sybil attacks. When the node validates transactions and broadcasts them, it needs to perform some proof of work to prove that it is a real identity. Therefore, the identification of the participant depends on the computational power, and not the number of identities, which can be fake [42]. In Bitcoin, the difficulty of PoW's puzzle relies on the target value. If the target requires the more zeros, the puzzle becomes more difficult to solve. The PoW of Bitcoin accomplishes the following tasks:

1. To allow everyone with the computational resource to participate in the competition of creating new blocks
2. To distribute bitcoins among network participants using fair distribution mechanism
3. To makes blocks difficult to tamper with since any small change to a block will require a new proof of work
4. To allow for a consensus mechanism, which is resistant to Sybil attacks

3.2.9. Blockchain as a distributed database

A blockchain is essentially a distributed and decentralized database of records or public ledger that contains chained blocks of transactions. Although Bitcoin is the most popular example of blockchain technology, it has been abstracted to refer any distributed database that records in continuously hash chained blocks. Consensus mechanisms verify every single transaction in the public ledger. Once the transaction is stored in the chain and broadcasted to the network, it can never be altered or deleted [6], [43]. Since the peer-to-peer networks do not depend on a central entity, blockchains have no single point of failure. If the network nodes are sufficiently distributed across different authorities and geographical locations, the blockchain becomes resilient to network outages by natural disasters, physical or cyber-attacks, and falsifications from authorities. History shows that governments, agencies, and enterprises do not always act in a way which the public expect and accept as ethical. This requires the revolution of existing database technologies that can store data that should be impossible to remove, modify or control by a centralized authority.

Another advantage of the blockchain technology is a built-in feature of digital assets to handle tradable assets. The blockchain is designed to use a UTXO mechanism as originally conceptualized by Bitcoin. This idea refers to the face that every new transaction input requires a referral to the output of another transaction. This proves that the asset referenced inside the transaction is traceable up to its creation. This feature of the blockchain is the most important because it will bring transparency, auditability, and traceability.

Immutability: Blockchain is append-only database technology. The principle of blockchain technology is that once the data is recorded into the blockchain, it cannot be manipulated. In other words, the blockchain is an immutable data storage method. The hash chain differentiates the blockchain technology from any other distributed technologies. Hashing chain the blocks together.

Transparency: It is one of the important aspects of the blockchain. Within the blockchain network, anyone can join, participate, and see the transparent record of transactions, and it is possible to track and record every single movement of digital assets using the decentralized ledger. Tracking the movements of digital assets using the blockchain technology enables the capability to verify an asset's provenance and authenticity directly. Every node can have a complete and constantly updated copy of the ledger, and it allows them to be used for the purpose of monitoring.

Decentralization: One of the basic goals of the web is decentralization and freedom [44]. During the last decade, the massive growth of the web led to a centralization issue. A few large companies own a huge amount of data on the internet, so the lack of transparency and control from these companies reveals negative aspects of centralization such as manipulation, surveillance, and frequent data breaches [45]–[47]. Fortunately, blockchain technology promises a new feature. Modern applications can be built with a decentralized architecture; no single party has absolute power and control. Equality is also the advantage of the blockchain technology. The blockchain is structured as a peer-to-peer network on top of the internet. Peer-to-peer (P2P) refers to the fact, that computers participate in the network and act as peers to each other. The participants are all equal, and there are no special nodes. The network nodes are interconnected in a mesh network using flat topology. There are no centralized services, no servers, no hierarchical structure, and no central gateway within the network. P2P networks are fundamentally resilient, decentralized and open.

3.2.10. Public and private block chains

In general, blockchain is publicly available and requires a large computational resource to run the whole network; as a result, this limits its potential use cases. In some cases, data is susceptible to manipulation, confidential, or private, and the blockchain cannot store them securely. The private data would be distributed through every node on the network and completely exposed. A centralized architecture can store secure data with less transparency. This is a contradiction between the centralized databases and decentralized public blockchains [48]. Blockchains can be classified based on the access of blockchain data in the following way:

Public blockchain: A blockchain grants the permissions to issue transactions and to read access for all data of the chain to all users.

Private blockchain: A blockchain is a blockchain that is available for a predefined list of entities to access the blockchain data and create a transaction on the chain.
Permissionless blockchain: A permissionless blockchain is a type of blockchain which does not restrict identities of the transactions.
Permissioned blockchain: A permissioned blockchain accepts only transactions that are performed by predefined entities with known identifications [49], [50].

Private and permissioned blockchains are not necessarily the same. For example, a permissioned blockchain can be available to the public to read the data of chain, but the public might not have permission to create transactions. Also, private blockchains can be accessible for preconfigured entities, and the entities may have full permission to access the blockchain. A permissioned blockchain can build a system, which operates distributed ledgers or timestamped registries, using the basis of blockchain innovations. Although permissioned blockchains do not require to use of PoW, consensus protocol can be utilized as an additional level of security and to increase auditability.
The blockchain can have multi-level permissions including:

1. Connect to the network
2. Create asset
3. Send transaction
4. Receive transaction
5. Confirm transaction (“mine’)
6. Read transaction from the chain

Blockchain data storages provide a secure and decentralized framework. One of the most significant advantages of blockchain technology compared to other distributed ledgers is consistency, and security using cryptographical mechanisms, which eliminates human related issues.

3.2.11. Applications of blockchain technologies

After the success of Bitcoin, various blockchains have been proposed and implemented. The diversity of blockchain technologies could be classified in the following ways:

- Expansion of the terms of recording and exchange on transferring digital assets and digital rights management

- Improvement or new creation of consensus algorithms to reduce workloads
- Enhancement of reliability and limiting participants in the network to increase efficiency and speed up the performance of transaction processing

Expansion of blockchain technology

Since Bitcoin has been recognized by its potential and advantages, there are more than **600 digital coins in the market** by changing various parameters and cryptography mechanisms of Bitcoin [51]. Once the industry recognized that blockchains could handle transactions without a central authority, enterprises and organizations started to work on using them for **tracking the transfer of assets and services**, and the ideas are not limited by a virtual currency. There is work to apply the blockchain technology to the transfer assets such as real states and automobile companies. Another application of blockchain technology is to guarantee of the **authenticity of the ownership or rights**, and it is a case not only about the transactions. Other examples are preserving documents and casting votes, etc. **Smart contracts** are another attractive extension of blockchain technology. With smart contracts, participants can push new scripts or procedures to the blockchain. This allows the blockchain to execute automated process and triggers.

a) Improvement or new creation of consensus algorithms to reduce workloads

The improvement of consensus algorithms addressed the existing issues of the Bitcoin blockchain:

- PoW is shown for every ten minutes in Bitcoin, and the system, which requires a quick response, cannot utilize Bitcoin.
- The block size of Bitcoin is approximately 1 MB, and it cannot handle a large volume of transactions.
- Bitcoin involves enormous computing power

In order to solve these problems, there are new algorithms being developed in the blockchain community. For example, Proof of Stake (PoS) is designed to become a more competitive form of PoW to solve energy consumption issue of PoW. PoS is implemented in Ethereum, Bitshares, and NXT, etc. [52]. Also, Proof of Importance (PoI) and Practical Byzantine Fault Tolerance (PBFT) algorithms are already developed and adopted for the blockchains.

b) Enhancement of reliability and limitation of participants in the network to increase efficiency

A consortium or private type of blockchain enables an enhanced transaction processing mechanism by reducing PoW workload based on limited and reliable participants and simplifying the consensus algorithm, which does not need a reward to build a consensus.

Public

- The network is publicly available to the world. Anyone can participate in the network.
- A consensus mechanism is important because the network should be protected from malicious participants.

Consortium

- A consortium blockchain is a blockchain where the consensus managed by a pre-defined set of nodes.
- For example, there is 20 financial insititutions that each of them manages its own node, and at least 15 nodes sign every block to create a valid block in the network.

Private

- A fully private blockchain is a blockchain where write permissions are controlled by only one organization.
- Private blockchains are likely database management and auditing, etc within in the internal organization. In this case, the auditability is useful for the organization.

Chart 3-1: Blockchain types

They are already various services using blockchain technology in the following fields:

- **Finance:** Bitcoin and other virtual currencies can be a use case for blockchains. Various financial services including remittance, security of the transaction, claims, and others are applying the blockchain technology. Ripple is one these applications.
- **Reward and loyalty points:** Blockchains can enhance reward and loyalty points. GyftBlock is a platform based on blockchain technology, and it enables end-to-end capabilities of gift cards.
- **Communication:** Getgems server is a social networking services (SNS) that grants virtual currency (GMZ tokens) to SNS's users when the users browse the advertisements. Users can exchange digital tokens within the SNS.
- **Asset Management:** Ownership and transfer of assets and properties are important events, so they should be immutable; therefore, blockchain technology can be useful in this area including land registration and property registration. Factom is a platform, which promises honesty, trust, and immutability based on blockchain technology.

- **Storage:** Storj provides a distributed file storage service using blockchain technology. Data are encrypted and stored in the P2P network.
- **Authentication:** Platforms handle the authentication of validity of arts, drugs, digital contents. Ascribe provides a copyright solution for creative people using blockchain technology.
- **Sharing:** Blockchain technology can contribute to managing the rights of goods and services in the sharing-economy. La`Zooz is a decentralized transportation platform by the community using blockchain, and it aims to provide a smart solution for utilization of available vehicles' space.
- **Distribution management:** Tracing capability of blockchain technology can be useful for distribution management by recording all histories of processing from beginning to end product or service. Everledger is a digital ledger that provides provenance of valuable assets and creates transparency using blockchain technology.
- **Content:** Content delivery can be another use case of blockchain technology. Streamium is a decentralized broadcasting solution using a blockchain, and the broadcasters earn bitcoins based on their viewers.
- **Prediction:** Augur is a decentralized platform for the prediction market where participants can vote on different events to predict the future, and it uses blockchain technology [53].

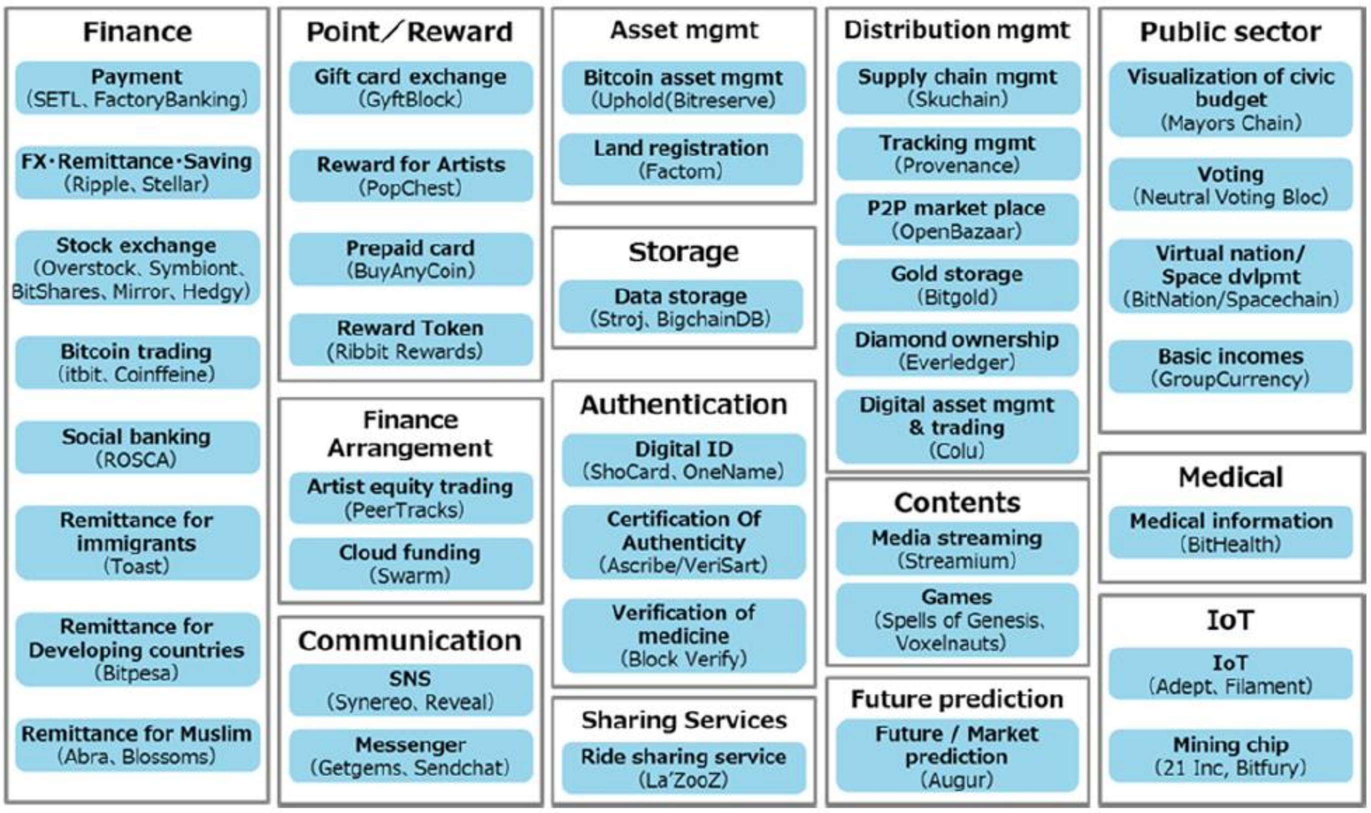

Finance
Payment (SETL、FactoryBanking)
FX・Remittance・Saving (Ripple、Stellar)
Stock exchange (Overstock、Symbiont、BitShares、Mirror、Hedgy)
Bitcoin trading (itbit、Coinffeine)
Social banking (ROSCA)
Remittance for immigrants (Toast)
Remittance for Developing countries (Bitpesa)
Remittance for Muslim (Abra、Blossoms)
Point/Reward
Gift card exchange (GyftBlock)
Reward for Artists (PopChest)
Prepaid card (BuyAnyCoin)
Reward Token (Ribbit Rewards)
Finance Arrangement
Artist equity trading (PeerTracks)
Cloud funding (Swarm)
Communication
SNS (Synereo、Reveal)
Messenger (Getgems、Sendchat)
Asset mgmt
Bitcoin asset mgmt (Uphold(Bitreserve)
Land registration (Factom)
Storage
Data storage (Stroj、BigchainDB)
Authentication
Digital ID (ShoCard、OneName)
Certification Of Authenticity (Ascribe/VeriSart)
Verification of medicine (Block Verify)
Sharing Services
Ride sharing service (La'ZooZ)
Distribution mgmt
Supply chain mgmt (Skuchain)
Tracking mgmt (Provenance)
P2P market place (OpenBazaar)
Gold storage (Bitgold)
Diamond ownership (Everledger)
Digital asset mgmt & trading (Colu)
Contents
Media streaming (Streamium)
Games (Spells of Genesis、Voxelnauts)
Future prediction
Future / Market prediction (Augur)
Public sector
Visualization of civic budget (Mayors Chain)
Voting (Neutral Voting Bloc)
Virtual nation/ Space dvlpmt (BitNation/Spacechain)
Basic incomes (GroupCurrency)
Medical
Medical information (BitHealth)
IoT
IoT (Adept、Filament)
Mining chip (21 Inc, Bitfury)

Figure 3-9: Blockchain use cases [53]

3.3. RESTful web services

In 2000, Roy Fielding introduced and defined Representational State Transfer (REST), which is an architectural style based on a set of principles that describe how network resources can be defined and addressed [54], [55]. REST uses a client-server architecture, and it does not require client-server communication to a specific protocol, but it is practically used with hypertext transfer protocol (HTTP) because HTTP is the primary protocol on the web [56]. The REST protocol simplifies the interactions between client and server using the following standard HTTP methods:

- HTTP GET – retrieve a resource
- HTTP POST – create a resource
- HTTP PUT – update a resource
- HTTP DELETE – delete a resource
- HTTP PATCH – update parts of resource without changing the entire resource
- HTTP HEAD – retrieves only HTTP header without body [57]

The formal constraints of REST architectural style are Uniform contract, Stateless, Cacheable, Client-Server, Layered System, Code-on-Demand.

- Uniform Interface: describe the uniform interface as such HTTP methods (GET, POST, PUT, and DELETE) between client and server
- Stateless: uses a stateless communication protocol, typically HTTP. All requests should contain all information required for a specific request. This interaction improves the following features:
  - Visibility: each request contains the full details of the request
  - Reliability: easy to recover from partial failures
  - Scalability: the server can work more efficiently without handling the states of clients on the server side
- Cacheable: enables the caching capability on the client side. Resources should be identified as cacheable or non-cacheable. Caching eliminates unnecessary interactions, and it improves efficiency, scalability, and performance.
- Client-server: the clients and server are separated from each other; therefore, the client is not concerned with the data storage, and the portability of the client code is improved while the server is not concerned with the clients' interference on the server side. The server is simpler and easy to scale
- Layered System: enables clients to connect an intermediary server rather than directly to the end server. Using intermediary servers improves the system scalability by enhancing load balancing and shared caching
- Code-on-demand: an optional constraint where the server temporarily extends the functionality of a client by the transfer of executable code.

3.4. Microservices

Microservices are a new paradigm for software architecture; they are characterized by breaking down one single large application to multiple autonomous services. The microservice is deployed as a single application, and it is a small independent service, which each microservice running its own process and communicating with lightweight remote calls such as HTTP-based RESTful web services or message queue [58]. Microservice-based applications provide the separation of the application from the underlying infrastructure. The microservice emphasizes that applications should be composed of small enough services to reflect independent concerns. Microservices are typically RESTful web services and communicate and share data with each other.

3.5. Summary

This chapter shows the traditional access control models proposed to protect the digital resources inside an organization, and the models use the centralized architecture [27]–[29]. The ABAC model is a more flexible and dynamic approach to manage the access permissions, but it is not directly designed to develop decentralized access control systems [30]. OAuth is an attractive approach for many mobile and web application because existing well-known social networks can provide the identification and some related information of their users. The problem of OAuth is that it is still centralized to few large social networks such as Facebook, Google, and LinkedIn. The OAuth's idea of transferring tokens is useful in decentralized access control systems because the tokens are self-descriptive and lightweight.

Blockchain is an emerging technology for distributed and decentralized architectures to share data across a large network of untrusted parties. Built-in features of blockchain technology including transparency, immutability, cryptography and operational resilience could potentially improve the access control systems' storage capability, and blockchain technology will be suitable for the decentralized architecture. Additionally, blockchain technology is a new approach to manage and store data for tracking the ownership of access rights. In this sense, it is possible to create a secure repository using blockchain technology.

CHAPTER 4

4. PROPOSED ARCHITECTURAL DESIGN

New technologies and the internet have offered new opportunities and revenue streams for organizations and enterprises, yet they come with new possibilities for cyber-attacks to exploit. Cyber-attacks have become increasingly targeted to steal valuable data such as intellectual property, personal information, health records, and financial information. Also, the attackers use ransomware techniques to monetize the data access or interrupt overall business processes through Distributed Denial of Service (DDoS) attacks. Deloitte's cyber risk professionals recommend that organizations need to follow a secure and resilient approach to manage cyber-attacks [59].

Blockchain technology could help enterprises to improve access control systems through consensus mechanisms, and underlying features of immutability, transparency, auditability, and cryptography. The previous chapters have already stated that the proposed architecture tries to answer "*how to create the access control system for across boundary organizations using the blockchain technology.*" The main purpose of the solution is to build an access control system, which uses a private blockchain as a data storage, across organizations. The purpose of a decentralized access control is a system of independent organizations connected to each other and managing the access rights of their own microservices. The private blockchain is a ledger, which tracks all data of the access control system. The application layer of access control will work with the immutable data in the blockchain. Each organization manages its own microservices and access control, which results in a decentralized access control system.

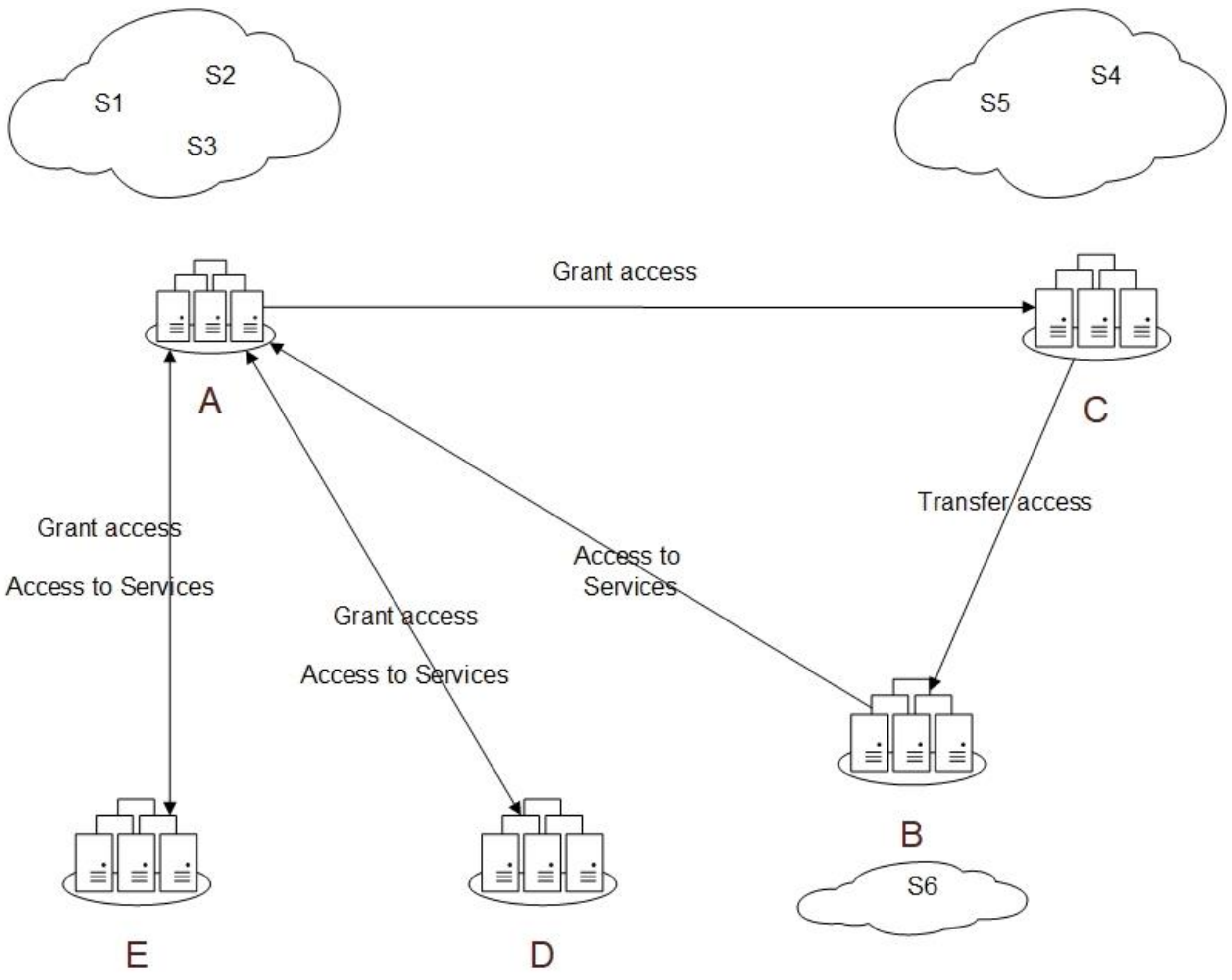


Figure 4-1: The general scenario between the Organizations

In the figure, A, B, C, D, E organizations are creating a collaborative environment. This Figure represents microservices in the cloud under each organization's provision. The organizations grant the access rights to other organizations for their own microservices in a decentralized environment.

Let us consider the scenario where Canadian universities look for a national collaborative platform using their high-performance computing research centres and infrastructures. We can assume that each university has its own high-computing resources, which could be accessible for other Canadian universities. For example, the following table shows the possible research centres, and I have chosen three examples to keep the scenario simple.

University name	Research Centre	Purpose
University of Saskatchewan	Imaging Research Centre	To analyze images using high-computing machines
University of Toronto	Artificial Intelligence (AI) Research Centre	To promote and maintain Canadian excellence in deep learning and machine learning more broadly
Queen’s University	Centre for Advanced Computing	To support the use of advanced computing and support for the research community

Table 4-1: Examples of the University Research Centre

Each university could expose microservices to make their computational resources available to others, and their microservices have different purposes and functionalities. When the research groups, graduate students, and faculty members try to collaborate and enhance the powerful computational resources between each other, the access control management becomes a problem. The access rights could be defined as subscriptions for each microservice, and the resource owner university can define how many times the consumer university can access the microservices within the period. One of the important features of this system is for the consumer university to transfer the access rights to internal departments, research groups, professors, and graduate students. This transferring process can continue from the consumer university to the department, the department to the supervisors, and supervisors to the graduate students. For example, the University of Toronto’s AI Research Centre grants the access rights to execute the high-computational AI tasks only 10000 times per month to the University of Saskatchewan (U of S). Then, the U of S will transfer these access permissions to the Department of Computer Science and the Department of Electrical and Computer Engineering. The Department of Computer Science could then transfer part of the access rights to professor Bob and Bob's graduate students to run his experiments. The figure 4-2 shows the possible access rights' transfers between the universities and their internal members. In the figure 4-2, AI-1, AI-2, AI-2, Q4, Q5, S4, and S5 represent microservices, and the colours distinguish the organizations. For instance, Green is for the U of S, yellow for the University of Toronto, and blue for the Queen's University. The consumers can access the microservices.

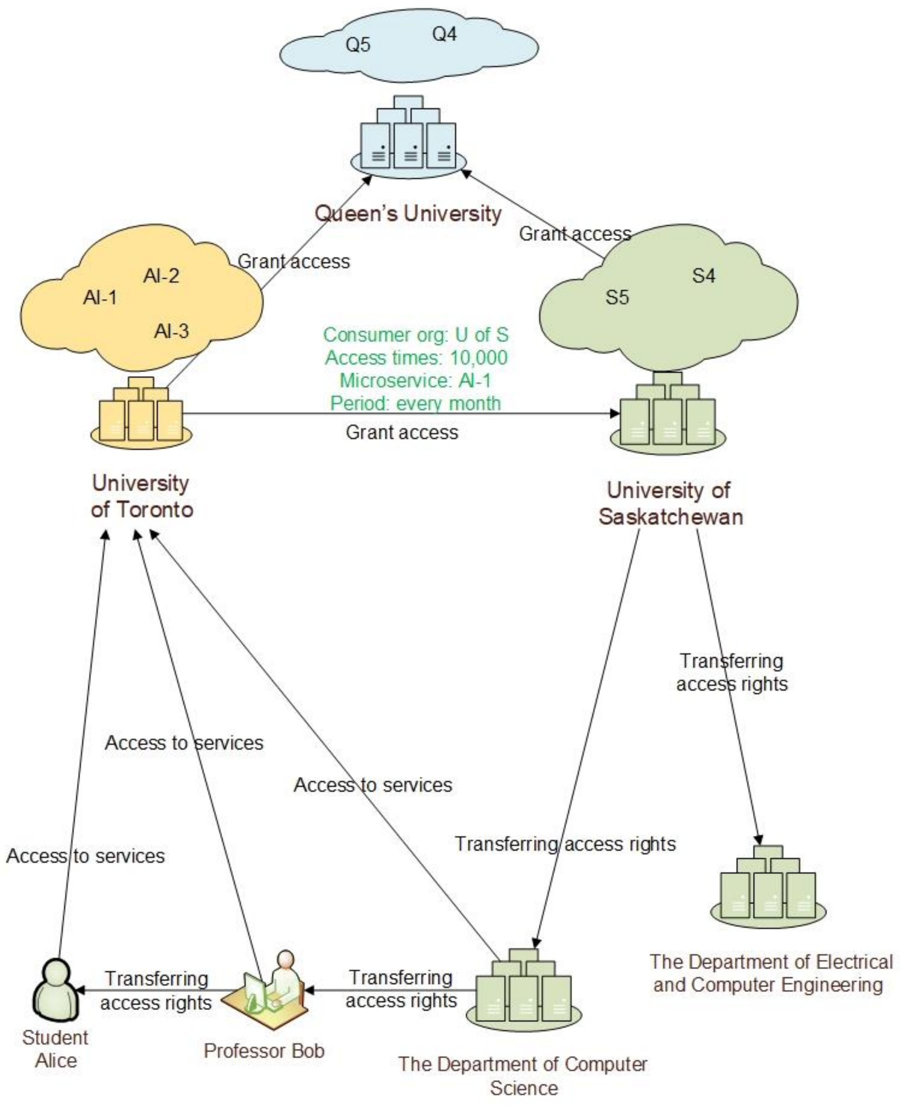


Figure 4-2: Access rights and transfers

During this collaboration, the access control systems should provide the following features:

- Define the microservices on the network and make the services discoverable
- Grant access to the consumer university
- Transfer access rights
- Trace the access rights' transfers

4.1. Architecture

Figure 4-3 shows the architectural overview between two organizations to demonstrate the components and their connections. Four organizations are in this figure to demonstrate the architecture is not limited to any number of organizations. Technically, this principle will support any number of organizations that can connect and communicate with each other using decentralized access control. Every organization will have its own microservices, local storage, and blockchain node in the blockchain network. The communication between organization is based on the blockchain network.

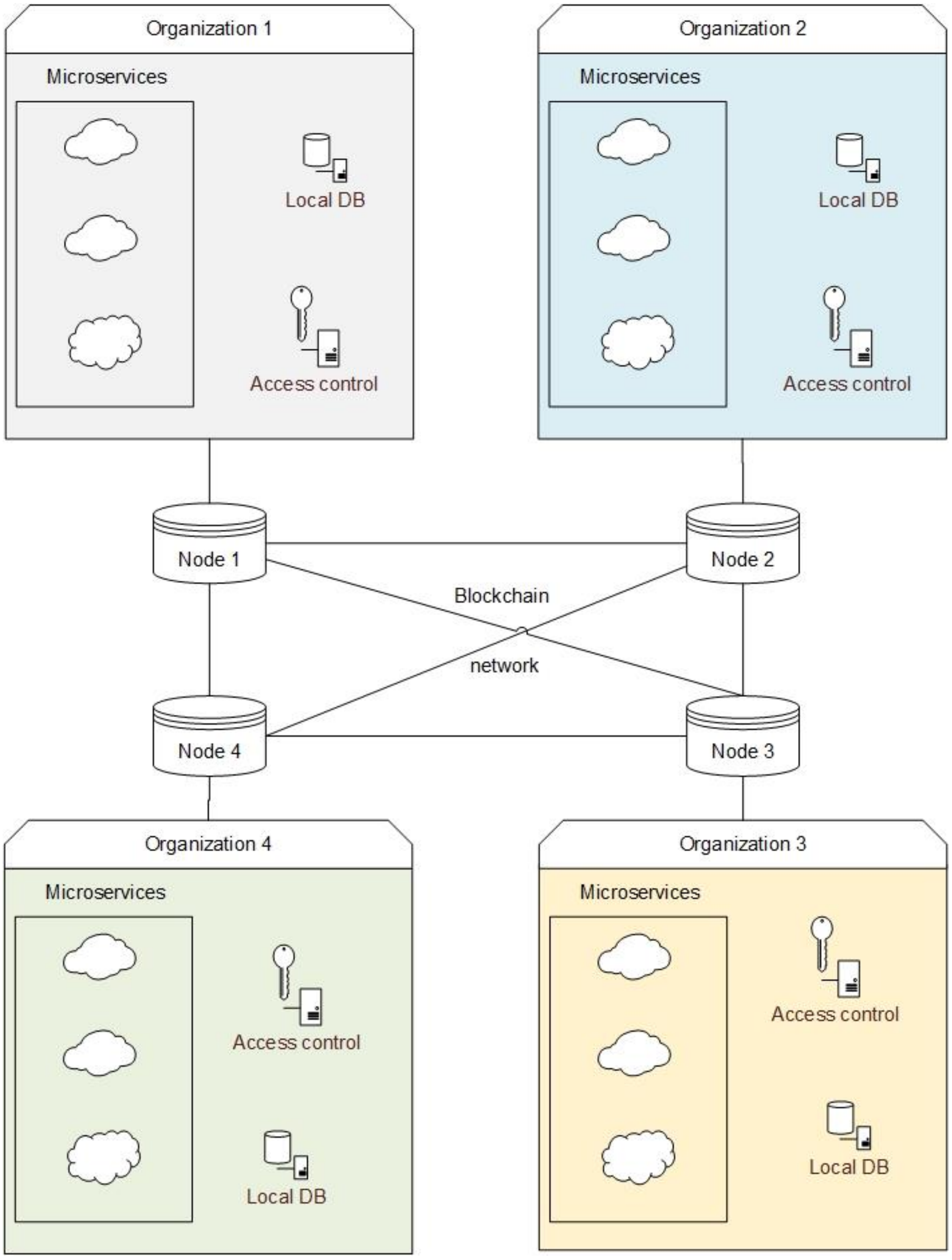


Figure 4-3: The overview of the architecture

Access control system

The proposed access control is decentralized, which supports independently work in each organization, and it is connected to each organization using a blockchain cluster in a peer-to-peer network. The system uses the blockchain as a shared-data storage.

4.2. System components

Figure 4-3 shows a proposed system consisting of the access control application, local database, microservices, and blockchain cluster. The detailed descriptions are follows:

i. *Access control application:* Each organization will have its own access control application, which decides whether access is allowed or disallowed. This application stores data in the local database as well as on the blockchain. The application is developed as RESTful web services, so it is easy to integrate with the client applications. The following features are implemented in this application:
   a. Defining the microservices
   b. Granting access right for the microservices
   c. Transferring access right(s)
   d. Revoking access rights(s)
   e. Authorizing access rights for the microservices
   f. Invoking the microservices

ii. *Local database:* The proposed solution uses a local database because the blockchain cluster will distribute all data to other nodes. The access control system should respond as quickly as possible to the clients, so using a blockchain for all purposes will reduce performance. The local database has two main purposes; keeping data secure and increasing the performance of the access control system.

iii. *Microservices:* This architecture assumes that organizations will have their own microservices that are already built and ready to use, and the access control system registers the microservices to make them discoverable to other consumer organizations. When the microservices are, available and granted to consume for the consumer organization, the resource owner's access control invokes it automatically.

iv. *Blockchain cluster:* Private and permission-based blockchain keeps the shared data and historical records of the access control system. Each organization will manage its own blockchain node on the blockchain cluster. Using the blockchain as data storage makes this solution more robust, resilient, immutable, transparent, and traceable. The traditional database systems supply Create, Read, Update, and Delete (CRUD) interfaces. The blockchain is an append-only data storage, so it does not support update and delete actions, but rather supports the creation of new transactions on the blockchain. The proposed solution follows this append-only principle.

User and key management

Every organization manages and certifies local users with its own authentication and authorization server, and the organization stores the users' public key on the blockchain nodes. The organizations manage their own public/private key pairs securely. The users are uniquely identified by their email address in this architecture. Organization administrators and internal access control system have the flexibility to use their own identification mechanism to identify and authorize locally. However, to achieve authentication and authorization across organization boundaries, they must have a mechanism by which all collaborative organizations agree on the PKI algorithms and definitions of it. The following figure shows how the organizations manage their users independently and exchange the public of the users by the blockchain network.

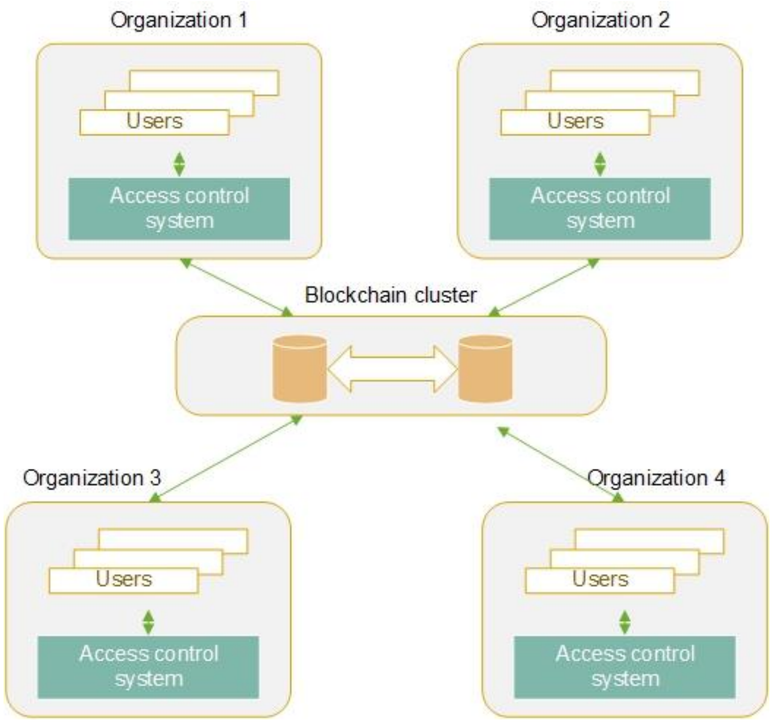


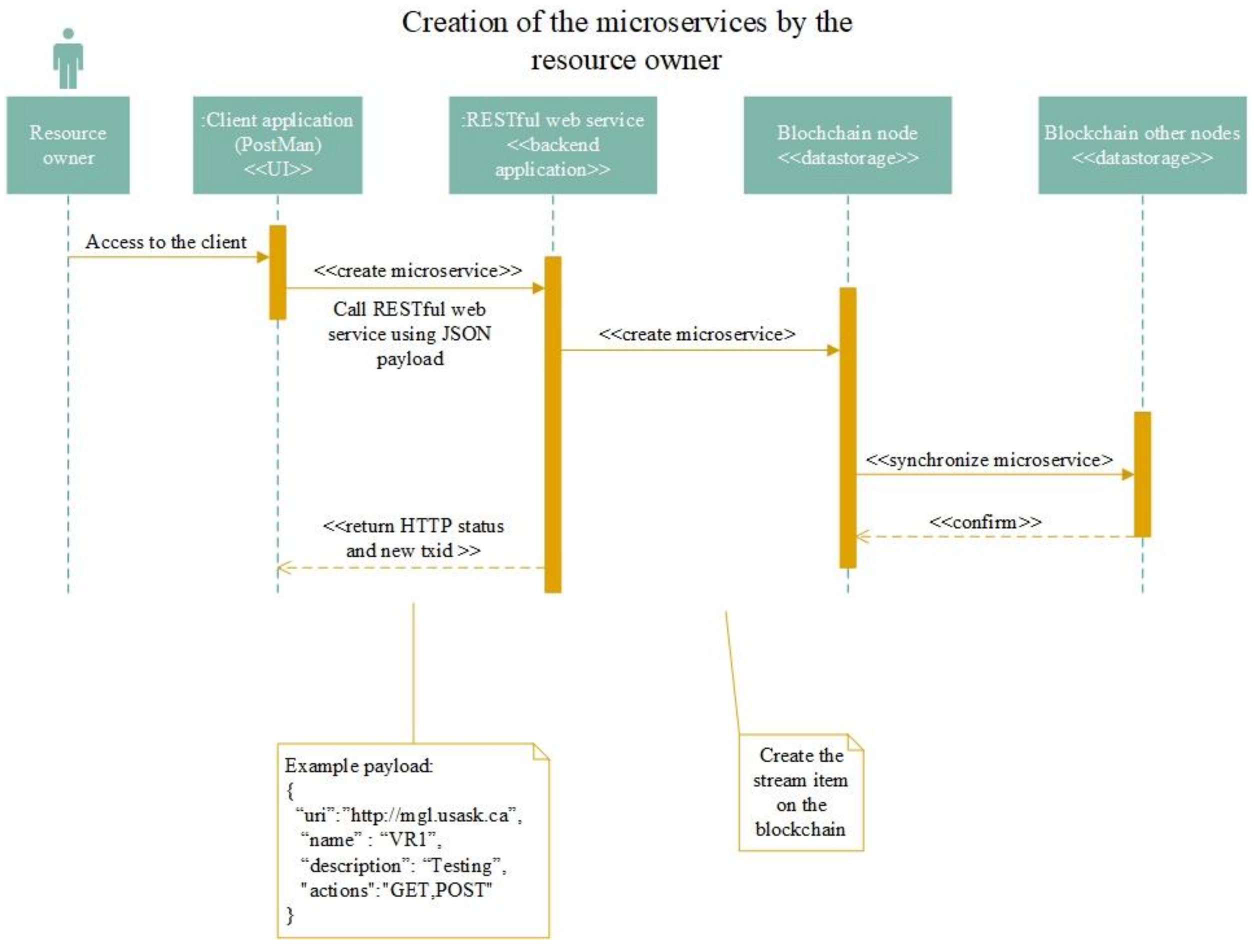
Figure 4-4: User and key management

4.3. System workflows

Defining microservices

In this architecture, microservices are digital resources, which are defined as RESTful web services. This work assumes that each organization will provide their own microservices for their digital artifacts. When the organization defines the microservices, the definition will be stored on the blockchain and distributed to the other blockchain nodes. The definition of microservices has the following attributes:

a) URI - uniform resource identifier that is reachable from the access control of its organization
b) name – name of the microservice
c) actions – the possible HTTP methods to call from the clients
d) description – additional description for the microservices


Creation of the microservices by the resource owner
Resource owner
:Client application (PostMan) <<UI>>
:RESTful web service <<backend application>>
Blochchain node <<datastorage>>
Blockchain other nodes <<datastorage>>
Access to the client
<<create microservice>>
Call RESTful web service using JSON payload
<<create microservice>
<<synchronize microservice>
<<confirm>>
<<return HTTP status and new txid >>
Create the stream item on the blockchain

```
Example payload:
{
  "uri":"http://mgl.usask.ca",
  "name" : "VR1",
  "description": "Testing",
  "actions":"GET,POST"
}
```

Figure 4-5: Definition workflow of the microservice

Figure 4.4 shows the workflow of defining microservices on the access control system. The RESTful web API stores the definition on the blockchain node, which the organization configures and manages, and the blockchain network will distribute the definition to other nodes.

Granting and transferring access rights

The proposed solution creates decentralized access control using the blockchain cluster, and granting and transferring works as sequentially as follows:

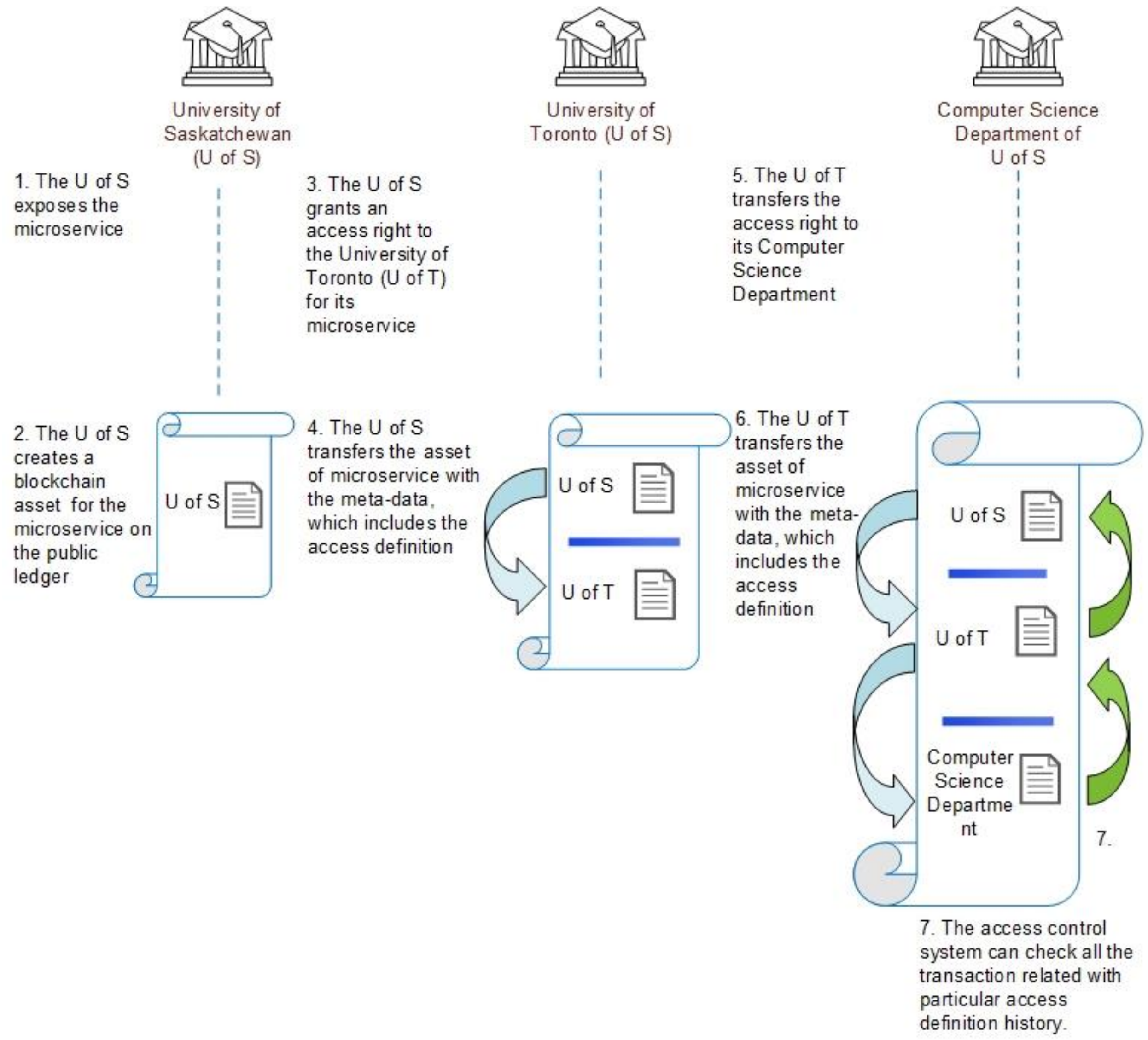

University of Saskatchewan (U of S)
University of Toronto (U of S)
Computer Science Department of U of S
1. The U of S exposes the microservice
3. The U of S grants an access right to the University of Toronto (U of T) for its microservice
5. The U of T transfers the access right to its Computer Science Department
2. The U of S creates a blockchain asset for the microservice on the public ledger
U of S
4. The U of S transfers the asset of microservice with the meta-data, which includes the access definition
U of S
U of T
6. The U of T transfers the asset of microservice with the meta-data, which includes the access definition
U of S
U of T
Computer Science Department
7.
7. The access control system can check all the transaction related with particular access definition history.

Figure 4-6: The workflow of the granting and transferring access right

Figure 4-6 shows the process of granting access, which can be divided into two following main actions.

1) The resource owner grants access rights to the service. The consumer organization:
    a. Retrieves the consumer organization's public key from the blockchain node
    b. Generates a unique token from the access definition
    c. Encrypts the token by the public key
    d. Stores the access definition in the local storage
    e. Stores the access definition on the blockchain
2) The blockchain nodes synchronize automatically

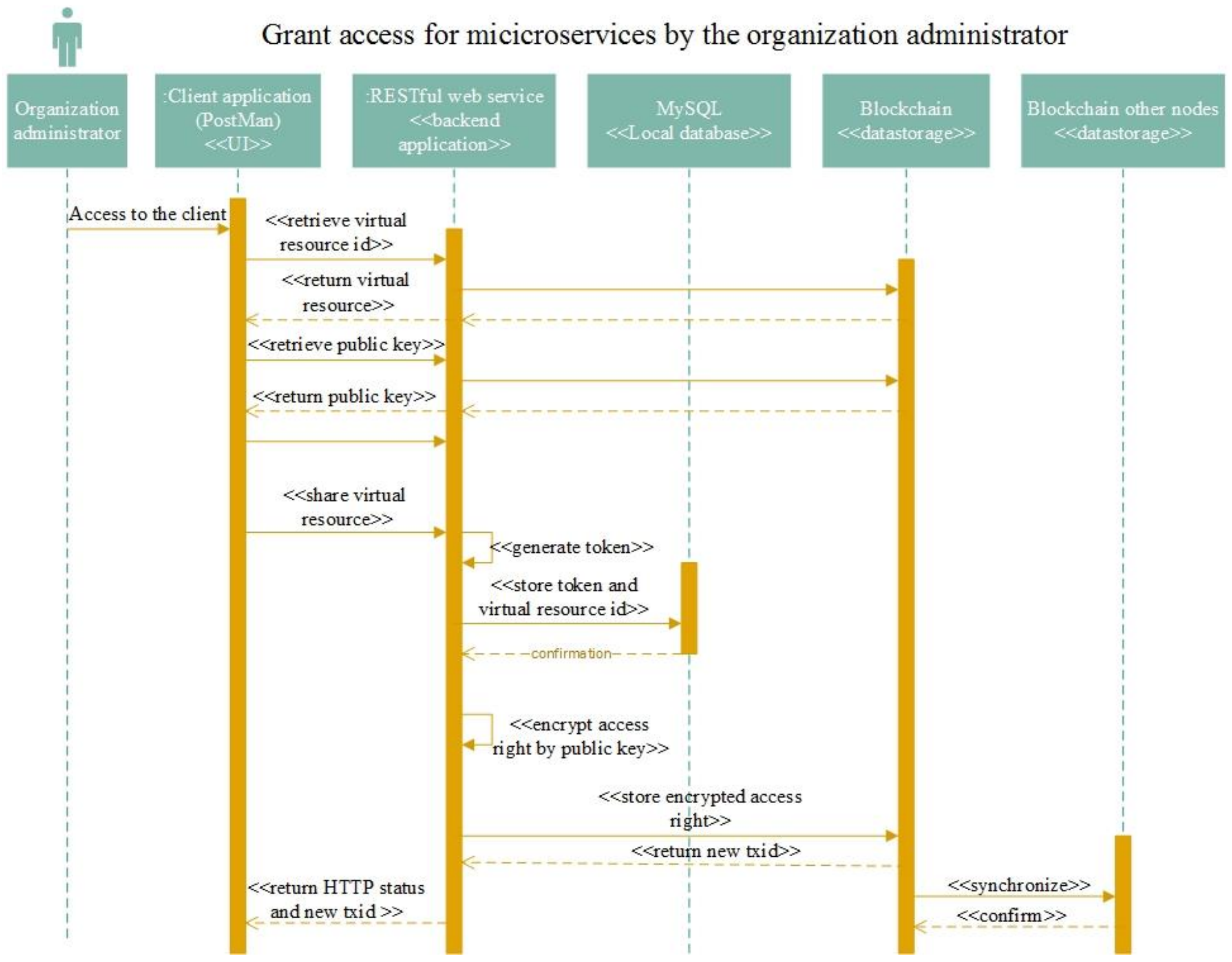

Grant access for micicroservices by the organization administrator
Organization administrator
:Client application (PostMan) <<UI>>
:RESTful web service <<backend application>>
MySQL <<Local database>>
Blockchain <<datastorage>>
Blockchain other nodes <<datastorage>>
Access to the client
<<retrieve virtual resource id>>
<<return virtual resource>>
<<retrieve public key>>
<<return public key>>
<<share virtual resource>>
<<generate token>>
<<store token and virtual resource id>>
confirmation
<<encrypt access right by public key>>
<<store encrypted access right>>
<<return new txid>>
<<return HTTP status and new txid >>
<<synchronize>>
<<confirm>>

Figure 4-7: Sequential diagram for granting access for the microservices

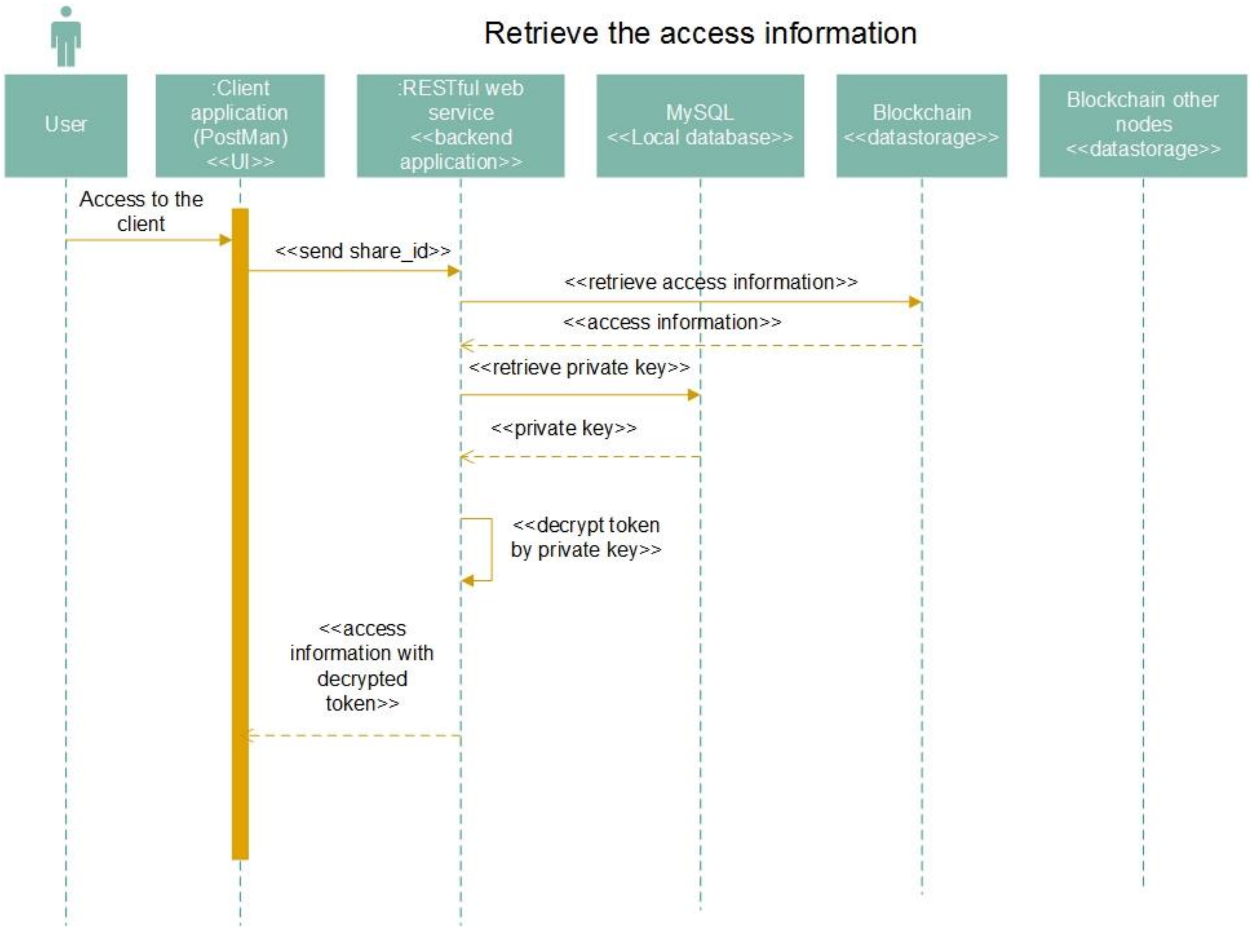

Retrieve the access information
User
:Client application (PostMan) <<UI>>
:RESTful web service <<backend application>>
MySQL <<Local database>>
Blockchain <<datastorage>>
Blockchain other nodes <<datastorage>>
Access to the client
<<send share_id>>
<<retrieve access information>>
<<access information>>
<<retrieve private key>>
<<private key>>
<<decrypt token by private key>>
<<access information with decrypted token>>

Figure 4-8: Retrieve the access definition

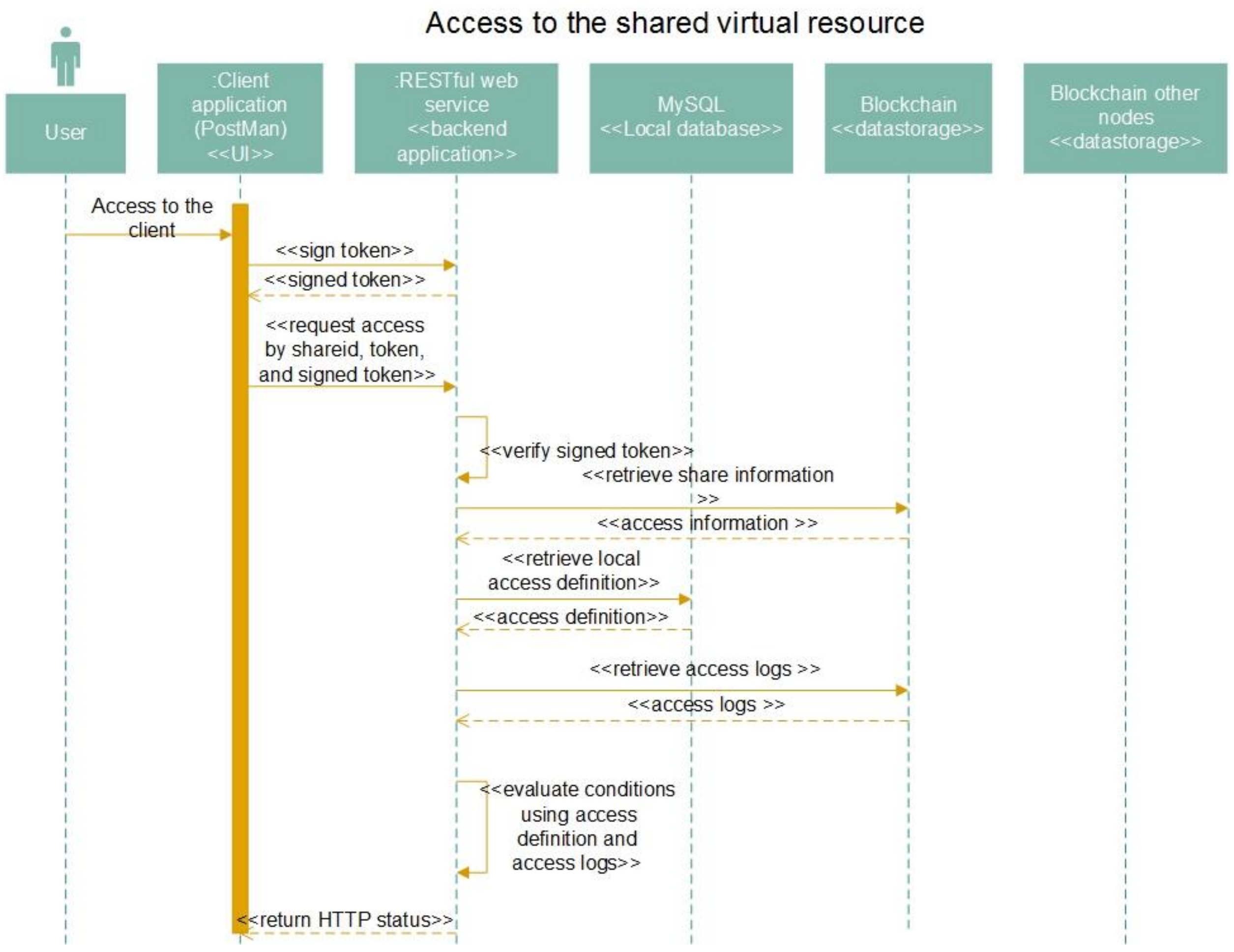
1) The system of the consumer organization receives the access definition from its own blockchain node, then:
    a. Retrieves the access definition from the blockchain
    b. Retrieves the corresponding private key from the local database
    c. Decrypts the encrypted token by the private key
2) The consumer organization sends a request to get an access token based on their access definition, then:
    a. Signs the token using the private key
    b. Sends the request using the access definition id, token, and signed token
    c. The service provider organization verifies the signed token by:
        i. Retrieving the access definition

ii. Retrieve the access logs

iii. Evaluating the conditions

3) The consumer organization accesses the microservices of the organization, which provides the service

Since the microservices are HTTP-based services, the access control system can invoke the microservices using the HTTP methods such as GET, POST, PUT and DELETE.



Access to the shared virtual resource
User
:Client application (PostMan) <<UI>>
:RESTful web service <<backend application>>
MySQL <<Local database>>
Blockchain <<datastorage>>
Blockchain other nodes <<datastorage>>
Access to the client
<<sign token>>
<<signed token>>
<<request access by shareid, token, and signed token>>
<<verify signed token>>
<<retrieve share information >>
<<access information >>
<<retrieve local access definition>>
<<access definition>>
<<retrieve access logs >>
<<access logs >>
<<evaluate conditions using access definition and access logs>>
<<return HTTP status>>

Figure 4-9: Access to the microservices

Fine-grained access control feature for the microservices

These types of access control systems facilitate granting differential access rights to users and allow flexibility in specifying the access rights of individual users. The figure 4-6 shows that the resource owner organization can grant access rights, and the receiver organization of the access rights can transfer the access rights to another organization. During this transfer, the access control system can support fine-grained access control. The microservices expose methods to manage entries with HTTP verbs: GET, POST, PUT, and DELETE. For example, U of T grants full-permission including GET, POST, PUT, DELETE actions for its A1 microservice to the U of S. Then, when the U of S transfers this access definition to its Computer Science Department, it can grant only GET, POST, DELETE actions. The access control system enforces and checks that the hierarchical structure is defined properly.

Access rights definition level	Access definition	Sender organization	Receiver organization
G1	microservice: A1, methods: (get, post, put, delete), times: 100000	U of T	U of S
G2	microservice: A1, methods: (get, post, put), times: 1000	U of S	The Computer Science Department of the U of S
…			
Gn			

Table 4-2: Example of fine-grained access control definition

4.4. Implementation

This architecture enables a decentralized access control system across organizational boundaries as presented in this chapter. Independent organizations can configure and create their own access control to share and access their virtual resources. This chapter describes the technologies, which are used in the implementation of the proposed architecture. The implementation utilizes Multichain blockchain, which is a platform for the creation and deployment of private and permission-based blockchains, as the data storage for the access control system [60]. To access Multichain's API, Multichain provides a multichain-cli command-line tool or JSON-RPC client. This work uses the multichain-cli command-line interfaces of the Multichain directly. The following figure shows the global architecture.

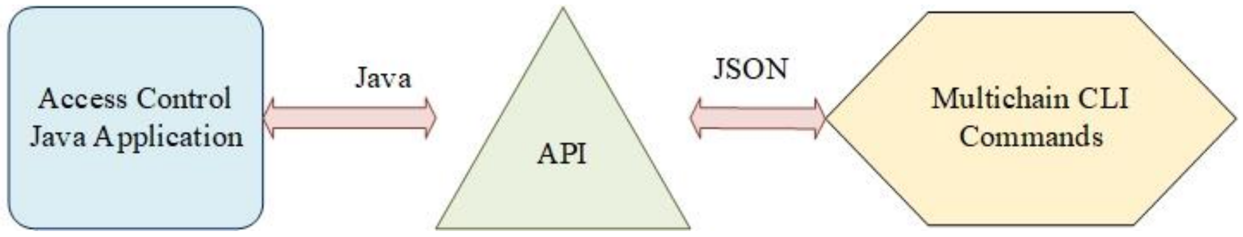


Figure 4-10: Technological stacks

This access control system is written in the Java programming language as RESTful web services. RESTful web services are simple because REST leverages existing well-known standards such as HTTP, XML, URI, MIME [61]. Since the access control system supports RESTful web services, client applications can integrate easily.

The blockchain transaction with attached metadata can provide the following features for data storage.

1) A time series database (ordering of entries)
2) An identity-driven database (entries are classified according to their author)

Data Structure

This implementation uses two types of data storage, which are blockchain and MySQL database. The access control system stores data on the following four types of Java Script Object Notation (JSON) documents on the blockchain:

Document type	Description
microservices	definition of the microservices
pkeys	users' information and public keys
access_definitions	the access definitions of the microservices
access_logs	access histories
access_revocations	access revocations

Table 4-3: JSON document types on the blockchain

The previous chapters already discussed the advantages and possibilities of the blockchain, but the confidentiality of the blockchain needs to be considered because the nature of the blockchain distributes all data to all other nodes. Therefore, storing the encrypted data and using local storage with each blockchain node are possible techniques to improve the security. For this reason, this implementation uses the local MySQL storage. The figure 5.1 shows the local MySQL database structure.

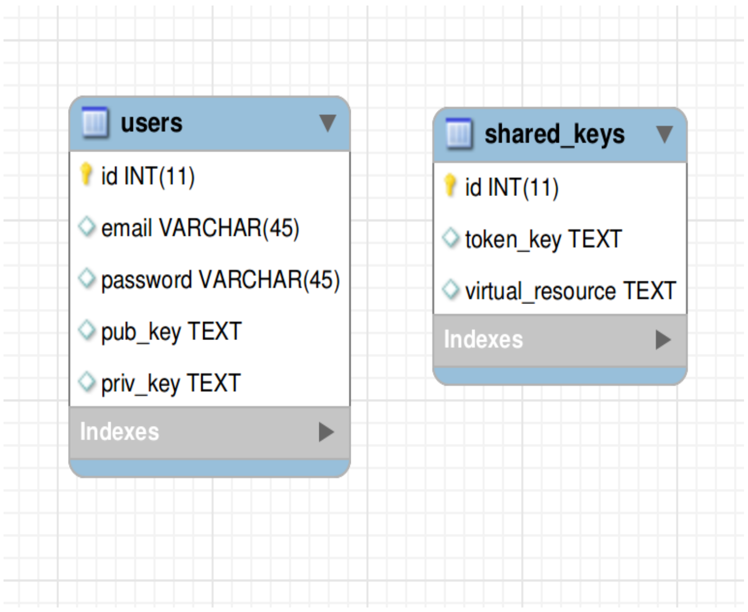


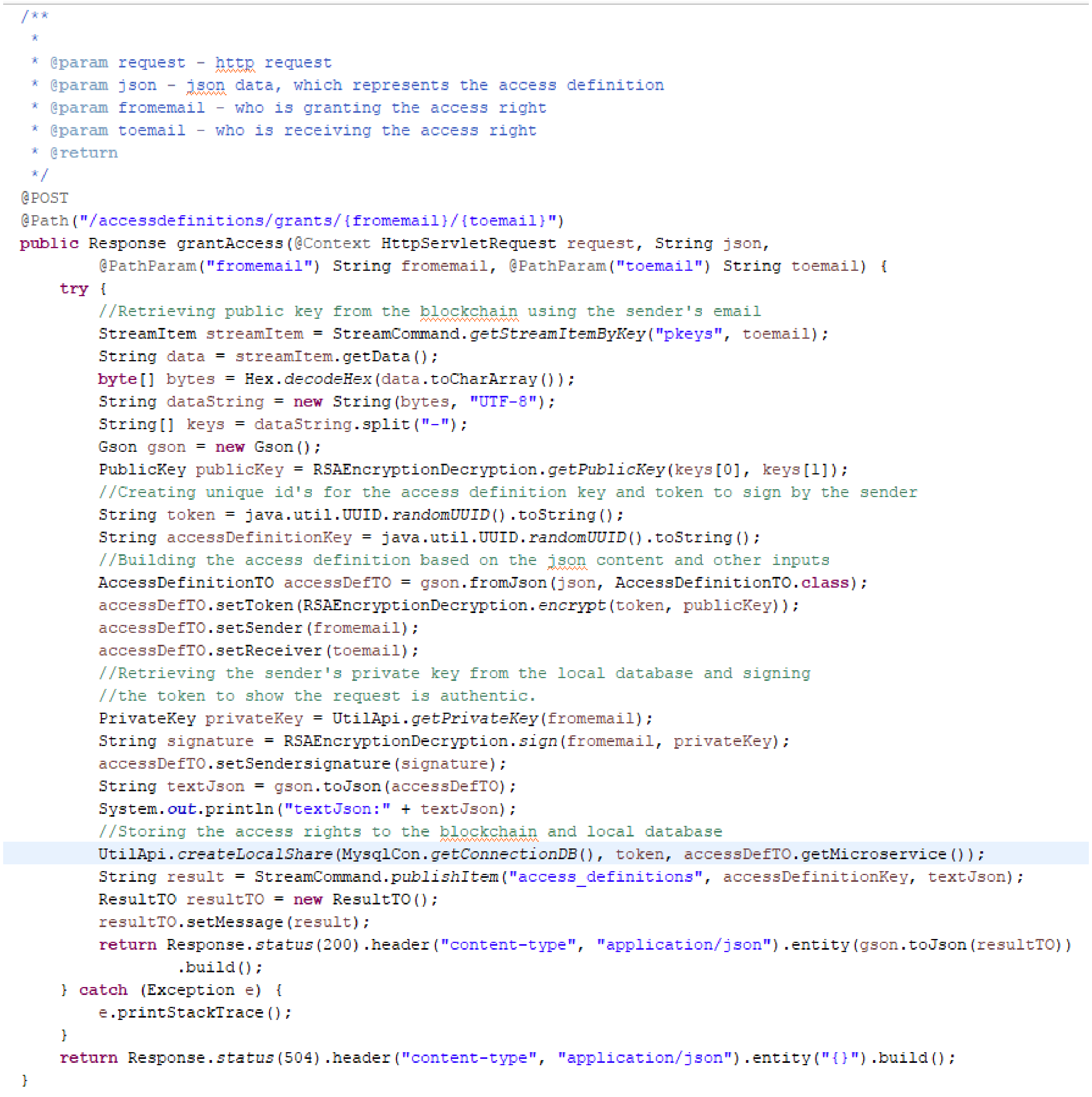
Figure 4-11: MySQL database tables

Code snippets for the core functions

The implementation creates and exposes the RESTful web services using the Java programming language. This section shows the code snippets from the functions of the access control system.

Granting access rights

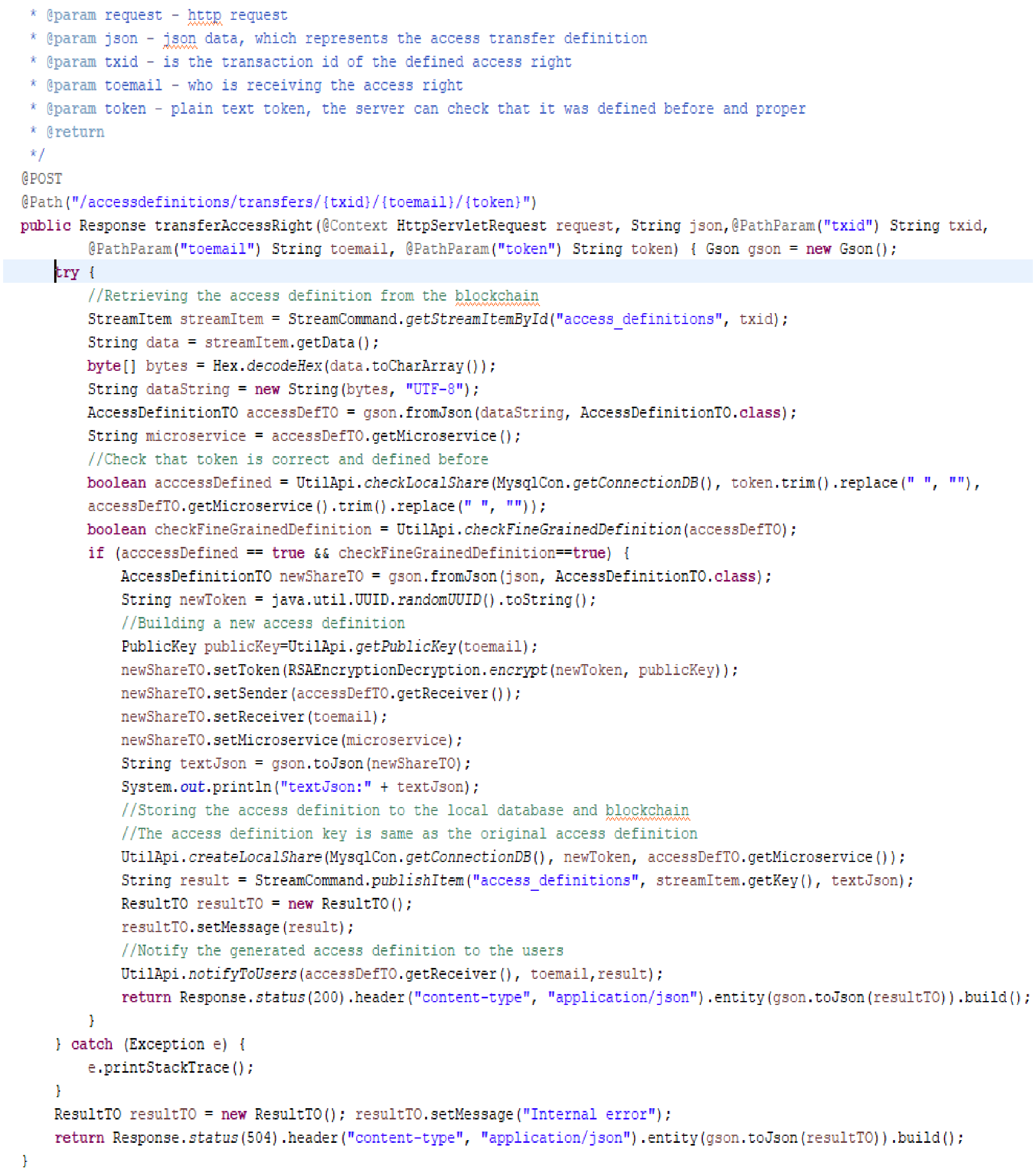
The following function grants the access right to the consumer using a JSON based definition and encrypts the token with the consumer's public key.



```java
/**
 *
 * @param request - http request
 * @param json - json data, which represents the access definition
 * @param fromemail - who is granting the access right
 * @param toemail - who is receiving the access right
 * @return
 */
@POST
@Path("/accessdefinitions/grants/{fromemail}/{toemail}")
public Response grantAccess(@Context HttpServletRequest request, String json,
        @PathParam("fromemail") String fromemail, @PathParam("toemail") String toemail) {
    try {
        //Retrieving public key from the blockchain using the sender's email
        StreamItem streamItem = StreamCommand.getStreamItemByKey("pkeys", toemail);
        String data = streamItem.getData();
        byte[] bytes = Hex.decodeHex(data.toCharArray());
        String dataString = new String(bytes, "UTF-8");
        String[] keys = dataString.split("-");
        Gson gson = new Gson();
        PublicKey publicKey = RSAEncryptionDecryption.getPublicKey(keys[0], keys[1]);
        //Creating unique id's for the access definition key and token to sign by the sender
        String token = java.util.UUID.randomUUID().toString();
        String accessDefinitionKey = java.util.UUID.randomUUID().toString();
        //Building the access definition based on the json content and other inputs
        AccessDefinitionTO accessDefTO = gson.fromJson(json, AccessDefinitionTO.class);
        accessDefTO.setToken(RSAEncryptionDecryption.encrypt(token, publicKey));
        accessDefTO.setSender(fromemail);
        accessDefTO.setReceiver(toemail);
        //Retrieving the sender's private key from the local database and signing
        //the token to show the request is authentic.
        PrivateKey privateKey = UtilApi.getPrivateKey(fromemail);
        String signature = RSAEncryptionDecryption.sign(fromemail, privateKey);
        accessDefTO.setSendersignature(signature);
        String textJson = gson.toJson(accessDefTO);
        System.out.println("textJson:" + textJson);
        //Storing the access rights to the blockchain and local database
        UtilApi.createLocalShare(MysqlCon.getConnectionDB(), token, accessDefTO.getMicroservice());
        String result = StreamCommand.publishItem("access_definitions", accessDefinitionKey, textJson);
        ResultTO resultTO = new ResultTO();
        resultTO.setMessage(result);
        return Response.status(200).header("content-type", "application/json").entity(gson.toJson(resultTO))
                .build();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return Response.status(504).header("content-type", "application/json").entity("{}").build();
}
```

Access transfer’s function:

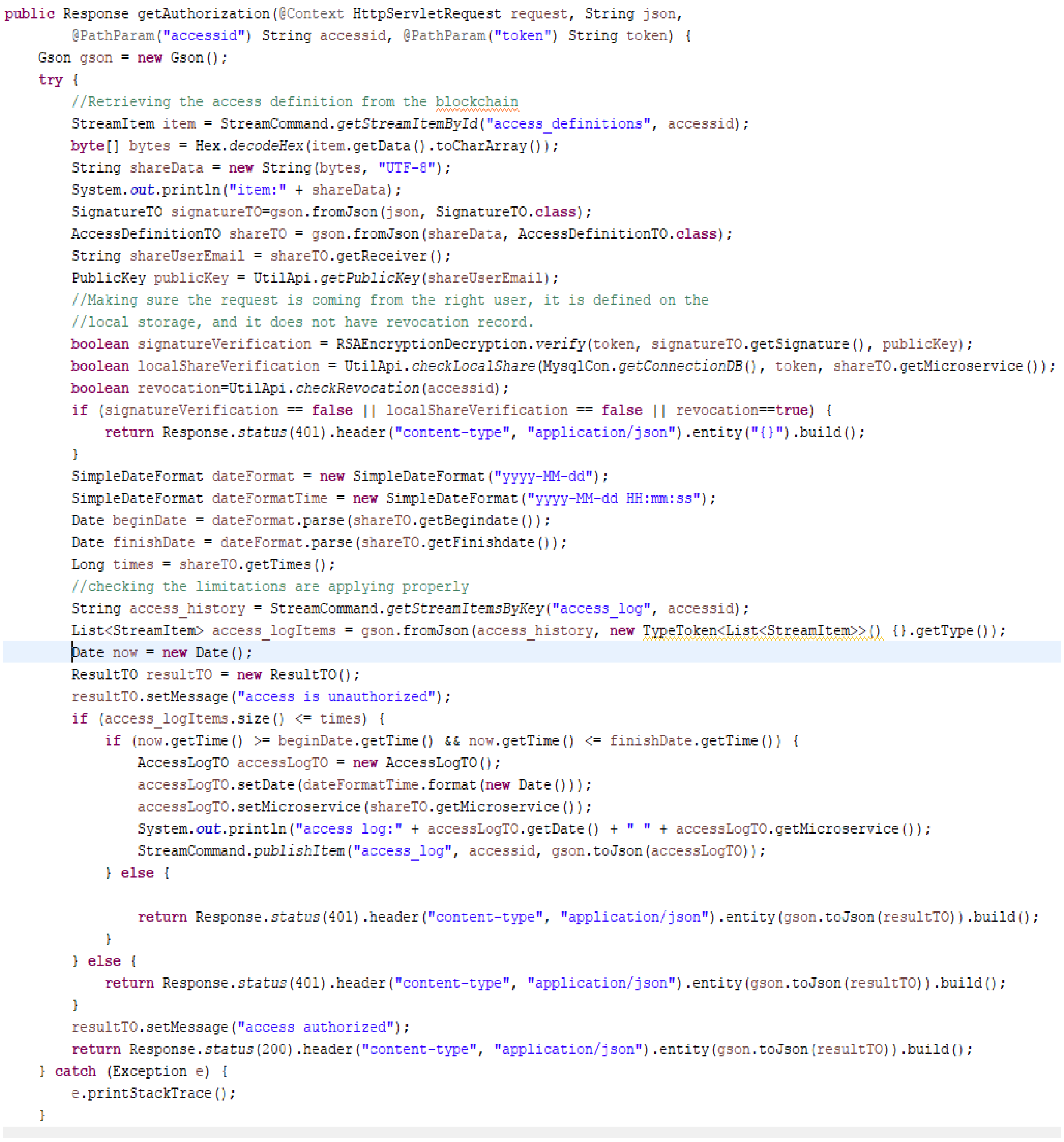
When the receiver of the access right transfers to their own access right, this function is called. Only the server, which issued the first access grant, can transfer the access rights because it is controlled by the organization that manages the microservice.



```
  * @param request - http request
  * @param json - json data, which represents the access transfer definition
  * @param txid - is the transaction id of the defined access right
  * @param toemail - who is receiving the access right
  * @param token - plain text token, the server can check that it was defined before and proper
  * @return
  */
 @POST
 @Path("/accessdefinitions/transfers/{txid}/{toemail}/{token}")
 public Response transferAccessRight(@Context HttpServletRequest request, String json,@PathParam("txid") String txid,
         @PathParam("toemail") String toemail, @PathParam("token") String token) { Gson gson = new Gson();
     try {
         //Retrieving the access definition from the blockchain
         StreamItem streamItem = StreamCommand.getStreamItemById("access_definitions", txid);
         String data = streamItem.getData();
         byte[] bytes = Hex.decodeHex(data.toCharArray());
         String dataString = new String(bytes, "UTF-8");
         AccessDefinitionTO accessDefTO = gson.fromJson(dataString, AccessDefinitionTO.class);
         String microservice = accessDefTO.getMicroservice();
         //Check that token is correct and defined before
         boolean acccessDefined = UtilApi.checkLocalShare(MysqlCon.getConnectionDB(), token.trim().replace(" ", ""),
         accessDefTO.getMicroservice().trim().replace(" ", ""));
         boolean checkFineGrainedDefinition = UtilApi.checkFineGrainedDefinition(accessDefTO);
         if (acccessDefined == true && checkFineGrainedDefinition==true) {
             AccessDefinitionTO newShareTO = gson.fromJson(json, AccessDefinitionTO.class);
             String newToken = java.util.UUID.randomUUID().toString();
             //Building a new access definition
             PublicKey publicKey=UtilApi.getPublicKey(toemail);
             newShareTO.setToken(RSAEncryptionDecryption.encrypt(newToken, publicKey));
             newShareTO.setSender(accessDefTO.getReceiver());
             newShareTO.setReceiver(toemail);
             newShareTO.setMicroservice(microservice);
             String textJson = gson.toJson(newShareTO);
             System.out.println("textJson:" + textJson);
             //Storing the access definition to the local database and blockchain
             //The access definition key is same as the original access definition
             UtilApi.createLocalShare(MysqlCon.getConnectionDB(), newToken, accessDefTO.getMicroservice());
             String result = StreamCommand.publishItem("access_definitions", streamItem.getKey(), textJson);
             ResultTO resultTO = new ResultTO();
             resultTO.setMessage(result);
             //Notify the generated access definition to the users
             UtilApi.notifyToUsers(accessDefTO.getReceiver(), toemail,result);
             return Response.status(200).header("content-type", "application/json").entity(gson.toJson(resultTO)).build();
         }
     } catch (Exception e) {
         e.printStackTrace();
     }
     ResultTO resultTO = new ResultTO(); resultTO.setMessage("Internal error");
     return Response.status(504).header("content-type", "application/json").entity(gson.toJson(resultTO)).build();
 }
```

Authorization function

The authorization function checks the local record, blockchain record, and the consumer's digital signature. If the verification is successful, it returns HTTP status code 200 with the message.



```java
public Response getAuthorization(@Context HttpServletRequest request, String json,
        @PathParam("accessid") String accessid, @PathParam("token") String token) {
    Gson gson = new Gson();
    try {
        //Retrieving the access definition from the blockchain
        StreamItem item = StreamCommand.getStreamItemById("access_definitions", accessid);
        byte[] bytes = Hex.decodeHex(item.getData().toCharArray());
        String shareData = new String(bytes, "UTF-8");
        System.out.println("item:" + shareData);
        SignatureTO signatureTO=gson.fromJson(json, SignatureTO.class);
        AccessDefinitionTO shareTO = gson.fromJson(shareData, AccessDefinitionTO.class);
        String shareUserEmail = shareTO.getReceiver();
        PublicKey publicKey = UtilApi.getPublicKey(shareUserEmail);
        //Making sure the request is coming from the right user, it is defined on the
        //local storage, and it does not have revocation record.
        boolean signatureVerification = RSAEncryptionDecryption.verify(token, signatureTO.getSignature(), publicKey);
        boolean localShareVerification = UtilApi.checkLocalShare(MysqlCon.getConnectionDB(), token, shareTO.getMicroservice());
        boolean revocation=UtilApi.checkRevocation(accessid);
        if (signatureVerification == false || localShareVerification == false || revocation==true) {
            return Response.status(401).header("content-type", "application/json").entity("{}").build();
        }
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        SimpleDateFormat dateFormatTime = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date beginDate = dateFormat.parse(shareTO.getBegindate());
        Date finishDate = dateFormat.parse(shareTO.getFinishdate());
        Long times = shareTO.getTimes();
        //checking the limitations are applying properly
        String access_history = StreamCommand.getStreamItemsByKey("access_log", accessid);
        List<StreamItem> access_logItems = gson.fromJson(access_history, new TypeToken<List<StreamItem>>() {}.getType());
        Date now = new Date();
        ResultTO resultTO = new ResultTO();
        resultTO.setMessage("access is unauthorized");
        if (access_logItems.size() <= times) {
            if (now.getTime() >= beginDate.getTime() && now.getTime() <= finishDate.getTime()) {
                AccessLogTO accessLogTO = new AccessLogTO();
                accessLogTO.setDate(dateFormatTime.format(new Date()));
                accessLogTO.setMicroservice(shareTO.getMicroservice());
                System.out.println("access log:" + accessLogTO.getDate() + " " + accessLogTO.getMicroservice());
                StreamCommand.publishItem("access_log", accessid, gson.toJson(accessLogTO));
            } else {

                return Response.status(401).header("content-type", "application/json").entity(gson.toJson(resultTO)).build();
            }
        } else {
            return Response.status(401).header("content-type", "application/json").entity(gson.toJson(resultTO)).build();
        }
        resultTO.setMessage("access authorized");
        return Response.status(200).header("content-type", "application/json").entity(gson.toJson(resultTO)).build();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

4.5. Answering the research questions

Blockchains have a potential to improve the cyber defense as the platform is secure, and can prevent fraudulent actions through consensus mechanisms, and can detect data tampering based on its underlying features of transparency, immutability, auditability, cryptography and operational resilience. Another key advantage of blockchain technology is disintermediation, which is the ability to have multiple parties sharing a single database and achieving a single view of the truth without having a central authority in charge of the database. Blockchain guarantees that everything is consistent and happened, and it is finalized without having a central authority. Blockchain's approach is genuinely new compared to how regular databases work today.

Disintermediation

The core value of the blockchain technology allows a database to be shared across boundaries without having a central trusted entity. Blockchain transactions contain their own proof of authorization and validity rather than some centralized application logic to enforce these constraints. This feature is important when independent organizations collaborate because none of them has a central authority. Disintermediation eliminates the problems of a central entity such as data manipulation and surveillance, and it guarantees transparency. Since the result of this research uses a blockchain as a shared database, disintermediation becomes a useful feature of the access control system.

Confidentiality

According to the National Institute of Standards and Technology (NIST), the property which is sensitive information is not accessible to unauthorized individuals, entities, or processes [62]. When organizations use blockchain technology, one of the common concerns is that protecting blockchain network access and data access is a fundamental requirement for these organizations.

Network access

In this proposed architecture, if cyber-attackers gain access to the blockchain network, they will be able to access the data. Although the blockchain technology was originally created without an access control mechanism, there are some blockchain distributions that have started working on the data confidentiality and access control issues. In public blockchains, there is no need to manage the network access of the blockchain network because its protocol allows everyone to access and participate in the network. In contrast, private blockchains require the participants to have appropriate access permissions to access the blockchain network. In a perfect world, private blockchain could connect by the local area network or VPN connections, and these networks are protected by several security layers such as firewalls, VLANs, intrusion detection and prevention systems, etc. In the real world, organizations could be on different networks and different geographical locations, and depending on network level protection is clearly insufficient. For this reason, the application layer of access control should protect its data.

Data access

When attackers gain access to the blockchain network and the data, it does not mean the attackers can read or retrieve the information. In the security community, one of the most important techniques to secure data transmission is end to end encryption, where only the actor who has permission to access the encrypted data by their own private key, can decrypt the data [63]. Using encryption keys with public key infrastructure(PKI) can provide higher-level security for organizations. Data encryption on the blockchain can provide a new degree of protection from a data confidentiality and data access control perspective. Cryptographic algorithms, which use public/private key generation, depend on integer factorization, which is hard to break with current computing resources.

Integrity

Integrity is defined as protecting against improper change or loss, and it ensures authenticity and acknowledgment of information according to NIST [62]. Data consistency and integrity are crucial in computer systems. Digital signature, hash comparison, or data encryption are the techniques, which information systems can use to assure data integrity. Blockchain technology has built in characteristics such as immutability and traceability to provide data integrity for organizations.

Immutability

Immutability is the inability to change, and it is a notion of an append-only data storage. Once information is stored on the blockchain, it cannot be modified.

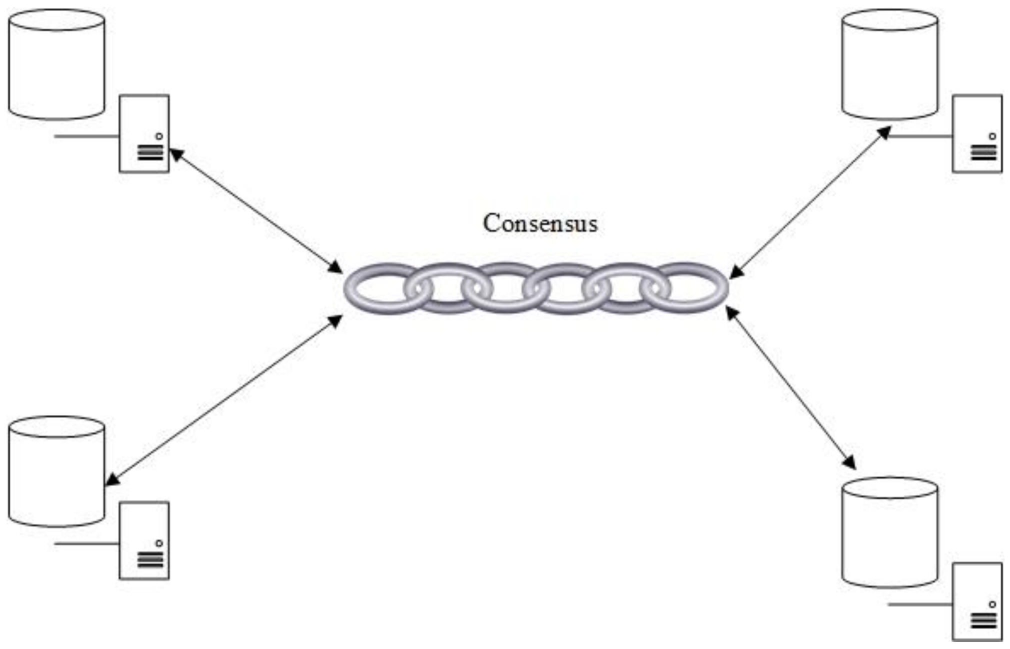


Figure 4-12: The consensus of blockchain technology

Figure 4.2 shows that the consensus mechanism of the blockchain is stronger than trusting one organization to keep records because the multiple participants or organizations cooperatively ensure that the data cannot change, and time stamps cannot change. This consensus mechanism differentiates blockchain technology from the regular databases. Also, access control systems that use a regular database, can be placed where administrators, security and risk teams can falsify or modify some information. Thus, the blockchain is one of the emerging technologies for the access control community.

Traceability

Every single transaction stored on a public or private blockchain is time-stamped and digitally signed. Therefore, organizations can trace back each transaction and corresponding identity. This feature enables a powerful audit record, providing a history of what happened, and when it happened. Adding a new transaction will change the global state of the ledger. Every new iteration of the system will be based on the previous stage of the system, so this process creates a fully traceable history log. This audit capability provides a new level of transparency and security for the organizations. For access control systems, it is important to know which resources are granted to which users, who accessed what kinds of resources, and when the events happened. From the cyber security perspective, blockchains provide organizations with an extra level assurance that the data is authentic and has not been tampered with.

Availability

NIST defines availability as making sure information is accessible in a timely manner and reliable to use. Cyber attacks attempt to impact information technology services availability. DDoS, which has been one of the most common cyberattacks, can disrupt the network and **can diminish the information systems' availability.** Peer-to-peer and decentralized characteristics of blockchain technology make the DDoS attack more challenging to disrupt information systems than traditional distributed application architectures such as client-server.

No single-point of failure

Blockchains have no single point of failure because the nodes are decentralized as peer-to-peer nodes, and the decentralized characteristic decreases the chances of DDoS attacks [64]. If a node is down and not responding, data is still accessible through other nodes with in the blockchain network as all nodes maintain a full copy of the ledger. This architectural design has chosen the private and permission-based blockchain, so it will be an additional security layer for the implementation.

Operational resilience

The combination of a number of blockchain networks within a network and peer-to-peer technology make blockchain based architecture operationally resilient. Also, each organization can create a redundant blockchain to increase the availability.

CHAPTER 5

5. PERFORMANCE EVALUATION

The experiments and evaluations of the decentralized access control system are described in this chapter. The experiments were run on the local network and cloud environment. The cloud environment is Amazon Web Services(AWS). The following experiments has been planned and executed.

Experiments	**Description**
Access control and blockchain nodes deployed on the local network	Test the performance of the access control deployed on the local network
Access control and blockchain nodes deployed on the AWS	Test the performance of the access control deployed on the AWS

Table 5-1: Description of experiments

The following figure shows the technological stacks, which are used in this implementation. These technological stacks will be the same in the local and cloud environment.

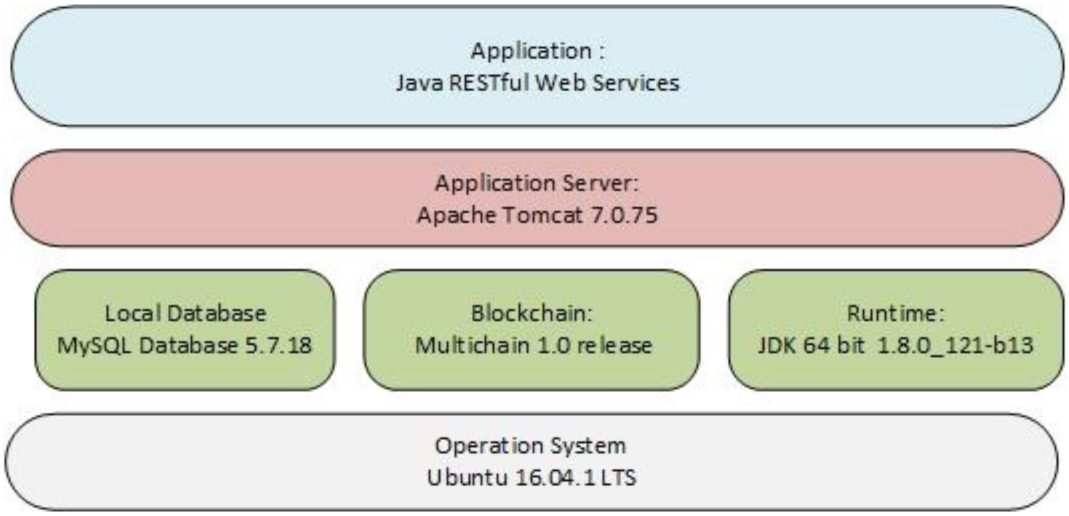


Figure 5-1: Technological stacks

The experimental elements are connected to the university’s local area network(LAN). The Windows machine called the Apache JMeter is a testing tool used to execute the performance tests and record the results.

A number of experiments were designed to assess system performance. The two-primary metrics investigated were average response time and throughput. Average response time is the time the server takes to respond to a request. This is an important metric for real world applications as significant delays in response time can negatively impact user experience and performance and cause HTTP requests to timeout. Throughput refers to the maximum rate at which requests can be processed by a system. Throughput is an important metric to assess real world viability and scalability of architecture. Average response time and throughput were assessed under a variety of conditions. The first experimentation variable was the number of clients trying to access the system. The system was tested with 1, 20, 40, 60, 120, 240, and 480 clients sending requests. These values provide insight into how user (client) load affects system performance. The second experimental variable investigated was a number of servers running the application and responding to requests. The system was tested with both 1 and 3 servers responding. This represents horizontal scaling of the application, which is one method that can be employed to increase system performance. The third variable in the experimental design was a delay between requests. Delays of 0ms, 250ms, and 500ms were considered under each condition. Real world requests do not necessarily occur simultaneously, so assessing system performance under difference delay scenarios can provide valuable insight into the system handles the load. Finally, each experimental setup was tested both on a local area network (LAN), and while deployed to an Amazon EC2 Instance (AWS). The LAN setup provides a good baseline of system performance under optimal conditions while testing the system deployed to cloud infrastructure represents a more likely real world use case. In total, these 84 different scenarios (2 server configurations x 7 client configurations x 3 delay configurations x 2 network settings) were assessed. The results of these experiments are presented and discussed below.

5.1. Experiment 1: Local area network

This first experiment evaluates the performance of the access control system, which is deployed on the local network. The experimental setup consists of a Windows machine, and three Linux servers, and Figure 5-2 shows the setup of the experiment. The Linux servers host the Multichain blockchain node as a distributed ledger, MySQL local database as an off-chain storage, and the access control application. The access control application is a RESTful web service application, and it is written in Java programming language.

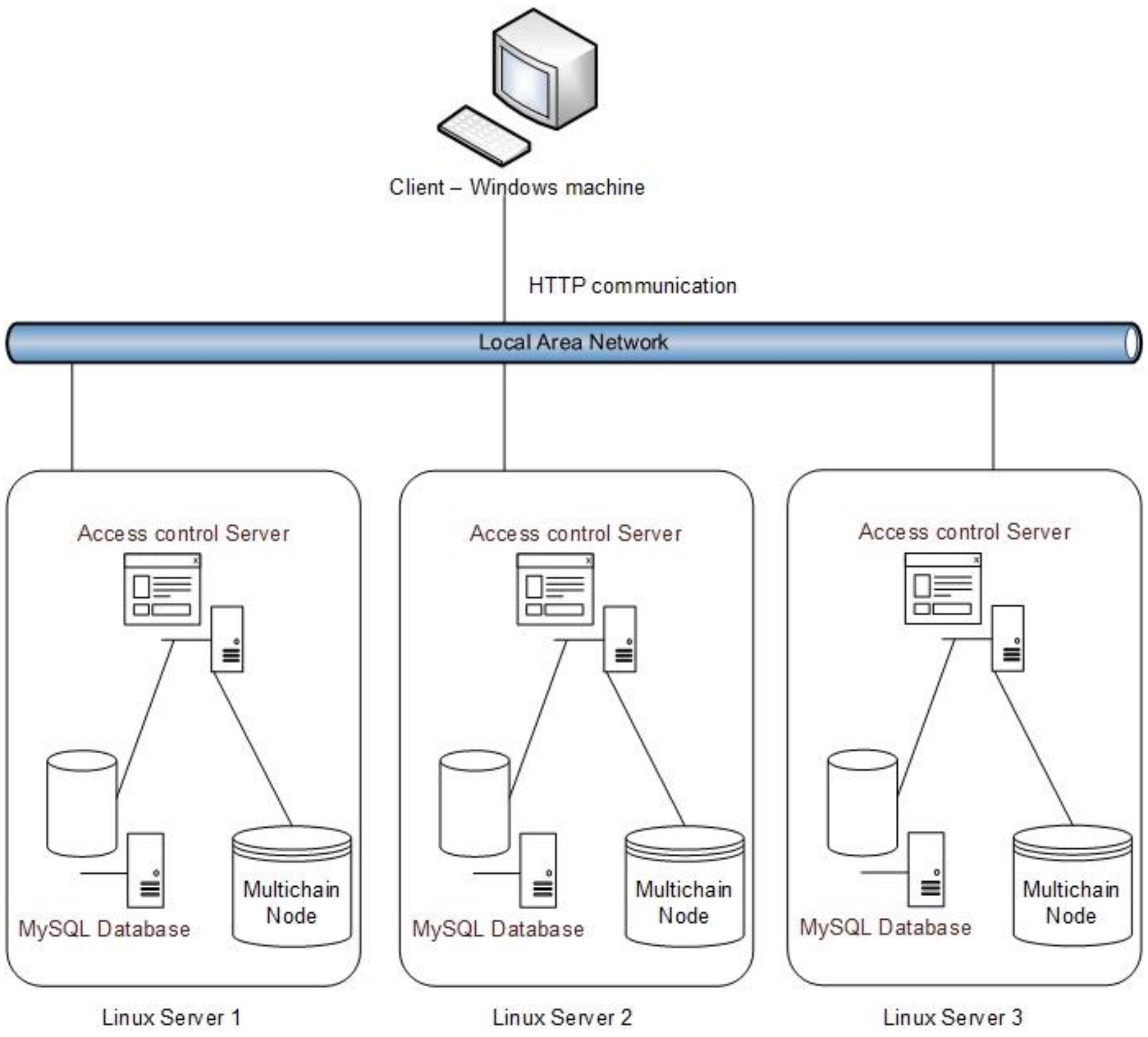


Figure 5-2: Experimental setup using LAN for the Experiment 1

Table 5-2 shows the hardware specification of configured servers and client.

Hardware	**Specifications**
Windows machine	**OS**: Windows 10 **Processor**: Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz (4 CPUs), ~3.1GHz

	Memory: 16 GB RAM
Linux Server 1	**OS**: Ubuntu 16.04 LTS **Processor:** Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz **Memory:** 4 GB RAM
Linux Server 2	**OS**: Ubuntu 16.04 LTS **Processor:** Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz **Memory:** 4 GB RAM
Linux Server 3	**OS**: Ubuntu 16.04 LTS **Processor:** Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz **Memory:** 4 GB RAM

Table 5-2: Hardware specifications for the Experiment 1

1. Average response time in the LAN without delay

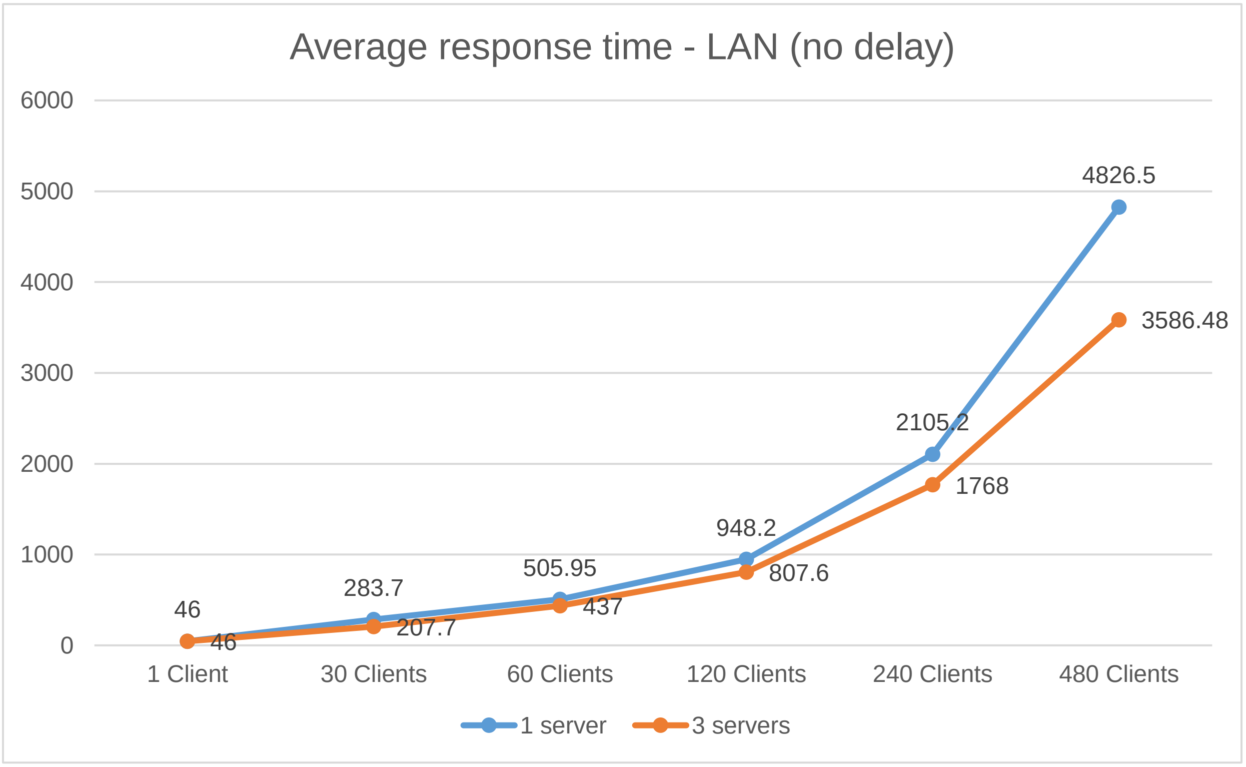


Figure 5-3: Average response time in the LAN without delay

The average response time of the system running over a LAN connection with no delay was 46 ms under a single client for both the 1 and 3 server configurations. Average response time with 480 clients increased to 4826.5ms for a single server and to 3586.5 ms for 3 servers. This clearly demonstrates that horizontal scaling by addition of more servers can have an impact on important metrics like average response time.

2. Throughput in the LAN without delay

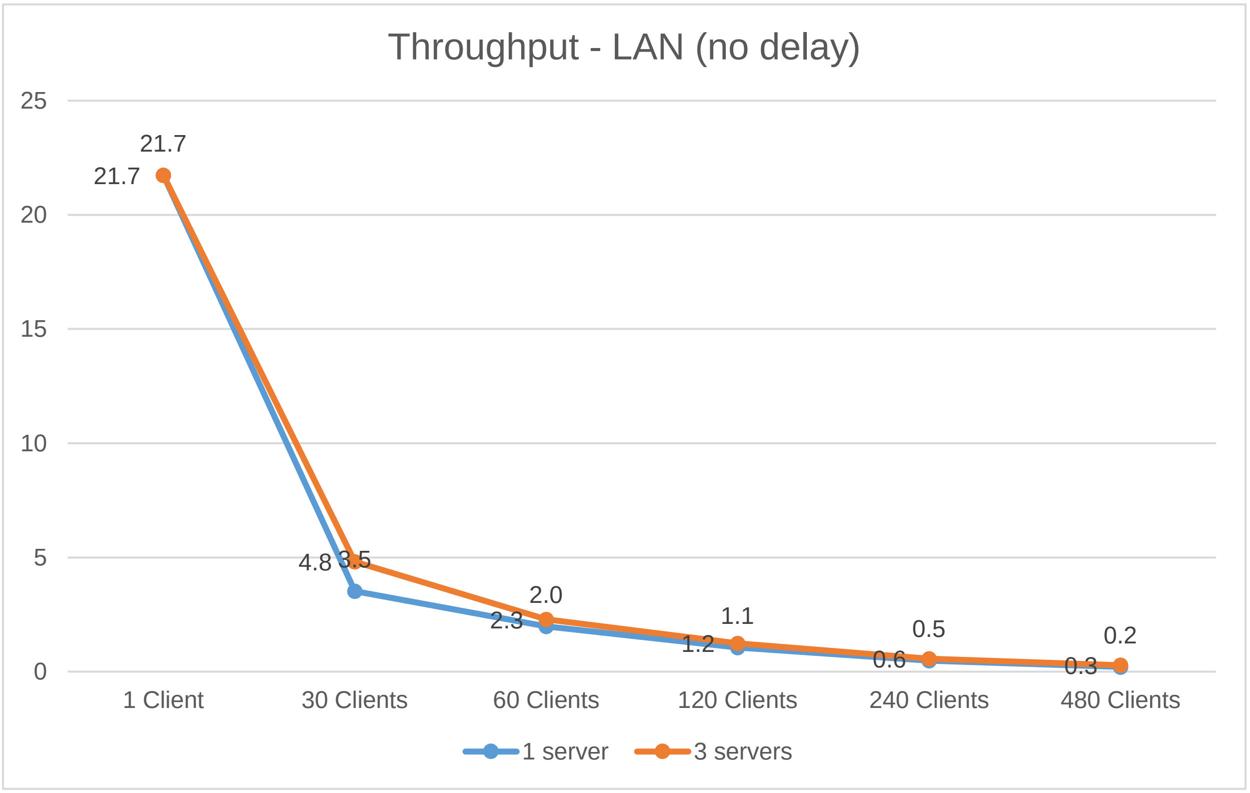


Figure 5-4: Throughput in the LAN without delay

The throughput when running the system over a LAN connection with no delay and a single server ranged from 21.7 requests per second with a single client load down to 0.2 requests per second with a 480-client load. With two additional servers responding to requests these numbers changed very little, still at 21.7 requests per second under a single client and down to 0.3 requests per second for 480 clients.

3. Average response time in the LAN with 250 ms delay

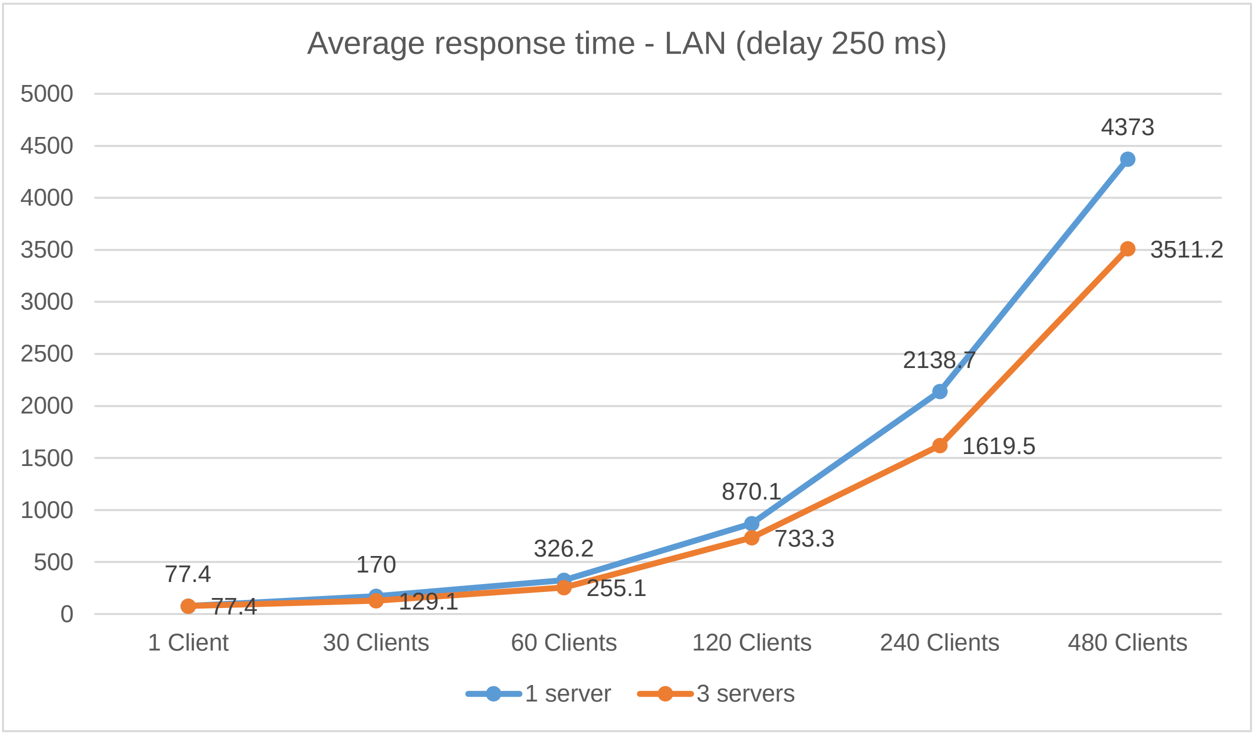


Figure 5-5: Average response time in the LAN with 250 ms delay

The average response time achieved by the system running over a LAN connection with a 250ms delay for a single client load and both a 1 and 3 server configuration was 77.4ms. Indeed, with a single client and single server, obtaining an average response time less than the delay between requests, it follows that additional servers will not provide a speed up. Average response time increased to 4373ms and 3511ms for a single and three server configurations, respectively. Comparing these results to the experimentation with no delay between request indicates only a slight increase in performance due to the 250ms delay.

4. Throughput in the LAN with 250 ms delay

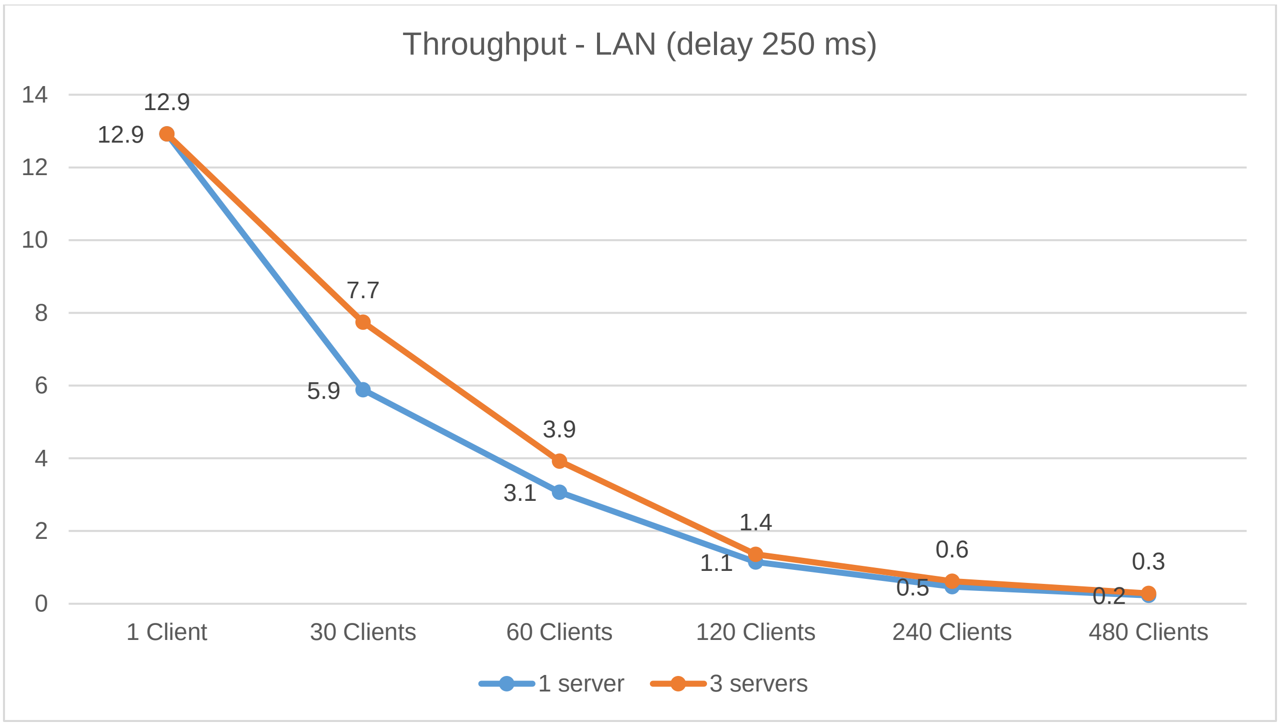


Figure 5-6: Throughput in the LAN with 250 ms delay

The throughput when running the system over a LAN connection with a 250ms delay between requests on both a single server and three server configurations with a single client was 12.9 requests per second. The throughput dropped to 0.2 and 0.3 requests per second under 480 clients load for a 3 and 1 server configuration, respectively. When comparing throughput by request delay, the data indicate that a 250ms delay does decrease overall system performance at lower levels of client load (12 requests per second vs. 21 requests per second), however the two scenarios are much more closely aligned when more than 200 clients are attempting to access the system.

5. Average response time in the LAN with 500 ms delay

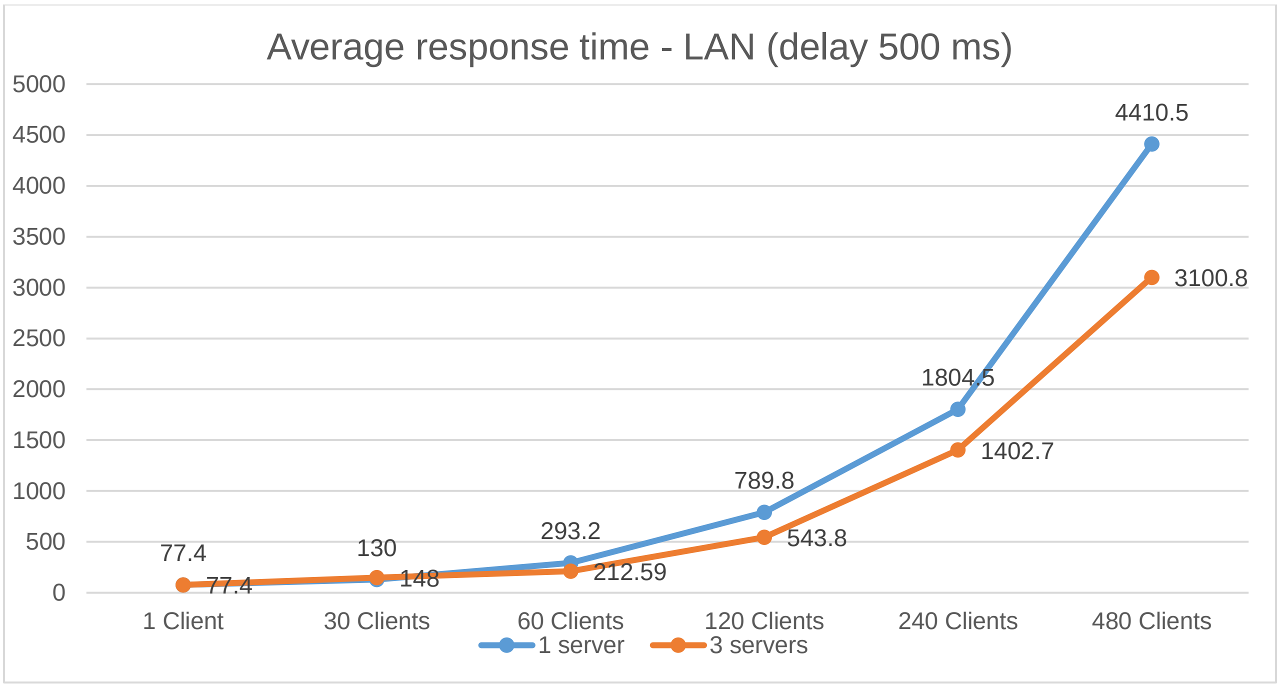


Figure 5-7: Average response time in the LAN with 500 ms delay

The average response time of the system over a LAN connection with a 500ms delay was again identical between a one and three server configuration at a single client load at 77.4ms. With increasing client load, the three servers' configuration again outperformed the single server, with a 3100ms average response time at 480 clients for the three servers' configuration, and 4410ms average response time for the single server. Again, comparing these results to the no delay or 250ms delay scenarios indicates that the increased delay between requests has a small but positive effect on system performance.

6. Throughput in the LAN with 500 ms delay

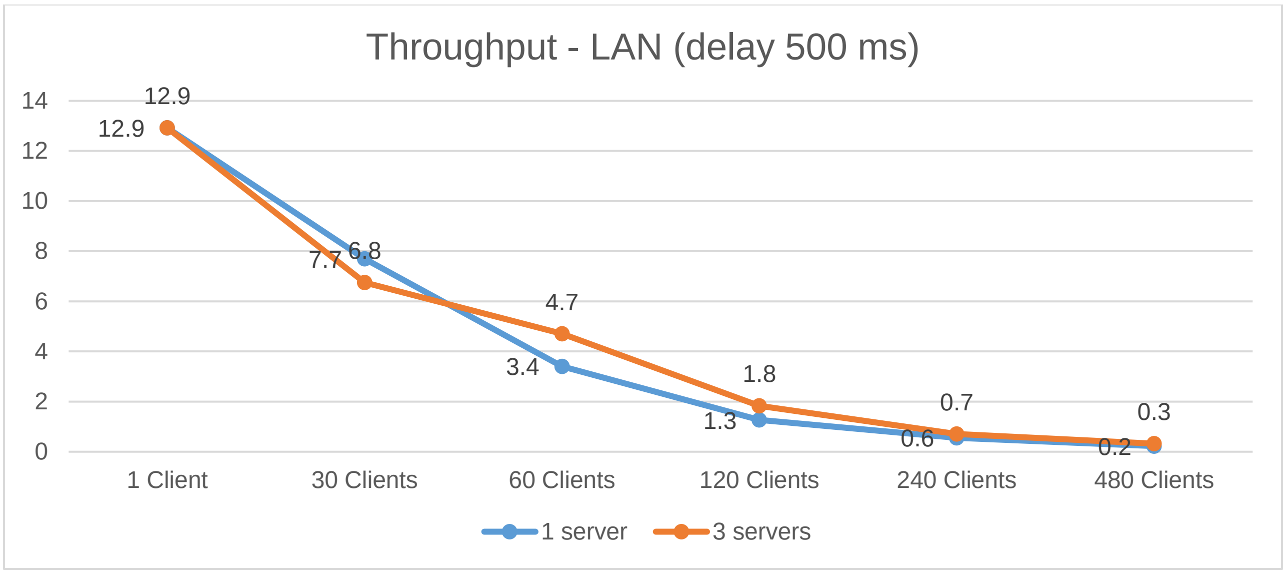


Figure 5-8: Throughput in the LAN with 500 ms delay

The throughput when running the system over a LAN connection with a 500ms delay between requests was 12.9 requests per second for both a single and three server configurations responding to a single client. Throughput decreased to 0.3 and 0.2 requests per second for a one and three server configuration, respectively. Interestingly, and unlike in the no delay and 250ms delay experiments, throughput under a 30 clients load was higher for a single client configuration at 7.7 requests per second than the 6.8 requests per second recorded for three servers. This small advantage may be a result of the overhead to route to three servers being larger than the three servers processing advantage, given that the 500ms delay allows the single server to catch up.

5.2. Experiment 2: Cloud environment – AWS

On AWS, when the servers are communicating with each other, Amazon makes sure that all internal traffic gets routed internally [65], so the blockchain cluster synchronizes their data internally. These experiments are performed on the university network to the Amazon EC2 network.

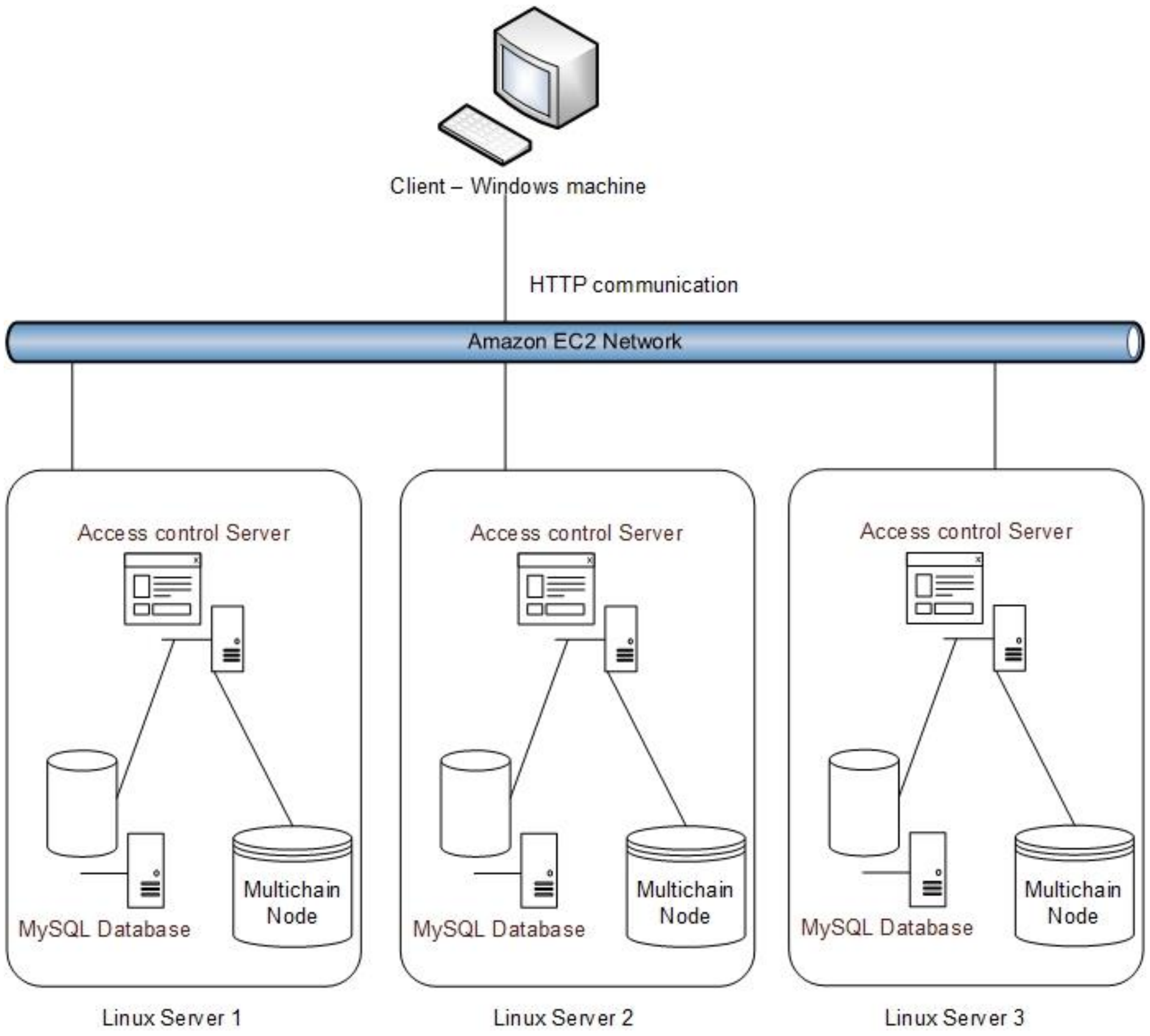


Figure 5-9: Experimental setup on the Amazon cloud services

The servers are hosted on AWS Elastic Computing (EC) as virtual instances. The three instances use the exact same configurations, which are described in *Table 5-3*.

EC2 Instances	**Specifications**
Instance 1	**Instance type:** t2.medium **Availability zone:** us-east-1b **OS**: Ubuntu 16.04 LTS **Processor:** Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz **Memory:** 4 GB RAM

Table 5-3: Instance specifications for the Experiment 2

1. Average response time in the AWS without delay

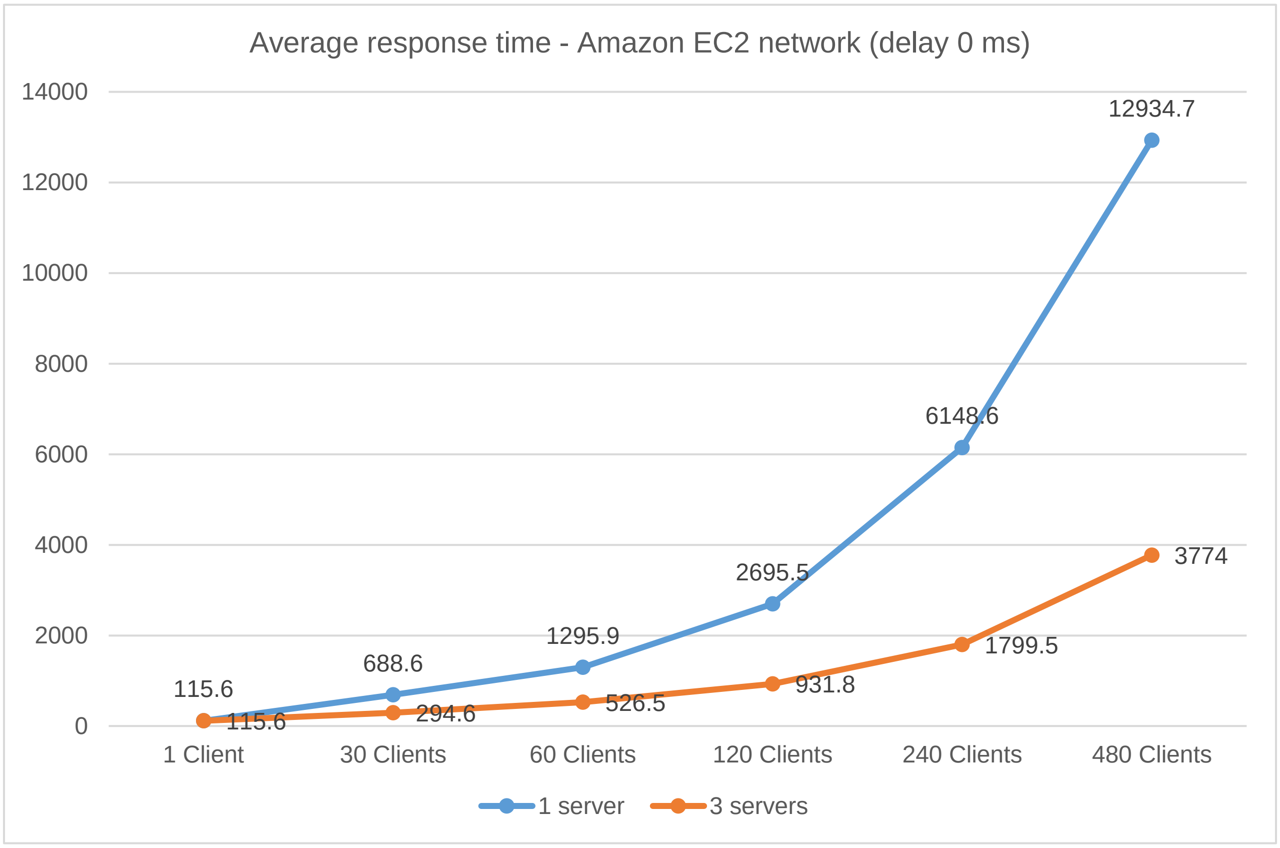


Figure 5-10: Average response time in the AWS without delay

Average response time when running the system over AWS was also longer than compared to a LAN connection. Average response time with no delay ranged from 115.6ms for a single client for both the one and three server configurations to up to 12,934ms for the single server configuration. The three-servers' configuration fared significantly better than the single, achieving an average response time of 3774ms, which is only ~200ms slower than the LAN connection under the same conditions. It appears that adding additional servers is indeed a viable way to horizontal scale the application's performance when running over cloud infrastructure.

2. Throughput in the AWS without delay

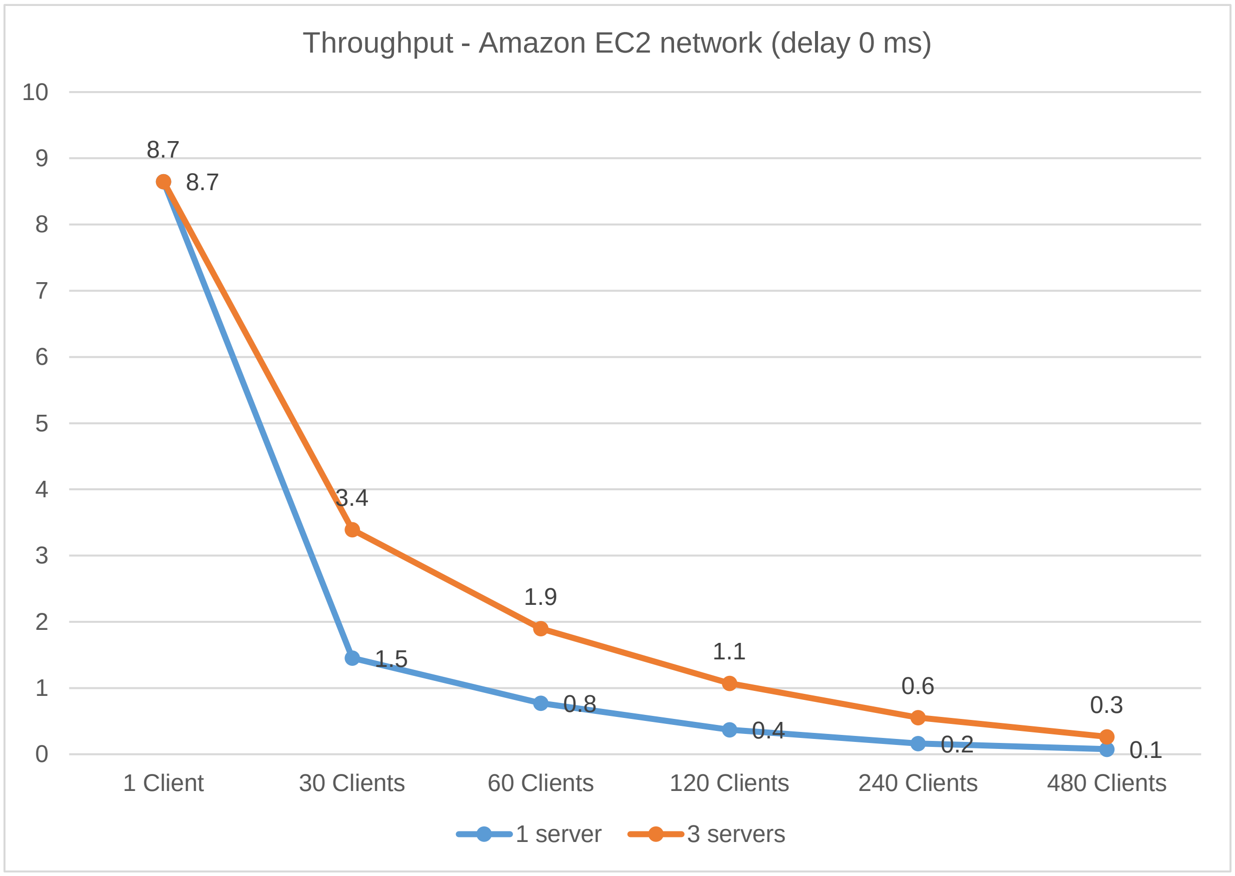


Figure 5-11: Throughput in the AWS without delay

Throughput while running the system on an Amazon EC2 instance with no delay was generally lower than when running locally. Throughput was highest with a single client at 8.7 requests per second for both the one and three server configurations, and lowest under 480 clients at 0.1 and 0.3 requests per second for the one and three server configurations, respectively.

3. Average response time in the AWS with 250 ms delay

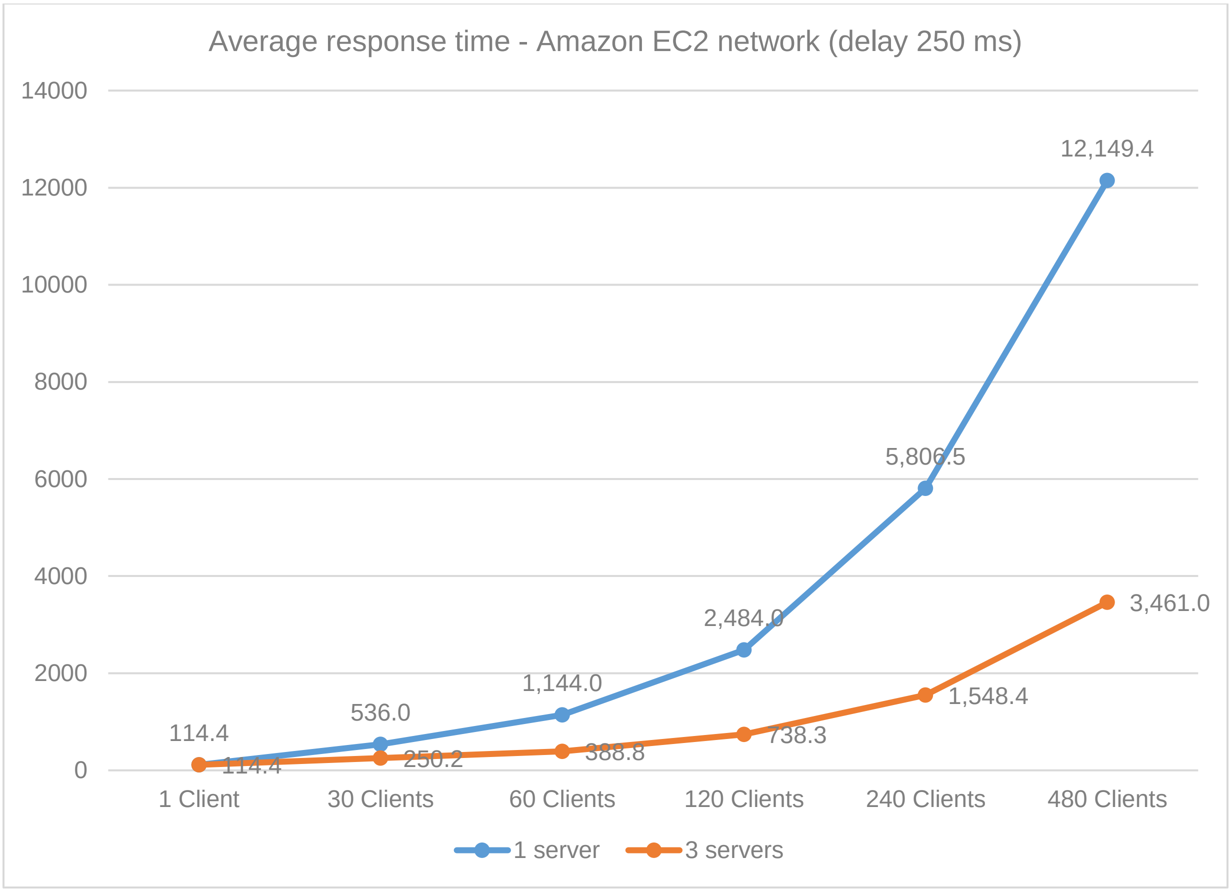


Figure 5-12: Average response time in the AWS with 250 ms delay

Average response time of the system with a 250ms delay was also very similar to the same conditions with no delay, ranging from 114ms for a single client and up to 12,149ms and 3461ms at 480 clients for a one and three server configuration, respectively. Although the difference is small the average response time was marginally better for both server configurations with the 250ms delay when compared against the 0ms delay.

4. Throughput in the AWS without delay

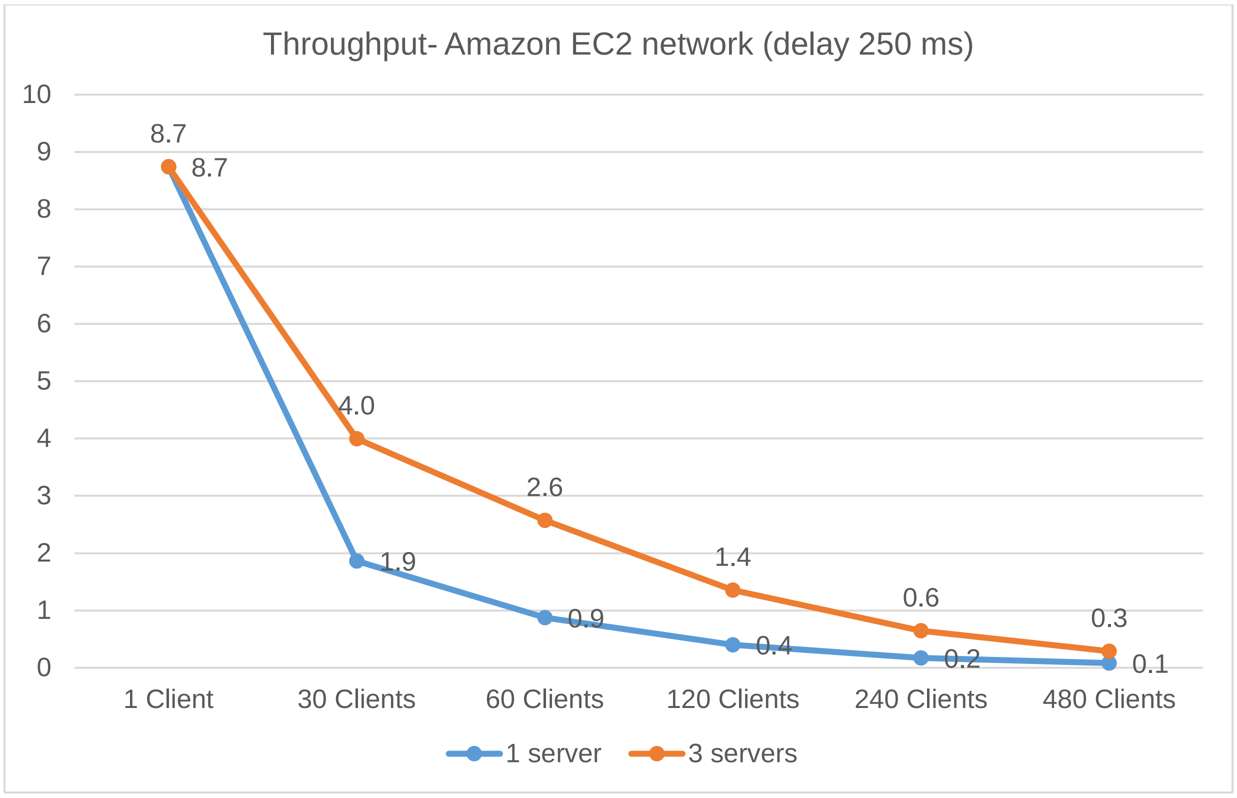


Figure 5-13: Throughput in the AWS without delay

Throughput when running the system over AWS with a 250ms delay was almost identical to the same situation with no delay. Throughput was highest with a single client at 8.7 requests per second for both the one and three server configurations, and lowest under 480 clients at 0.1 and 0.3 requests per second for the one and three server configurations, respectively. These data suggest that adding a delay between client requests has a negligible impact on system performance when it is running in the cloud.

5. Average response time in the AWS with 500 ms delay

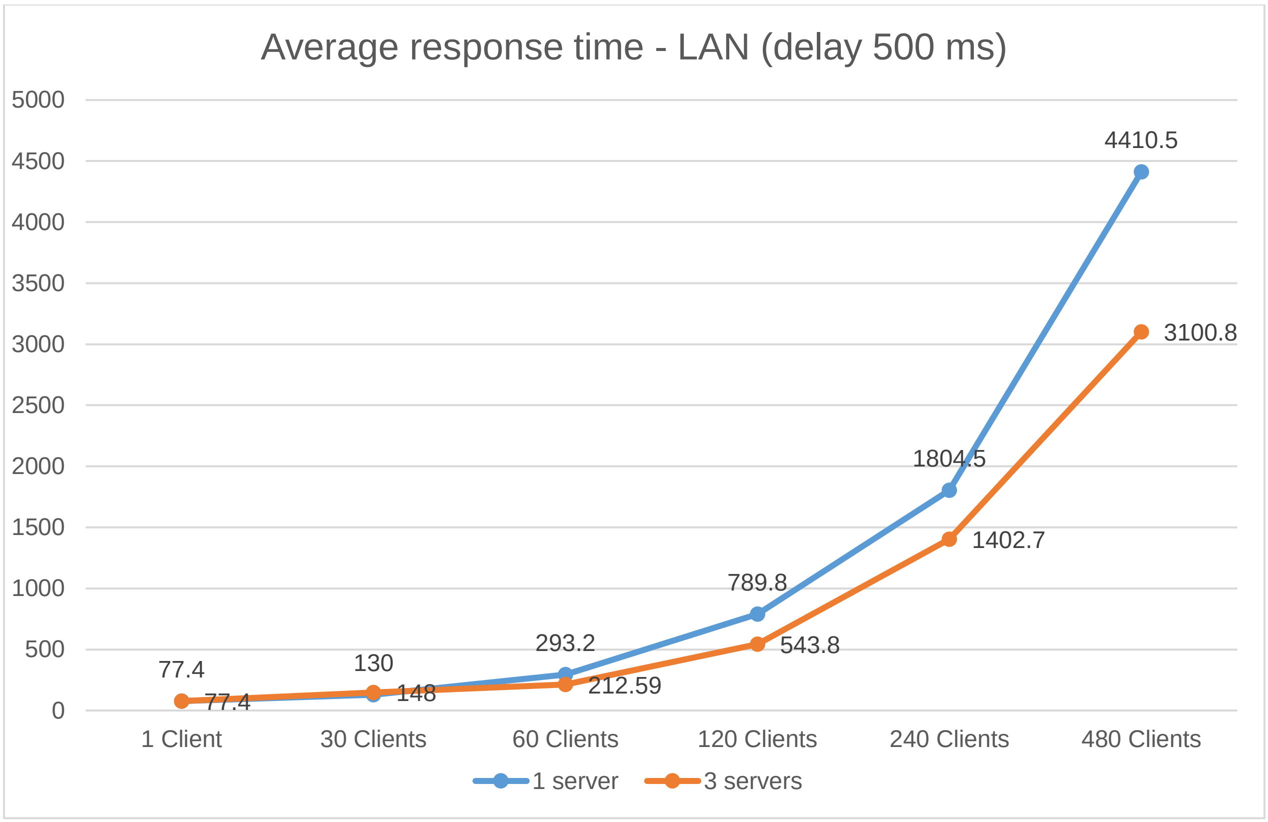


Figure 5-14: Average response time in the AWS with 500 ms delay

The average response time of the system with a 500ms delay is lowest under a single client at 128ms, and increases sharply for the single server configuration up to 12,139ms. The three servers' configuration is again much more performant, obtaining average response times of 3422ms. Clearly, the delay between requests has a negligible impact on system performance when served over AWS, however, as mentioned, additional servers have a large increase in performance at high client loads.

6. Throughput in the AWS with 500 ms delay

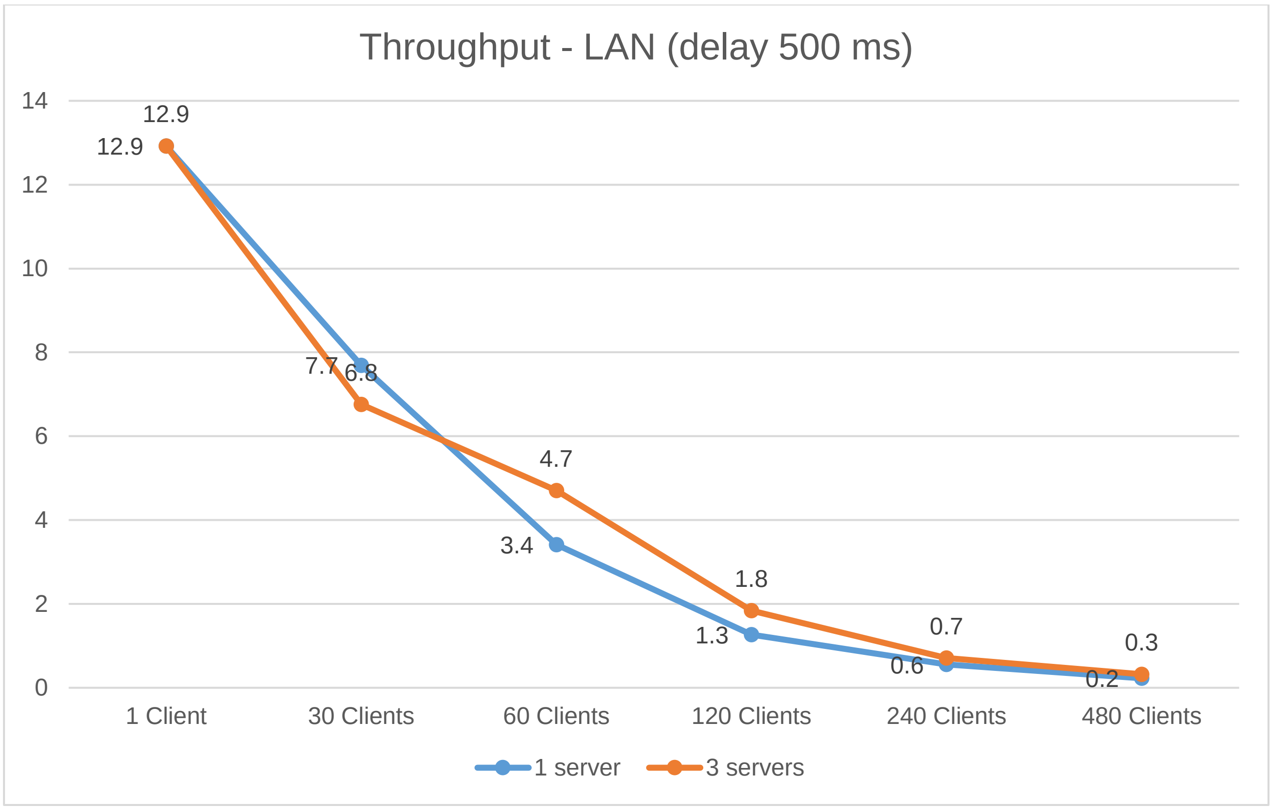


Figure 5-15: Throughput in the AWS with 500 ms delay

The throughput when running the system over AWS with a 500ms delay between requests was largest under a single client load, managing 7.8 requests per second for both a single and three server setups. Throughput drops off expectedly with increasing client load, minimizing at 0.1 and 0.3 requests per second for the one and three server configurations, respectively. Although these values are slightly higher than the AWS experiments at other delay lengths, the effect is more pronounced at very low client loads, and disappears when client load is maximized.

7. Throughput comparisons between AWS and LAN (3 servers)

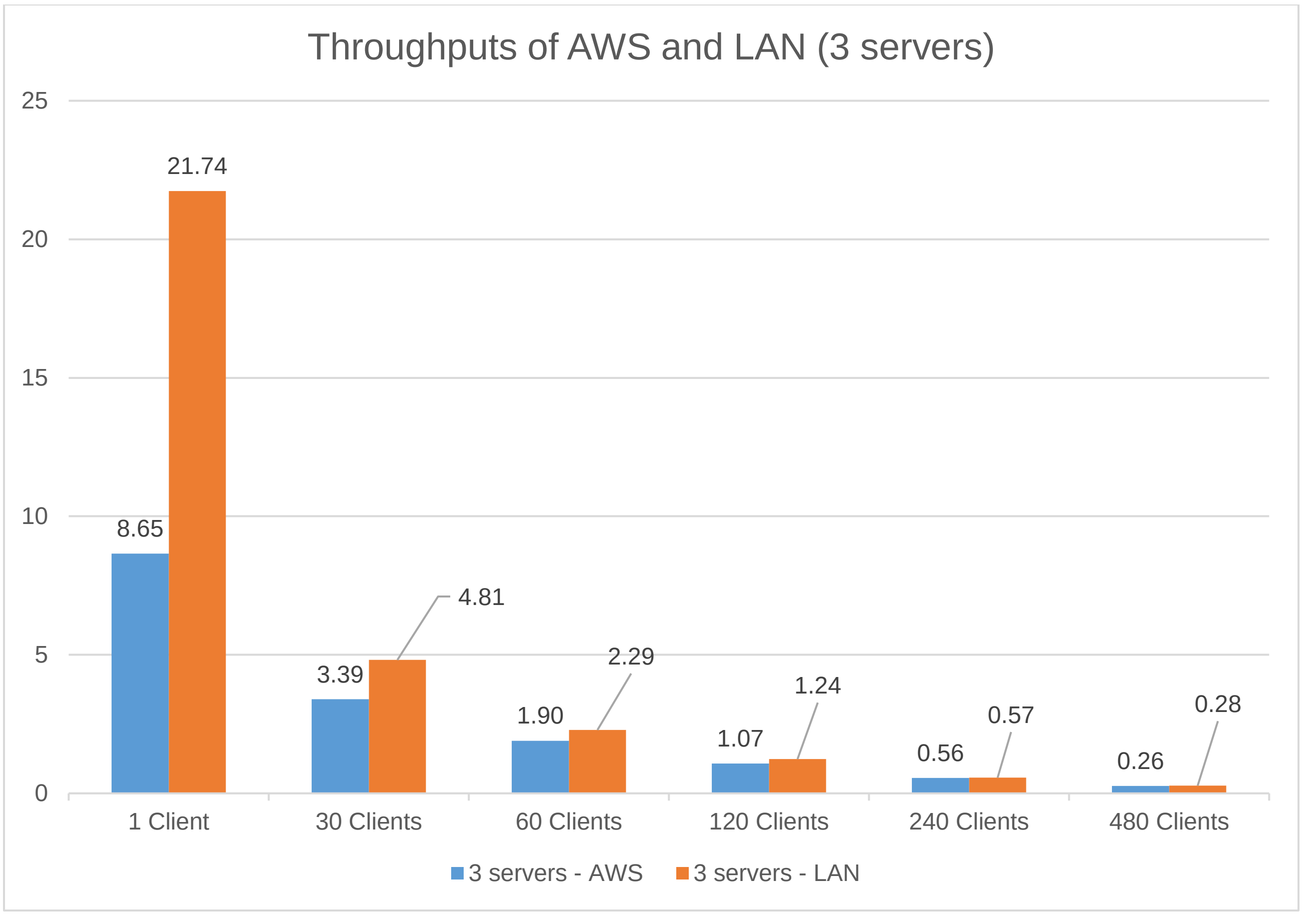


Figure 5-16: Throughput comparisons between AWS and LAN (3 servers)

8. Throughput comparisons between AWS and LAN (1 server case)

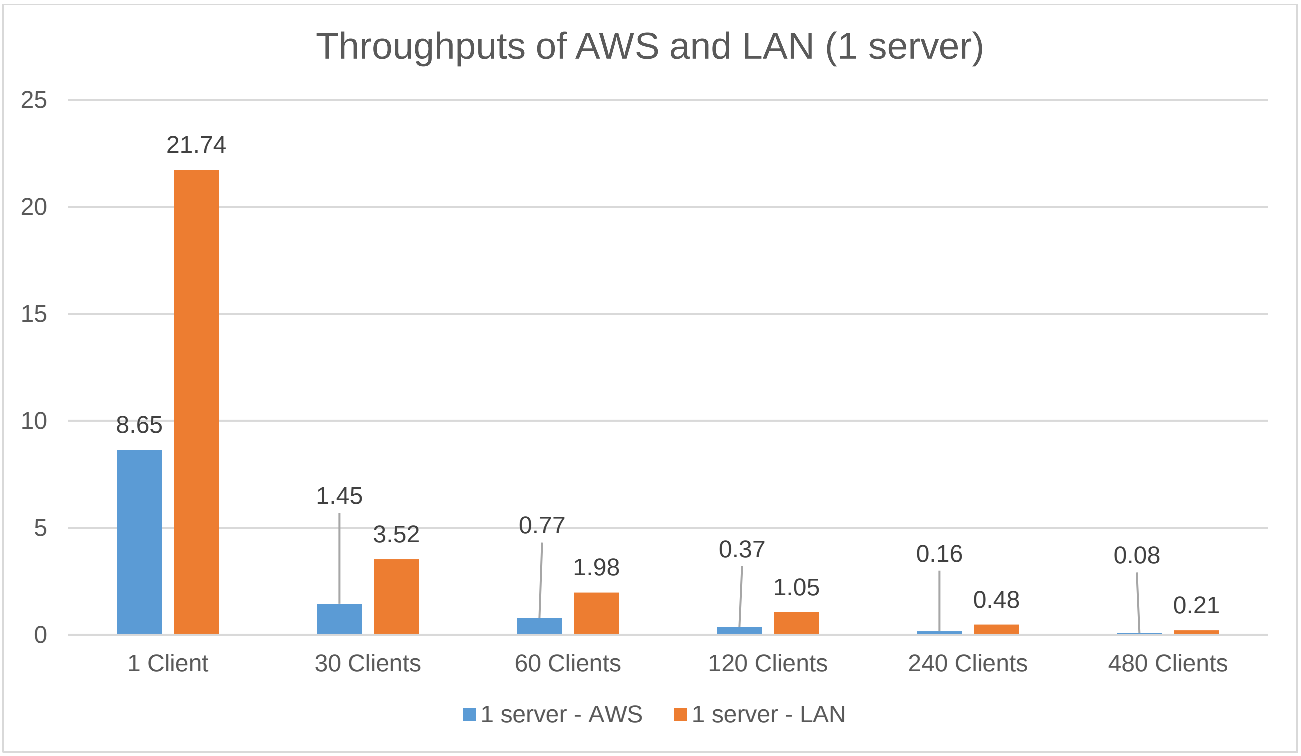


Figure 5-17: Throughput comparisons between AWS and LAN (1 server)

5.3. Summary

When comparing the throughput of all AWS scenarios against all LAN scenarios, clear patterns emerge for both the single-server and three-server test configurations. Generally, at low client loads, the LAN out performs AWS by a rough factor of 2. This performance difference is larger when only a single server is responding to requests. Indeed, at client loads greater than 100 and a single server configuration, the LAN connection completes up to 3 times the work as the AWS connection. When three servers are added, this discrepancy is significantly reduced. With three servers, at loads greater than 100 clients, the LAN and the AWS connection have only a marginal difference in performance. This again highlights the potential performance increases of horizontally scaling the application across many cloud servers.

In general, these experiments indicate that Multichain blockchain can handle requests of the access control system in the decentralized environment. Obviously, the LAN performance is better than the AWS environment, but the *Figure 5-16* and *Figure 5-17* shows that more horizontal scaling will result in better performance.

6. CONCLUSION AND FUTURE WORK

6.1. Summary

To conclude, blockchain technology is a new and innovative approach to manage and structure data for tracking, auditing and tracing. The blockchain data structure creates a well-defined solution for tracking the ownership of securities using private and public key infrastructure. The blockchain features are useful to build a decentralized access control system for the collaborative environment, where participants try to share their digital resources. Protecting and sharing digital resources are contradictory, and the collaboration is a real-world requirement for the digitized world. This work attempts to build a decentralized access control, which allows the sharing of microservices between independent organizations, and enhances the features of blockchain technology. This research evaluates the access control models and existing technologies to develop a decentralized access control system. With the proposed solution, organizations can define their microservices and grant, revoke, and transfer the access rights of defined microservices. This access control system differentiates from traditional access control systems in the following ways:

- All transactions on the blockchain are distributed to all participants. The blockchain is an append-only database, and it will not accept any data modification and falsification, even database or system administrators cannot change recorded data. In the most traditional databases, database administrators have absolute control of the database. However, with the blockchain technology, the data of decentralized access control is immutable.
- All organizations have equal rights over the access control. No single organization has absolute power to manage blockchain cluster. In the collaborative environment, equality is fundamental. Otherwise, the central authority or administrator can manipulate the data without the knowledge or permissions of the other organizations.
- In the blockchain cluster, all transactions are publicly accessible to all participants, thereby creating transparency.

The application is written in the Java programming language. The Multichain blockchain platform is used as a decentralized data storage. For the performance evaluation, the experiments were carried out on the local area network and the Amazon cloud infrastructure. The results of the experiments show that the performance in the LAN environment is better than that of the AWS environment. In addition, the results indicate that the decentralized access control system can scale horizontally.

6.2. Contributions

Previous research works in the literature have been focused on the centralized access control system. However, my work focuses on creating a decentralized access control system for the collaborative environment using blockchain technology. My contribution to the body of knowledge is the development of a decentralized access control system using blockchain technology, which provides a number of benefits that are not realizable in the centralized access control system. They are as follows:

1. The decentralized access control system provides transparency and equality among the participating organizations.
2. It enables the immutability of data to prevent manipulation, modification, falsification, and deletion.

In addition, I developed a generic data management RESTful API, which can store, retrieve, and query data on the Multichain blockchain platform. This API can be utilized by developers in different blockchain uses cases such as Blockchain IoT, Digital Identity, and decentralized storage and delivery.

6.3. Limitation and Future work

At the time of this research work, the blockchain technology is still in the development stage. When the blockchain technology and platforms become mature, this prototype and several concepts will require to be revisited. For the future work, the following features could be added and improved:

1. **Smart contracts:** The current implementation of decentralized access control system deploys the Java-based RESTful web application using the blockchain technology. The access control rules of this system are defined and declared in the source code, which makes it hard to change and maintain. Also, it is difficult for organizations to agree on these rules. However, smart contracts are a beneficial mechanism for the agreement between

participating organizations. Therefore, future works in this area can focus on the development of smart contract based solution to manage access control rules and policies.

2. **Standardized definitions of access control policies:** In this research work, I have used my own access control definitions. This is one of the main limitations of this research. However, the definitions of access policies and rules could use more standardized markup languages such as eXtensible Access Control Markup Language (XACML) [66] and Security Assertion Markup Language (SAML) [67].
3. **Service support:** Another limitation of my work is that the service definition, discovery, and invocation are supporting only RESTful web services. In future works, it would be useful to support other types of web service technologies and network protocols.
4. **Evaluation of the blockchain platforms:** Apart from the Multichain blockchain platform, which is utilized in this research, future works could evaluate other blockchain platforms such as Hyperledger, Etherium, BigChainDB, and Eris.

REFERENCES

[1] S. J. Shackelford, *Managing cyber attacks in international law, business, and relations: In search of cyber peace*. Cambridge University Press, 2014.

[2] T. Reeve, “CEO sacked after aircraft company grounded by whaling attack,” 2016. [Online]. Available: https://www.scmagazineuk.com/ceo-sacked-after-aircraft-company-grounded-by-whaling-attack/article/530984/. [Accessed: 31-Jul-2017].

[3] L. J. Trautman and P. C. Ormerod, “Corporate Directorss and Officerss Cybersecurity Standard of Care: The Yahoo Data Breach,” *SSRN Electron. J.*, Feb. 2016.

[4] Pagliery Jose, “Hackers selling 117 million LinkedIn passwords - May. 19, 2016,” 2016. [Online]. Available: http://money.cnn.com/2016/05/19/technology/linkedin-hack/. [Accessed: 31-Jul-2017].

[5] M. Balasubramanian, A. Bhatnagar, N. Chaturvedi, A. D. Chowdhury, and A. Ganesh, “A framework for decentralized access control,” *Proc. 2nd ACM Symp. Information, Comput. Commun. Secur. (ASIACCS ’07)*, pp. 93–104, 2007.

[6] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system.” 2008.

[7] X. Ao and N. H. Minsky, “Flexible regulation of distributed coalitions,” *Lect. notes Comput. Sci.*, vol. 2808, pp. 39–60, 2003.

[8] V. C. Hu, D. F. Ferraiolo, and D. R. Kuhn, “Assessment of Access Control Systems,” 2006.

[9] A. A. El Kalam and Y. Deswarte, “Multi-OrBAC: A new access control model for distributed, heterogeneous and collaborative systems,” in *8th IEEE International Symposium on Systems and Information Security*, 2006, p. 1.

[10] A. Abou, E. Kalam, Y. Deswarte, A. Bana, and M. Kaâniche, “Access Control for Collaborative Systems: A Web Services Based Approach,” pp. 1064–1071, 2007.

[11] H. Gomi, M. Hatakeyama, S. Hosono, and S. Fujita, “A delegation framework for federated identity management,” in *Proceedings of the 2005 workshop on Digital identity management*, 2005, pp. 94–103.

[12] D. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su, and T. A. Nguyen, “PERMIS: a modular authorization infrastructure,” *Concurr. Comput. Pract. Exp.*, vol. 20, no. 11, pp. 1341–1357, 2008.

[13] B. Crispo, S. Sivasubramanian, P. Mazzoleni, and E. Bertino, “P-hera: Scalable fine-

grained access control for p2p infrastructures," in *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, 2005, vol. 1, pp. 585–591.

[14] P. Samarati, S. De Capitani, and D. Vimercati, "Access Control: Policies, Models, and Mechanisms," 2000.

[15] R. S. Sandhu and P. Samarati, "Access control: principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, 1994.

[16] M. Chapple, B. Ballad, T. Ballad, and E. Banks, *Access control, authentication, and public key infrastructure*. Jones and Bartlett Publishers, Inc., 2013.

[17] U. S. DoD, "Department of defense trusted computer system evaluation criteria (orange book)," 1985.

[18] D. E. Bell and L. J. La Padula, "Secure computer system: Unified exposition and multics interpretation," 1976.

[19] M. Clarkson, "Access Control." [Online]. Available: http://www.cs.cornell.edu/courses/cs5430/2011sp/NL.accessControl.html. [Accessed: 31-Jul-2017].

[20] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *ACM workshop on Role-based access control*, 2000, vol. 2000, pp. 1–11.

[21] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, and R. Sandhu, "Role-Based Access Control Models," *IEEE Comput.*, vol. 29, no. 2, pp. 38–47, 1996.

[22] R. W. Baldwin, "Naming and grouping privileges to simplify security management in large databases," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, 1990, pp. 116–132.

[23] J. S. Park, R. Sandhu, and G.-J. Ahn, "Role-based access control on the web," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 1, pp. 37–71, Feb. 2001.

[24] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[25] L. Chen and J. Crampton, "Inter - domain Role Mapping and Least Privilege," 2007.

[26] "OrBAC: Organization Based Access Control | The official OrBAC model website." [Online]. Available: http://orbac.org/. [Accessed: 31-Jul-2017].

[27] "Attribute Based Access Control (ABAC) Overview." [Online]. Available: http://csrc.nist.gov/projects/abac/index.html. [Accessed: 31-Jul-2017].

[28] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, A. Schnitzer Booz, A. Hamilton, and S. Cybersecurity, "Draft Special Publication 800-162, Guide to Attribute Based Access Control (ABAC) Definition and Considerations," 2013.

[29] L. Sun and H. Wang, "A Purpose Based Usage Access Control Model," 2010.

[30] C. A. Ardagna, S. De Capitani Di Vimercati, G. Neven, S. Paraboschi, F.-S. Preiss, P. Samarati, and M. Verdicchio, "Enabling Privacy-Preserving Credential-Based Access Control with XACML and SAML," 2010.

[31] W. Gordon, "Understanding OAuth: What Happens When You Log Into a Site with Google, Twitter, or Facebook." [Online]. Available: http://lifehacker.com/5918086/understanding-oauth-what-happens-when-you-log-into-a-site-with-google-twitter-or-facebook. [Accessed: 31-Jul-2017].

[32] D. Hardt, "The OAuth 2.0 authorization framework," 2012.

[33] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[34] J. A. Bergstra and K. de Leeuw, "Questions related to Bitcoin and other Informational Money," *arXiv Prepr. arXiv1305.5956*, 2013.

[35] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, "APPLIED CRYPTOGRAPHY," 1996.

[36] A. Badev and M. Chen, "Bitcoin: Technical Background and Data Analysis," 2014.

[37] R. Schollmeier, "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications," in *Proceedings First International Conference on Peer-to-Peer Computing*, 2001, pp. 101–102.

[38] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, "MuR-DPA: Top-down Levelled Multi-replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud," *IEEE Trans. Comput.*, 2015.

[39] "Mastering Bitcoin." [Online]. Available: http://chimera.labs.oreilly.com/books/1234000001802/ch07.html#merkle_trees.

[Accessed: 31-Jul-2017].

[40] “Proof of work - Bitcoin Wiki.” [Online]. Available: https://en.bitcoin.it/wiki/Proof_of_work. [Accessed: 31-Jul-2017].

[41] J. R. Douceur, “The Sybil Attack,” 2002.

[42] F. Tschorsch and B. Scheuermann, “Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies,” 2015.

[43] M. Crosby, “BlockChain Technology: Beyond Bitcoin,” *Appl. Innov. Rev. Issue*, no. 2, 2016.

[44] J. P. BARLOW’S, “DECLARATION OFINDEPENDENCE FOR CYBERSPACE.” 1996.

[45] V. Goel, “Facebook tinkers with users’ emotions in news feed experiment, stirring outcry,” *New York Times*, 2014.

[46] J. Ball, “NSA’s Prism surveillance program: how it works and what it can do,” *Guard.*, vol. 8, 2013.

[47] B. Hardekopf, “The big data breaches of 2014,” *Forbes, January*, vol. 13, 2015.

[48] G. Zyskind, O. Nathan, and A. Pentland, “Enigma: Decentralized Computation Platform with Guaranteed Privacy,” *arXiv1506.03471 [cs]*, pp. 1–14, 2015.

[49] Z. Beaven, D. Neilson, R. Osborne, and P. Pacifico, “Blockchain for Creative Industries Music On The Blockchain,” 2016.

[50] “Public versus Private Blockchains Part 1: Permissioned Blockchains,” 2015.

[51] “All Currencies | CryptoCurrency Market Capitalizations.” [Online]. Available: https://coinmarketcap.com/currencies/views/all/. [Accessed: 31-Jul-2017].

[52] “On Stake - Ethereum Blog.” [Online]. Available: https://blog.ethereum.org/2014/07/05/stake/. [Accessed: 31-Jul-2017].

[53] “Survey on Blockchain Technologies and Related Services FY2015 Report,” 2016.

[54] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[55] P. Selonen, P. Belimpasakis, and Y. You, “Experiences in Building a RESTful Mixed Reality Web Service Platform,” *LNCS*, vol. 6189, pp. 400–414, 2010.

[56] J. C. Mogul, J. Gettys, T. Berners-Lee, and H. Frystyk, “Hypertext Transfer Protocol-- HTTP/1.1,” 1997.

[57] H. Hamad, M. Saad, and R. Abed, “Performance Evaluation of RESTful Web Services for Mobile Devices,” *Int. Arab J. e-Technology*, vol. 1, no. 3, 2010.

[58] P. Kookarinrat and Y. Temtanapat, “Design and implementation of a decentralized message bus for microservices,” in *Computer Science and Software Engineering (JCSSE), 2016 13th International Joint Conference on*, 2016, pp. 1–6.

[59] “Blockchain & Cyber Security.” [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-telecommunications/Blockchain-and-Cyber.pdf. [Accessed: 31-Jul-2017].

[60] G. Greenspan, “MultiChain Private Blockchain — White Paper.”

[61] C. Pautasso, O. Zimmermann, and F. Leymann, “Restful web services vs. big’web services: making the right architectural decision,” in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 805–814.

[62] R. Kissel, “NISTIR 7298 Revision 2 Glossary of Key Information Security Terms,” 2013.

[63] “Hacker Lexicon: What Is End-to-End Encryption? | WIRED.” [Online]. Available: https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption/. [Accessed: 31-Jul-2017].

[64] “Transaction spam attack: Next Steps - Ethereum Blog.” [Online]. Available: https://blog.ethereum.org/2016/09/22/transaction-spam-attack-next-steps/. [Accessed: 31-Jul-2017].

[65] “Using Elastic IP to Identify Internal Instances on Amazon EC2 - Alestic.com.” [Online]. Available: https://alestic.com/2009/06/ec2-elastic-ip-internal/. [Accessed: 09-Aug-2017].

[66] “eXtensible Access Control Markup Language (XACML) Version 3.0.” [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. [Accessed: 09-Aug-2017].

[67] “FrontPage - SAML Wiki.” [Online]. Available: https://wiki.oasis-open.org/security/FrontPage#SAML_V2.0_Standard. [Accessed: 09-Aug-2017].